

Implementation of multilayer Perceptron

Course :Connectionist Networks

● Members:

Ahmadou Tidiane

Djibril

Faly Beydary

Alassane

Ndongo

Sow

Ndiaye

Kane

● Academic Info

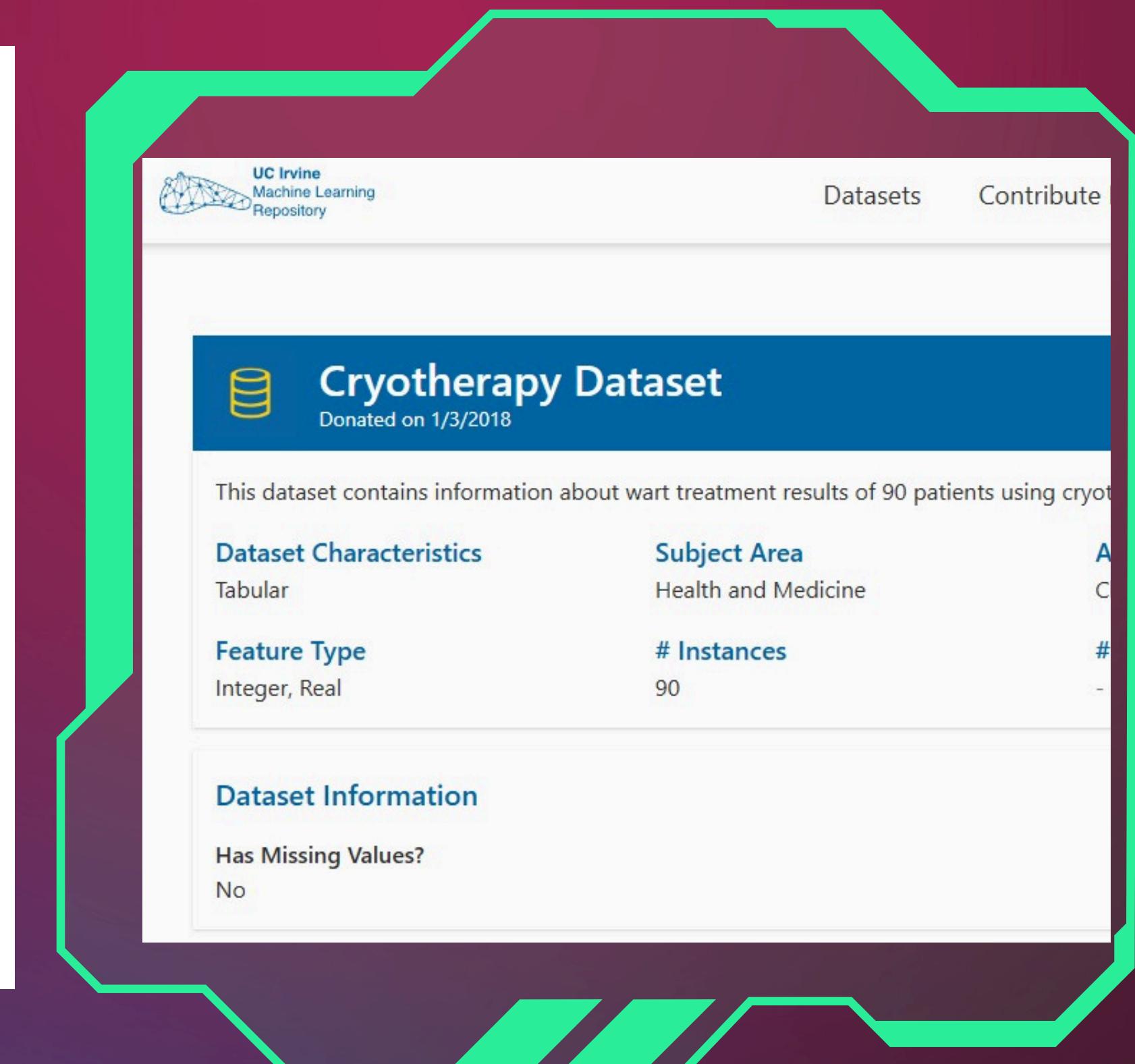
Master 1 GDIL/R2SD
2024 - 2025

next slide ➤

Dataset

[Home](#)[About Us](#)[Contact](#)

