TP – KNIME & Jupyter Setup

Jacky Casas, Joël Dumoulin, Leonardo Angelini

Goals

The laboratories of the Machine Learning course will be done with different tools. One of them is **KNIME**. We'll use it to speed up the creation of pipelines for our classifiers. It's a graphical tool that will help us understand how ML models work. We'll also use Python to code our own models. For this we'll use the **Scikit-learn** library. For the ease of prototyping, coding and sharing our code, we suggest using **Jupyter**. This document will guide you through the process of setting up all what's needed to use these ML tools with ease.

KNIME

KNIME[1] (pronounced /naɪm/), the Konstanz Information Miner, is an open source data analytics, reporting and integration platform. KNIME integrates various components for machine learning and data mining through its modular data pipelining concept. A graphical user interface allows assembly of nodes for data preprocessing (ETL: Extraction, Transformation, Loading), for modeling and data analysis and visualization without, or with only minimal, programming.[2] This platform is based on Eclipse and developed in Java.

Download & install

Download "KNIME Alanytics Platform" (not Server, nor Software in the Cloud) from the official website: https://www.knime.com/downloads

From there, you can click on the second tab "Download KNIME" and choose the last version for your OS (should be version 3.5.2 or higher).

On the 3rd tab, you have useful links

- Building a Workflow: https://www.knime.com/getting-started
- 7 things to do after installing KNIME Analytics Platform: https://www.knime.com/blog/seven-things-to-do-after-installing-knime-analytics-platform
- Learning Hub: https://www.knime.com/learning-hub
- Examples of usage: https://www.knime.com/applications

[1] https://www.knime.com
[2] source: https://en.wikipedia.org/wiki/KNIME

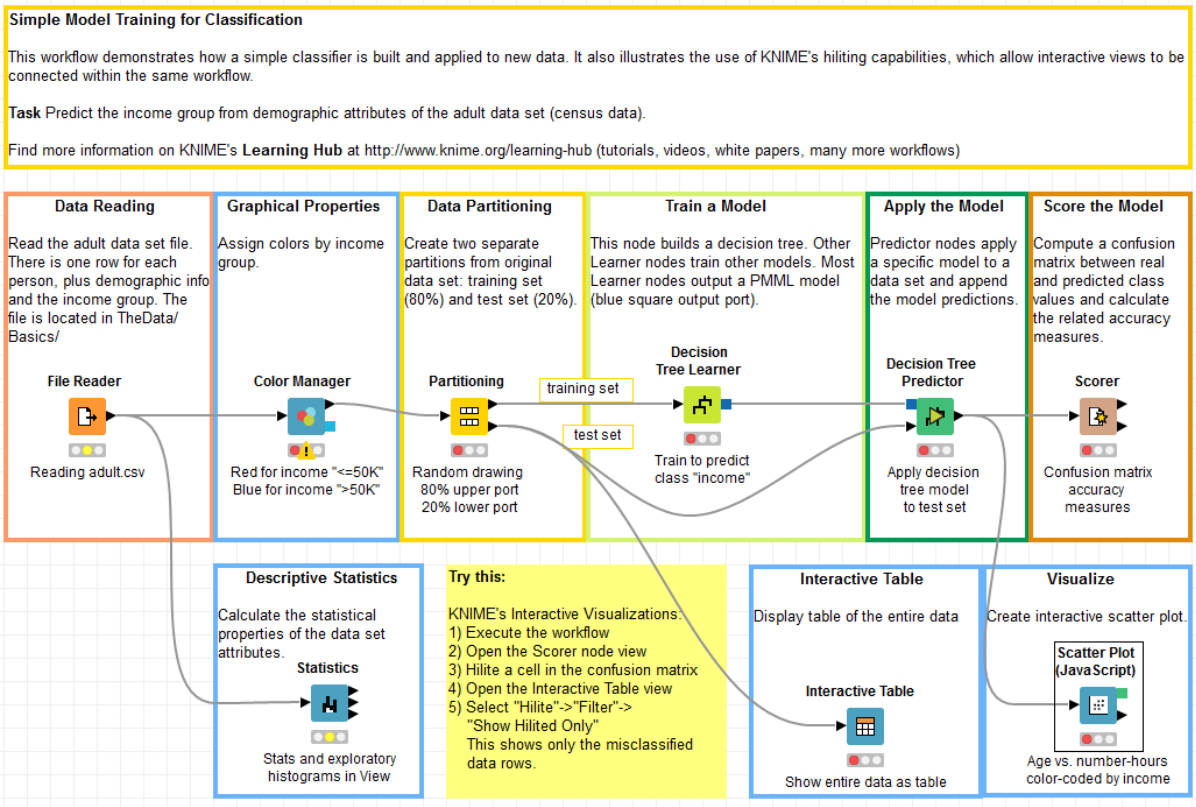
Tasks

Our first mission is to get to know the tool. For this we will open one of the existing examples directly in Knime. Go in the KNIME Explorer (top left of Knime) and click on "LOCAL", then "Basic Examples", and finally on **"Building a Simple Classifier"**. This will open a full experiment with explanations of each of the nodes used (see figure below).

Under each node, you see traffic lights. Red means you can't execute it yet, yellow means you can execute it, and red means it's already executed. You can execute nodes one by one or all at the same time.

To set the parameters of a node, just double-click on it, a window will pop up. To run actions on a node, right-click on it.

You mission is to navigate through **each node**, understand its purpose, execute it. Try to understand the data you have on the input and output of each node.



