Université IBN TOFAIL – Kénitra
ECOLE NATIONALE DES SCIENCES APPLIQUEES

Cours Intelligence Artificielle

CH5: Machine Learning - Apprentissage Supervisé:

SVM (Support Vector Machines)

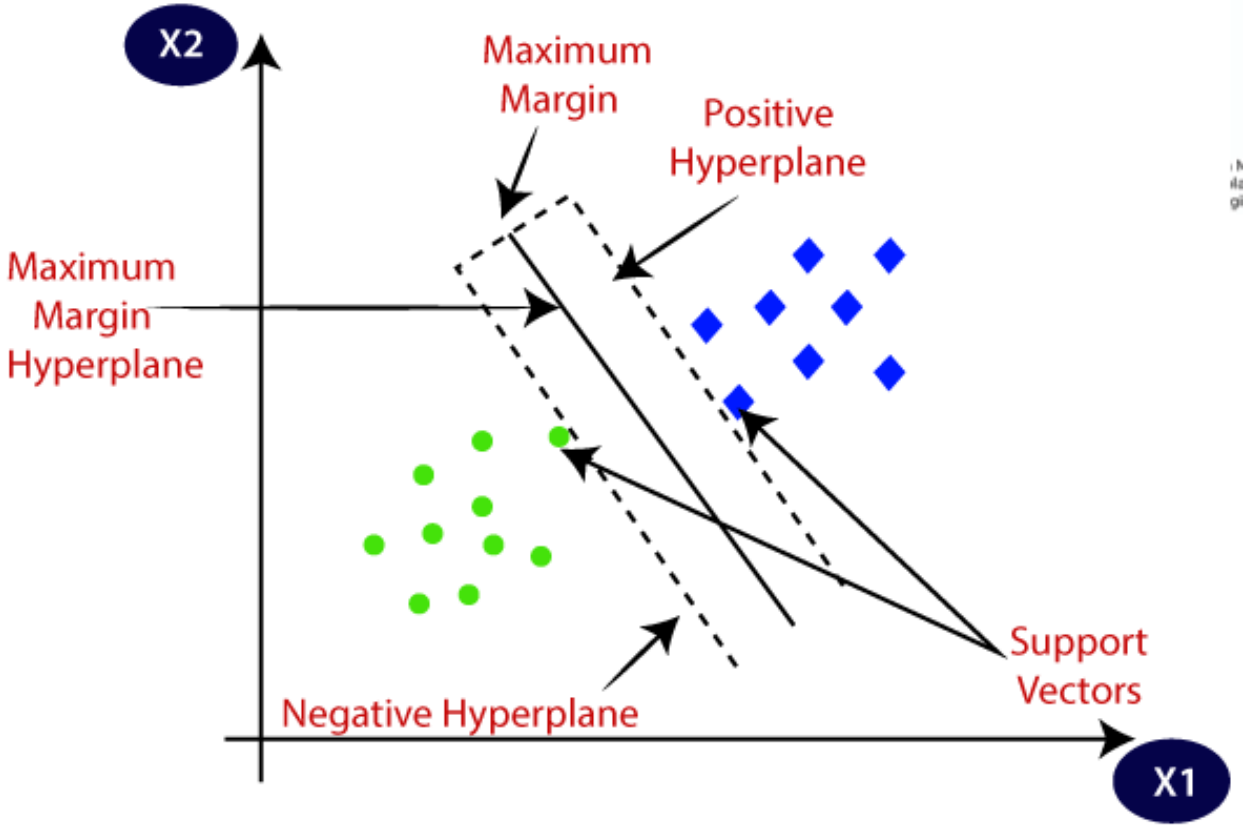
Par:

Hassan EL FADIL

elfadil.h@gmail.com

1. Introduction aux SVM

- Machines à Vecteurs de Support: Support Vector Machines (**SVM**).
- Les SVM sont des algorithmes supervisés utilisés pour la classification et la régression.
- Proposés par Vladimir Vapnik dans les années **1990**.
- Principes de base : Trouver l'hyperplan qui sépare au mieux deux classes en maximisant la marge entre elles.



