1 Topic

Image classification using Knime.

The aim of image mining is to extract valuable knowledge from image data. In the context of supervised image classification, we want to assign automatically a label to image from their visual content. The whole process is identical to the standard data mining process. We learn a classifier from a set of classified images. Then, we can apply the classifier to a new image in order to predict its class membership. The particularity is that we must extract a vector of numerical features from the image before to launch the machine learning algorithm, and before to apply the classifier in the deployment phase.

The subject is not really new. But its democratization is more recent. I see two main reasons. First, the abundance of images with the web data makes necessary this skill to statisticians and data miner. We note for instance that image processing is increasingly present in the challenges. Second, there are more and more easy to use tools for data miner. They greatly facilitate our task. Formerly, a good dose of computer programming ability was needed for handling this type of data. Today, efficient tools allow us to perform the analysis without being a specialist of image processing. Some packages are also available for high level programming languages such as Python (e.g. <u>scikit-image</u>). I took advantage to these properties in recent years for my teachings. The power of the tools allows me to go to the essentials without having to spend hours to explain in detail the structures of low level of images. Even if these knowledges may be important when it becomes necessary to finely adjust the parameters of our analysis.

"Knime Image Processing" module is quite symbolic of this evolution. It is not even necessary to learn programming language. We can complete an analysis without writing a single line of source code. The most important is to have a global vision of the basic outline of the study. We simply define the sequence of treatments in order to obtain results that are relevant. The entire process can be summed up in several key stages. We must load our collection of images. We may transform the image characteristics to improve its properties. We extract the descriptors ("features") to build the data table (attribute-value table), on which we can apply the machine learning algorithms. To make the parallel with text mining, another typical domain of unstructured data processing, the main steps are: load the collection of texts, perform various clean-ups (e.g. remove stop words, remove punctuations, etc.), extract the dictionary of terms (features), construct the term-document matrix, on which we can launch the data mining algorithms. We note that the analogy is relevant. The skills developed in one of the domains is transposable to the other. Merely, the nature of data - and thus the tools used to handle them - is modified.

We deal with an image classification task in this tutorial. The goal is to detect automatically the images which contain a car. The main result is that, even if I have a basic knowledge about the image processing, I can lead the analysis with a facility which is symptomatic of the <u>usability</u> of Knime in this context.

2 "Car Detection" dataset

The "<u>UIUC Image Database for Car Detection"</u> contains images of side views of cars for use in evaluation object detection algorithms. The images with cars are positive instances. The others are the negative instances. The images are all grey-scale and are available in raw <u>PGM</u> format. The initial package contains: (E1) 1050 training images; (E2.a) 170 single-scale test images i.e. the images are of different sizes themselves but contain cars of approximately the same scale as in the training images; (E2.b) 108 multi-scale test images i.e. the images are of different sizes and contain cars of various scales.

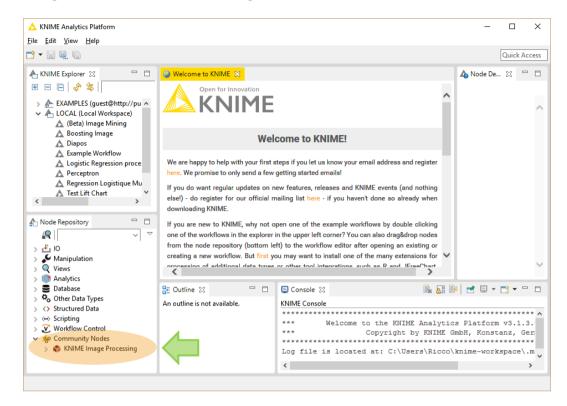
Handling the test samples (E₂) requires additional processing (e.g. identifying the position of the car in the image, put the car on the same scale as the learning sample, or use insensitive to scale descriptors, etc.) that go beyond the framework of a basic tutorial. So I decided to partition randomly (E₁) to build and evaluate predictive models that we will develop.

Images are grouped in a specific folder. The first three letters of the filenames allow to identify the class membership (pos: positive instances; neg: negative instances). Therefore, we must parse the filename to create the target column used in the learning process.

