**Institut Mines-Telecom EMSI, Departement GMI**

**2020 – 2021**

**Apprentissage. Supervise**

**Group :**

**YutengWANG**

**Wenxu Zhao**

**Liwei Xu**

1 TP No 02 : Arbres de decision & Forets Aleatoires

Exercice 1 (DM-ML-DT). Dans cet exercice, il s'agit d'explorer le processus de la classification supervisee (phase d'apprentissage et phase de classement). L'objectif aussi, est d'experimenter plusieurs implementation de l'arbre de decision.

1. A partir du repertoire data de Weka, charger le fichier weather.nominal.arff. Ensuite, remplir le tableau suivant:

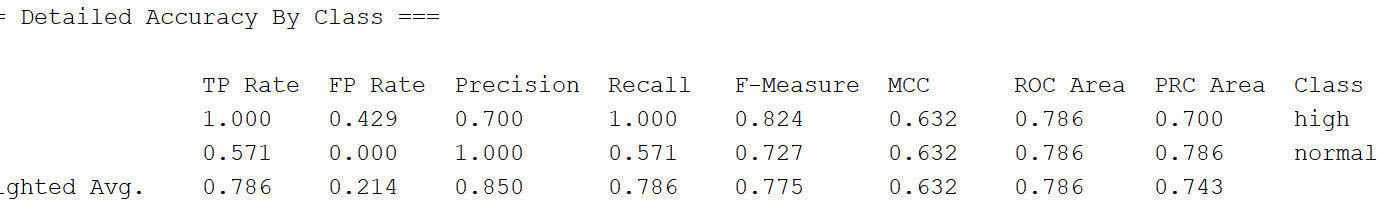
|  |  |  |  |
| --- | --- | --- | --- |
| Attribut | Regles | Taux d'erreur | Taux d'erreur total |
| *Outlook* | sunny - no overcast - yes  rainy - yes | 2/5  0  2/5 | 2/7 |
| *Temperature* | hot - no  mild - yes  cool - yes | 1/2  1/3  1/4 | 5/14 |
| *Humidity* | high - no  normal - yes | 3/7  1/7 | 2/7 |
| *Windy* | false - yes  true - yes | 1/4  1/2 | 5/14 |

1. Le classifieur OneR utilise un seul attribut (celui ayant le plus faible taux d'erreur) pour effectuer la classification. Quelles regles de classification aura-t-on si on utilise OneR ?

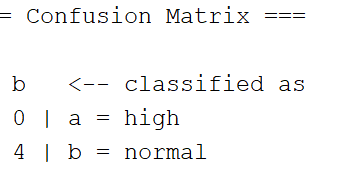
Pour chaque attribut，Pour chaque valeur de cet attribut, établir la règle suivante

* Calculer la fréquence d'occurrence de chaque catégorie
* Identifier les catégories d'événements les plus fréquents
* Créer des règles pour attribuer cette catégorie à cette valeur d'attribut
* Taux d'erreur pour les règles de calcul
* Sélectionnez la règle ayant le taux d'erreur le plus faible

1. Verifier la reponse de la question precedente (Onglet *Classify : Classi- fler/rules/OneR).*
2. Explorer le resultat de la classification
3. *Detailed Accuracy By Class (TP, FP, Precision, Recall,...)*



1. *Confusion Matrix*



1. A partir du repertoire en ligne *nEchantillons de donnees'"* de *TENT,* choisissez un echantillon de donnees[[1]](#footnote-0) [[2]](#footnote-1). Utiliser les differentes implementations de l'arbre de decision sous Weka a savoir :

—HoeffdingTree\*

—J48

—LMT

— RandomTree

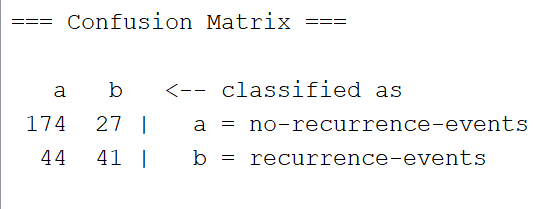
—REPTree

—SimpleCart

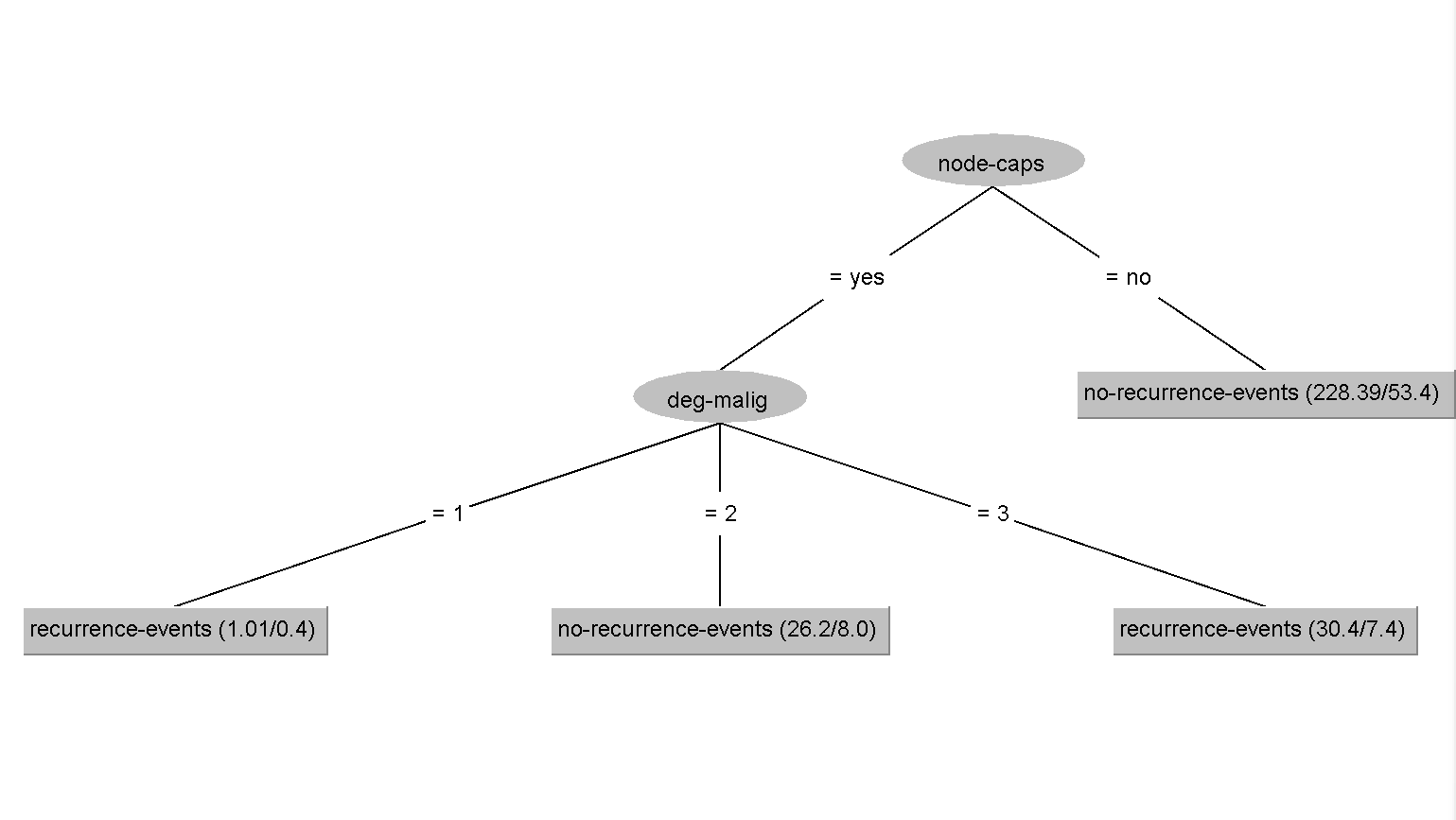
Pour chaque implementation :

