Projet Prédiction Conformelle

MASTER 2 MIASHS

Ugo ZENNARO

UFR Anthropologie Sociologie Science Politique (ASSP)
Université de Lyon, Université Lumière Lyon 2

Enseignant : Rémi VAUCHER

1 Introduction

Dans le cadre d'un cours sur la prédiction conformelle, des méthodes de classification et de prédiction seront présentées dans ce rapport, ainsi qu'une analyse de chacun des cas sur la qualité et la justification des choix des prédicteurs et des méthodes étudiées. Les deux problématiques à l'étude sont les suivantes :

1. Prédiction de performance (poids en kilogrammes, qu'on notera Y) en haltérophilie selon l'âge, le poids et le nombre de points IPF (qui est indice la qualité de l'athlète). Ces variables explicatives seront notées X. En ayant en tête que les haltéorphiles professionelles savent exactement les performances qu'ils peuvent faire, que leur objectif est probablement de faire partie des outliers et être extraordinairement performant, les outils proposées ne leurs seront pas dédiés. En revanche dans le cadre d'un débutant, d'un athlète qui changerait de catégorie de poids ou qui reprendrait la pratique après un certain nombre d'années de pause, l'objectif est de leur donner un a priori sur les performances auxquelles ils devraient pouvoir prétendre. (Source des données : openpowerlifting)
2. Classification d'images de vêtements pour retrouver l'étiquette qui correspond au vêtement illustré. Cet outil pourra être utile pour des sites de ventes de particuliers à particuliers pour proposer des descriptions automatiques des photos, propre à chaque annonce, prises par les vendeurs. Cela pourrait par exemple aider le vendeur à remplir plus rapidement la fiche descriptive de son produit, mais ne doit pas le faire patienter plusieurs minutes pour lui dire quelque chose qu'il sait déjà sur son produit. (Source des données : fashionMNIST)

2 Performances d'haltérophilie

Analyse et visualisation des données

Toutes les variables observées pendant cette tâche sont des variables quantitatives continues.

