

# Présentation 4

- Mise en place d'une grille pour l'optimisation des paramètres de XGBoost
  - Comparaison selon différents critères
  - Problèmes et à venir

# Rappels :

Travail sur PPMI\_Gait\_ArmSwing :



- The gait system used includes three lightweight wireless wearable sensors containing three axial accelerometers, gyroscopes and magnetometers
- 6 tests made : SWAY eyes open/closed + TUG 1 + TUG 2 + Usual walk + Dual task walk (voir pdf)
- Motor features extracted from the raw accelerometer and gyroscope signals
- Data processing with 54 measures calculated

# XGBoost



Procédure pour le meilleur set d'hyperparamètre mais sans faire d'overfitting :

- CrossValidation

Demande en calcul enorme car pour chaque set d'hyperparamètre il faut tester K-Fold CV => Nombre d'execution de l'algo = (nbre\_set\*K):

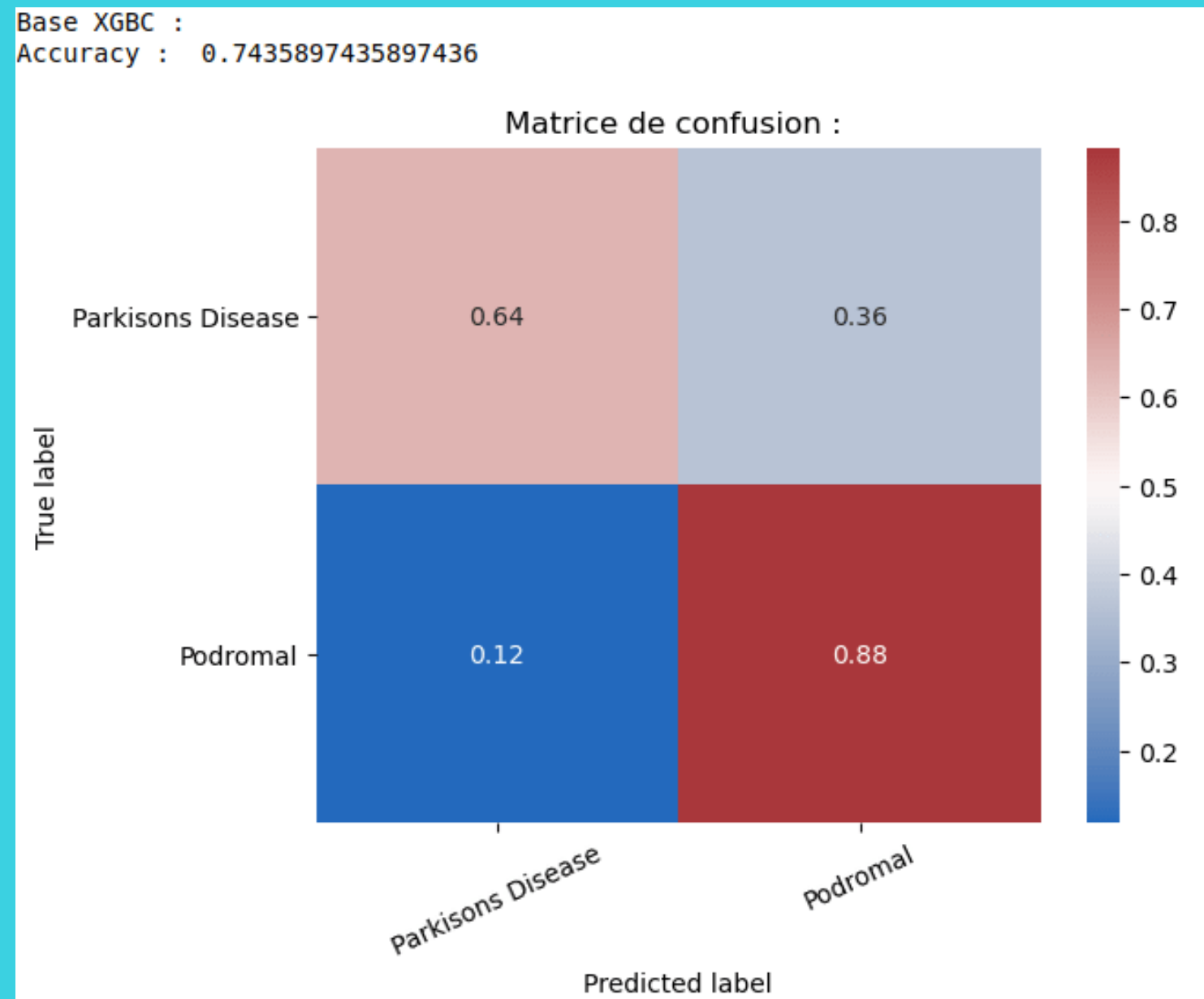
- Faire un premier balayage Random généralisé sur les set d'hyperparamètre les plus performant + comparer avec set de base
- Puis effectuer un recherche plus précise avec une grille sur le set random le plus précis précédent + comparer avec set de base
- Enfin, tableau récapitulatif des performances

