1 Objectif

Image mining avec Knime. Classement automatique d'images.

La fouille d'images ou image mining est une discipline assez ancienne. Schématiquement, il s'agit d'appliquer des techniques de machine learning au contenu des images c.-à-d. à partir de leurs caractéristiques visuelles. Sa démocratisation est plus récente en revanche. J'y vois plusieurs raisons : la profusion des données images avec le web (big data, etc., etc.) nécessite un savoir faire supplémentaire, on observe d'ailleurs que le traitement d'images est de plus en plus présent dans les challenges ; l'apparition d'outils faciles à appréhender pour les férus de data mining. Auparavant, il fallait une bonne dose de compétences en programmation informatique pour prétendre appréhender ce type de données. Aujourd'hui, grâce à la disponibilité de bibliothèques performantes, relativement faciles à mettre œuvre avec des langages de haut niveau (scikit-image sous Python par exemple), des manipulations naguères hors de portée des non spécialistes du traitement d'images deviennent aujourd'hui accessibles aux statisticiens entre autres. Je ne m'en suis pas privé auprès de mes étudiants ces dernières années. Les vertus des outils me permettent d'aller à l'essentiel sans avoir à passer des heures à expliquer dans le détail les structures de bas niveau des images. Connaissances qui, néanmoins, peuvent être importantes lorsqu'il s'avère indispensable d'ajuster finement les paramètres de l'étude que nous entendons mener.

Le module « Image Processing » de Knime est assez symbolique de cette évolution. Il n'est même pas nécessaire de faire l'apprentissage langage de programmation. On peut réaliser une analyse complète sans avoir à écrire une seule ligne de code. Le plus important est d'avoir une vision globale de la trame de l'étude. Il nous suffit alors de définir dans le bon ordre la séquence des traitements pour obtenir des résultats qui tiennent la route.

Les étapes sont finalement assez standardisées dans le cadre du traitement des « données non structurées » (dans le sens, nous ne disposons pas initialement d'un tableau attribut-valeur, nous devons le construire justement). Il nous faut d'abord charger dans notre outil la collection d'images, procéder à des opérations de transformation pour en améliorer l'exploitabilité, en extraire les descripteurs (les ''features'') pour constituer le tableau de

données, appliquer enfin les méthodes de machine learning. Pour faire le parallèle avec le text mining (analyse statistique des données textuelles), un autre domaine typique de traitement de données non structurées, nous avons : charger les documents textuels, procéder à divers nettoyages (ex. harmoniser la casse, retirer les ponctuations), extraire le dictionnaire des termes (ex. mots), construire la matrice documents - termes propice au traitement statistique. Nous constatons que l'analogie est tout à fait fondée. Les compétences développées dans l'un des domaines est transposable à l'autre, la nature des données manipulées et les outils changent tout simplement.

Ce tutoriel a pour objet un problème de classement. On souhaite discerner automatiquement les photos contenant un véhicule de celles contenant tout autre type d'objet. La principale information est que, malgré des connaissances relativement succinctes en traitement d'images, j'ai pu mener à son terme l'étude avec une aisance qui en dit long sur l'utilisabilité du logiciel. Le plus difficile aura été d'identifier le composant le plus adapté à chaque étape, les tutoriels didactiques sont rares, en français n'en parlons même pas, il faut prendre un peu de temps pour lire attentivement la documentation.

2 Données ''Car Detection''

Nous utilisons la base « UIUC Image Database for Car Detection »[1]. Elle contient des photos de véhicules vues de profil (les observations « positives »), et des photos de tout autre objet (les « négatifs »). Les images sont toutes en niveau de gris, au format PGM (portable graymap file format). Le package initial comporte 1050 observations pour l'apprentissage (E1), et deux ensembles de test : (E2.a) 170 images de tailles diverses mais où les véhicules sont de la même échelle qu'en (E1) ; (E2.b) 108 images de tailles diverses, avec des véhicules à des échelles quelconques.

La généralisation sur les échantillons tests (E2) exige des traitements supplémentaires (ex. identifier la position de la voiture dans l'image, la mettre à la même échelle que l'échantillon d'apprentissage ou alors s'appuyer sur des descripteurs insensibles à l'échelle, etc.) qui

[1] S. Agarwal, A. Awan, D. Roth, « UIUC Image Database for Car Detection » ; https://cogcomp.cs.illinois.edu/Data/Car/.

dépassent largement le cadre d'un tutoriel d'initiation. J'ai donc décidé de partitionner au hasard (E1) pour construire et évaluer les modèles prédictifs que nous développerons.

Les images sont regroupées dans un dossier dédié. Les trois premières lettres des noms de fichiers permettent d'identifier la classe « positif » (pos) ou « négatif » (neg). Il faudra donc parser la chaîne de caractère correspondante pour créer la colonne « variable cible » nécessaire aux traitements.

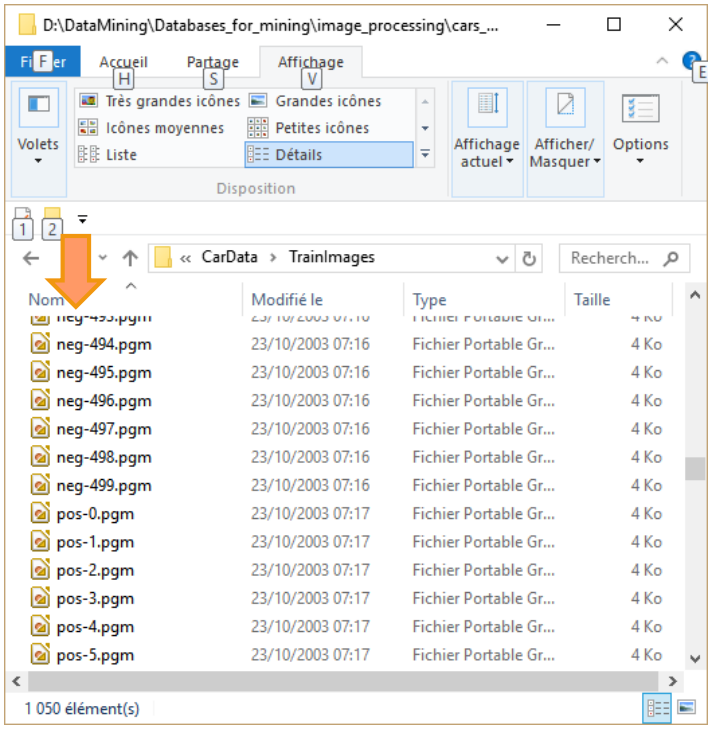
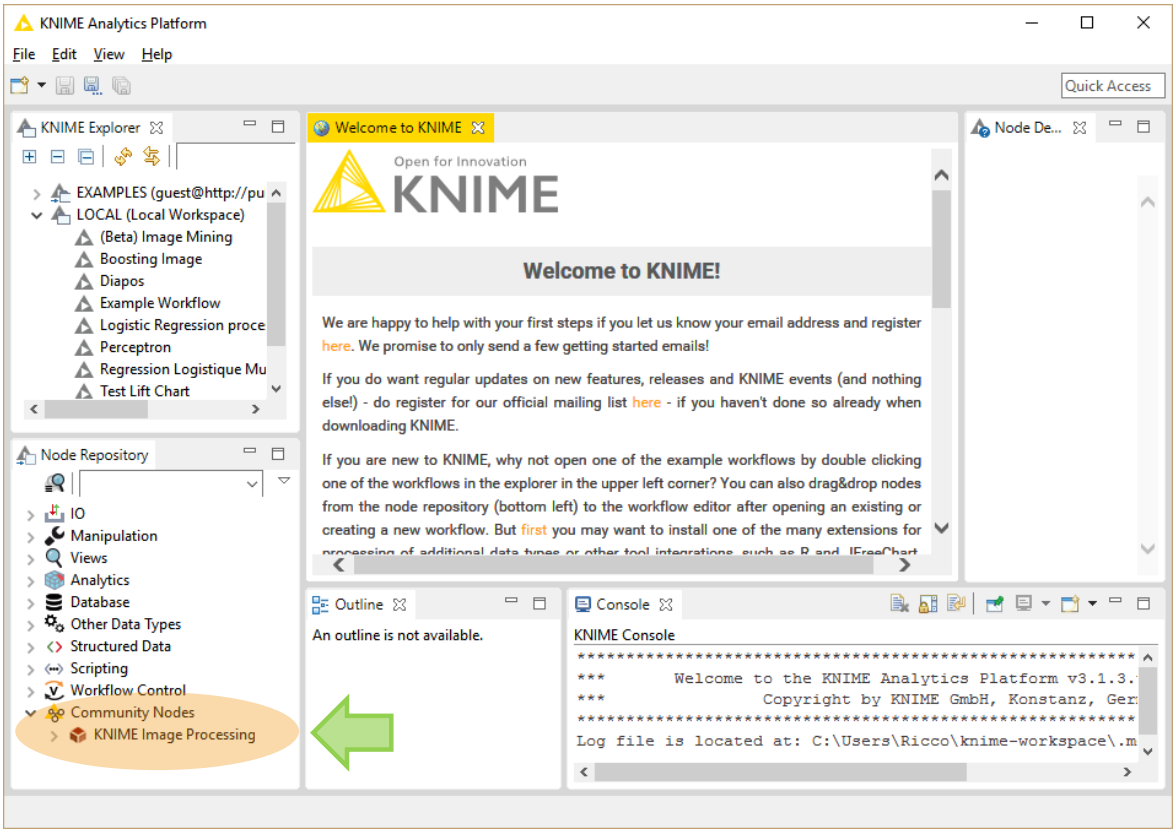


Figure 1 - Explorateur Windows - Fichiers images, les 3 premières lettres indiquent la classe

La première étape consiste à parcourir les éléments du dossier pour charger les images. Tout cela sans avoir à programmer. On commence fort déjà.