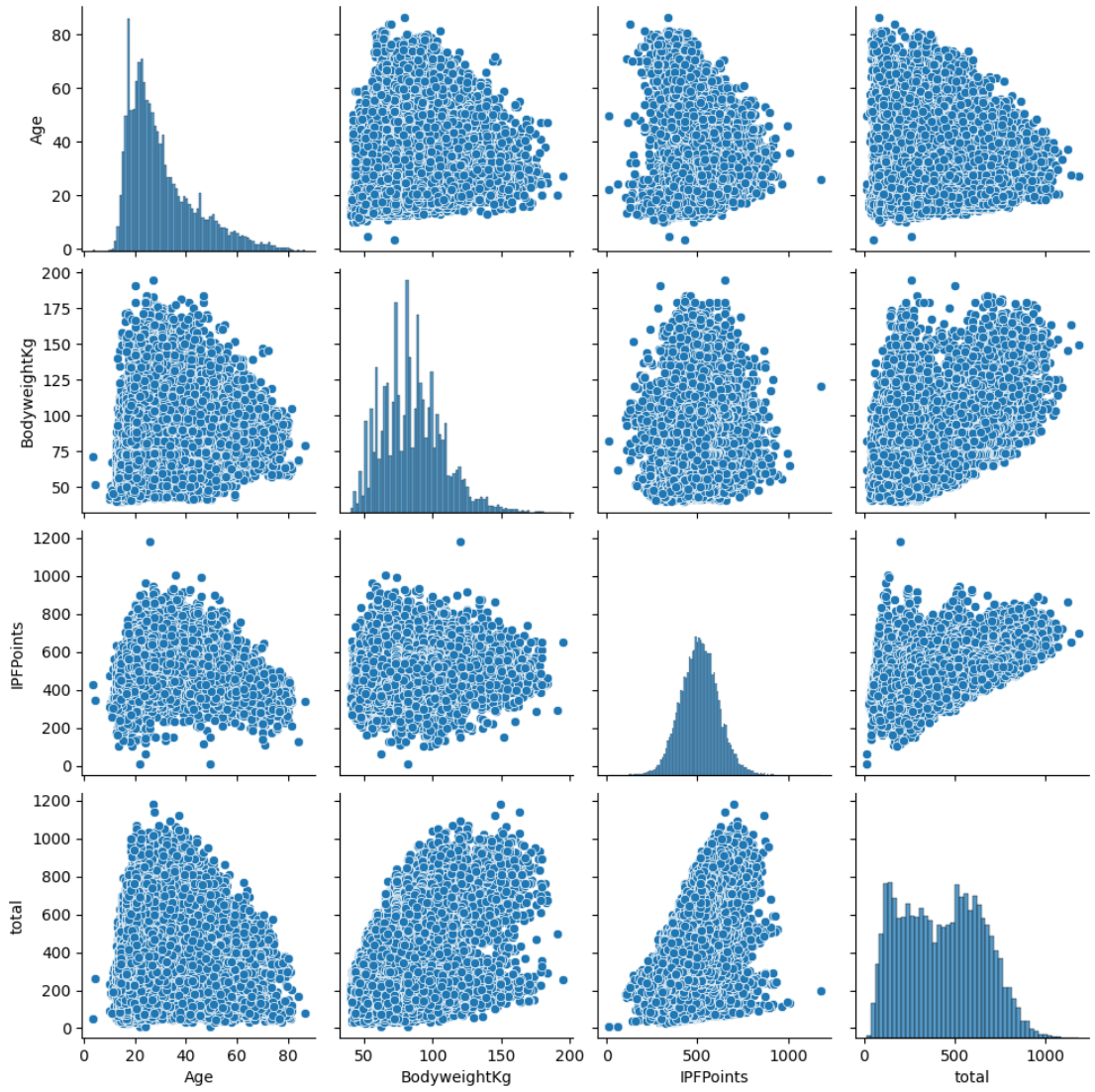

Illustration 1 – Répartition par paires de variable

Ici, on observe déjà que, sur la population observée, les pratiquants sont moins nombreux à partir d'un certain âge. Au niveau de la repartition des poids, on observe, par dessus la densité naturelle, une forme de fréquence des densités qui correspond a priori aux catégories de poids des athlètes : ils ont tendance à être à la limite du poids maximum autorisée pour concourir dans une catégorie. Les points IPF semblent distribués normalement, centrées aux alenturs de 500. En revanche les performances totales semblent suivre une distribution multimodale. Aussi quand on observe la variable d'interêt des performances totales par rapport aux variables explicatives, on observe différentes formes de corrélations. Pour l'âge déjà, il semblerait que la distribution indique

que les meilleures performances sont observées entre 20 et 50 ans, avant on voit une progression des performances maximales et une régression après. Ensuite, il semblerait que la performance soit corrélée positivement avec le poids et le nombre de points IPF. Cela se confirme en regardant les scores de corrélations tous les deux proches de 40% pour ces deux variables avec la performance totale. On notera que la répartition des performances semble éparse quelque soit la variable explicative par rapport à laquelle on l'observe. Enfin on observe que la variable d'interêt du jeu de données étudié s'étend sur un intervalle qui commence à 10 kgs et qui plafonne à 1180 kgs, la médianne est de 430 kgs, sa moyenne est de 428 kgs et l'écart-type est de 220 kgs. Les travaux présentées utilisent un échantillon de 26305 individus.

Pour la suite les données sont séparés en deux : les données d'entrainement et de test qui représentent respectivement 80% et 20% du jeu de données complet.

2.1 Regression quantiles

Pour commencer, la première méthode de prédiction étudiée est la régression quantile. Cette méthode consiste à trouver la droite qui donne le quantile de l'ordre choisi conditionnellement aux variables explicatives. Pour avoir un interval pour lequel on fixe $\alpha = 10\%$ le niveau de rejet, on va ici entraîner deux régressions quantile, d'ordres $\frac{\alpha}{2}$ et $1 - \frac{\alpha}{2}$ pour retrouver $1 - \alpha = 90\%$ des valeurs de Y entre les deux droites conditionnées par X. La régression quantile proposée ici n'est pas polynomiale car après avoir comparé les performances d'une régression linéaire et d'une regression polynomiale, on a vu que la racine de l'erreur quadratique moyenne était similaire entre les deux (respectivement de 182 kgs et 176 kgs d'erreur), aussi car la multiplication du nombre de variables explicative aurait tendance à faire exploser le temps de calcul de la régression quantile. Le code python de la méthode présentée est ci-dessous.

