Measuring performances of sklearn tools in classifying astroparticles detected by Cherenkov telescopes

Francesco Visconti - INAF/SSDC

This work

This (ongoing) work aims at a couple of things:

- comparing performances of a `C++ Shark` implementation of machine learning tools to recognize gammas from hadrons, with a `Python scikit-learn` one;
- comparing performances of different models implemented in `scikit-learn`, mainly `RandomForest` and `MultiLayerPerceptron`.

Models other than Neural Networks (MLP) and Forests have been tested, but Random Forest outperforms them all for this problem.

ASTRI Data Challenge #1

I used the ASTRI Data Challenge for this work:

- extracted from a dedicated ASTRI MC production (**end to end** Data challenge).
- two samples: a Crab Nebula observation (ON sample, ~5.8 hours equivalent) and an observation of an empty sky region (OFF sample, ~5.5 hours equivalent).

Workflow

- ➔ train three different classifiers:
  - ◆ Random Forest;
  - ◆ Extra Trees;
  - ◆ Multi Layer Perceptron.
- ➔ predict probabilities for events;
- ➔ estimate events' energy;
- ➔ compute a number of metrics to test the output.

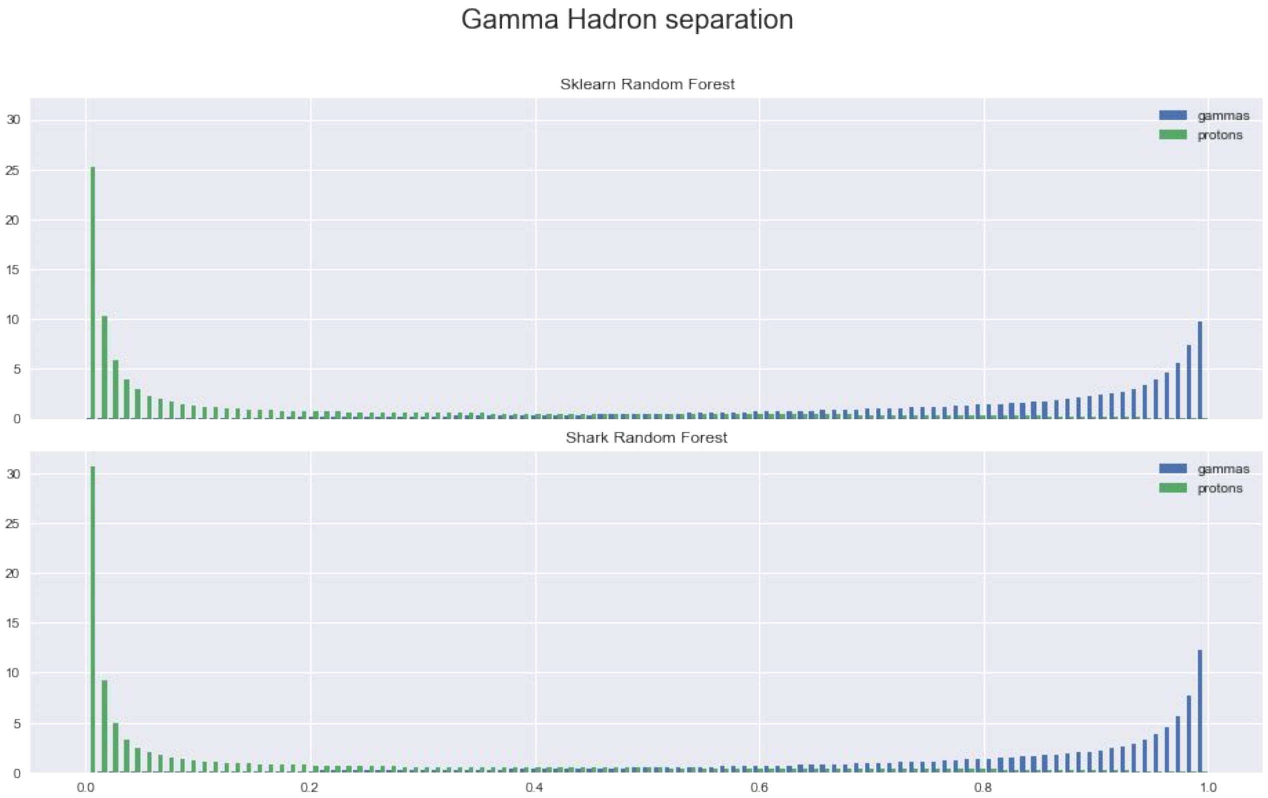
The inputs for this set of tools are Hillas parameters files. The multi layer perceptron method is used with one only hidden layer (*shallow network*) which **has been verified** to perform as well as a multiple hidden layers model (*deep network*) for this task.

