Quelques notes sur l'apprentissage machine

felipe@iro.umontreal.ca

RALI
Dept. Informatique et Recherche Opérationnelle
Université de **Montréal**

V0.001

Last compiled: 5 octobre 2018

Plan

Portraits de quelques approches

Perceptron

Une plateforme amusante

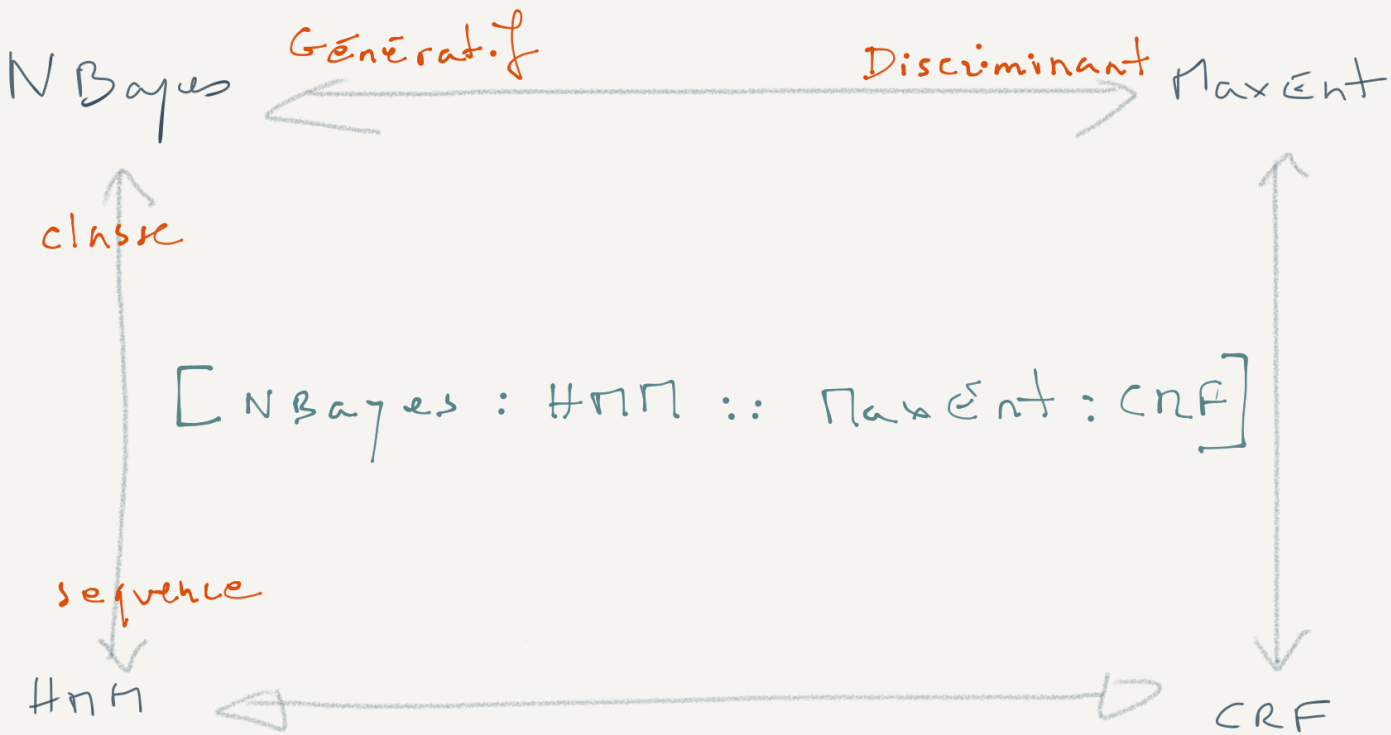
Plan

Portraits de quelques approches

Perceptron

Une plateforme amusante

Analogie quand tu nous tiens



Naive Bayes

in : $D = \{(x_1^n, y)\}$ où $x_i \in \mathcal{X}$ et $y \in \mathcal{Y}$

- $$\begin{aligned} \hat{y} &= \text{argmax}_{y \in \mathcal{Y}}\, p(y|x_1^n) \\ &\propto \text{argmax}_{y \in \mathcal{Y}}\, p(x_1^n, y) \\ &= \text{argmax}_{y \in \mathcal{Y}}\, p(y) \times p(x_1^n|y) \\ &= \text{argmax}_{y \in \mathcal{Y}}\, p(y) \times \prod_i p(x_i|y) \end{aligned}$$