Algorithme 1 Régression Quantile sur python

```
alpha = 0.1
# peu de données d'entrées donc on ne regularise pas
qr_down = QuantileRegressor(quantile=alpha/2, alpha = 0, solver='highs').fit(X_train,
↪  y_train)
qr_up = QuantileRegressor(quantile=1-alpha/2, alpha = 0, solver='highs').fit(X_train,
↪  y_train)

# intervalle de prédiction défini par les regressions quantile
lower = qr_down.predict(X_test)
upper = qr_up.predict(X_test)
prop = np.mean([(y_test > lower) & (y_test < upper)])
```

La variable *prop* contient le taux de Y qui sont réellement dans l'intervalle prédit et on trouve que 89,1% d'entre eux y sont. On peut regretter que le niveau de rejet soit ici excédé, aussi on observe une taille moyenne des intervalles de 501 kgs. On se

rend bien compte du manque de précision de l'intervalle. Ces résultats nous serviront pour interpreter ceux des méthodes de prédiction conformelle. Les intervalles calculés ici reposent sur des incertitudes empiriques. La prédiction conformelle va maintenant nous permettre d'étudier l'erreur plus spécifiquement et de créer des intervalles qui reposent sur une incertitude plus rigoureuse car on la basera sur une étude statistique des erreurs.

2.2 Regression conformelle

Maintenant, l'objectif est de construire des scores de conformités, qui, pour la regression, reposeront sur l'erreur absolue entre la valeur prédite et la valeur étalon. On manipulera ces scores différemments selon la méthode utilisée en commençant par la SCP puis JackKnife+ qui integrent de manière différente la valeur à prédire dans la construction de l'intervalle et calibrent différemment ces intervalles. La Cross-Validation+ constitue un entre-deux qui pourrait constituer un gain de temps par rapport à JackKnife+ et qui intègre mieux les valeurs à prédire dans la construction de l'intervalle que la SCP mais, au vu des résultats de la regression quantile et des regressions conformelles qu'on présentera, l'apport de ces distinctions m'a semblé trop limité pour être faites.

Split Conformal Prediction

La SCP (*Split Conformal Prediction*) demande que les données d'entrainement soit à nouveau séparées en 2. On prendra 40% d'entre elles pour entrainer le modèle de regression $f(X)$. Au vu des répartitions de la variables d'interêt par rapport aux variables explicative, il ne semble pas que le modèle soit censé comprendre des géométries particulièrement complexes, nous utiliseront donc simplement un modèle de regression linéaire qui propose une vitesse de calcul agréablement rapide. Les données restantes (60%) sont utilisés pour ce qu'on appelle la calibration, c'est-à-dire que c'est avec ces données qu'on observera les erreurs absolues du modèle. On définit ces erreurs comme les scores de conformité. L'idée ici est donc d'utiliser q_α, le quantile d'ordre $1-\alpha$ de l'ensemble des scores de conformités comme distance étalon à la valeur prédite par le modèle. On construit ensuite l'intervalle pour chaque valeur des données test par $[f(X) - q_\alpha; f(X) + q_\alpha]$.

Cette méthode à l'avantage de pouvoir, une fois entrainée, donner l'intervalle de prédiction très rapidement pour chaque valeur X_{new} suplémentaire car on a juste à utiliser la prédiction du modèle entrainée et de faire une somme et une soustraction avec le quantile calibré. Dans notre cas d'étude, un utilisateur aurait juste à rentrer ses informations dans une application dédiée pour immédiatement avoir son intervalle de performance. En ce qui conserne la performance de l'algorithme, on observe, déjà un intervalle encore plus gros qu'avec la regression quantile, qu'on obtient en multipliant q_α par 2, ce qui nous donne 586 kgs d'intervalle. Ces 77kgs d'intervalles suplémentaires nous rapproche tout juste de la valeur du taux de rejet α qu'on visait, avec un taux de conformité de 89,64%. Les performances ne sont pas grandement meilleurs que pour la regression quantile et les regrets qu'on pouvait avoir quant à l'ampleur de l'intervalle se

Algorithme 2 SCP sur python

```
## SCP
X_train2, X_calib, y_train2, y_calib = train_test_split(X_train, y_train, test_size=0.6,
↪  random_state=13)
modele = LinearRegression().fit(X_train2, y_train2) # Entrainement

# Calibration
y_hat_calib = modele.predict(X_calib)
S = np.abs(y_hat_calib - y_calib) # Scores de conformité
q_S = np.quantile(S, 1-alpha)

# Construction de l'intervalle
y_hat_test = modele.predict(X_test)
lower_scp = y_hat_test - q_S
upper_scp = y_hat_test + q_S
prop_scp = np.mean([(y_test > lower_scp) & (y_test < upper_scp)])
```

retrouvent amplifiés sans un gain significatif sur la performance de la prédiction de cet intervalle.

