100% ACTUAIRES &
100% DATA SCIENCE

INSTITUT DES ACTUAIRES

Vers un modèle de tarification ML interprétable : Modèle de fréquence basé sur des données telematics

Arthur MAILLART : Université Lyon 1/Forsides
Christian ROBERT : ENSAE/Chaire DAMI

29 / NOV / 2019

Hôtel Marriott Rive Gauche
Paris 14ème

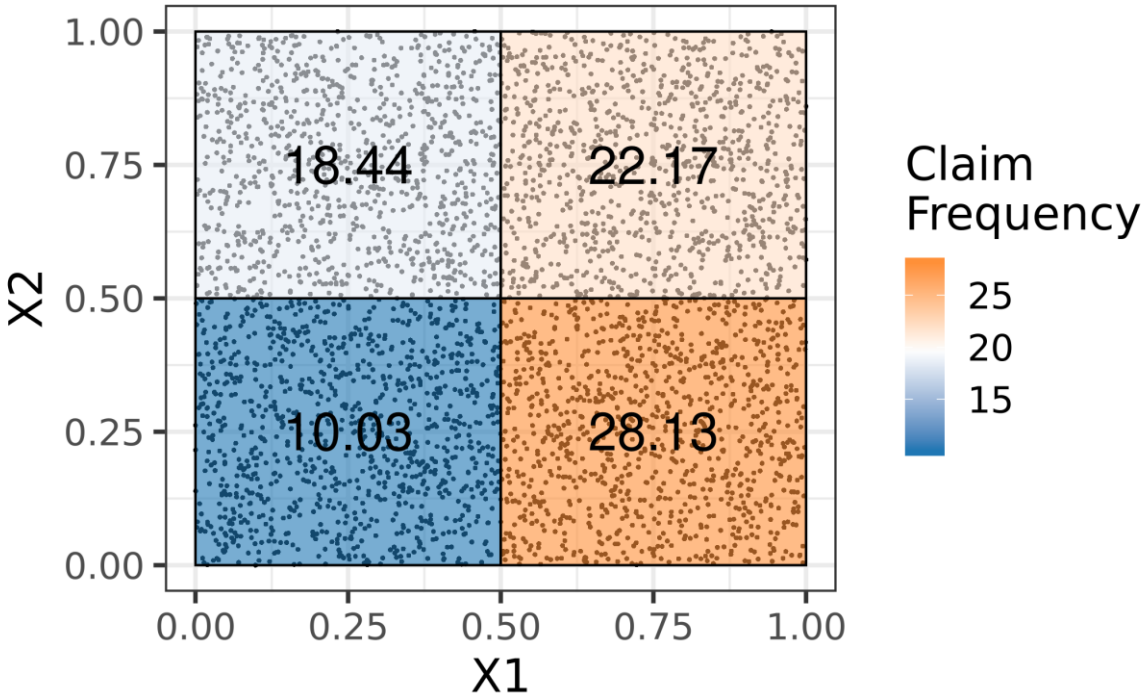
1. Modèle de fréquence sinistre
2. Améliorer la connaissance de ses données grâce aux arbres
3. Extraire un ensemble parcimonieux de règles
4. Use case
5. Résultats
6. Bibliographie

En modélisant la fréquence sinistre, on cherche à créer des groupes homogènes de risques que l'on appelle cellules tarifaires [Ohlsson and Johansson, 2010].

Ces dernières doivent être suffisamment larges pour qu'il y ait une mutualisation mais suffisamment différenciées pour attirer certains prospects.

Ohlsson, E. and Johansson, B. (2010).
Non-life insurance pricing with generalized linear models, volume 2.
Springer.

Pour représenter notre problème en deux dimensions et illustrer les méthodes qui vont suivre, nous créons un jeu de données simulées.



Nous avons un jeu de données *telematics* collectées dans le cadre d'une assurance PAYD et nous voulons ajuster un modèle de fréquence. Une méthode classique en actuariat consiste à ajuster un modèle GLM pour obtenir ce modèle de fréquence. Cependant, lorsqu'il y a des variables continues dont la relation avec la fréquence sinistre est non-linéaire, il devient nécessaire de discrétiser la variable explicative.

- Souvent, l'ajustement de ces modèles nécessite de la "connaissance d'expert" pour découper correctement ces variables.
- Il n'y a aucune garantie que le découpage est optimal.
- Cette opération peut être très longue selon le nombre de colonnes.

- Les méthodes de Machine Learning telles que Random Forest ou XGBoost constituent de bonnes heuristiques d'apprentissage pour des données tabulaires. Ce qui signifie qu'elles performent généralement bien sur ce genre de données.

- Elles n'ont pas besoin qu'on leur spécifie une structure a priori et prennent naturellement en compte des interactions entre les variables.

- Elles ne requièrent pas trop de paramétrage.

Seule ombre au tableau, la chaîne de décision de ces méthodes est difficile voire impossible à appréhender pour un humain. Elles sont donc difficiles à expliquer à un public non-expert.

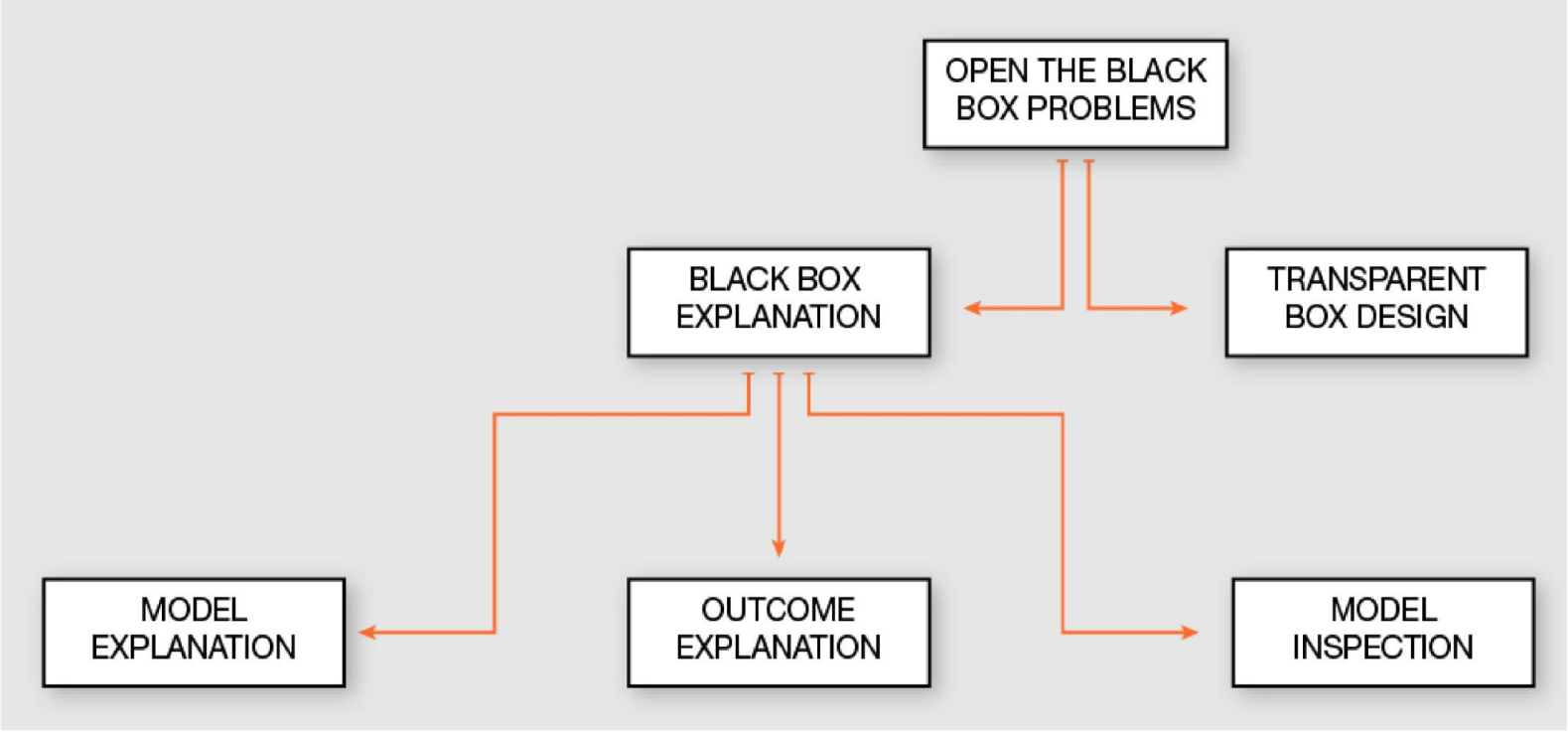
Selon [Guidotti et al., 2018], interpréter (en Machine Learning) signifie fournir une explication ou le sens d'un concept en des termes compréhensibles par l'humain.

Une explication est une interface entre l'humain et la machine. Cette interface doit

- être un bon proxy du modèle Black-Box
- être compréhensible pour l'humain.

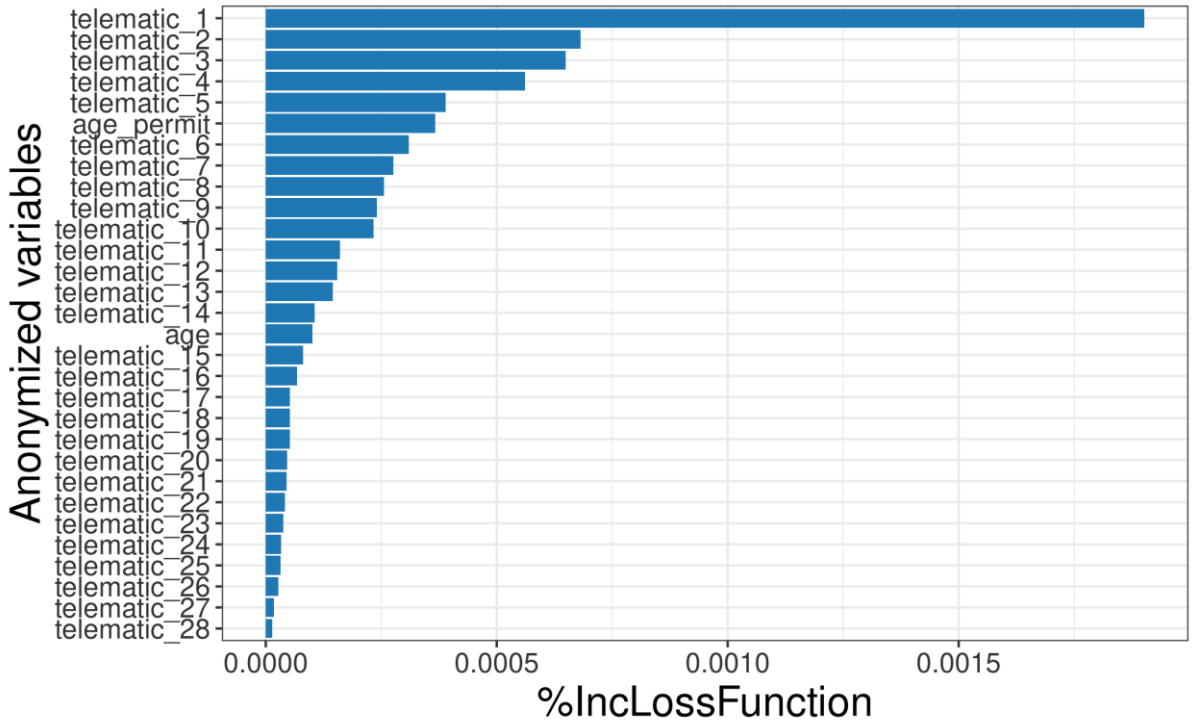
Guidotti, R., Monreale, A., Ruggieri, S., Turini, F., Giannotti, F., and Pedreschi, D. (2018).
A survey of methods for explaining black box models.
ACM computing surveys (CSUR), 51(5):93.

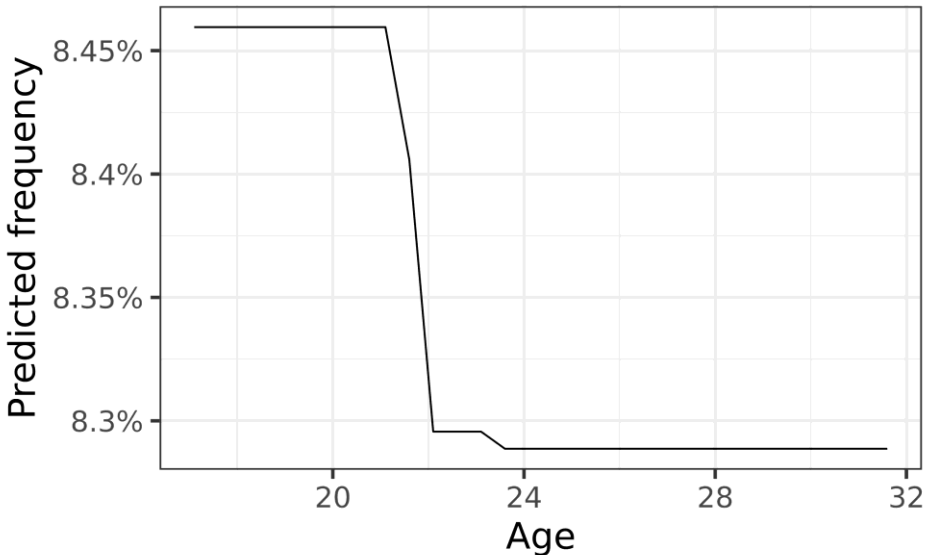
De cela découle 4 types de problèmes que l'on cherche à résoudre pour rendre les modèles interprétables.

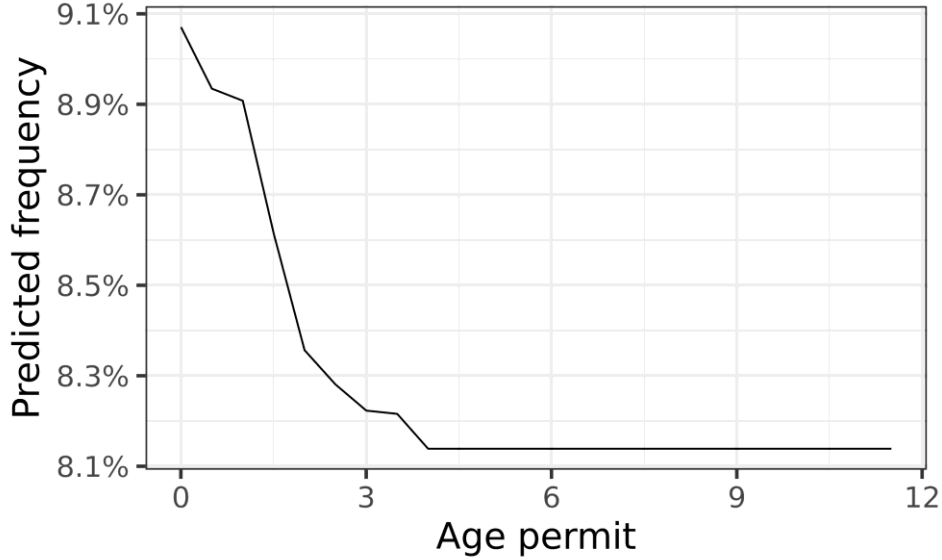


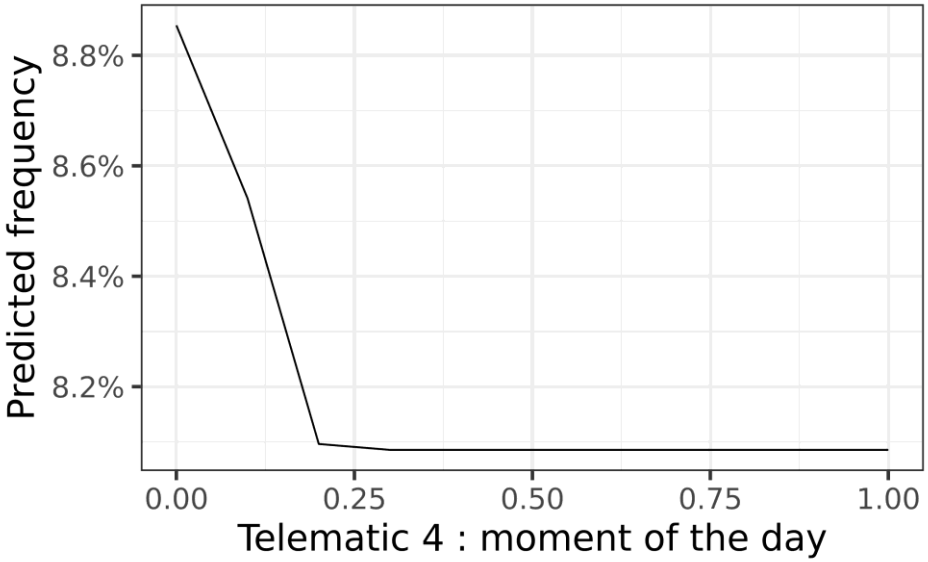
Source : [Guidotti et al., 2018]

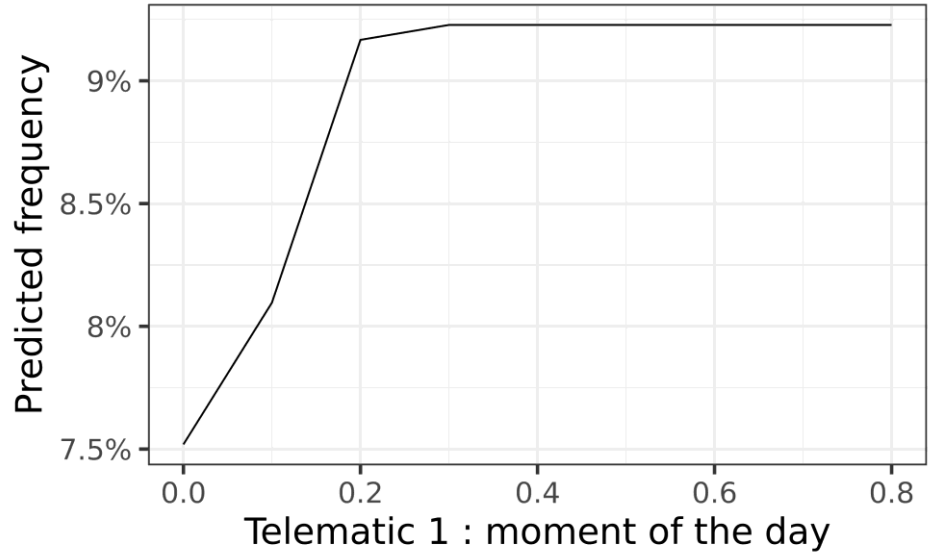
Les méthodes bien connues telles que la "Variable Importance" ou les "Partial Dependence Plots" apportent de l'information mais celle-ci est insuffisante.