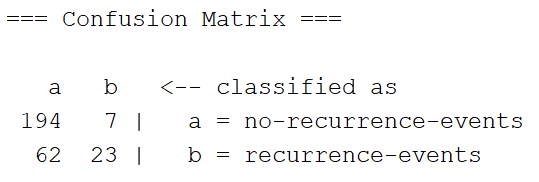
— Selon le mode operatoire *(Test Options: Use training set),* visualisez I'arbre genere *(Visualize Tree).*

HoeffdingTree

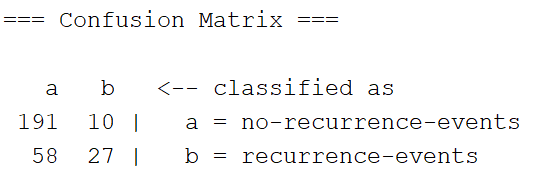


J48

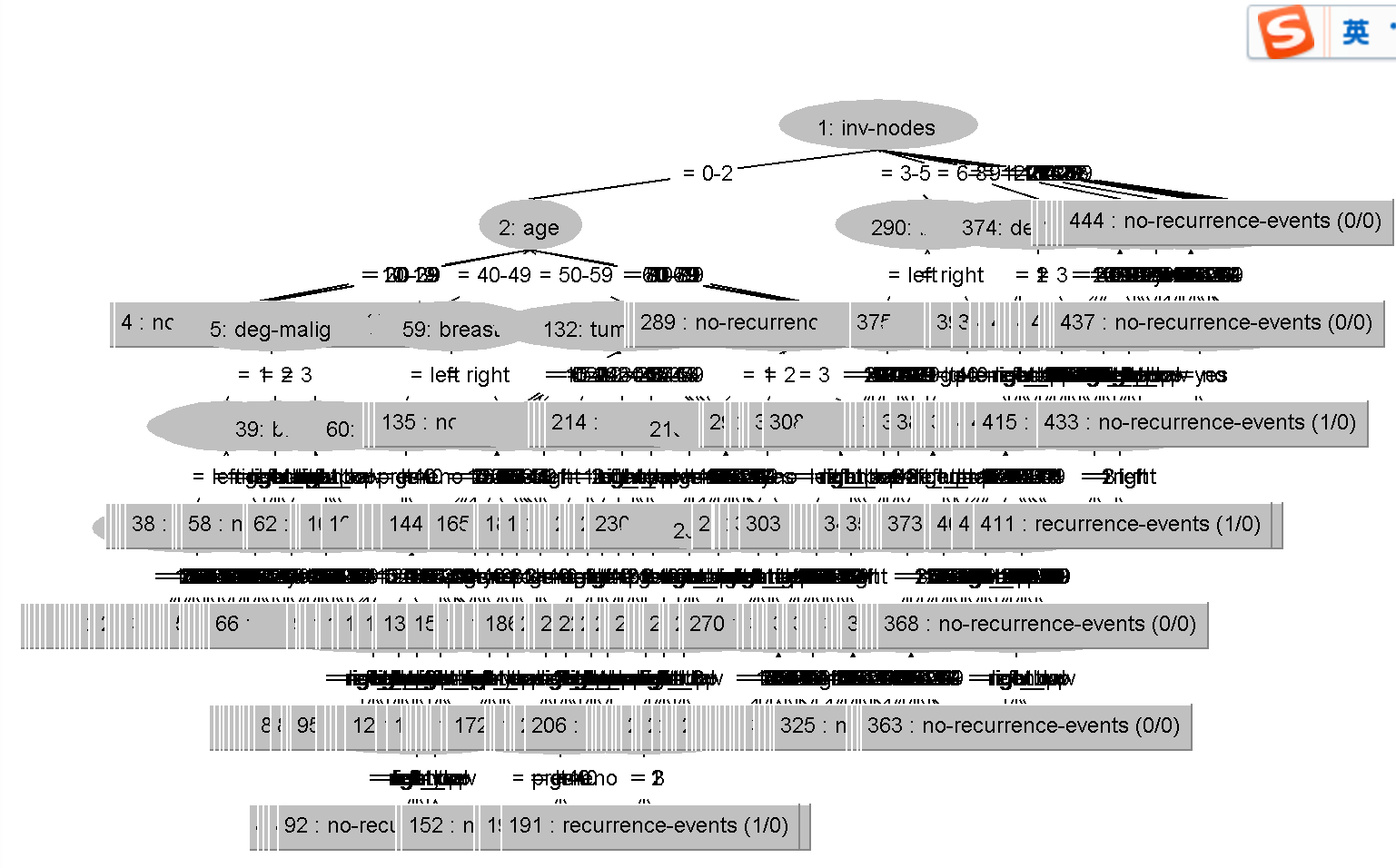


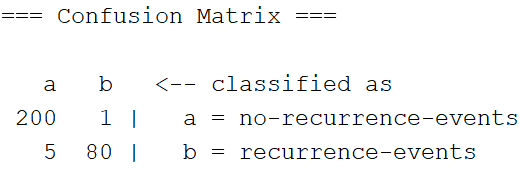


LMT

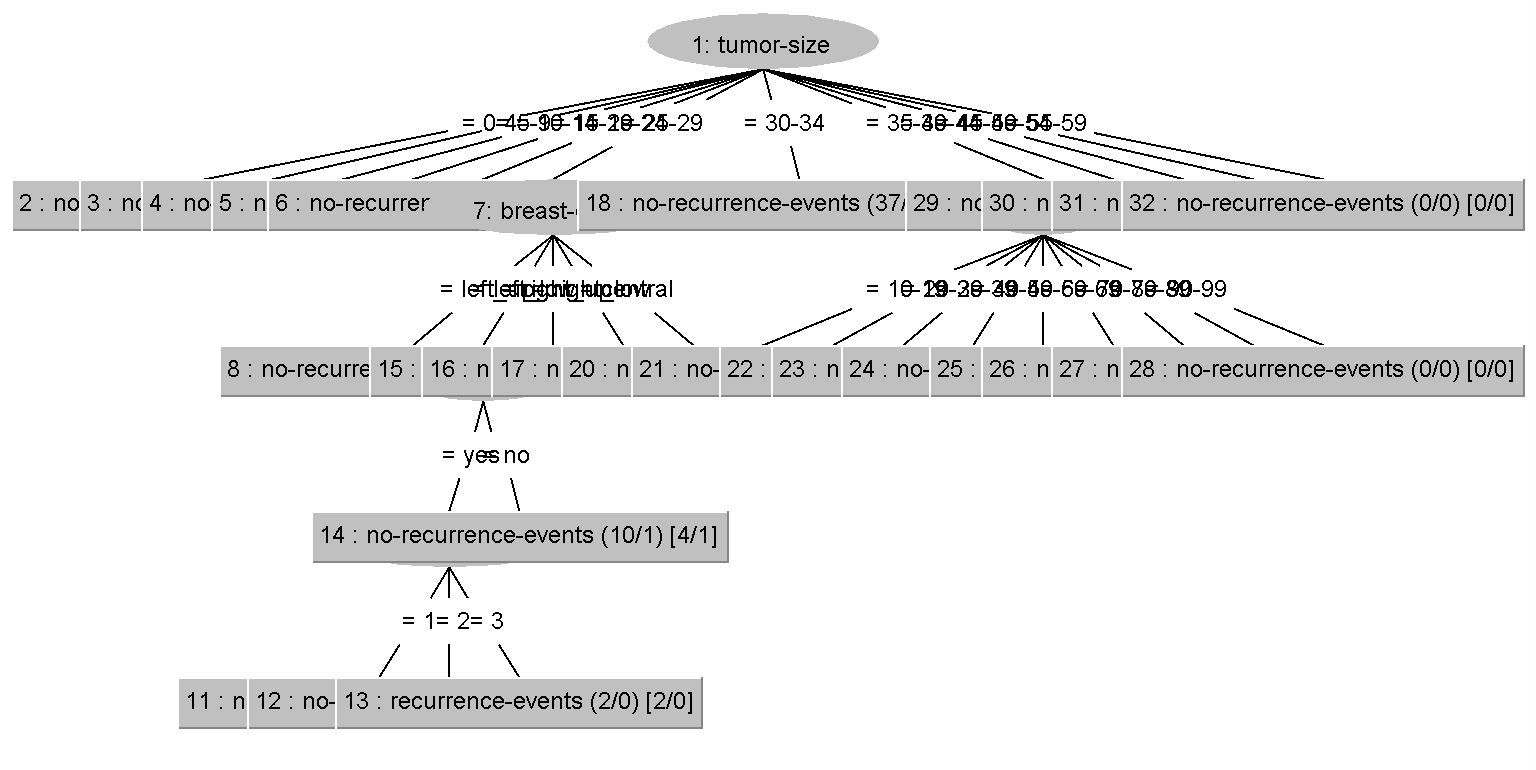


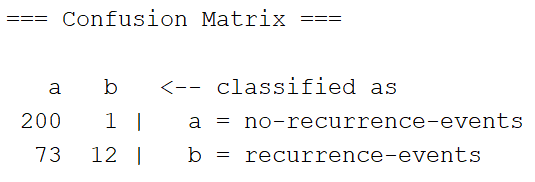
RandomTree