- si l'input est un document d, on peut considérer que x_i est booléen (présence ou non du ie mot du vocabulaire dans d) ou un scalaire (compte ie mot dans d) ; la jointe peut alors être représentée par une binomiale ou une multinomiale respectivement
- on peut sélectionner un sous-ensemble de mots, les lemmatiser ou non, etc. un bon baseline
- Lire ***[McCallum and Nigam, 1998]*** pour une application à la classification de documents

rali

MaxEnt

in : $D = \{(x_1^n, y)\}$ où $x_i \in \mathcal{X}$, $y \in \mathcal{Y}$, et m fonctions booléennes $f_j(x, y)$

- soit $n(x, y) = \exp\left(\sum_{j=1}^{m} \lambda_j f_j(x, y)\right)$,

$$\begin{aligned} \hat{y} &= \operatorname{argmax}_{y \in \mathcal{Y}} p_\lambda(y|x) \\ &= \operatorname{argmax}_{y \in \mathcal{Y}} \frac{n(x,y)}{\sum_{y \in \mathcal{Y}} n(x,y)} \end{aligned}$$

- entraînement (λ_i) par une variante de EM ou descente de gradient (une couche softmax sans couche cachée). Intuitivement, c'est le modèle parmi ceux *compatibles avec le corpus d'entraînement* qui a la plus grande entropie croisée : $H(y|x) = \sum_{(x,y)} p(y, x) \log p(y|x)$
- très souple, les fonctions peuvent se recouvrir en terme d'information. Lire ***[Berger et al., 1996]***.
- **note :** on peut simuler un naive bayes avec des fonctions $f_{w,c}(x, y)$ retournant vrai si w apparait au moins c fois dans x

HMM

in : $D = \{(x_1^n, y_1^n)\}$; $x_i \in \mathcal{X}$, $y_i \in \mathcal{Y}$

- décodage à l'aide de Viterbi en temps $o(E^2 \times n)$; E est le nombre d'états et n la longueur d'une séquence

$$\begin{aligned} \hat{y}_1^n = \text{argmax}_{y_1^n} p(y_1^n | x_1^n) &= \text{argmax}_{y_1^n} p(y_1^n, x_1^n) \\ &= \text{argmax}_{y_1^n} \prod_{i=1}^n p(y_i | y_{i-1}) \times p(x_i | y_i) \end{aligned}$$

- estimée des paramètres par fréquence relative (supervisé) ou via EM (transition = bigramme, émission = unigramme)
- cons : ne capture pas les dépendances longues
- Lire ***[Rabiner, 1989]***

CRF (linéaire)

in : $D = \{(x_1^n, y_1^n)\}$; $x_i \in \mathcal{X}$, $y_i \in \mathcal{Y}$, K fonctions valuées $f_k \in \mathbb{R}$

- Inférence via décodage (extension de Viterbi) :

$$\begin{aligned} \hat{y}_1^n &= \text{argmax}_{y \in \mathcal{Y}^n} p(y|x) \\ &= \text{argmax}_{y \in \mathcal{Y}^n} \frac{1}{Z(x)} \underbrace{\exp\left(\sum_i \sum_{k=1}^{K} \lambda_k f_k(y_i, y_{i-1}, x, i)\right)}_{\phi(x,y)} \end{aligned}$$

où $Z(x) = \sum_{y'} \phi(x, y')$

- De très bon packages (ex : Wapiti), approche forte.
- Lire ***[Lafferty, 2001]***.

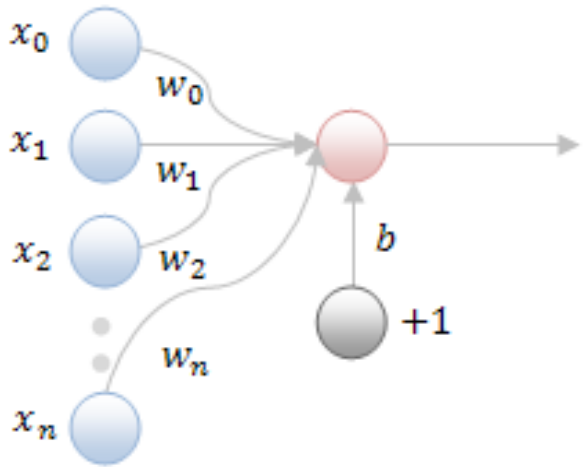
Plan

Portraits de quelques approches

Perceptron

Une plateforme amusante

Perceptron (RosenBlatt, 1957)



- x_j input (valeur réelle)
- w_j poids synaptique (valeur réelle)
- b biais (peut être représenté par un poids supplémentaire et une entrée fixe)
- $h = \sum_j x_j \times w_j + b$ est le signal arrivant au neurone
- l'activation du neurone est déterminée par une fonction d'activation $\hat{y} \equiv f(h)$ (ex : $f = sign$)