# XGBoost



Première approche avec initialisation basique qui sert de référence :

```
xgbc = xgb.XGBClassifier(eval_metric='mlogloss')
```




Rapport :

	precision	recall	f1-score	support
1.0	0.88	0.64	0.74	22
3.0	0.65	0.88	0.75	17
accuracy			0.74	39
macro avg	0.76	0.76	0.74	39
weighted avg	0.78	0.74	0.74	39

K-fold Cross Validation vaut en moyenne: 0.68

# XGBoost

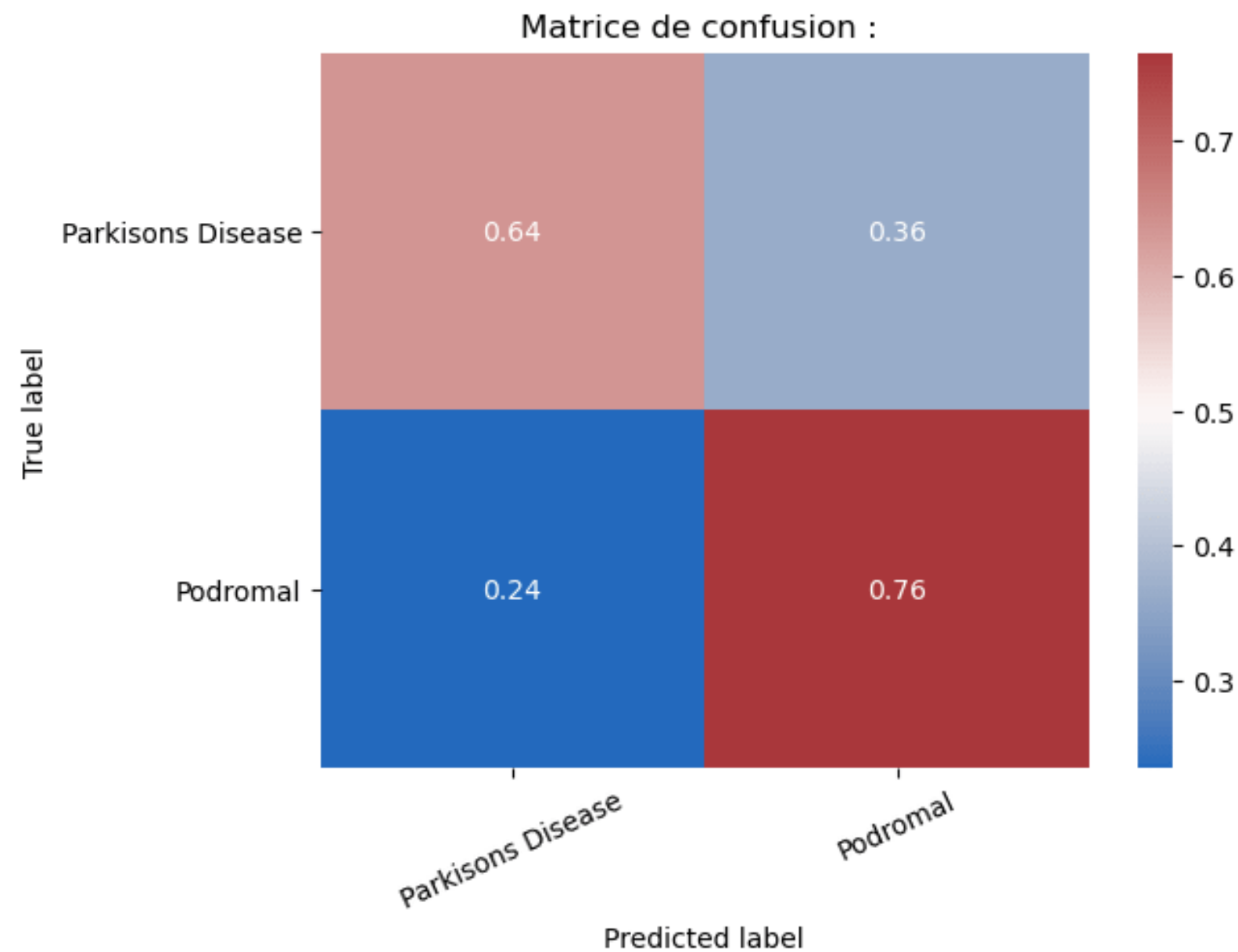
```
# 1er balayage sur large éventail de sets d'hyperparamètres avec RandomizedSearchCV
grille_xgb = {
    'learning_rate': [0.05, 0.1, 0.3, 0.7],
    'n_estimators': [50, 200, 500, 1000],
    'max_depth': [i for i in range(3, 20, 2)],
    'lambda': [1],
    'min_child_weight': [1, 5, 10],
    'gamma': [0.01, 0.1, 0.5, 1],
    'subsample': [0.1, 0.5, 1, 5],
    'colsample_bytree': [0.05, 0.1, 0.5, 1],
    'max_depth': [2, 5, 10, 20]
}
# Choix aléatoire parmi 4*4*9*1*3*4*4*4*4=110592 combinaisons de paramètres possibles
```



```
{'subsample': 1,
 'n_estimators': 200,
 'min_child_weight': 10,
 'max_depth': 10,
 'learning_rate': 0.7,
 'lambda': 1,
 'gamma': 0.01,
 'colsample_bytree': 0.05}
```