Results

<u>Gamma - Hadron separation</u>

Comparison between Shark and scikit-learn libraries

Normed histograms showing the density of particles falling in bins of gammaness.



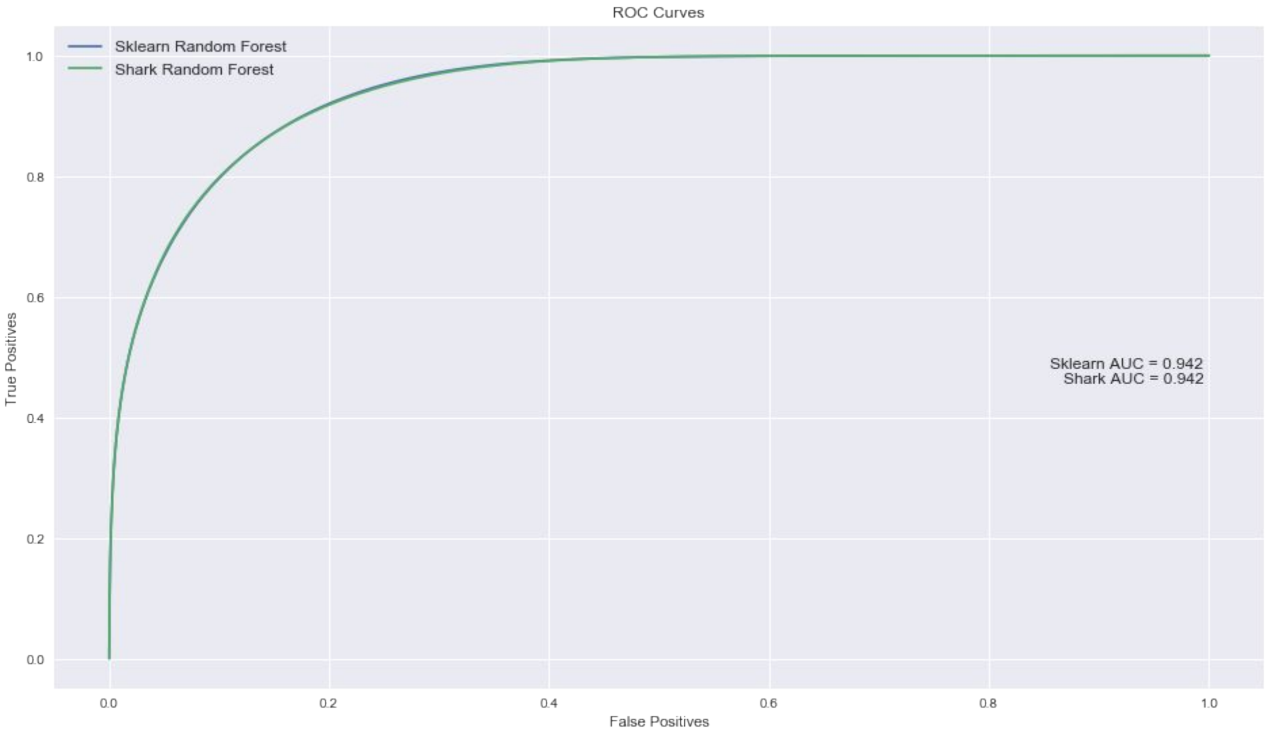
Results

<u>ROC curve</u>

ROC curves typically feature true positive rate on the Y axis, and false positive rate on the X axis. This means that the **top left corner of the plot is the “ideal” point**.

Larger area under the curve (**AUC**) is usually better.

It can be seen that the two methods are somewhat equivalent, a direct measure being the score, namely the area under the curves.