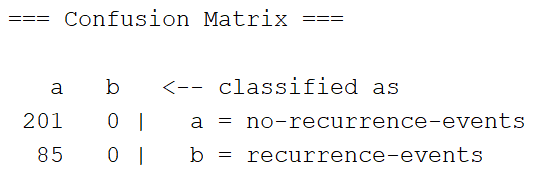


REPTree





SimpleCart

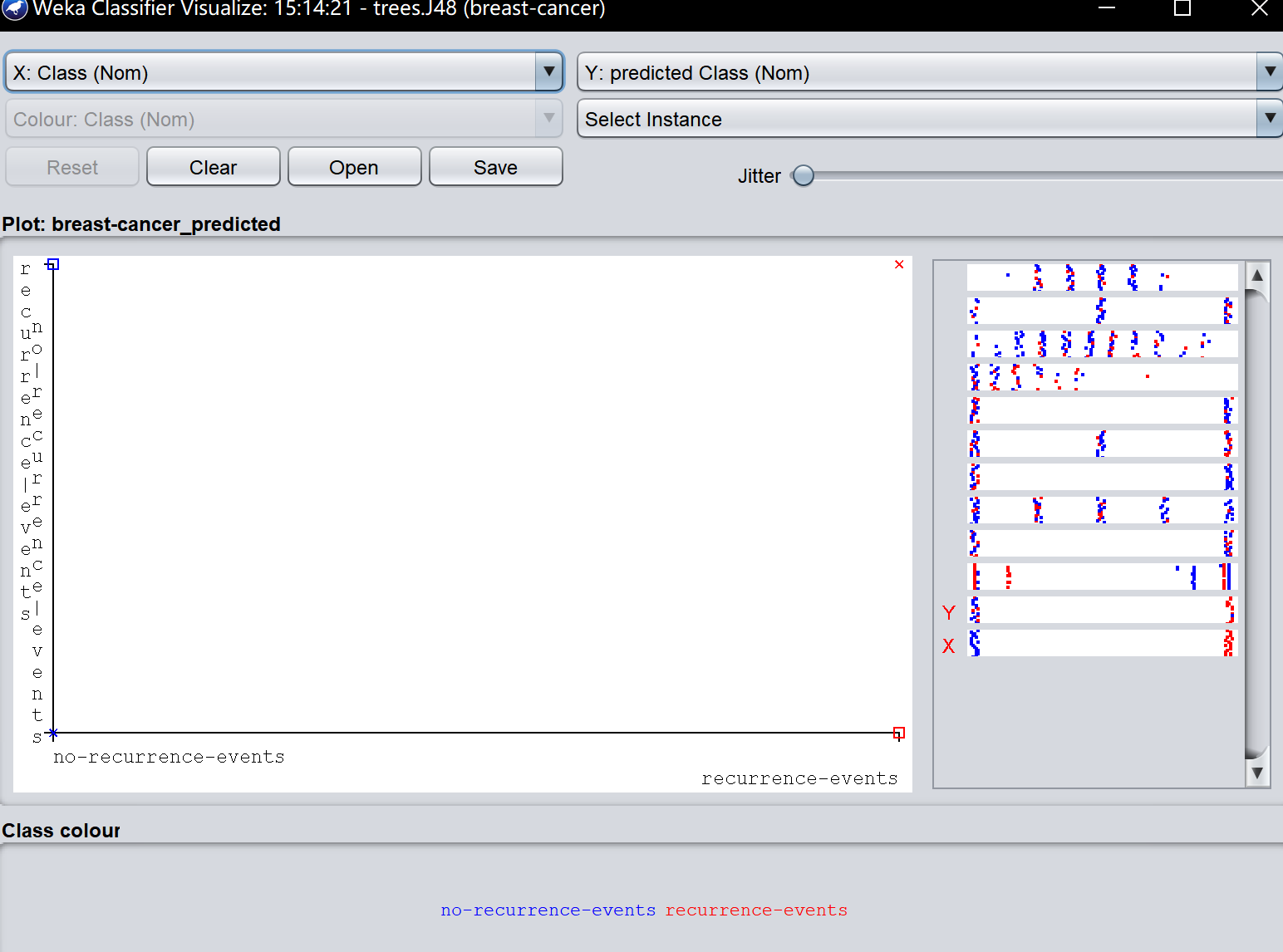


— Faites varier les techniques d'evaluation (Use *training Set, 2 and 10 Cross­validation, 66% Percentage Split*).

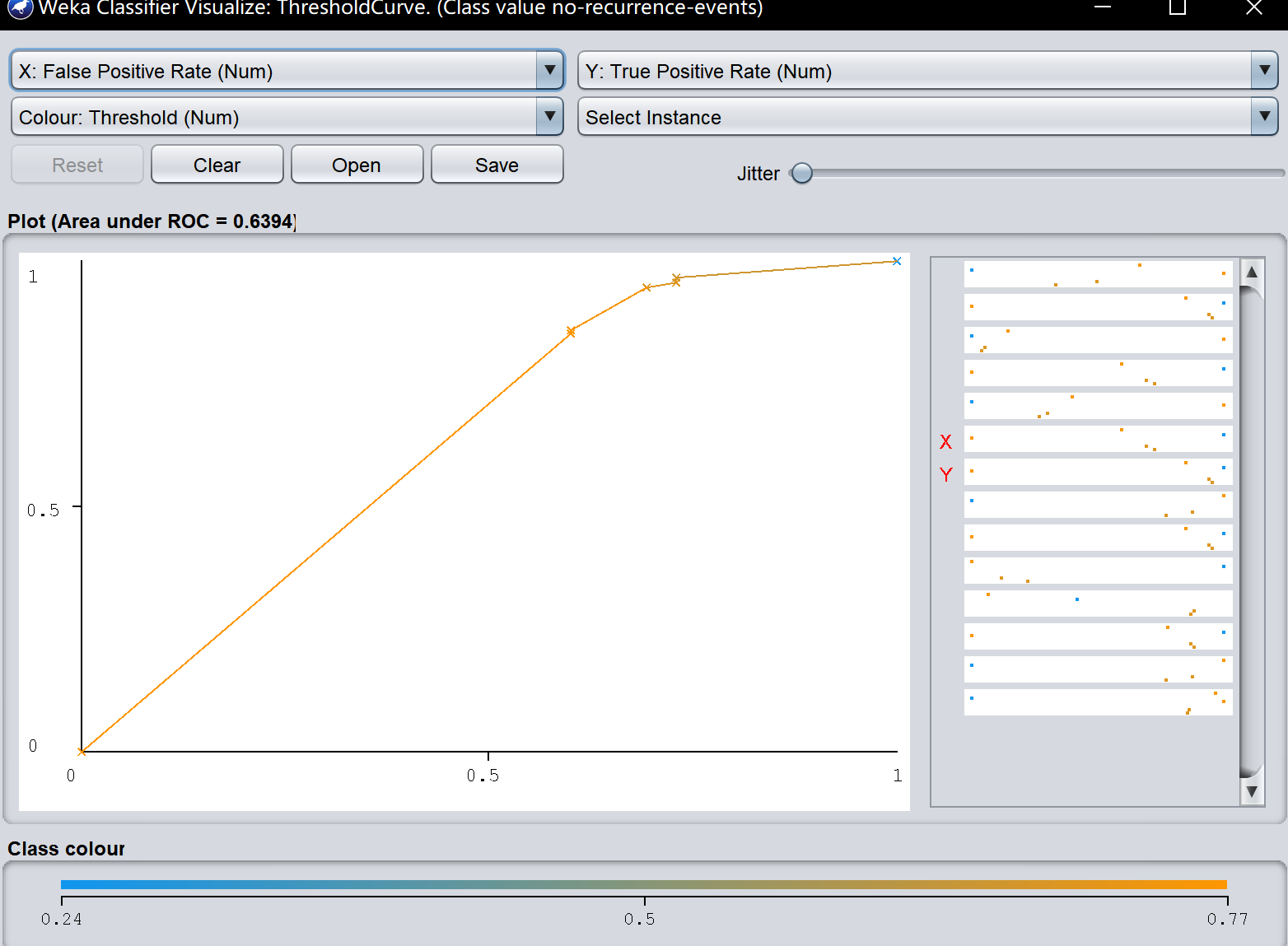
* Ajouter plus d'options et de criteres d'evaluation de classification.
* Visualiser graphiquement: les erreurs de classification, les courbes de threshold.

