PROJET N°8

Toni Adriano

TABLE DES MATIERES

TABLE DES FIGURES

LA THEMATIQUE

Pour le projet n°8, j'ai décidé de reprendre un projet personnel, celui qui m'a amené à la data science. Il s'agit d'un travail concernant les paris sportifs, plus particulièrement les paris sur les matchs de football européens. La technique utilisée pourra être très facilement élargie plus tard à d'autres continents ou d'autres sports.

Je souhaite créer une méthode qui me permettra de parier plus efficacement sur n'importe quel match de football européen. Pour la créer, je vais m'appuyer sur les 15 dernières années de résultats pour entraîner des algorithmes de classification.

ETAT DE L'ART

Pour déterminer l'état de l'art, je me suis appuyé sur deux articles :

- Pattern Detection Applied to Soccer Results Forecast, **Diogo Reis**, Janvier 2018. [1]
- "Quality vs Quantity": Improved Shot Prediction in Soccer using Strategic Features from Spatiotemporal Data par **Patrick Lucey, Alina Bialkowski, Mathew Monfort, Peter Carr and Iain Matthews**, 2015 [2]

Il faut noter que ce projet est le premier pas pour l'application numérique décrite dans l'article [1].

En effet, cet article, écrit par Diogo Reis, est basé sur nos réflexions communes à nous deux sur cette thématique. L'objectif final étant de pouvoir écrire un deuxième article visant à présenter les conclusions de notre étude.

PREMIER ARTICLE

Cet article présente le fonctionnement des paris sportifs sur les matchs de football.

Il présente également le fonctionnement global du *machine learning*, dont je reprends ici les grandes lignes :

Machine learning is a field of study that includes knowledge of statistics, computer science and domain knowledge. The machine learning programming consists of learning from data itself, in opposition to the traditional rule-based programming.

Machine learning is a field of computer science that gives computers the ability to learn without being explicitly programmed. Machine learning can be divided into two large groups, supervised and unsupervised learning. The prediction of the sports result Home Win, Draw, Away Win is inserted in the category of supervised learning, where the model will be fed previously with a set of features related to a soccer game and labelled with the outcome of that same game, in order to predict a categorial data class. The big challenge is to create a model that can learn from the input dataset with high prediction accuracy and at the same time prevent overfitting. This will allow that this model can be applied to unseen data.

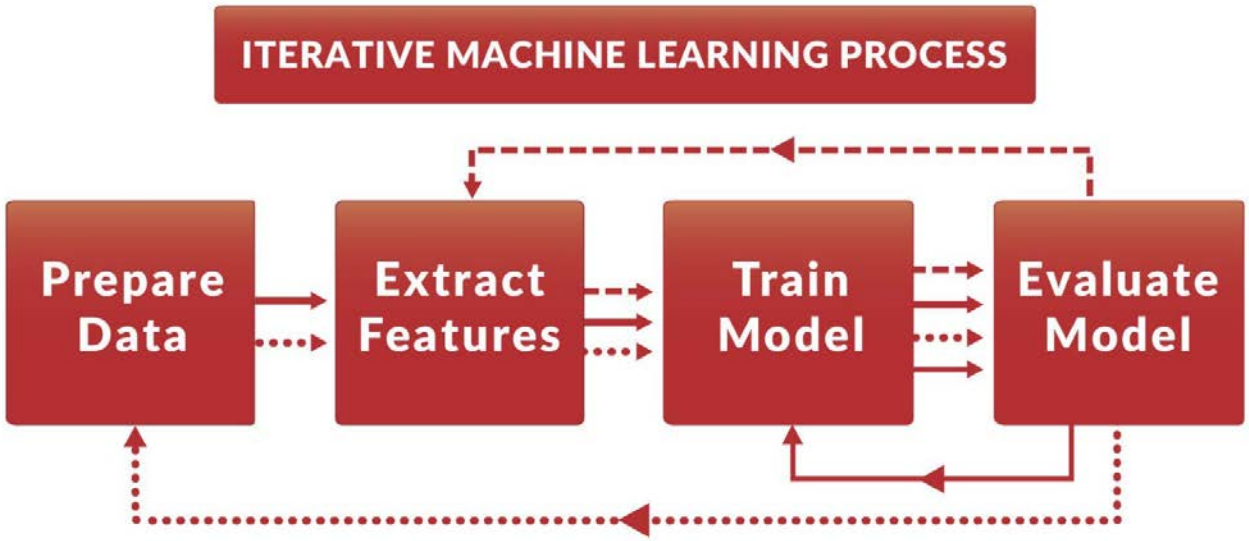

Figure 1 : Processus de Machine Learning

DEUXIEME ARTICLE

Ce papier analyse les données dans l'espace de jeu et des joueurs. On y trouve également des références de contexte tels que la date et l'heure de match. Enfin, il analyse les confrontations entre la « défense » et l'« attaque » des équipes qui participent à chaque match.

L'objectif de cet article est d'essayer de comprendre si le résultat final du match est lié (de près ou de loin) aux caractéristiques du match telles que le nombre de tir, de passes, de cartons, etc. ou si ce résultat n'est pas absolument pas lié à ces données.

Une régression logistique est utilisée avec quelques-uns de ces données et il est par la suite prouvé qu'une quantité de données plus importante en entrée réduit les erreurs d'estimations des résultats finaux.

De cet article, je retire l'idée que les statistiques des matches précédents vont être importantes et que je dois les inclure dans la recherche.

LE JEU DE DONNEES CHOISI

Un dataset est construit à partir des résultats stockés sur un site de référence : http://www.football-data.co.uk.