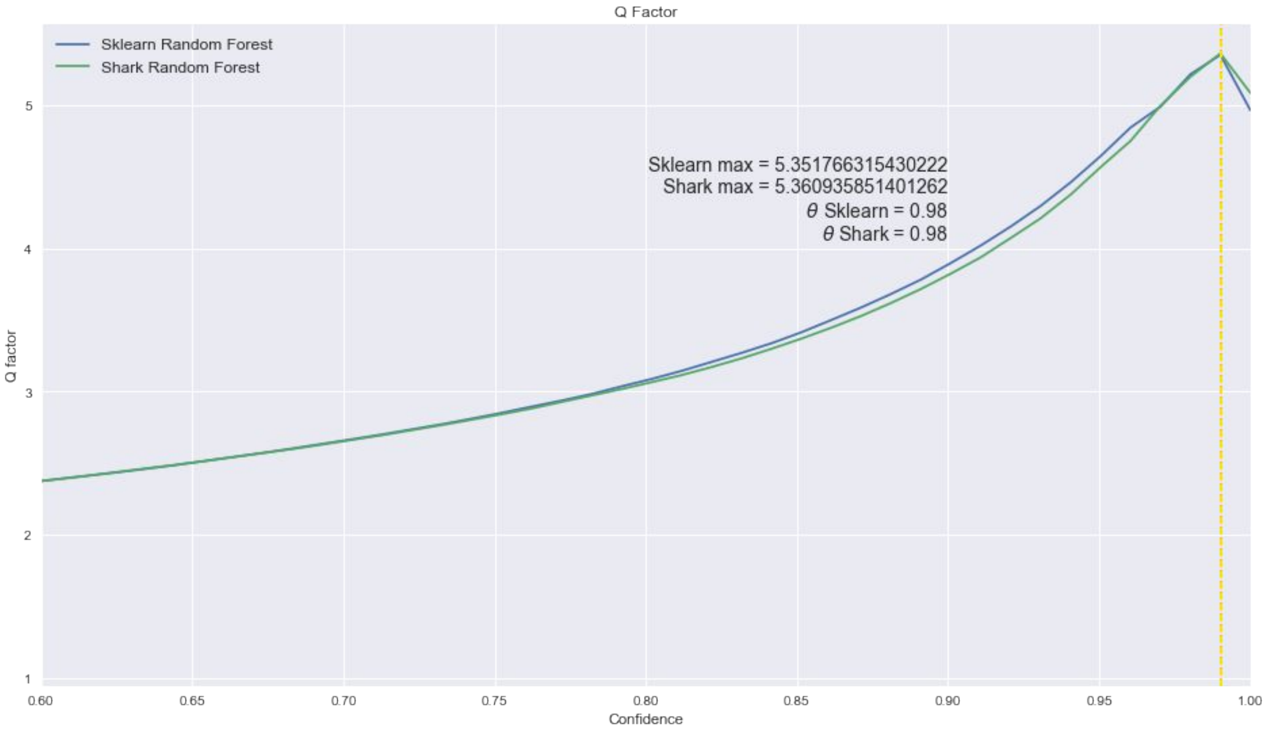
Results

Q-factor

Q factor is widely used in astrophysics literature to measure the goodness of classification: it is measured as

$$Q = \varepsilon\gamma \,/\, \sqrt{\varepsilon p}$$

where $\varepsilon\gamma$ is the fraction of **well classified gammas**, and εp is the fraction of **badly classified protons** (false positives). This is usually plotted against the *acceptance* (or *threshold*) for the probabilities.