	A	B	C	D	E	F	G
1	sex	age	Time	Number_of_Warts	Type	Area	Result_of_Treatment
2	1	35	12	5	1	100	0
3	1	29	7	5	1	96	1
4	1	50	8	1	3	132	0
5	1	32	11,75	7	3	750	0
6	1	67	9,25	1	1	42	0
7	1	41	8	2	2	20	1
8	1	36	11	2	1	8	0
9	1	59	3,5	3	3	20	0
10	1	20	4,5	12	1	6	1
11	2	34	11,25	3	3	150	0
12	2	21	10,75	5	1	35	0
13	2	15	6	2	1	30	1
14	2	15	2	3	1	4	1
15	2	15	3,75	2	3	70	1
16	2	17	11	2	1	10	0
17	2	17	5,25	3	1	63	1
18	2	23	11,75	12	3	72	0
19	2	27	8,75	2	1	6	0
20	2	15	4,25	1	1	6	1
21	2	18	5,75	1	1	80	1
22	1	22	5,5	2	1	70	1



The image shows a screenshot of the UC Irvine Machine Learning Repository website. The specific dataset page for the "Cryotherapy Dataset" is displayed. The page has a dark blue header with the repository logo and navigation links for "Datasets" and "Contribute". Below the header, the dataset title "Cryotherapy Dataset" is prominently displayed with a yellow database icon. It was "Donated on 1/3/2018". A brief description states: "This dataset contains information about wart treatment results of 90 patients using cryotherapy." The dataset characteristics table includes rows for "Dataset Characteristics" (Tabular), "Subject Area" (Health and Medicine), "Feature Type" (Integer, Real), and "# Instances" (90). The dataset information table includes rows for "Has Missing Values?" (No) and "Dataset Information" (empty). The URL of the page is <https://archive.ics.uci.edu/ml/datasets/Cryotherapy+Treatment>.

connectionist Networks

next slide



What is Cryotherapy?



A medical treatment that uses extreme cold to destroy abnormal tissues

next slide

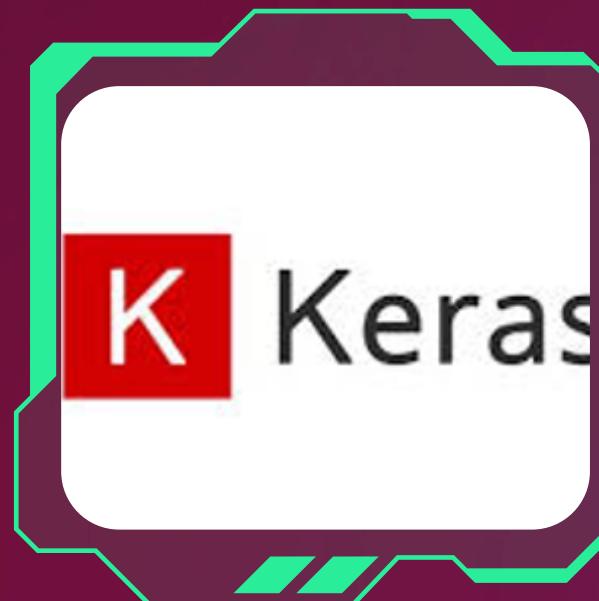


Tools & Libraries Used

Home

About Us

Contact



● **Programming Language::**



next slide ➤

```
# Charger les données
df = pd.read_excel("Cryotherapy.xlsx")
print("Colonnes : ", df.columns.tolist())

# Encodage one-hot (pour 'sex' et 'Type')
df_encoded = pd.get_dummies(df, columns=['sex', 'Type'], drop_first=True)

# Séparation features / target
X = df_encoded.drop("Result_of_Treatment", axis=1)
y = df_encoded["Result_of_Treatment"]

# Split train/test
X_train, X_test, y_train, y_test = train_test_split(
    X, y, test_size=0.2, random_state=42
)
```

```
# Normalisation
scaler = StandardScaler()
X_train = pd.DataFrame(scaler.fit_transform(X_train), columns=X_train.columns)
X_test = pd.DataFrame(scaler.transform(X_test), columns=X_test.columns)

# Conversion float32
X_train = X_train.astype("float32")
X_test = X_test.astype("float32")
y_train = y_train.astype("float32")
y_test = y_test.astype("float32")

test_data = X_test.copy()
test_data["Result_of_Treatment"] = y_test.values

# Enregistrer dans un fichier Excel
test_data.to_excel("donnees_test.xlsx", index=False)
print("✓ Fichier 'donnees_test.xlsx' sauvegardé avec succès.")
```

Preprocessing Workflow

- Import the dataset using Pandas (*Cryotherapy.xlsx*).

- Split data into Training (80%) and Testing (20%) sets

- Normalize features using StandardScaler .