Ces données sont les plus complètes que l'on peut trouver gratuitement aujourd'hui. Éventuellement, après cette première étape, des données payantes mais beaucoup plus complètes pourront être utilisées si cela s'avère nécessaire et si cette étape s'avère concluante.

Les données fournies par ce site le sont de manière brute. Une grande partie de data engeneering est nécessaire afin de pouvoir créer un dataset avec les données exploitables.

LISTE DES DONNEES BRUTES UTILISEES

Div = Championnat
Date = Date du match
HomeTeam = Equipe à domicile
AwayTeam = Equipe à l'extérieur
F T HG = Nombre de buts de l'équipe à domicile
F T AG = Nombre de buts de l'équipe à l'extérieur
F T R = Résultat du match
HS = Nombre de tirs de l'équipe à domicile
AS = Nombre de tis de l'équipe à l'extérieur
HST = Nombre de tirs cadrés de l'équipe à domicile
AST = Nombre de tirs cadrés de l'équipe à l'extérieur
HC = Corners en faveur de l'équipe à domicile
AC = Corners en faveur de l'équipe à l'extérieur
HF = Fautes commises par l'équipe à domicile
AF = Fautes commises par l'équipe à l'extérieur
HY = Nombre de cartons jaune reçu par l'équipe à domicile
AY = Nombre de cartons jaune reçu par l'équipe à l'extérieur
HR = Nombre de cartons rouge reçu par l'équipe à domicile
AR = Nombre de cartons rouge reçu par l'équipe à l'extérieur
BbAvH = Mise moyenne en faveur de l'équipe à domicile
BbAvD = Mise moyenne en faveur du match nul
BbAvA = Mise moyenne en faveur de l'équipe à l'extérieur

LA METHODE IMPLEMENTEE ET SES PERFORMANCES EN COMPARAISON AVEC LA METHODE BASELINE.

L'architecture normale d'un projet de machine learning a été adoptée.

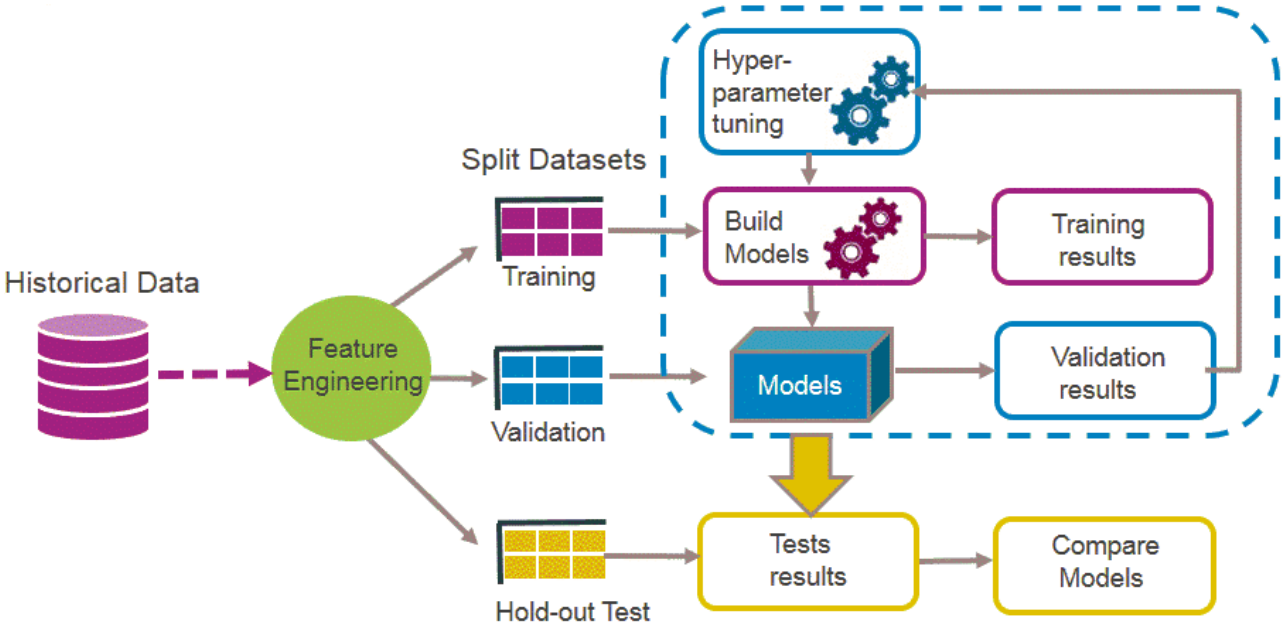

Figure 2 : Architecture retenue

Historical Data : Récupération du dataset sur le site internet de référence

Feature Engineering : Etapes de feature engineering pour créer de nouvelles variables

Split Datasets : Séparation des dataset en train/validation/test avec une validation et optimisation faites sous forme de cross-validation

Model Creation : Création du modèle final en évitant un overfitting dû à la taille réduite du dataset.

Evaluation : Résultats finaux.

En pratique, la méthode implémentée est divisée en deux fichiers :

- Création du dataset
- Recherche du meilleur algorithme possible

HISTORICAL DATA

Toutes les données sont stockées, gratuitement, sur un site de référence : http://www.football-data.co.uk.

Avec un script de type webgrab, il est possible de les récupérer facilement sous format csv. Ce format est parfait pour une exploitation en langage python.