2. Formulation mathématique du problème des SVM

2.1. L'hyperplan:

Un hyperplan dans un espace $\mathbb{R}^n$ est défini par :

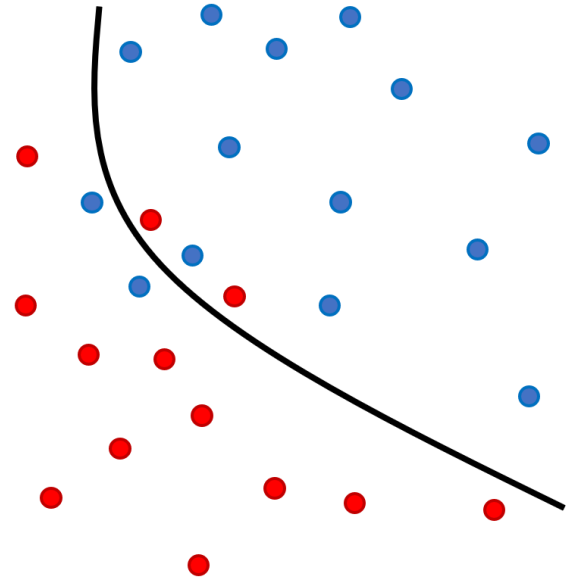
$$\omega \cdot x + b = 0 \quad \Leftrightarrow \quad \sum_{i=1}^{d} \omega_i x_i + b = 0$$

- ω : vecteur normal à l'hyperplan, appelé le vecteur de poids.
- b : biais ou constante.
- x : vecteur de données (point de l'espace des caractéristiques).
- d : dimension de l'espace des caractéristiques

2.2. Fonction de décision:

Pour une donnée x_i, la fonction de décision est :

$$f(x) = \text{sign}(\omega \cdot x + b)$$



2.3. Condition de séparation:

Pour une séparation correcte des deux classes, les contraintes suivantes doivent être respectées :

$$y_i(\omega \cdot x_i + b) \geq 1 \ , \qquad \forall i$$

où $y_i \in \{-1, +1\}$ est la classe de l'échantillon x_i (Problème de classification à 2 classes).

… … Formulation mathématique
Maximisation de la marge

- La marge est la distance entre l'hyperplan et les points de données les plus proches (vecteurs de support).

- La distance entre un point x_i et l'hyperplan est donnée par :