JackKnife+

JackKnife+ utilise toutes les données d'entrainements à la fois pour l'entrainement et la calibration. En ça, la méthode CrossValidation+ est similaire car elle sépare le jeu de données en k groupes de même taille et utilise successivement chaque groupe pour la calibration et le reste pour l'entrainement. JackKnife+ est un cas particuliers de la CrossValidation+ où on considère un groupe par individu. À la différence de la SCP étudiée ces méthodes sont adaptatives, c'est-à-dire que la probabilité que l'intervalle de prédiction soit bon est conditionnée par l'observation X_{new}. Pour cela, les scores de conformités dépendront de ces obsevations. Pour chaque valeur X_i utilisée successivement pour l'entrainement du modèle $f_{-i}()$, on aura une erreur absolue entre la prédiction de X_i par la fonction $f_{-i}()$ et Y_i. On utilisera ensuite ces erreurs absolues pour construire deux ensembles de score de conformités qui, respectivement, somment et soustraient les erreurs absolues avec les predictions de X_{new} par $f_{-i}()$ affiliées. L'intervalle est ensuite simplement construit avec le quantile d'ordre α de l'ensemble des scores de conformités qui soustrait l'erreur absolue à la prédiction du modèle correspondant, et avec le quantile d'ordre $1-\alpha$ de l'ensemble qui somme.

Là encore, dans une tendance qui permet d'augmenter l'efficacité statistique, on obtient un intervalle encore plus grand : 588 kgs cette fois-ci, une valeur plus proche de celle donnée par la SCP que celle de la SCP par rapport à la régression quantile. Et on retrouve également une valeur de α empirique qui se rapproche de 10% mais qui reste supérieure avec un taux de conformité de 89,89%. Cette légère différence de performance s'effectue au prix d'une multiplication considérable du temps de calcul de l'intervalle

Algorithme 3 JackKnife sur python

```
## jackknife

# scores de conformité
lower_jn = []
upper_jn = []
for i in X_train.index :
  modele_jack = LinearRegression().fit(X_train.drop(i), y_train.drop(i))
  diff_jn = np.absolute(modele_jack.predict([X_train.loc[i]]) - y_train.loc[i])
  y_hat = modele_jack.predict(X_test)
  upper_jn.append(y_hat + diff_jn)
  lower_jn.append(y_hat - diff_jn)

# construction des intervalles
y_low = []
y_up = []
for i in range(np.array(upper_jn).shape[1]) :
  y_up.append(np.quantile(np.array(upper_jn)[ :, i], 1- alpha))
  y_low.append(np.quantile(np.array(lower_jn)[ :, i], alpha))

prop_jn = np.mean([(y_test > y_low) & (y_test < y_up)])
```

puisque'on passe de quelques secondes (0 si on en croit Google Colab) pour la SCP à plusieurs dizaines de minutes pour JackKnife+ (3 pour le calcul des scores de conformités et 48 pour la construction des intervalles).

2.3 Conclusion

Au vu des résultats pour chacune des méthodes on peut considérer, sachant notre contexte, que la précision octroyée par JackKnife+ n'est pas suffisante pour immobiliser un utilisateur plusieurs dizaines de minutes si il a plusieurs valeurs a prédire, en revanche, dans le cadre d'une utilisation personnelle ou chaque utilisateur calcule son intervalle, le temps de calcul s'en trouverait réduit et pourrait être envisageable surtout si cet intervalle était utilisé pour construire une courbe de progression de l'athlète, auquel cas l'ampleur de l'intervalle serait un avantage. Dans le cadre d'une utilisation plus informative, pour se faire une vague idée de ce qui est envisageable du point de vue de l'athlète, et à plus grande échelle, l'utilisation de la SCP serait plus indiquée car, même si on a un taux de conformité légérement plus faible, le temps de calcul compense largement. Pour finir on dira que, sans doute à cause du faible nombre de variables explicative, on a quand même un manque flagrant de précision dans la prédiction et que, même si l'intervalle fournit peut trouver un interêt dans le cadre d'une programmation refléchie pour un athlète, un athlète débutant ne pourra probablement pas être capable de performer à hauteur d'une performance prise aléatoirement dans l'intervalle qui lui est donné, puisque celui-ci est basé sur des performances maximales d'athlètes en compétitions, ce qui indique que cela

résulte d'un certain travail et d'un entrainement préalable à la performance.