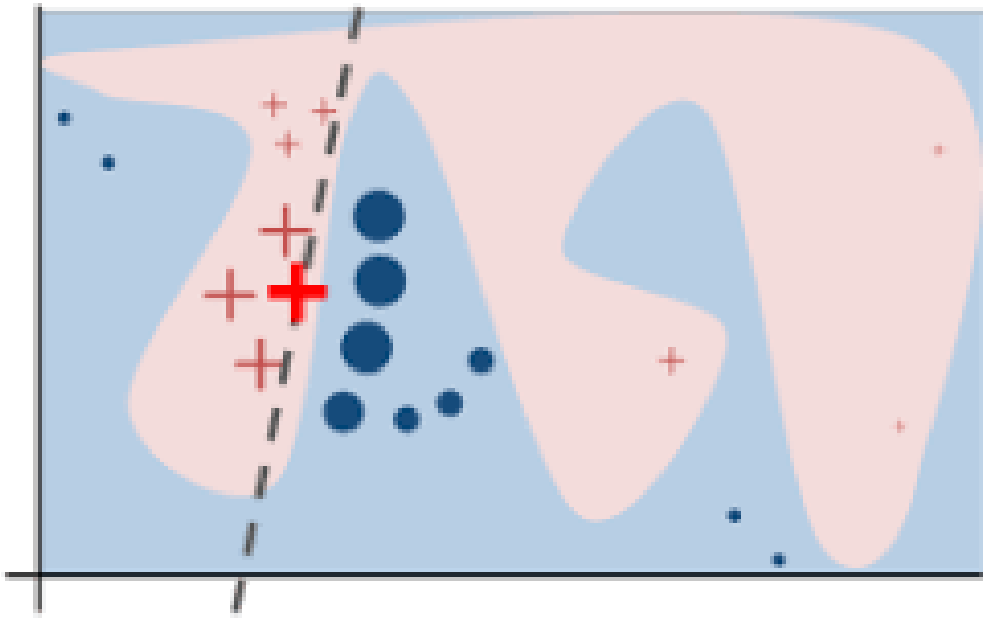



Predicted frequency
8.45%
8.4%
8.35%
8.3%
20
24
28
32
Age

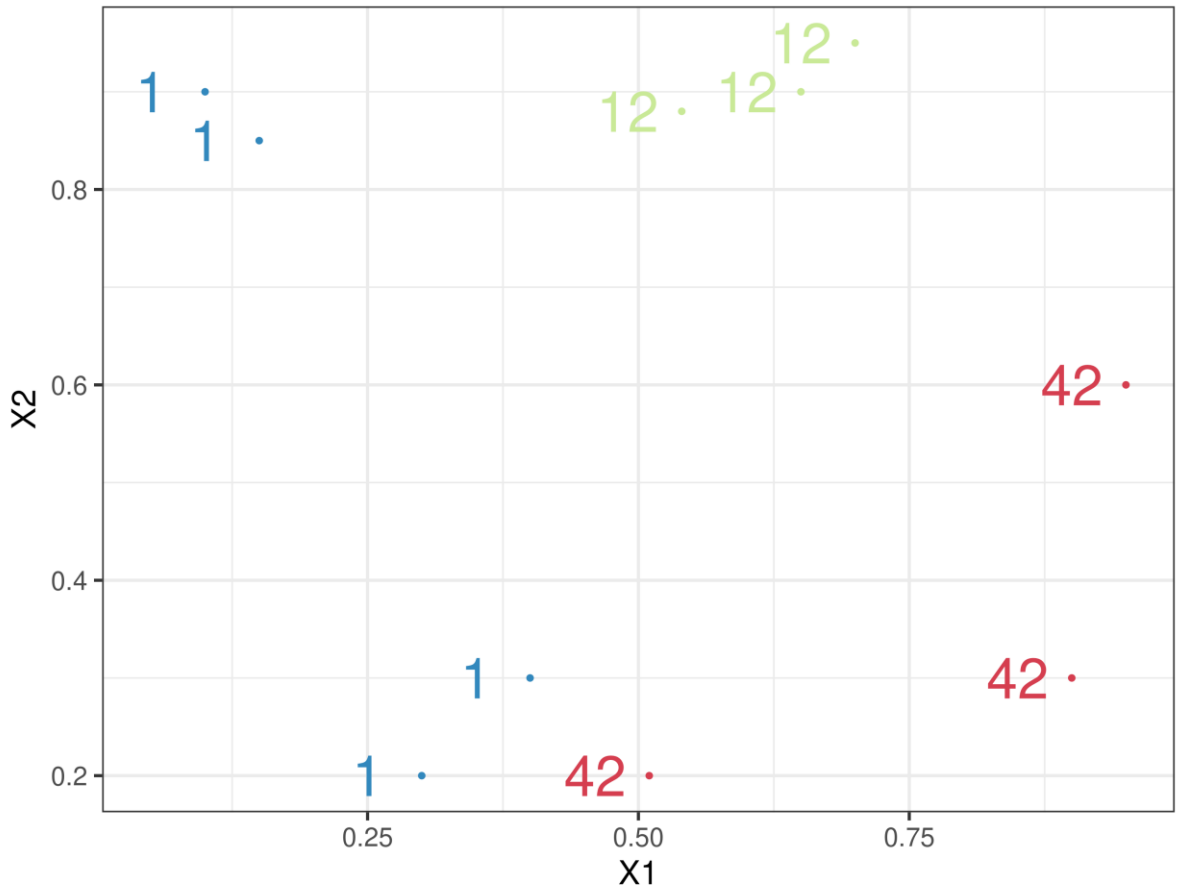

Predicted frequency
9.1%
8.9%
8.7%
8.5%
8.3%
8.1%
0
3
6
9
12
Age permit

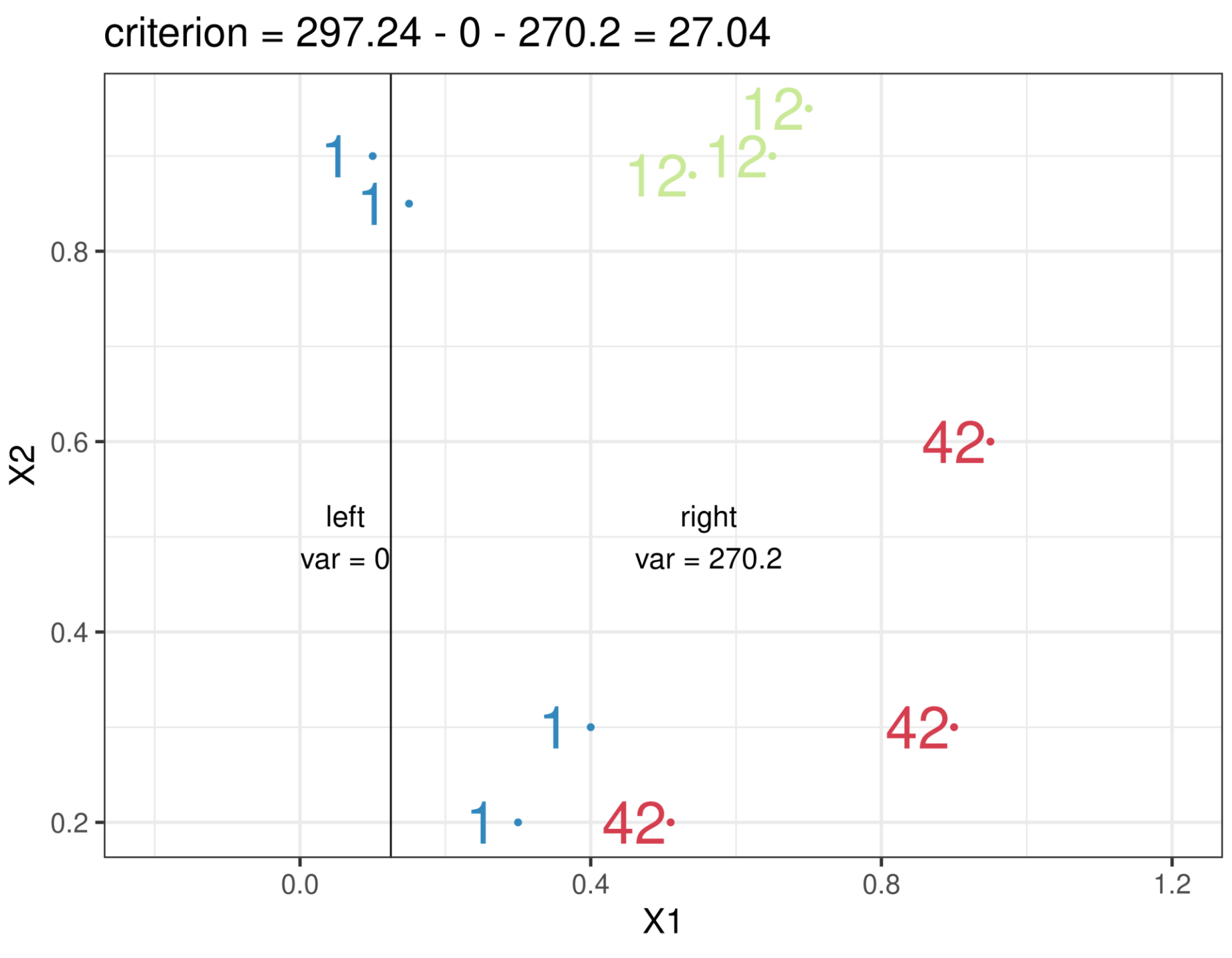

Predicted frequency
8.8%
8.6%
8.4%
8.2%
0.00
0.25
0.50
0.75
1.00
Telematic 4 : moment of the day

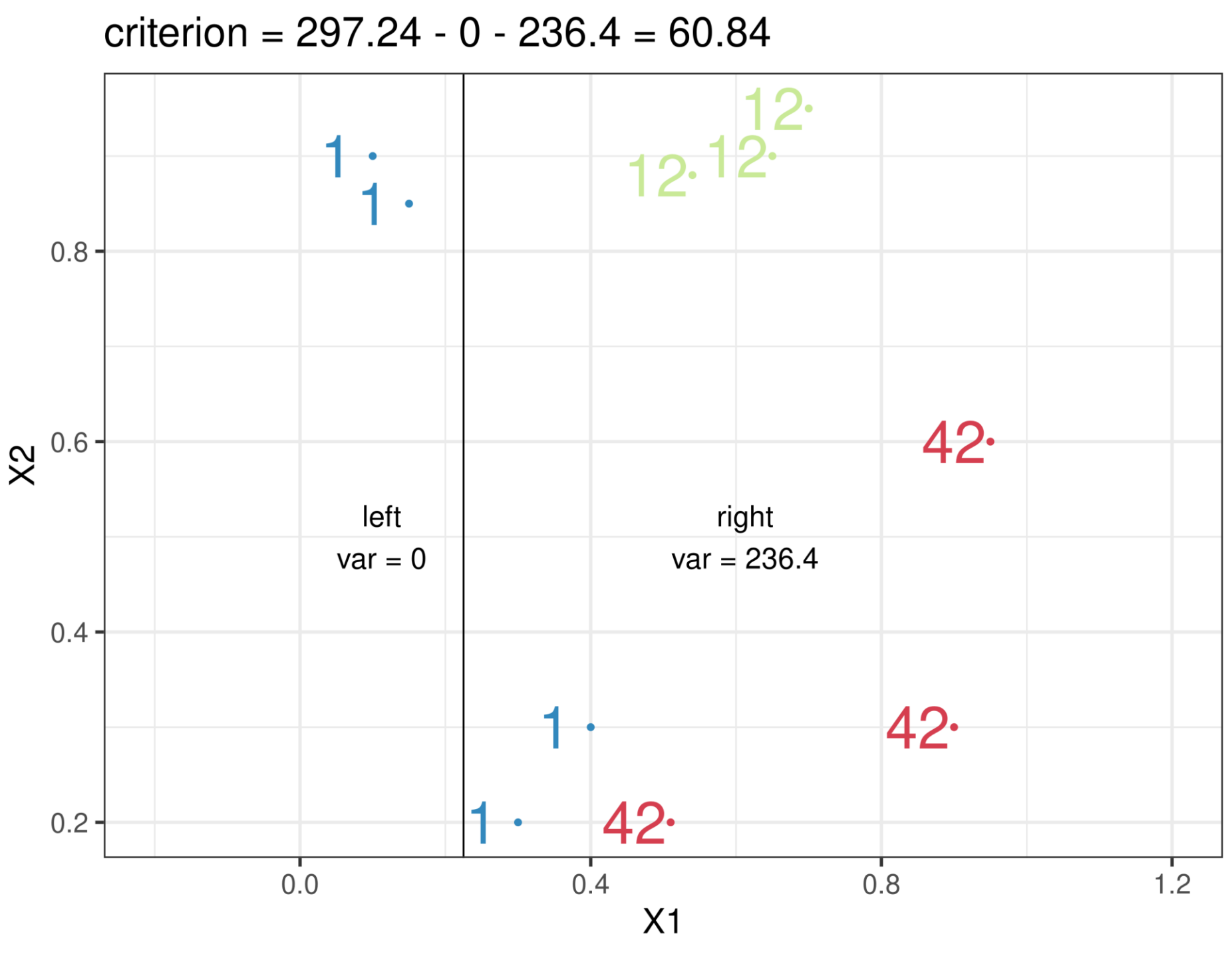

Predicted frequency
9%
8.5%
8%
7.5%
0.0
0.2
0.4
0.6
0.8
Telematic 1 : moment of the day

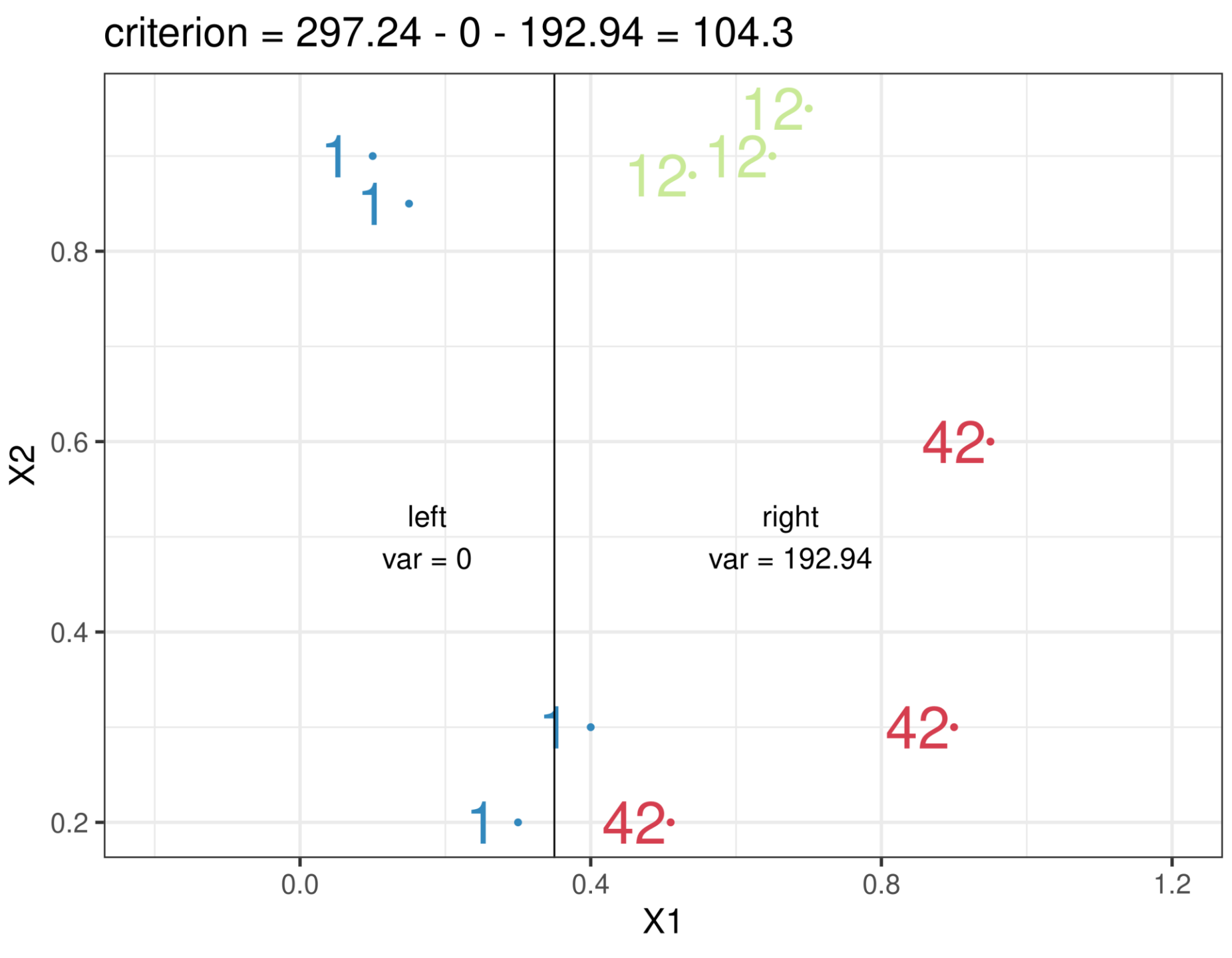


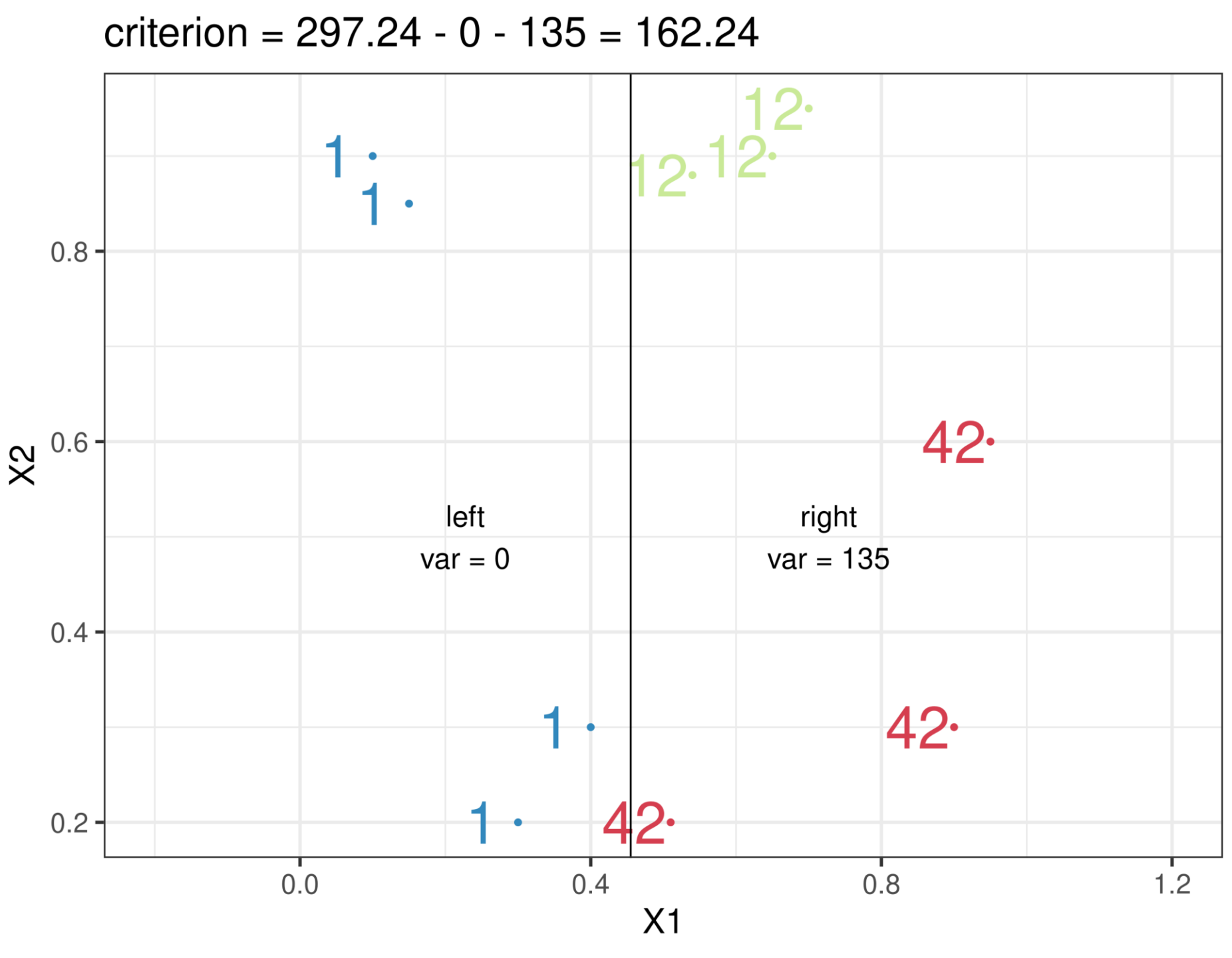
Source : https://arxiv.org/pdf/1602.04938.pdf

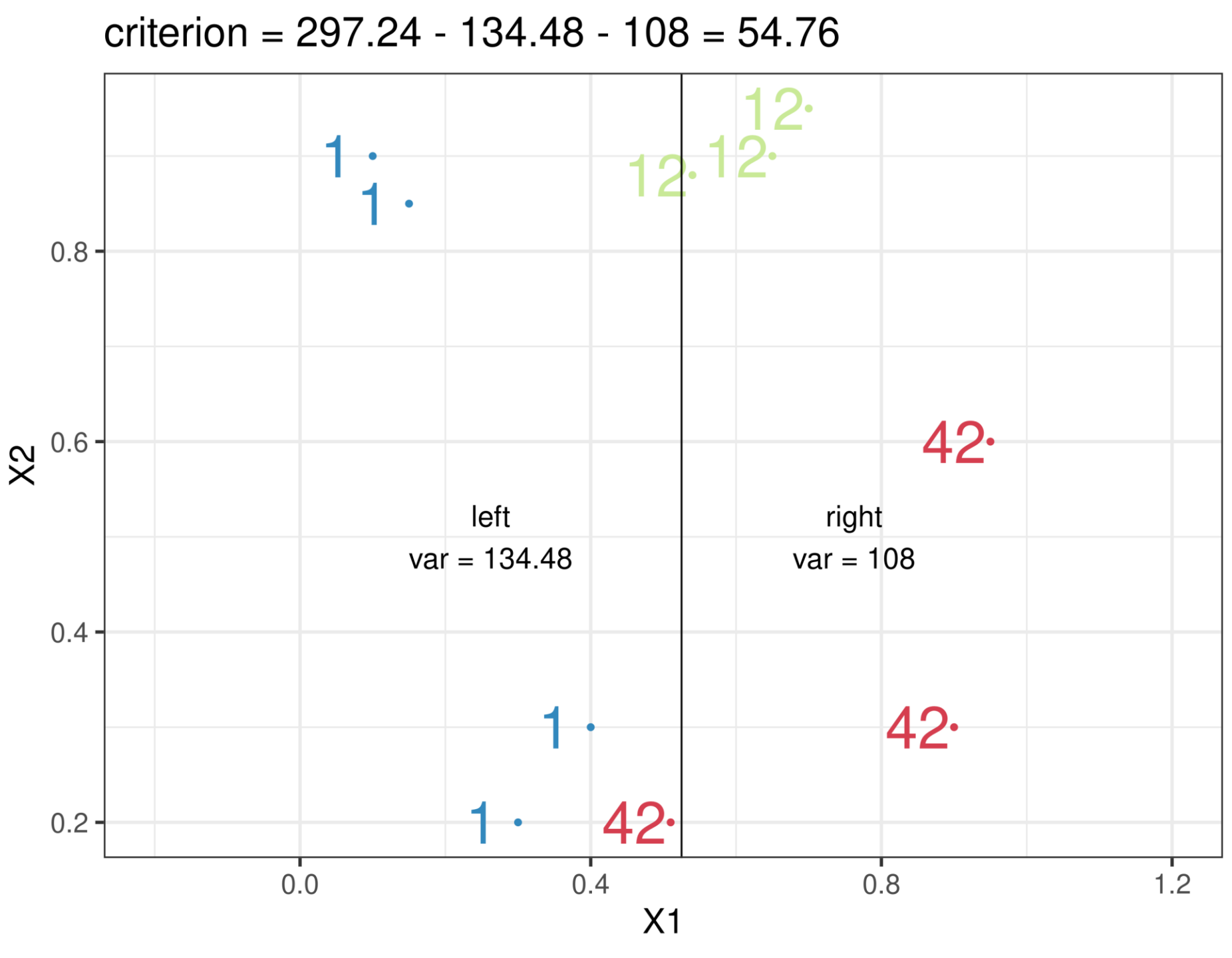

1
1
12
12
12
42
1
42
1
42
0.8
0.6
0.4
0.2
X2
0.25
0.50
0.75
X1

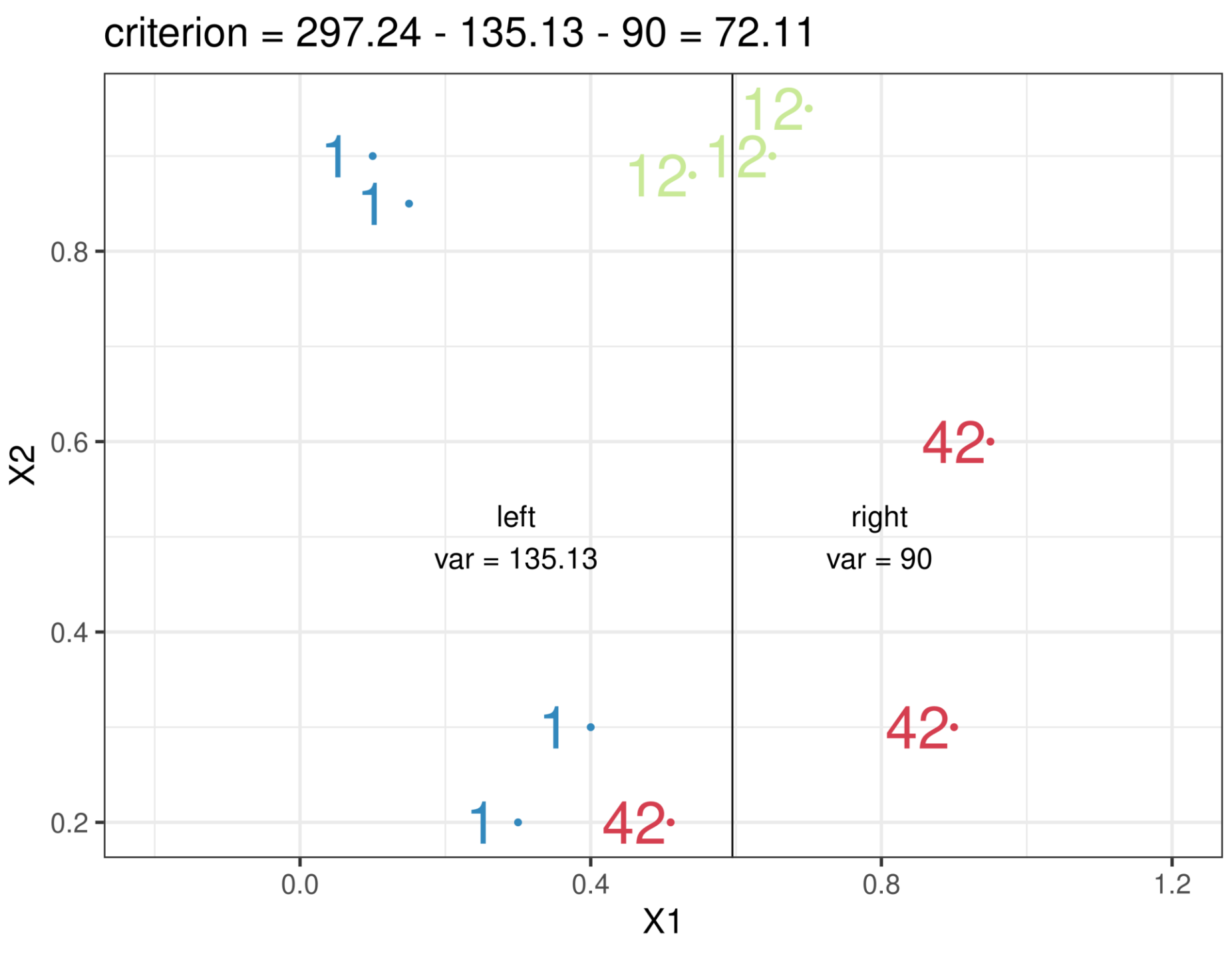

criterion = 297.24 - 0 - 270.2 = 27.04
left
var = 0
right
var = 270.2
X1
X2
0.0
0.4
0.8
1.2
0.2
0.4
0.6
0.8

