Machine Learning: Supervised

Exercice 3: Prédiction du gagnant d'un match de NBA

Pour prédire avec la plus grande certitude l'équipe gagnante d'un match de la NBA, nous disposons d'un dataset appelé "inputs.npy" qui contient les données d'un match à la mi-temps. Ces données ont été préalablement transformées pour correspondre à notre modèle. Nous disposons également d'un autre dataset appelé "labels.npy" qui contient les résultats de ces matchs, ce qui nous permettra de former notre intelligence artificielle (IA).

Tout d'abord nous allons diviser nos dataset afin d'avoir des échantillons tests, de la façon suivante :

- 90% sera consacré à l'entraînement (soit 450 lignes)
- 10% sera consacré aux tests (soit 50 lignes)

Maintenant nous allons pouvoir procéder aux test de plusieurs model différents et ajuster les hyperparamètres afin d'avoir la meilleur précision possible.

Logistic Regression

Dans le cas du modèle "Régression logistique", il y a 3 hyperparamètres que nous allons tester, car ils ont le plus grand impact sur les performances du modèle, tandis que les autres ont un impact négligeable dans notre situation. Voici la liste de ces hyperparamètres avec leurs options :

Solver : Optimisation du model selon le dataset

- Lbfgs : options de base, bonne performance et utilise peu de mémoire cependant produit parfois des erreurs de convergence
- Newton-cg : coûteux en calcul dû à l'utilisation de la matrice hessienne
- Sag : option la plus rapide pour de gros dataset avec un nombre de caractéristique et d'échantillons élevé
- Saga : option faite pour la régression logistique multinomial, également correct pour de gros dataset

- Penalty : Vise à réduire l'erreur de généralisation du modèle et minimise et réglemente le surajustement

	Solvers				
Penalties	**'liblinear'**	**'lbfgs'**	**'newton-cg'**	**'sag'**	**'saga'**
Multinomial + L2 penalty	no	yes	yes	yes	yes
OVR + L2 penalty	yes	yes	yes	yes	yes
Multinomial + L1 penalty	no	no	no	no	yes
OVR + L1 penalty	yes	no	no	no	yes
Elastic-Net	no	no	no	no	yes
No penalty ('none')	no	yes	yes	yes	yes
Behaviors					
Penalize the intercept (bad)	yes	no	no	no	no
Faster for large datasets	no	no	no	yes	yes
Robust to unscaled datasets	yes	yes	yes	no	no

- C : Lorsque l'option "Penalty" est activée pour réguler le surajustement, une valeur faible implique une minimisation intense, tandis qu'une valeur élevée accorde un poids important aux données d'apprentissage pour le modèle.

Pour obtenir la meilleure combinaison d'hyperparamètres possible, nous allons procéder simplement en créant une liste contenant toutes les combinaisons envisageables. Ensuite, nous obtiendrons un tableau récapitulant les performances de chaque combinaison, nous permettant ainsi d'identifier les hyperparamètres de la combinaison la plus performante.

Logisitic Regression model

	Train Accuracy	Test Accuracy	Precision	Recall	AUC
0	0.99111	0.90	0.90476	0.86364	0.89610
1	0.99111	0.90	0.90476	0.86364	0.89610
2	0.99111	0.90	0.90476	0.86364	0.89610
3	0.99111	0.90	0.90476	0.86364	0.89610
4	0.99111	0.90	0.90476	0.86364	0.89610
5	0.99111	0.90	0.90476	0.86364	0.89610
6	0.99111	0.90	0.90476	0.86364	0.89610
7	0.99111	0.90	0.90476	0.86364	0.89610
8	0.90444	0.82	0.80952	0.77273	0.81494
9	0.90444	0.82	0.80952	0.77273	0.81494
10	0.90444	0.82	0.80952	0.77273	0.81494
11	0.90222	0.82	0.80952	0.77273	0.81494
12	0.90444	0.82	0.80952	0.77273	0.81494
13	0.90444	0.82	0.80952	0.77273	0.81494
14	0.90444	0.82	0.80952	0.77273	0.81494
15	0.90222	0.82	0.80952	0.77273	0.81494

The best hyperparameter are {'solver': 'newton-cg', 'penalty': 'l2', 'max_iter': 1000}

Tableau de performances des combinaisons d'hyperparamètre

Support Vector Machine (SVM)

Dans le cas du modèle SVM (Support Vector Machine), nous allons nous pencher sur 3 hyperparamètres qui ont le plus d'impact sur le modèle dans notre situation, tandis que les autres hyperparamètres auront un impact négligeable. Voici la liste de ces 3 hyperparamètres :

Kernel : Transforme le dataset en augmentant la dimension pour le séparé linéairement

- Rbf : Option par défaut du Kernel
- Poly : Option cherchant a combiner les caractéristiques en plus de chercher leur similarité

- C : Régularise le surajustement selon la valeur affecté, une valeur faible impliquera une forte minimisation du surajustement tandis qu'une valeur forte laissera aux données un poids plus important dans l'apprentissage
- Gamma : Détermine le coefficient kernel, si la valeur est faible le rayon d'influence du vecteur support est élevé et inversement

Afin d'obtenir la meilleur combinaison d'hyperparamètre, nous allons cette fois utilisé la méthode « Grid Search », qui consiste à définir les valeurs des hyperparamètres dans une grid puis, par la suite de tester toutes les combinaisons depuis cette grid.

```
param_grid = {
    "C": c_range,
    "kernel": ['rbf', 'poly'],
    "gamma": gamma_range.tolist()+['scale', 'auto']
}

scoring = ['accuracy']

kfold = StratifiedKFold(n_splits=3, shuffle=True, random_state=0)

grid_search = GridSearchCV(estimator=SVC(),
                           param_grid=param_grid,
                           scoring=scoring,
                           refit='accuracy',
                           n_jobs=1,
                           cv=kfold,
                           verbose=0)

grid_result = grid_search.fit(x_train, y_train)
```

Source:

Model:
https://moncoachdata.com/blog/modeles-de-machine-learning-expliques/

LR hyperparamètre:
https://medium.com/codex/do-i-need-to-tune-logistic-regression-hyperparameters-1cb2b81fca69

SVM:
https://www.mltut.com/svm-implementation-in-python-from-scratch/

SVM hyperparamètre :
https://medium.com/grabngoinfo/support-vector-machine-svm-hyperparameter-tuning-in-python-a65586289bcb