Seveyrat Camille
22303443

Boccaccio melissa
22400372

L2 MIASHS
2024/2025

Science des données 1

Classification supervisée et non supervisée pour l'analyse de donné textuelle portant sur des tweets sexiste

Nefissa Khiari

Dans ce rapport, nous avons présenté l'ensemble de nos analyses effectuées sur un seul et même jeu de données textuelles constitué de tweets, récupéré sur le site kaggle.com. Nous avons commencé par détailler le contenu de notre jeu de données ainsi que la démarche que nous avons adoptée pour rendre notre jeu de données exploitable. Avec le logiciel Orange, nous avons ensuite mis en place une chaîne de traitement pour chaque type d'apprentissage, à savoir l'apprentissage supervisé et l'apprentissage non supervisé. Ainsi, nous avons pu interpréter les résultats obtenus pour chaque type d'apprentissage et comparer nos résultats en cherchant à comprendre la raison de la qualité de leurs performances.

Table des matières

1 - Apprentissage non supervisé

1.1 - Jeu de données

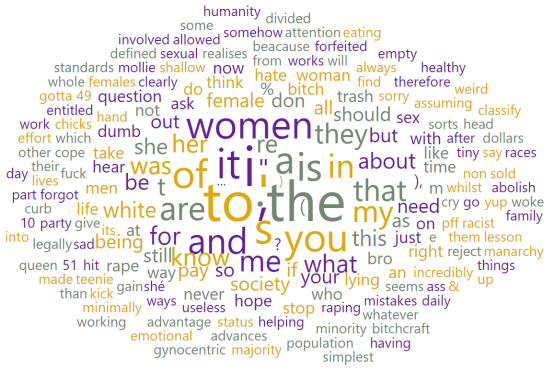
Nous avons choisi un jeu de données où chaque individu est une donnée textuelle représentant un tweet. Nous avons pris 40 tweets parmis plus de 2000. Parmi ces 40 tweets, certains sont catégorisés comme sexistes et d'autres comme non sexistes, les non sexistes étant des tweets de la vie quotidienne. Ce jeu de données nous servant également pour l'apprentissage supervisé, nous savons par avance que parmi ces 40 tweets, 20 sont classés dans la catégorie *sexist* et les 20 autres sont dans la catégorie *not sexist*. Le but de cet apprentissage non supervisé est d'observer la capacité de l'algorithme à effectuer un bon clustering des types de tweets (sans que la catégorisation soit fournie à l'algorithme). Le fait de connaître la catégorie de chaque tweet est intéressant pour voir l'efficacité de l'algorithme, mais nous en reparlerons après.

Nous avons trouvé notre jeu de données sur le site Web kaggle.com, "Sexism Detection in English texts".

1.2 - Chaîne de traitement

Pour chaque tweet, nous avons fait un pré-nettoyage : nous avons remplacé ou enlevé les caractères spéciaux ou incorrects dans le fichier texte directement.

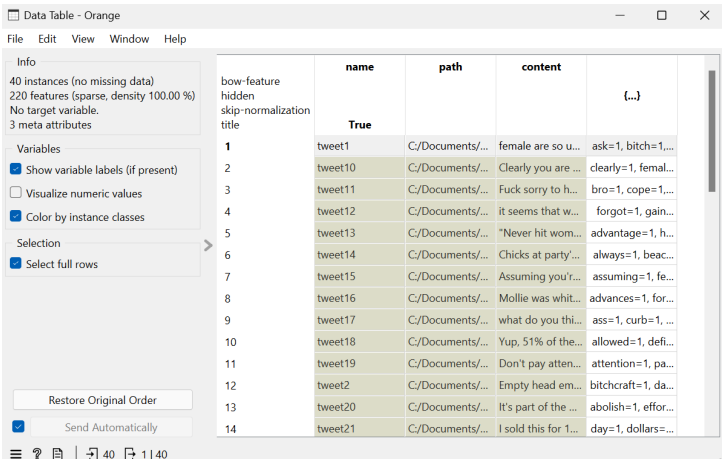
Ensuite, dans le logiciel Orange, nous avons vu avec l'outil *Word Cloud* (directement en sortie de Document Import, comportant le fichier "non supervisé"), tous les mots qu'il fallait filtrer. Nous avons utilisé l'outil *Preprocess Text*, en s'assurant de bien avoir choisi l'anglais, pour nettoyer nos données en enlevant la ponctuation, les nombres, les majuscules, les verbes….On a également ajouté un document texte contenant des *stop words* (mots d'arrêt) pour les mots qui ne sont pas pertinents pour notre analyse et qui pourraient fausser nos résultats, tel que "way", "need", "like"...