FiFer Accueil Pa	rtage Affichage		^ (
Très grandes i	cônes 📧 Grandes icônes		2 33
/olets	nes 👫 Petites icônes	- A66i alta ma A66i	cher/ Options
Liste	EE Détails		quer • •
	Disposition		
1 2 -			
← × ↑	CarData → TrainImages	5 V	Recherch 🔎
Nom	Modifié le	Туре	Taille
🔟 неў-чээ.руш	23/ 10/ 2003 01.10	richier ronable of	4 NU
卤 neg-494.pgm	23/10/2003 07:16	Fichier Portable Gr	4 Ko
🙍 neg-495.pgm	23/10/2003 07:16	Fichier Portable Gr	4 Ko
🙍 neg-496.pgm	23/10/2003 07:16	Fichier Portable Gr	4 Ko
🙆 neg-497.pgm	23/10/2003 07:16	Fichier Portable Gr	4 Ko
🙆 neg-498.pgm	23/10/2003 07:16	Fichier Portable Gr	4 Ko
neg-499.pgm	23/10/2003 07:16	Fichier Portable Gr	4 Ko
pos-0.pgm	23/10/2003 07:17	Fichier Portable Gr	4 Ko
pos-1.pgm	23/10/2003 07:17	Fichier Portable Gr	4 Ko
pos-2.pgm	23/10/2003 07:17	Fichier Portable Gr	4 Ko
pos-3.pgm	23/10/2003 07:17	Fichier Portable Gr	4 Ko
pos-4.pgm	23/10/2003 07:17	Fichier Portable Gr	4 Ko
pos-5.pgm	23/10/2003 07:17	Fichier Portable Gr	4 Ko
(a) pos s.pgm	20, 10, 2000 01111		4100

Figure 1 – Image files - The three first letters specify the class membership

3 Image classification using Knime



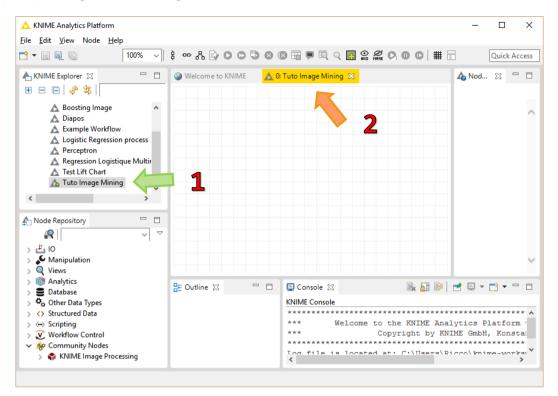
<u>KNIME Analytics Platform</u> is a free data mining tool. We must install the "<u>Knime Image Processing</u>" module which appears as a new branch into the Node Repository.

3.1 Building a workflow

We create a new workflow. We set the name: Tuto Image Mining.

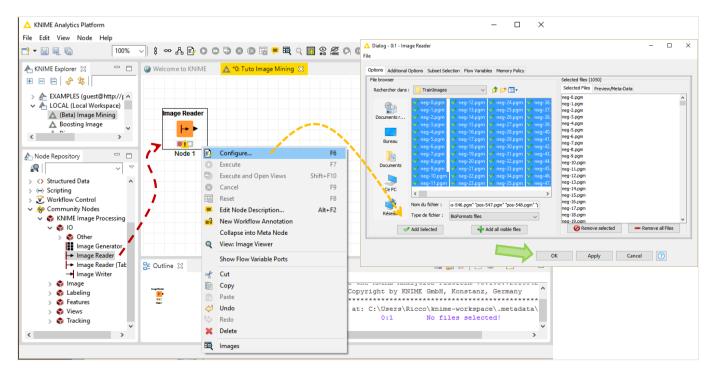
Δ	KNIME Analytics Platfor	m					_		×
File	Edit View Help								
	New	Ctrl+N					C	Quick Acc	cess
	Save	Ctrl+S						w =	
<u>.</u>	Save As)	Welcome to KNIME SS				Ao Node Descripti	× -	
G	Save All	Ctrl+Shift+S	Open for Innovat	ion		~			
	Close All	Ctrl+Shift+W		1F	7				
	Print	🛆 New		– 🗆 X		Δ	- 0	×	
		Select a wizard				New KNIME Workflow Wizard			
R	Export KNIME Workflc	This wizard creates a new KI	NIME workflow project.		to KN	Create a new KNIME workflow.			
	Switch Workspace				know you				
	Preferences	Wizards: type filter text			Kilow you	Name of the workflow to create: Tuto Image Mining			
R	Export Preferences	New KNIME Workflo	W		es and KN				
÷	Import Preferences	Rew KNIME Workflo			ady when		Bro	wse	
	Install KNIME Extensio	> 🦢 General > 🎘 KNIME			ple workf		0.0		
	Update KNIME				g&drop n				
	Exit				ig a new				
\rightarrow	Analytics				data typ kstart Gu				
2	Database So Other Data Types				kstart ou				
Ś	<>> Structured Data	_ /	*		you can				
	↔ Scripting	Show All Wizards.			you carr				~
	Workflow Control Community Node				ole 🖂				
l í	A community Node				onsole				
	-				onsole *****				* ^
		< 1	Back <u>N</u> ext >	inish Cancel	We	< <u>B</u> ack <u>N</u> ext >	<u>F</u> inish Can	:el	3;
				Log f	ile is	located at: C:\Users\Ricco\knime-w	orkspace\.metada	a\knir	m
									\sim
				<				3	Þ
									_

The project is visible into the "Knime Explorer" (1). The workspace is opened (2). We can perform our analysis by adding and connecting the nodes.