Results

Precision - Recall

$$P = T_p \;/\; (T_p+F_p)$$
$$R = T_p \;/\; (T_p+F_n)$$

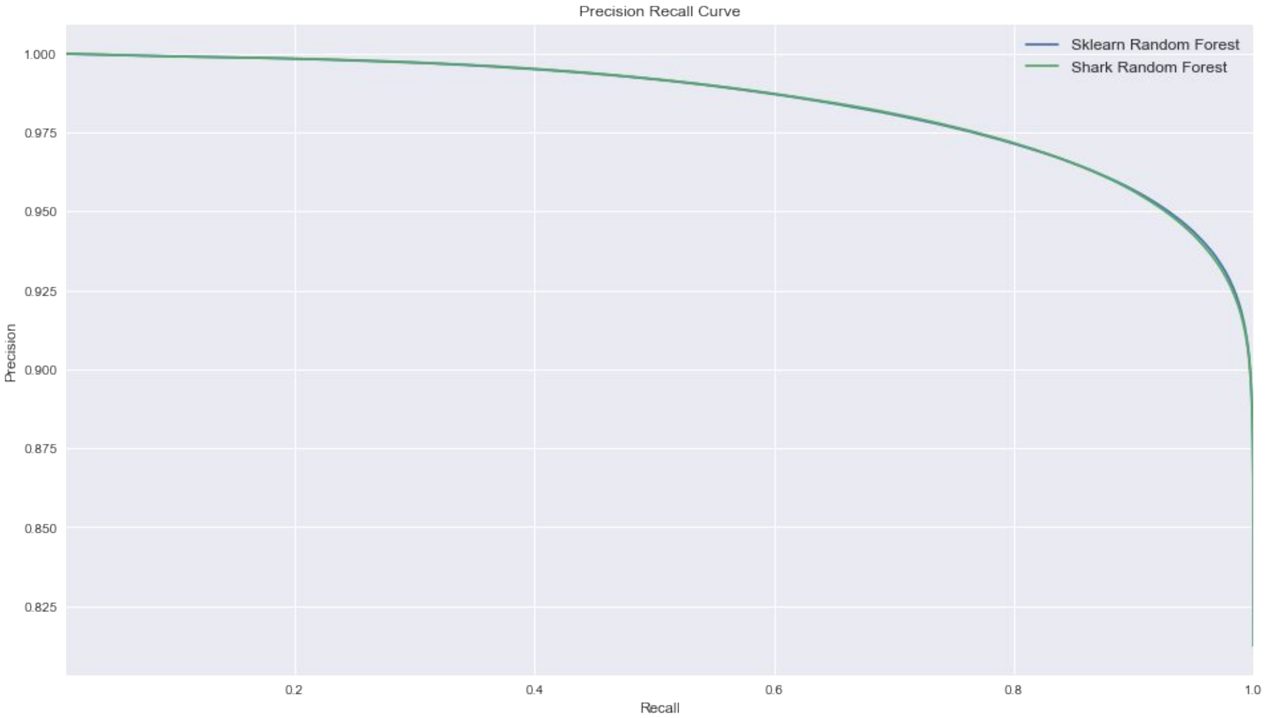
Where

T_p are gammas correctly labelled

F_p are hadrons labelled as gammas

F_n gammas labelled as hadrons

Low Precision means many hadrons have been mis-labelled, while a low Recall means many gammas have been classified as protons (and then lost!)



Comments

It can be said `Shark` and `scikit-learn` implementations of Random Forest seem to have the same performances on this dataset.

From the gamma-hadron separation plot, it seems `Shark` is able to better fill the edge bins, thus assigning probabilities close to 0 (hadrons) and 1 (gammas) to more individuals in the population.

Other thing worth noting, `sklearn` is much **faster** in execution: the same dataset used for training the classifier object is run in minutes from the `Python` library, hours from `Shark`.

Confusion Matrix for Sklearn Random Forest Classifier

From wikipedia:

> *In the field of machine learning and specifically the problem of statistical classification, a confusion matrix, also known as an error matrix, is a specific table layout that allows visualization of the performance of an algorithm. The name stems from the fact that **it makes it easy to see if the system is confusing two classes** (i.e. commonly mislabelling one as another).*

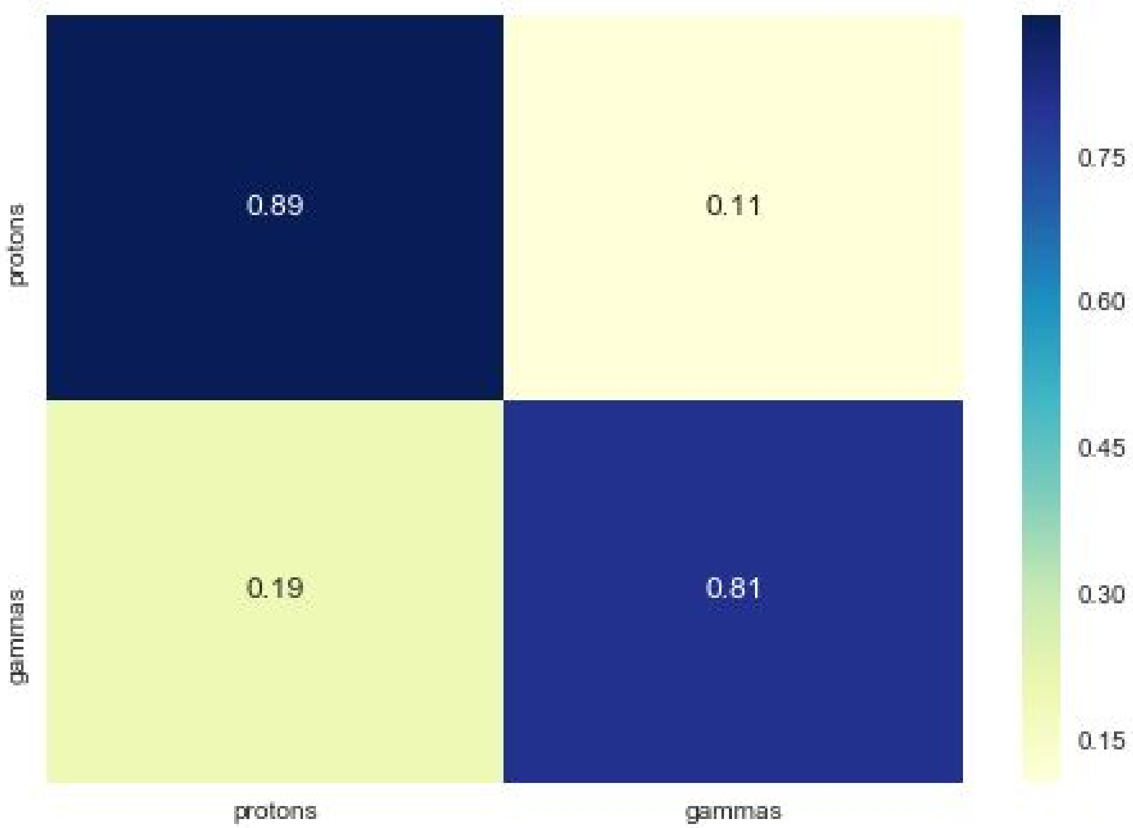
Sounds useful!