# XGBoost

Best Random XGBC :  
Accuracy : 0.6923076923076923



Rapport :


	precision	recall	f1-score	support
1.0	0.78	0.64	0.70	22
3.0	0.62	0.76	0.68	17
accuracy			0.69	39
macro avg	0.70	0.70	0.69	39
weighted avg	0.71	0.69	0.69	39

K-fold Cross Validation vaut en moyenne: 0.69

# XGBoost

## Analyse plus fine des paramètres :

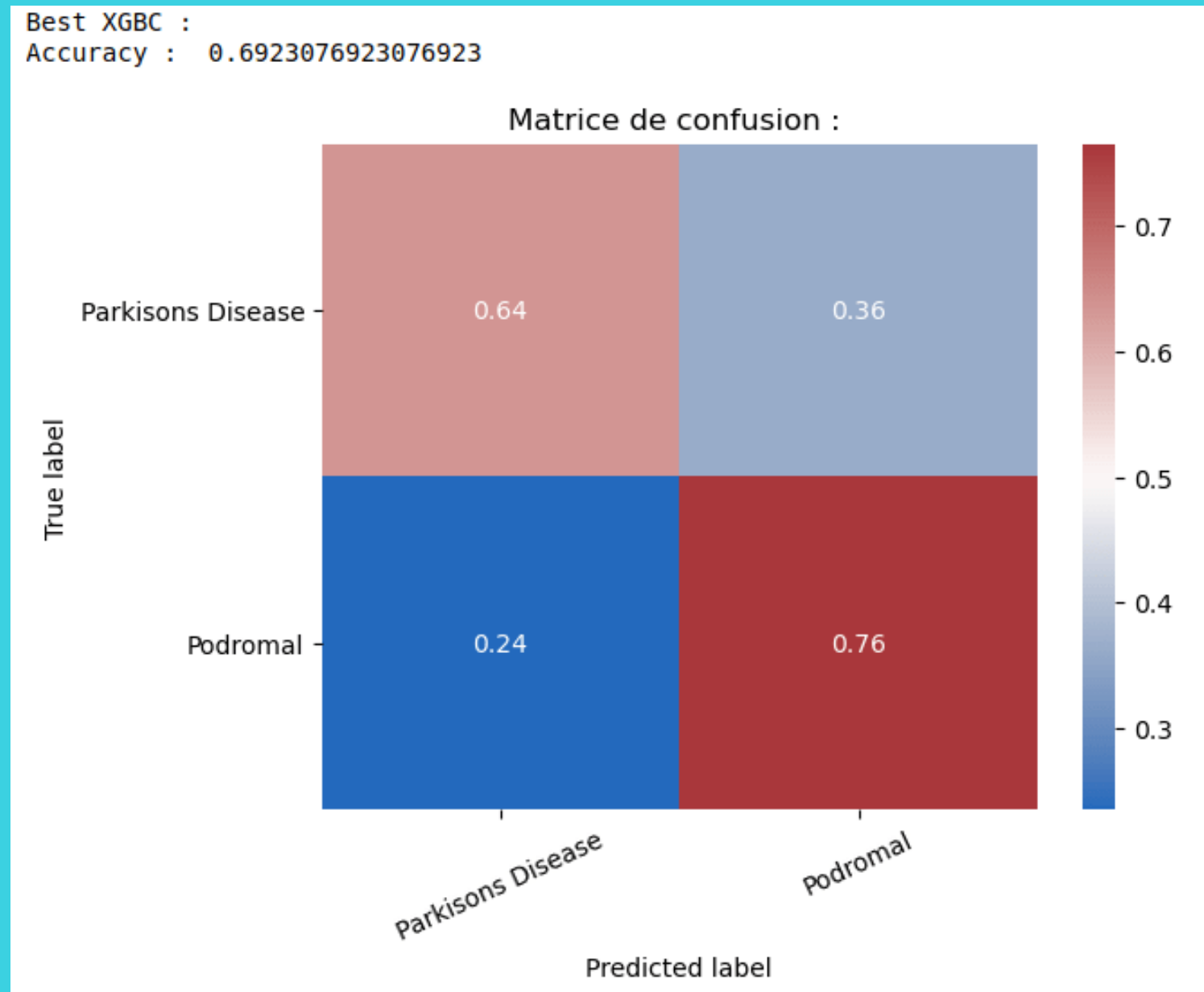
```
best_grille_xgb = {  
    'learning_rate': [0.5, 0.7, 0.9],  
    'n_estimators': [100, 200, 300, 400, 500],  
    'max_depth': [8, 10, 12],  
    'lambda': [1],  
    'min_child_weight': [8, 10, 12],  
    'gamma': [0.01, 0.05],  
    'subsample': [0.8, 1, 2],  
    'colsample_bytree': [0.01, 0.03],  
    'max_depth': [8, 10, 12]  
}
```



```
{  
    'colsample_bytree': 0.01,  
    'gamma': 0.01,  
    'lambda': 1,  
    'learning_rate': 0.5,  
    'max_depth': 8,  
    'min_child_weight': 8,  
    'n_estimators': 400,  
    'subsample': 0.8  
}
```

# XGBoost

## Résultats :



Rapport :

	precision	recall	f1-score	support
1.0	0.78	0.64	0.70	22
3.0	0.62	0.76	0.68	17
accuracy			0.69	39
macro avg	0.70	0.70	0.69	39
weighted avg	0.71	0.69	0.69	39