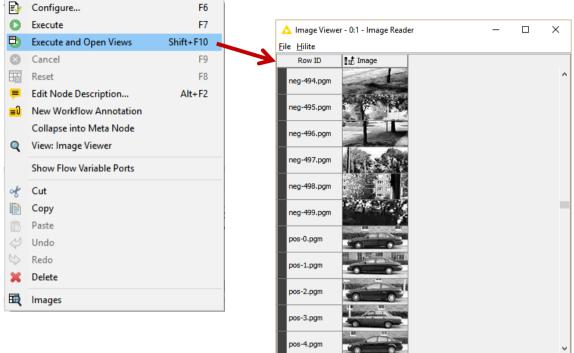


3.2 Loading et visualizing the images

The first step is to browse the items in the folder in order to load the images. All this without having to write one line of source code. We use the **IMAGE READER** component.



We click on Configure. We select the images to handle (Add Selected). We click on OK.



Click on a cell or drag and select multiple cells to continue ...

In order to visualize the images, we click on the contextual menu Execute and Open Views of IMAGE READER¹. They appear in a specific window.

3.3 Feature extraction

The <u>feature extraction</u> process consists in to extract from each image a numeric vector which characterizes the image and which will be used in the modeling step. These features must be informative and non-redundant. Clearly, the feature extraction step is crucial for the accuracy of the subsequent classifier.

<u>Note</u>: I was talking about democratization of image processing. It should be noted that rudimentary knowledge about the domain is necessary in order to understand and choose the right technique in the feature extraction step. The trial and error method is not efficient when the number of tools and the corresponding parameters are high.

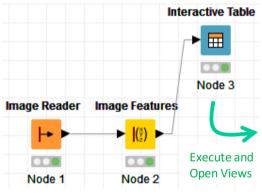
We add the **IMAGE FEATURES** node (Community Nodes / Knime Image Processing / Features). We set the following parameters (contextual menu Configure).

¹ Knime uses the Bio-Formats API (<u>https://www.openmicroscopy.org/site/support/bio-formats5.1/</u>).

Image Reader Image Feat	LITES A Dialog - 0:2 - Ir File	mage Features — — X
Node 1 Node	Column selection	

In the **Column Selection** tab, we select the "Image" column. In the **Features** tab, we select the features to extract. We set only First Order Statistics in a first time. We validate by clicking on the OK button.

The features values can be visualized with the **INTERACTIVE TABLE** node (Views). We click on the Execute and Open Views contextual menu.



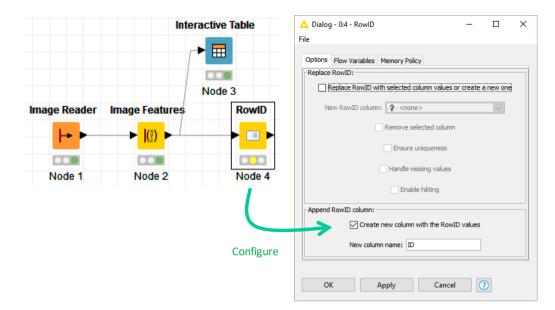
🛕 Table View - (0:3 - Inte	ractive Tal	ble		_		
<u>F</u> ile <u>H</u> ilite <u>N</u> avig	ation <u>V</u> i	iew Outp	ut				
Row ID	D Min	D Max	D Mean	D Geo	D Sum	D Square.	
neg-495.pgm	3	254	128.796	97.321	515,184	87,411,228	\mathbf{x}
neg-496.pgm	3	251	128.184	94.728	512,736	87,396,792	
neg-497.pgm	4	255	130.801	97.656	523,203	92,050,499	
neg-498.pgm	2	251	127.174	92.391	508,697	86,538,189	
neg-499.pgm	0	255	63.133	0	252,530	31,949,552	
pos-0.pgm	1	255	134.251	103.377	537,002	94,181,534	
pos-1.pgm	4	255	132.752	102.492	531,006	92,306,254	
pos-2.pgm	4	255	132.626	101.799	530,503	92,316,849	
pos-3.pgm	3	255	133.725	101.556	534,899	93,849,289	
pos-4.pgm	4	255	133.52	103.477	534,079	93,326,707	
pos-5.pgm	5	255	134.066	102.434	536,265	93,621,633	
200 £ 20m	-	255	104 711	0	100 045	02 740 200	1

We obtain a tabular data on which we can apply a machine learning algorithm. The rows correspond to the images. The column are the features extracted from the images.