The `scikit-learn` functions operate on discrete labels (not probabilities), so I wrote a function to have these results **for each** threshold as defined for other metrics: `thresholds = np.linspace(0, 1, 102)`

Confusion Matrix

The diagonal elements represent the number of points for which the predicted label is equal to the true label, while off-diagonal elements are those that are mislabeled by the classifier.

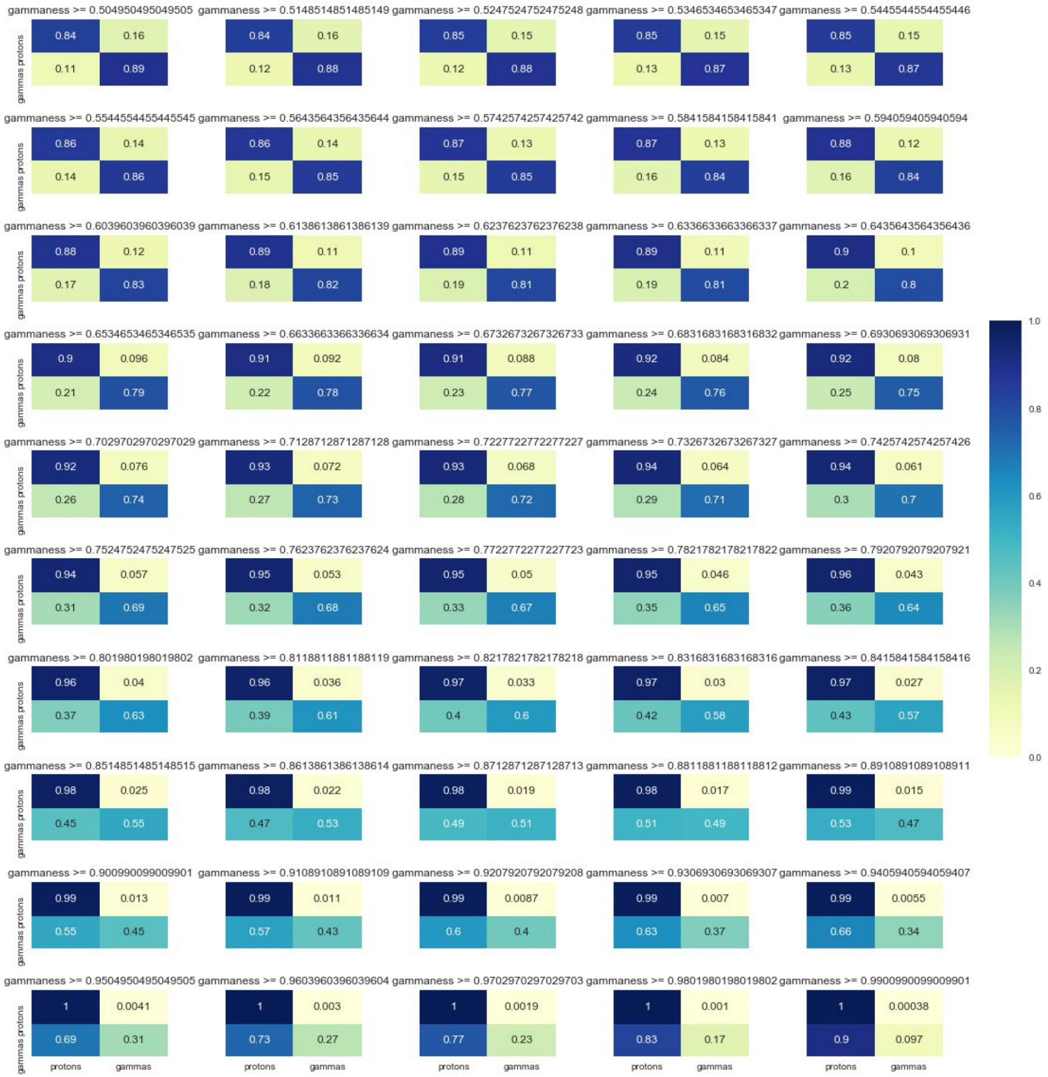
['True negatives', 'False positives']
['False negatives', 'True positives']



Confusion Matrix

Some interpretations:

- from a gammaness value of ~64% on, you start to throw away more than 20% of true gammas: this can be useful to know for if you need a proper amount of gammas to perform some scientific analysis (DL4 data);
- Q-factor is max at gammaness = 98%; this is because √fp (**false positives, background, protons classified as gammas**) are the denominator of the Q-factor and its value become very small as gammaness increase. The counter effect is for those values you are wasting ~80% of gammas.