FEATURE ENGINEERING

Le feature engineering est divisé en deux parties :

- Création du dataset qui sera utilisé dans le projet.
- Création des nouvelles variables à partir de ce dataset.

Ces deux parties sont regroupées dans le fichier n_coeff_dataset.py.

CREATION DU DATASET

La création du dataset requiert les étapes décrites ci-dessous :

- Récupération du dataset complet sur le site internet de référence
- Choix des données qui sont conservées. Ceci est fait en sélectionnant les colonnes du dataset initial qui nous intéressent
- Suppression des données qui ne sont pas conservées

CREATION DE NOUVELLES VARIABLES

La création de nouvelles variables requiert les étapes décrites ci-dessous :

- Création d'une nouvelle famille de données de « forme ». Cette famille agrège les données sur les 5 derniers matchs pour une équipe donnée pour tous les matchs disponibles. Par exemple, pour une équipe A qui jouera un match lors de la journée n°30, les données des journées 29, 28, 27, 26 et 25 seront utilisées
- Création d'une donnée dénommée « Expected Goals ». Cette donnée est dérivée également des 5 derniers matchs
- Enfin, on regroupe toute ces données dans un nouveau dataset qui servira de référence.

SPLIT DATASETS

La bibliothèque sklearn pour le langage python permet une séparation rapide et effiace d'un dataset global en deux datasets d'entrainement et de test.

La méthode est la suivante : *sklearn.model_selection.train_test_split*

MODEL CREATION

Cette étape est la principale composante du fichier n_coeff4.py.

Un certain nombre d'algorithmes de classification est utilisé et confronté. La validation est faite sous forme de cross validation. La liste des hyperparamètres est également indiquée.

- RandomForestClassifier
  - 'max_depth' : [None, 20, 35, 50, 65, 80, 110, 200]
  - 'n_estimators' : [5, 20, 35, 50, 65, 80, 110, 200]
- GradientBoostingClassifier
  - 'max_depth' : [None, 10, 20, 40, 70, 90, 120, 150]
  - 'n_estimators' : [5, 20, 35, 50, 65, 80, 95, 110]
- LogisticRegression
  - 'penalty' : ['l1'],
  - 'C' : [0.001, 0.01, 0.1, 1, 10, 100, 1000]
  - 'class_weight' : [None, 'balanced']
  - 'solver' : ['liblinear', 'saga']
- LogisticRegression
  - 'penalty' : ['l2']
  - 'C' : [0.001, 0.01, 0.1, 1, 10, 100, 1000]
  - 'class_weight' : [None, 'balanced']
  - 'solver': ['newton-cg', 'lbfgs', 'sag']

- MultinomialNB
  - 'alpha' : [0.001, 0.01, 0.1, 1, 10, 100, 1000]
  - 'fit_prior' : [True, False]

EVALUATION

Les deux métriques importantes sont le gain final et l'accuracy des pronostics des résultats.

Les deux sont important car on pourrait facilement augmenter l'accuracy au détriment du gain (en ne pariant que sur les favoris, qui rapportent peu) et de même, on peut augmenter le gain au détriment de l'accuracy (en ne pariant que sur les underdogs, qui rapportent beaucoup mais ne gagnent que peu souvent).

L'objectif de n'importe quel parieur est d'obtenir un gain élevé. On peut donc penser que cette métrique sera le plus important à optimiser.

ACCURACY

L'accuracy représente le pourcentage des bons pronostics qui auront été fait par les algorithmes qui sont testés.

$$Accuracy = \frac{\#Correct}{\#Prédictions}$$

GAIN

Valeur d'une mise unique :

$$Mise = valeur\ constante = 1$$

En cas de bon pronostic :

$$Gain\ positif = Mise * (Cote - 1)$$

En cas de mauvais pronostic :

$$Gain\ négatif = -Mise$$

Gain final total :

$$Gain\ total = \sum Gain\ positif + \sum Gain\ négatif$$

ANALYSE DES RESULTATS

Les résultats présentés ici sont limités à un championnat pour des raisons de simplicité. Ils sont obtenus grâce à une cross validation testé sur plusieurs algorithmes de classification. Toutes les étapes de cette cross validation peuvent être consultées dans l'Annexe : Résultats de la cross validation.

On retrouve les deux métriques définis dans le chapitre ci-dessus.

Les graphiques dans lesquels les deux métriques sont inscrites sont également montrés avec :

- En bleu le gain en pourcentage.
- En rouge, l'accuracy.

BASELINE

Voici le tableau récapitulatif de différente baselines étudiées.

	Gain (%)	Accuracy (%)
Parier sur les favoris	-5,8	49,2
Parier sur les underdogs	-8,5	23,1
Parier sur la victoire à domicile	-13,5	42,2
Parier sur la victoire à l'extérieur	-6,1	28
Parier sur les matchs nuls	-5,9	29

Figure 3 : Tableau des résultats « Baseline »