$$distance: d = \frac{|\omega \cdot x_i + b|}{\|\omega\|}$$

3. Maximisation de la marge

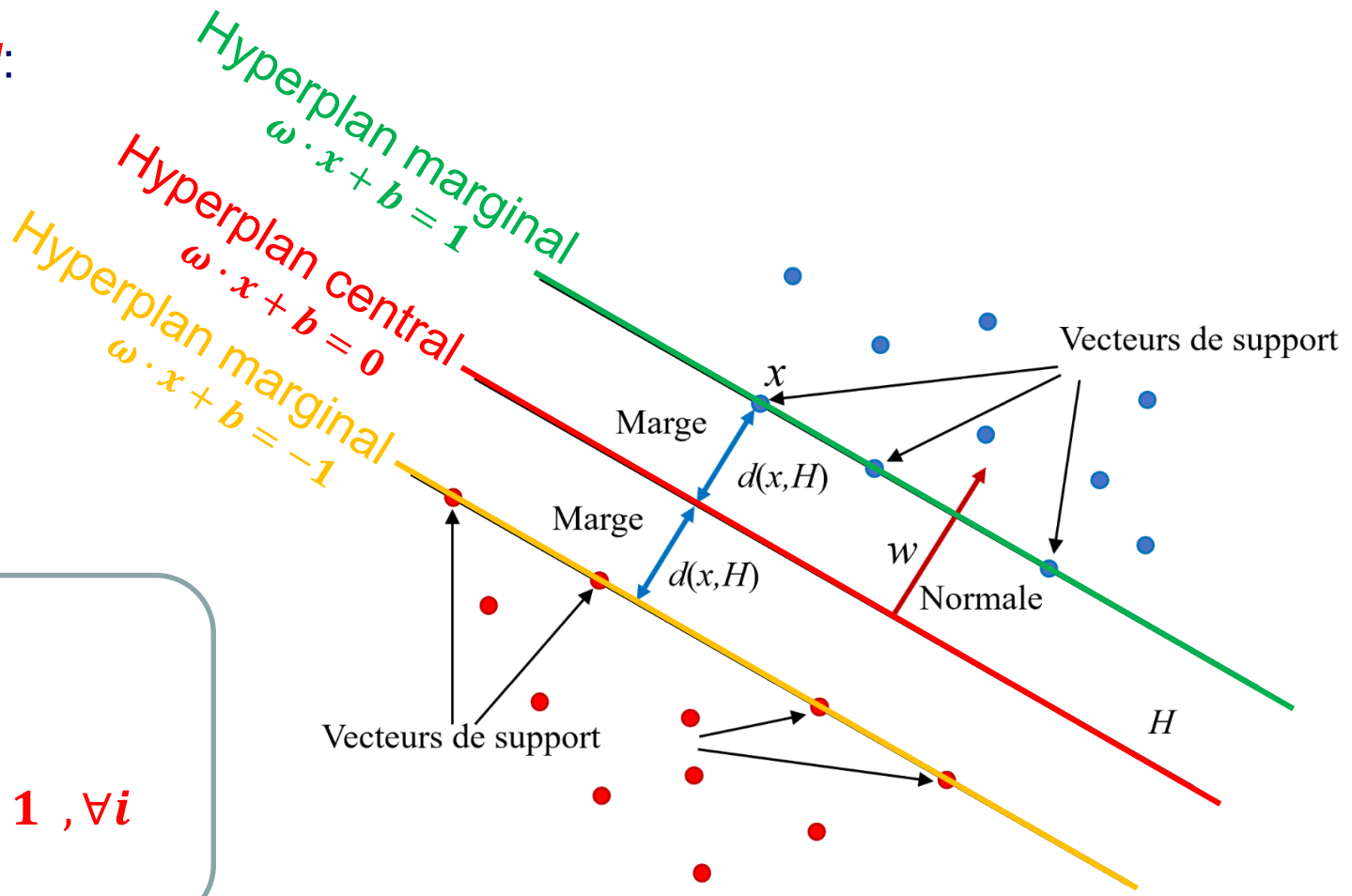
- L'objectif est de maximiser la marge ***M***:

$$M = \frac{2}{\|\omega\|}$$

- ce qui revient à minimiser $\|\omega\|^2$.

- Le problème d'optimisation devient :

$$\begin{cases} \min\limits_{\omega,b} \frac{1}{2}\|\omega\|^2 \\ \textbf{sous contrainte} \quad y_i(\omega \cdot x_i + b) \geq 1 \ , \forall i \end{cases}$$



4. SVM avec données non linéaires : le noyau

- Lorsque les données ne sont pas linéairement séparables, on utilise une transformation non linéaire $\phi(x)$ pour projeter les données dans un espace de dimension supérieure. Dans cet espace, elles deviennent linéairement séparables.

4.1. Fonction noyau:

- Le noyau $K(x_i, x_j)$ est défini comme un produit scalaire dans l'espace projeté :

$$K(x_i, x_j) = \phi(x_i) \cdot \phi(x_j)$$

4.2. Noyaux courants:

1. Noyau linéaire:

$$K(x_i, x_j) = x_i \cdot x_j$$

2. Noyau polynomial:

$$K(x_i, x_j) = (x_i \cdot x_j + c)^d$$

Où c est une constante et d est le degré du polynôme.