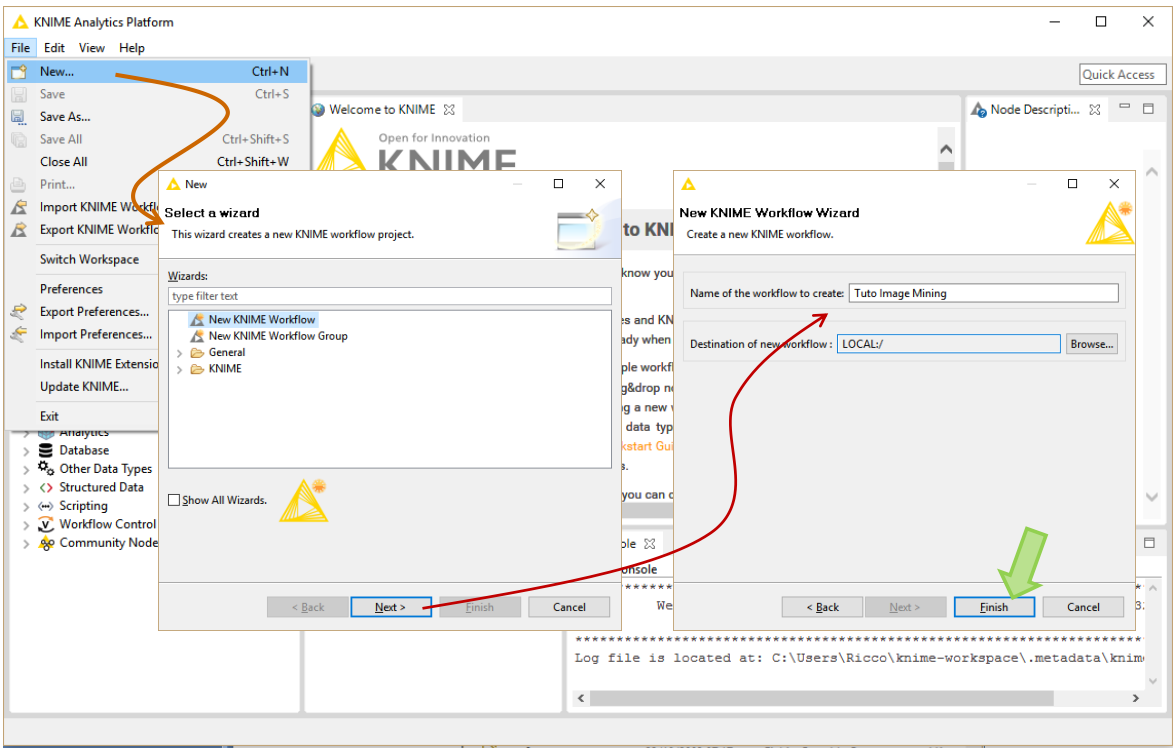
3 Classement automatique d'images avec Knime

Knime est un logiciel de data mining que j'étudie depuis un moment, le premier tutoriel où j'en parle date de 2008 ! La version « Analytics Platform » est libre, il faut le télécharger et l'installer. Il faut également ajouter le module « Knime Image Processing », il doit apparaître dans la branche « Community Nodes » dans la palette d'outils.

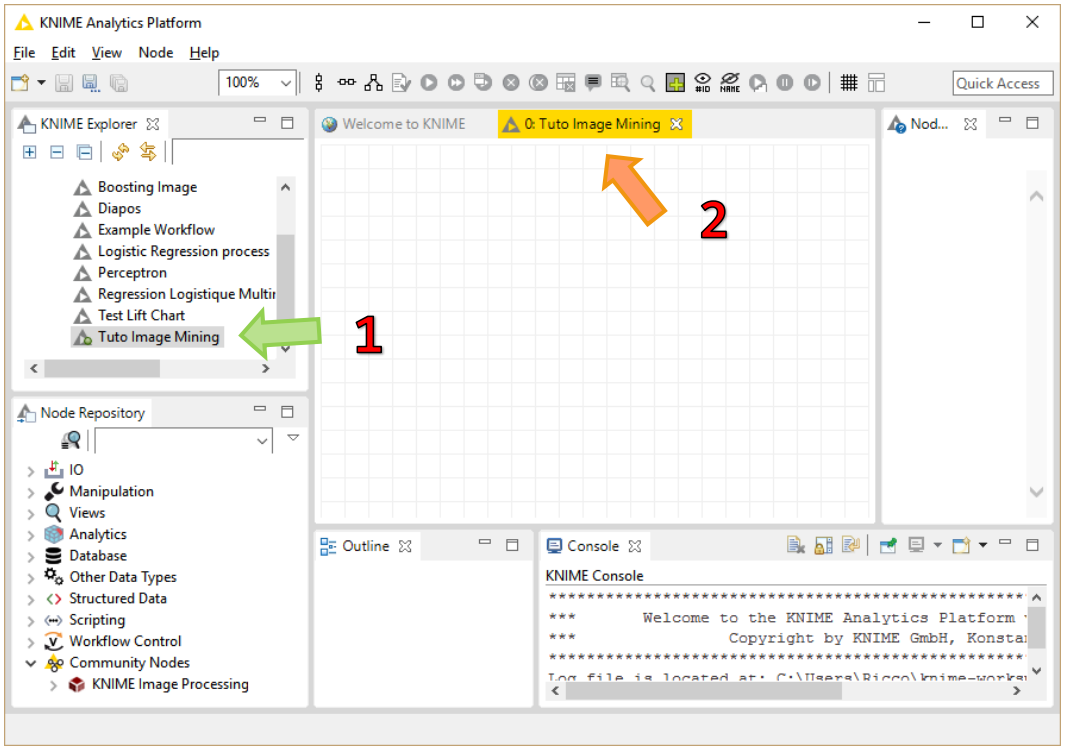


3.1 Création d'un « workflow »

Nous créons un nouveau « workflow » que nous nommons « Tuto Image Mining ».

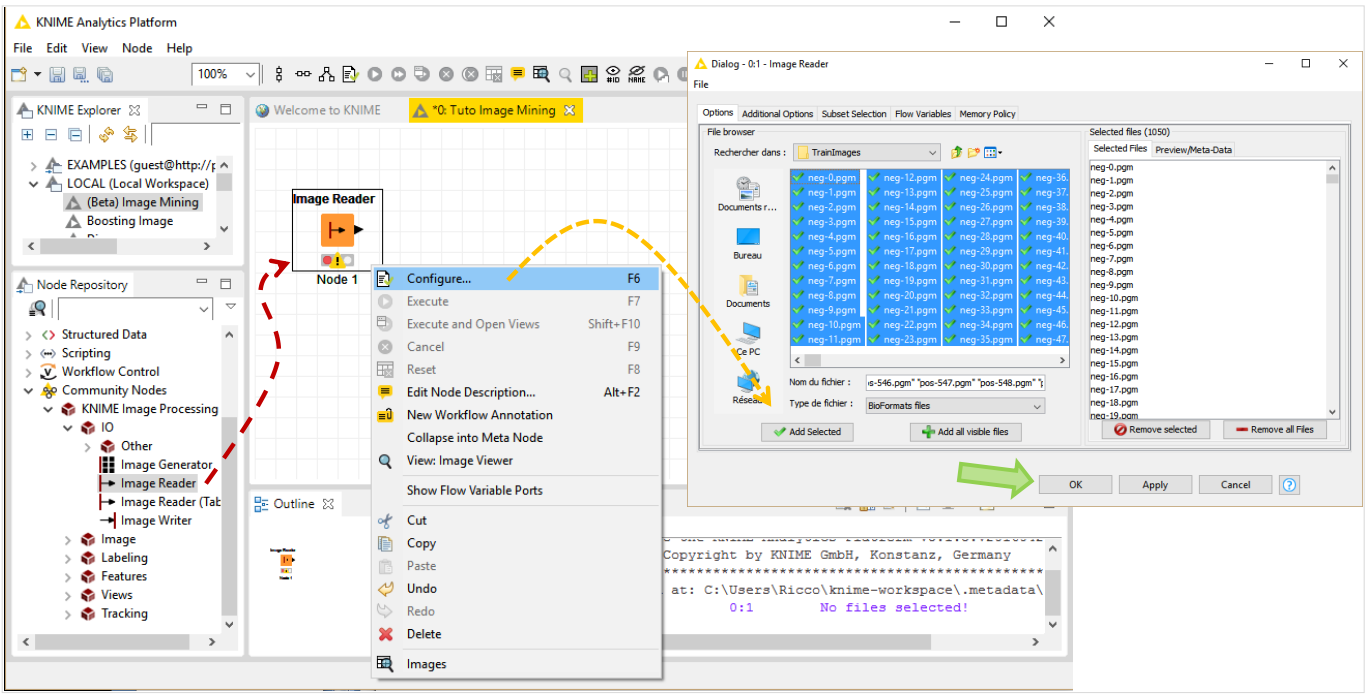


Le projet apparaît dans « Knime Explorer » (1) et l'espace de travail est ouvert pour recevoir les composants de traitements (2).



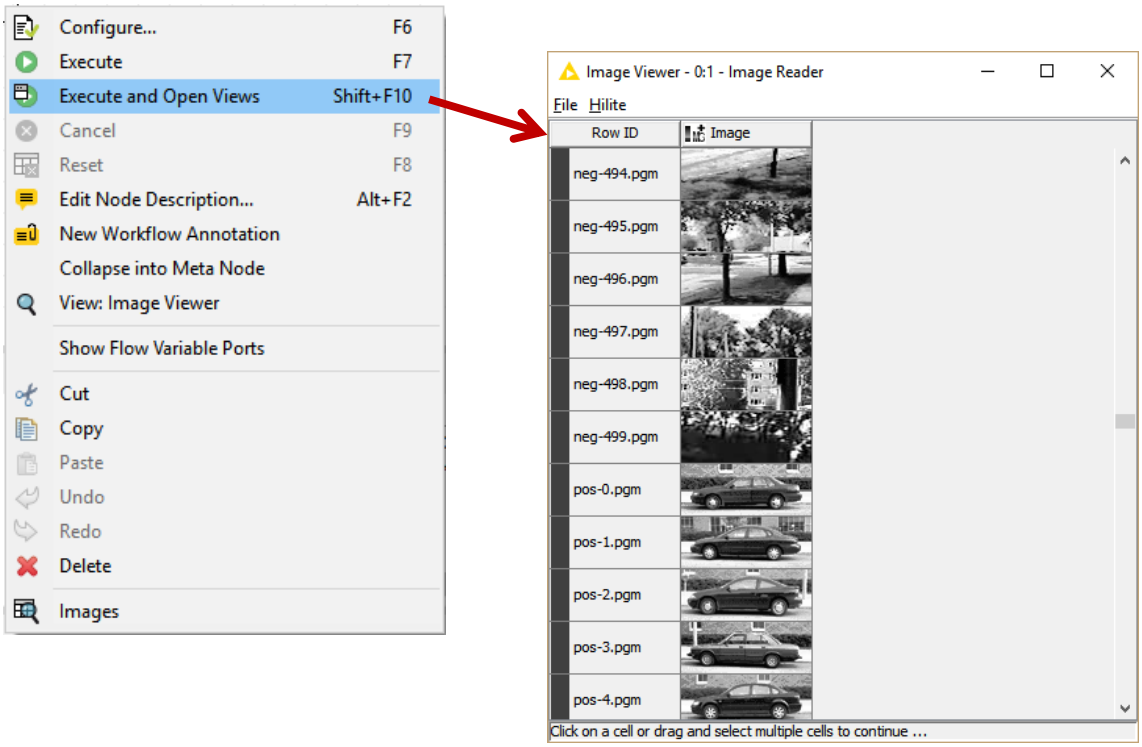
3.2 Chargement et visualisation des images

La première étape consiste à charger les fichiers images avec l'outil **IMAGE READER**.