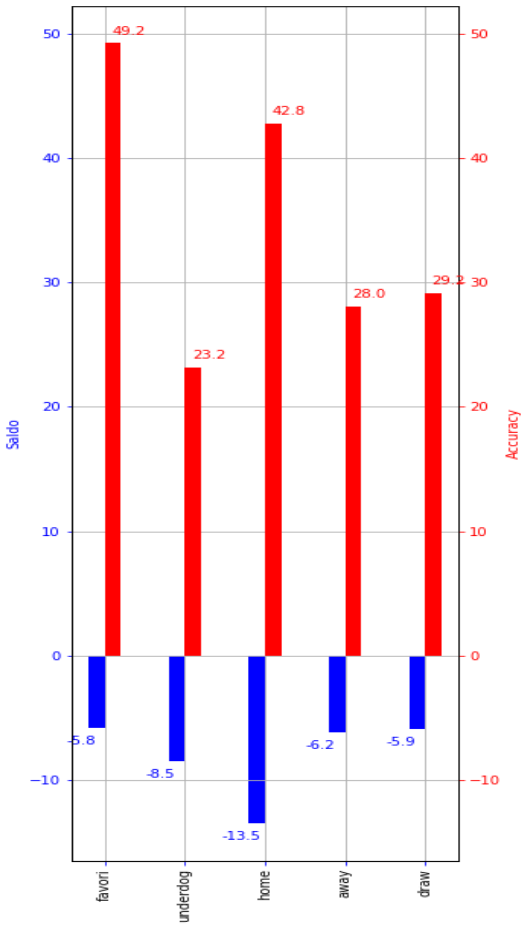

Figure 4 : Graphique des résultats « Baseline »

L'accuracy est souvent très mauvaise, et les résultats en gain sont toujours négatif. Le meilleur « gain » que l'on obtient est à -5,8 %.

SUR LE DATASET

Voici les résultats après améliorations via la recherche des meilleurs hyperparamètres.

	Gain total	Accuracy	Paramètres
RandomForestClassifier	-1,12	51,08	{'n_neighbors': 50}
GradientBoostingClassifier	-0,7	53,23	{'max_depth': None, 'n_estimators': 50}
RegressionLogistic (l1)	-0,51	50,8	{'C': 1, 'class_weight': 'balanced', 'penalty': 'l1', 'solver': 'saga'}
RegressionLogistic (l2)	-0,31	50,8	{'C': 0.1, 'class_weight': 'balanced', 'penalty': 'l2', 'solver': 'saga'}
MultinomialNB	0,99	50,17	{'alpha': 10, 'fit_prior': False}

Figure 5 : Tableau des résultats

On remarquera que les résultats pour le gain sont meilleurs que pour toutes les baselines. Le résultat pour le MultinomialNB est positif ce qui est très motivant à continuer cette recherche.

Cependant ces premiers résultats sont donc très encourageants car ils sont meilleurs que ceux des baselines.