word cloud avant Preprocess Text

après Preprocess Text

En sortie de *Preprocess Text*, nous avons mis l'outil *Bag of Words* qui est essentiel pour pouvoir faire des clustering hiérarchiques car il permet de transformer du texte brut en données numériques exploitables pour la classification. Nous avons choisi "*Count"* pour *Term Frequency* car cela permet de compter la fréquence des mots par tweet, et c'est ce qui nous intéresse pour le clustering.
Puis à la sortie de cela, nous avons mis *Distance* et *Data Table*. Le *Data Table* nous sert juste à mieux visualiser le travail du *Bag of Words*.

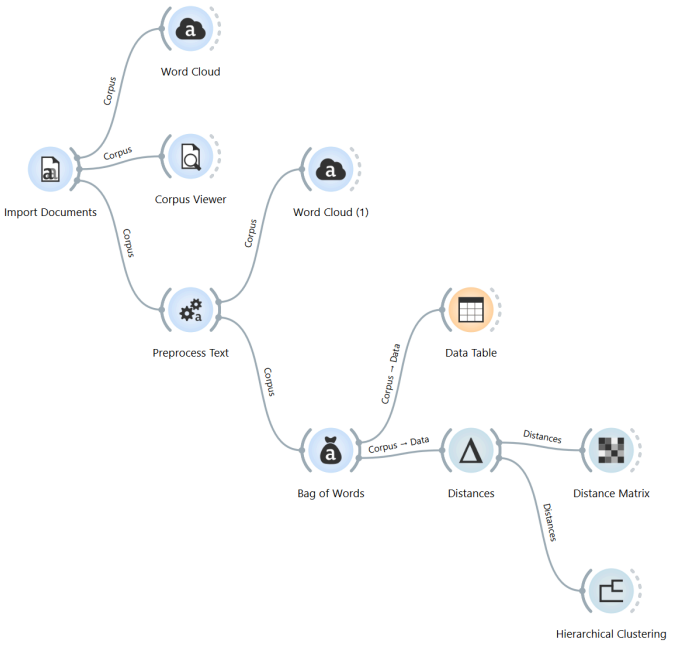


Data Table

La *Distance* quant à elle est également primordiale dans cette chaîne de traitement, elle sert à calculer et visualiser les distances (ou similitudes) entre les éléments d'un jeu de données. Dans ce cas, nous avons choisi la *Distance Metric "Cosine"* car c'est celle qu'on utilise le plus souvent pour des données textuelles. Pour finir cette chaîne, en sortie de *Distance* nous avons mis une Distance Matrix pour visualiser les écarts entre les tweets calculés numériquement grâce à Distance.