Perceptron (RosenBlatt, 1957)

- soit $(x_i, y_i)_{i \in [1,L]}$ un corpus d'entraînement où y_i (ici à valeur dans $\{-1, +1\}$) est la réponse (supervision) et $x_i \in R^{n+1}$
- apprendre les w_j peut se faire en ligne en ajustant itérativement les poids une fois chaque observation x_i rencontrée :
  $\forall j \in [0, n], \quad w_j \leftarrow w_j + \eta(y_i - \hat{y}_i).x_i$
  où η est le learning rate

  résout des problèmes linéairement séparables

Voted-Perceptron *[Freund and Schapire, 1999]*

- Entraînement en ligne :

Require: $\{(\mathbf{x}_i, y_i)\}_{i\in[1,L]}$ où $y_i \in \{+1, -1\}$ et $x_i \in \mathbb{R}^n$
Ensure: a pool of K perceptrons $\{(\mathbf{v}_k, c_k)\}_{k\in[1,K]}$, $v_k \in \mathbb{R}^n$, $c_k \in \mathbb{N}$

$k, c_1 \leftarrow 0$
$\mathbf{v}_1 \leftarrow [0...0]^T$
for all epoch **do**
  for all $i \in [1, L]$ **do**
    $\hat{y} \leftarrow sign(\mathbf{v}_k.\mathbf{x}_i)$
    if $\hat{y} \neq y_i$ **then**
      $\mathbf{v}_{k+1} \leftarrow \mathbf{v}_k + y_i\mathbf{x}_i$
      $c_{k+1} \leftarrow 1$
      $k \leftarrow k + 1$
    else
      $c_k \leftarrow c_k + 1$

Garanties de convergence (même dans les cas non linéairement séparables)

Voted-perceptron *[Freund and Schapire, 1999]*

- Test :

$$\hat{y} = sign\left(\sum_k c_k.sign(\mathbf{v}_k.\mathbf{x})\right)$$

- requiert la sauvegarde de K perceptrons
- calcul en $O(K \times n)$. On peut également ne retenir qu'un sous-ensemble des perceptrons (à l'extrême un seul, par exemple celui de plus fort c_k).

rali

Perceptron structuré *[Collins, 2002]*

$\mathbf{w}_0, \mathbf{w}_a \leftarrow \mathbf{0}$
repeat
  for all example $(x, y) \in D$ **do**
    $\hat{y} = \text{argmax}_y \, \mathbf{w}_0^T \Phi(x, y)$
    if $\hat{y} \neq y$ **then**
      $\mathbf{w}_0 \leftarrow \mathbf{w}_0 + \Phi(x, y) - \Phi(x, \hat{y})$
      $\mathbf{w}_a \leftarrow \mathbf{w}_a + c\Phi(x, y) - c\Phi(x, \hat{y})$
      $c \leftarrow c + 1$
until converged
return $\mathbf{w}_0 - \mathbf{w}_a / c$