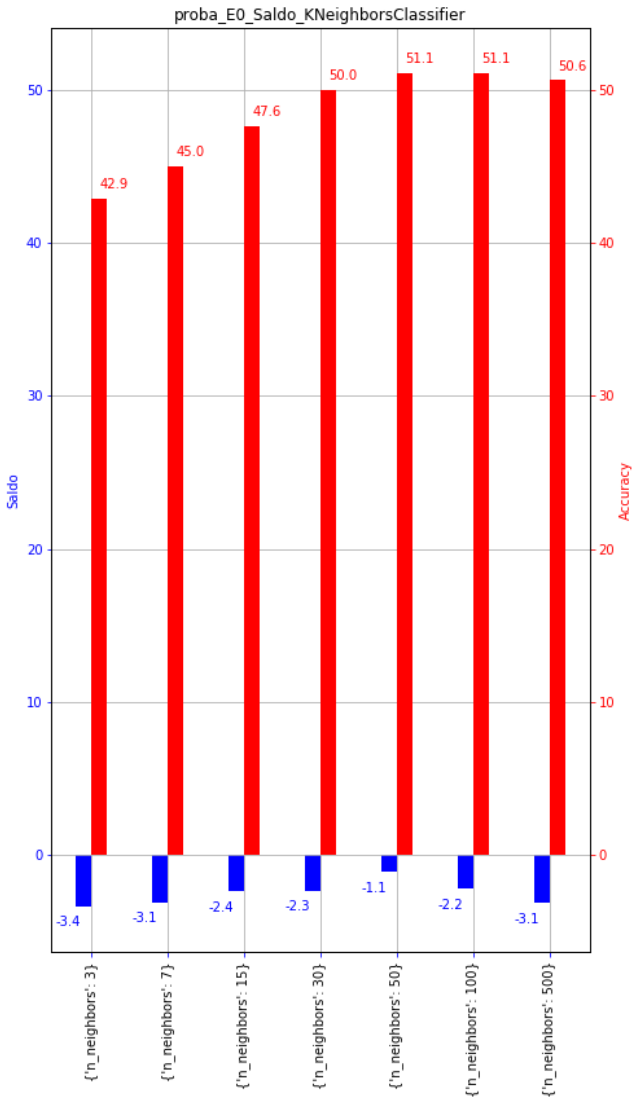

Figure 6 : Exemple de résultat pour le Knn

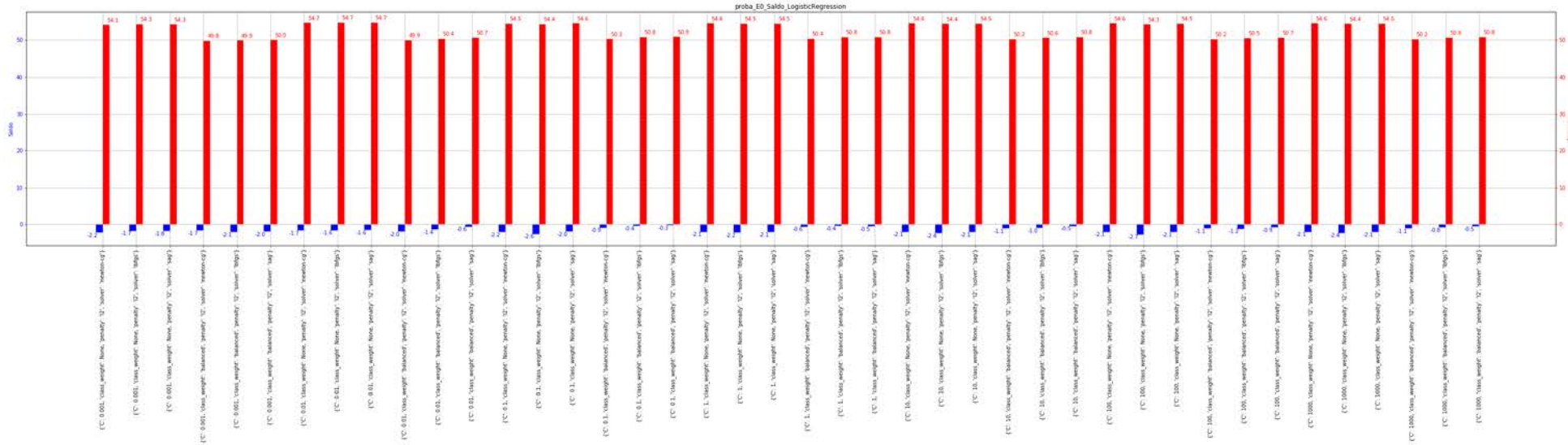

Figure 7 : Exemple de résultat pour le Logistic Regression

ANNEXE : RESULTATS DE LA CROSS VALIDATION.

KNeighborsClassifier

Gain : -3.37 Accur : 42.93 pour {'n_neighbors': 3}
Gain : -3.11 Accur : 45.01 pour {'n_neighbors': 7}
Gain : -2.4 Accur : 47.62 pour {'n_neighbors': 15}
Gain : -2.33 Accur : 49.97 pour {'n_neighbors': 30}
Gain : -1.12 Accur : 51.08 pour {'n_neighbors': 50}
Gain : -2.23 Accur : 51.08 pour {'n_neighbors': 100}
Gain : -3.15 Accur : 50.64 pour {'n_neighbors': 500}

Best Gain : -1.12 Best Accur : 51.08 pour {'n_neighbors': 50}

RandomForestClassifier

Gain : -4.34 Accur : 46.45 pour {'max_depth': None, 'n_estimators': 5}
Gain : -5.51 Accur : 50.44 pour {'max_depth': None, 'n_estimators': 20}
Gain : -3.21 Accur : 51.97 pour {'max_depth': None, 'n_estimators': 35}
Gain : -0.7 Accur : 53.24 pour {'max_depth': None, 'n_estimators': 50}
Gain : -3.39 Accur : 52.61 pour {'max_depth': None, 'n_estimators': 65}
Gain : -2.72 Accur : 53.02 pour {'max_depth': None, 'n_estimators': 80}
Gain : -2.74 Accur : 53.24 pour {'max_depth': None, 'n_estimators': 110}
Gain : -2.68 Accur : 53.44 pour {'max_depth': None, 'n_estimators': 200}
Gain : -5.03 Accur : 46.56 pour {'max_depth': 20, 'n_estimators': 5}
Gain : -2.75 Accur : 51.47 pour {'max_depth': 20, 'n_estimators': 20}
Gain : -1.33 Accur : 53.02 pour {'max_depth': 20, 'n_estimators': 35}
Gain : -4.76 Accur : 52.08 pour {'max_depth': 20, 'n_estimators': 50}
Gain : -0.73 Accur : 53.49 pour {'max_depth': 20, 'n_estimators': 65}
Gain : -4.19 Accur : 52.44 pour {'max_depth': 20, 'n_estimators': 80}
Gain : -3.72 Accur : 52.88 pour {'max_depth': 20, 'n_estimators': 110}
Gain : -3.96 Accur : 53.0 pour {'max_depth': 20, 'n_estimators': 200}
Gain : -7.39 Accur : 45.2 pour {'max_depth': 35, 'n_estimators': 5}
Gain : -5.37 Accur : 50.36 pour {'max_depth': 35, 'n_estimators': 20}
Gain : -1.17 Accur : 52.8 pour {'max_depth': 35, 'n_estimators': 35}
Gain : -1.86 Accur : 52.72 pour {'max_depth': 35, 'n_estimators': 50}
Gain : -1.93 Accur : 53.27 pour {'max_depth': 35, 'n_estimators': 65}
Gain : -3.24 Accur : 52.91 pour {'max_depth': 35, 'n_estimators': 80}
Gain : -4.48 Accur : 52.5 pour {'max_depth': 35, 'n_estimators': 110}
Gain : -4.39 Accur : 52.66 pour {'max_depth': 35, 'n_estimators': 200}
Gain : -4.98 Accur : 46.2 pour {'max_depth': 50, 'n_estimators': 5}
Gain : -4.97 Accur : 50.25 pour {'max_depth': 50, 'n_estimators': 20}
Gain : -4.54 Accur : 51.5 pour {'max_depth': 50, 'n_estimators': 35}
Gain : -5.32 Accur : 51.64 pour {'max_depth': 50, 'n_estimators': 50}
Gain : -3.05 Accur : 52.91 pour {'max_depth': 50, 'n_estimators': 65}
Gain : -3.14 Accur : 52.86 pour {'max_depth': 50, 'n_estimators': 80}
Gain : -3.15 Accur : 52.88 pour {'max_depth': 50, 'n_estimators': 110}
Gain : -2.53 Accur : 53.47 pour {'max_depth': 50, 'n_estimators': 200}
Gain : -4.87 Accur : 46.28 pour {'max_depth': 65, 'n_estimators': 5}
Gain : -4.29 Accur : 50.97 pour {'max_depth': 65, 'n_estimators': 20}
Gain : -4.25 Accur : 51.72 pour {'max_depth': 65, 'n_estimators': 35}
Gain : -3.98 Accur : 52.14 pour {'max_depth': 65, 'n_estimators': 50}
Gain : -3.48 Accur : 52.58 pour {'max_depth': 65, 'n_estimators': 65}
Gain : -5.3 Accur : 51.97 pour {'max_depth': 65, 'n_estimators': 80}
Gain : -2.96 Accur : 53.11 pour {'max_depth': 65, 'n_estimators': 110}
Gain : -4.38 Accur : 52.77 pour {'max_depth': 65, 'n_estimators': 200}
Gain : -8.85 Accur : 44.76 pour {'max_depth': 80, 'n_estimators': 5}
Gain : -3.52 Accur : 50.69 pour {'max_depth': 80, 'n_estimators': 20}
Gain : -3.21 Accur : 52.16 pour {'max_depth': 80, 'n_estimators': 35}
Gain : -4.55 Accur : 52.14 pour {'max_depth': 80, 'n_estimators': 50}
Gain : -4.27 Accur : 52.25 pour {'max_depth': 80, 'n_estimators': 65}