3.4 Creating the target attribute

Into the data grid above, the file names appear as **Row ID**. We must turn this column into a categorical variable by extracting the 3 first letters.

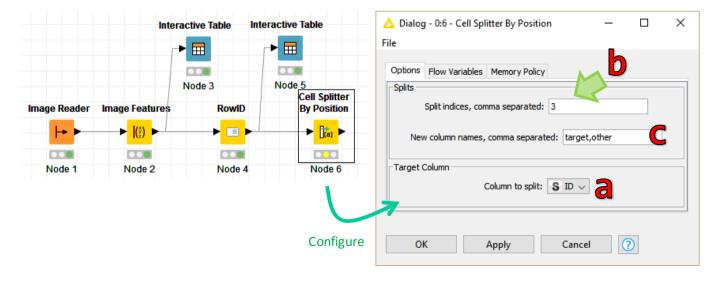
We use the **RowID** node (Manipulation / Row / Other) in a first step. We connect IMAGE FEATURES node to this new node. We set the following settings (menu Configure). A new column named **ID** is created from the RowID values.



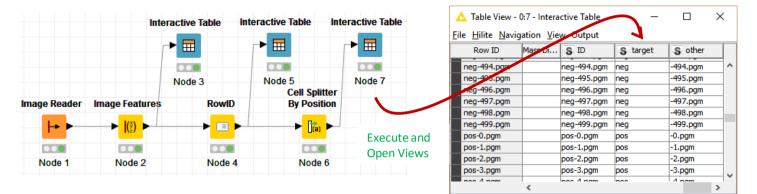
We launch the calculations by clicking on Execute. We add an INTERACTIVE TABLE (Views) to display the results. The ID column is a character string (**S**).

	Inte	ractive Table	Interactive Table	▲ Table View - <u>F</u> ile <u>H</u> ilite <u>N</u> avig					_		×	
	Γ	▶ 🖽		Row ID	D Weight	D	D	D.	D.	SID		
				neg-495.pgm	22.454	0	0	0	0	neg-495.pgm		L
			Nada 5	neg-496.pgm	17.018	0	0	0	0	neg-496.pgm		L
		Node 3	Node 5	neg-497.pgm	15.788	0	0	0	0	neg-497.pgm		L
Image Reader	Image Features	RowID		neg-498.pgm	18.566	0	0	0	0	neg-498.pgm		L
	-			neg-499.pgm	14.46	0	0	0	0	neg-499.pgm		L
 → 	► <mark> (ἔ)</mark> ►/	□		pos-0.pgm	19.479	0	0	0	0	pos-0.pgm		L
			Execute and	pos-1.pgm	20.839	0	0	0	0	pos-1.pgm		l
			Open Views	pos-2.pgm	19.434	0	0	0	0	pos-2.pgm		L
Node 1	Node 2	Node 4	open views	pos-3.pgm	19.762	0	0	0	0	pos-3.pgm		L
				nos-4 nom	10 791	n	n	n	n	nos-4 nom		l
					<						>	Ł

In a second step, we extract the first ₃ characters in order to create a new column that will represent the target variable. We use the **CELL SPLITTER BY POSITION** node (Manipulation / Column / Split & Combine). We set the following parameters values:

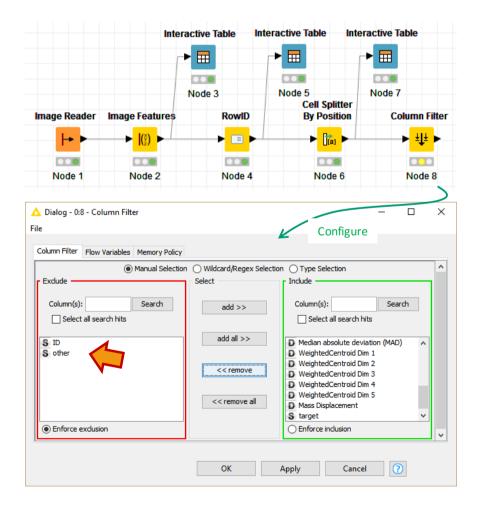


We handle the column ID (a). We cut from the 3rd character (b). Two new columns – "target" and "other" – are created (c). We check this with the INTERACTIVE TABLE node. The "target" column is now usable for supervised learning process.