3. Noyau RBF (Radial Basis Function) :

$$K(x_i, x_j) = \exp\left(-\frac{\|x_i - x_j\|^2}{2\sigma^2}\right)$$

Où σ contrôle la largeur du noyau.

5. Problème d'optimisation dual

Le problème primal peut être reformulé comme un problème dual pour faciliter le calcul avec les noyaux. La formulation duale est :

$$\max_{\alpha} \sum_{i=1}^{n} \alpha_i - \frac{1}{2} \sum_{i=1}^{n} \sum_{j=1}^{n} \alpha_i \alpha_j y_i y_j K(x_i, x_j)$$

Sous contraintes :

$$\begin{cases} \sum_{i=1}^{n} \alpha_i y_i = 0 \quad \textit{(stationarité)} \\ 0 \leq \alpha_i \leq C \quad \textit{(admissibilité duale)} \end{cases}$$

… Problème d'optimisation dual

- La solution du problème dual donne les multiplicateurs de Lagrange optimaux α_i.
- A partir des α_i on obtient ω par les relations :

$$\omega = \sum_{i=1}^{n} \alpha_i y_i x_i$$

- Le paramètre b est obtenu à partir de la relation:

$$\left| x_s^T \omega + b \right| = 1$$

valable pour tous les vecteurs de support x_s.

6. Application: Classification d'images avec le jeu de données Digits

Scikit-learn est une bibliothèque open-source de **machine learning** (apprentissage automatique) en **Python**.
Elle est largement utilisée pour les tâches d'apprentissage supervisé et non supervisé, telles que la classification, la régression, la réduction de dimensionnalité, le clustering, et bien plus encore.
La bibliothèque **Scikit-learn**, offre plusieurs jeux de données:

- **Iris** : jeu de données qui contient des informations sur trois espèces de fleurs d'iris, à savoir Setosa, Versicolor, et Virginica,
- **Digits:** Jeu de données pour la Reconnaissance de chiffres manuscrits
- **Breast Cancer** : Jeu de données pour la Classification des tumeurs mammaires)
- **California Housing :** Jeu de données pour la Prédiction du prix des maisons
- **20 Newsgroups :** Jeu de données pour la Classification de texte
- etc.

6.1. Fonction noyau:

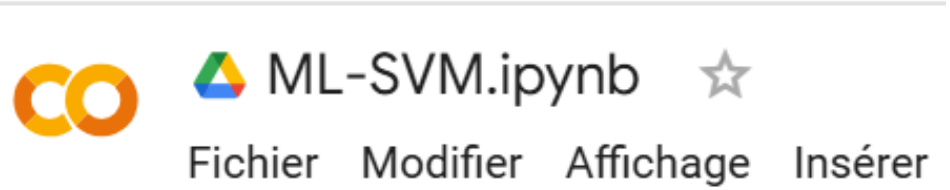
Implémentation d'un modèle **SVM** sur le jeu de données **Digits** en utilisant **Google Colab**.

6.2. Etapes:

1. Charger le jeu de données Digits.
2. Diviser les données en ensembles d'entraînement et de test.
3. Créer et entraîner un modèle SVM.
4. Évaluer les performances du modèle.

6.3. Nouveau Notebook:

- Ouvrir Google Colab et créer un **Nouveau Notebook**.
  Renommer le **ML-SVM.ipynb**