Distance Matrix

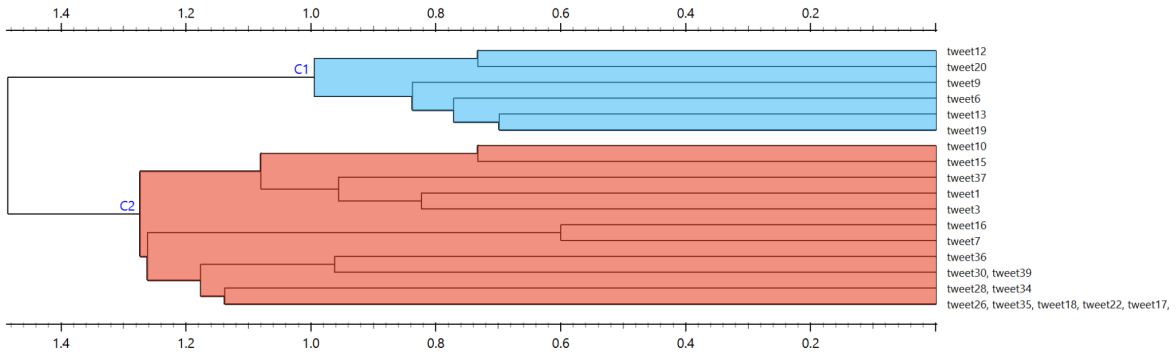
Et on retrouve également l'élément principal, le clustering hiérarchique.



Chaîne de traitement pour l'apprentissage non supervisé

1.3 - Interprétation

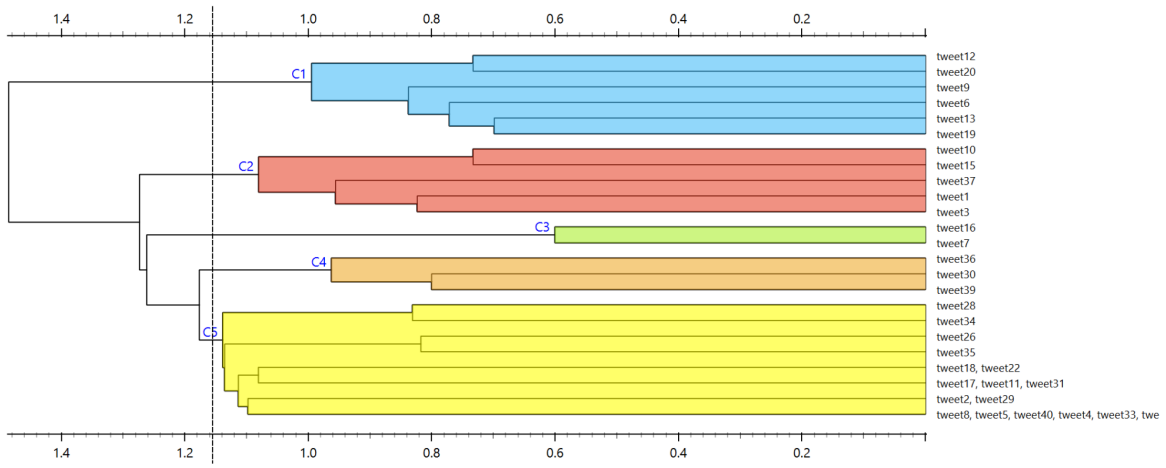
Quand on ouvre notre clustering hiérarchique, on voit ceci :



Pour l'instant on choisit de prendre le nombre de 2 clusters, nos catégories étant *sexist* et *not sexist*. Premièrement, on voit que le nombre d'individus répartis dans les clusters est inéquitable. C'est ici que le fait d'avoir pris le même jeu de données pour les deux apprentissages nous est bénéfique, car on sait d'avance que dans les deux catégories le nombre d'individus est identique (i.e 20 individus par cluster). Or ici on observe que ce n'est pas le cas.

Comme nous connaissons la catégorie de chaque tweet, nous pouvons nous y intéresser de plus près : les tweets numérotés de 1 à 20 sont censés appartenir à la catégorie *sexist*, et les 20 autres à la catégorie *not sexist*.
On peut donc noter que le cluster C1 ne comporte que des tweets appartenant à la catégorie *sexist*.