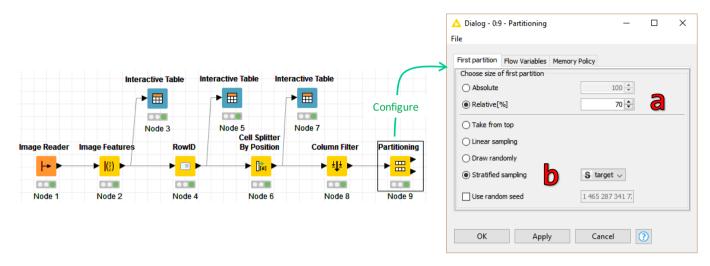


3.5 Learning and evaluating the predictive model

Removing the unused columns. Our goal is to predict the values of the target attribute from the features extracted from the images. We must remove the unused columns from the current dataset (ID and other). We use the **COLUMN FILTER** node (Manipulation / Column / Filter).



Partitioning the data into train and test sets. To obtain an honest evaluation of the classifier performance, we use the holdout approach i.e. we partition the data into training sample on which we fit the classifier, and test sample on which we measure its predictive performance.



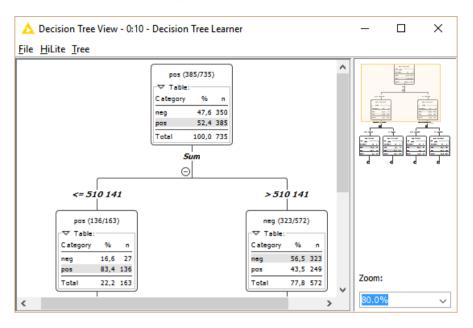
We insert the **PARTITIONING** node (Manipulation / Row / Transform) into the workflow. We select 70% of the instances as training sample (a). We carry out a stratified sampling on the "target" attribute so that the proportions of the positive and negative are strictly the same in the two samples (b).

Fit the model. We want to use a decision tree algorithm in a first time (we will use another machine learning method below, section 4.1). We add the **DECISION TREE LEARNER** node (Analytics / Mining / Decision Tree) into the workflow.

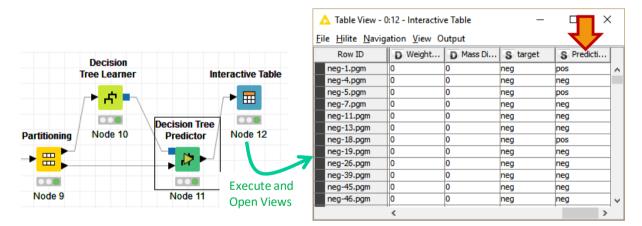
	🛆 Dialog - 0:10 - Decision Tree Learner 🛛 🚽 🗙
	File
	Options PMMLSettings Flow Variables Memory Policy
\longrightarrow	General Class column S target v
Configure	Quality measure Gini index 🗸
	Pruning method No pruning 🗸
Decision	Reduced Error Pruning
Tree Learner	Min number records per node 2
Partitioning ► 古	Number records to store for view 10 000 🜩
	Average split point
Node 10	Number threads 8 🜩
Node 9	Skip nominal columns without domain information
	Binary nominal splits
	Binary nominal splits
	Max #nominal 10 🛓
	Filter invalid attribute values in child nodes
	OK Apply Cancel

We check that Class column is "target". We do not modify the other settings.

We visualize the tree by clicking on the Execute and Open Views contextual menu. The tree itself is anecdotal in our context. We do not interpret it.



Evaluation of the classifier. We apply the classifier on the test set to evaluate its accuracy. We insert the **DECISION TREE PREDICTOR** node (Analytics / Mining / Decision Tree) to obtain the prediction of the model. The tool takes as input the learned model and the test sample. We use the INTERACTIVE TABLE to visualize these predictions.

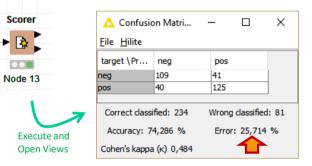


The column Prediction(target) has been generated.

We build the <u>confusion matrix</u> by using the **SCORER** node (Analytics / Mining / Scoring). Each row represents the instances in an actual class [target]. Each column represents the instances in a predicted class [Prediction(target)]. We launch the calculations by clicking on the Execute and Open Views contextual menu.