- Save processed files:
training_data.xlsx and **test_data.xlsx**.

next slide ➤





```
model = Sequential([
    Dense(16, input_dim=X_train.shape[1], activation='relu',
          kernel_regularizer=regularizers.l2(0.01)),
    Dropout(0.3),
    Dense(8, activation='relu',
          kernel_regularizer=regularizers.l2(0.01)),
    Dropout(0.2),
    Dense(1, activation='sigmoid')
])

model.compile(optimizer='adam', loss='binary_crossentropy', metrics=['accuracy'])

early_stop = EarlyStopping(monitor='val_loss', patience=5, restore_best_weights=True)
```

Perceptron Structure

- ***Input Layer: 7 neurons (one for each feature).***
- ***Hidden Layer 1: Dense(16, ReLU)***
- ***Hidden Layer 2: Dense(8, ReLU)***
- ***Output Layer: Dense(1, Sigmoid) for binary classification***

next slide ➤



History Tracking

[Home](#)[About Us](#)[Contact](#)

```
history = model.fit(  
    X_train, y_train,  
    epochs=100,  
    batch_size=16,  
    validation_split=0.2,  
    callbacks=[early_stop],  
    verbose=1  
)
```

- ***Training the model with 100 epochs***
- ***Using batch size of 16 and 20% validation split***
- ***Early stopping callback to prevent overfitting***

```
# === 5. Évaluation ===  
  
loss, accuracy = model.evaluate(X_test, y_test)  
print(f"\nPrécision sur les données de test : {accuracy:.2f}")  
  
# Prédictions  
y_pred_prob = model.predict(X_test)  
y_pred = (y_pred_prob > 0.5).astype(int)  
  
print("Accuracy (sklearn):", accuracy_score(y_test, y_pred))
```

```
Précision sur les données de test : 0.89  
1/1 [=====] - 0s 135ms/step  
Accuracy (sklearn): 0.8888888888888888  
1/1 [=====] - 0s 66ms/step
```

[next slide](#)

Results

The screenshot shows a Jupyter Notebook interface with the following components:

- File Bar:** File, Edit, Search, Source, Run, Debug, Consoles, Projects, Tools, View, Help.
- File Explorer:** Shows files `temp.py` and `projet2.py`.
- Code Editor:** Displays the `projet2.py` file content:1 # === 1. Importation des bibliothèques ===
2 import pandas as pd
3 from sklearn.model_selection import train_test_split
4 from sklearn.preprocessing import StandardScaler
5 from sklearn.metrics import accuracy_score, confusion_matrix, ConfusionMat
6 import matplotlib.pyplot as plt
7 from tensorflow.keras.models import Sequential
8 from tensorflow.keras.layers import Dense, Dropout
9 from tensorflow.keras.callbacks import EarlyStopping
10 from tensorflow.keras import regularizers
11 import tkinter as tk
12 from tkinter import messagebox
13 import random
14
=== 2. Chargement et prétraitement des données ===
15
Charger les données
16 df = pd.read_excel("Cryotherapy.xlsx")
17 print("Colonnes : ", df.columns.tolist())
18
Encodage one-hot (pour 'sex' et 'Type')
19 df_encoded = pd.get_dummies(df, columns=['sex', 'Type'], drop_first=True)
20
Séparation features / target
21 X = df_encoded.drop("Result_of_Treatment", axis=1)
22 y = df_encoded["Result_of_Treatment"]
23
Split train/test
24 X_train, X_test, y_train, y_test = train_test_split(
25 X, y, test_size=0.2, random_state=42
26)
27
Normalisation
28 scaler = StandardScaler()
29 X_train = pd.DataFrame(scaler.fit_transform(X_train), columns=X_train.columns)
30 X_test = pd.DataFrame(scaler.transform(X_test), columns=X_test.columns)
31
Conversion float32
32 X_train = X_train.astype("float32")
33 X_test = X_test.astype("float32")
34 y_train = y_train.astype("float32")
35 y_test = y_test.astype("float32")
- Variable Explorer:** Shows "No variables to show".
- Console:** Shows training logs:

```
0.8667  
Epoch 98/100  
4/4 [=====] - 0s 17ms/step - loss: 0.3348 - accuracy: 0.9474 - val_loss: 0.5455 - val_accuracy:  
0.8667  
Epoch 99/100  
4/4 [=====] - 0s 16ms/step - loss: 0.3339 - accuracy: 0.9298 - val_loss: 0.5447 - val_accuracy:  
0.8667  
Epoch 100/100  
4/4 [=====] - 0s 17ms/step - loss: 0.3557 - accuracy: 0.9298 - val_loss: 0.5436 - val_accuracy:  
0.8667  
1/1 [=====] - 0s 35ms/step - loss: 0.5769 - accuracy: 0.8889
```
- Bottom Bar:** Includes a play button, time controls (1:02 to 3:50), volume control, settings, and close buttons.