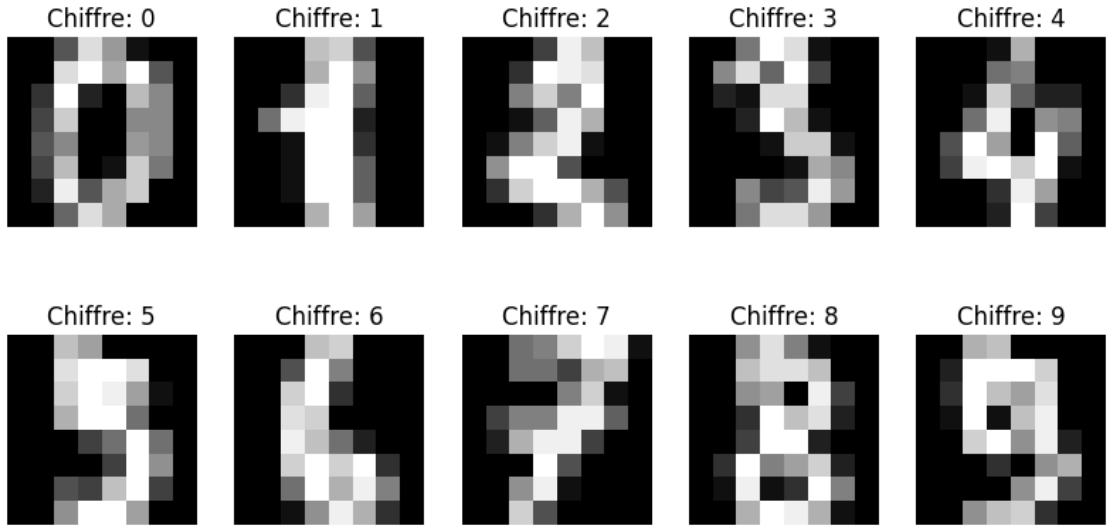


6.4. Code Python…… 1/6:

```
# Étape 1 : Importer les bibliothèques nécessaires
import numpy as np
import matplotlib.pyplot as plt
from sklearn import datasets
from sklearn.model_selection import train_test_split
from sklearn.preprocessing import StandardScaler
from sklearn.svm import SVC
from sklearn.metrics import classification_report, confusion_matrix

# Étape 2 : Charger le jeu de données Digits
digits = datasets.load_digits()
```

6.4. Code Python…… 2/6:

```
# Étape 3 : Visualiser quelques chiffres du jeu de données
fig, axes = plt.subplots(2, 5, figsize=(10, 5))
axes = axes.ravel()

for i in np.arange(10):
    axes[i].imshow(digits.images[i], cmap='gray')  # Afficher chaque image en niveaux de gris
    axes[i].set_title('Chiffre: %i' % digits.target[i])  # Titre de l'image avec la valeur cible
    axes[i].axis('off')  # Masquer les axes

plt.show()  # Afficher les images
```



6.4. Code Python…… 3/6:

```
# Étape 4 : Prétraiter les données
# Aplatir les images (chaque image contient 64 pixels)
X = digits.images.reshape((digits.images.shape[0], -1))  # Reshaper les images en un vecteur 1D
y = digits.target  # Labels des chiffres

# Diviser les données en ensembles d'entraînement et de test
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.3, random_state=42)

# Standardiser les données (SVM fonctionne mieux avec des données mises à l'échelle)
scaler = StandardScaler()
X_train = scaler.fit_transform(X_train)  # Ajuster et transformer les données d'entraînement
X_test = scaler.transform(X_test)  # Transformer les données de test
```

6.4. Code Python…… 4/6:

```
# Étape 5 : Entraîner un classificateur SVM
svm = SVC(kernel='linear')  # Utilisation d'un noyau linéaire pour simplifier l'exemple
svm.fit(X_train, y_train)  # Entraîner le modèle SVM avec les données d'entraînement

# Étape 6 : Faire des prédictions sur l'ensemble de test
y_pred = svm.predict(X_test)
```

6.4. Code Python…… 5/6:

```
# Étape 7 : Évaluer le modèle
print("Rapport de classification :")
print(classification_report(y_test, y_pred))  # Afficher le rapport de classification

print("Matrice de confusion :")
print(confusion_matrix(y_test, y_pred))  # Afficher la matrice de confusion
```

Rapport de classification :