Par exemple:

Use *training Set,J48*

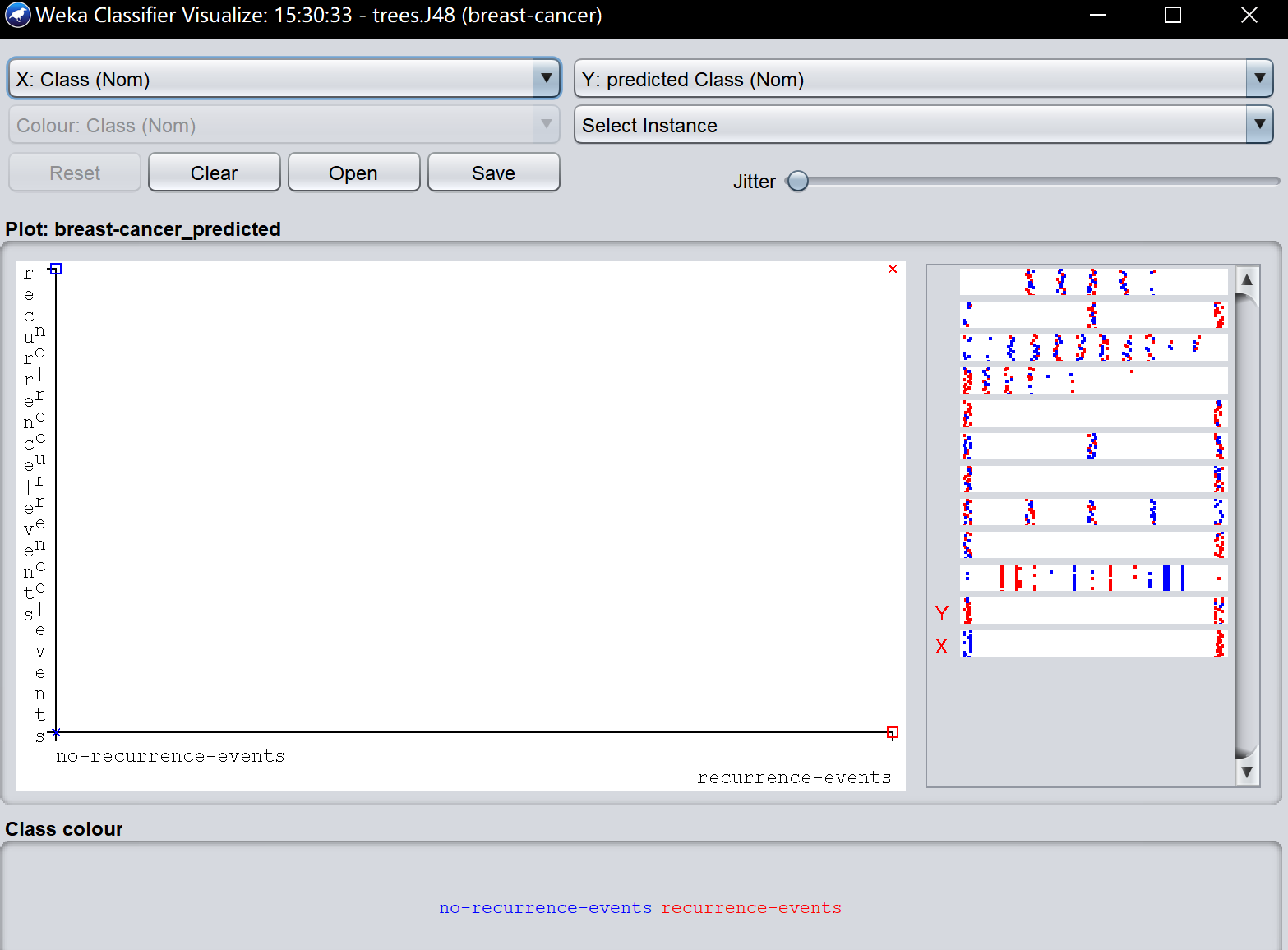


Classifier errors

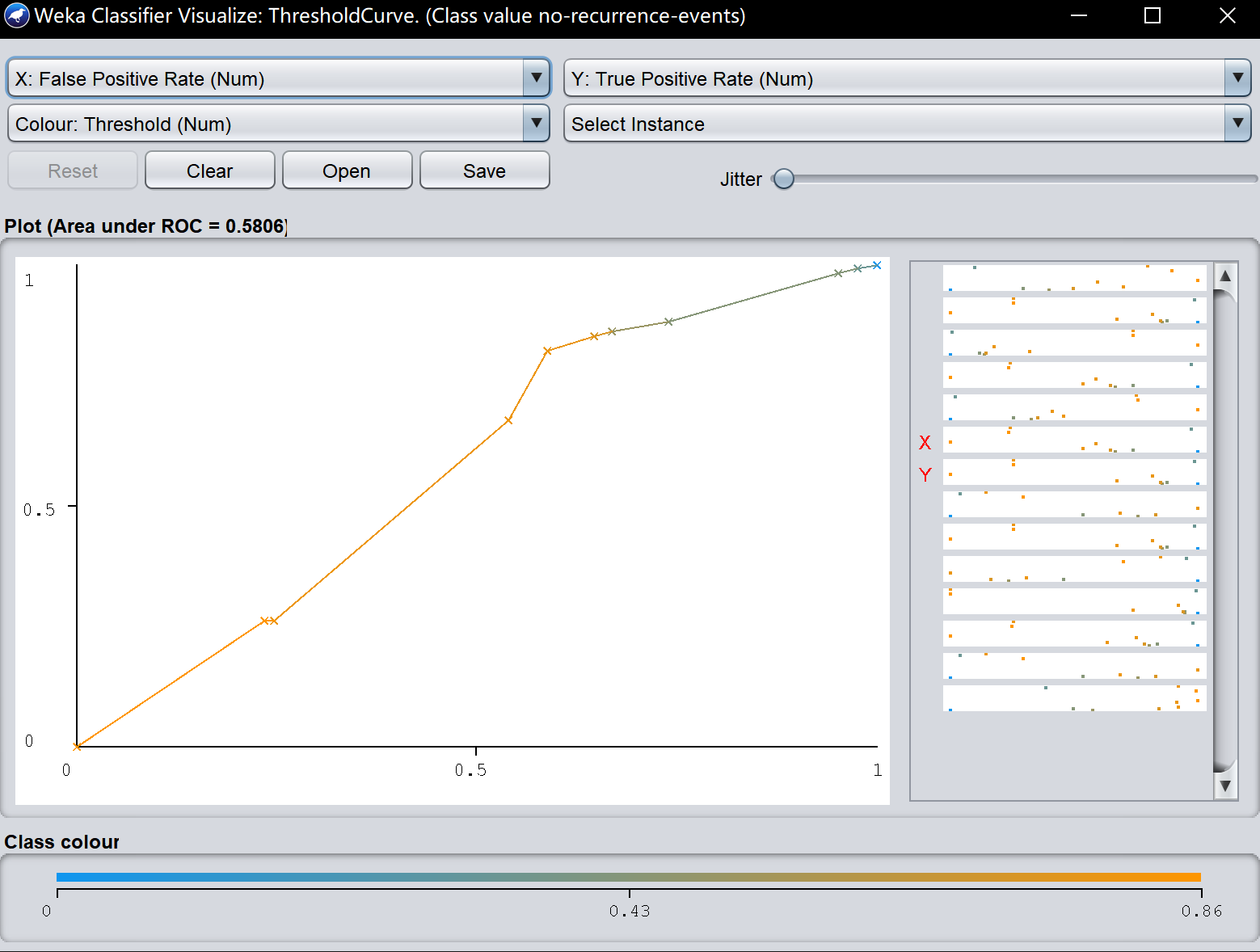


courbes de threshold

*2 Cross­validation,J48*

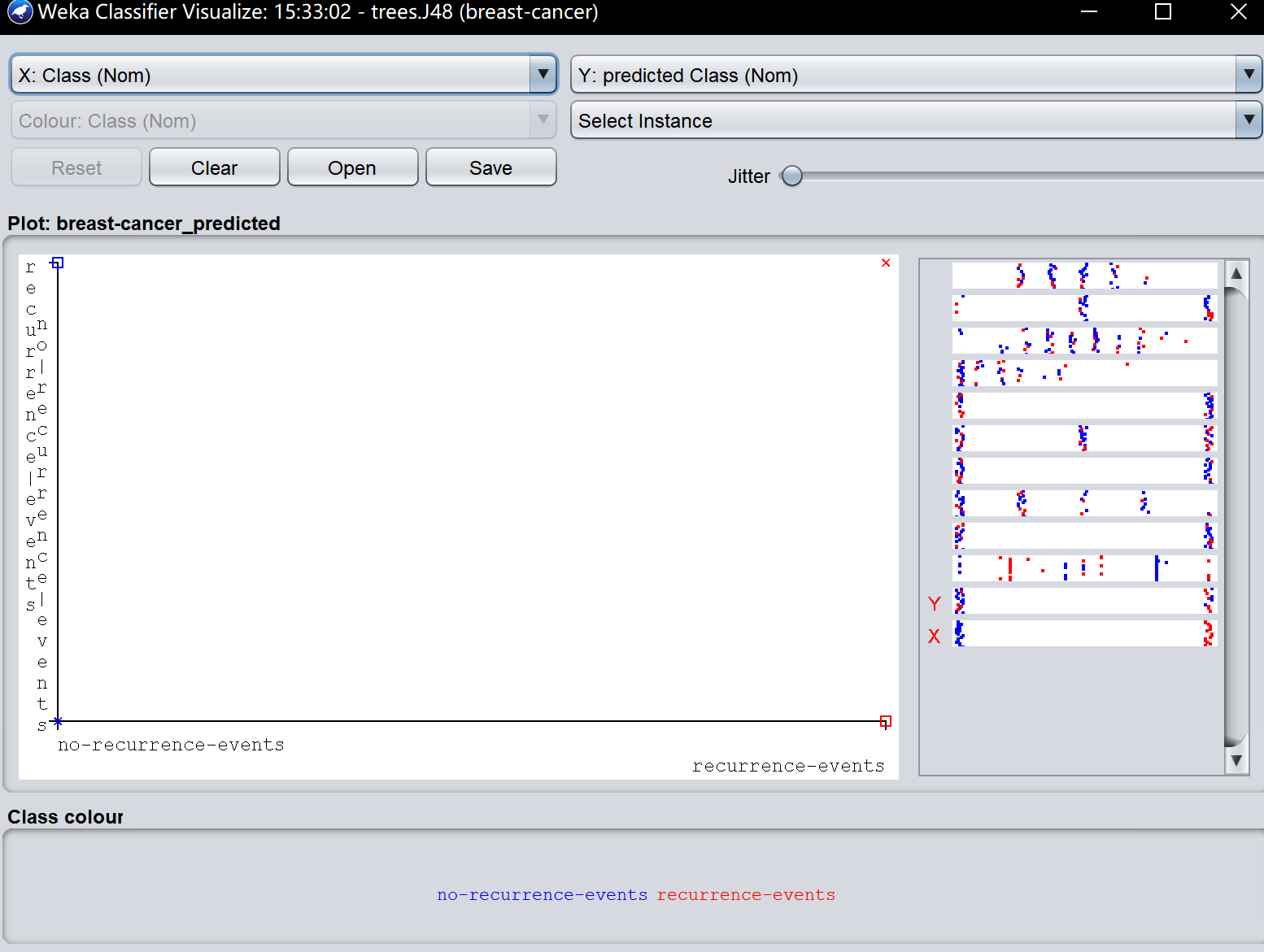


Classifier errors

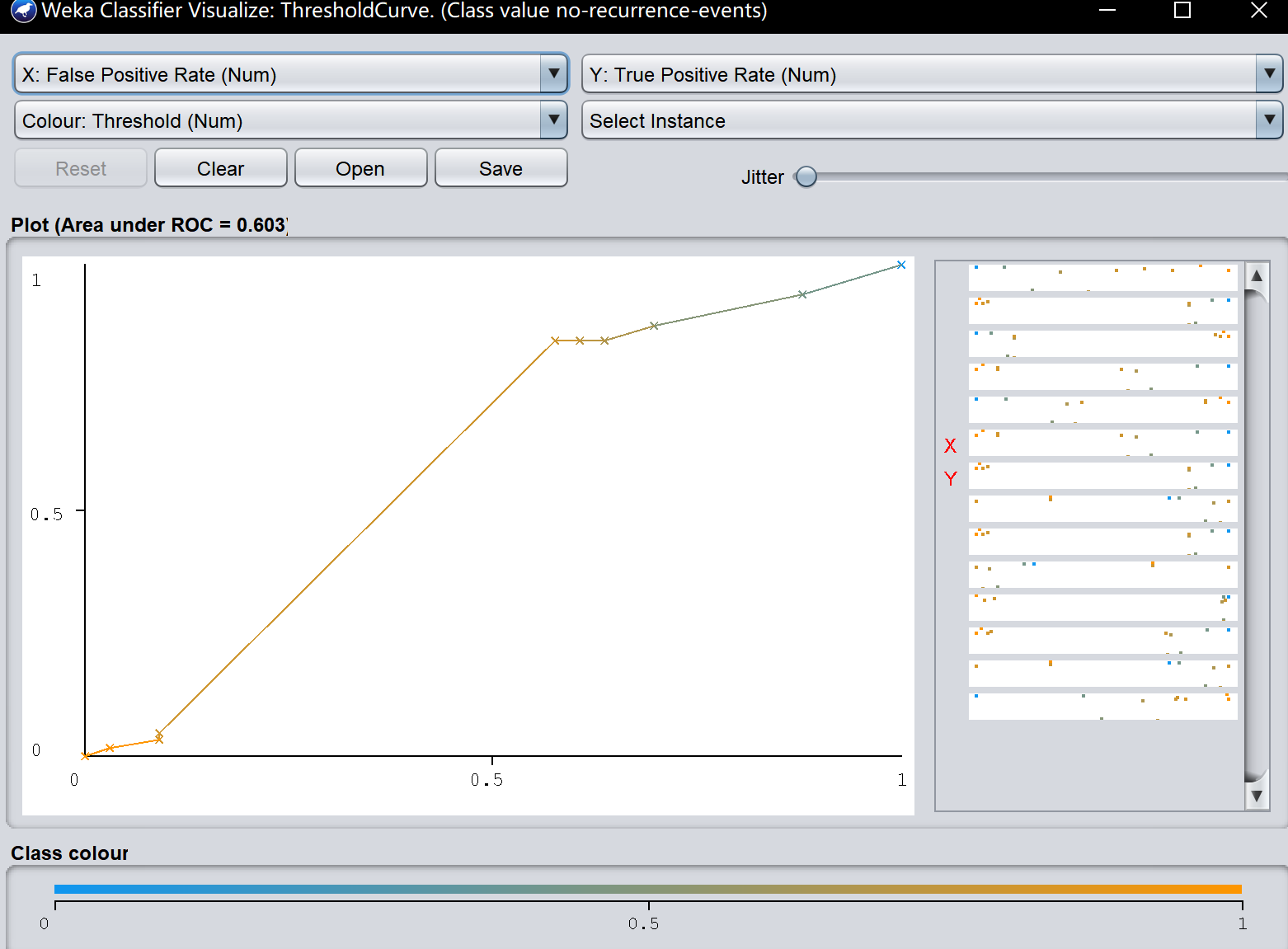


courbes de threshold

*66% Percentage Split,J48*



Classifier errors



courbes de threshold

— Quelles remarques peut-on faire par rapport aux resultats obtenus ?

On peux voir le TP Rate et FP Rate et Precision pour juge les classify

Comment des valeurs plus élevées de TP et de précision signifient une meilleure formation et des taux d'erreur plus faibles.

1. Dresser un tableau pour comparer les differentes approches. Interpreter les performances de chaque implementation d'arbre de decision.

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| Train Test | HoeffdingTree | J48 | LMT | RandomTree | REPTree | SimpleCart |
| TP | 0.866 0.482 | 0.965  0.271 | 0.95  0.318 | 0.995 0.941 | 0.995  0.141 | 1 0 |
| précision | 0.798 0.603 | 0.758  0.767 | 0.767  0.730 | 0.976 0.988 | 0.733  0.923 | 0.703 ？ |

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| Cross-validation 10 | HoeffdingTree | J48 | LMT | RandomTree | REPTree | SimpleCart |
| TP | 0.851 0.341 | 0.960  0.271 | 0.930  0.329 | 0.781 0.400 | 0.910  0.224 | 0.935  0.118 |
| précision | 0.753 0.753 | 0.757  0.742 | 0.766  0.667 | 0.755 0.436 | 0.735  0.514 | 0.715  0.435 |

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| Cross-validation 10 | HoeffdingTree | J48 | LMT | RandomTree | REPTree | SimpleCart |
| TP | 0.851 0.341 | 0.875  0.303 | 0.938  0.364 | 0.828 0.455 | 0.969  0.061 | 0.935  0.118 |
| précision | 0.753 0.753 | 0.709  0.556 | 0.741  0.750 | 0.746 0.577 | 0.667  0.500 | 0.715  0.435 |