A l'aide du menu Configure, nous sélectionnons les images à traiter (Add Selected) et nous validons en cliquant sur **OK**.

Pour visualiser les images, nous actionnons le menu contextuel Execute and Open Views d'IMAGE READER[2]. Elles apparaissent dans une fenêtre spécifique.



3.3 Extraction des descripteurs

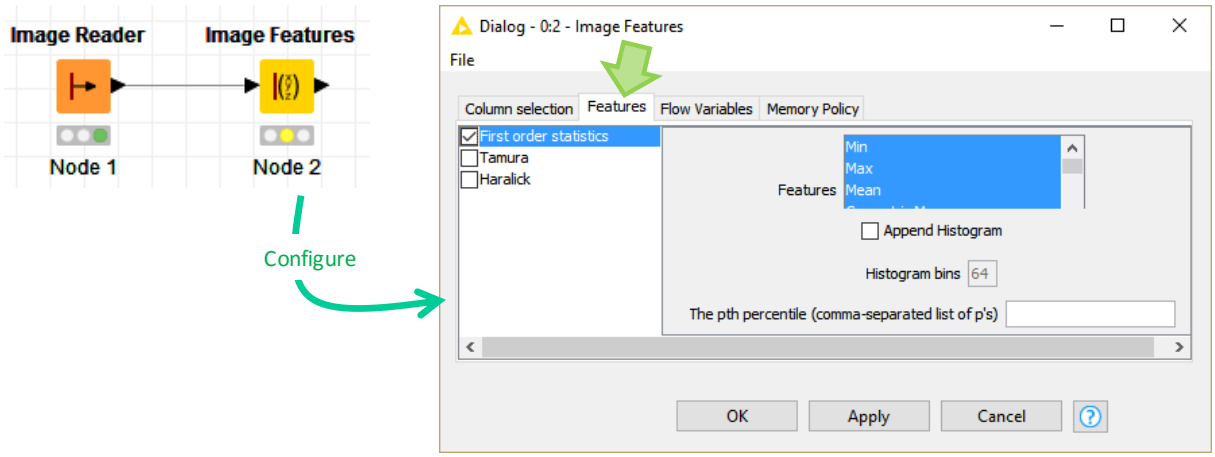
Il faut transformer chaque image en vecteur numérique pour pouvoir exploiter les algorithmes de machine learning. On parle de « feature extraction ». Les enjeux sont multiples : les descripteurs générés doivent être porteurs d'informations pertinentes, ils doivent être aussi succincts que possible pour éviter la malédiction de la dimensionnalité, on doit autant que possible limiter la redondance.

Remarque : Je parlais de démocratisation du traitement d'image, il faut quand même des compétences ne serait-ce que rudimentaires pour comprendre ce que l'on manipule et percevoir les bonnes pistes. Je ne m'attarderai pas ici puisque la finalité de ce tutoriel est de décrire la démarche globale sous Knime mais, à l'évidence, un certain investissement est

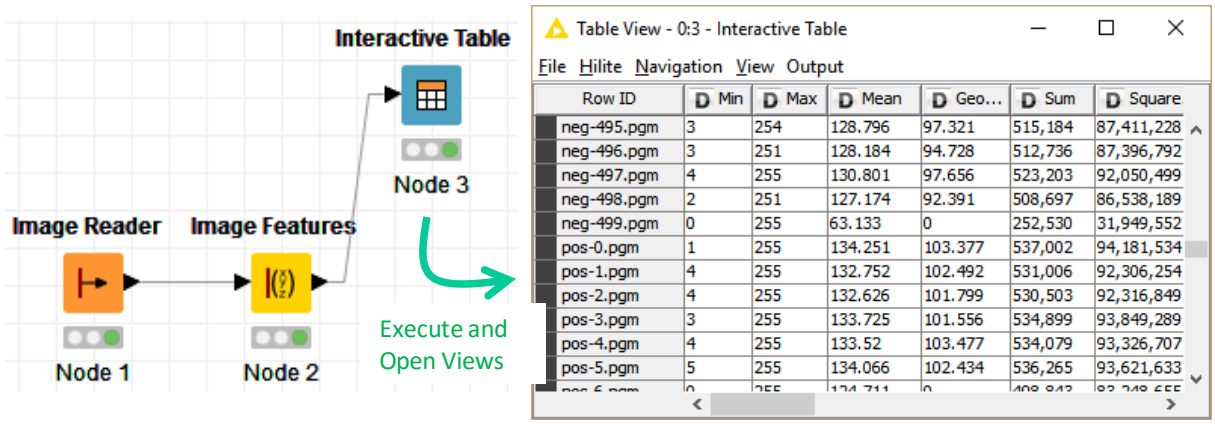
[2] Knime s'appuie sur l'API Bio-Formats (https://www.openmicroscopy.org/site/support/bio-formats5.1/).

indispensable pour pouvoir exploiter pleinement les outils. Par exemple, les roues semblent être un élément de discrimination intéressant. Dispose-t-on de paramètres mesurables dans une image pour arriver à les distinguer ? Sans connaissances du traitement d'images et des techniques courantes dans le domaine, il est difficile d'avancer de manière efficace.

Nous insérons le composant **IMAGE FEATURES** (Community Nodes / Knime Image Processing / Features). Voyons les paramètres qu'il nous propose (menu contextuel Configure)



Dans ''**Column Selection**'', nous indiquons traiter la colonne ''Image''. Il ne peut pas y avoir d'ambiguïtés là-dessus. Dans ''**Features**'', nous indiquons les descripteurs à extraire. Pour l'heure, nous nous en tenons à ''First Order Statistics'' qui correspondent à des statistiques calculées sur les niveaux de gris de l'image[3]. L'ensemble de données peut être visualisé avec **INTERACTIVE TABLE** (Views). Nous actionnons le menu Execute and Open Views.

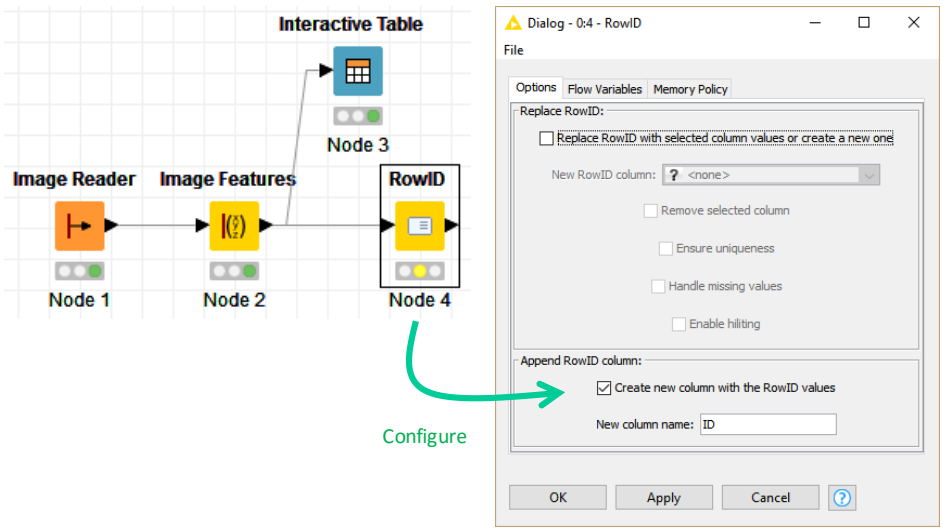


[3] Voir les nombreuses références sur le web, notamment : A. Materka, M. Strzelecki, « Texture Analysis Methods - A Review », Technical University of Lodz, 1998.

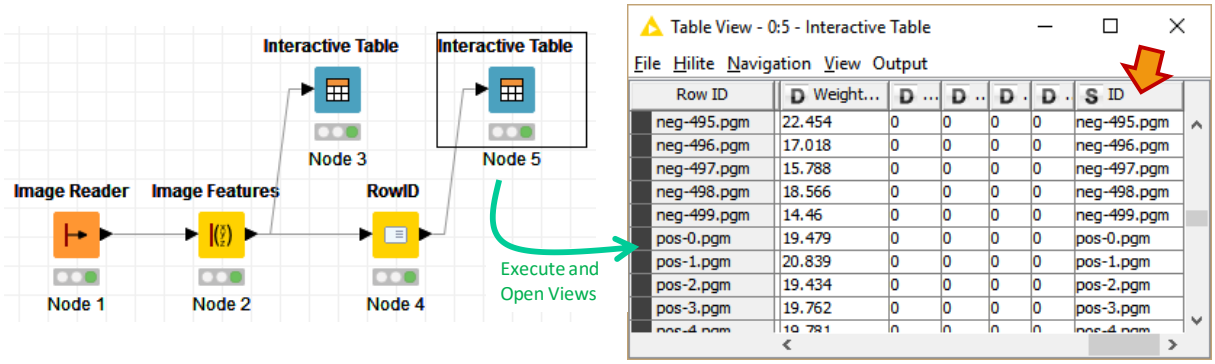
Nous disposons d'un tableau individus x variables propice au data mining avec, en ligne, les images, en colonne, les descripteurs issus du processus d'extraction des caractéristiques.