	Decision Tree Learner	Inte	ractive Tab	le	▲ Dialog - 0:13 - Scorer — □ × File Scorer Flow Variables Memory Policy First Column
Partitioning	Node 10	Decision Tree Predictor	Node 12	Scorer	Second Column S Prediction (target) Sorting of values in tables Sorting strategy: Insertion order Provide scores as flow variables Use name prefix Missing values In case of missing values In case of missing values
				Configure	OK Apply Cancel

We get the following confusion matrix. The test error rate is **25.714%**.



3.6 Conclusion

Here is the workflow as a whole.

	Interact	ive Table Inte	eractive Table Intera	active Table				
						Decision		
		===	→ 📰 🚽	• ==		Tree Learner	Inte	eractive Table
						ר <mark>י ה'</mark> ∎ר		▶ 표
	N	ode 3		Node 7			Desision Tree	
maga Daadar II		DavidD	Cell Splitter	Column Filter	Destitioning	Node 10	Decision Tree Predictor	
Image Reader Ir	nage Features	RowID	By Position	Column Filter	Partitioning	Node To	Predictor	Node 12 Scorer
→	→ (() ►	──▶ 🔲 ▶┴		── <mark>▶ ॑</mark>	<mark>_ </mark>			
	N ₂ /							
Node 1	Node 2	Node 4	Node 6	Node 8	Node 9		Node 11	Node 13

An error rate of 25.714% is a little disappointing. We will see how to improve it in the next section.

But the most important here is that we were able to carry out the complete analysis very easily. To achieve the same result under R or Python, we must firstly determine the good libraries, identify the proper commands, and write the right instructions by respecting the syntax of the programming language.

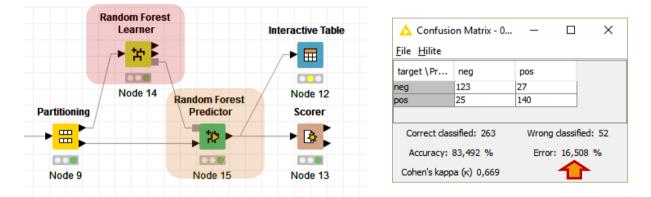
4 Improvement opportunities

The basic outline seems right. But the results are disappointing. In this section, we explore some simple sources of improvement.

4.1 Machine learning algorithm

The easiest way is to try out other machine learning algorithms. To compare them in our image classification context, we measure the test error rate.

Let us try the random forest approach². In the last part of the workflow, we replace the TREE nodes (LEARNER and PREDICTOR) with the **RANDOM FOREST** nodes. For the LEARNER, we define "target" as the Target column.



The test error rate becomes **16.508%**.

The improvement is dramatic. We know that the method is often efficient. In addition, in a representation space where a large number of descriptors - of which the relevance is not always proven - are likely to be generated, the ability of the Random Forest to prevent overfitting is a valuable property.

4.2 Extraction of other features

During the feature extraction step (section 3.3), we selected only the First order statistics into the IMAGE FEATURES tool. Other options are possible. We activate them even if we do not really understand the underlying calculations (Tamura and Haralick)³. Of course, the computation time will be longer because additional data are produced and processed by subsequent machine learning algorithms.

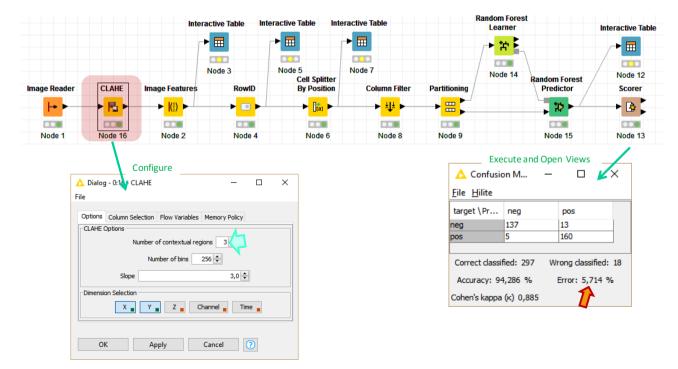
² R. Rakotomalala, "Bagging, Random Forest, Boosting", December 2015.

http://data-mining-tutorials.blogspot.fr/2015/12/bagging-random-forest-boosting-slides.html ³ N. Bagri, P. K. Johari, "<u>A Comparative Study on Feature Extraction using Texture and Shape for Content</u> <u>Based Image Retrieval</u>", in International Journal of Advanced Science and Technology, 80, 2015 ; pp. 41-52.