Maintenant, nous pouvons essayer de pousser notre analyse en examinant nos clusters avec différentes lignes de coupe, ce qui permet d'obtenir un nombre variable de clusters.
Notre jeu de données comporte en effet deux catégories, *sexist* et *not sexist*, mais à l'origine, il existait quatres sous-catégories pour les tweets *sexist* : "*aggressive and emotive attacks*","*discrimination against women*", "*gender stereotypes*"et "*plans to harm*".
Nous savons également à quelle sous-catégorie chaque tweet *sexist* appartient. On peut donc se demander si, lorsque l'on change le nombre de clusters, l'algorithme réussit à créer des sous-catégories similaires.
On change donc la ligne de coupe pour pouvoir avoir 5 clusters, qui correspondent au 5 catégories que nous connaissons : "*aggressive and emotive attacks*","*discrimination against women*", "*gender stereotypes*", "*plans to harm*" et "*not sexist*".

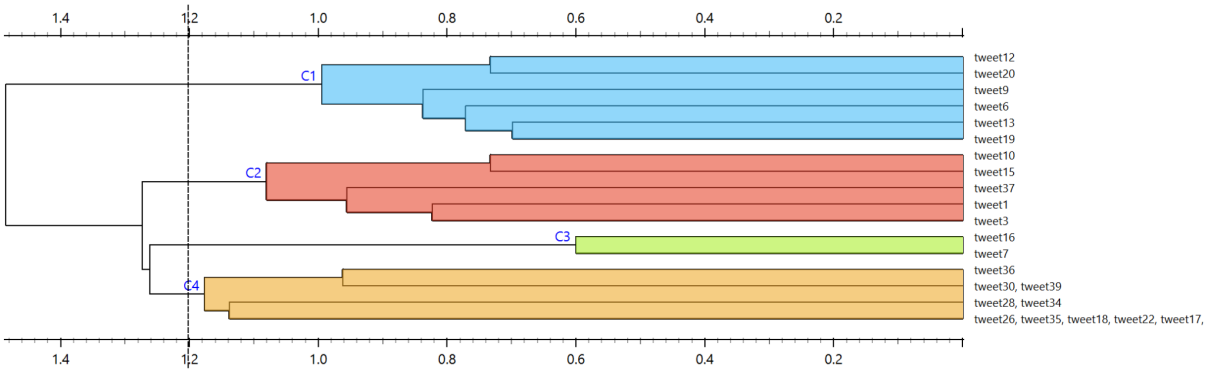


En mettant en sortie du clustering hiérarchique un *Corpus Viewer*, il nous est plus simple de voir quels tweets appartiennent à quels clusters, il nous faut juste sélectionner le ou les clusters qui nous intéressent en même temps que le *Corpus Viewer* soit ouvert.

Premièrement, dans le cluster C5, on peut supposer qu'il s'agit des tweets *not sexist*, car ils y sont proportionnellement plus nombreux. Dans ce cas, 8 tweets sur 24 sont mal classés et donc **33,33 %** des données sont mal classées (tweets *sexist*) et **66,67 %** des

données sont bien classées (tweets *not sexist*) (dans ce calcul, on ne prend pas en compte qu'il n'y a que 20 tweets *not sexist*).
Ensuite, dans le cluster C4, il n'y a que des tweets *not sexist*.
Quant au cluster C3, il ne comprend que 2 tweets, les deux appartenant à la catégorie *sexist*.
Dans le cluster C2, on peut supposer que c'est une des sous-catégories de *sexist* car il y a 4 tweets sur les 5 qui sont *sexist*.
Et comme déjà vu dans le 1er cluster, tous les tweets sont *sexist*.

Avec cette information, il est donc plus intéressant de faire 4 clusters pour faire rejoindre le clusters C4 et C5, que nous avons supposé être des clusters de tweets *not sexist*. Il ne faut pas en faire 3 car l'algorithme trouve plus de similitudes entre les tweets du cluster C3 (*sexist*) et les tweets des clusters C4 et C5 (*not sexist*) ce qui les rassemblera et créera donc une plus grande proportion de données mal classées.
On pourrait également calculer le coefficient de silhouette, mais dans ce cas ce ne sera pas vraiment nécessaire.