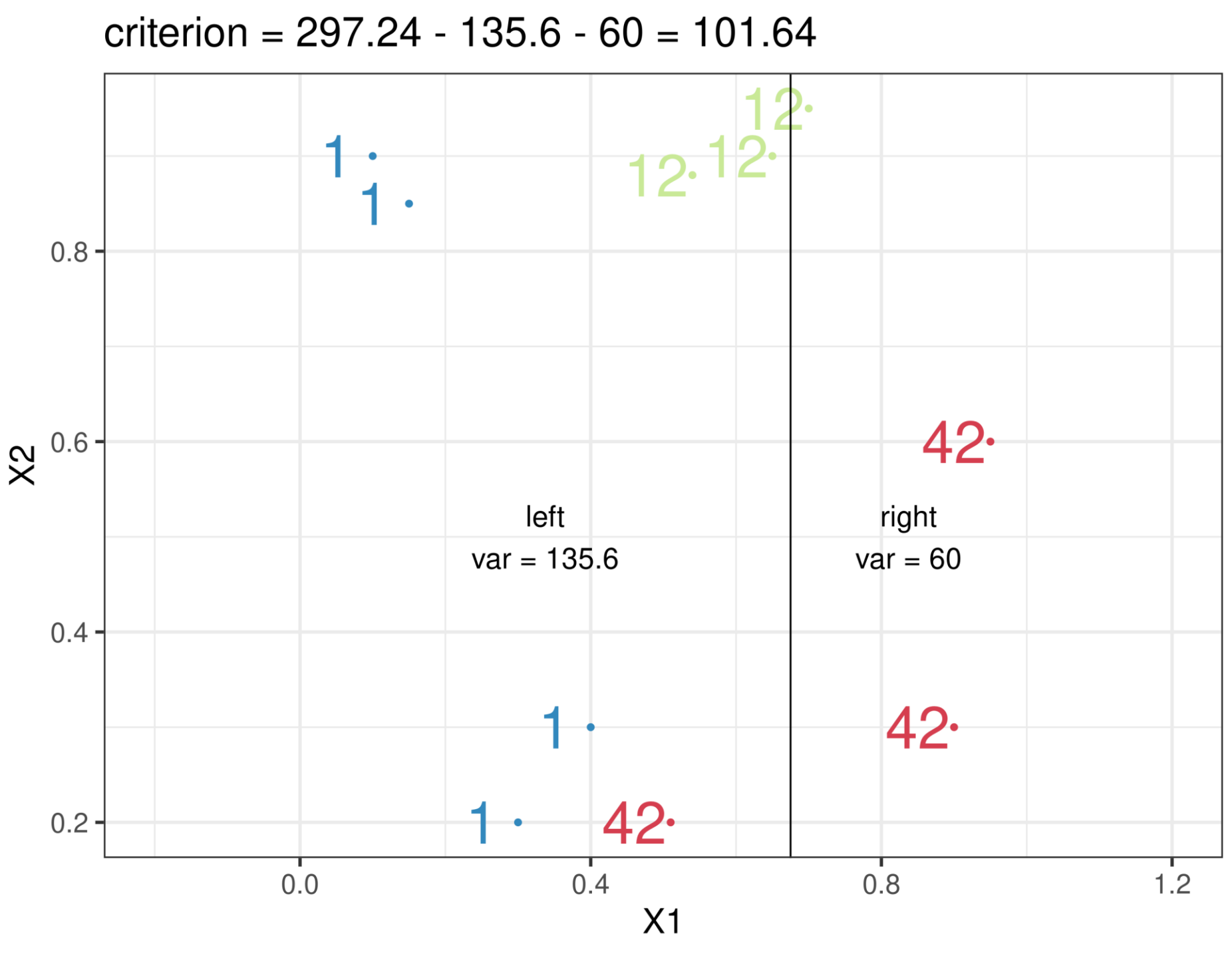

criterion = 297.24 - 0 - 236.4 = 60.84
left
var = 0
right
var = 236.4
X1
X2
0.0
0.4
0.8
1.2
0.2
0.4
0.6
0.8

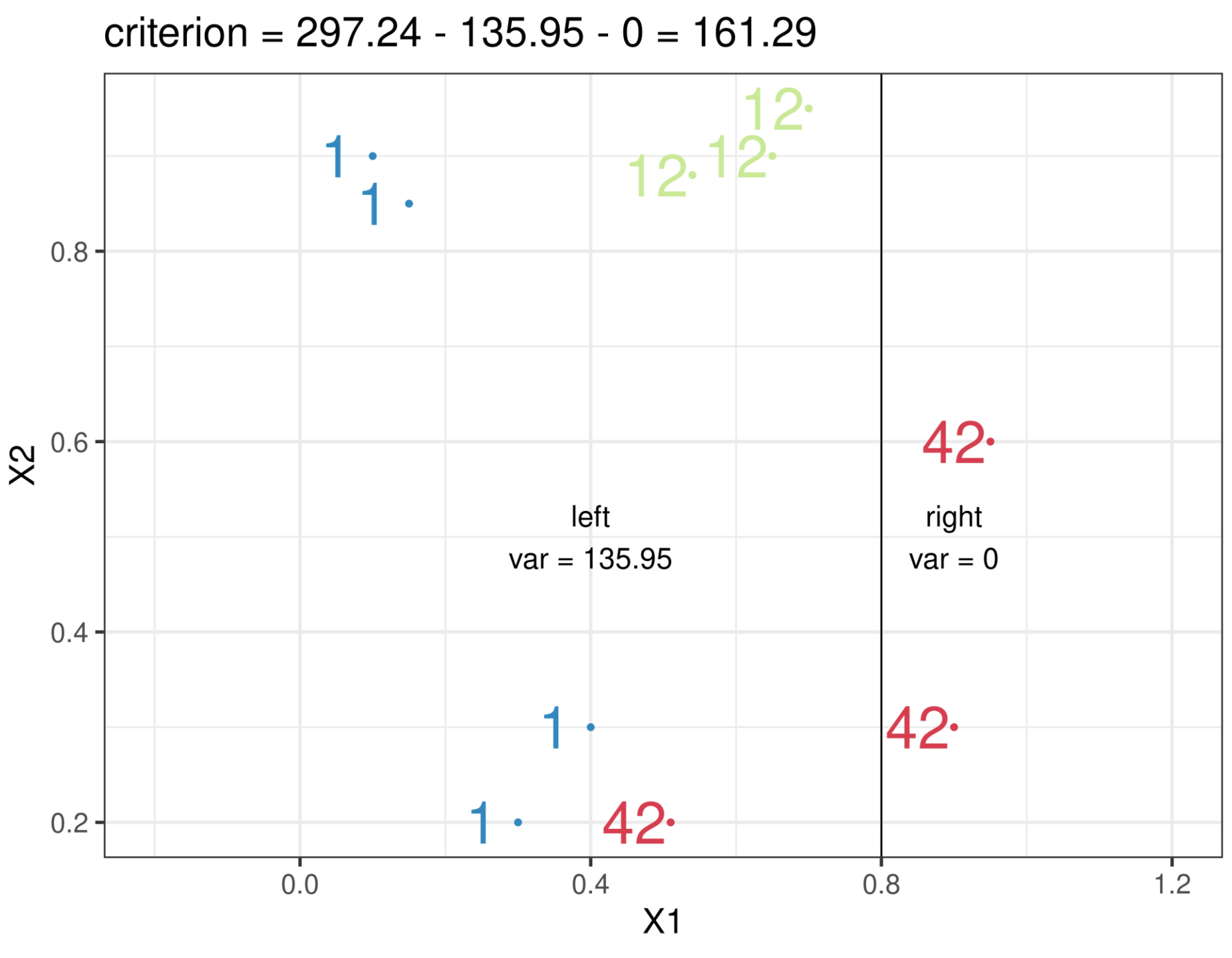

criterion = 297.24 - 0 - 192.94 = 104.3
left
var = 0
right
var = 192.94
X1
X2
0.0
0.4
0.8
1.2
0.2
0.4
0.6
0.8

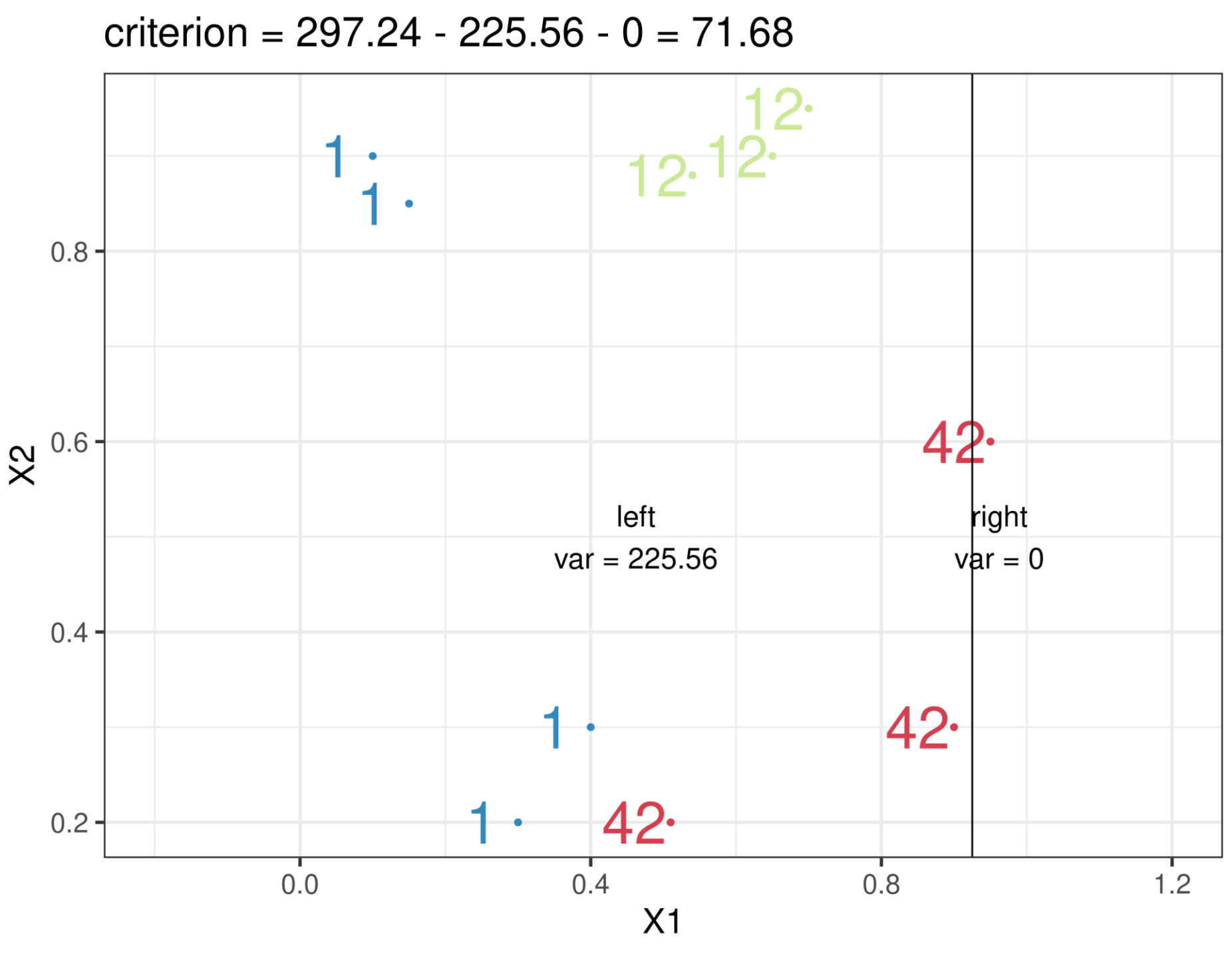

criterion = 297.24 - 0 - 135 = 162.24
left
var = 0
right
var = 135
X1
X2
0.0
0.4
0.8
1.2
0.2
0.4
0.6
0.8

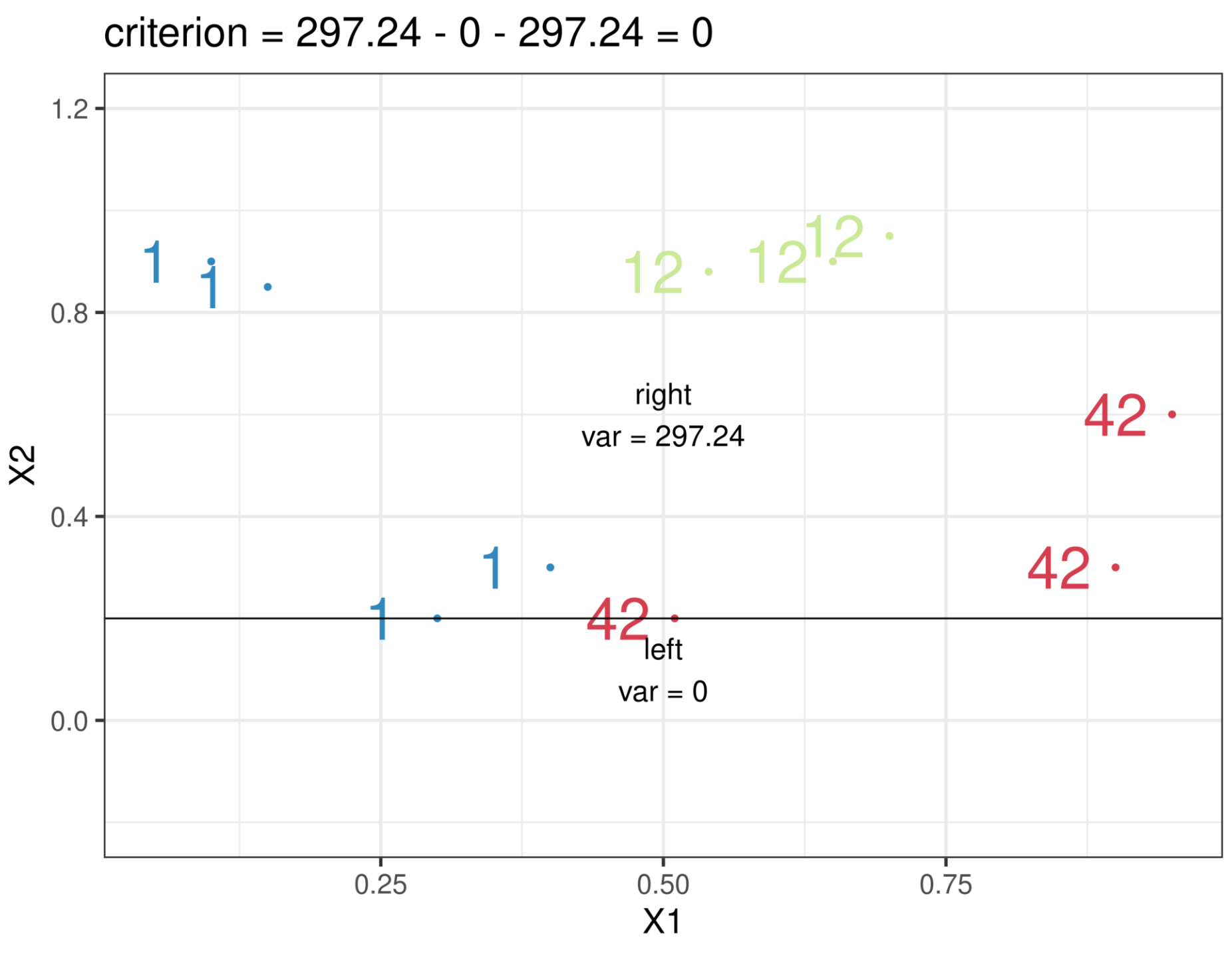

criterion = 297.24 - 134.48 - 108 = 54.76
left
var = 134.48
right
var = 108
X1
X2
0.0
0.4
0.8
1.2
0.2
0.4
0.6
0.8

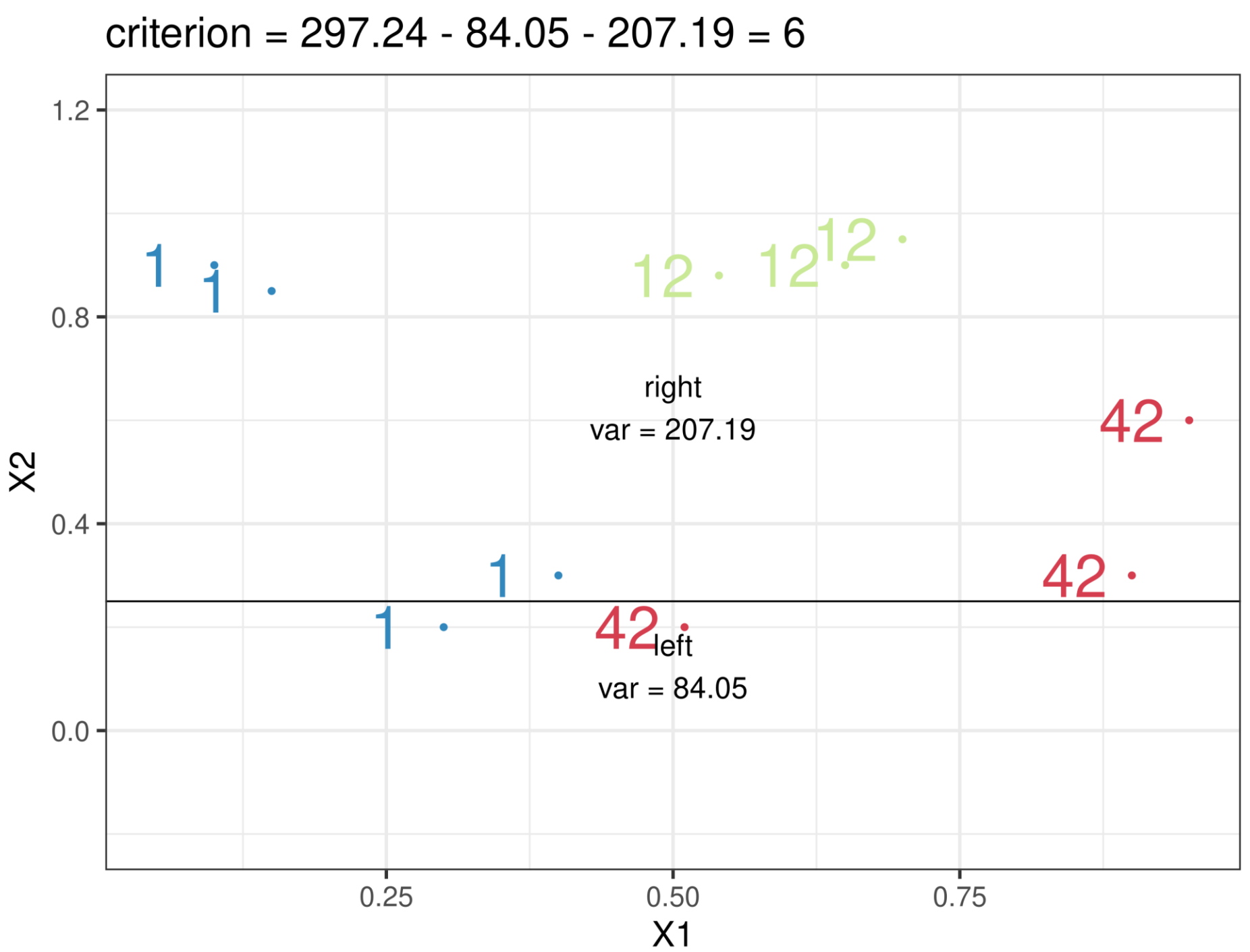

criterion = 297.24 - 135.13 - 90 = 72.11
left
var = 135.13
right
var = 90
X1
X2
0.0
0.4
0.8
1.2
0.2
0.4
0.6
0.8