next slide

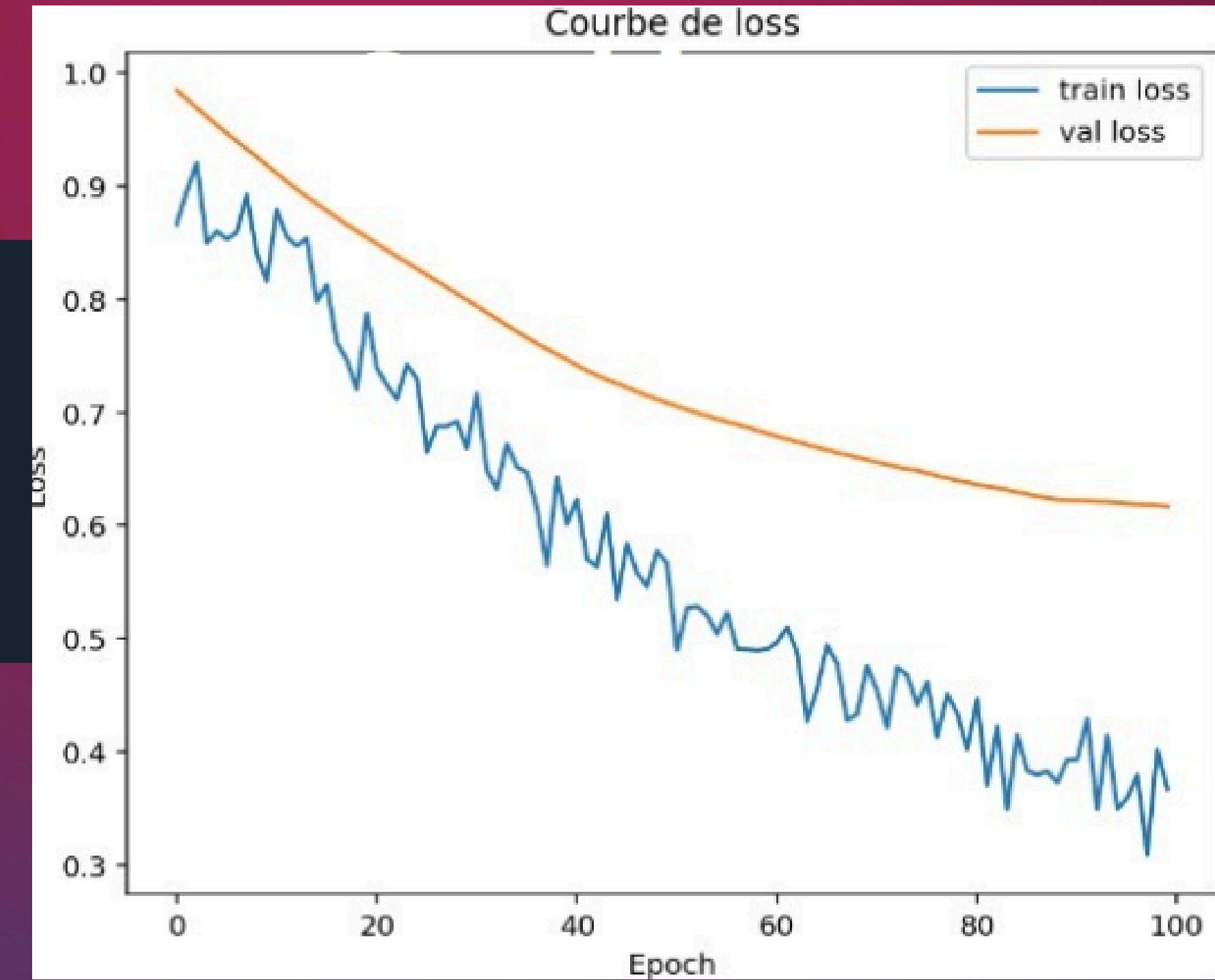
Graphics

[Home](#)[About Us](#)[Contact](#)