Other Classifiers

Same input data have been passed through other machine learning methods, namely:

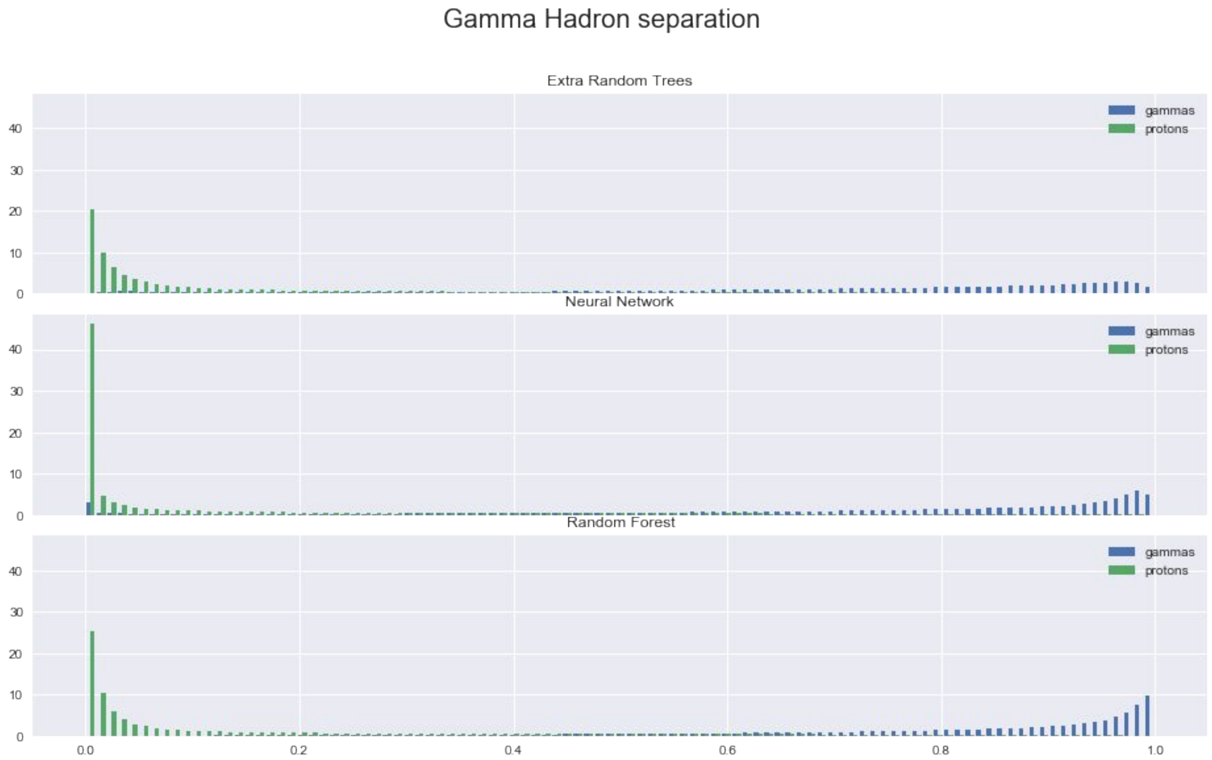
- Extra Random Trees
- Multi Layer Perceptron (Neural Network)

About Multi Layer Perceptron (Neural Network): it's been used a shallow network with 20 neurons in an only hidden layer, after a number of tests with two hidden layers and varying the number of neurons also: this shallow network gives the best result on this dataset, in a reasonable execution time.