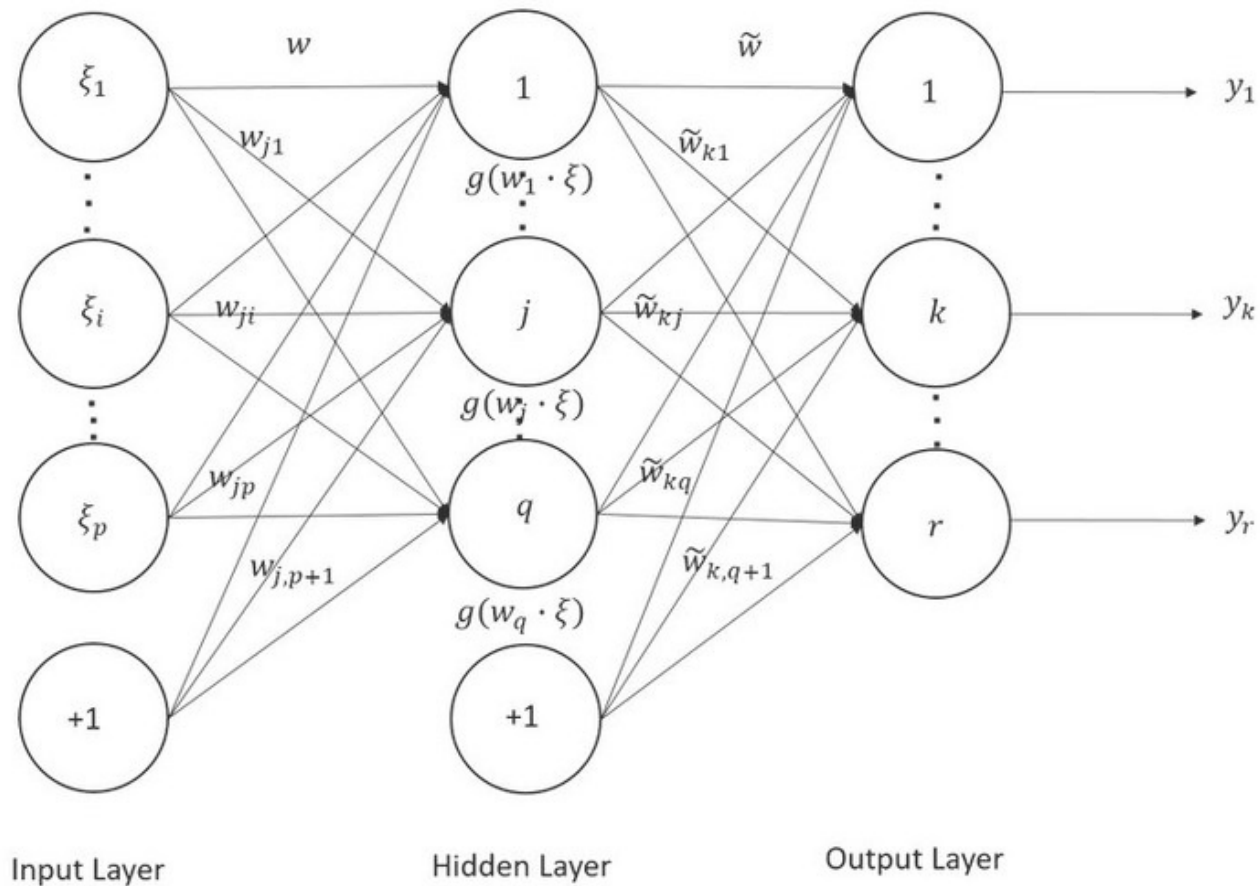
K-fold Cross Validation vaut en moyenne: 0.70

Les données du rapport sont les mêmes pour recherche random et best grille, mais le plus étonnant est que en réalisant plusieurs fois pour chaque set d'hyperparamètre tous alternent entre 0.67 et 0.73 pour le score de la CV mais il n'y a pas de grande différence .

Test effectués avec 5-folds, peut-être modifier le nombre de sous-groupe lors de la CV?



# Feed forward Neural Network Classifier



Paramètres de mon modèle:

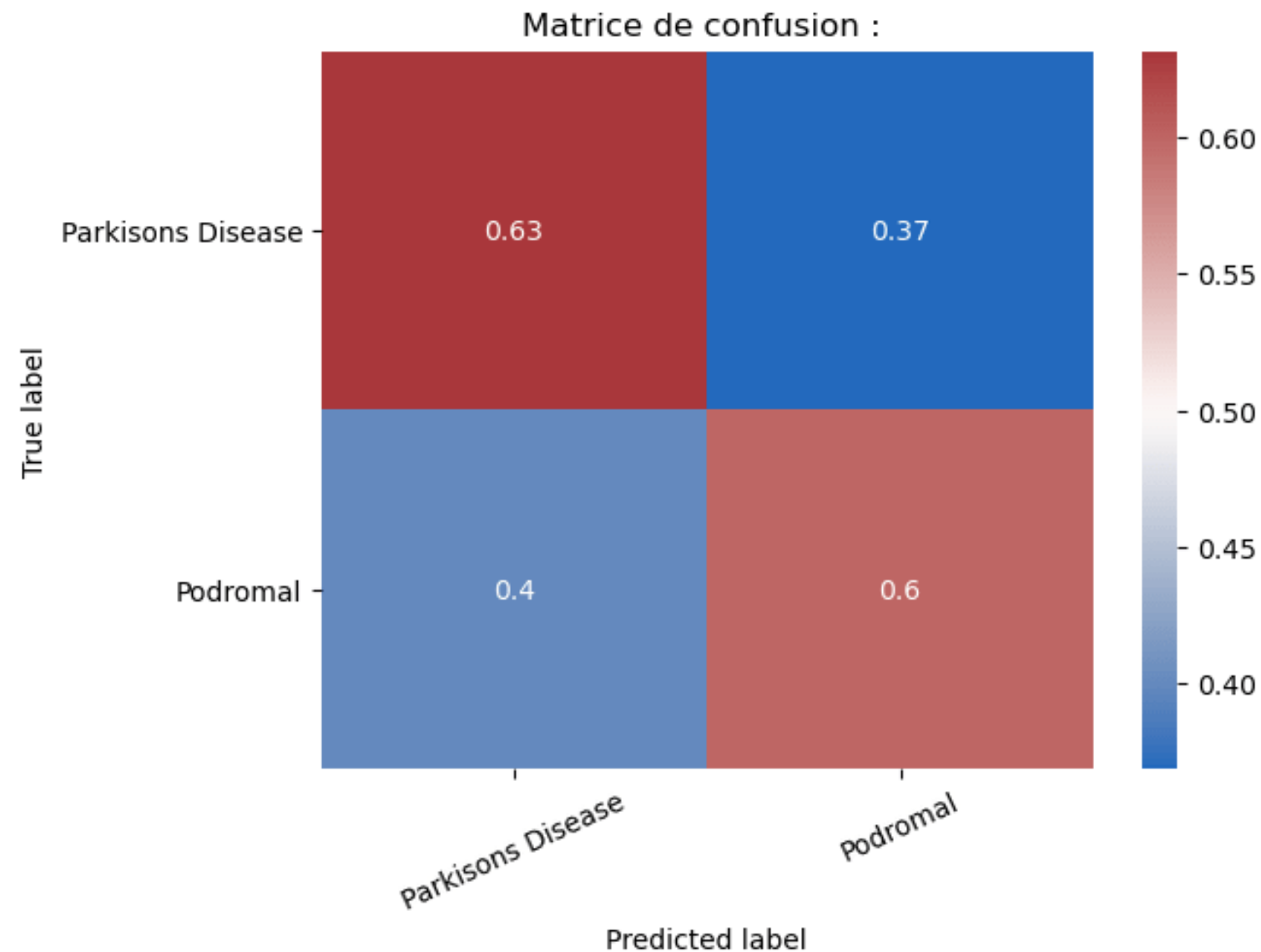
```
1 from tensorflow.keras.models import Sequential
2 from tensorflow.keras.layers import Dense
3 from tensorflow.keras.layers import Flatten
4
5 # Construction of Neural Network
6 model_nn = Sequential()
7 model_nn.add(Dense(100, input_dim = len(x_train_scaled[1]), activation='relu'))
8 model_nn.add(Dense(50, activation='relu'))
9 model_nn.add(Dense(1, activation='sigmoid'))
10 model_nn.summary()
11 model_nn.compile(loss='binary_crossentropy', optimizer='adam', metrics=['accuracy'])
```

Model: "sequential"

Layer (type)	Output Shape	Param #
dense (Dense)	(None, 100)	5700
dense_1 (Dense)	(None, 50)	5050
dense_2 (Dense)	(None, 1)	51
Total params: 10,801		
Trainable params: 10,801		
Non-trainable params: 0		

# Feed forward Neural Network Classifier

Fully connected Neural Network :  
Accuracy : 0.6153846153846154



Rapport :

	precision	recall	f1-score	support
0.0	0.60	0.63	0.62	19
1.0	0.63	0.60	0.62	20
accuracy			0.62	39
macro avg	0.62	0.62	0.62	39
weighted avg	0.62	0.62	0.62	39

# A venir:

- Interpretation des résultats plus précise
- Changer K de K-Fold CV pour XGBoost
- Comparer Random Forest / XGBoost / Feed-Forward NN