Il a été observé que le modèle logistique et les algorithmes de forêt aléatoire donnent une précision très proche. Les algorithmes de modèle logistique d'arbre sont les plus performants de tous, mais lorsqu'on considère l'implantation en temps réel de ces algorithmes pour les prévisions futures, le temps nécessaire à la construction du modèle est une préoccupation. En comparaison avec le LMT et la forêt aléatoire, le temps nécessaire pour construire le modèle est vraiment moindre pour la forêt aléatoire. Si la précision n'est qu'une contrainte indépendamment du temps, alors le modèle LMT est le meilleur. On peut aussi considérer l'algorithme J48, si le temps est contraint, car il a nécessité moins de temps que les autres.

REPTree (Reduced-Error Pruning) est un autre algorithme spécifique à Weka. Il s'agit d'un arbre de décision à apprentissage rapide, optimisé pour la simplicité et la rapidité. L'algorithme utilise un élagage à erreur réduite avec réajustement pour trouver la plus petite représentation du sous-arbre le plus précis par rapport à l'ensemble d'élagage

**Exercice 2** (Challenge-RF)**.** Sous R et en utilisant la base de donnees "reading­Skills" depuis le package "Party", construisez une foret aleatoire de haute perfor­mance qui a pour but de bien classer les observations de la variable "nativeSpeaker" entre "yes" ou "no". Respecter les consignes suivantes :

1. Diviser votre base de donnees en 70% pour l'apprentissage et 30% pour le test en utilisant le parametre d'initialisation suivant "set.seed(1234)".
2. Vous etes libre a choisir les parametres de votre modele afin d'optimiser la performance.