Now you'll see a **live demo** from me (Jacky) of KNIME. Pay attention to it, and **do the process at the same time on your computer**. It's the best way to learn a tool. The dataset used can be downloaded from this link: www.iainpardoe.com/teaching/dsc433/data/Churn.xls

Jupyter

Jupyter Notebook App is an application running in the browser. With it you can create notebooks. Notebooks are documents that contain both code and rich text elements (figures, links, equations, ...). Notebooks use kernels that are related to programming languages (Python, Julia, R, and so on) to execute you code.

For this course, we'll use Python 3 to run all of our experiments.

To install Jupyter, you'll need Python first. Then you can install it with Anaconda (recommended) or pip. Here are the instructions: http://jupyter.org/install

Once done, just launch Jupyter with the command "jupyter notebook" in your terminal. The app will be launched on your browser, but you don't need an internet connection to use it (it's local).

Note: JupyterLab, the evolution of Jupyter Notebook, was announced and is in beta since 20.02.2018. You can use it (at your own risks, but it's more beautiful). Check the doc: http://jupyterlab.readthedocs.io.

Our first notebook

To create a notebook, just click on "New" and select "Python 3" as kernel. A new tab will open with your empty notebook. From here you can start writing.

Note: The notebook will be saved in the folder where you were while launching Jupyter.

A notebook is composed of cells, that you can run one by one or all at once. A cell is either a "code cell" or a "text cell". By default it's a Python code cell. To change a cell to text, click on "Cell/Cell Type/Markdown". Markdown is a lightweight markup language (remember the Git TP of the SI-I course last year? :P) you'll use to format your text.

For those who are new to Python and/or Notebooks, we suggest you to **read the tutorial** made by Jean Hennebert. It's a notebook called "intro-python-3.ipynb" that you can find on Moodle.

Play a bit with the tool: try to run different cells, import new Python libraries, display images, and so on. Here are useful links for you:

- Markdown: https://en.wikipedia.org/wiki/Markdown
- Markdown cheatsheet: https://github.com/adam-p/markdown-here/wiki/Markdown-Cheatsheet
- Tips & tricks for Jupyter: https://www.dataquest.io/blog/jupyter-notebook-tips-tricks-shortcuts
- Jupyter documentation: http://jupyter-notebook.readthedocs.io

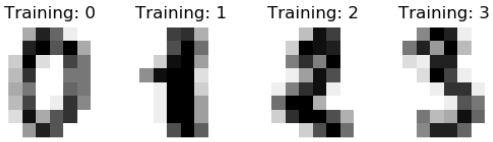
Scikit-learn

Scikit-learn is a Python library containing simple and efficient tools for data mining and data analysis. Plus it's open source (BSD license). We'll use it a lot during this course.

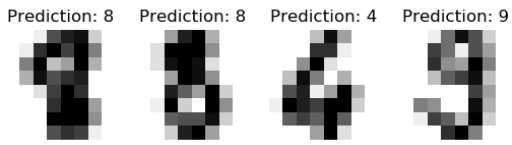
Official website: http://scikit-learn.org

As an example, we'll use a code that recognize hand-written digits. Go here: http://scikit-learn.org/stable/auto_examples/classification/plot_digits_classification.html

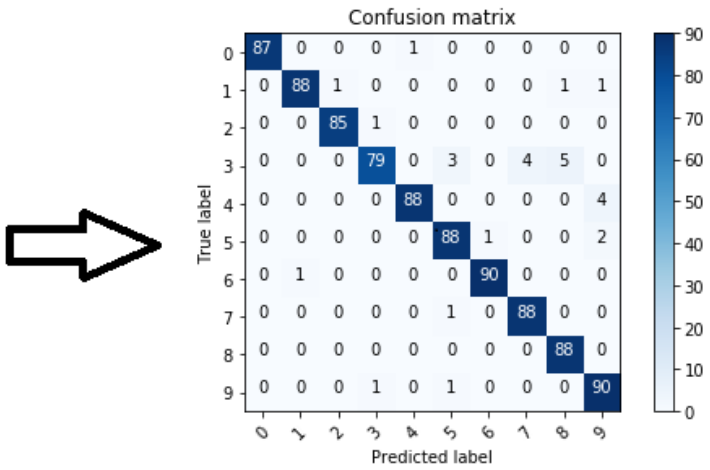
At the bottom of the page, you'll find a Jupyter notebook containing the code. Download it and run it locally. Try to understand how it works.



Then, **your mission is to create a confusion matrix with scikit-learn and matplotlib**[3]. The current code prints the confusion matrix (as you can see down here on the left). We want you to generate a chart of it (see on the right).




```
Confusion matrix:
[[87  0  0  0  1  0  0  0  0  0]
 [ 0 88  1  0  0  0  0  0  1  1]
 [ 0  0 85  1  0  0  0  0  0  0]
 [ 0  0  0 79  0  3  0  4  5  0]
 [ 0  0  0  0 88  0  0  0  0  4]
 [ 0  0  0  0  0 88  1  0  0  2]
 [ 0  1  0  0  0  0 90  0  0  0]
 [ 0  0  0  0  0  1  0 88  0  0]
 [ 0  0  0  0  0  0  0  0 88  0]
 [ 0  0  0  1  0  1  0  0  0 90]]
```



Hint: The documentation of Scikit-learn, Matplotlib, Numpy, and so on might be useful ;)

[3] https://matplotlib.org