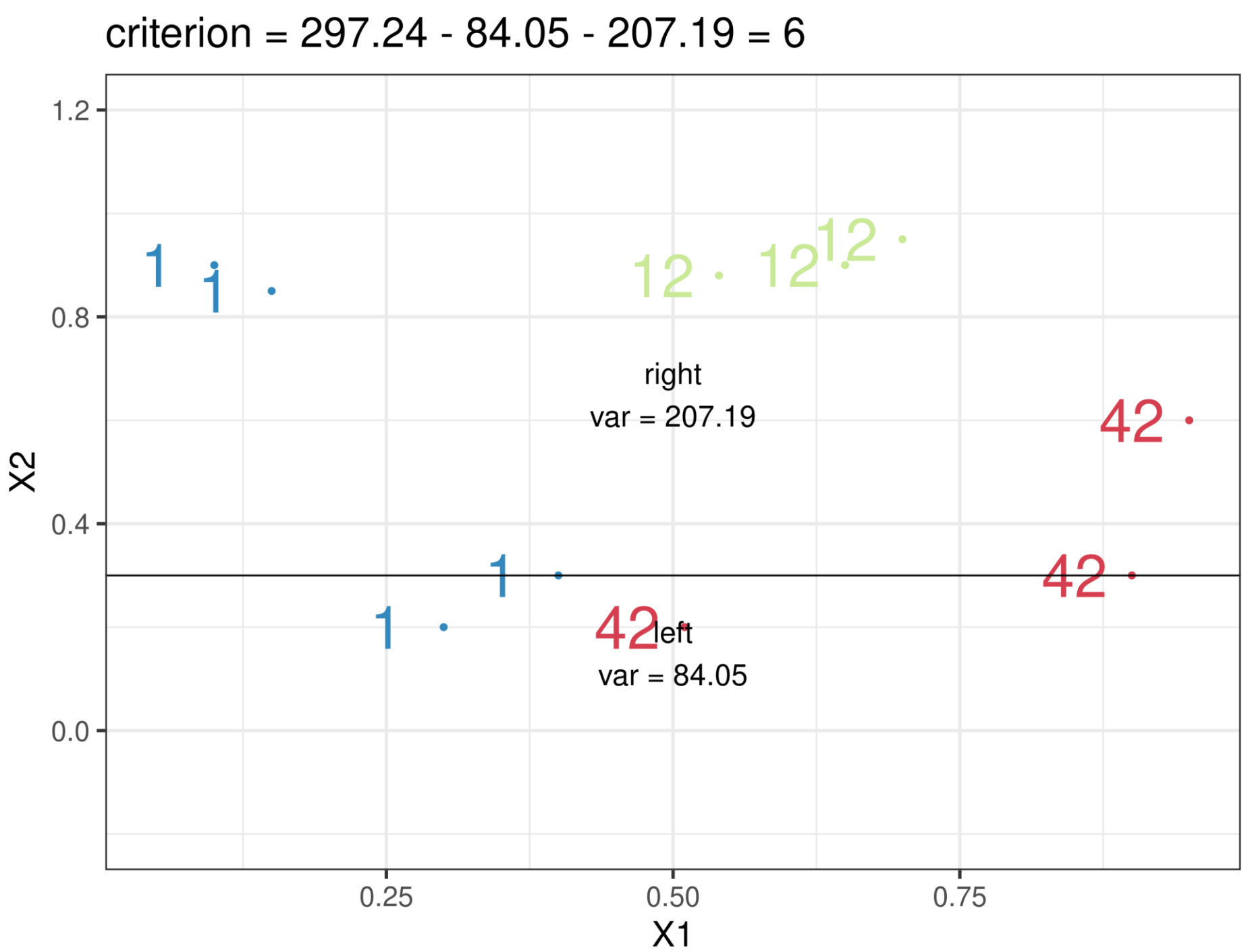

criterion = 297.24 - 135.6 - 60 = 101.64
left
var = 135.6
right
var = 60
X1
X2
0.0
0.4
0.8
1.2
0.2
0.4
0.6
0.8

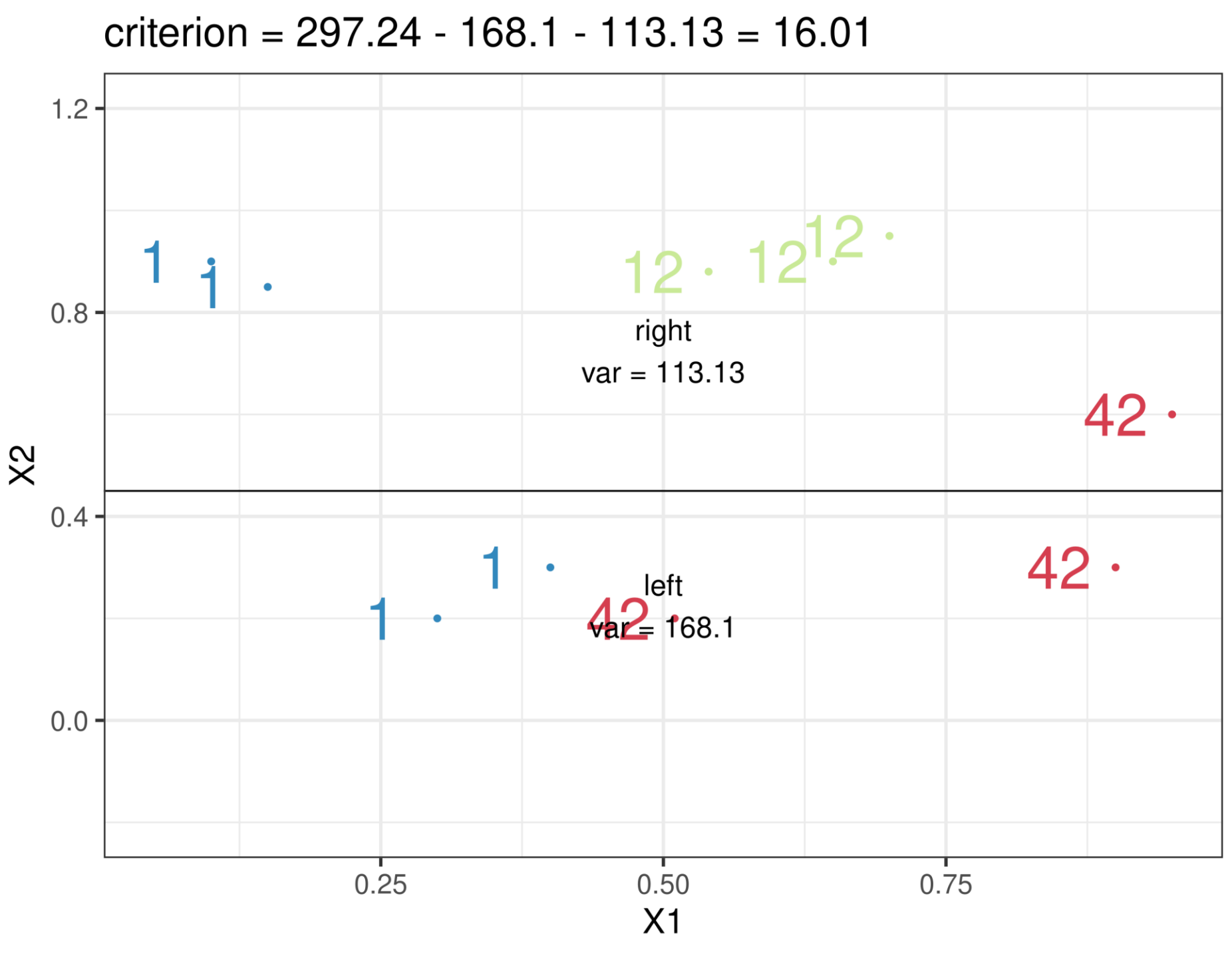

criterion = 297.24 - 135.95 - 0 = 161.29
left
var = 135.95
right
var = 0
X1
X2
0.0
0.4
0.8
1.2
0.2
0.4
0.6
0.8

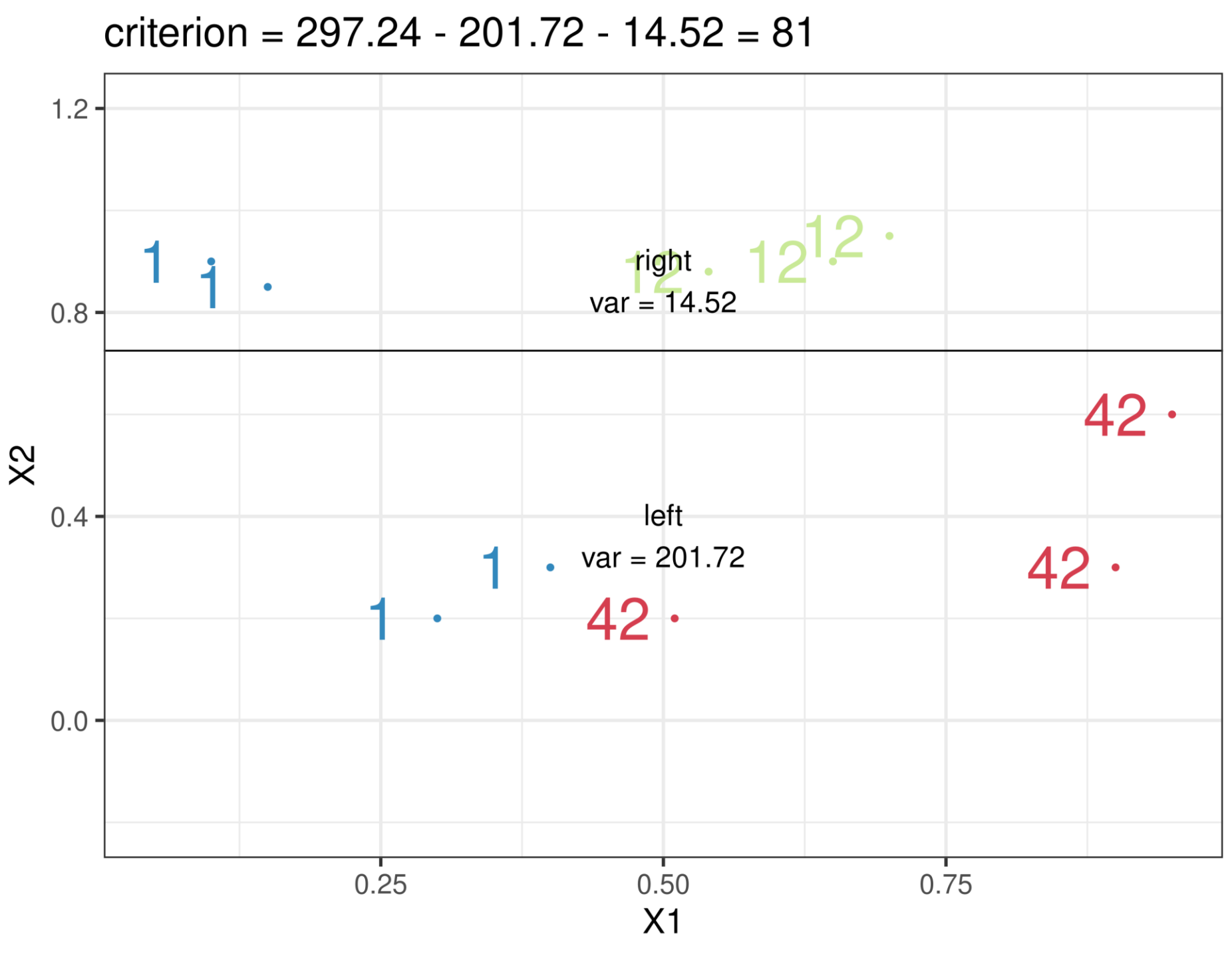

criterion = 297.24 - 225.56 - 0 = 71.68
left
var = 225.56
right
var = 0
X1
X2
0.0
0.4
0.8
1.2
0.2
0.4
0.6
0.8

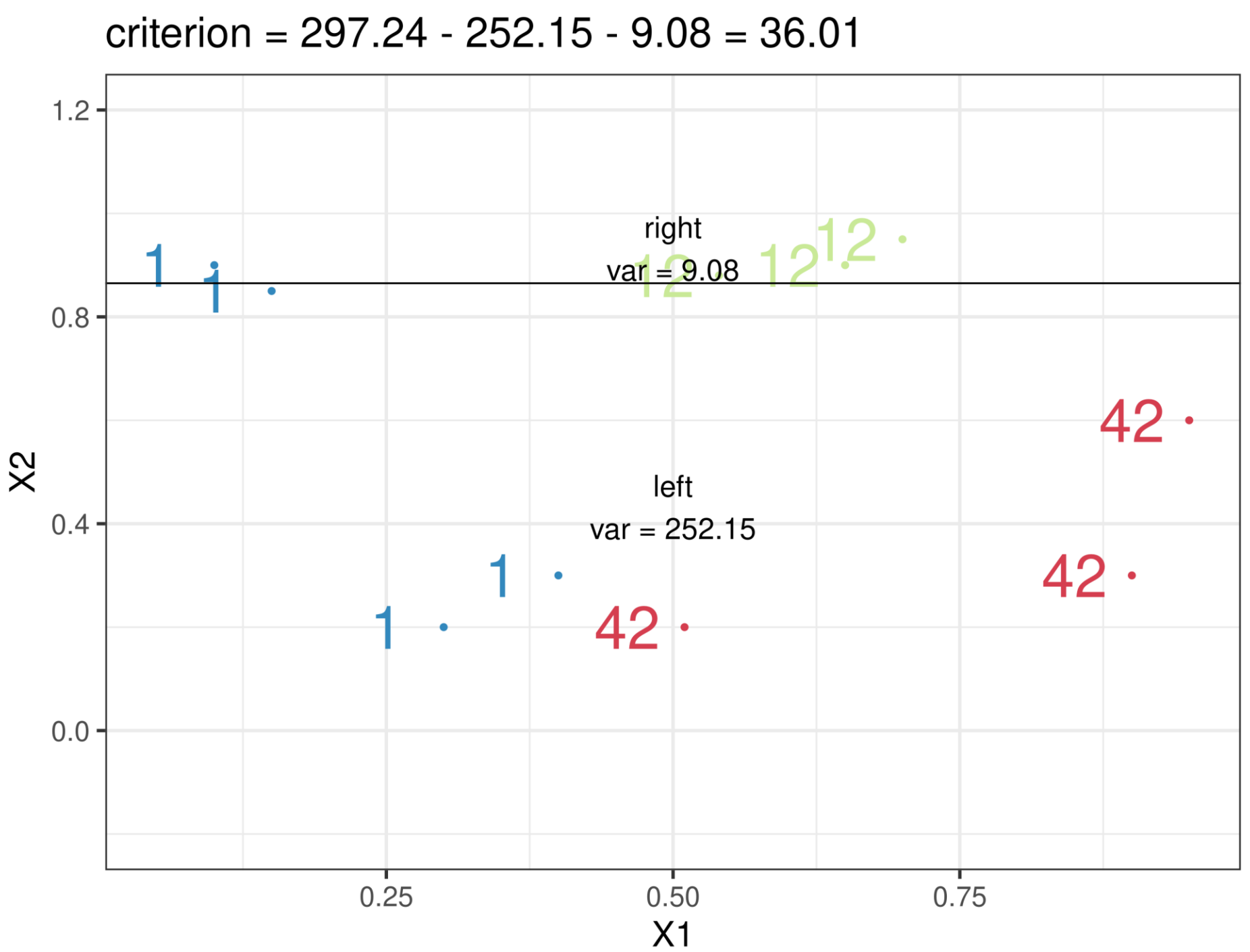

criterion = 297.24 - 0 - 297.24 = 0
1.2
0.8
0.4
0.0
X2
0.25
0.50
0.75
X1
right
var = 297.24
left
var = 0

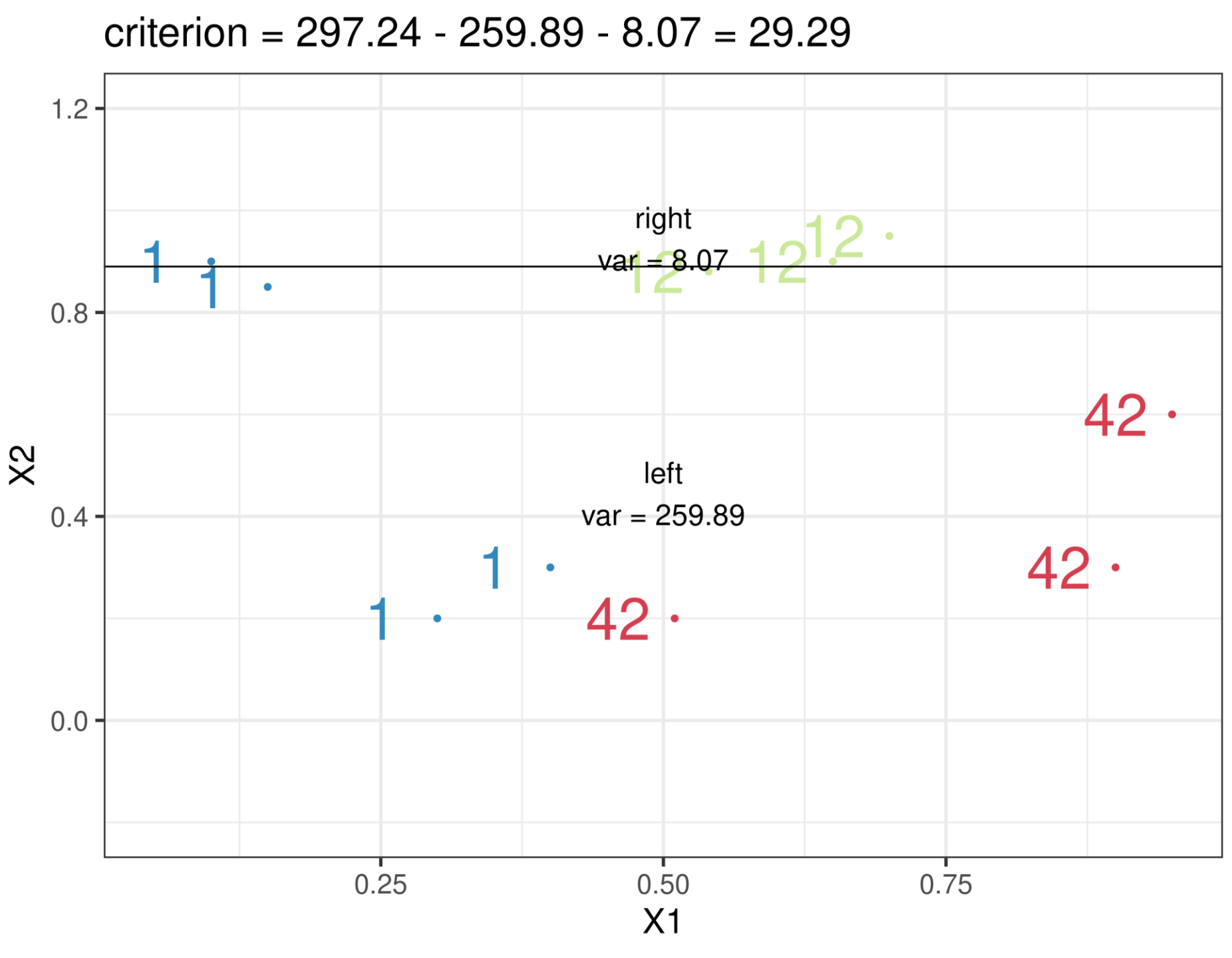

criterion = 297.24 - 84.05 - 207.19 = 6
1.2
0.8
0.4
0.0
X2
0.25
0.50
0.75
X1
right
var = 207.19
left
var = 84.05

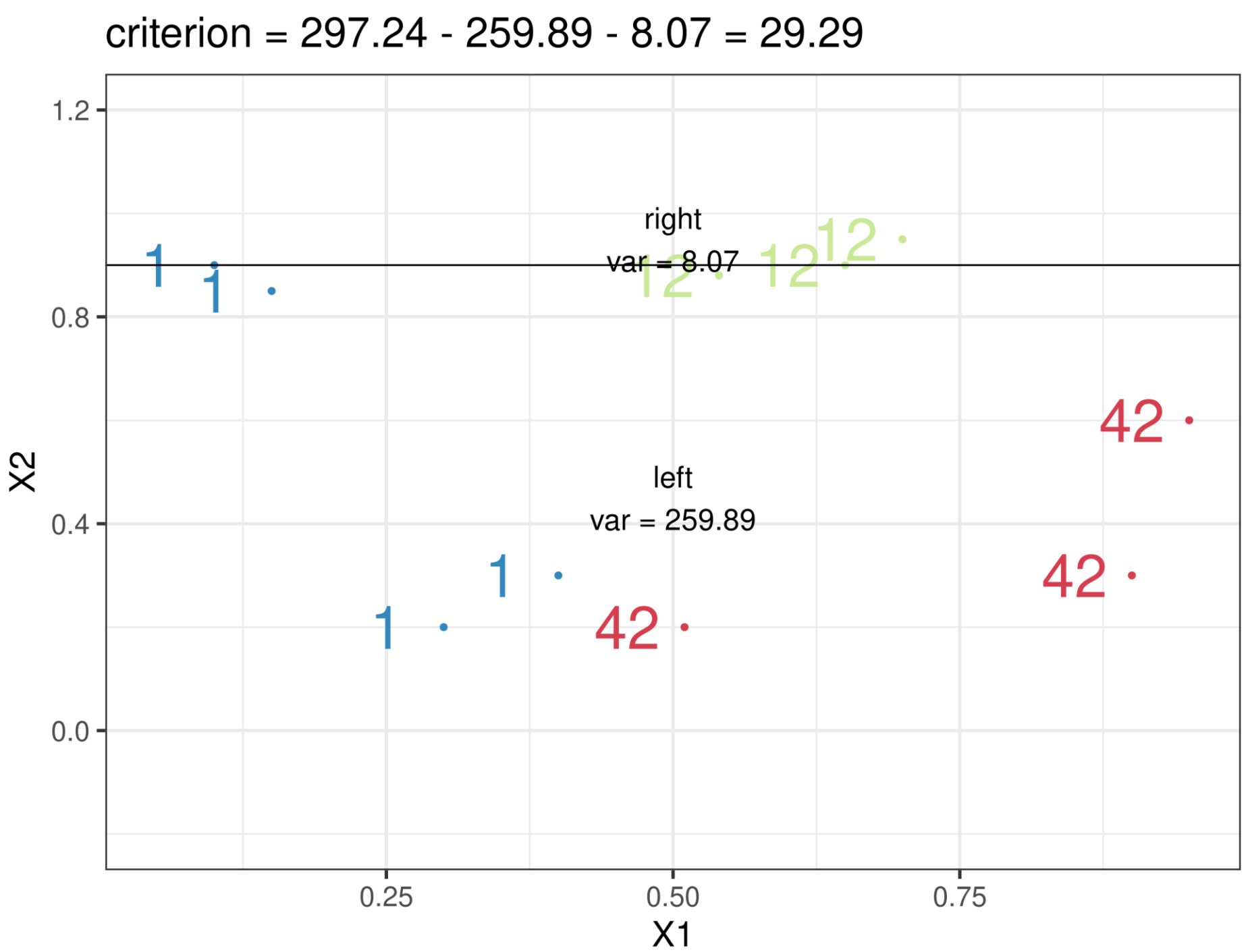

criterion = 297.24 - 84.05 - 207.19 = 6
1.2
0.8
0.4
0.0
X2
0.25
0.50
0.75
X1
right
var = 207.19
left
var = 84.05