Gain : -5.24 Accur : 51.97 pour {'max_depth': 80, 'n_estimators': 80}
Gain : -3.53 Accur : 52.8 pour {'max_depth': 80, 'n_estimators': 110}
Gain : -3.88 Accur : 52.94 pour {'max_depth': 80, 'n_estimators': 200}
Gain : -3.14 Accur : 46.62 pour {'max_depth': 110, 'n_estimators': 5}
Gain : -5.34 Accur : 50.17 pour {'max_depth': 110, 'n_estimators': 20}
Gain : -4.61 Accur : 51.36 pour {'max_depth': 110, 'n_estimators': 35}
Gain : -4.46 Accur : 52.02 pour {'max_depth': 110, 'n_estimators': 50}
Gain : -2.25 Accur : 52.97 pour {'max_depth': 110, 'n_estimators': 65}
Gain : -3.83 Accur : 52.63 pour {'max_depth': 110, 'n_estimators': 80}
Gain : -3.11 Accur : 53.16 pour {'max_depth': 110, 'n_estimators': 110}
Gain : -3.01 Accur : 53.24 pour {'max_depth': 110, 'n_estimators': 200}
Gain : -5.13 Accur : 45.81 pour {'max_depth': 200, 'n_estimators': 5}
Gain : -3.36 Accur : 51.25 pour {'max_depth': 200, 'n_estimators': 20}
Gain : -2.93 Accur : 52.02 pour {'max_depth': 200, 'n_estimators': 35}
Gain : -2.65 Accur : 52.61 pour {'max_depth': 200, 'n_estimators': 50}
Gain : -2.97 Accur : 52.58 pour {'max_depth': 200, 'n_estimators': 65}
Gain : -5.03 Accur : 52.14 pour {'max_depth': 200, 'n_estimators': 80}
Gain : -1.56 Accur : 53.55 pour {'max_depth': 200, 'n_estimators': 110}
Gain : -2.65 Accur : 53.27 pour {'max_depth': 200, 'n_estimators': 200}

