Sorbonne Université

ue reds

Rapport TP 2 : Higgs Boson Machine Learning Challenge

Salomé Attiach, Yannis Karmim

Table des matières

1 Objectif de notre séance. **2**

2 Modèles de références : Baselines. **3**
2.1 Nos différentes métriques pour nos évaluations : . . . 3
2.1.1 Accuracy. . . . 3
2.1.2 Precision. . . . 3
2.1.3 Rappel ou Recall. . . . 3
2.1.4 Score F1. . . . 3
2.2 Modèle random. . . . 4
2.3 Modèle probabiliste naïf. (Stratified) . . . 4
2.4 Modèle fréquentiel. . . . 4
2.5 Classifieur faible : Arbre de décision de faible profondeur. . . . 4
2.6 Méthodes sklearn de base pour la classification. . . . 5
2.7 Récapitulatif de nos résultats. . . . 6

3 Méthodes d'ensembles. **7**
3.1 Modèles. . . . 7
3.1.1 Forêt d'arbres décisionnels (Random Forest) . . . 7
3.1.2 Extra Randomized Tree . . . 7
3.1.3 AdaBoost . . . 7
3.2 Récapitulatif de nos résultats . . . 7

4 Références **8**

1 Objectif de notre séance.

L'objectif de cette séance est de définir et d'appliquer des méthodes de bases pour notre challenge. Ces modèles doivent être naïf ou assez peu élaboré afin de nous donner une idée des scores que l'on cherche à battre pour cette tâche.
Dans ce rapport nous allons détailler les modèles de bases que l'on a choisi (notre baseline) puis présenter nos résultats. Nous allons également détailler nos différentes métriques d'évaluations, ainsi que plusieurs méthodes d'ensemble un peu plus complexe afin de comparer avec notre baseline.

2 Modèles de reférences : Baselines.

2.1 Nos différentes métriques pour nos évaluations :

Nous définissons nos éléments positifs comme étant les fois où l'on détecte un Boson de Higgs (Label 's'). Nos éléments négatifs sont les observations négatives du Boson de Higgs (Label 'b').

Notation :
$TruePositive = TP$ Les éléments positifs que l'on a bien classé comme étant positif.
$TrueNegative = TN$ Les éléments négatifs que l'on a bien classé comme étant négatif.
$FalsePositive = FP$ Les éléments négatifs que l'on a classé comme étant positif.
$FalseNegative = FN$ Les éléments positifs que l'on a classé comme étant négatif.

2.1.1 Accuracy.

L'accuracy est notre première métrique d'évaluation ainsi que celle qui est la plus utilisée dans les tâches de classifications. Elle permet d'évaluer le pourcentage d'éléments que l'on a bien classé, qu'il soit positif ou négatif.

$$Accuracy = \frac{TP+TN}{TP+FP+FN+TN}$$

2.1.2 Precision.

La précision s'intéresse aux éléments que l'on a classé positif et qui sont bel et bien positif. On pénalise donc les faux positifs.

$$Precision = \frac{TP}{TP+FP}$$

2.1.3 Rappel ou Recall.

Le rappel s'intéresse a sélectionné le maximum d'éléments positifs. On pénalise ici les faux négatifs.

$$Rappel = \frac{TP}{TP+FN}$$

2.1.4 Score F1.

Le score F1 cherche à combiner la mesure de précision et de rappel.

$$F1score = \frac{Precision*Rappel}{Precision+Rappel}$$

2.2 Modèle random.

Un modèle uniformément aléatoire avec une probabilité de 1/2 pour la classe **'s'** et 1/2 pour la classe **'b'**

2.3 Modèle probabiliste naïf. (Stratified)

Un modèle probabiliste qui calcul le pourcentage des classes **'b'** et **'s'** et qui choisit aléatoirement en suivant les probabilités associés à la fréquences d'apparitions de nos labels.

2.4 Modèle fréquentiel.

Un modèle naïf qui calcul quelle est la classe la plus fréquente de notre ensemble d'apprentissage, et renvoie systématiquement cette dernière.