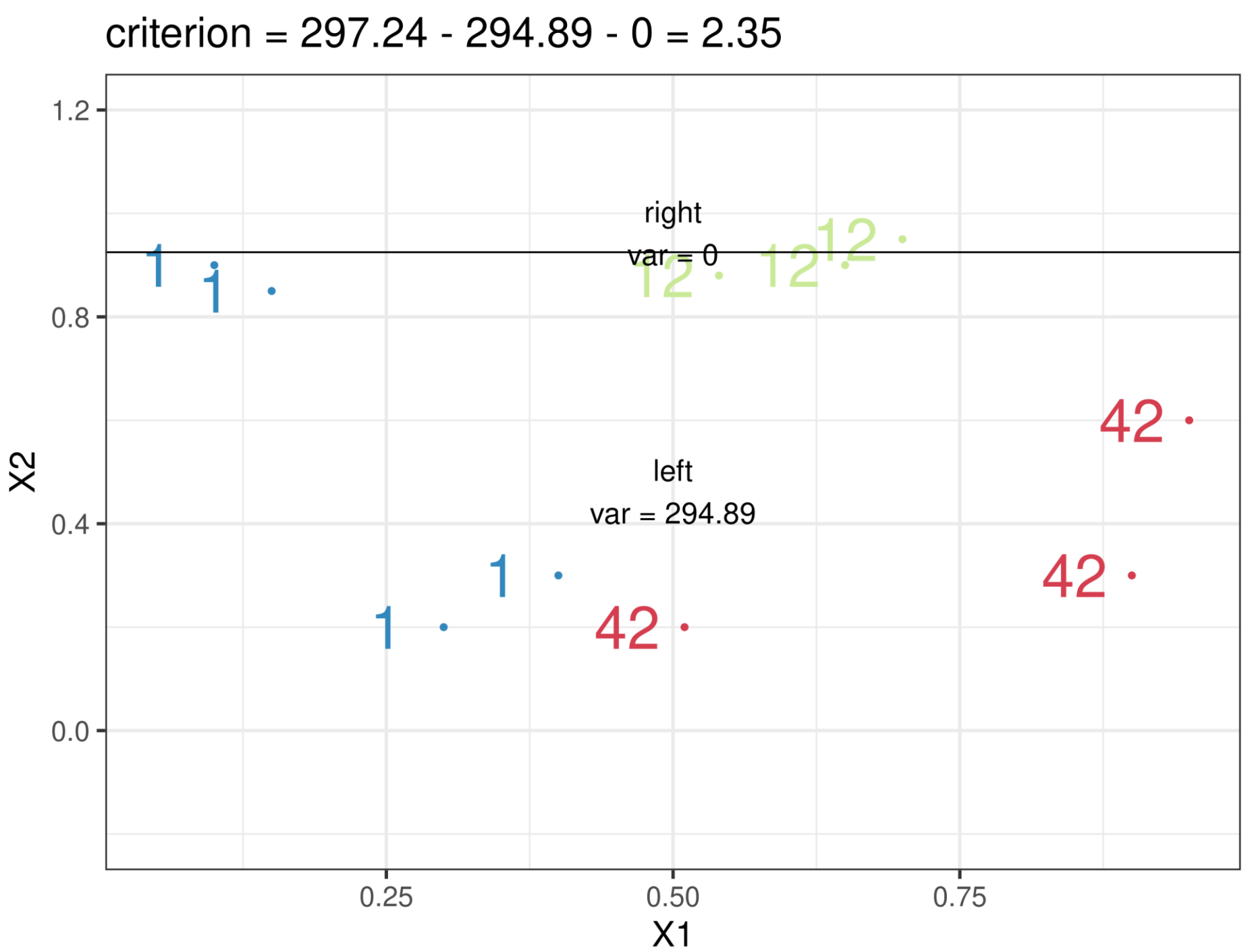

criterion = 297.24 - 168.1 - 113.13 = 16.01
1.2
0.8
0.4
0.0
X2
right
var = 113.13
left
var = 168.1
0.25
0.50
0.75
X1

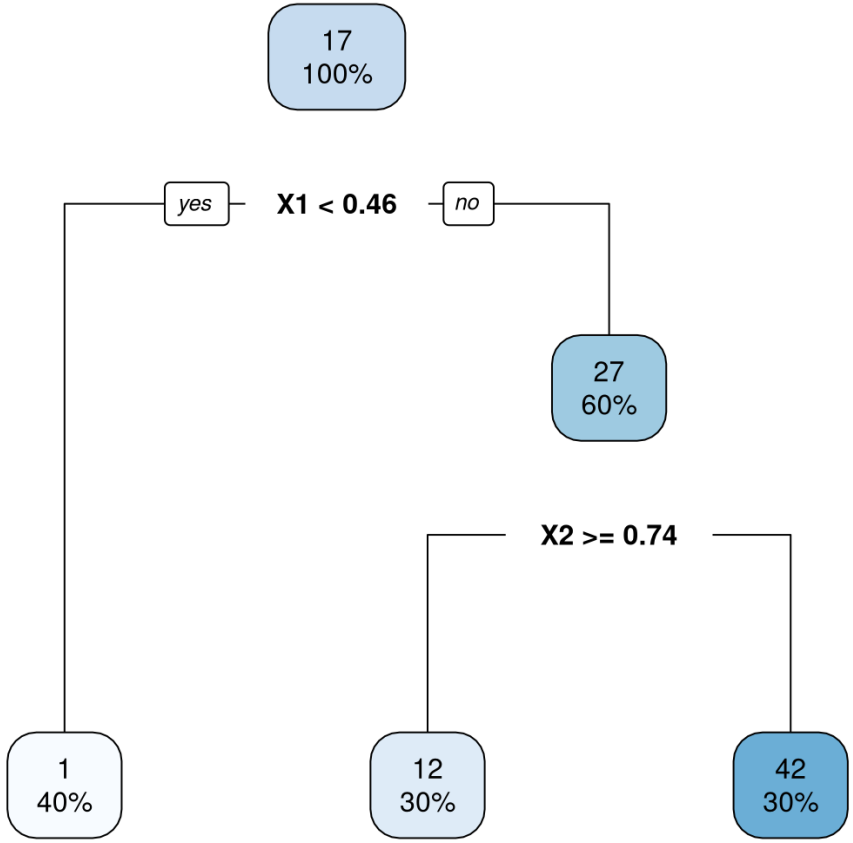

criterion = 297.24 - 201.72 - 14.52 = 81
1.2
0.8
0.4
0.0
X2
right
var = 14.52
left
var = 201.72
0.25
0.50
0.75
X1

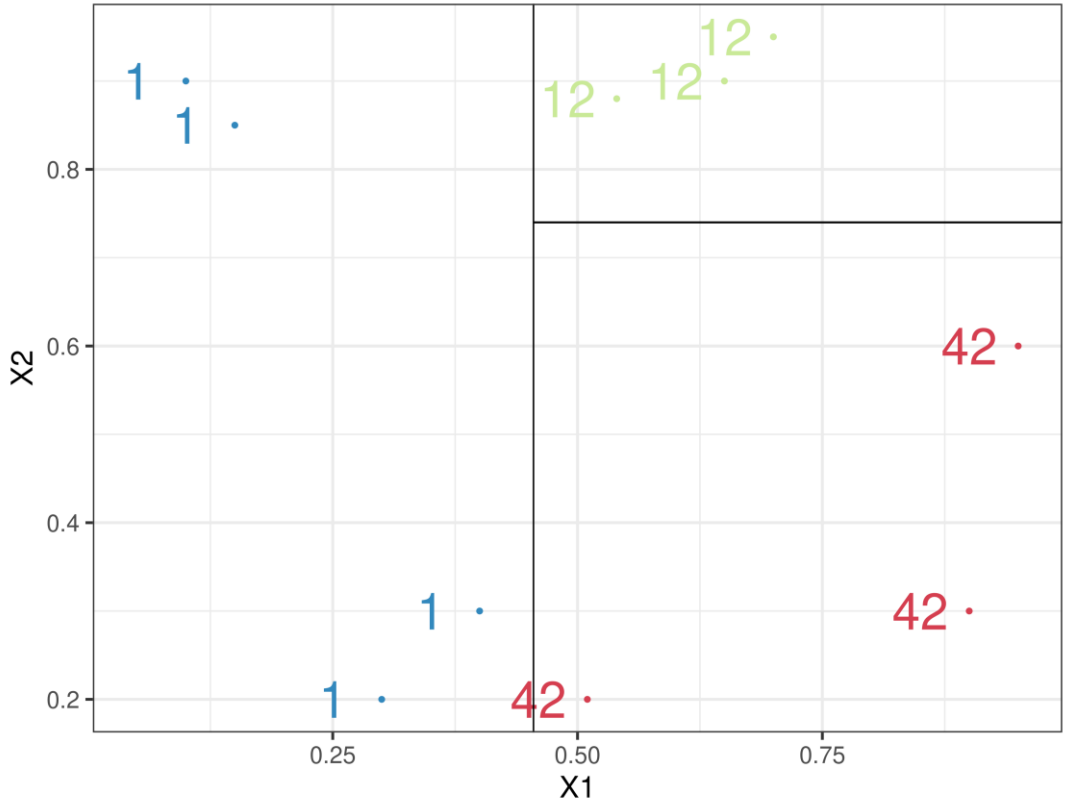

criterion = 297.24 - 252.15 - 9.08 = 36.01
right
var = 9.08
left
var = 252.15
X1
X2
0.25
0.50
0.75
0.0
0.4
0.8
1.2


criterion = 297.24 - 259.89 - 8.07 = 29.29
right
var = 8.07
left
var = 259.89
X1
X2
0.25
0.50
0.75
0.0
0.4
0.8
1.2


criterion = 297.24 - 259.89 - 8.07 = 29.29
1.2
0.8
0.4
0.0
X2
0.25
0.50
0.75
X1
right
var = 8.07
left
var = 259.89


criterion = 297.24 - 294.89 - 0 = 2.35
1.2
0.8
0.4
0.0
X2
0.25
0.50
0.75
X1
right
var = 0
left
var = 294.89


17
100%
yes
X1 < 0.46
no
27
60%
X2 >= 0.74
1
40%
12
30%
42
30%


X2
0.8
0.6
0.4
0.2
0.25
0.50
0.75
X1
1
1
1
1
12
12
12
42
42
42

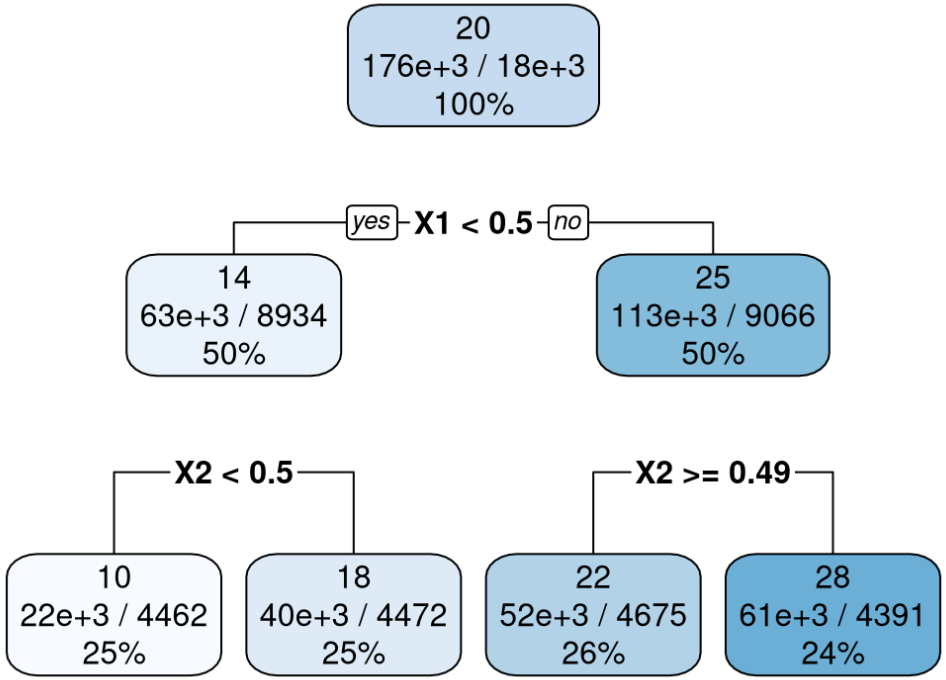
Pour chaque *split* potentiel, la somme des déviances des deux noeuds/régions filles :

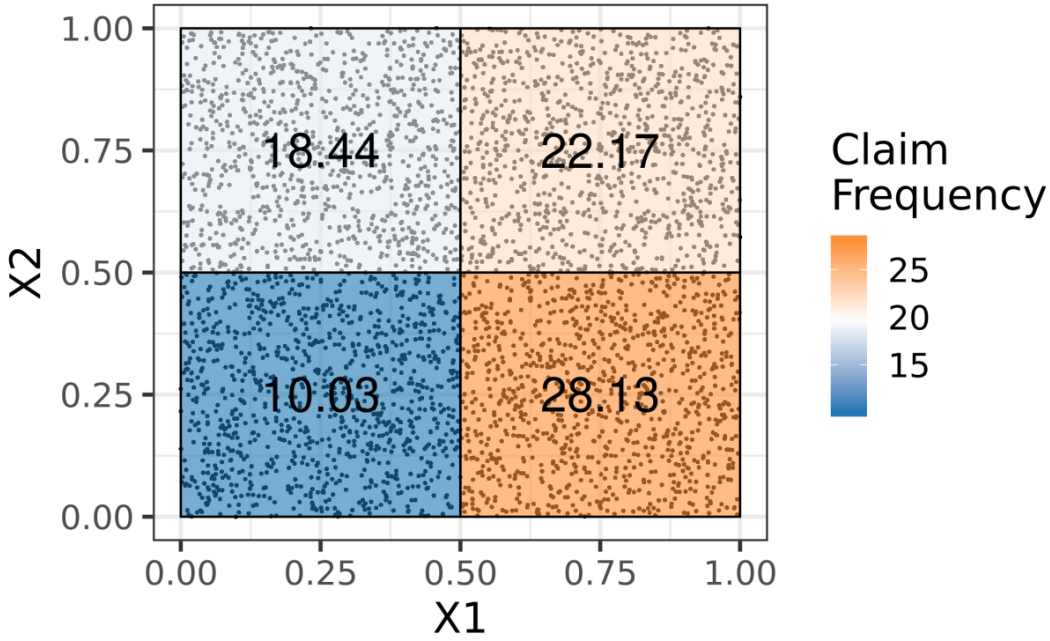
$$D^*_{\tilde{R}_{l,0}}(N, \tilde{\lambda}_{l,0}) + D^*_{\tilde{R}_{l,1}}(N, \tilde{\lambda}_{l,1})$$

Où la déviance s'exprime pour chaque région fille $\tilde{R}_{l,\tau},\ \tau = \{0, 1\}$

$$D^*_{\tilde{R}_{l,\tau}}(N, \tilde{\lambda}_{l,\tau}) = \sum_{k=1}^{n} 2N_k \left[\frac{\tilde{\lambda}_{l,\tau}\omega_k}{N_k} - 1 - \log\left(\frac{\tilde{\lambda}_{l,\tau}\omega_k}{N_k} \right) \right] \mathbb{1}_{x_k \in \tilde{R}_{l,\tau}}$$

avec N le nombre total de sinistres observés, N_k le nombre de sinistres pour l'assuré k, $\tilde{\lambda}_{l,\tau}$ la fréquence sinistre prédite au sein d'une région fille $\tilde{R}_{l,\tau},\ \tau = \{0, 1\}$ et ω_k l'exposition pour l'assuré k.





Comme on peut s'y attendre, l'arbre obtenu est cohérent avec les données initiales.

Un Additive Tree Model (ATM) [Cui et al., 2015] est un ensemble d'arbres de régression ou de classification qui s'écrit sous la forme :

$$F(x_i) = \sum_{t=1}^{T} w_t f_t(x_i)$$

où $f_t(x_i)$ est l'output de l'arbre t pour l'observation x_i et $w_t \in \mathcal{W}$ est le poids de l'arbre t parmi l'ensemble des T arbres.

Cui, Z., Chen, W., He, Y., and Chen, Y. (2015).
Optimal action extraction for random forests and boosted trees.
In Proceedings of the 21th ACM SIGKDD international conference on knowledge discovery and data mining, pages 179–188. ACM

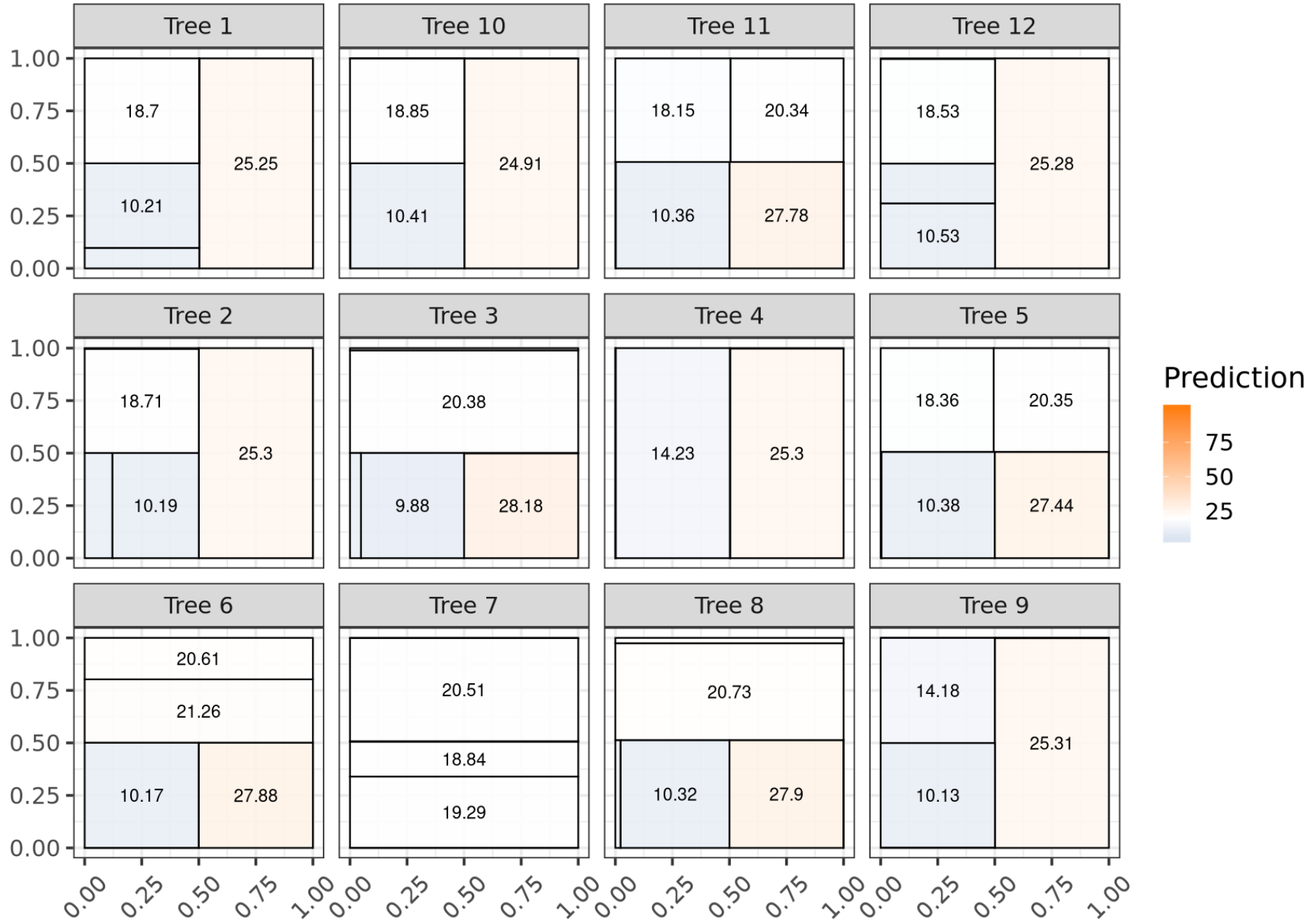
Le Poisson Random Forest est une combinaison de T arbres de régression de Poisson. En conséquence, il est possible d'écrire ses prédictions :