Results

Gamma - Hadron separation

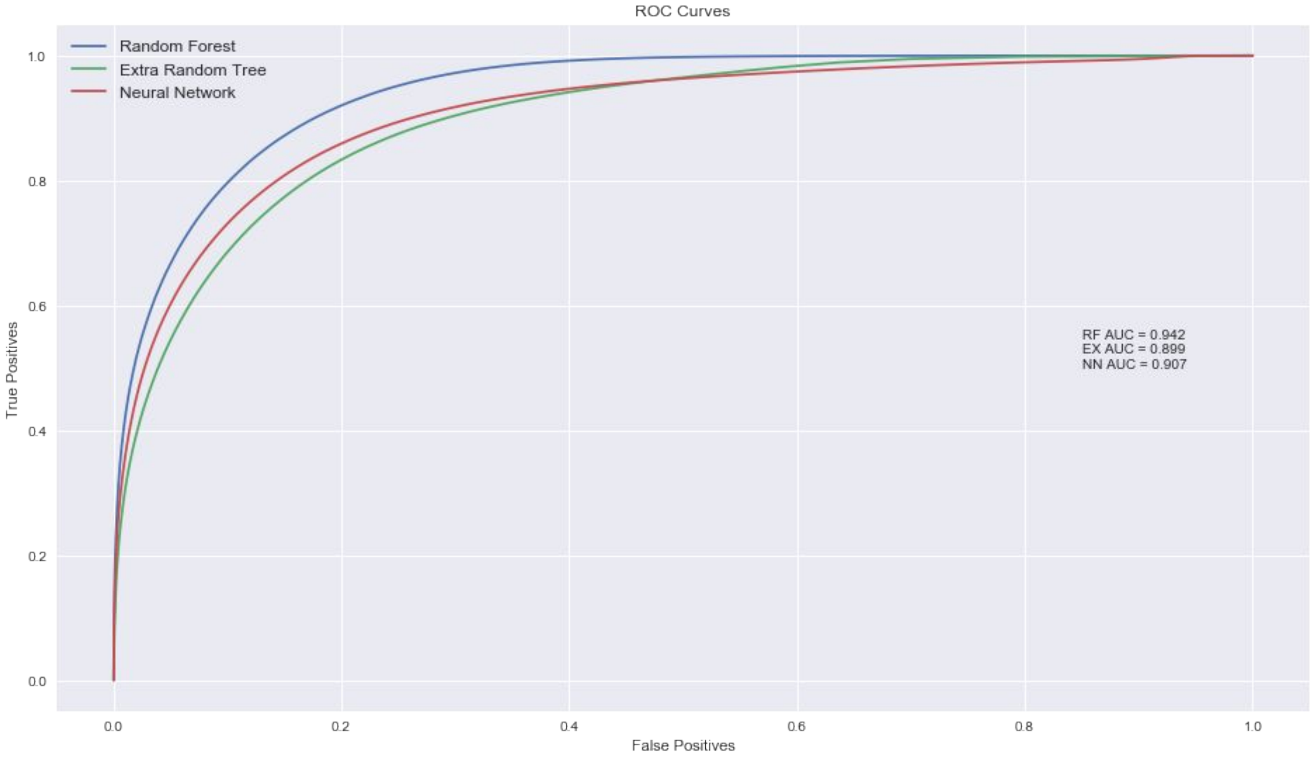
Random Forest leads



Results

<u>ROC curve</u>

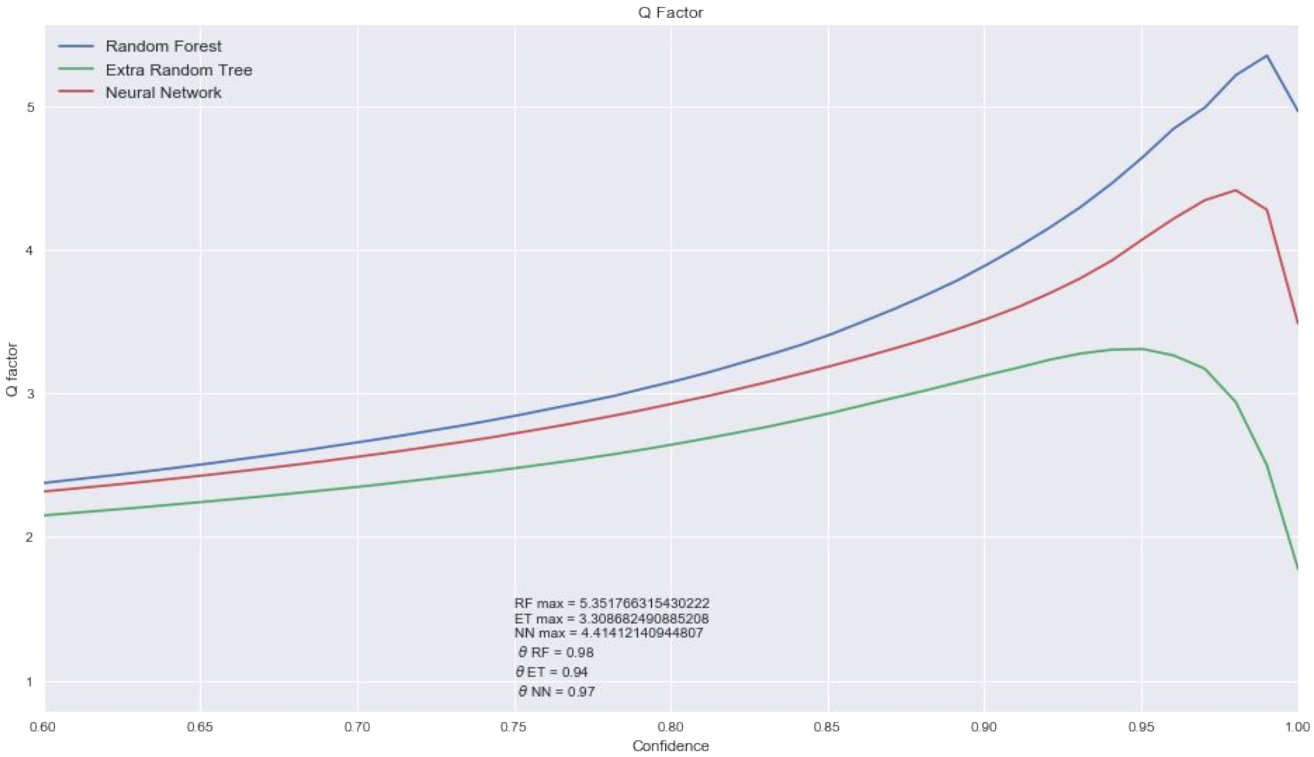
Random Forest leads