Donc quand on choisit 4 clusters :
On a C1, C2 et C3 qui sont des clusters représentant les tweets *sexist*.
60% des données sont bien classées, pourquoi 60% :

13 − 1 = 12 (car 1 est mal classé parmi les 13 annoncées comme bien classées)

pourcentage bien classé = (12/20)×100 = **60 %** (ici on prend en compte les 20)

Et dans le clusters C4 on a **70,37 %** des données bien classées. C'est avec 4 clusters que les données sont les mieux réparties.

Maintenant, nous allons nous intéresser aux sous-catégories, car c'est une des raisons pour lesquelles nous avons modifié le nombre de clusters.
Dans les clusters contenant en majorité les tweets *sexist* (C1, C2 et C3), on n'y retrouve pas les sous-catégories que nous connaissons. En effet, par exemple C1 ne regroupe pas que les tweets *gender stereotypes* mais un mélange de toutes les sous-catégories

que le jeu de données nous a fournies. C'est la même chose pour C2 et C3. On peut donc se dire que l'algorithme n'a pas bien sous catégorisé les tweets *sexist*.

Mais ce n'est pas si simple, car si on prend les tweets de chaque cluster on voit qu'il y a des similitudes. Pour C1, chaque tweet comporte le mot "women", pour C2 c'est une majorité de "female". Pour C3, c'est "white" et "trash". L'apprentissage non supervisé repose sur l'absence de connaissances préalables sur les classes, il crée donc des classes en fonction de la similitude ou des différences entre les caractéristiques des individus. Il est donc normal que les tweets *sexist* soient classés de cette façon (en fonction des mots similaires et de leur fréquence), et non comme nous avons pu les classer au début du projet, car notre classification était plutôt basée sur un ressenti, des sous-entendus, des sentiments...

Pour résumer, l'apprentissage non supervisé n'est pas très performant pour ce jeu de données et la tâche de classification (trier *sexist* et *not sexist*).
Plusieurs raisons expliquent cela, comme le fait que pour ce jeu de données, il ne faut pas seulement se baser sur les mots des tweets, mais il est important d'analyser le sens de ces tweets. Il y a aussi le fait que lors du traitement de texte, il est difficile de savoir à l'avance si certains mots gênent ou bien s'ils seront utiles.
Cependant, si on change la demande pour créer davantage de clusters (supérieur à deux), on voit que l'algorithme devient un peu plus performant et répond mieux à la demande. Il y a certes quelques marges d'erreur, mais c'est souvent le cas dans ce type d'analyse.

On pourrait donc se demander si avec un apprentissage supervisé, qui n'est pas seulement basée sur la distance entre les mots, la catégorisation serait plus performante.

2 - Apprentissage supervisé

2.1 - Jeu de données

Pour l'apprentissage supervisé nous avons utilisé le même jeu de données car il est intéressant de faire la comparaison entre les deux types d'apprentissage.
Même si le jeu de données est le même, ce n'est pas pour ça que nous allons l'utiliser de la même manière. Tout d'abord, nous allons le diviser : dans les 40 tweets, nous allons en prendre 75% pour l'entraînement du modèle, soit 30 tweets. L'objectif de cet entraînement est de permettre à l'algorithme supervisé d'apprendre à distinguer les catégories. Ainsi, parmi ces 30 tweets, 15 sont classés dans *sexist* et 15 autres sont classés dans *not sexist*. Pour les 10 tweets restants, il y a 5 tweets sexist et 5 autres tweets *not sexist* ; ils nous serviront pour la phase de test, où l'algorithme devra prédire les classes lui-même.Connaître les catégories des tweets permet de mesurer l'efficacité

de l'entraînement en comparant les prédictions de l'algorithme supervisé avec les classes réelles.