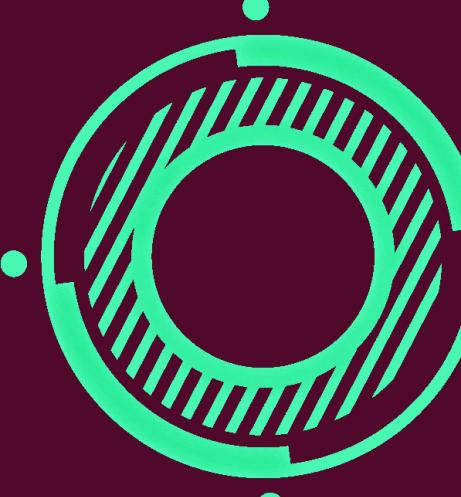
```
plt.subplot(1, 2, 1)
plt.plot(history.history['loss'], label='train loss')
plt.plot(history.history['val_loss'], label='val loss')
plt.title('Courbe de Loss')
plt.xlabel('Epoch')
plt.ylabel('Loss')
plt.legend()
```

how the model's error (loss) decreases over epochs for training and validation

connectionist Networks



next slide

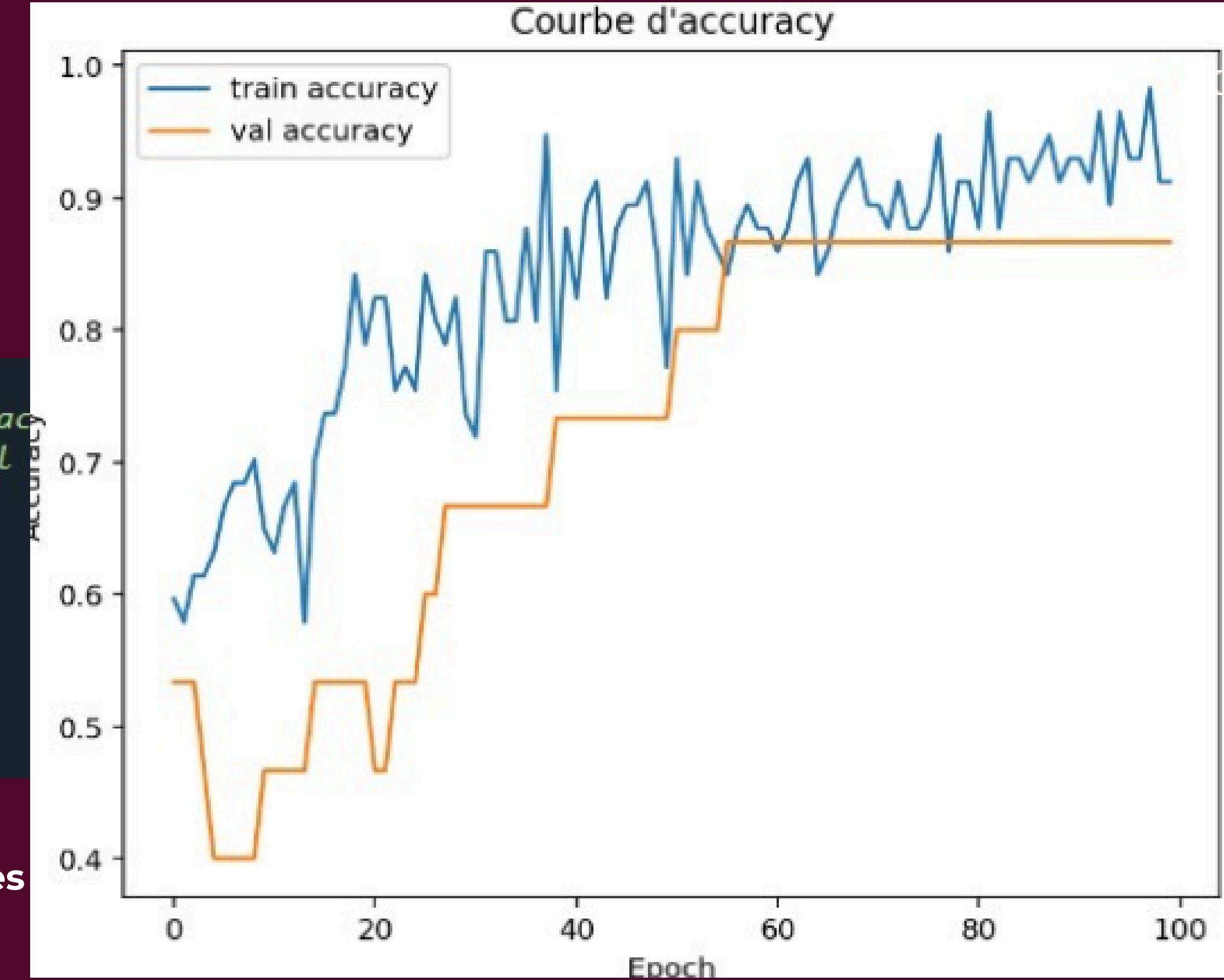


Graphics

```
plt.subplot(1, 2, 2)
plt.plot(history.history['accuracy'], label='train accuracy')
plt.plot(history.history['val_accuracy'], label='val accuracy')
plt.title('Courbe d\'accuracy')
plt.xlabel('Epoch')
plt.ylabel('Accuracy')
plt.legend()

plt.tight_layout()
plt.show()
```