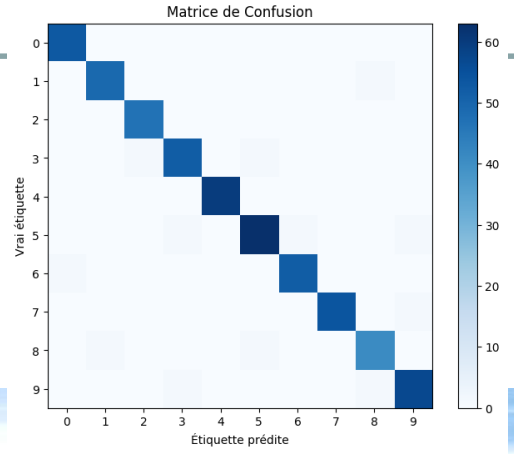
	precision	recall	f1-score	support
0	0.98	1.00	0.99	53
1	0.98	0.98	0.98	50
2	0.98	1.00	0.99	47
3	0.96	0.96	0.96	54
4	1.00	1.00	1.00	60
5	0.97	0.95	0.96	66
6	0.98	0.98	0.98	53
7	1.00	0.98	0.99	55
8	0.95	0.95	0.95	43
9	0.97	0.97	0.97	59
accuracy			0.98	540
macro avg	0.98	0.98	0.98	540
weighted avg	0.98	0.98	0.98	540

Matrice de confusion :

```
[[53  0  0  0  0  0  0  0  0  0]
 [ 0 49  0  0  0  0  0  0  1  0]
 [ 0  0 47  0  0  0  0  0  0  0]
 [ 0  0  1 52  0  1  0  0  0  0]
 [ 0  0  0  0 60  0  0  0  0  0]
 [ 0  0  0  1  0 63  1  0  0  1]
 [ 1  0  0  0  0  0 52  0  0  0]
 [ 0  0  0  0  0  0  0 54  0  1]
 [ 0  1  0  0  0  1  0  0 41  0]
 [ 0  0  0  1  0  0  0  0  1 57]]
```

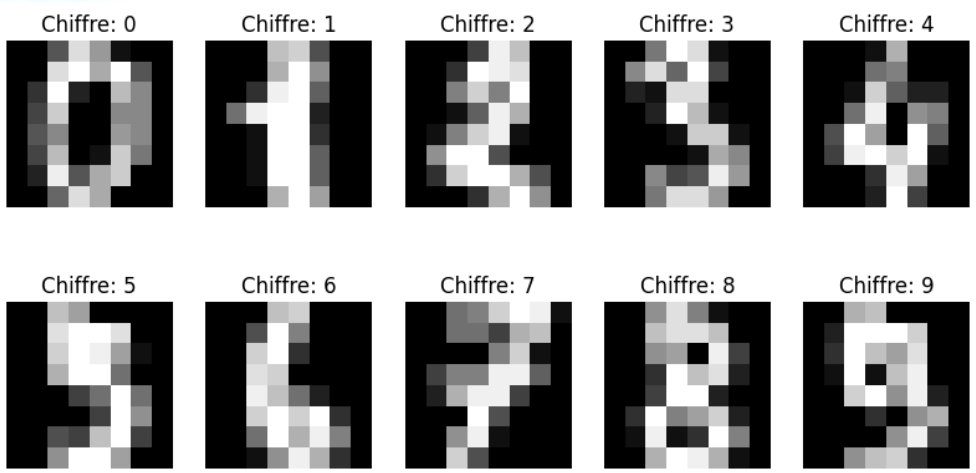
6.4. Code Python…… 6/6:

```
# Étape 8 : Tracer la matrice de confusion
plt.figure(figsize=(8, 6))
# Afficher la matrice de confusion en bleu
plt.imshow(confusion_matrix(y_test, y_pred), cmap='Blues', interpolation='nearest')
plt.title("Matrice de Confusion")
plt.colorbar()  # Ajouter une barre de couleur
plt.ylabel('Vrai étiquette')  # Étiquette de l'axe des ordonnées
plt.xlabel('Étiquette prédite')  # Étiquette de l'axe des abscisses
plt.xticks(np.arange(10), np.arange(10))  # Étiqueter l'axe x avec les chiffres
plt.yticks(np.arange(10), np.arange(10))  # Étiqueter l'axe y avec les chiffres
plt.show()  # Afficher la matrice de confusion
```