3 Images de vêtements

Description des données et des problématiques

Pour la classification des images, chaque image (qui prendra ici la notation X) est décrite par un total de 784 pixels (28x28) qui peuvent prendre une valeur entre 0 (noir) et 255 (blanc) pour définir l'intensité des pixels qui composent l'image en noir et blanc. La liste des étiquettes (les classifications qui seront associées à la notation Y) est la suivante :

— T-shirt/top
— Trouser
— Pullover
— Dress
— Coat
— Sandal
— Shirt
— Sneaker
— Bag
— Ankle boot

Le travail présenté ici aura donc pour objectif d'utiliser des modèles de classification, ici on utilisera RandomForest qui aura l'avantage d'être faciliment utilisable et qui, comme on en aura besoin, renvoit des probabilités de classification pour chacune des classes énumérées ci-dessus. Cela est possible car cette méthode est ensembliste, c'est-à-dire qu'elle utilise différents classifieurs pour obtenir les probabilités que les individus se classent dans différentes classes. Ici, comme on l'a vu pour la regression, on utilisera la *Split Conformal Prediction* et on montrera différents scores pour créer des ensembles conformes de classification, l'objectif général étant d'avoir, avec un taux de conformité $1 - \alpha$, la bonne étiquette dans l'ensemble de classes proposé. Dans la veine de la méthode JackKnife+ présentée pour la regression, la *Full Conformal Prediction* permet d'intégrer les observations dans la construction des scores de conformités et de conditionner les intervalles aux observations. La FCP est en cela intéressant qu'elle exploite et implique le plus les observations à classifier dans la construction des modèles. Cela implique une multiplication du nombre de modèle utilisé, comme JackKnife+ multipliait le seul modèle utilisé par la SCP par par le nombre d'individus utilisés pour l'entrainement, ici on multiplie ce modèle par le nombre de classes et par le nombre d'individus à classifier. Cependant l'utilité prévue à la classification est de pouvoir aider des vendeurs d'une plateforme de vente de vêtements de particuliers à particuliers en suggérant, voir en automatisant la description du produit grâce à la classification automatique. Alors cette multiplication du nombre de modèle pourrait impliquer de faire patienter le client ce qu'on se refuse de faire, c'est pour cela que dans ce contexte, seule la SCP sera étudiée.

3.1 Classification conformelle

Pour commencer, on a donc entrainé notre modèle sur 8000 données d'entrainements, utilisé 8000 données pour la calibration, et 10000 données servent au test.

Score 1

Le premier score étudie la probabilité $f(X_i)_{Y_i}$ associée à l'étiquette étalon Y_i et pour chaque élément de l'ensemble de calibration $X \in Calib$ on construit $S = \{s(X_i, Y_i) = 1 - f(X_i)_{Y_i}, (X_i, Y_i) \in Calib\}$ l'ensemble des scores de conformité. Ainsi, en prenant le quantile d'ordre $1 - \alpha$ de ces scores, pour toutes données test à prédire X_{new}, on selectionne toutes les classes C_i où les scores de conformités $s(X_{new}, f(X_{new})_{C_i})$ sont supérieurs à ce score quantile. En faisant cela, l'ensemble renvoyé est construit pour que sa taille dépende de la qualité du modèle, et plus le modèle sera performant plus l'intervalle sera petit.

Score 2

Le second score $s(X_i, Y_i)$ étudié se calcule comme la somme des probabilités $f(X)_{C_i}$ supérieures ou égales à $f(X)_{Y_i}$. Ensuite on construit l'intervalle pour qu'il contienne toutes les étiquettes C_i associées aux probabilités $f(X_{new})_{C_i}$ les plus grandes telle que la somme de ces probabilités soit la plus petit possible et soit supérieur au quantile d'ordre $1 - \alpha$ de $S = \{s(X_i, Y_i), (X_i, Y_i) \in Calib\}$. Comme cet algorithme est construit sur une somme de probabilités, on peut se retrouver avec un plus grand intervalle qu'avec l'autre score car ici on pourra y trouver des étiquettes assoiées à des probabilités plus faibles.