3. Vous serez evalue sur les resultats de votre modele applique aux donnees test. Ainsi, vous etes invites a mettre, dans le compte rendu, la courbe ROC et l'AUC de votre modele ainsi que le code sous R.

Les parametres adin d’optimiser la performance : mtry = 2, ntree = 500

文本, 信件

描述已自动生成

图表, 折线图

描述已自动生成

Le code sous R:

library(caret)

library(e1071)

data("readingSkills")

print(head(readingSkills))

#diviser le base de données et set seed(1234)

set.seed(1234)

smp\_size <- floor(0.7\*nrow(readingSkills))

train\_ind <- sample (seq\_len(nrow(readingSkills)),size = smp\_size)

train <- readingSkills[train\_ind,]

test <- readingSkills[-train\_ind,]

nrow(train)

nrow(test)

#choisir les paramètres

ranf=train(x=train[,names(train)!="nativeSpeaker"],y=train$nativeSpeaker, data=train,method="rf",importance=T)

print(ranf)

#évalué les résultats et la courbe ROC et l'AUC

pre\_ran <- predict(fit.forest, newdata = test\_data)

table(pre\_ran, test\_data$nativeSpeaker, dnn = c("oberservations","predictions"))

# modele optimal

print(ranf$finalModel)

# variable importnce des Forets aleatoires

varImp(ranf)

plot(varImp(ranf))

#matrice de confusion partie test et erreur de classement.

prf=predict.train(ranf,newdata = test)

mrf=table(test$nativeSpeaker, prf, dnn=c("reelle", "Predicted"))

mrf

err=1-((mrf[1,1]+mrf[2,2])/sum(mrf))

err

#Courbe Roc et AUC des forets par Caret

library(pROC)

prf1=predict(ranf,test, type="prob")

RCad= roc(test$nativeSpeaker, prf1[,2])

rocAD = plot.roc(test$nativeSpeaker, prf1[,2],

main = " ROC de Random Forest avec Caret",

type="l", col="blue", grid=c(0.1,0.2),print.auc = TRUE)# print the AUC (will contain the CI)

**Exercice 3** (ML-RF)**.** Dans cet exercice, il s'agit d'experimenter plusieurs imple­mentation du foret d'arbres de decision.