Results

Q-factor

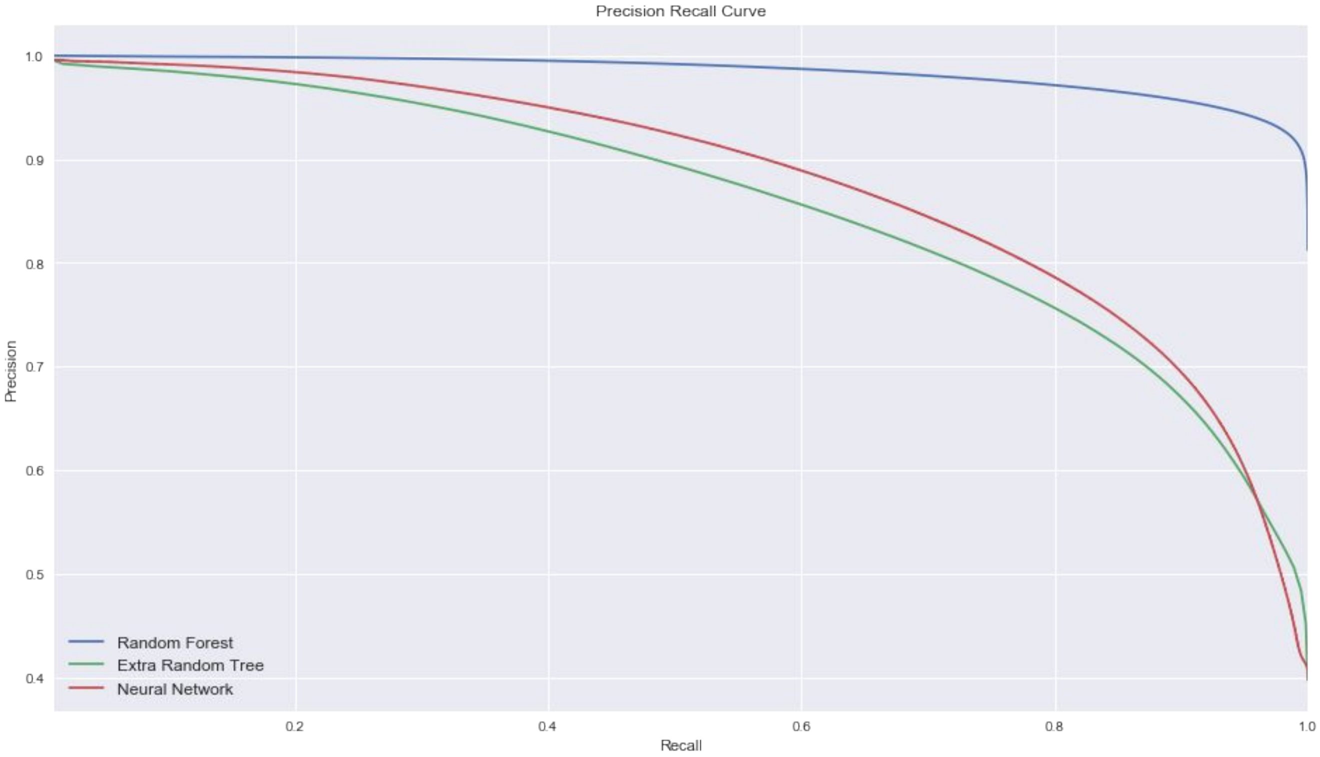
Random Forest leads



Results

<u>Precision Recall curve</u>

Random Forest leads



Comments

Random Forest leads