Best Gain : -0.7 Best Accur : 53.24 pour {'max_depth': None, 'n_estimators': 50}

LogisticRegression
Gain : -2.85 Accur : 49.58 pour {'C': 0.001, 'class_weight': None, 'penalty': 'l1', 'solver': 'liblinear'}
Gain : -2.89 Accur : 49.58 pour {'C': 0.001, 'class_weight': None, 'penalty': 'l1', 'solver': 'saga'}
Gain : -3.2 Accur : 49.33 pour {'C': 0.001, 'class_weight': 'balanced', 'penalty': 'l1', 'solver': 'liblinear'}
Gain : -1.91 Accur : 47.95 pour {'C': 0.001, 'class_weight': 'balanced', 'penalty': 'l1', 'solver': 'saga'}
Gain : -1.53 Accur : 54.8 pour {'C': 0.01, 'class_weight': None, 'penalty': 'l1', 'solver': 'liblinear'}
Gain : -1.53 Accur : 54.8 pour {'C': 0.01, 'class_weight': None, 'penalty': 'l1', 'solver': 'saga'}
Gain : -1.61 Accur : 53.88 pour {'C': 0.01, 'class_weight': 'balanced', 'penalty': 'l1', 'solver': 'liblinear'}
Gain : -1.71 Accur : 51.0 pour {'C': 0.01, 'class_weight': 'balanced', 'penalty': 'l1', 'solver': 'saga'}
Gain : -1.84 Accur : 54.74 pour {'C': 0.1, 'class_weight': None, 'penalty': 'l1', 'solver': 'liblinear'}
Gain : -1.9 Accur : 54.71 pour {'C': 0.1, 'class_weight': None, 'penalty': 'l1', 'solver': 'saga'}
Gain : -2.7 Accur : 52.86 pour {'C': 0.1, 'class_weight': 'balanced', 'penalty': 'l1', 'solver': 'liblinear'}
Gain : -1.73 Accur : 50.42 pour {'C': 0.1, 'class_weight': 'balanced', 'penalty': 'l1', 'solver': 'saga'}
Gain : -2.36 Accur : 54.46 pour {'C': 1, 'class_weight': None, 'penalty': 'l1', 'solver': 'liblinear'}
Gain : -2.2 Accur : 54.49 pour {'C': 1, 'class_weight': None, 'penalty': 'l1', 'solver': 'saga'}
Gain : -1.09 Accur : 53.02 pour {'C': 1, 'class_weight': 'balanced', 'penalty': 'l1', 'solver': 'liblinear'}
Gain : -0.51 Accur : 50.8 pour {'C': 1, 'class_weight': 'balanced', 'penalty': 'l1', 'solver': 'saga'}
Gain : -2.49 Accur : 54.41 pour {'C': 10, 'class_weight': None, 'penalty': 'l1', 'solver': 'liblinear'}
Gain : -2.02 Accur : 54.55 pour {'C': 10, 'class_weight': None, 'penalty': 'l1', 'solver': 'saga'}
Gain : -2.24 Accur : 52.55 pour {'C': 10, 'class_weight': 'balanced', 'penalty': 'l1', 'solver': 'liblinear'}
Gain : -0.55 Accur : 50.8 pour {'C': 10, 'class_weight': 'balanced', 'penalty': 'l1', 'solver': 'saga'}
Gain : -2.28 Accur : 54.49 pour {'C': 100, 'class_weight': None, 'penalty': 'l1', 'solver': 'liblinear'}
Gain : -2.09 Accur : 54.52 pour {'C': 100, 'class_weight': None, 'penalty': 'l1', 'solver': 'saga'}
Gain : -1.8 Accur : 52.66 pour {'C': 100, 'class_weight': 'balanced', 'penalty': 'l1', 'solver': 'liblinear'}
Gain : -0.64 Accur : 50.78 pour {'C': 100, 'class_weight': 'balanced', 'penalty': 'l1', 'solver': 'saga'}
Gain : -2.19 Accur : 54.52 pour {'C': 1000, 'class_weight': None, 'penalty': 'l1', 'solver': 'liblinear'}
Gain : -2.09 Accur : 54.52 pour {'C': 1000, 'class_weight': None, 'penalty': 'l1', 'solver': 'saga'}
Gain : -1.92 Accur : 52.63 pour {'C': 1000, 'class_weight': 'balanced', 'penalty': 'l1', 'solver': 'liblinear'}
Gain : -0.64 Accur : 50.78 pour {'C': 1000, 'class_weight': 'balanced', 'penalty': 'l1', 'solver': 'saga'}

Best Gain : -0.51 Best Accur : 50.8 pour {'C': 1, 'class_weight': 'balanced', 'penalty': 'l1', 'solver': 'saga'}

LogisticRegression
Gain : -2.24 Accur : 54.13 pour {'C': 0.001, 'class_weight': None, 'penalty': 'l2', 'solver': 'newton-cg'}
Gain : -1.74 Accur : 54.33 pour {'C': 0.001, 'class_weight': None, 'penalty': 'l2', 'solver': 'lbfgs'}
Gain : -1.8 Accur : 54.3 pour {'C': 0.001, 'class_weight': None, 'penalty': 'l2', 'solver': 'sag'}
Gain : -1.68 Accur : 49.83 pour {'C': 0.001, 'class_weight': 'balanced', 'penalty': 'l2', 'solver': 'newton-cg'}