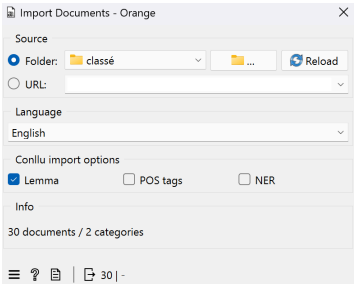
2.2- Chaîne de traitement

Pour le nettoyage des données, c'est exactement la même étape que pour l'apprentissage non supervisé.

Pour l'apprentissage supervisé il faut suivre :

données étiquetées→ modèles d'apprentissage → validation croisée → matrice de confusion.

Pour la chaîne de traitement, nous avons donc commencé avec *Import Documents,*

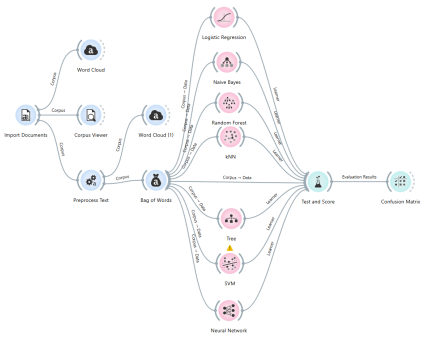


Import Documents

où on retrouve bien les 30 tweets séparés en deux catégories.
Après *Import Documents*, la chaîne de traitement et les processus sont les mêmes que pour l'apprentissage non supervisé jusqu'à *Bag of Words.*

Pour l'apprentissage supervisé, nous avons utilisé sept modèles d'entraînement : *Logistic Regression*, *Naive Bayes*, *Random Forest*, *kNN*, *Tree*, *SVM* et enfin *Neural Network*.
Nous allons donc comparer ces modèles entre eux et prendre le plus performant pour notre jeu de données. Il pourra effectuer par la suite les prédictions dans la phase de test, à savoir classer les tweets en *sexist* ou *not sexist*.
Pour trouver le modèle le plus performant, nous nous sommes appuyées sur deux outils différents et complémentaires.

En sortie de tous ces modèles et de Bag of Words, nous allons relier l'outil *Test and Score*, puis *Confusion Matrix*. Ce sont ces deux outils qui nous servent à comparer nos modèles.



Chaîne de traitement de l'entraînement pour l'apprentissage supervisé

Test and Score - Orange

Cross validation
Number of folds: 2
Stratified
Cross validation by feature
Random sampling
Repeat train/test: 10
Training set size: 66 %
Stratified
Leave one out
Test on train data
Test on test data

Evaluation results for target (None, show average over classes)

Model	AUC	CA	F1	Prec	Recall	MCC
Tree	0.678	0.667	0.625	0.800	0.667	0.447
Random Forest	0.562	0.500	0.384	0.500	0.500	0.000
kNN	0.511	0.467	0.467	0.467	0.467	-0.067
Naive Bayes	0.636	0.600	0.600	0.600	0.600	0.200
Logistic Regression	0.724	0.633	0.612	0.670	0.633	0.302
SVM	0.509	0.467	0.467	0.467	0.467	-0.067
Neural Network	0.680	0.600	0.593	0.608	0.600	0.208

Test and Score

Fold est une option dans le widget de *Test and Score* qui est utilisée pour la validation croisée (*cross-validation*). La validation croisée consiste à diviser les données disponibles en plusieurs sous-ensembles, à entraîner et tester le(s) modèle(s) plusieurs fois sur ces partitions, et à calculer une performance moyenne pour réduire les biais et la variance dans l'évaluation. *Test and Score* nous est utile pour mesurer la qualité d'un modèle.