6.5. Résultats affichés:

Images des digits



Rapport de classification

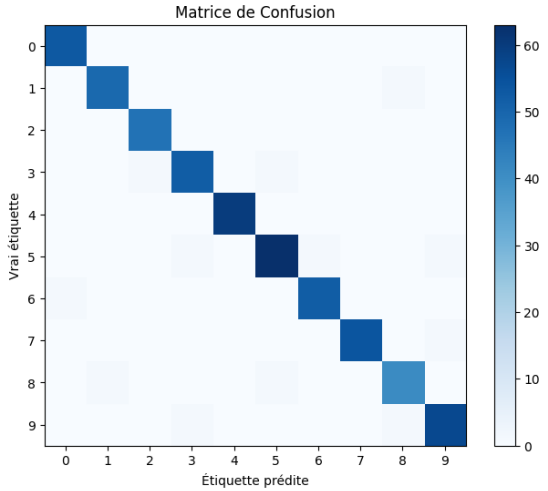
Rapport de classification :

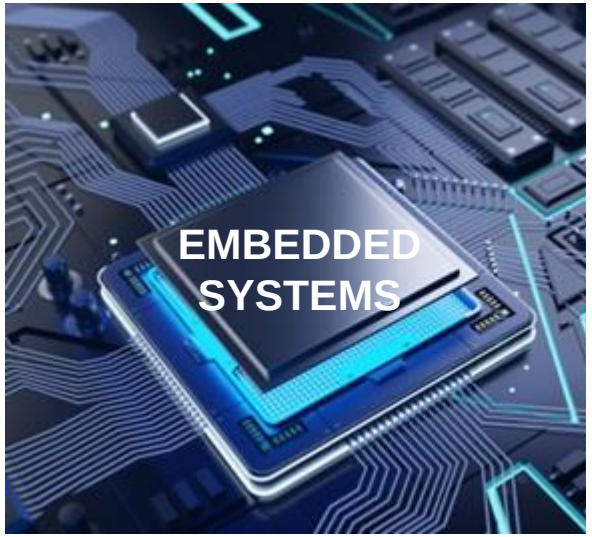
	precision	recall	f1-score	support
0	0.98	1.00	0.99	53
1	0.98	0.98	0.98	50
2	0.98	1.00	0.99	47
3	0.96	0.96	0.96	54
4	1.00	1.00	1.00	60
5	0.97	0.95	0.96	66
6	0.98	0.98	0.98	53
7	1.00	0.98	0.99	55
8	0.95	0.95	0.95	43
9	0.97	0.97	0.97	59
accuracy			0.98	540
macro avg	0.98	0.98	0.98	540
weighted avg	0.98	0.98	0.98	540

Matrice de confusion sous forme de texte

```
Matrice de confusion :
[[53  0  0  0  0  0  0  0  0  0]
 [ 0 49  0  0  0  0  0  0  1  0]
 [ 0  0 47  0  0  0  0  0  0  0]
 [ 0  0  1 52  0  1  0  0  0  0]
 [ 0  0  0  0 60  0  0  0  0  0]
 [ 0  0  0  1  0 63  1  0  0  1]
 [ 1  0  0  0  0  0 52  0  0  0]
 [ 0  0  0  0  0  0  0 54  0  1]
 [ 0  1  0  0  0  1  0  0 41  0]
 [ 0  0  0  1  0  0  0  0  1 57]]
```

Matrice de confusion sous forme d'image (heatmap)





FIN !
Questions ?