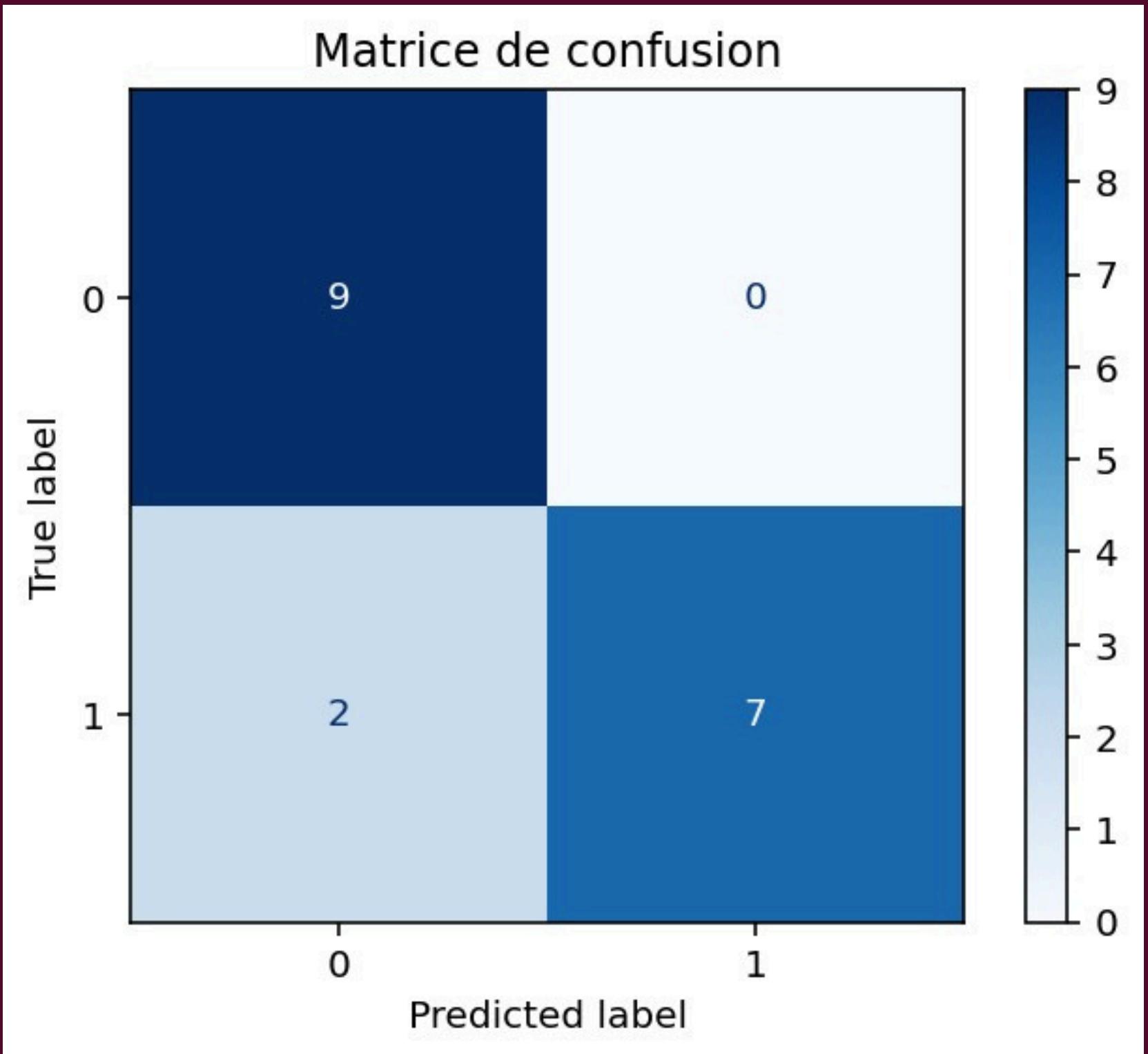
how the model's prediction accuracy improves over epochs for training and validation