	Interact	ive Table Intera	active Table Intera	ctive Table		idom Forest Learner	Interactive Tal
			•			★ ★ ▲	_ ▶ 🔳
nage Reader		ode 3		Node 7 Column Filter	Partitioning	Node 14 Random Forest Predictor	Node 12
 -	→ <mark> (≬)</mark> →	→□		─ <mark>─</mark>			
Node 1	Node 2	Node 4	Node 6	Node 8	Node 9	Node 15	Node 13
Column select	tion Features Flow Variables	ASM			<u>F</u> ile <u>H</u> target	<u>-</u> ilite :\Pr neg pos	
		Memory Policy				Confusion M — <u>H</u> ilite	×
Haralick			trast DIAGONAL ANTIDIAGONAL		neg pos	136 14 8 157	
			oute Average		Corre	ct classified: 293 Wrong	classified: 22
		greylevel distance	40 🜩			racy: 93,016 % Error 's kappa (к) 0,86	: 6,984 %
	0	K Apply	Cancel		Conen	3 rappa (N) 0,00	

The test error rate becomes **6.984%**. The improvement is also substantial as previously (modifying the machine learning algorithm). In a real study context, it is worth to look at the characteristics of these new descriptors and what they provide in predictive modeling. Studying the features extraction techniques is essential when one wishes to go further.

4.3 Modifying the properties of the images



But we can still go more upstream and look at the properties of images themselves. Various image processing operators allow to enhance the characteristics of the images (filtering noise, modifying the contrast, transforming the images into binary scale, etc.).

Here also, advanced skills are required if you wish to choose the right tools and finely adjust their settings. For our dataset, we want to adjust the contrasts with the CLAHE node (Community Nodes / Knime Image Processing / Image / Process) (see the workflow above).

The test error rate becomes **5.714%**. The improvement is minor here. We note above all that this kind of operation can influence the efficiency of the process.

4.4 Visualizing the misclassified images

Often, visualizing the misclassified images allows us to understand the inadequacies of the predictive model. It may suggest the way to improve the classification process.

First, we merge the images with the test set using the JOINER tool (Manipulation / Column / Split & Combine).

	Internetive Table II				andom Fores			
	interactive table in	iteractive Table In	teractive Table		Learner		Interactive Table	
		► ■			r 🕂		/ ▶ 🔳	
	Node 3	Node 5 Cell Splitter	Node 7		Node 14	Random Forest	Node 12	
E Image Featur	res RowID	By Position	Column Filter	Partitioning		Predictor	Scorer	Interactive Table
►	── <mark>─</mark> ■►		─ <mark>+</mark> ‡►	→ <mark></mark>		*	→ 🛃	/▶
16 Node 2	Node 4	Node 6	Node 8	Node 9		Node 15	Node 13	Node 18
							Joiner	
							_	
							Node 17	
		E Image Features RowID	E Image Features RowID By Position	E Image Features RowID By Position Column Filter	E Image Features Node 3 Node 5 By Position Column Filter Partitioning Column Filter Partitioning Column Filter Partitioning	Node 3 E Image Features RowID By Position ► ((2) ► ► Cell Splitter By Position Column Filter Partitioning Column Filter Partitioning	Node 3 Node 5 Cell Splitter Node 7 Node 14 Random Forest E Image Features RowID E E Column Filter Partitioning Image Features Image Features Image Features Image Features Image Features Image Features	Node 3 Node 5 Cell Splitter By Position Node 7 Image Features RowID Image Features RowID Image Features Node 12 Image Features RowID Image Features Image Features Node 12 Image Features RowID Image Features Image Features Node 12 Image Features RowID Image Features Image Features Node 12 Image Features RowID Image Features Image Features Node 12 Image Features RowID Image Features Image Features Node 12 Image Features RowID Image Features Image Features Image Features Image Features RowID Image Features Image Features Image Features Image Features Image Features Image Features Image Features Image Features Image Features Image Features Image Features Image Features Image Features Image Features Image Features Image Features Image Features Image Features Image Features Image Features Image Features Image Features Image Features Image Features Image Features Image Features Image Features Image Features Image Feat

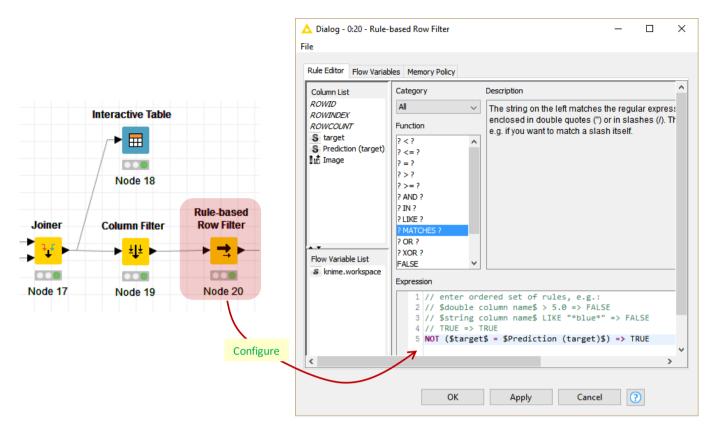
Into the settings dialog box (menu Configure), we set **Row ID** as joining column.