3.4 Création de la variable cible

Dans la grille ci-dessus, les noms de fichiers apparaissent comme identifiants des objets (**Row ID**). Nous devons l'exploiter pour constituer la variable cible, en extrayant les 3 premiers caractères. Nous utilisons l'outil **RowID** (dans Node Repository : branche Manipulation / Row / Other) que nous plaçons à la suite d'IMAGE FEATURES. Nous le paramétrons (menu Configure) de manière à créer une nouvelle colonne **ID** avec les noms d'identifiants.

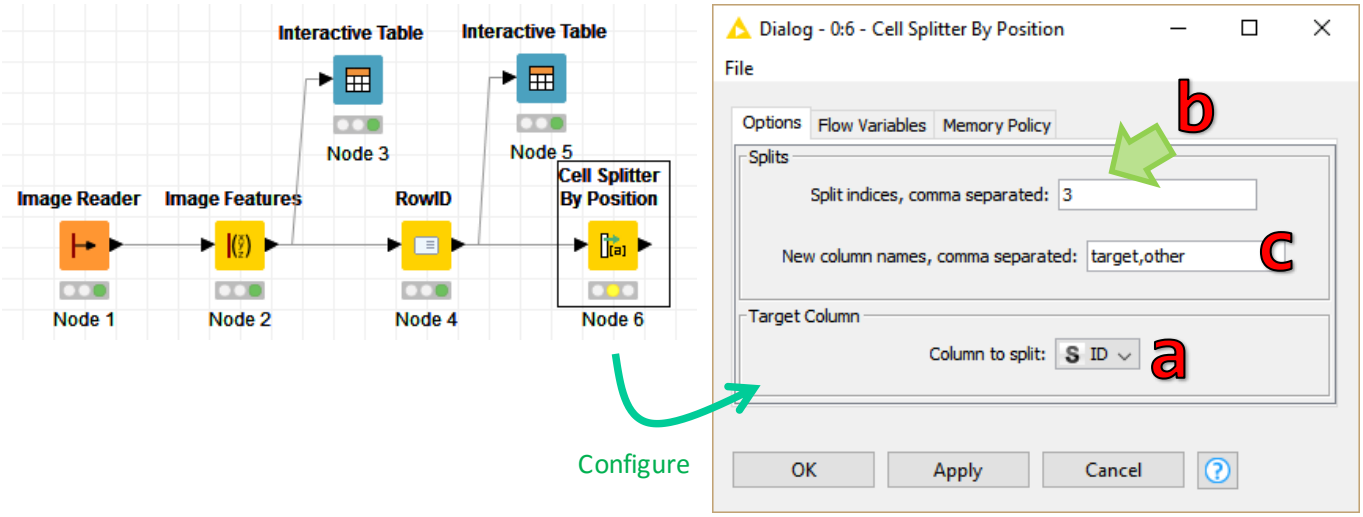


Nous exécutons (menu Execute) et nous ajoutons le composant **INTERACTIVE TABLE** (Views) pour examiner les résultats.



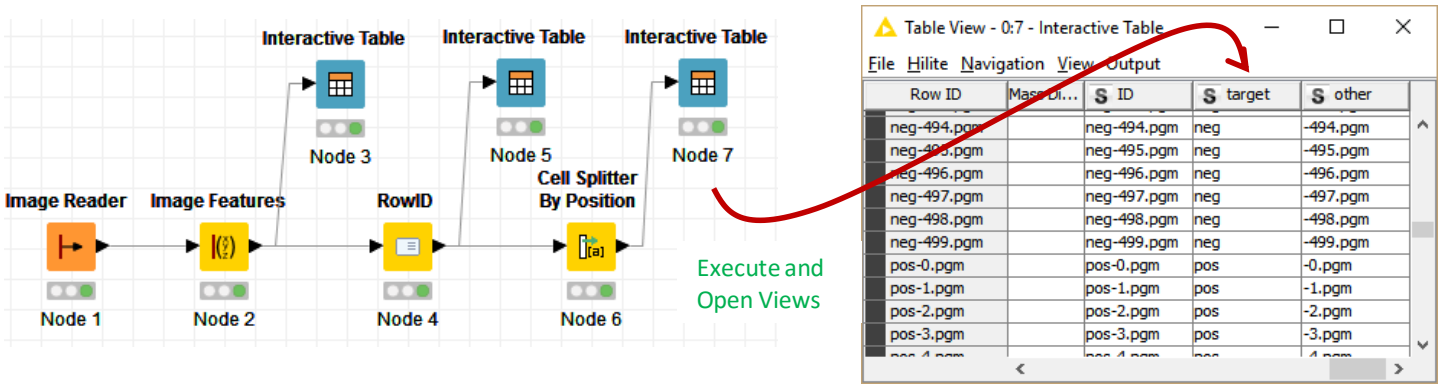
La colonne ID est composée de chaînes de caractères (S). Nous allons la décomposer de manière à extraire les 3 premiers caractères et créer une nouvelle colonne qui jouera le rôle de variable cible.

Nous insérons le composant **CELL SPLITTER BY POSITION** (Manipulation / Column / Split & Combine). Nous le paramétrons comme suit :



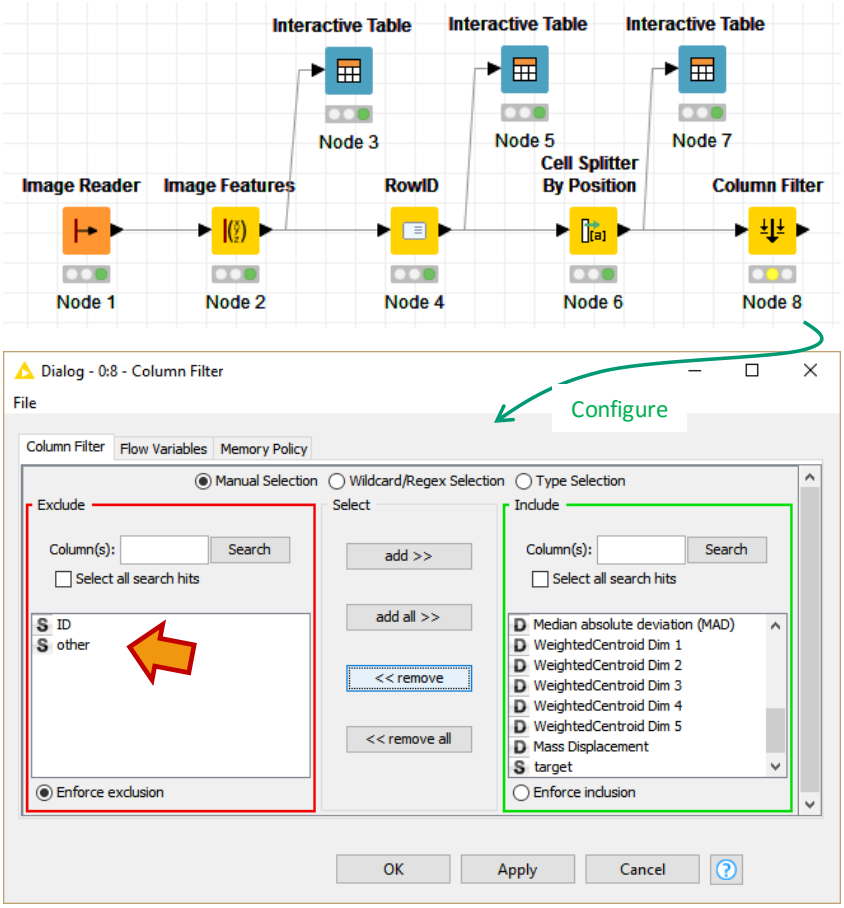
Nous traitons la colonne ID (a), nous découpons à partir du 3ème caractère (b), deux colonnes ''target'' et ''other'' (c) découlent de l'opération.

Vérifions cela avec **INTERACTIVE TABLE**. La colonne ''target'' est maintenant exploitable pour l'analyse prédictive.

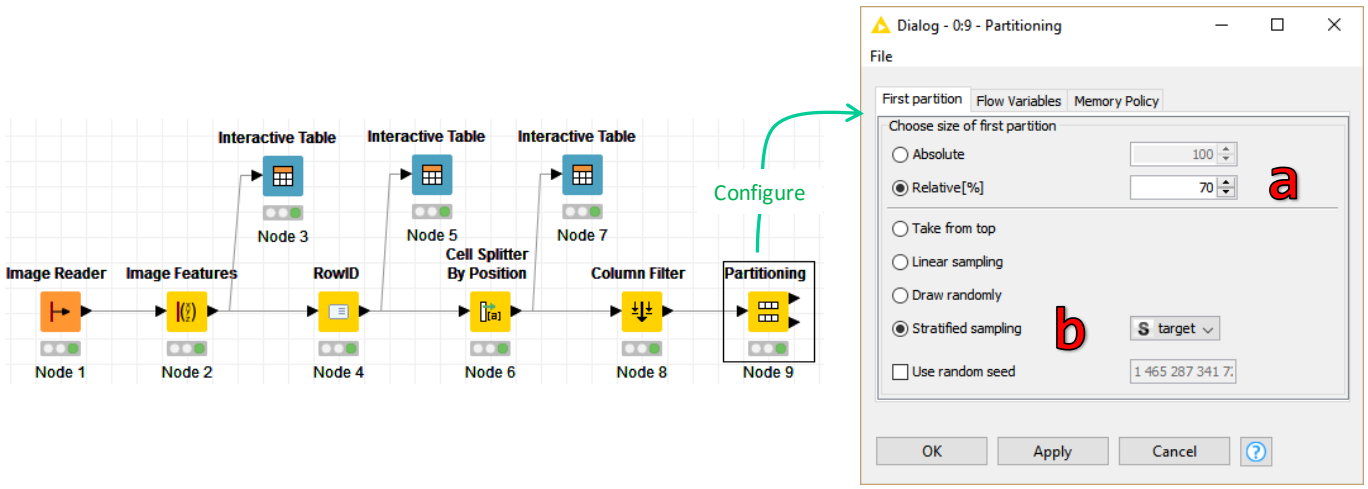


3.5 Construction et évaluation du modèle prédictif

Retrait des colonnes inutilisées. Notre objectif est de prédire au mieux « target » en fonction des descripteurs extraits des images. Il nous faut évacuer de la table de données les colonnes inutilisées pour les traitements (« ID » et « other ») à l'aide du composant **COLUMN FILTER** (Manipulation / Column / Filter).

