iTeam University	**Année Universitaire :** 2023/2024
Département : Informatique	**Niveau :** LA_GL_3
Module : Machine Learning	**Enseignante :** Noussaiba Jaafar

TP2 – Breast Cancer Classification Using Support Vector Machine (SVM) (Classification du cancer du sein en utilisant les machines à Vecteurs de Support)

Background et objectif :

Le cancer du sein est le cancer le plus fréquent chez les femmes dans le monde. Il représente 25 % de tous les cas de cancer et a touché plus de 2,1 millions de personnes rien qu'en 2015. Cela commence lorsque les cellules du sein commencent à se développer de manière incontrôlable. Ces cellules forment généralement des tumeurs qui peuvent être vues par rayons X ou ressenties sous forme de grumeaux dans la région du sein.

Un diagnostic précoce augmente considérablement les chances de survie. Les principaux défis contre sa détection sont de savoir comment classer les tumeurs en malignes (cancéreuses) ou bénignes (non cancéreuses). Une tumeur est considérée comme maligne si les cellules peuvent se développer dans les tissus environnants ou se propager à des zones éloignées du corps. Une tumeur bénigne n'envahit pas les tissus voisins ni ne se propage à d'autres parties du corps comme le peuvent les tumeurs cancéreuses. Mais les tumeurs bénignes peuvent être graves si elles appuient sur des structures vitales telles que les vaisseaux sanguins ou les nerfs.

La technique d'apprentissage automatique peut considérablement améliorer le niveau de diagnostic du cancer du sein. La recherche montre que les médecins expérimentés peuvent détecter le cancer avec une précision (accuracy) de 79 %, tandis qu'une précision de 91 % (parfois jusqu'à 97 %) peut être obtenue à l'aide de techniques d'apprentissage automatique.

Dans ce TP, l'objectif consiste à classer les tumeurs en tumeurs malignes (cancéreuses) ou bénignes (non cancéreuses) à l'aide de caractéristiques obtenues à partir de plusieurs images cellulaires.

Les caractéristiques sont calculées à partir d'une image numérisée d'une aspiration à l'aiguille fine (FNA) d'une masse mammaire. Ils décrivent les caractéristiques des noyaux cellulaires présents dans l'image.

1. Jeux de données (Dataset)

a. Breast Cancer Wisconsin (Diagnostic) DataSet

Cet ensemble de données contient des caractéristiques dérivées de l'imagerie numérisée d'aspirations à l'aiguille fine d'une masse de cellules tumorales du sein. L'objectif de cette analyse est d'entrainer un algorithme d'apprentissage automatique supervisé pour distinguer avec exactitude une tumeur bénigne d'une tumeur maligne afin de faciliter le diagnostic clinique.

b. A vous!

```
# Load data from a csv file
import pandas as pd
bcdf = pd.read_csv('data.csv')
print(bcdf)

# Numerize diagnosis "M" malignant; "B" benign using a dictionary and map()
diagnosis_coder = {'M':1, 'B':0}
bcdf.diagnosis = bcdf.diagnosis.map(diagnosis_coder)

# Drop unecessary columns
bcdf.drop(['id'], axis = 1, inplace = True)

# Reorder columns so diagnosis is right-most
# First define a diagnosis series object
diagnosis = bcdf.diagnosis
print(diagnosis)
print(bcdf)

# Then drop diagnosis from dataframe
bcdf.drop('diagnosis', axis = 1, inplace = True)
```

```
# Then append diagnosis to end of dataframe
bcdf['Diagnosis'] = diagnosis

# Take a quick glimpse of the dataset
print(bcdf.head())

# Quick glimpse of tumor features (mean values) in relation to diagnosis
print(bcdf.groupby('Diagnosis').mean())

# For visual comparisons of differential diagnosis...
# create two dataframes - one for benign, one for malignant tumor data

bcdf_n = bcdf[bcdf['Diagnosis'] == 0]
print(bcdf_n)
bcdf_y = bcdf[bcdf['Diagnosis'] == 1]
print(bcdf_y)

# Let's plot the first 5 variables (features)
import matplotlib.pyplot as plt
import seaborn as sns
sns.pairplot(bcdf, hue= 'Diagnosis', vars = bcdf.columns[0:5])
plt.show()

# Quick visualization of relationships between features and diagnosis
# Correlations between the features
sns.heatmap(bcdf.corr())
sns.set_style("whitegrid")
plt.show()
```