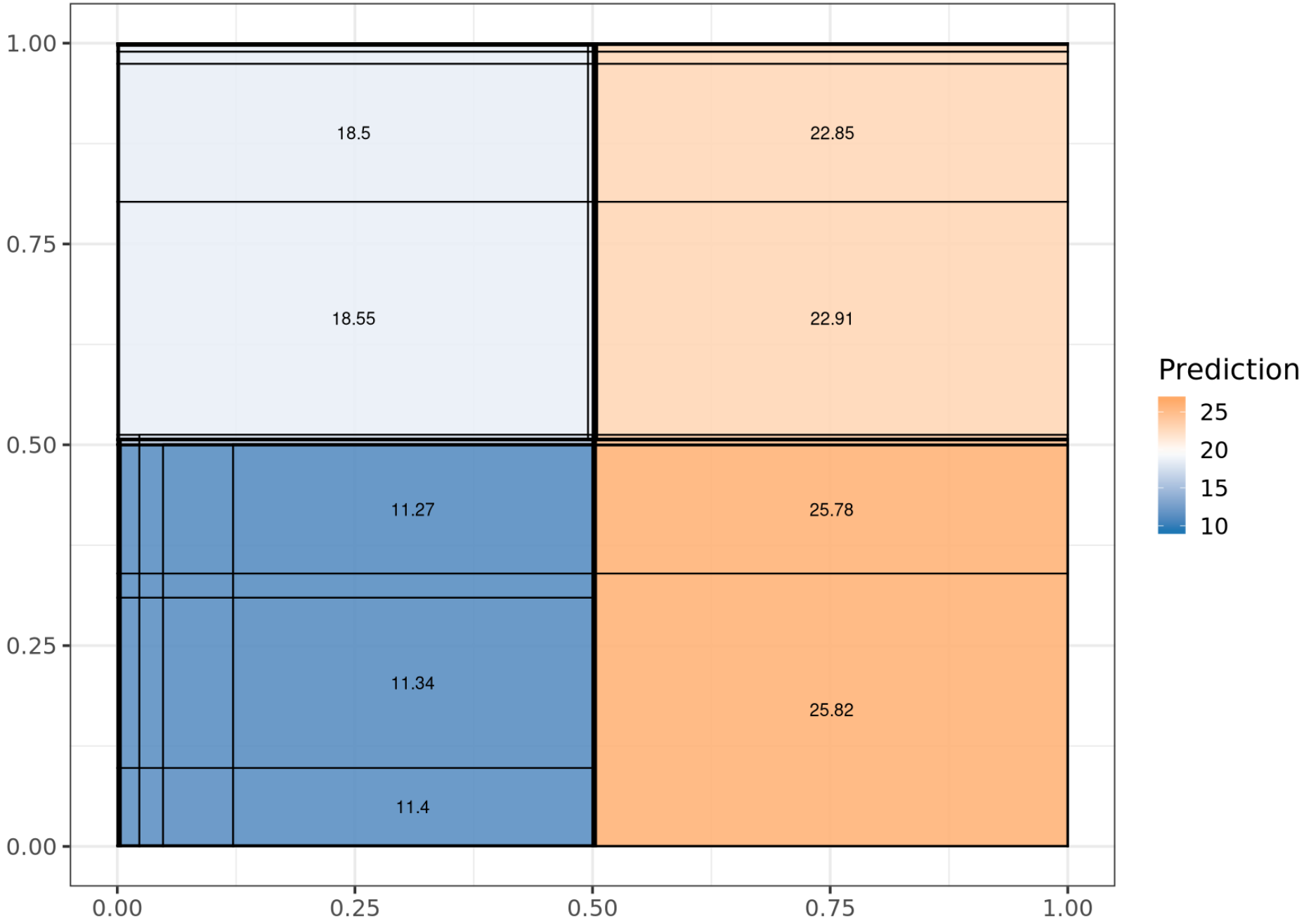
$$\forall i \in [\![1, n]\!],\ F(x_i) = \sum_{t=1}^{T} \frac{1}{T} f_t(x_i; \tilde{\mathcal{R}}_t, \tilde{\Lambda}_t, I_t)$$

où $\tilde{\mathcal{R}}_t$ est la partition formée par l'arbre t, $\tilde{\Lambda}_t$ est l'ensemble des fréquences sinistres prédites par l'arbre et I_t le nombre de noeuds.

Dès lors, en posant $w_t = 1/T$ il devient aisé de voir que le Poisson Random Forest est un ATM.

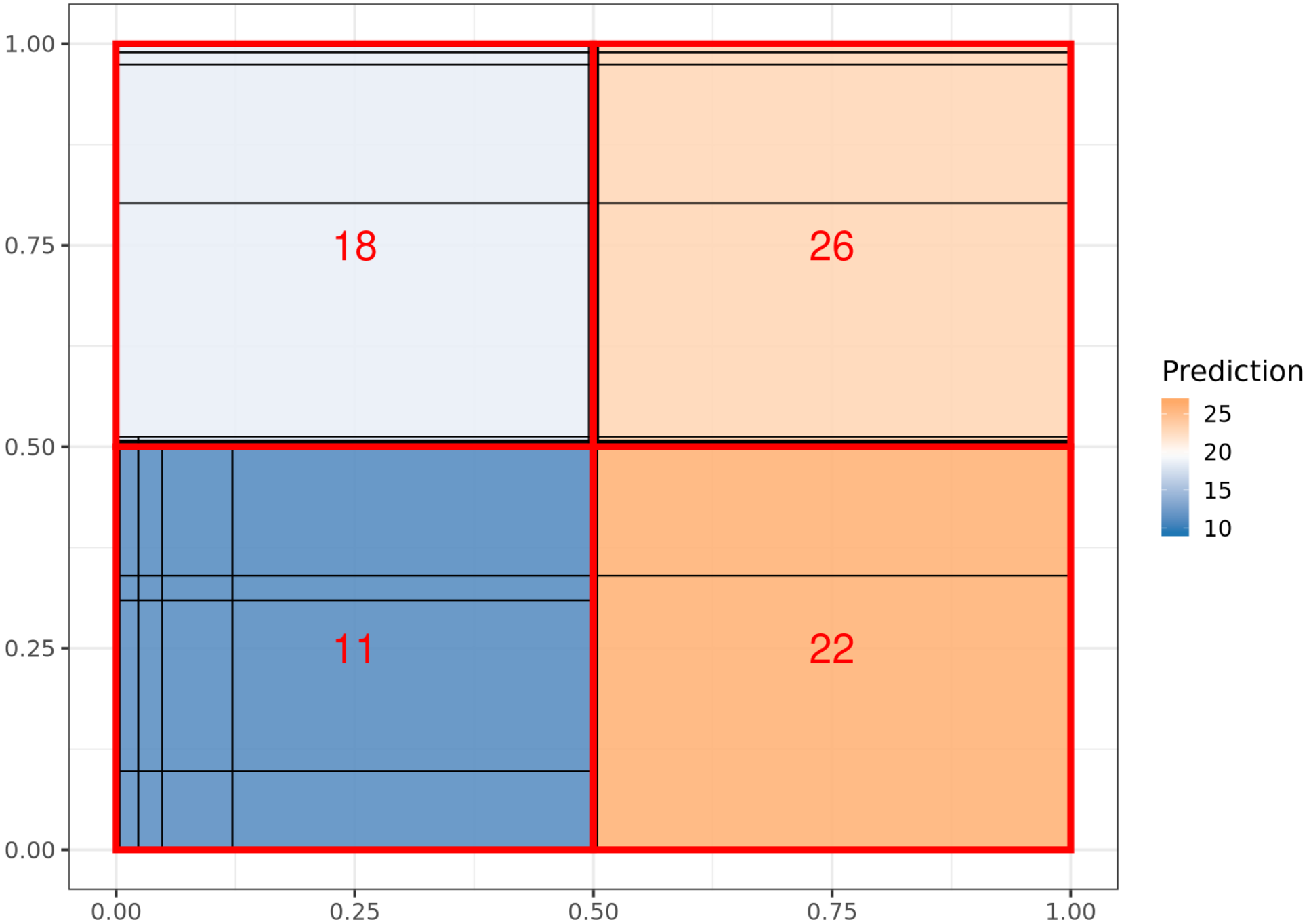
Par définition, un modèle Gradient Boosting appartient à la classe des ATM.

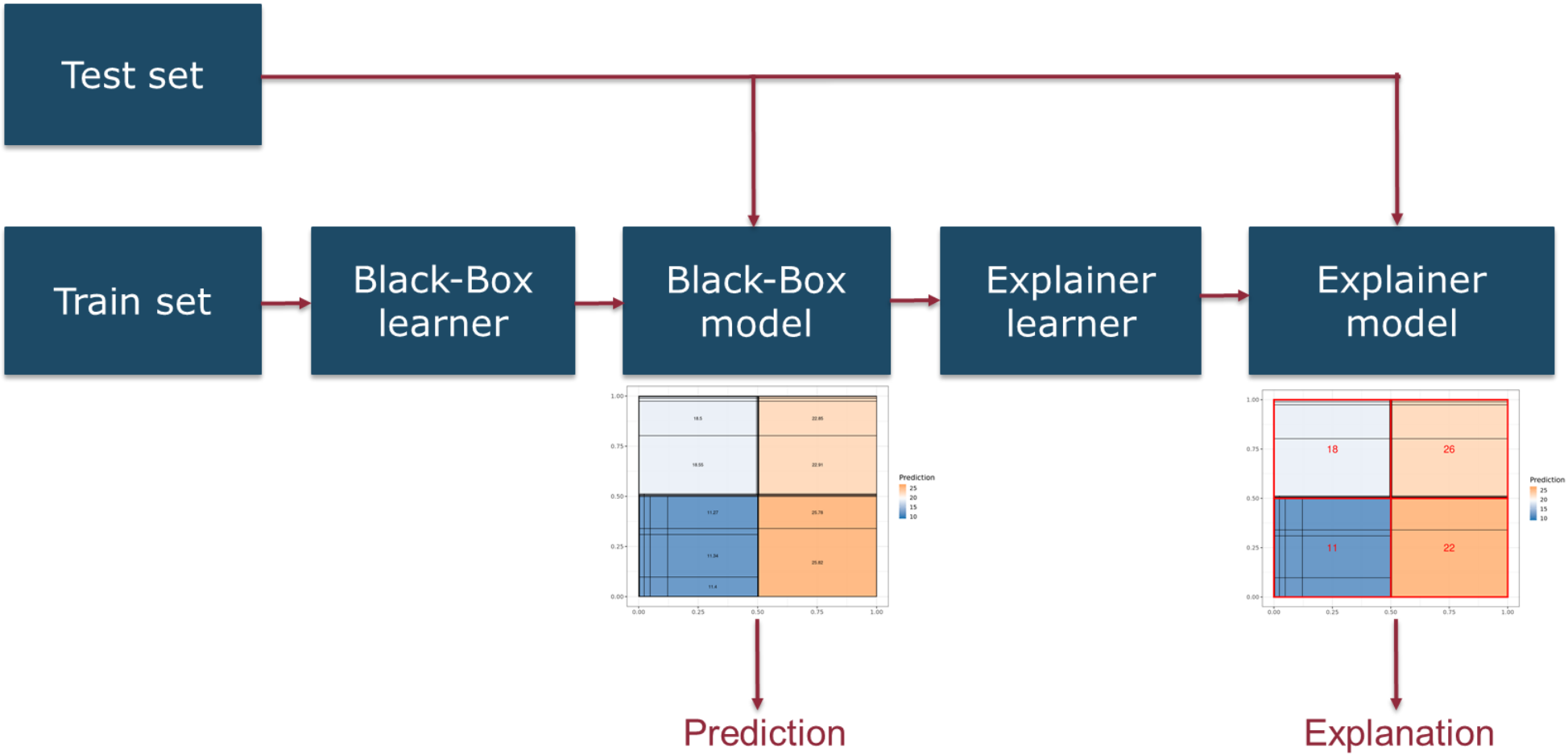

Tree 1
18.7
25.25
10.21
Tree 10
18.85
24.91
10.41
Tree 11
18.15
20.34
10.36
27.78
Tree 12
18.53
25.28
10.53
Tree 2
18.71
25.3
10.19
Tree 3
20.38
9.88
28.18
Tree 4
14.23
25.3
Tree 5
18.36
20.35
10.38
27.44
Tree 6
20.61
21.26
10.17
27.88
Tree 7
20.51
18.84
19.29
Tree 8
20.73
10.32
27.9
Tree 9
14.18
25.31
10.13
0.00
0.25
0.50
0.75
1.00
Prediction
75
50
25


1.00
0.75
0.50
0.25
0.00
0.00
0.25
0.50
0.75
1.00
18.5
22.85
18.55
22.91
11.27
25.78
11.34
25.82
11.4
Prediction
25
20
15
10

L'exemple précédent montre que sur un ATM bien connu (Random Forest), le problème d'intelligibilité peut être formulé comme une problème de parcimonie.

En effet, la prédiction du Random Forest a été rendue plus fine que celle de l'arbre grâce aux multiples régions qui ont été créées. Cependant, cette fragmentation rend le modèle peu parcimonieux et donc très compliqué à comprendre.


1.00
0.75
0.50
0.25
0.00
0.00
0.25
0.50
0.75
1.00
18
26
11
22
Prediction
25
20
15
10


Test set
Train set
Black-Box learner
Black-Box model
Explainer learner
Explainer model
Prediction
Explanation
Prediction
25
20
15
10
18
26
11
22

Idée générale : aggréger des fragments d'espace pour simplifier la partition de la Black-Box et en extraire un modèle surrogate.

Inputs : Les observations du dataset, la prédiction du modèle Black-Box (Random Forest, GBM, XGBoost) ainsi que leurs splits.

Output : Un arbre de décision à K feuilles.

L'ATM génère $G \in \mathbb{N}$ régions notées $\mathcal{R} = \{\mathcal{R}_g\}_{g=1}^{G}$ dont les valeurs prédites sont $\Lambda = \{\lambda_g\}_{g=1}^{G}$. Nous voulons trouver $K \in \mathbb{N}$ régions avec $K << G$, $\Lambda^{'} = \{\lambda_k^{'}\}_{k=1}^{K}$ et $\mathcal{R}^{'} = \{\mathcal{R}_k^{'}\}_{k=1}^{K}$ telles que

$$R^2 = 1 - \frac{\sum_{i=1}^{n} \left(f_{bb}(x_i, \omega_i; \Lambda, \mathcal{R}, G) - f_{surrogate}(x_i, \omega_i; \Lambda^{'}, \mathcal{R}^{'}, K)\right)^2}{\sum_{i=1}^{n} \left(f_{bb}(x_i, \omega_i; \Lambda, \mathcal{R}, G) - \bar{y}_{bb}\right)^2} > C$$

où $\bar{y}_{bb}$ est la moyenne des prédictions de l'ATM Black-Box et C est une constante définie par l'utilisateur.

Comme le proposent [Hara and Hayashi, 2016], nous voulons transformer le problème en un problème d'optimisation de vraisemblance. Pour cela, il nous faut une expression generative probabiliste de l'ATM $p(y, \boldsymbol{x}; \Pi)$. Le problème de maximisation devient,

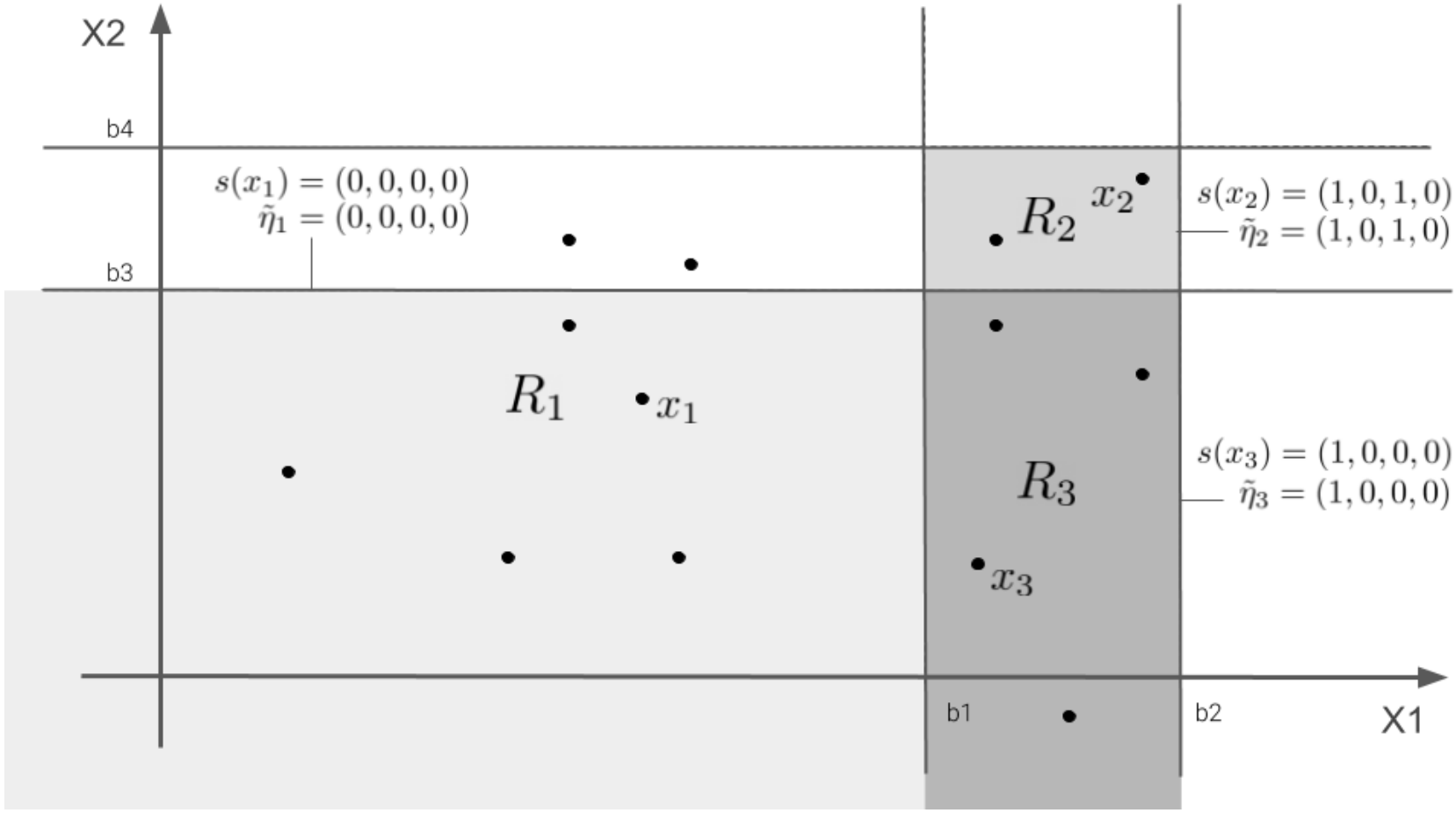
$$\max_{\Pi} p(y, \boldsymbol{x}; \Pi)$$

où Π est l'ensemble des paramètres de l'ATM, $\boldsymbol{x}$ les observations et y les prédictions du modèle Black-Box.

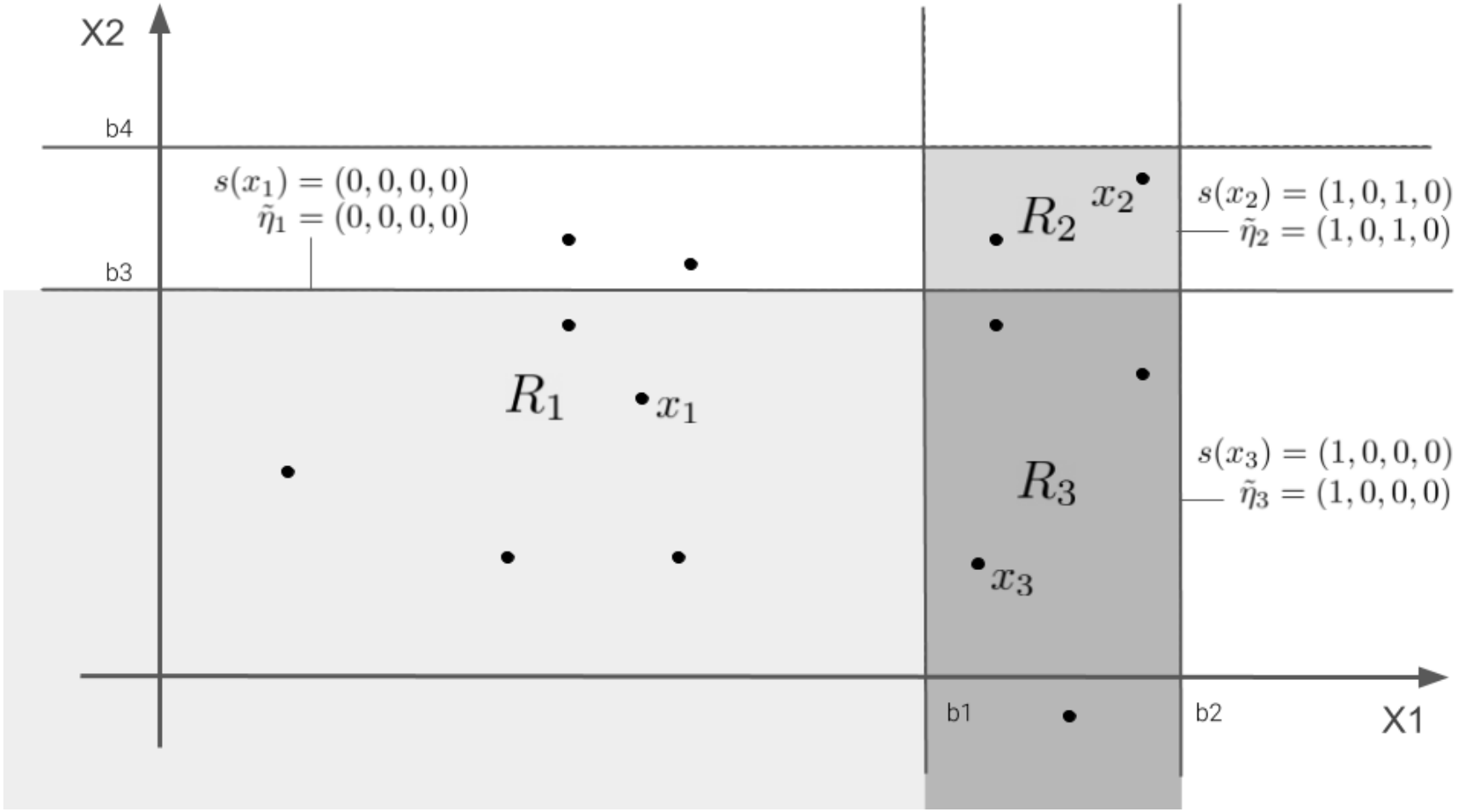
Malheureusement, il n'y a pas de formule fermée pour obtenir les paramètres. Néanmoins, nous pouvons utiliser l'algorithme Expectation Maximisation qui est courante pour des problèmes complexes d'optimisation de log-vraisemblance. Un cas d'école est le mélange gaussien.