	Column Selection	Flow Variables	Memory Policy					
Ioin Mode								
		Join mode	Inner Join	\sim				
Joining Colum	ns							
Match all	l of the following (Match any of	the <mark>foll</mark> owing					
	Top Input ((left' table)		Bottom Input ('right' table)			
Row ID			~	Row ID	~	+	-	
						+		
			7	r				
			7	r				
			1	r				
Performance 1	Tunina							
Performance 1	-	nber of open file:	5: 200					
Performance 1	Maximum nun		5: 200					
	-		5: 200					
Performance T	Maximum nun	iliting						
	Maximum nun	iliting	s: 200 tor in joined table					
	Maximum nun	iliting		::				
	Maximum nun	iliting		2:	Cancel			

Then, with the COLUMN FILTER node, we keep the columns "images", "target" and "prediction(target)" in the current dataset.

	🛕 Dialog - 0:19 - Column Filter	– 🗆 X
	File	
Interactive Table	Exclude Select Induct Configure Select all search hits add >> D DifferenceVariance VERTICAL 1 40 add all >> D DifferenceVariance VERTICAL 1 40 Select D ClusterShade VERTICAL 1 40 Select D ClusterShade VERTICAL 1 40 Select D Min Mean D Geometric Mean V	umn(s): Search Select all search hits get ediction (target) age
Node 17 Node 19	Enforce exclusion	force inclusion
	ОК Арріу	Cancel

Last, we filter the dataset in order to retain only the misclassified instances. We use the **RULE-BASED ROW FILTER** node (Manipulation / Row / Filter).



We visualize these instances with the INTERACTIVE TABLE tool.

				🛕 Table View -	_		×				
				<u>File H</u> ilite <u>N</u> avigation <u>V</u> iew Output							
				Row ID	S Predicti	In <mark>t</mark> Ima	age				
	Interactive Table			neg-432.pgm	neg	pos		0	^		
				neg-436.pgm	neg	pos		Shere &			
	Node 18		\bigcap	neg-464.pgm	neg	pos					
Joiner	Column Filter	Rule-based Row Filter	Interactive Table	pos-69.pgm	pos	neg	-0				
	<mark>→ ∜</mark>	→ <mark>→</mark> ≻		pos-97.pgm	pos	neg					
Node 17	Node 19	Node 20	Node 21	pos-101.pgm	pos	neg	9		~		

The image n°69 (pos-69.pgm) for instance corresponds to a car [target = pos] which was not correctly identified [prediction(target) = neg]. We understand the reason. Its rear part is hidden. In doing so, the analyst has the opportunity to explore the different situations and, in some situations, to propose solutions that improve performance.

Here is again the workflow as a whole:

						R	andom Fores	at 👘					
		Int	eractive Table In	teractive Table In	nteractive Table Learner			Interactive Table					
			▶	▶ ■			→ ' * ►	1					
			Node 3	Node ⁵ Cell Splitter	Node 7		Node 14	Random Forest	Node 12		Interactive Table		
	CLAHE	Image Features	RowID	By Position	Column Filter	Partitioning		Predictor	Scorer				
	• 🖪 🕨		_	Fin +	▶ ± ± ▶	<mark></mark>		* *					
		- 1 (2)	· · · · · · · · · · · · · · · · · · ·			· · · ·							
											Node 18		
	Node 16	Node 2	Node 4	Node 6	Node 8	Node 9		Node 15	Node 13		/		
Image Reader										Joiner	Column Filter	Rule-based Row Filter	Interactive Table
										` <mark>`}</mark> ₩₩	<mark>→</mark> ⋕▶	<mark></mark>	
										Node 17	Node 19	Node 20	Node 21
Node 1													

5 Conclusion

The first goal of this tutorial is to show the image classification process on a simple example. We realize that the approach fits perfectly into the analysis of unstructured data framework. In the current context where the data scientist must know working on various data sources and formats, the skill about image processing becomes necessary to statisticians and data miners, and even essential. The analysis may be mixed with text mining process when both images and textual information are available.

The second objective was to show that powerful tools are at our disposal. With Knime, we can perform an efficient analysis with a lot of ease. I am very impressed by the capabilities of the tool.

6 References

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

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