1. Apartir du repertoire en ligne *nEchantillons de donnees'"* de *VENT,* considerez le meme echantillon de donnees [[3]](#footnote-2) de l'exercice 1. Utiliser les differentes implementations du foret d'arbres de decision sous Weka a savoir :

—**CSForest**

—**ForestPA**

— **RandomForest**

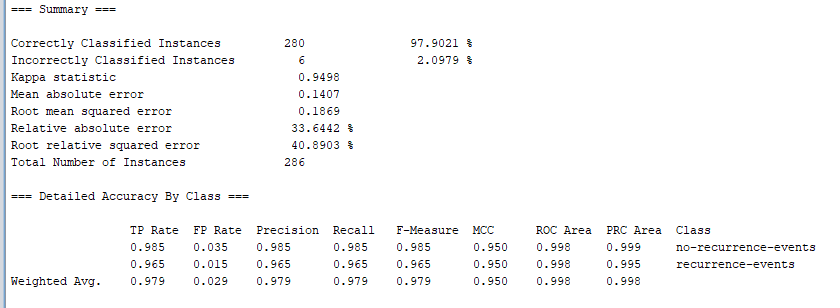
— **SmoothPrivateForest**

— **SysFor**

Pour chaque implementation :

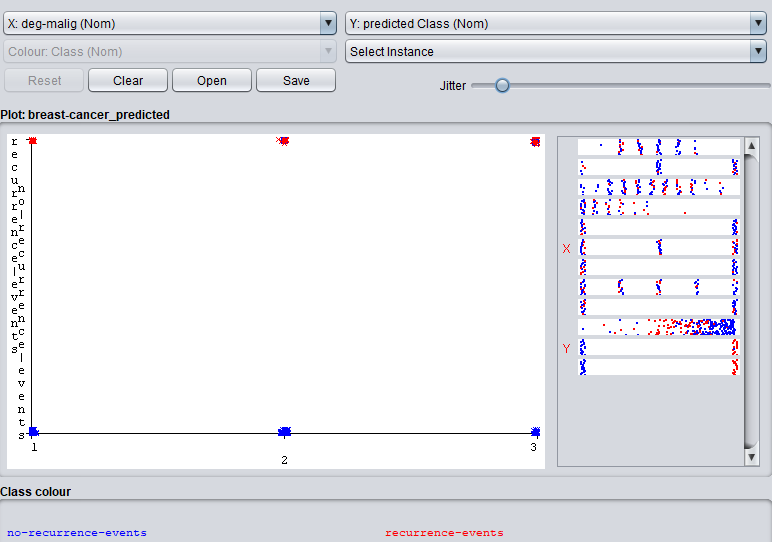
* Faites varier les techniques d'evaluation *(Use training Set, 2 and 10 Cross­validation, 66% Percentage Split*).
* Ajouter plus d'options et de criteres d'evaluation de classification.

Nous prenons RandomForest comme exemple,de nombreux indicateurs d'évaluation sont fournis.

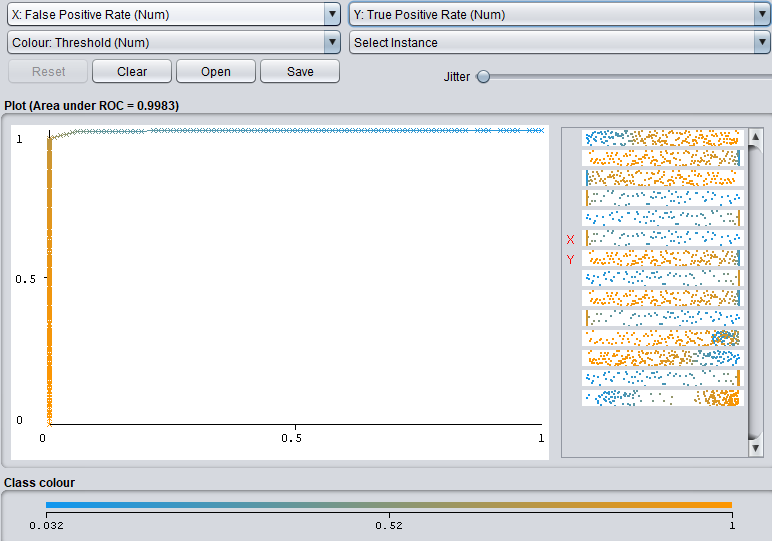


* Visualiser graphiquement: les erreurs de classification, les courbes de threshold.

Visualiser graphiquement pour Use training set (Nous prenons RandomForest comme exemple):

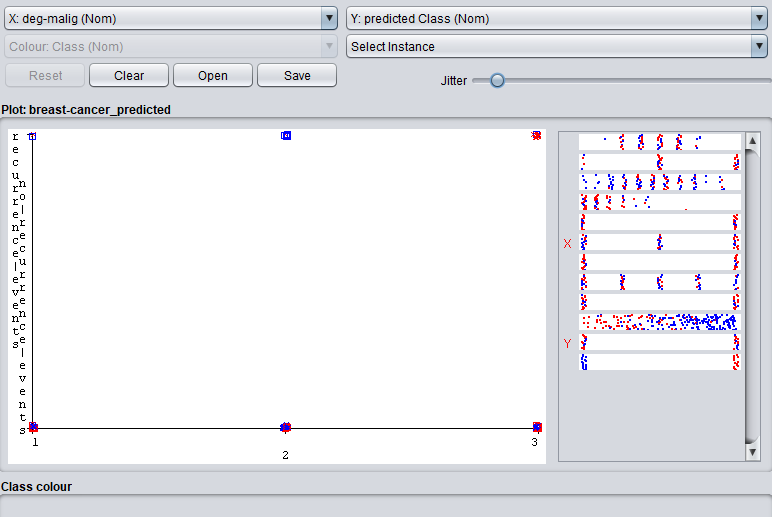


le erreur de classification de deg-malig à prédite classe

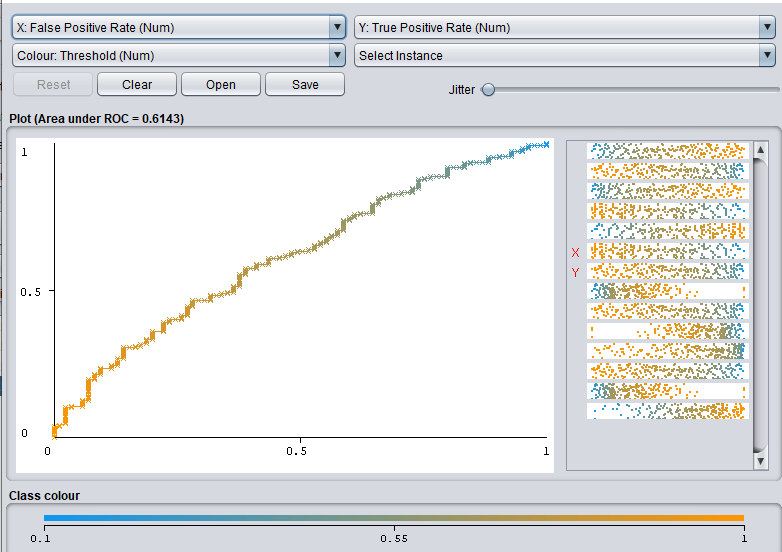


le courbe de threshold de TP Rate à FP Rate(class value no-recurrence-events)

Visualiser graphiquement pour *2 Crossvalidation*  (Nous prenons RandomForest comme exemple):

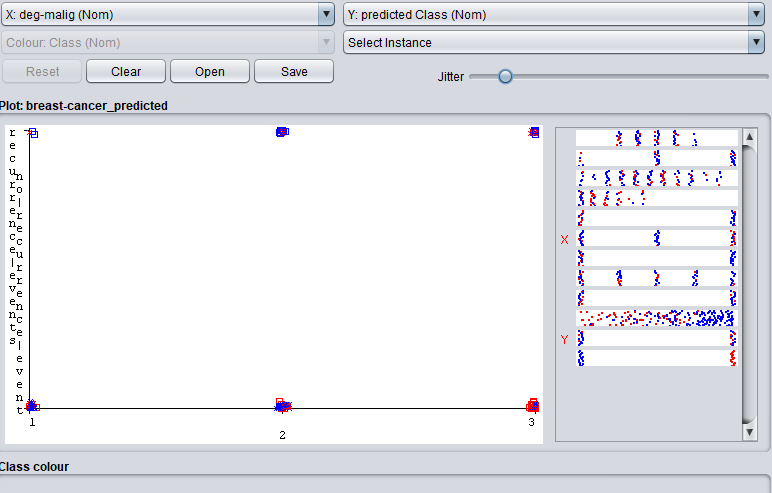


le erreur de classification de deg-malig à prédite classe

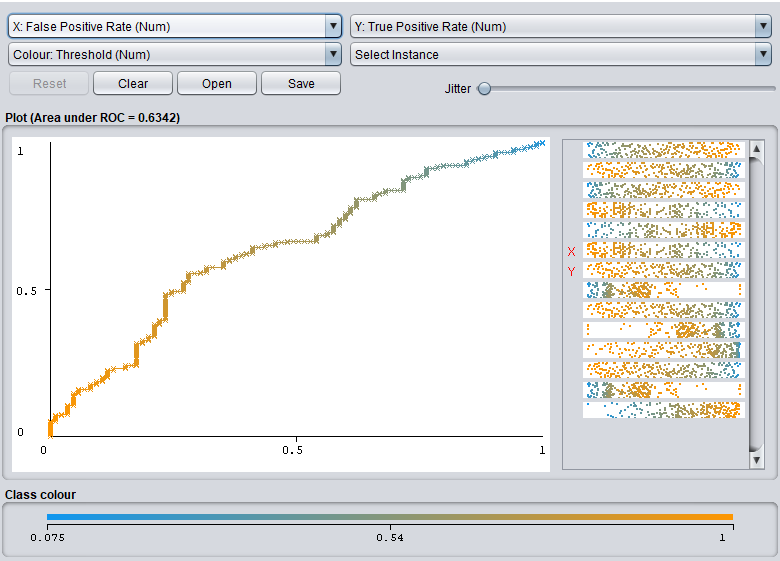


le courbe de threshold de TP Rate à FP Rate(class value no-recurrence-events)