On construit les ensembles de scores calculés sur les données de calibration avec la fonction suivante :

Algorithme 4 Scores de classification sur python

```
# fonction score
def score_classif(y_prob,
                  y_gold, # Probabilités données aux vraies étiquette (algo 2)
                  classes, # Vraie étiquette (algo 1)
                  algo = 1) :
  n = len(y_gold)
  prob_y_gold = np.array([y_prob[i][y_gold[i] == classes] for i in range(n)])
  if algo == 1 :
    S = 1 - prob_y_gold
  elif algo == 2 :
    S = np.array([np.sum(y_prob[i][y_prob[i] > prob_y_gold[i]]) for i in range(n)])
  return S
```

Et on construit les ensembles de prédiction selon le code suivant :

Algorithme 5 Construction des intervalles

```
## Score 1

s1 = 1 - y_hat_test
y_conformal = [list(classes[s1[i] < q_S1]) for i in range(len(y_hat_test))]
np.mean([y_test[i] in y_conformal[i] for i in range(len(y_test))])


## Score 2

def s2(proba, q_S, classes) :
  S = []
  for p in proba :
    sum = 0 # score de conformité
    yc = [] # liste des prédictions
    usable = np.array([True] * len(classes)) # index des probas utiliables
    while sum < q_S :
      max = np.max(p[usable])
      usable[p == max] = False # enlève les probas utilisés de l'index
      yc  = yc + list(classes[p == max])
      sum = sum + max * np.sum(p == max)
    S.append(yc)
  return S

y_conformal2 = s2(y_hat_test, q_S2, classes)
np.mean([y_test[i] in y_conformal2[i] for i in range(len(y_test))])
```

Résultats et analyse

Tout d'abord on peut examiner les scores de conformités et observer qu'on obtient, pour le premier score 89,31% et 89,45% pour le second. Il n'y a pas une grande différence entre ces deux taux, mais on note que le deuxième score est légerement plus performant. Ensuite il est intéressant d'observer la taille des ensembles de classes prédites. En moyenne, pour le premier score, il y a 1,108 classes prédites et 1,147 pour le deuxième. On voit ce qu'on observait pour la regression, c'est-à-dire que les ensembles prédits sont plus grands quand on utilise des scores qui ont de meilleurs taux de conformité. Aussi on peut expliquer la différence de taille moyenne des intervalles par la manière dont ils sont construits. Le premier à la particularité de prendre les classes qui satisfont une condition, et dans certains cas, aucune classe ne satisfait cette condition et on se retouve avec des ensembles de prédiction vides. La méthode nous dit à ce moment là que l'incertitude est trop grande, même pour renvoyer la plus grande probabilité. Tandis qu'avec la seconde méthode on prend au moins une étiquette et plus l'incertitude est forte, plus la méthode renvoie d'étiquettes. Ainsi, si pour le premier score on n'observe que les prédictions qui proposent des solutions (plus de 98% des cas), on passe à un score de conformité de 90,87%.

3.2 Conclusion

Pour conclure, on a observé la SCP, une méthode de prédiction conformelle computationellement efficace pour les raisons imposées par le contexte présenté plus tôt. Aussi on a observé deux scores concurents pour l'application de cette méthode et on a vu que ce qui les distinguait était que l'un regardait les probabilités associées à chaque classe de manière indépendante tandis que l'autre construisait son intervalle en sommant ces probabilités. On a observé que cela nous permettait dans le premier cas d'avoir une méthode qui, d'une certaine manière, exprime qu'elle ne sait donner un ensemble conformel de prédiction. Dans notre contexte, si on considère que l'objectif est simplement d'aider un vendeur à remplir plus efficacement une annonce, on peut considérer que cette approche (qui semble rester un cas largement minoritaire avec moins de 2% des cas) permet de renoncer à l'aide initialement prévue, plutôt que de noyer le vendeur dans une longue liste d'étiquette possibles mais sans aucune ayant une probabilité vraiment forte selon le modèle classifieur, comme on pourrait l'observer avec la deuxième méthode. Même si le taux de conformité est un peu plus faible, on négligera ce faible écart pour favoriser un ensemble de prédiction un peu plus petit, bien que, comme on l'a vu, l'écart des tailles de prédiction pourrait lui aussi être conidéré comme négligeable. Ainsi, j'aurai tendance à conseiller d'utilier le premier score dans ce contexte sans être catégorique sur l'utilité absolue de la différence que choix entraine.