Nous pouvons ici déterminer les modèles qui semblent les plus performants en fonction de l'accuracy (CA, taux de prédictions correctes), au rappel (recall), à la précision (prec) et à F-mesure (f-score).

$$accuracy = \frac{nb\ de\ données\ correctement\ classés}{nb\ de\ données}$$

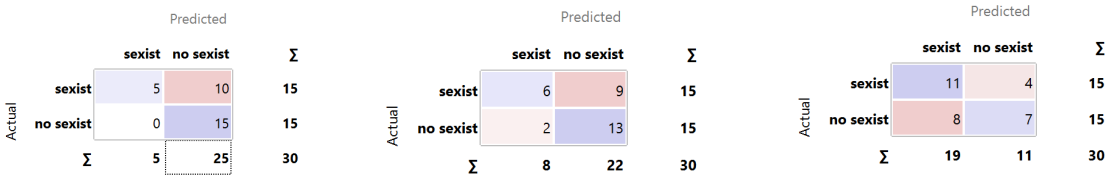
$$rappel_A = \frac{nb\ de\ données\ bien\ classés\ dans\ A}{nb\ de\ données\ réellement\ dans\ A} \qquad rappel = moyenne\ des\ rappels\ de\ toutes\ les\ classes$$

$$précision_A = \frac{nb\ de\ données\ bien\ classés\ dans\ A}{nb\ de\ données\ prédites\ dans\ A} \qquad précision = moyenne\ des\ précisions$$

$$F - mesure = F_1 = moyenne\ harmonique = 2 \cdot \frac{précision \cdot rappel}{précision + rappel}$$

Formules du cour de Science des Données 2

Pour notre jeu de données, nous observons de meilleurs résultats pour les modèles *Tree*, *Logistic Regression* et *Neural Network*. Maintenant, nous allons comparer leur matrice de confusion pour déterminer le meilleur d'entre eux.



Tree *Logistic regression* *Neural Network*

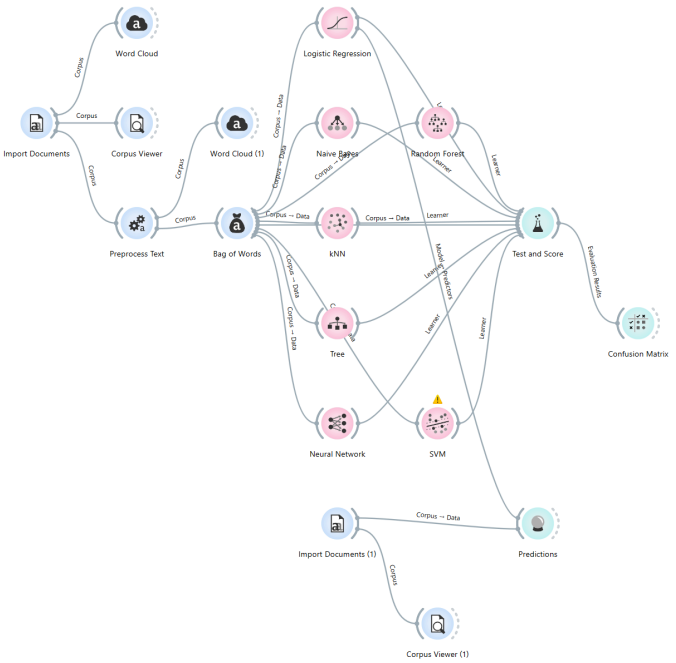
Dans une matrice de confusion, les données bien classées sont visibles dans la diagonale négative, c'est-à-dire lorsque les données *Predicted* et les données *Actual* correspondent.
On voit que *Tree* n'est finalement pas si performant : les données sont presque toutes classées en *not sexist*.
Neural Network n'est pas un mauvais modèle mais il ne se distingue pas par ses performances. En s'appuyant sur les méthodes de validation / comparaison, on en déduit qu'il est mieux de garder *Logistic regression* pour les prédictions.

Méthodes de validation/comparaison :

- Validation croisée
- Matrice de confusion
- Taux de prédictions correctes (accuracy)
- Rappel
- Précision
- F-mesure

Maintenant que nous avons vu la phase d'entraînement dans l'apprentissage supervisé, nous pouvons passer à la phase de test avec les prédictions.