2.5 Classifieur faible : Arbre de décision de faible profondeur.

On a utilisé des arbres de décisions de profondeurs 1 et 2 comme classifieurs faibles de notre baseline. L'arbre de décision va choisir le meilleur paramètre sur lequel prendre sa décision pour prédire la bonne classe.
En gardant toutes les variables y compris la variable weight on obtient les résultats suivants :

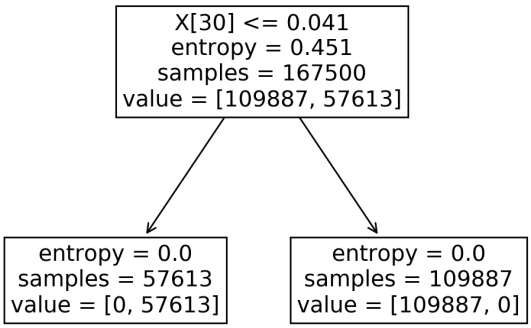

FIGURE 1 – Visualisation arbre de profondeur 1 en gardant l'attribut 'Weight'

La variable X[30] correspond ici à notre attribut 'Weight'. On obtient un score parfait sur notre ensemble de test : $Accuracy = Precision = Recall = 1.0$.
C'est pourquoi pour la suite de notre projet nous retirerons systématiquement cet attribut.

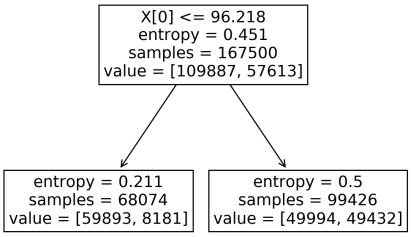

FIGURE 2 – Visualisation arbre de profondeur 1 en supprimant l'attribut 'Weight'

L'attribut sélectionné X[0] est désormais . On détaillera les résultats et nos paramètres dans la section récapitulative.
Les attributs sélectionnés X[1] et X[0] sont : DER_mass_transverse_met_lep , DER_mass_MMC.

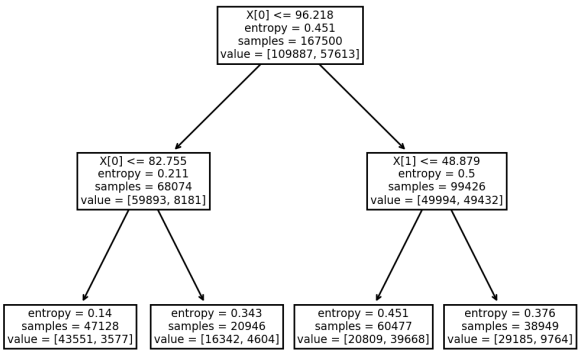

FIGURE 3 – Visualisation arbre de profondeur 2 en supprimant l'attribut 'Weight'

2.6 Méthodes sklearn de base pour la classification.

On a également utilisé un modèle de **Perceptron**, **Naïves Bayes** et **SVM Linear SVC** sans chercher à optimiser leurs paramètres.

2.7 Récapitulatif de nos résultats.

MODELES	Weights	Paramètres modèles	Accuracy	Rappel	Precision	F1_score
__Baselines__						
Pure_random	True/False		0.4989	0.4955	0.3383	0.4021
Stratified	True/False		0.4991	0.4951	0.3384	0.4020
Frequency	True/False		0.6600	None	None	None
Decision Tree	True	{max_depth=1}	1.0	1.0	1.0	1.0
Decision Tree	False	{max_depth=1}	0.66	None	None	None
Decision Tree	False	{max_depth=2}	0.7673	0.6845	0.6498	0.6667
Naives Bayes	False	Gaussian Naives Bayes	0.6760	0.5477	0.5225	0.5348
Perceptron	False	Par defaut : (alpha=0.0001, class_weight=None, early_stopping=False, eta0=1.0, fit_intercept=True, max_iter=1000, n_iter_no_change=5, n_jobs=None, penalty=None, random_state=0, shuffle=True, tol=0.001, validation_fraction=0.1, verbose=0, warm_start=False)	0.6283	0.7535	0.4709	0.5796
SVM Linear SVC for classification	False		0.6966	0.1666	0.7387	0.2719

FIGURE 4 – Nos résultats avec les différentes métriques de nos modèles de bases.