Graphics

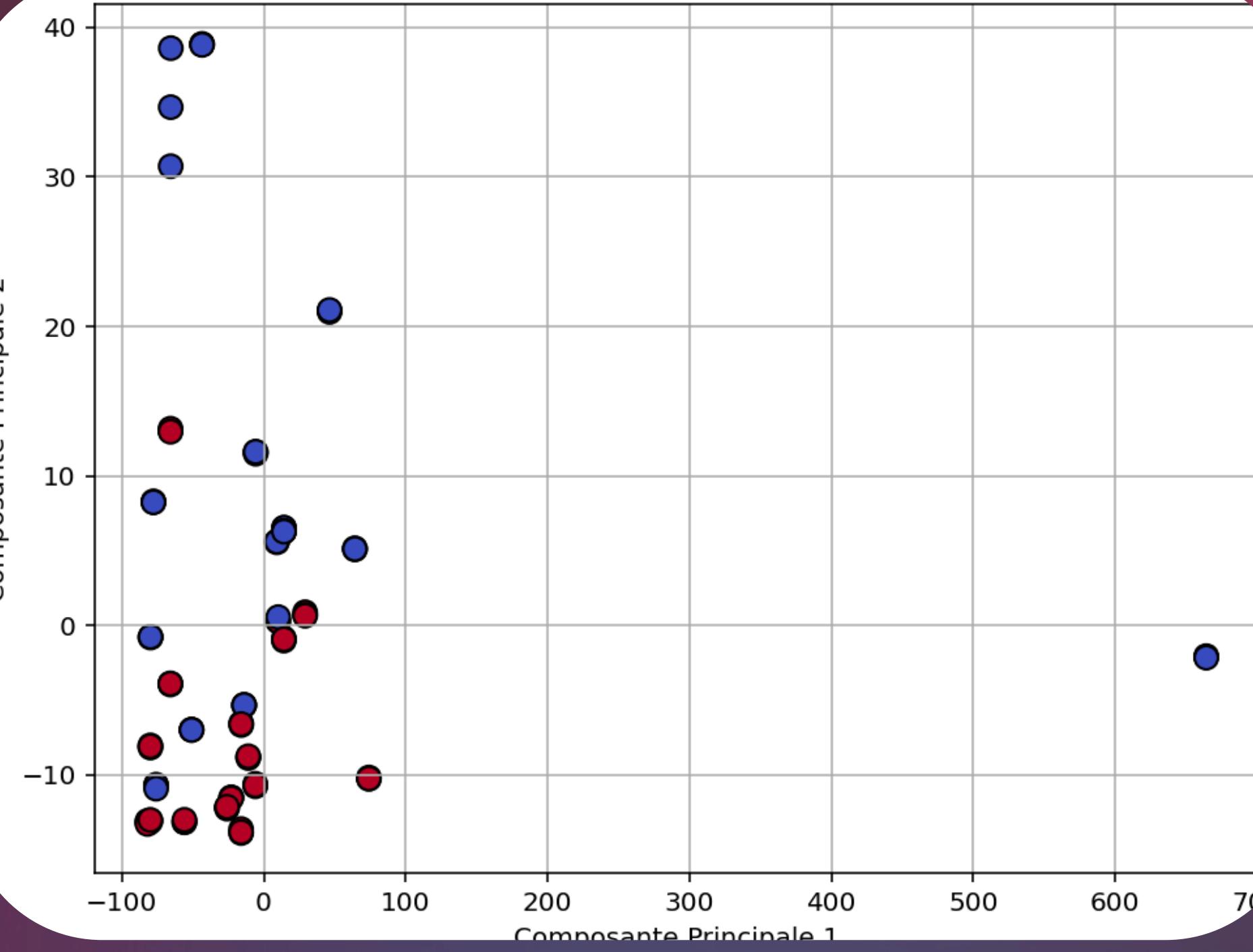
```
# Matrice de confusion
cm = confusion_matrix(y_test, y_pred)
disp = ConfusionMatrixDisplay(confusion_matrix=cm, display_labels=[0, 1])
disp.plot(cmap=plt.cm.Blues)
plt.title("Matrice de confusion")
plt.show()
```

the number of correct and incorrect predictions for each class (0 or 1)



Graphics

Projection PCA : visualisation des classes



next slide ➤

Graphical User Interface

[Home](#)[About Us](#)[Contact](#)

27 2 34 12 3 3 95 0

Cryotherapy Prediction

Cryotherapy Prediction

Âge :
34.0

Durée (Time) :
12.0

Nombre de verrues :
3.0

Zone totale :
95.0

Sexe :
 Homme
 Femme

Type de verrue :
Type 3

Résultat de la prédiction

Échec du traitement
(Probabilité : 0.44)