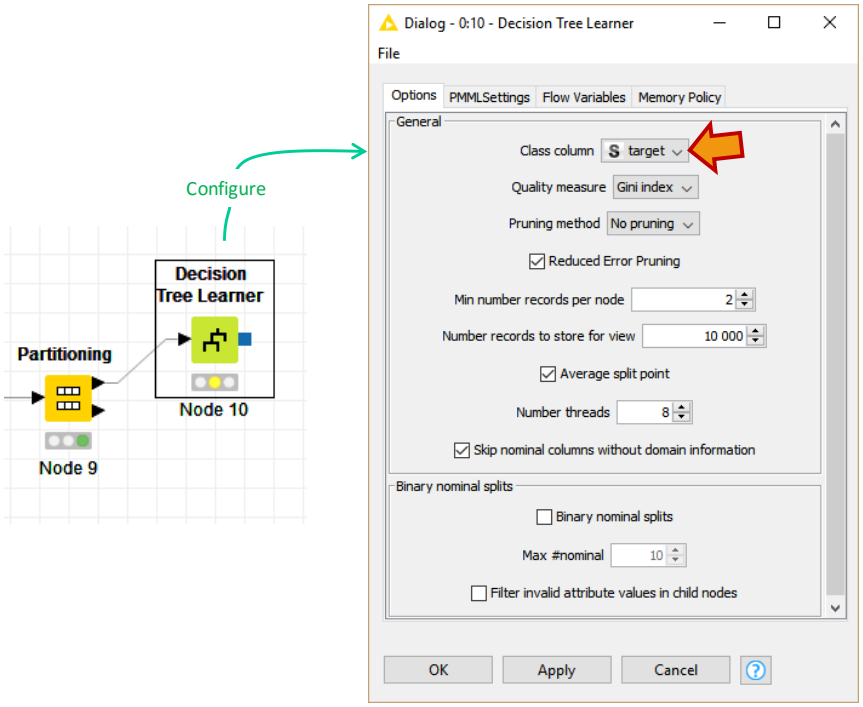


Partition des données en échantillons d'apprentissage et test. Pour obtenir une évaluation honnête des capacités prédictives, il est conseillé de partitionner les données en échantillons d'apprentissage et de test, le premier pour élaborer les modèles, le second pour en mesurer les performances. Nous insérons le composant **PARTITIONING** (Manipulation / Row / Transfrom) dans le workflow.

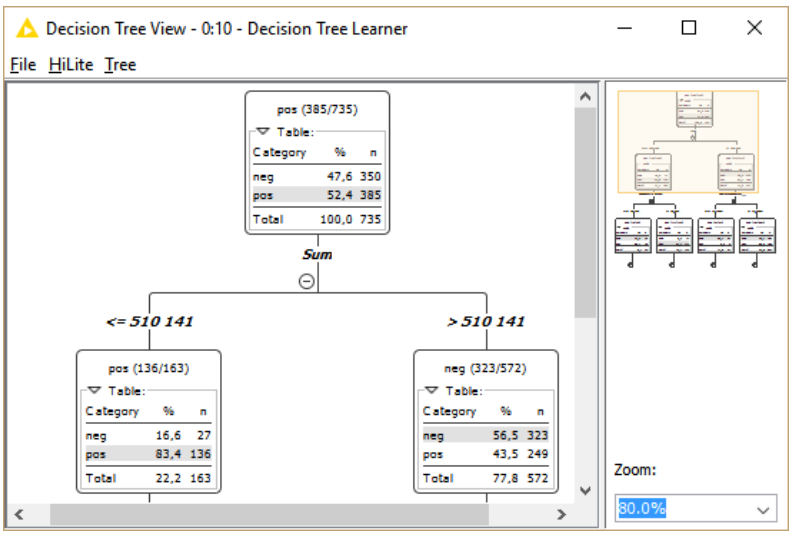


Nous réservons 70% des observations pour l'apprentissage (a) ; nous effectuons un tirage stratifié sur la cible « target » pour que les proportions des positifs et négatifs soient strictement respectées dans les deux échantillons.

Construction du classifieur. Nous souhaitons utiliser un arbre de décision pour prédire le type d'image (nous reviendrons dessus dans la section 4.1). Nous insérons le composant **DECISION TREE LEARNER** (Analytics / Mining / Decision Tree).



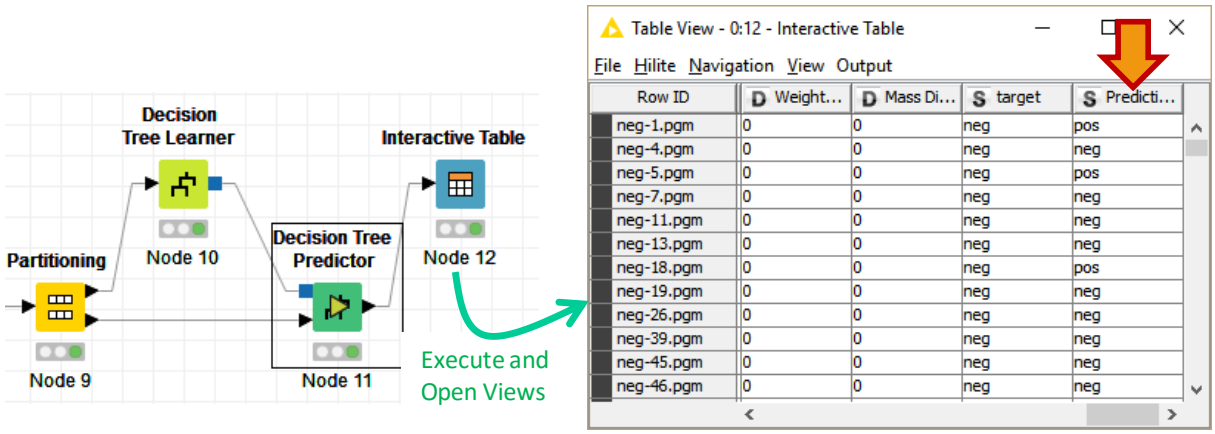
Nous veillons à ce que la variable cible Class column soit bien « target ». Les autres paramètres sont laissés tels quels.



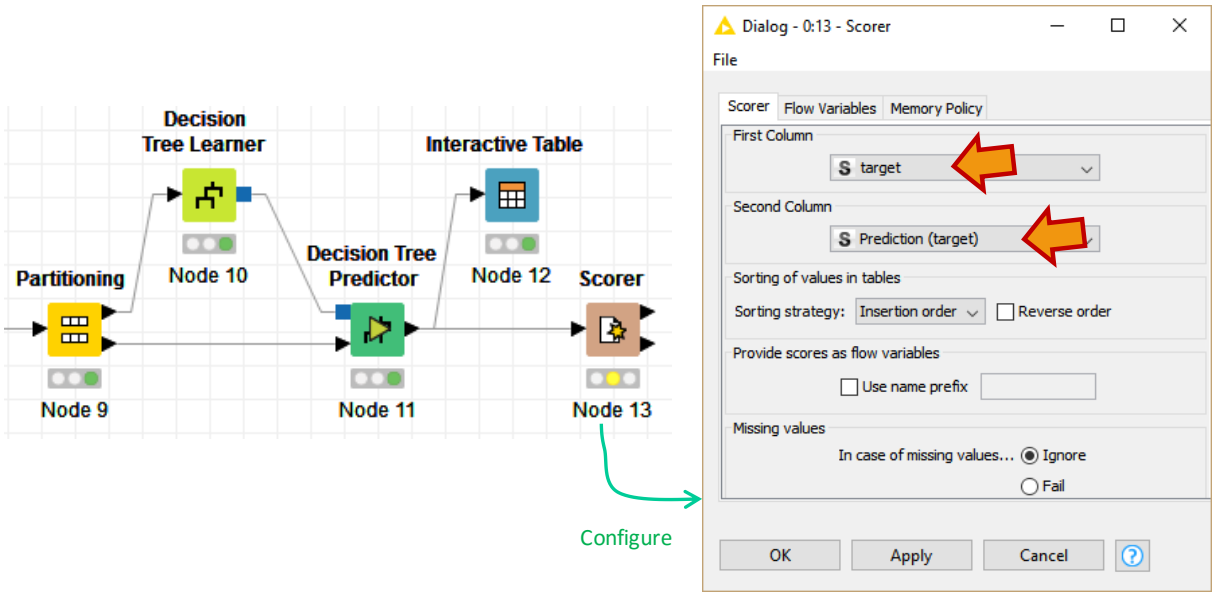
Nous visualisons les résultats en cliquant sur le menu contextuel Execute and Open Views. L'arbre en lui-même est anecdotique dans notre contexte, nous ne le commenterons pas.

Evaluation du classifieur. Pour évaluer les performances, nous devons appliquer le modèle sur l'échantillon test puis confronter les prédictions avec les classes observées.

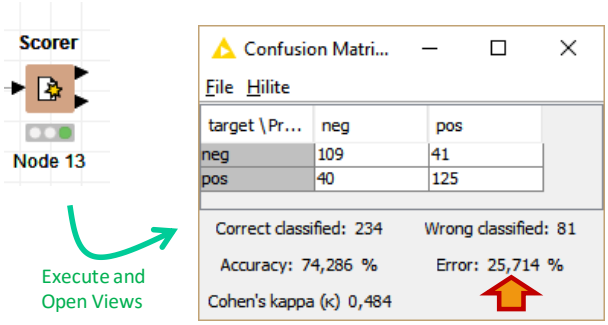
Nous insérons le composant **DECISION TREE PREDICTOR** (Analytics / Mining / Decision Tree) pour la prédiction. Il prend en entrée le modèle et la fraction « test » des données. Nous rajoutons l'outil INTERACTIVE TABLE pour vérifier la bonne tenue de l'opération.



La colonne « Prediction(target) » a bien été générée. Il s'agit de la croiser avec « target » avec l'outil **SCORER** (Analytics / Mining / Scoring).