Notre meilleur résultat pour notre baseline est l'arbre de décision d'une profondeur 2. Cela veut dire que les deux attributs DER_mass_transverse_met_lep , DER_mass_MMC, expliquent en grande partie les labels **'s'** et **'b'**. C'est ce modèle que l'on va essayer de battre, tout d'abord en utilisant des méthodes d'ensemble.

3 Méthodes d'ensembles.

3.1 Modèles.

Nous avons utilisés plusieurs méthodes d'ensembles afin de déterminer si ces modèles sont efficaces pour notre tâche et s'ils dépassent nos modèles de bases.
On a testé plusieurs modèles de bagging et de boosting, sans pour autant essayer d'optimiser les paramètres avec de la cross-validation. L'objectif était de déterminer si ces méthodes d'ensembles semblent plus efficace.

3.1.1 Forêt d'arbres décisionnels (Random Forest)

La forêt d'arbres décisionnels Breiman 2001 est une méthode dit de bagging sur des classifieurs faibles qui sont dans ce cas des arbres de décisions de profondeur 1 ou 2 généralement. Il était intéressant pour nous de tester ce modèle puisque notre meilleur baseline est un arbre de décision de profondeur 2. On peut maintenant voir si le moyennage de plusieurs de ces arbres, afin de réduire la variance, est efficace.

3.1.2 Extra Randomized Tree

La forêt d'arbres décisionnels extrêmement aléatoire (Geurts, Ernst et Wehenkel 2006) est très similaire à une forêt d'arbres décisionnels excepté que le split sur les arbres de décisions se fait complètement aléatoirement. On voulait comparer par rapport à la forêt d'arbres décisionnels, pour voir laquelle de ces méthodes semble la plus efficace.

3.1.3 AdaBoost

Afin de comparer avec nos méthodes de bagging on a utilisé le classifieur AdaBoost Freund, Schapire et Abe 1999 de la librairie SkLearn. Comme le bagging, les méthodes de boosting utilisent un ensemble de classifieurs faibles, mais ces classifieurs sont appris séquentiellement avec une pondération dans les exemples d'apprentissages.

3.2 Récapitulatif de nos résultats

MODELES	Dataset	Weights	Paramètres modèles	Accuracy	Rappel	Precision	F1_score
__Méthodes d'ensembles__							
Random Forest	Test	False	n_estimators=100,bootstrap=True,max_depth=5	0.8155	0.6365	0.7803	0.7011
Random Forest	Test	False	n_estimators=100,bootstrap=False,max_depth=5	0.8143	0.6350	0.7782	0.6993
Extra Trees	Test	False	n_estimators=100,bootstrap=True,max_depth=5	0.7077	0.1870	0.8009	0.3033
Extra Trees	Test	False	n_estimators=100,bootstrap=False,max_depth=5	0.7094	0.1939	0.7999	0.3122
Bagging	Test	False	n_estimators = 10	0.8212	0.6625	0.7788	0.7159
Bagging	Test	False	n_estimators = 100	0.8360	0.7144	0.7841	0.7476
AdaBoost	Test	False	default	0.8130	0.7021	0.7359	0.7186

Figure 5 – Nos résultats avec les différentes métriques de nos modèles de bases.

On voit que nos meilleurs résultats proviennent de la classe Bagging de sklearn, qui est très proche d'AdaBoost. Même si ces résultats sont encore à vérifier puisque on a pas optimisé les hyper-paramètres des modèles. Néanmoins on peut affirmer que les méthodes d'ensembles semblent meilleures que notre baseline.

4 Références

Breiman, Leo (oct. 2001). "Random Forests". In : *Mach. Learn.* 45.1, p. 5-32. ISSN : 0885-6125. DOI : 10.1023/A:1010933404324. URL : https://doi.org/10.1023/A:1010933404324.

Freund, Yoav, Robert Schapire et Naoki Abe (1999). "A short introduction to boosting". In : *Journal-Japanese Society For Artificial Intelligence* 14.771-780, p. 1612.

Geurts, Pierre, Damien Ernst et Louis Wehenkel (2006). "Extremely randomized trees". In : *Machine learning* 63.1, p. 3-42.