On ajoute un *Imports Documents* où nous allons mettre notre fichier avec les 10 tweets non classés. En sortie, on ajoute *Predictions* et un *Corpus Viewer* (qui nous sert seulement à visualiser les tweets et leur chemin). En entrée de *Predictions*, on ajoute le modèle que nous avons identifié comme étant le plus performant, *Logistic regression*.



Chaîne de traitement pour l'apprentissage supervisé

2.3 - Interprétation

En ouvrant *Predictions* on tombe sur ceci :

Predictions - Orange

Show probabilities for (None)

	Logistic Regression	name	path	content
1	sexist	tweet16	C:/Users/Meliss...	Mollie was whit...
2	sexist	tweet17	C:/Users/Meliss...	what do you thi...
3	sexist	tweet18	C:/Users/Meliss...	Yup, 51% of the...
4	sexist	tweet19	C:/Users/Meliss...	Don't pay atten...
5	sexist	tweet20	C:/Users/Meliss...	It's part of the ...
6	no sexist	tweet21	C:/Users/Meliss...	I sold this for 1...
7	no sexist	tweet22	C:/Users/Meliss...	But if you hate ...
8	no sexist	tweet23	C:/Users/Meliss...	The queen & h...
9	no sexist	tweet24	C:/Users/Meliss...	It is the simples...
10	no sexist	tweet25	C:/Users/Meliss...	Just woke up. H...

Notre modèle entraîné a donc trié les tweets en 2 catégories, *sexist* et *not sexist*. En comparant les résultats de la phase de test obtenus avec les classes réelles, on constate que les prédictions du modèle sont parfaites. En effet, les tweets de 16 à 20 appartiennent bien à la catégorie *sexist* et le reste à *not sexist*.

La prédiction étant seulement basée sur 10 tweets, il serait intéressant de tester la performance du modèle pour un nombre plus élevé de tweets. De même pour les tweets choisis : en effet, si l'on avait choisi des tweets moins explicites, nous pouvons nous demander si le taux de réussite resterait aussi bon ou s'il baisserait légèrement.

L'apprentissage supervisé est une méthode de machine learning qui, si elle est entraînée correctement avec des demandes spécifiques, peut être très utile.

3 - Conclusion

Au cours de ce projet, nous avons pu analyser les deux types d'apprentissage, supervisé et non supervisé, et nous avons donc pu comparer leurs performances sur un même jeu de données, à savoir ici différencier des tweets à caractère sexiste et d'autres tweets n'étant pas sexistes et portant sur des sujets divers de la vie quotidienne.

Cette question d'analyse dépend en effet de plusieurs facteurs à savoir le type de jeu de données, la qualité du nettoyage de ces données, la qualité de l'entraînement pour l'apprentissage supervisé etc. Nous pouvons donc répondre à cette question à partir des choix que nous avons fait lors de ce projet concernant notre jeu de données.

Dans un premier temps, nous avons vu que l'apprentissage non supervisé n'a pas été très performant pour rassembler les tweets d'une même catégorie ensemble. Par la suite, nous avons pu voir qu'au contraire, l'apprentissage supervisé lui, a été très efficace et a su placer les différents tweets fournis lors de la phase de test dans la bonne catégorie.

Nous pouvons donc dire, à partir de nos observations, que l'apprentissage supervisé est bien meilleur pour traiter notre jeu de données. Cela s'explique probablement par la nature des tweets, qui peuvent contenir des sous-entendus, des émotions et tout type de nuance propre au langage humain. Ces éléments subtils sont donc seulement détectables par l'apprentissage supervisé grâce aux données similaires fournies lors de la phase d'entraînement. L'apprentissage non supervisé lui, ne peut pas avoir accès à ces éléments car il ne repose que sur des similitudes présentes entre les données, à savoir ici le vocabulaire.