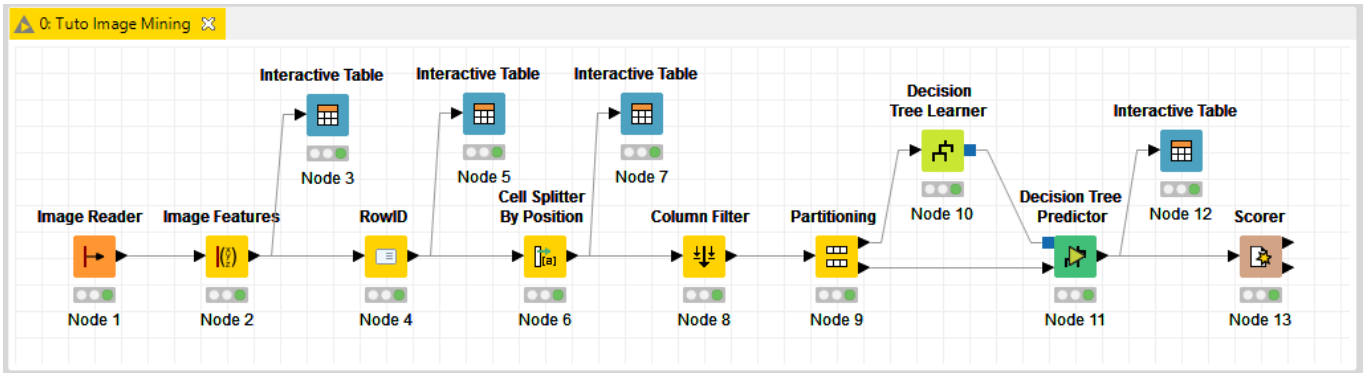


Il ne reste plus qu'à lancer le traitement avec le menu Execute and Open Views. Nous obtenons la matrice de confusion suivante avec un taux d'erreur de **25.714%**.



3.6 Bilan

Voici le workflow dans la totalité.



Un taux d'erreur de 25.714 % (un objet mal classé sur 4) semble perfectible. Nous verrons comment améliorer cela dans la section suivante. Le plus important ici est que nous avons pu mener l'analyse complète sans avoir à aligner une seule ligne de code. Réaliser la même chose sous R ou Python nous obligerait à déterminer tout d'abord les bonnes bibliothèques, puis à identifier les commandes adéquates. Certes Google est là pour nous aider, mais arriver à reconnaître les informations qui nous permettent d'avancer n'est pas toujours très facile. En ce qui me concerne, je passe un temps incalculable à scruter les échanges sur Stackoverflow pour voir ce qui me pourrait être utile dans mes enseignements. Dans un contexte d'obligation de productivité avec des contraintes fortes de délais, c'est un luxe dont on ne dispose pas toujours.

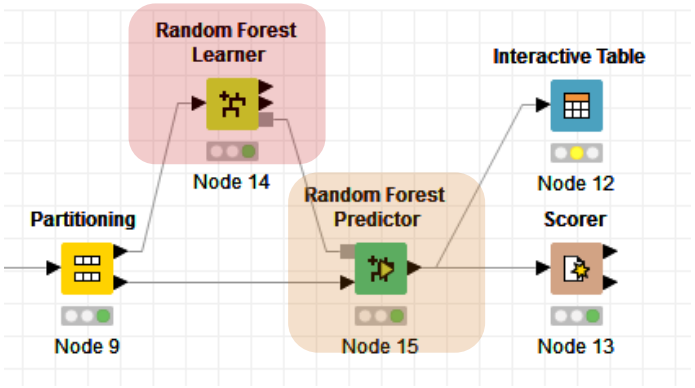
4 Pistes d'amélioration

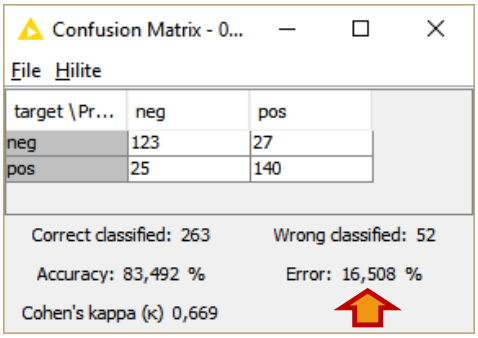
La trame y est, mais pas les résultats. Dans cette section, nous explorons les pistes « simples » d'améliorations des performances.

4.1 Choix de l'algorithme d'apprentissage

La solution la plus immédiate consiste à essayer d'autres algorithmes de machine learning. Ils sont plus ou moins performants selon les données et types de problèmes. Pour les comparer, nous mesurons le taux d'erreur en test.

Il paraît que les Random Forests ont de bonnes capacités prédictives[4]. Dans la dernière partie du workflow, nous remplaçons les composants DECISION TREE par **RANDOM FOREST** (LEARNER et PREDICTOR). Dans le LEARNER, nous veillons à ce que « target » soit bien désignée comme variable cible (Target column).



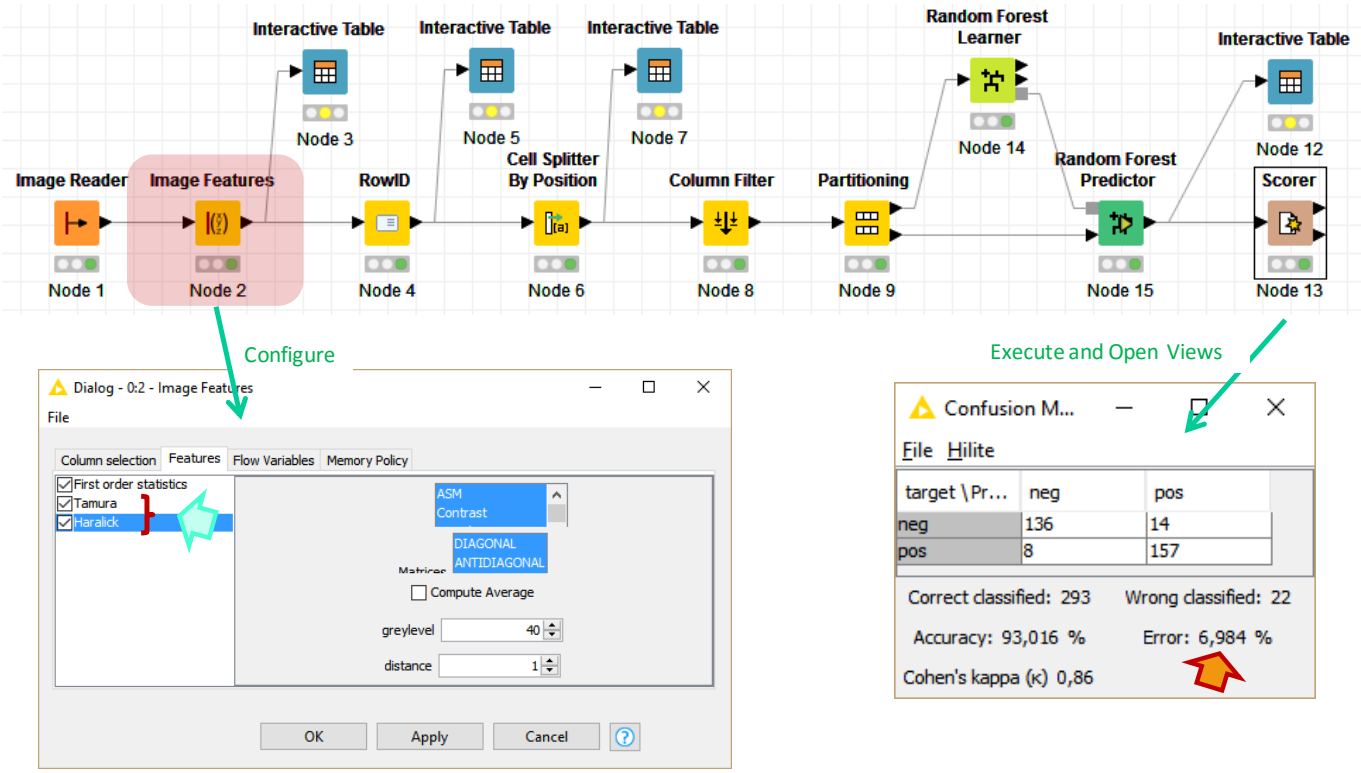


Le taux d'erreur passe à **16.508%**. L'amélioration est spectaculaire. On sait que la méthode est souvent performante. Elle le montre encore une fois ici. De plus, dans un espace de représentation où un grand nombre de descripteurs - dont la pertinence n'est pas toujours avérée - sont susceptibles d'être générées, sa capacité à résister au sur apprentissage est une propriété précieuse.

4.2 Extraction d'autres descripteurs

Parlons-en des descripteurs justement. Dans **IMAGE FEATURES**, Nous nous sommes restreints aux moments statistiques calculés sur les niveaux de gris. D'autres options étaient possibles. Nous les activons même si nous ne cernons pas trop les principes qui les sous-tendent (« Tamura » et « Haralick »). Bien évidemment, les calculs seront plus lents puisque des données supplémentaires sont produites et traitées par les algorithmes en aval.

[4] R. Rakotomalala, « Bagging, Random Forest, Boosting - Diapos », novembre 2015.