2. Diviser les données en ensemble d'entraînement et ensemble de test

```
# Split data into testing and training set. Use 80% for training
```

```
from sklearn.model_selection import cross_val_score, train_test_split
X_train, X_test, y_train, y_test = train_test_split(bcdf.iloc[:,:-1], bcdf['Diagnosis'],
train_size = .8 )
```

3. Pré-traitement : Normalisation des caractéristiques (entre 0 et 1)

```
# Feature Normalization
# Instantiate
from sklearn.preprocessing import Normalizer
norm = Normalizer()

# Fit
norm.fit(X_train)

# Transform both training and testing sets
X_train_norm = norm.transform(X_train)
X_test_norm = norm.transform(X_test)
```

4. L'algorithme SVM (Support Vector Machine) ou Machines à Vecteurs de support

Les machines à vecteurs de support (ou Support Vector Machine, SVM) sont une famille d'algorithmes d'apprentissage automatique de type supervisé et qui peuvent être utilisées pour des problèmes de discrimination (à quelle classe appartient un échantillon), de régression et de détection d'anomalies.

Dans les SVM, les données sont séparées en plusieurs classes en recourant à la "marge maximale", avec une frontière de séparation choisie pour maximiser la distance entre les groupes de données. La difficulté est de déterminer cette frontière optimale : pour y parvenir, il faut formuler le problème comme un problème d'optimisation quadratique. Les points les plus proches de la frontière sont nommés "vecteurs support", et c'est d'eux que provient le nom de cette famille d'algorithmes. La seconde idée des SVM est de transformer l'espace de représentation des données d'entrées pour en faire un espace de plus grande dimension, dans lequel il existe une séparation linéaire, pour pouvoir traiter des données difficiles à séparer linéairement.

a. Paramètres d'un classifieur SVM

```
# SVM classifier
# Define parameters for optimization
```

- C : paramètre de régularisation
- Gamma : correspond au coefficient de noyau pour les noyaux « rbf », « poly », et « sigmoid ». Prend comme valeur soit « scale », « auto » ou un nombre à virgule.
- Kernel : spécifie le type de noyau à utiliser dans l'algorithme, et qui peut être soit « linear », « poly », « rbf », « sigmoid », par défaut il prend la valeur « rbf ».

```
from sklearn.svm import SVC
svm_clf = SVC(C = 10, gamma = 'auto', kernel = 'linear')
```

b. Entrainement du classifieur

```
svm_clf.fit(X_train_norm, y_train)
```

c. Effectuer les prédictions

```
y_pred = svm_clf.predict(X_test_norm)
```

d. Evaluer les prédictions

```
From sklearn.metrics import confusion_matrix, accuracy_score,
classification_report



# Confusion matrix
cm = pd.DataFrame(confusion_matrix(y_test, y_pred), index = ['Actual Negative','Actual Positive'], columns = ['Predicted Negative','Predictive Positive'])
```

```
print(cm)

# Accuracy
accuracy = accuracy_score(y_test, y_pred)*100
print('Accuracy of our model is equal ' + str(round(accuracy, 2)) + ' %.')

from sklearn.model_selection import StratifiedKFold
kfold = StratifiedKFold(n_splits=10, random_state=7, shuffle=True)
cv_results = cross_val_score(svm_clf, X_train, y_train, cv=kfold,
scoring='accuracy')
print('%s: %f (%f)' % ('svm', cv_results.mean(), cv_results.std()))

# Classification Report
print('SVM Model Classification Report')
print(classification_report(y_test, y_pred ))
```