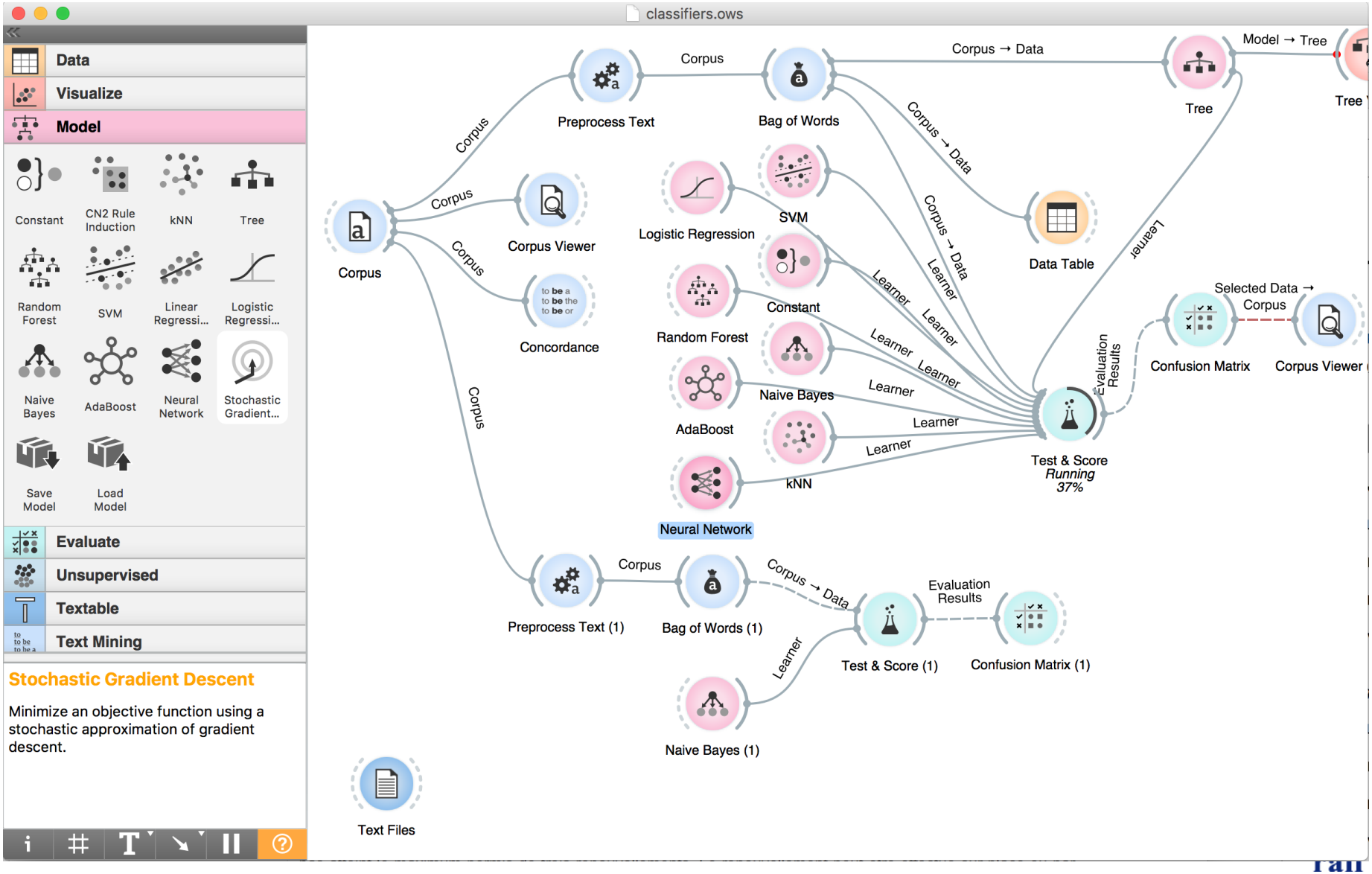
Plan

Portraits de quelques approches

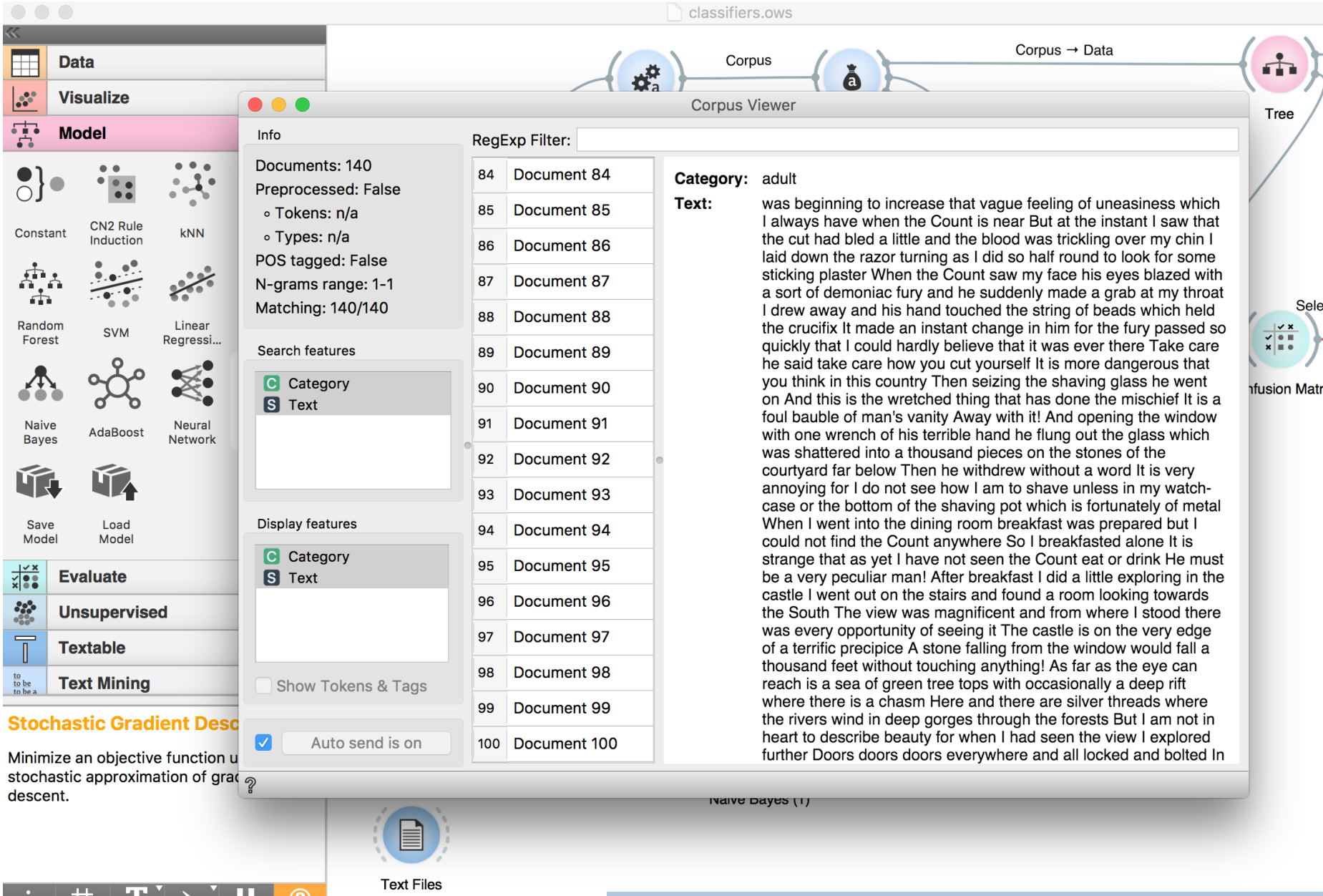
Perceptron

Une plateforme amusante

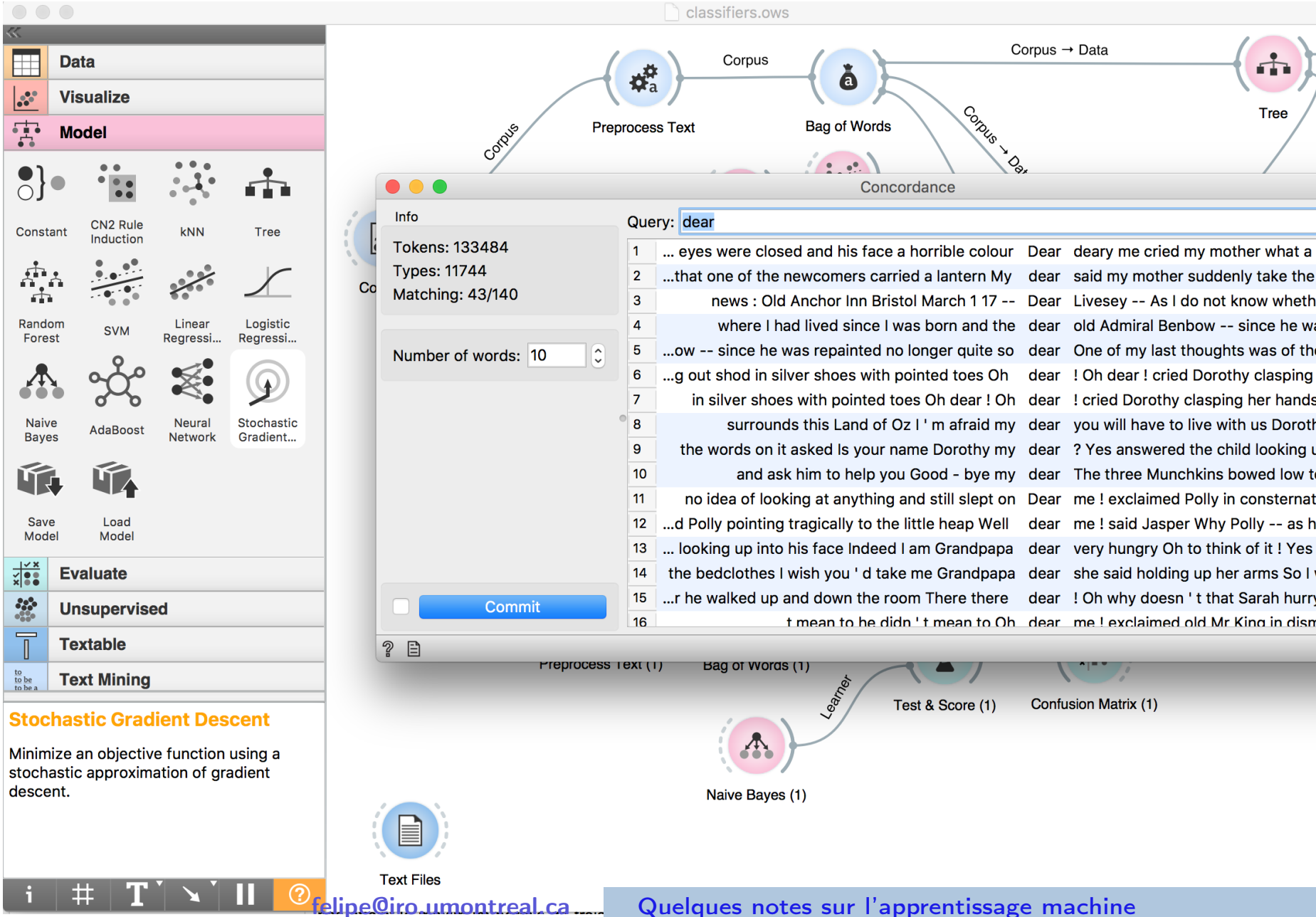
Orange : une expérience



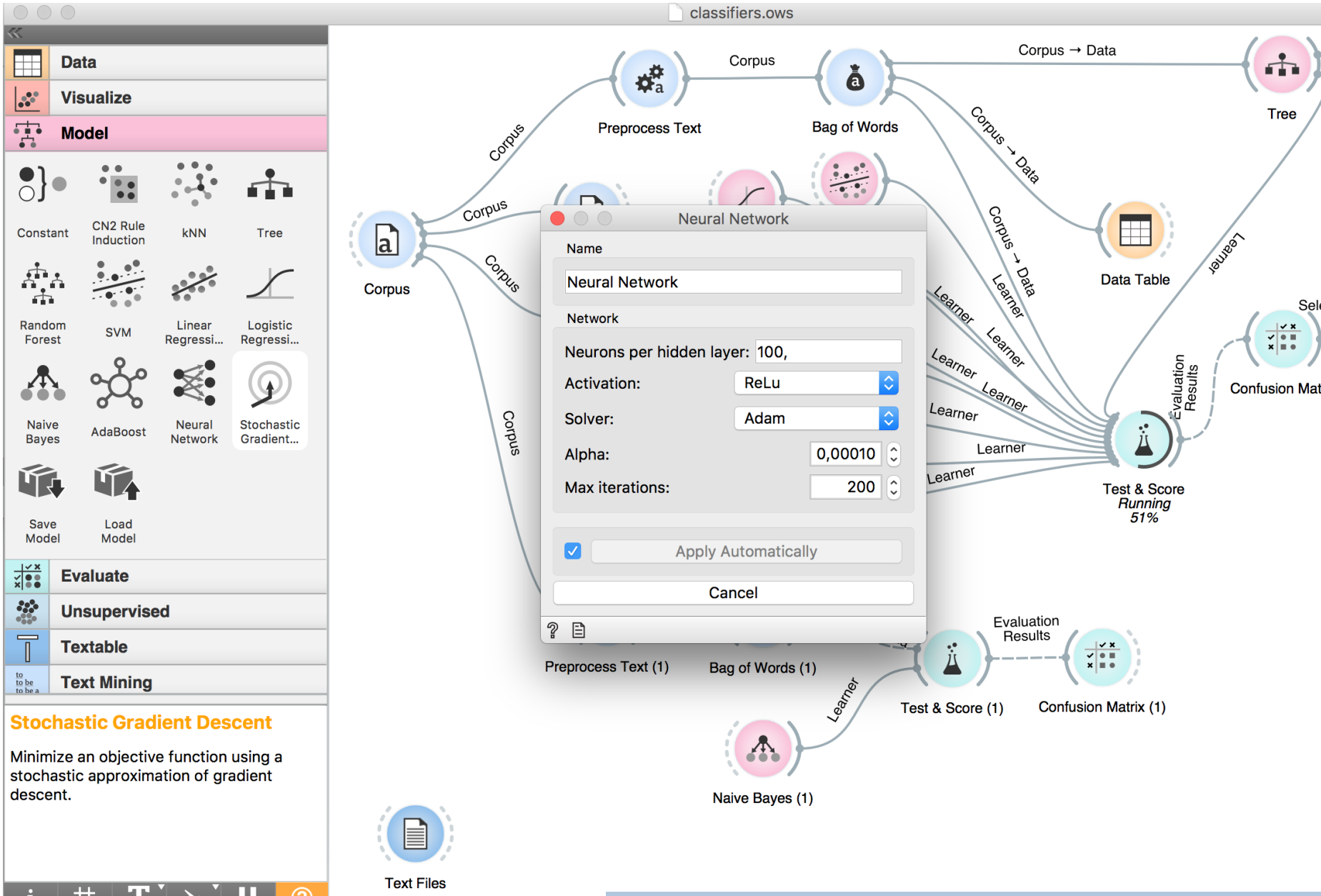
Orange : visualisation