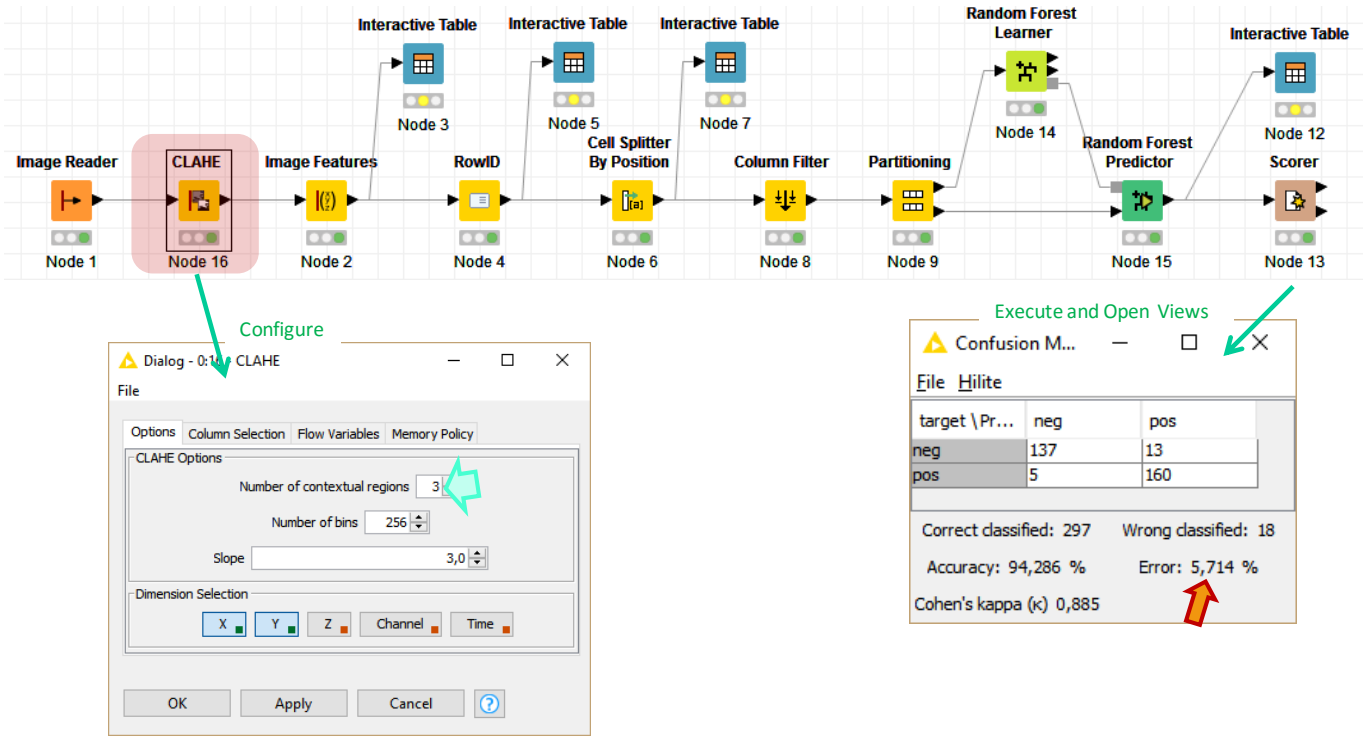


Le taux d'erreur est passé à **6.984%**. L'amélioration est tout aussi substantielle que précédemment. Dans le contexte d'une étude réelle, cela vaut le coup de se pencher sur les particularités de ces nouveaux descripteurs et de ce qu'elles amènent dans la modélisation prédictive. Se documenter sur les techniques d'extraction de caractéristiques est indispensable quand on souhaite aller plus loin.

4.3 Modifier les propriétés des images

Mais nous pouvons encore remonter plus en amont et nous intéresser aux propriétés des images elles-mêmes. Le traitement d'images nous offre un grand nombre d'opérateurs de transformation qui permettent d'améliorer leur qualité, en travaillant sur la luminosité, le contraste, en filtrant le bruit, en passant les images couleurs en niveaux de gris, etc.

Ici également, des compétences avancées sont nécessaires si l'on souhaite aller plus loin. Dans notre exemple, nous nous contentons d'améliorer les contrastes (ça semble de bon sens) à l'aide de l'outil **CLAHE** de Knime (Community Nodes / Knime Image Processing / Image / Process). Voici le nouveau diagramme avec le paramétrage de CLAHE.

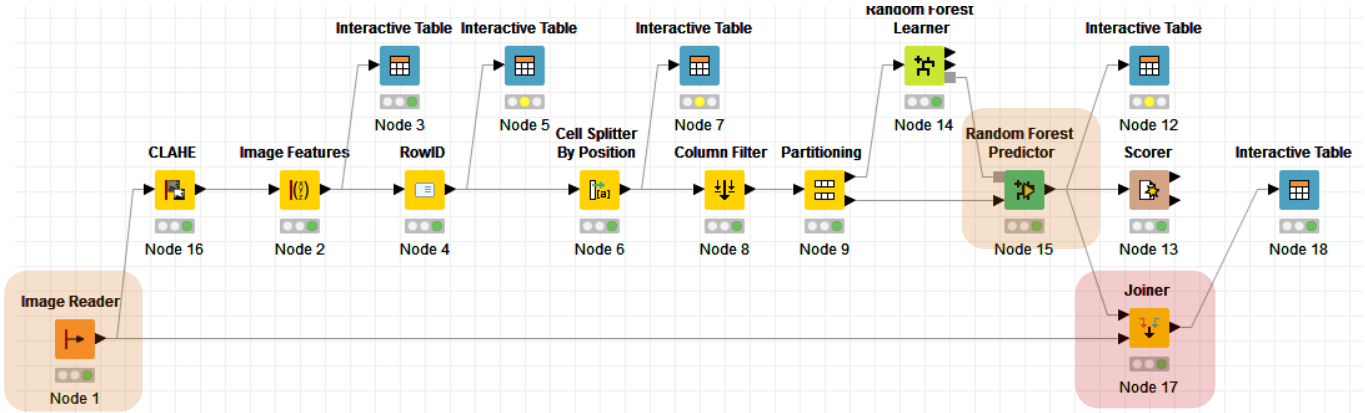


Le taux d'erreur passe à **5.714%**. Le gain est minime, 4 individus mal classés en moins. Nous constatons surtout que ce type de transformation peut peser sur la prédiction.

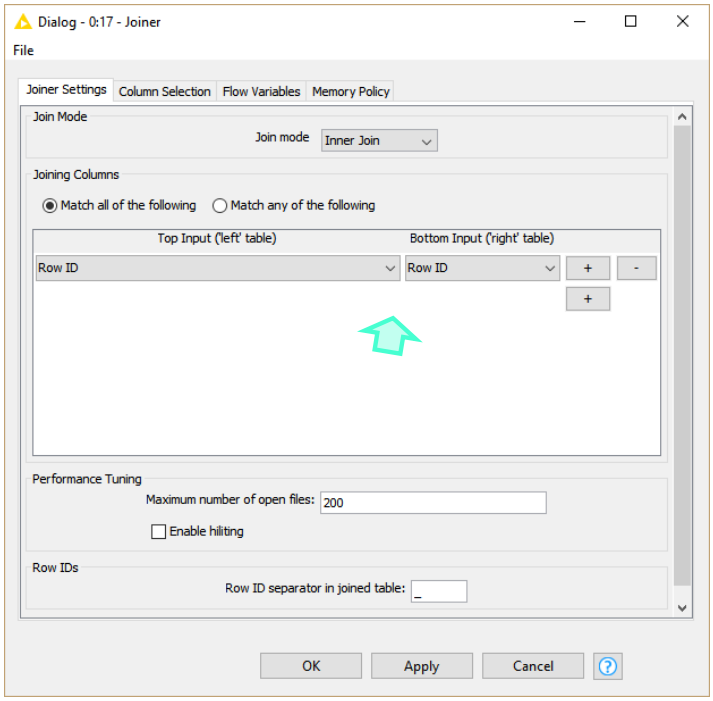
4.4 Visualiser les images mal classées

Rien de tel que de revenir aux données pour comprendre les carences d'un système de classement. Nous souhaitons visualiser les images mal classées sur l'échantillon test. Il peut y avoir des problèmes d'étiquetages, du bruit particulièrement pénalisant, ou encore des singularités qui ont échappé à notre sagacité, etc.

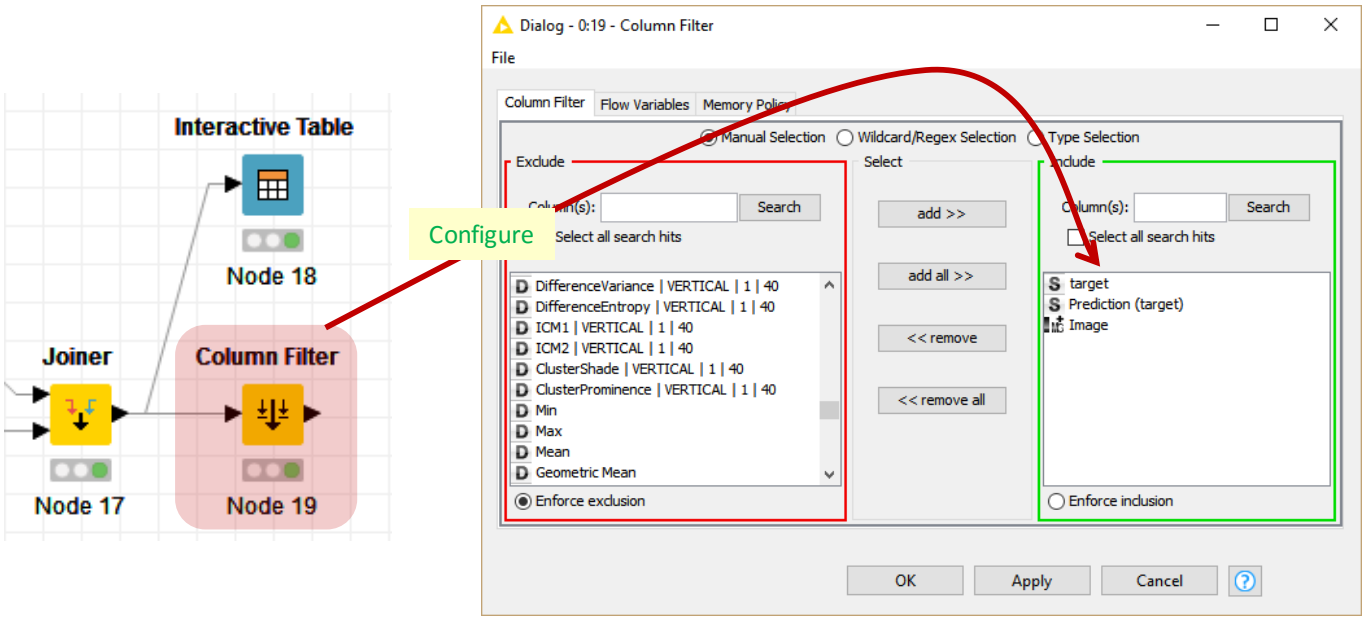
Pour cela, nous devons tout d'abord joindre les images à cet échantillon avec l'outil **JOINER** (Manipulation / Column / Split & Combine).