Visualiser graphiquement pour 10 *Crossvalidation*  (Nous prenons RandomForest comme exemple):

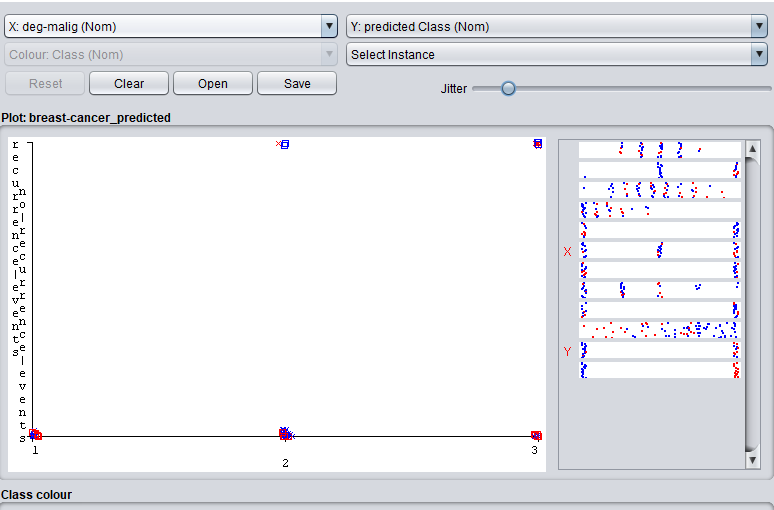


le erreur de classification de deg-malig à prédite classe

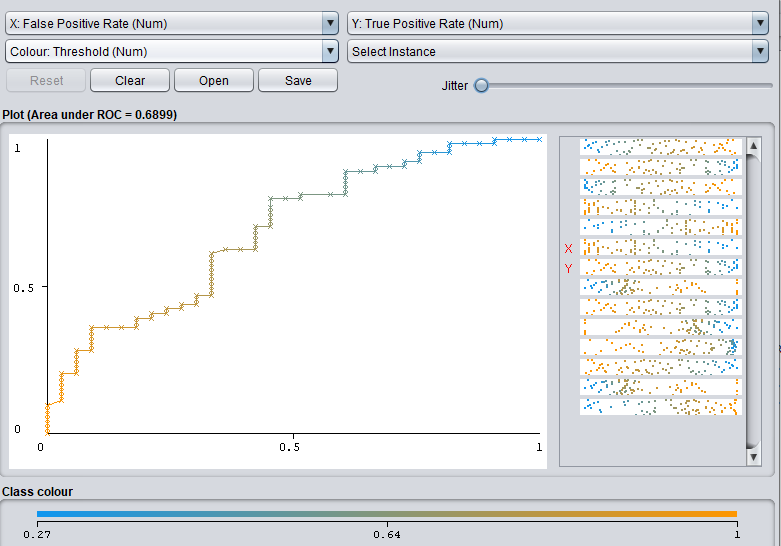


le courbe de threshold de TP Rate à FP Rate(class value no-recurrence-events)

Visualiser graphiquement pour 66% Percentage Split (Nous prenons RandomForest comme exemple):



le erreur de classification de deg-malig à prédite classe



le courbe de threshold de TP Rate à FP Rate(class value no-recurrence-events)

* Quelles remarques peut-on faire par rapport aux resultats obtenus ?

Ici, nous choisissons également TP Rate, FP Rate et Precision pour évaluer les résultats de la régression. TP Rate , FP Rate et Precision, plus les deux valeurs sont élevées, meilleur est l'effet de classification.

1. Dresser un tableau pour comparer les differentes approches. Interpreter les performances de chaque implementation de foret aleatoire.

Use training Set

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Train Test | CSForest | ForestPA | RandomForest | SmoothPrivateForest | SysFor |
| TP | 0.234 0.918 | 0.970  0.506 | 0.985  0.965 | /  / | 0.945  0.271 |
| précision | 0.870 0.336 | 0.823  0.878 | 0.985  0.965 | /  / | 0.754  0.676 |

2 Crossvalidation

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Train Test | CSForest | ForestPA | RandomForest | SmoothPrivateForest | SysFor |
| TP | 0.164 0.847 | 0.896  0.200 | 0.876  0.259 | /  / | 0.841  0.341 |
| précision | 0.717 0.300 | 0.726  0.447 | 0.736  0.468 | /  / | 0.751  0.475 |

10 Crossvalidation

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Train Test | CSForest | ForestPA | RandomForest | SmoothPrivateForest | SysFor |
| TP | 0.189 0.812 | 0.910  0.271 | 0.871  0.282 | /  / | 0.965  0.094 |
| précision | 0.704 0.297 | 0.747  0.561 | 0.742  0.480 | /  / | 0.716  0.533 |

66% Percentage Split

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Train Test | CSForest | ForestPA | RandomForest | SmoothPrivateForest | SysFor |
| TP | 0.109 0.758 | 0.953  0.182 | 0.906  0.303 | /  / | 0.828  0.394 |
| précision | 0.467 0.305 | 0.693  0.667 | 0.716  0.625 | /  / | 0.726  0.542 |

Si TP Rate et la précision sont utilisés comme indicateurs complets pour évaluer les résultats, d'un point de vue macro, nous évaluons grossièrement la performance globale dans les quatre situations ci-dessus, RandomForest a les meilleures performances.

Les algorithmes SysFor et CSForest sont exceptionnels dans certaines situations spécifiques (comme SysFor dans le cas de 10 Crossvalidation et CSForest in Use Train Test), mais en même temps, cela apportera un autre mauvais résultat de classement. Le résultat est quelque peu instable.

ForestPA est généralement légèrement pire que RandomForest, sauf dans certaines circonstances particulières, comme lorsque nous définissons 10 Crossvalidation.

En termes d'événements de modélisation, l'algorithme SysFor prend le temps le plus court. ForestPA prend le plus de temps. D'autres ne sont pas très différents. Lors de la modélisation de données à grande échelle, nous devons éviter d'utiliser ForestPA.

SmoothPrivateTree. Il s'agit d'une forêt décisionnelle privée différenciée conçue pour minimiser le nombre de requêtes requises et la sensibilité de ces requêtes. Il semble que notre ensemble de données ne prend pas en charge cet algorithme. Il ne peut pas courir.

1. Ce travail demande doit etre demarre durant la seance de TP, a terminer chez soi pour etre remis a votre enseignant avant le 04/12/2020 a 23h55. A remettre dans l'espace de depot un dossier numerique contenant le(s) fichier(s)de donnees generees + un compte-rendu detaillant le travail realise et justifiant les resultats obtenus. [↑](#footnote-ref-0)
2. Le choix de l'echantillon de donnees doit etre valide par votre enseignant [↑](#footnote-ref-1)
3. Le choix de l'echantillon de donnees doit etre valide par votre enseignant tuteur [↑](#footnote-ref-2)