OK



next slide ➤

Graphical User Interface

[Home](#)[About Us](#)[Contact](#)

23 2 16 8,5 1 2 60 1

Cryotherapy Prediction

Cryotherapy Prediction

Âge :
16.0

Durée (Time) :
8.5

Nombre de verrues :
1.0

Zone totale :
60.0

Sexe :
 Homme
 Femme

Type de verrue :
Type 2

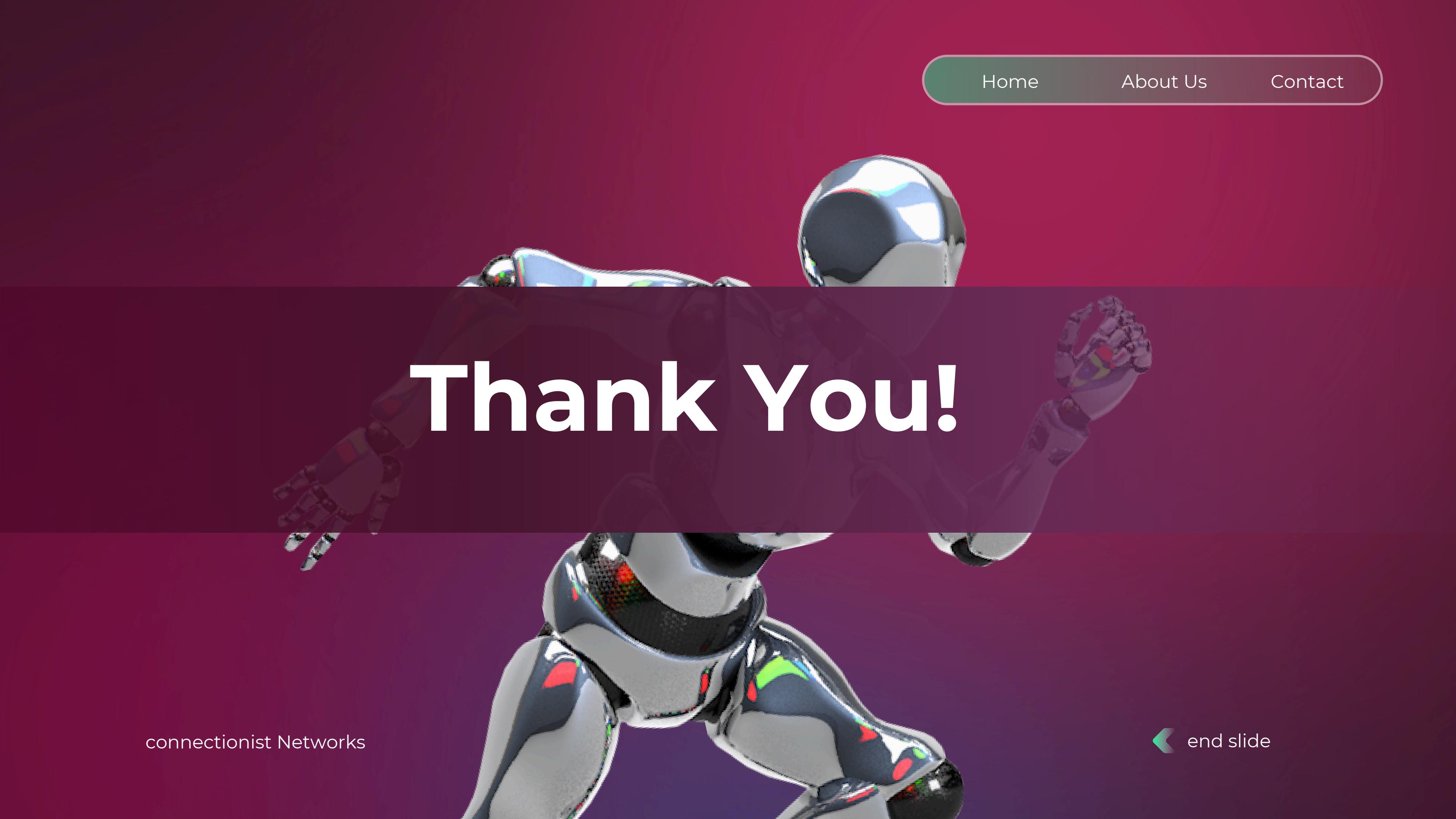
Résultat de la prédition

i Traitement efficace (Probabilité : 0.76)

OK



next slide >



Thank You!