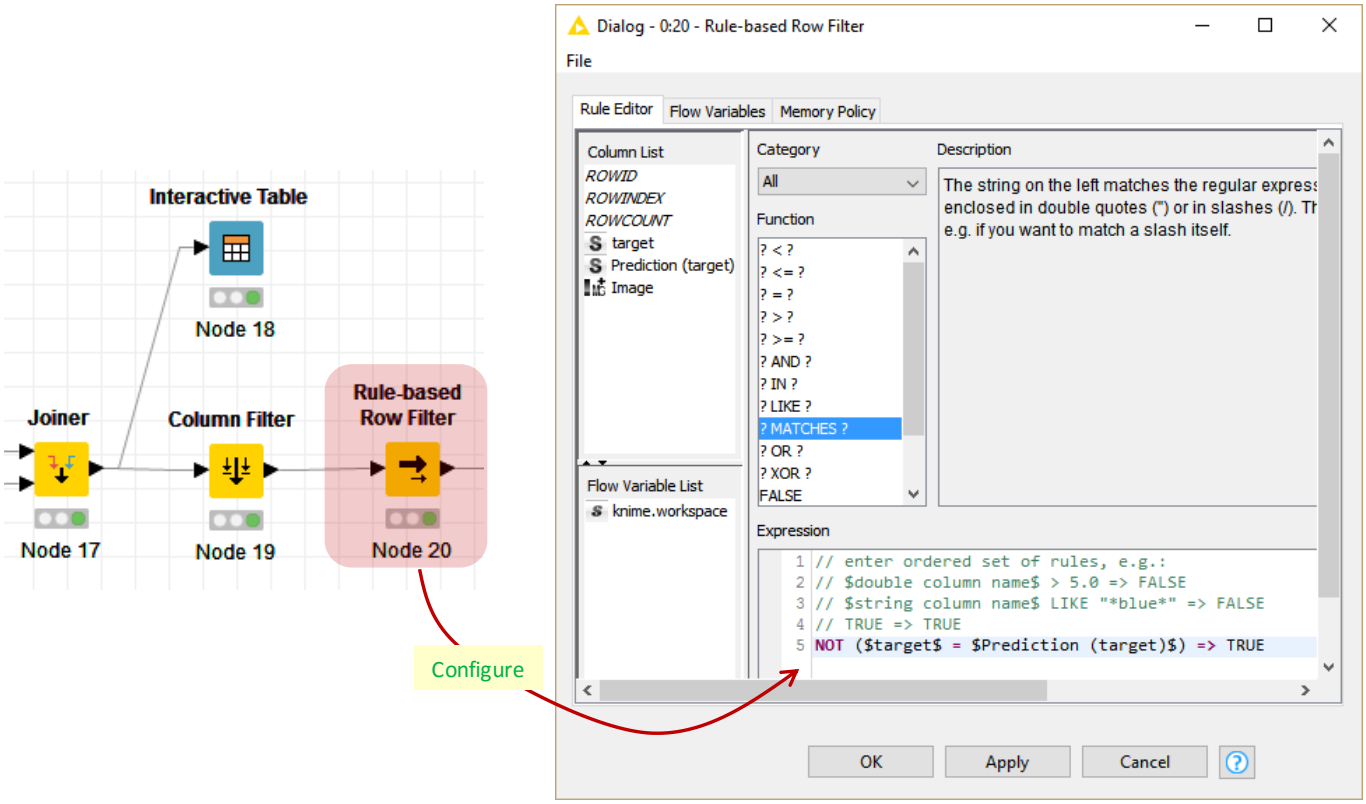
Dans la fenêtre de paramétrage (menu Configure), nous spécifions que la jointure est basée sur l'identifiant des observations **Row ID**.



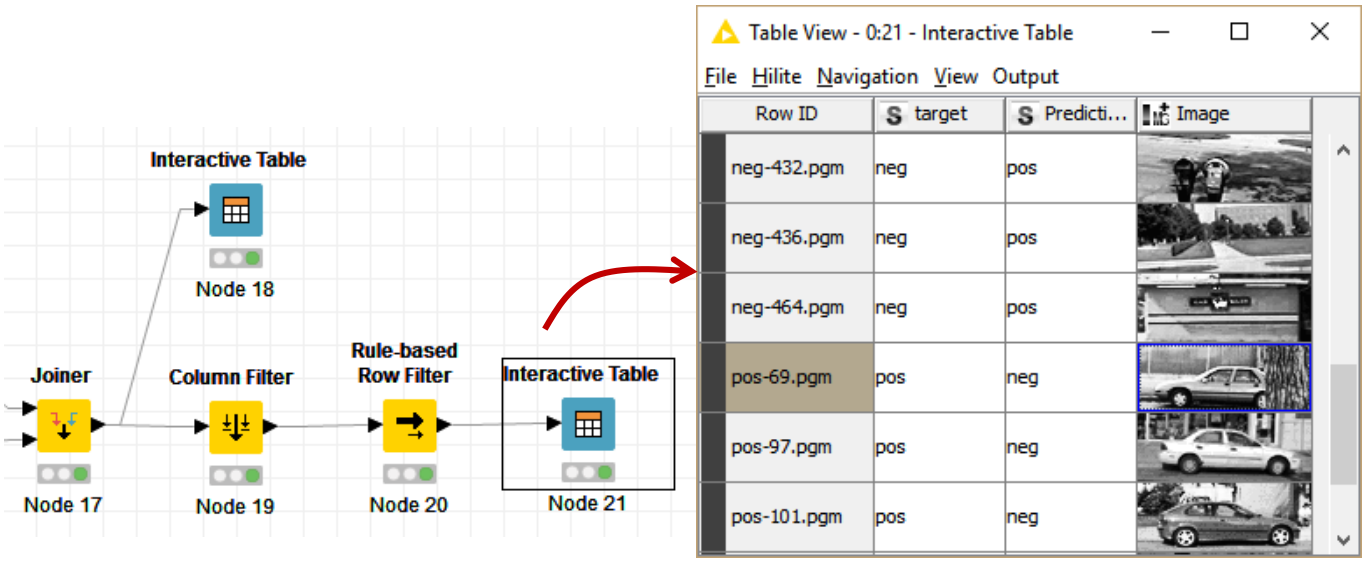
Nous ne conservons que les colonnes « image », « target » et « prediction(target) » avec l'outil **COLUMN FILTER**.



Enfin, nous filtrons les observations de manière à ne conserver que les individus mal classés avec le composant **RULE-BASED ROW FILTER** (Manipulation / Row / Filter).

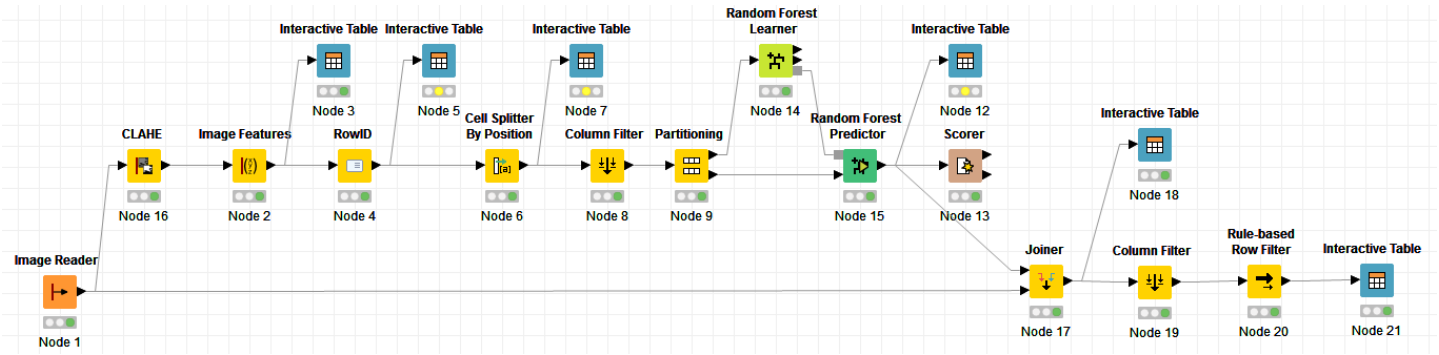


Nous nous intéressons aux individus pour lesquels la classe prédite ne correspond pas à la classe observée. Il ne reste qu'à les visualiser avec INTERACTIVE TABLE.



L'objet n°69 (pos-69.pgm) par exemple est un véhicule [target = pos] qui n'a pas été identifié comme tel [prediction(target) = neg]. On comprend pourquoi, sa partie arrière est masquée. En procédant ainsi, l'analyste a tout le loisir d'explorer les différentes situations et, éventuellement, de proposer des solutions qui améliorent les performances.

Voici de nouveau le workflow dans sa globalité, il fait son petit effet :



5 Conclusion

Ce tutoriel a pour premier objectif de présenter dans ses grandes lignes un exemple type d'image mining (fouille d'images), via un problème d'analyse prédictive. On se rend compte que la démarche s'inscrit parfaitement dans la trame de l'analyse des données non structurées. C'est un espace que je n'avais pas beaucoup exploré jusqu'à présent, le domaine était surtout l'apanage des spécialistes du traitement d'images. Dans le contextuel actuel où le data scientist se doit de savoir travailler sur des données de sources et de formats divers, cette compétence devient nécessaire aux statisticiens et autres data miner, voire indispensable. De fait, je demande de plus en plus à mes étudiants de travailler dessus dorénavant.

Le second objectif était de montrer que des outils performants sont à notre disposition, nous facilitant grandement la vie. J'aurais pu faire tous les traitements sous Python (ce n'est pas exclu que j'écrive un tutoriel dessus un jour), mais il m'a semblé plus symbolique de les réaliser sous Knime où toutes les opérations ont été menées sans avoir à écrire aucune ligne de code programme.

6 Références

Knime Image Processing, https://tech.knime.org/community/image-processing

S. Agarwal, A. Awan, D. Roth, « UIUC Image Database for Car Detection » ; https://cogcomp.cs.illinois.edu/Data/Car/