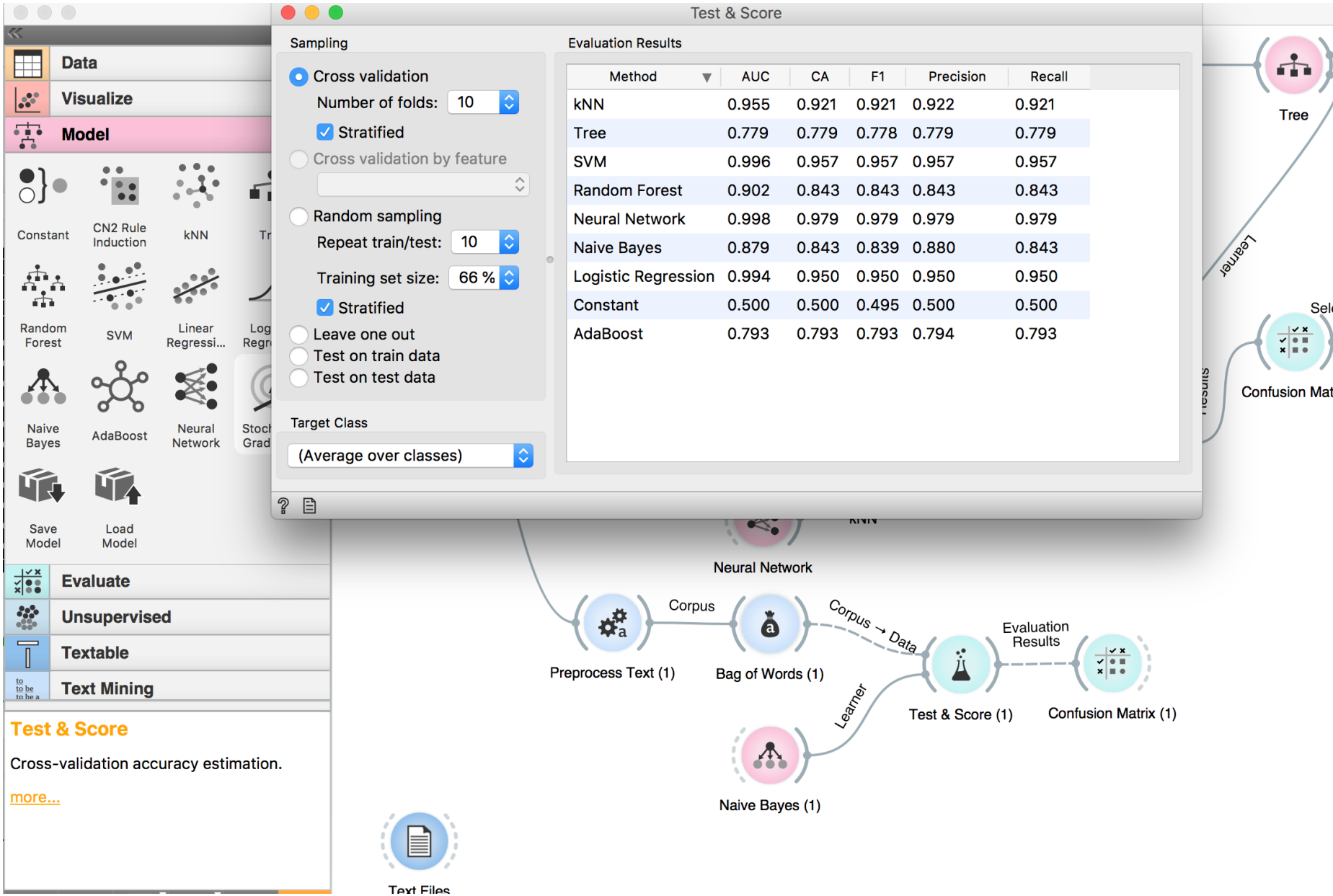
Orange : concordancier



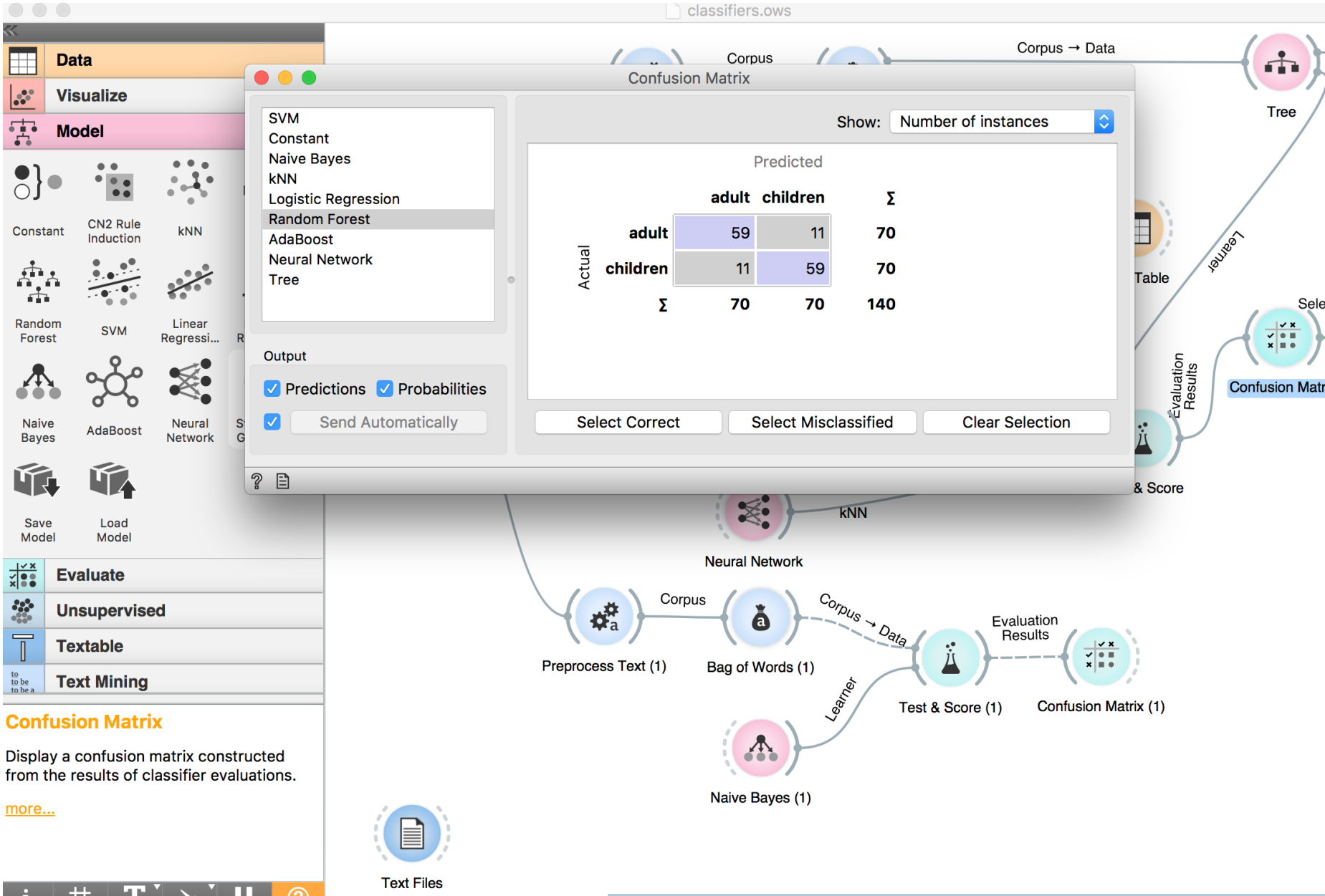
Orange : neural network



Orange : résultats



Orange : matrice de confusion



Berger, A. L., Pietra, V. J. D., and Pietra, S. A. D. (1996).
A maximum entropy approach to natural language processing.
Comput. Linguist., 22(1) :39–71.

Collins, M. (2002).
Discriminative training methods for hidden markov models : Theory and experiments with perceptron algorithms.
In *Proceedings of the ACL-02 Conference on Empirical Methods in Natural Language Processing - Volume 10*, EMNLP '02, pages 1–8.

Freund, Y. and Schapire, R. E. (1999).
Large margin classification using the perceptron algorithm.
Mach. Learn., 37(3) :277–296.

Lafferty, J. (2001).
Conditional random fields : Probabilistic models for segmenting and labeling sequence data.
In *Proc. 18th International Conf. on Machine Learning*, pages 282–289. Morgan Kaufmann.

rali

McCallum, A. and Nigam, K. (1998).
A comparison of event models for naive Bayes text classification.
In *Learning for Text Categorization : Papers from the 1998 AAAI Workshop*, pages 41–48.

Rabiner, L. R. (1989).
A Tutorial on Hidden Markov Models and Selected Applications in Speech Recognition, chapter 6.
IEEE.