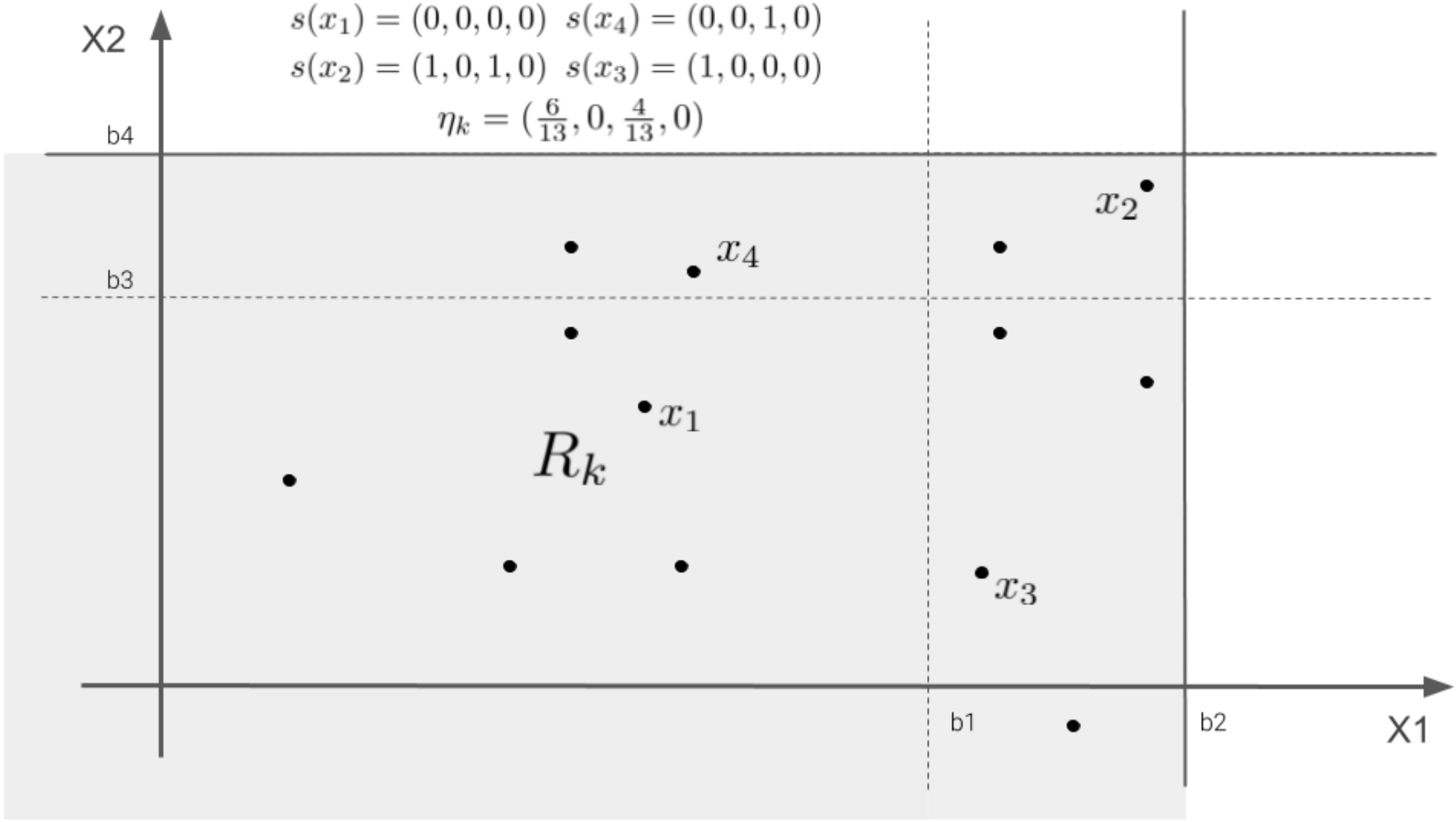
Hara, S. and Hayashi, K. (2016).
Making Tree Ensembles Interpretable: A Bayesian Model Selection Approach.
arXiv:1606.09066 [stat]

La forêt généré au total L splits. Nous pouvons exprimer chaque région R_g de la partition comme un vecteur binaire $\tilde{\boldsymbol{\eta}}_{\boldsymbol{g}} \in \{0,1\}^L$ dont les composantes $\tilde{\eta}_{g_l}$ valent 1 si la région satisfait $\forall \boldsymbol{x} \in R_g,\ x_{.,d_l} > b_l$, où d_l est l'index de la variable splittée au noeud l.



Chaque $\boldsymbol{x}$ appartient à une unique région R_g de la partition formée par l'output de l'ATM. Ce qui signifie que la représentation binaire de x notée $\boldsymbol{s(x)}$ est égale à la représentation binaire $\boldsymbol{\tilde{\eta}_g}$ de la région R_g si et seulement si $\boldsymbol{x} \in R_g$.




$s(x_1) = (0, 0, 0, 0)$ $s(x_4) = (0, 0, 1, 0)$
$s(x_2) = (1, 0, 1, 0)$ $s(x_3) = (1, 0, 0, 0)$
$\eta_k = (\frac{6}{13}, 0, \frac{4}{13}, 0)$
X2
b4
b3
b1
b2
X1
x_1
x_2
x_3
x_4
R_k

Considérons maintenant une potentielle région R_k créée à partir de la concaténation des régions R_1, R_2 et R_3. Désormais, certains $\boldsymbol{x} \in R_k$ vérifient $x_{.,d_2} > b_2$ alors que d'autres non. Mais tous les $\boldsymbol{x} \in R_k$ vérifient $x_{.,d_1} > b_1$ et $x_{.,d_3} \leq b_3$ par exemple. Il est maintenant possible d'exprimer la probabilité d'observer une représentation binaire pour une région k par :

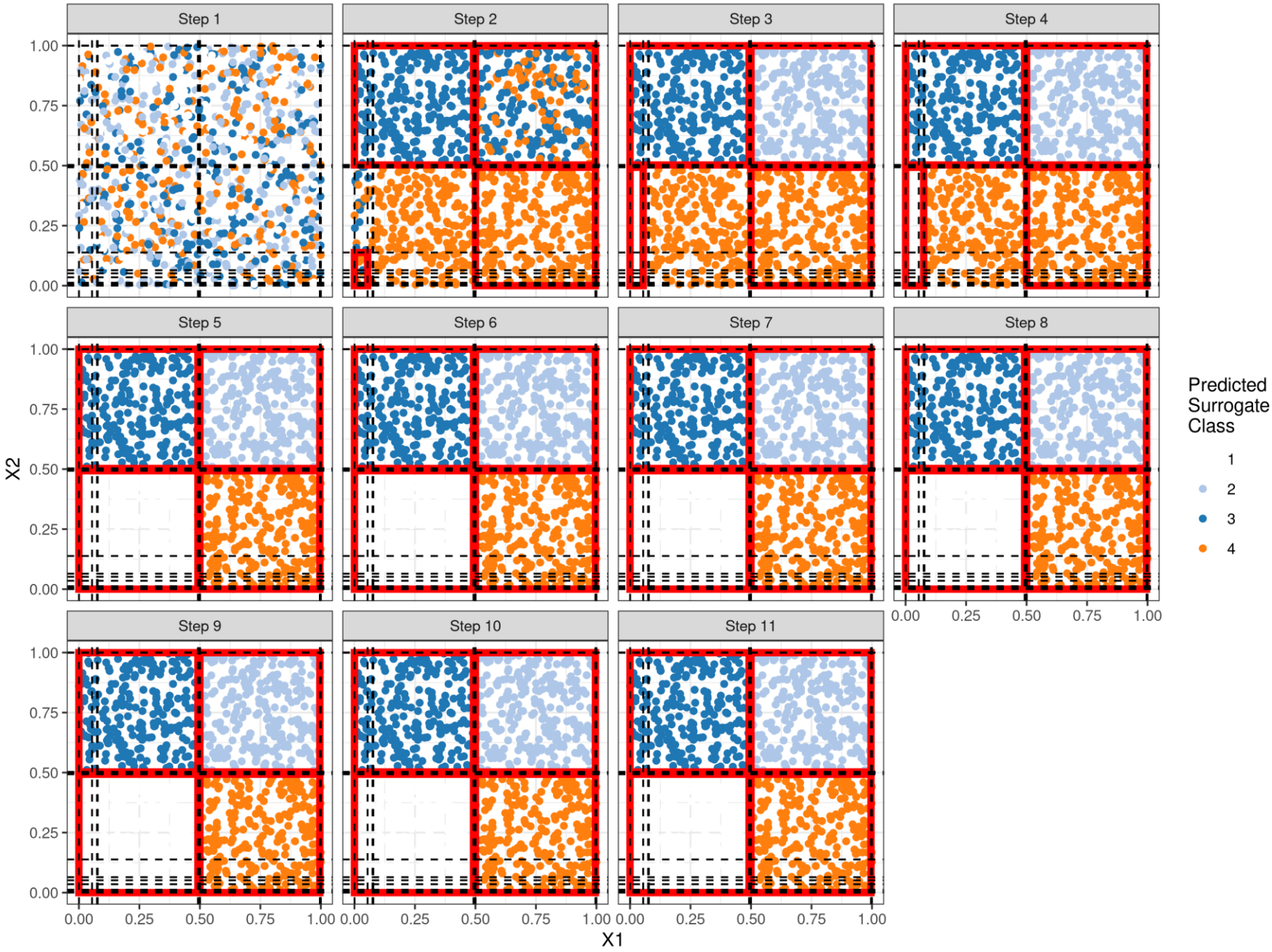
$$p(\boldsymbol{s}(\boldsymbol{x}); \boldsymbol{\eta}, k) = \prod_{l=1}^{L} \eta_{kl}^{s_l} (1 - \eta_{kl}^{s_l})^{(1-s_l)}$$

Pour rentrer dans le cadre de l'optimisation par l'algorithme EM, nous avons besoin d'une variable latente $\boldsymbol{u}$. Ce vecteur $\boldsymbol{u} \in \{0,1\}^K$ décrit à quelle région $\boldsymbol{x}$ appartient ($u_k = \mathbb{1}_{x \in R_k}$). Nous pouvons ainsi définir la probabilité d'observer le couple $(y, \boldsymbol{s(x)})$ sachant $\boldsymbol{u}$:

$$p(y, \boldsymbol{s(x)}|\boldsymbol{u}) = \prod_{k=1}^{K} \left(p(y; \boldsymbol{\lambda}, \boldsymbol{\omega}, k) \times p(\boldsymbol{s(x)}; \boldsymbol{\eta}, k)\right)^{u_k}$$

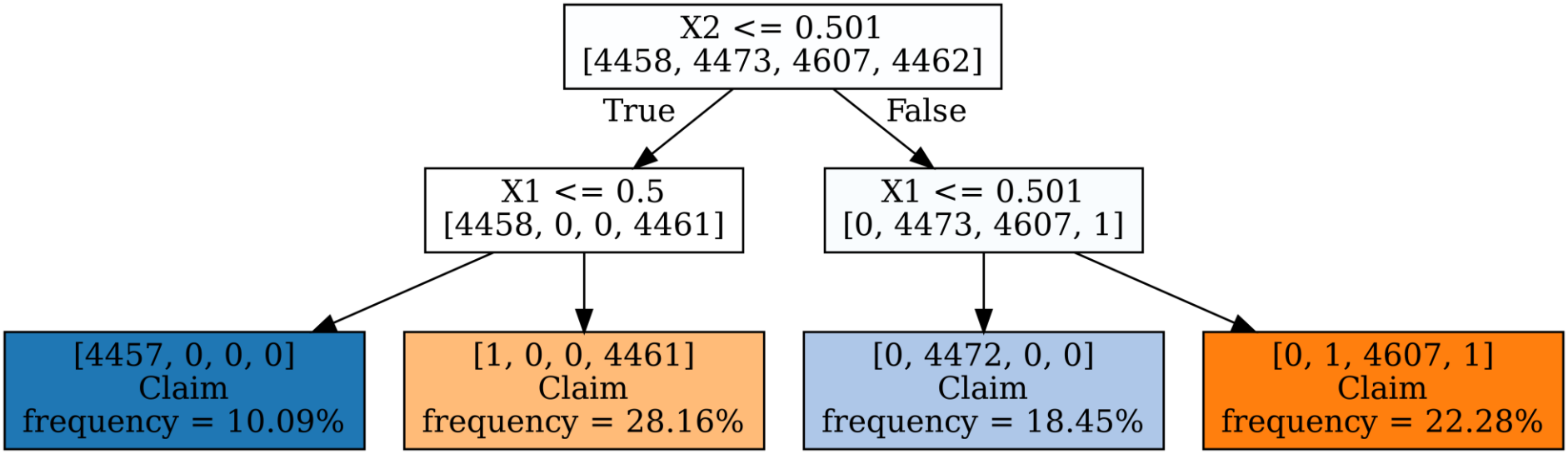
Nous choisissons une distribution de Poisson $p(y; \boldsymbol{\lambda}, \boldsymbol{\omega}, k) \sim \mathcal{P}(\lambda_k \omega_k)$ pour être cohérent avec notre ATM Black-Box.

$$p(y, \boldsymbol{s}(\boldsymbol{x}), \boldsymbol{u}; \boldsymbol{\lambda}, \boldsymbol{\omega}, \boldsymbol{\eta}, \boldsymbol{\alpha}, K) = \prod_{k=1}^{K} \left(p(y; \boldsymbol{\lambda}, \boldsymbol{\omega}, k) \times p(\boldsymbol{s}(\boldsymbol{x}); \boldsymbol{\eta}, k) \times \alpha_k\right)^{u_k}$$


Step 1
Step 2
Step 3
Step 4
Step 5
Step 6
Step 7
Step 8
Step 9
Step 10
Step 11
X1
X2
0.00
0.25
0.50
0.75
1.00
Predicted Surrogate Class
1
2
3
4

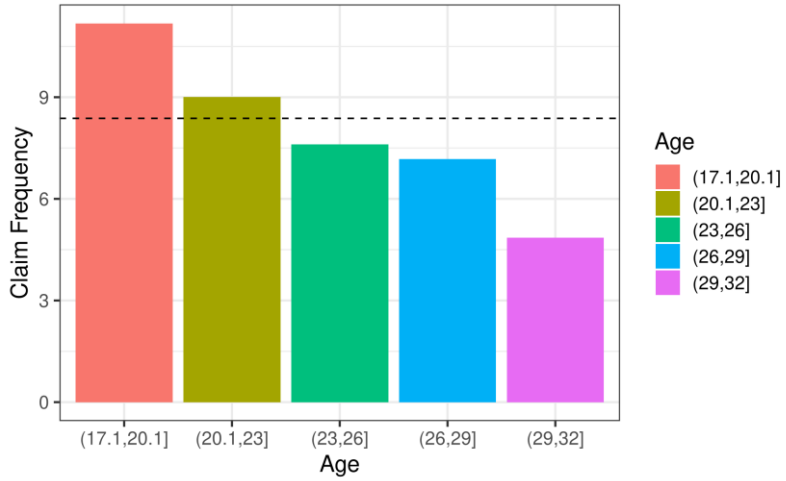
L'algorithme converge vers les paramètres optimaux : $\boldsymbol{\eta}^\star$, $\boldsymbol{\lambda}^\star$, $\boldsymbol{\omega}^\star$, $\boldsymbol{\alpha}^\star$. Ces derniers forment une "soft partition" de l'espace. En conséquence, une observation de l'espace peut appartenir à plusieurs des K classes du modèle surrogate.

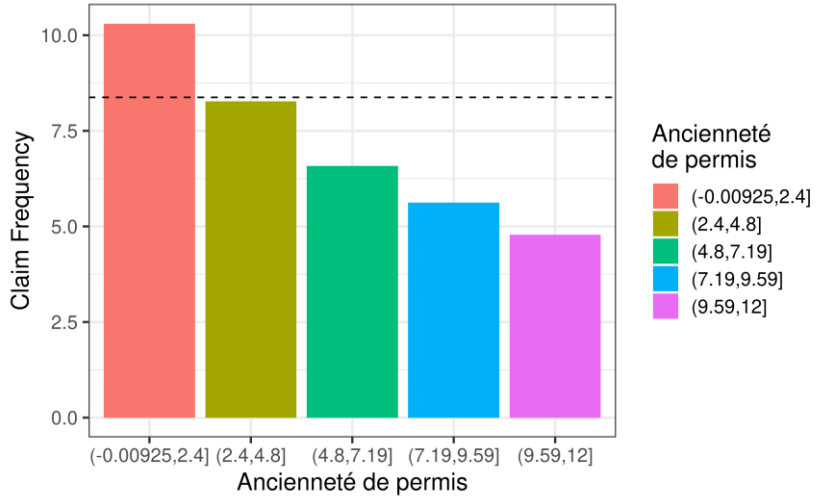
Pour remédier à ce problème nous proposons d'utiliser une version modifiée de l'arbre de classification pour reformer une partition de l'espace. Cet arbre apprend sur les observations et a pour cible la classe affectée par le surrogate. Sa recherche est restreinte aux splits initiaux pour assurer la cohérence de l'explication.

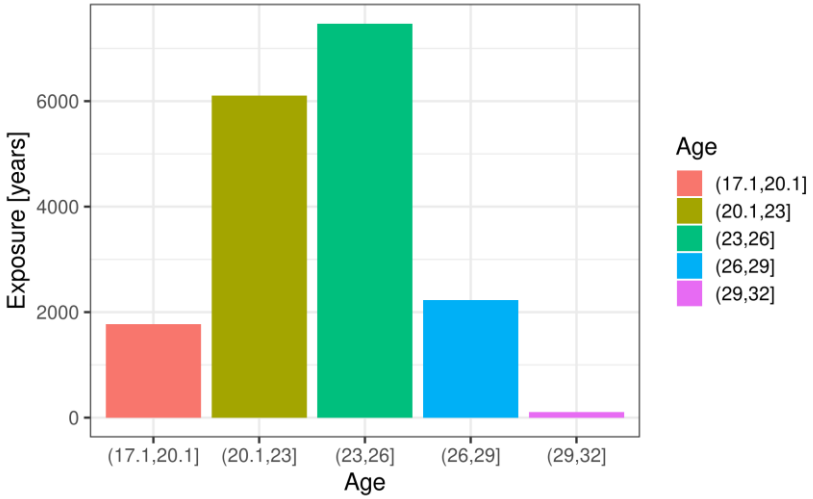

X2 <= 0.501
[4458, 4473, 4607, 4462]
True
False
X1 <= 0.5
[4458, 0, 0, 4461]
X1 <= 0.501
[0, 4473, 4607, 1]
[4457, 0, 0, 0]
Claim
frequency = 10.09%
[1, 0, 0, 4461]
Claim
frequency = 28.16%
[0, 4472, 0, 0]
Claim
frequency = 18.45%
[0, 1, 4607, 1]
Claim
frequency = 22.28%

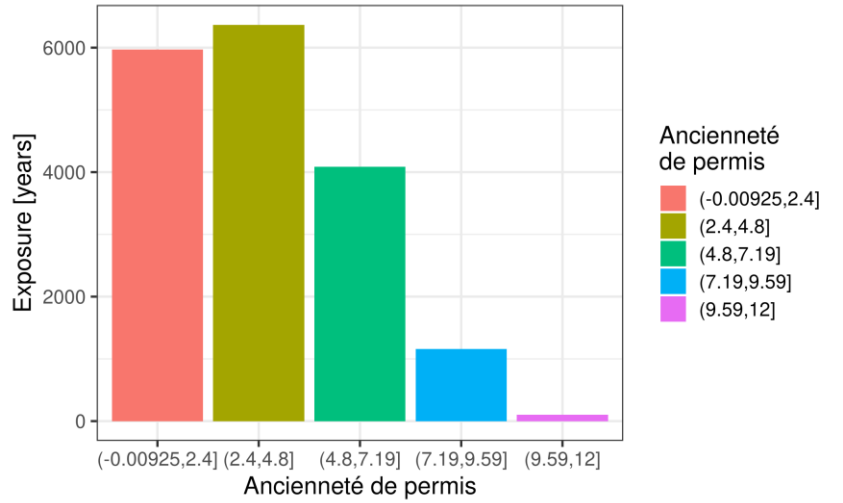
- Le jeu de données contient 33 259 observations
- Le portefeuille est majoritairement composé de jeunes conducteurs.
- Il y a une fréquence sinistre moyenne de 8.38% sur le portefeuille global si l'on considère le temps comme exposition.

Dans le jeu de données, nous avons des variables classiques de la police telles que l'âge, l'ancienneté de permis et le sexe par exemple. Ces variables semblent être discriminantes pour la prédiction de la fréquence sinistre.