Gain : -2.12 Accur : 49.86 pour {'C': 0.001, 'class_weight': 'balanced', 'penalty': 'l2', 'solver': 'lbfgs'}
Gain : -1.96 Accur : 50.0 pour {'C': 0.001, 'class_weight': 'balanced', 'penalty': 'l2', 'solver': 'sag'}
Gain : -1.72 Accur : 54.69 pour {'C': 0.01, 'class_weight': None, 'penalty': 'l2', 'solver': 'newton-cg'}
Gain : -1.62 Accur : 54.74 pour {'C': 0.01, 'class_weight': None, 'penalty': 'l2', 'solver': 'lbfgs'}
Gain : -1.56 Accur : 54.74 pour {'C': 0.01, 'class_weight': None, 'penalty': 'l2', 'solver': 'sag'}
Gain : -1.98 Accur : 49.94 pour {'C': 0.01, 'class_weight': 'balanced', 'penalty': 'l2', 'solver': 'newton-cg'}
Gain : -1.41 Accur : 50.36 pour {'C': 0.01, 'class_weight': 'balanced', 'penalty': 'l2', 'solver': 'lbfgs'}
Gain : -0.62 Accur : 50.69 pour {'C': 0.01, 'class_weight': 'balanced', 'penalty': 'l2', 'solver': 'sag'}
Gain : -2.15 Accur : 54.52 pour {'C': 0.1, 'class_weight': None, 'penalty': 'l2', 'solver': 'newton-cg'}
Gain : -2.59 Accur : 54.35 pour {'C': 0.1, 'class_weight': None, 'penalty': 'l2', 'solver': 'lbfgs'}
Gain : -1.95 Accur : 54.58 pour {'C': 0.1, 'class_weight': None, 'penalty': 'l2', 'solver': 'sag'}
Gain : -0.94 Accur : 50.31 pour {'C': 0.1, 'class_weight': 'balanced', 'penalty': 'l2', 'solver': 'newton-cg'}
Gain : -0.42 Accur : 50.83 pour {'C': 0.1, 'class_weight': 'balanced', 'penalty': 'l2', 'solver': 'lbfgs'}
Gain : -0.31 Accur : 50.86 pour {'C': 0.1, 'class_weight': 'balanced', 'penalty': 'l2', 'solver': 'sag'}
Gain : -2.05 Accur : 54.58 pour {'C': 1, 'class_weight': None, 'penalty': 'l2', 'solver': 'newton-cg'}
Gain : -2.25 Accur : 54.49 pour {'C': 1, 'class_weight': None, 'penalty': 'l2', 'solver': 'lbfgs'}
Gain : -2.09 Accur : 54.52 pour {'C': 1, 'class_weight': None, 'penalty': 'l2', 'solver': 'sag'}
Gain : -0.63 Accur : 50.39 pour {'C': 1, 'class_weight': 'balanced', 'penalty': 'l2', 'solver': 'newton-cg'}
Gain : -0.44 Accur : 50.8 pour {'C': 1, 'class_weight': 'balanced', 'penalty': 'l2', 'solver': 'lbfgs'}
Gain : -0.54 Accur : 50.8 pour {'C': 1, 'class_weight': 'balanced', 'penalty': 'l2', 'solver': 'sag'}
Gain : -2.06 Accur : 54.58 pour {'C': 10, 'class_weight': None, 'penalty': 'l2', 'solver': 'newton-cg'}
Gain : -2.41 Accur : 54.44 pour {'C': 10, 'class_weight': None, 'penalty': 'l2', 'solver': 'lbfgs'}
Gain : -2.09 Accur : 54.52 pour {'C': 10, 'class_weight': None, 'penalty': 'l2', 'solver': 'sag'}
Gain : -1.09 Accur : 50.22 pour {'C': 10, 'class_weight': 'balanced', 'penalty': 'l2', 'solver': 'newton-cg'}
Gain : -0.96 Accur : 50.58 pour {'C': 10, 'class_weight': 'balanced', 'penalty': 'l2', 'solver': 'lbfgs'}
Gain : -0.48 Accur : 50.83 pour {'C': 10, 'class_weight': 'balanced', 'penalty': 'l2', 'solver': 'sag'}
Gain : -2.06 Accur : 54.58 pour {'C': 100, 'class_weight': None, 'penalty': 'l2', 'solver': 'newton-cg'}
Gain : -2.73 Accur : 54.3 pour {'C': 100, 'class_weight': None, 'penalty': 'l2', 'solver': 'lbfgs'}
Gain : -2.09 Accur : 54.52 pour {'C': 100, 'class_weight': None, 'penalty': 'l2', 'solver': 'sag'}
Gain : -1.09 Accur : 50.22 pour {'C': 100, 'class_weight': 'balanced', 'penalty': 'l2', 'solver': 'newton-cg'}
Gain : -1.21 Accur : 50.5 pour {'C': 100, 'class_weight': 'balanced', 'penalty': 'l2', 'solver': 'lbfgs'}
Gain : -0.86 Accur : 50.69 pour {'C': 100, 'class_weight': 'balanced', 'penalty': 'l2', 'solver': 'sag'}
Gain : -2.06 Accur : 54.58 pour {'C': 1000, 'class_weight': None, 'penalty': 'l2', 'solver': 'newton-cg'}
Gain : -2.36 Accur : 54.44 pour {'C': 1000, 'class_weight': None, 'penalty': 'l2', 'solver': 'lbfgs'}
Gain : -2.09 Accur : 54.52 pour {'C': 1000, 'class_weight': None, 'penalty': 'l2', 'solver': 'sag'}
Gain : -1.09 Accur : 50.22 pour {'C': 1000, 'class_weight': 'balanced', 'penalty': 'l2', 'solver': 'newton-cg'}
Gain : -0.82 Accur : 50.64 pour {'C': 1000, 'class_weight': 'balanced', 'penalty': 'l2', 'solver': 'lbfgs'}
Gain : -0.54 Accur : 50.8 pour {'C': 1000, 'class_weight': 'balanced', 'penalty': 'l2', 'solver': 'sag'}

Best Gain : -0.31 Best Accur : 50.86 pour {'C': 0.1, 'class_weight': 'balanced', 'penalty': 'l2', 'solver': 'sag'}

MultinomialNB

Gain : -1.04 Accur : 50.67 pour {'alpha': 0.001, 'fit_prior': True}
Gain : 0.93 Accur : 50.11 pour {'alpha': 0.001, 'fit_prior': False}
Gain : -1.04 Accur : 50.67 pour {'alpha': 0.01, 'fit_prior': True}
Gain : 0.93 Accur : 50.11 pour {'alpha': 0.01, 'fit_prior': False}
Gain : -1.04 Accur : 50.67 pour {'alpha': 0.1, 'fit_prior': True}
Gain : 0.93 Accur : 50.11 pour {'alpha': 0.1, 'fit_prior': False}
Gain : -1.04 Accur : 50.67 pour {'alpha': 1, 'fit_prior': True}
Gain : 0.86 Accur : 50.08 pour {'alpha': 1, 'fit_prior': False}
Gain : -1.48 Accur : 50.53 pour {'alpha': 10, 'fit_prior': True}
Gain : 0.99 Accur : 50.17 pour {'alpha': 10, 'fit_prior': False}
Gain : -1.07 Accur : 50.72 pour {'alpha': 100, 'fit_prior': True}
Gain : -0.44 Accur : 49.75 pour {'alpha': 100, 'fit_prior': False}
Gain : -1.17 Accur : 52.58 pour {'alpha': 1000, 'fit_prior': True}
Gain : -1.49 Accur : 52.05 pour {'alpha': 1000, 'fit_prior': False}

Best Gain : 0.99 Best Accur : 50.17 pour {'alpha': 10, 'fit_prior': False}