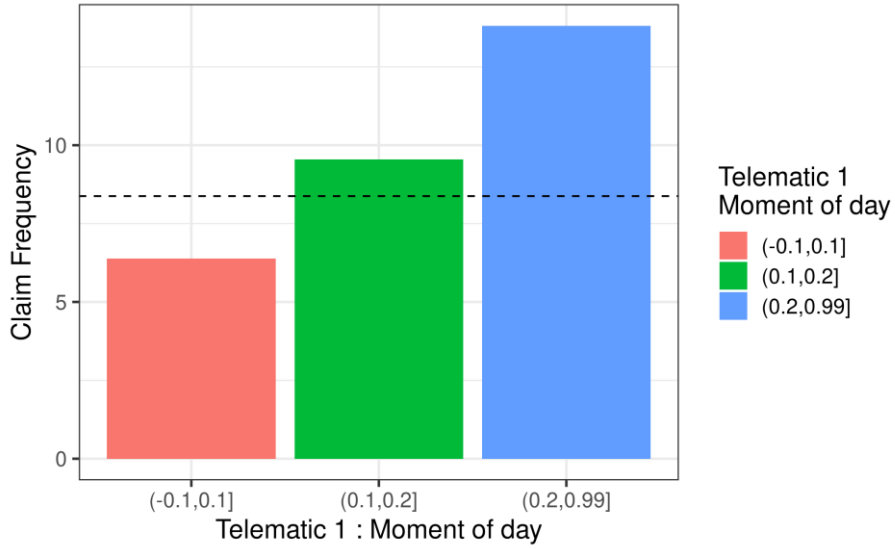


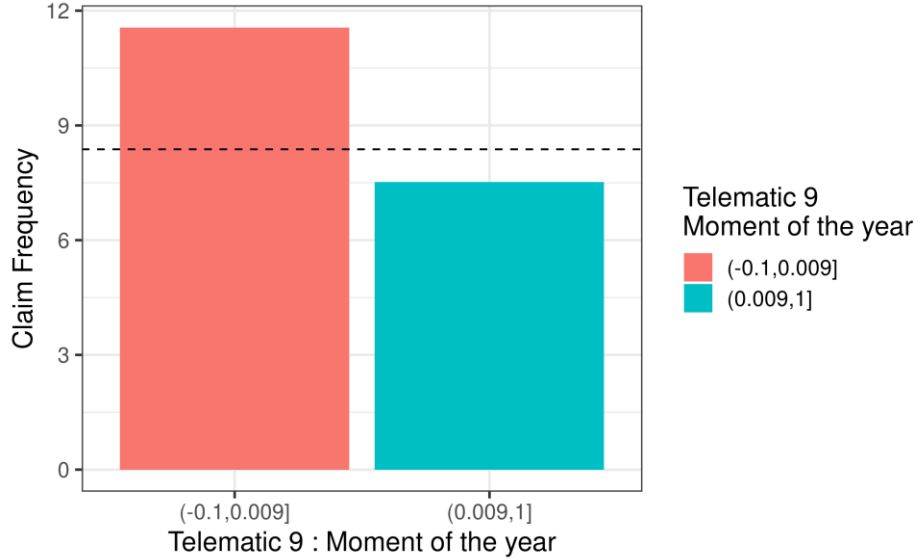


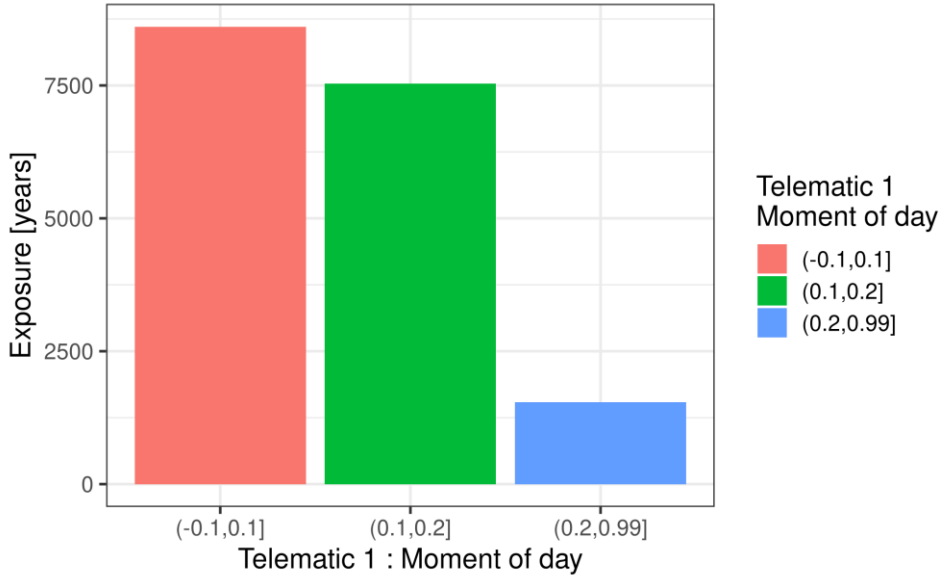


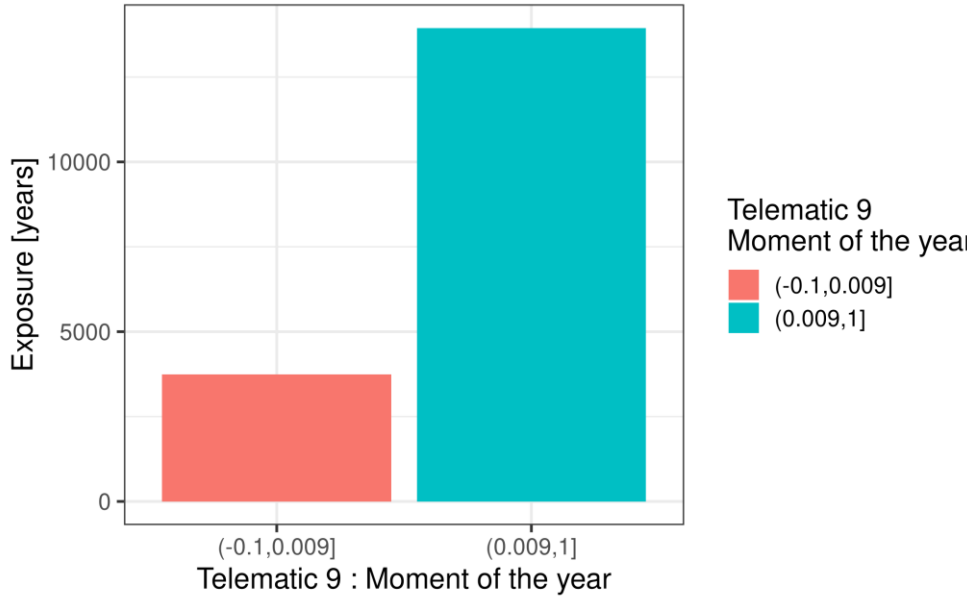


Le jeu de données contient aussi des variables originales collectées par des boîtiers embarqués dans les voitures pendant les trajets.



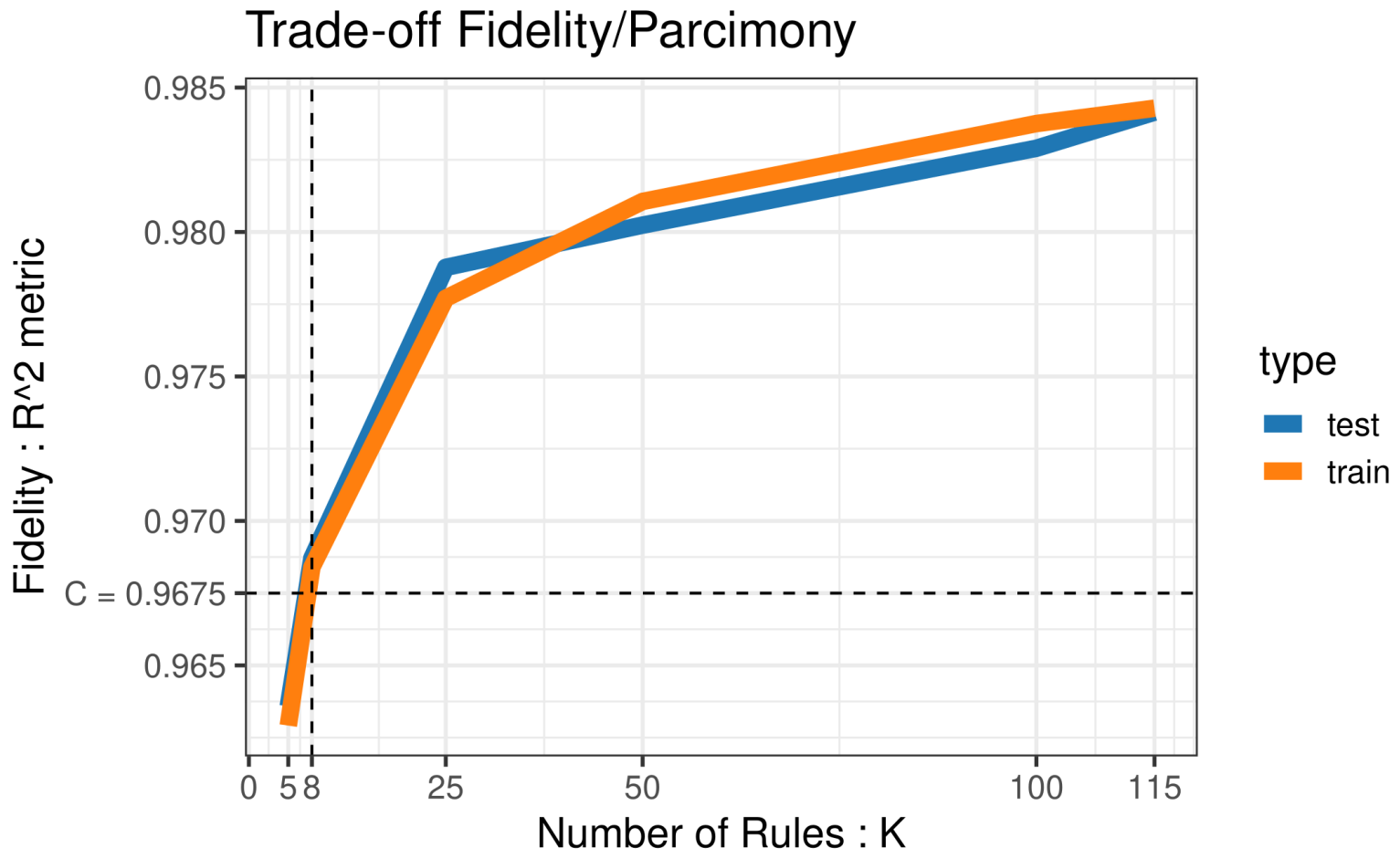


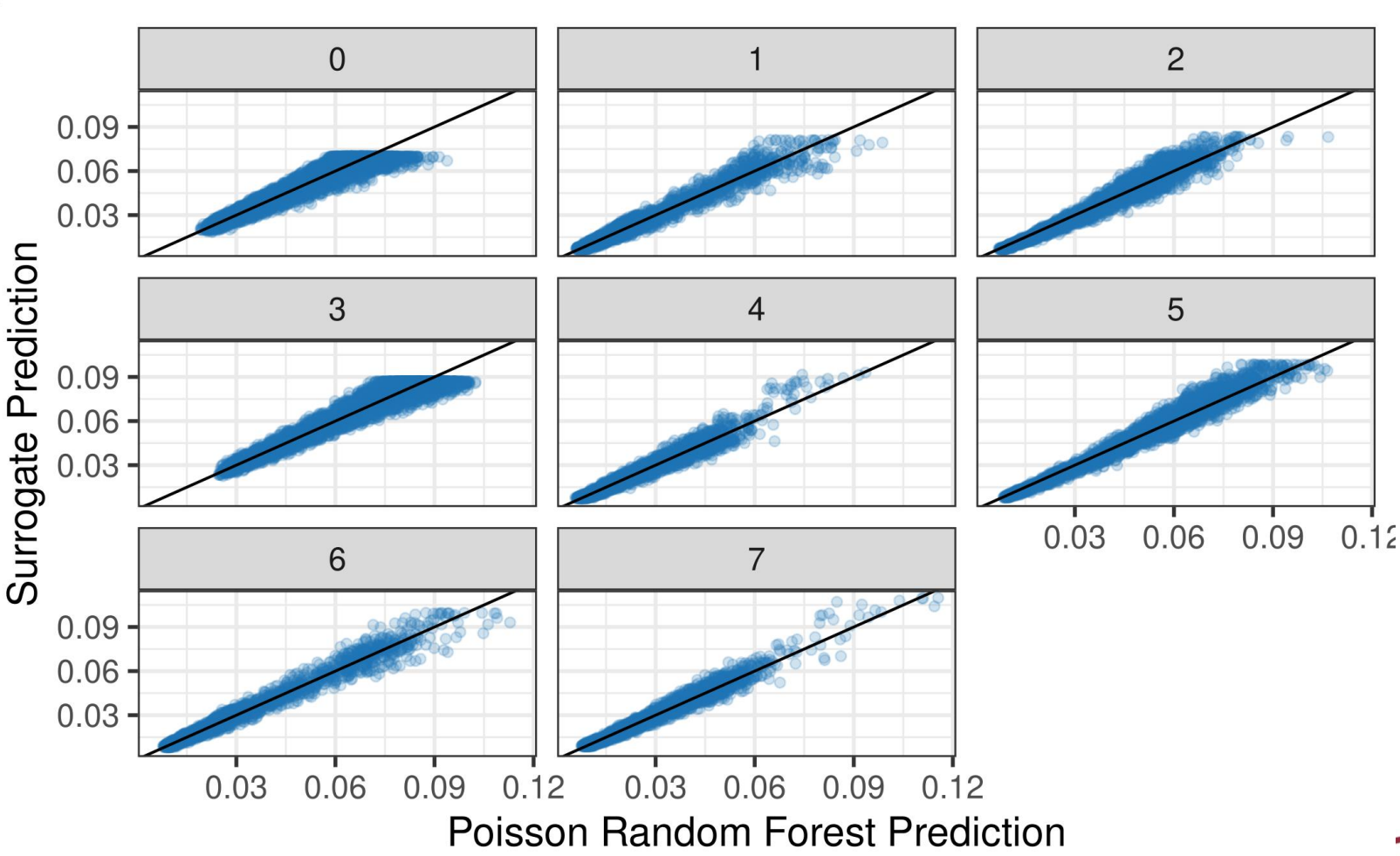


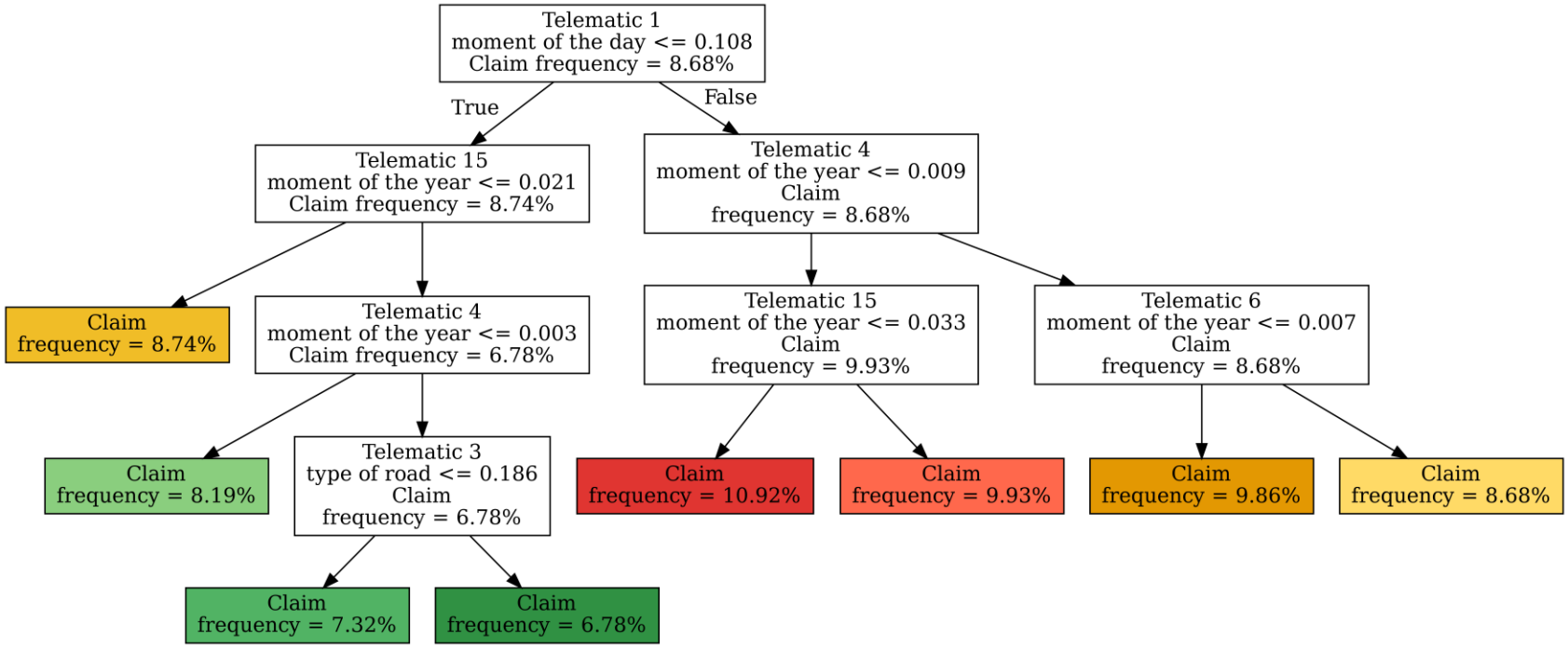


Pour modéliser la fréquence sinistre avec des ATM poissonniens, nous avons choisi le Poisson Random Forest implémenté dans le package R **RfCountData** ainsi que le Gradient Boosting implémenté dans le package **XGBoost**

Tous les deux permettent de gérer la variable d'offset aussi appelée **exposure** chez les actuaires.


Trade-off Fidelity/Parcimony
Fidelity : R^2 metric
0.985
0.980
0.975
0.970
C = 0.9675
0.965
0
5
8
25
50
100
115
Number of Rules : K
type
test
train


0
1
2
3
4
5
6
7
0.09
0.06
0.03
0.03
0.06
0.09
0.12
Surrogate Prediction
Poisson Random Forest Prediction


Telematic 1
moment of the day <= 0.108
Claim frequency = 8.68%
True
False
Telematic 15
moment of the year <= 0.021
Claim frequency = 8.74%
Telematic 4
moment of the year <= 0.009
Claim
frequency = 8.68%
Claim
frequency = 8.74%
Telematic 4
moment of the year <= 0.003
Claim frequency = 6.78%
Telematic 15
moment of the year <= 0.033
Claim
frequency = 9.93%
Telematic 6
moment of the year <= 0.007
Claim
frequency = 8.68%
Claim
frequency = 8.19%
Telematic 3
type of road <= 0.186
Claim
frequency = 6.78%
Claim
frequency = 10.92%
Claim
frequency = 9.93%
Claim
frequency = 9.86%
Claim
frequency = 8.68%
Claim
frequency = 7.32%
Claim
frequency = 6.78%

- DefragTrees permet de donner un bon surrogate en termes de fidélité
- Le framework est assez général pour s'adapter à la majorité des distributions utilisées en actuariat
- L'algorithme ne fournit pas une suite d'arbres emboités

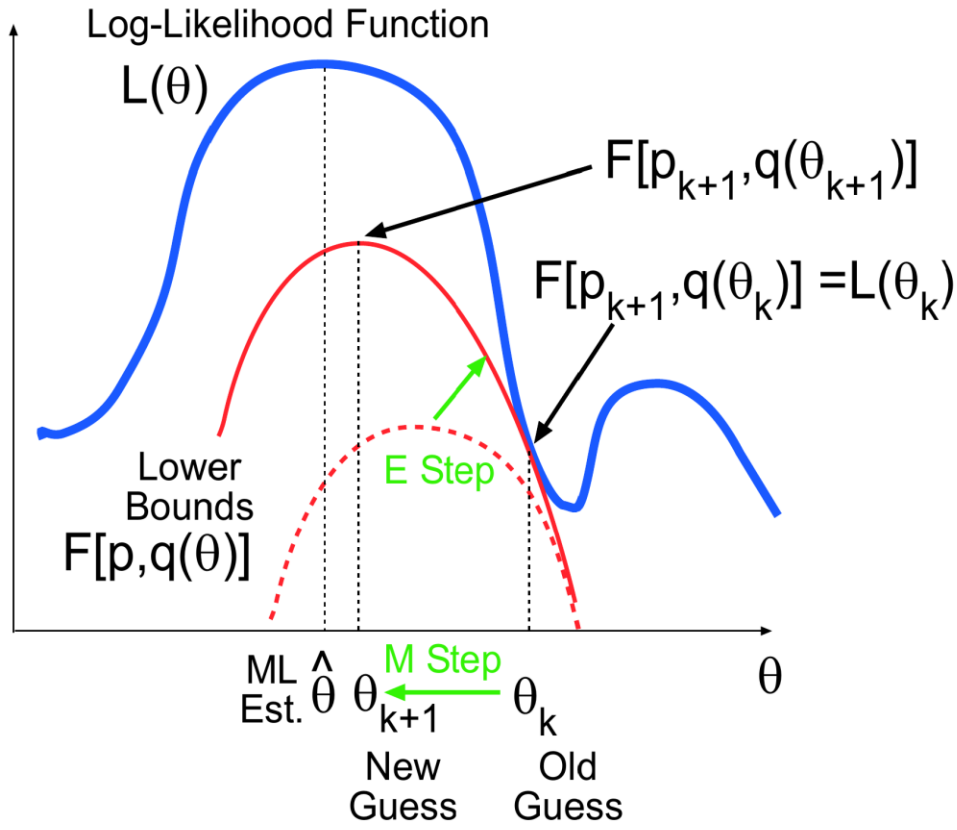


Figure 1: The EM algorithm alternates between finding a greatest lower bound ("E step"), and maximizing this bound ("M step").

Grâce à l'expression précédente, il est possible d'obtenir une expression pour les étapes Expectation (E) et Maximization (M) de l'algorithme EM.

E step :

Pour les paramètres courants $\Pi = \{\boldsymbol{\lambda}, \boldsymbol{\omega}, \boldsymbol{\eta}, \boldsymbol{\alpha}, k\}$

$$q(u_k^{(n)} = 1) = \frac{p(y^{(n)}|\boldsymbol{\lambda}, \boldsymbol{\omega}, k)p(s^{(n)}|\boldsymbol{\eta}, k)p(k; \boldsymbol{\alpha})}{\sum q(u_k^{(n)})p(y^{(n)}, s^{(n)}, u_k^{(n)}|\Pi, K)}$$

M step :

En posant $q(u_k^{(n)} = 1) = \beta_k^{(n)}$

$$\eta_{k,l} = \frac{\sum_{n=1}^{N} \beta_k^{(n)} s_l^{(n)}}{\sum_{n=1}^{N} \beta_k^{(n)}}, \quad \lambda_k = \frac{\sum_{n=1}^{N} \beta_k^{(n)} y^{(n)} \omega_k^{(n)}}{\sum_{n=1}^{N} \beta_k^{(n)} \omega_k^{(n)}}, \quad \alpha_k = \frac{1}{N} \sum_{n=1}^{N} \beta_k^{(n)}$$

Pour implémenter ceci, nous avons modifié le package python **DefragTrees**.