Comparing Random Forest and Convolutional Neural Networks on ASTRI dedicated MC production

Francesco Visconti - INAF/SSDC
francesco.visconti@oa-roma.inaf.it

Outline

This (ongoing) work aims at:

- 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 RandomForest model implemented in scikit-learn with a CNN implementation in keras.

INAF
ISTITUTO NAZIONALE DI ASTROFISICA
NATIONAL INSTITUTE FOR ASTROPHYSICS

ASTRI MC Data

I used an ASTRI dedicated MC simulated data (CTA Prod3b) for this work:

- 33 ASTRI telescopes in square layout
- Events from each telescope are used as *independent*, as if coming from a single *average* telescope
- A **data challenge** has been extracted from this dataset, but not used in this work

Details here: https://pos.sissa.it/301/804/pdf, section 4

Workflow

Predict class probabilities, energy, direction with:

➔ Scikit-learn libraries

➔ Shark libraries

The inputs for the above set of tools are Hillas parameters files (lv1b) filtered like this:

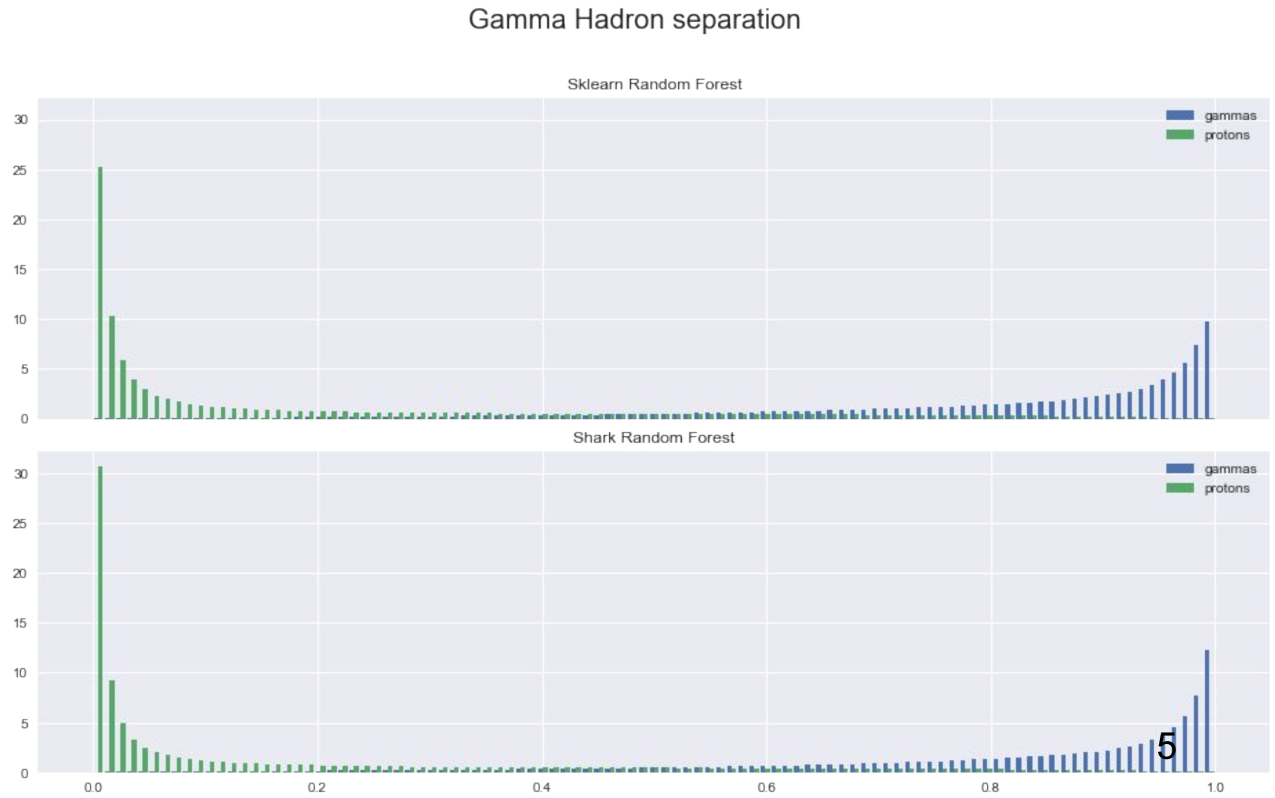
```
COLUMNS = ['SIZE', 'WIDTH', 'LENGTH', 'CONC', 'DIST', 'M3LONG', 'COSDELTAALPHA']
FILTERS = "SIZE > 50 && WIDTH > 0 && LEAKAGE < 0.1 && NUMISLAND < 2 && NUMCORE > 2 && LENGTH > 0"
```

Results

Gamma - Hadron separation

Comparison between Shark and scikit-learn libraries

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



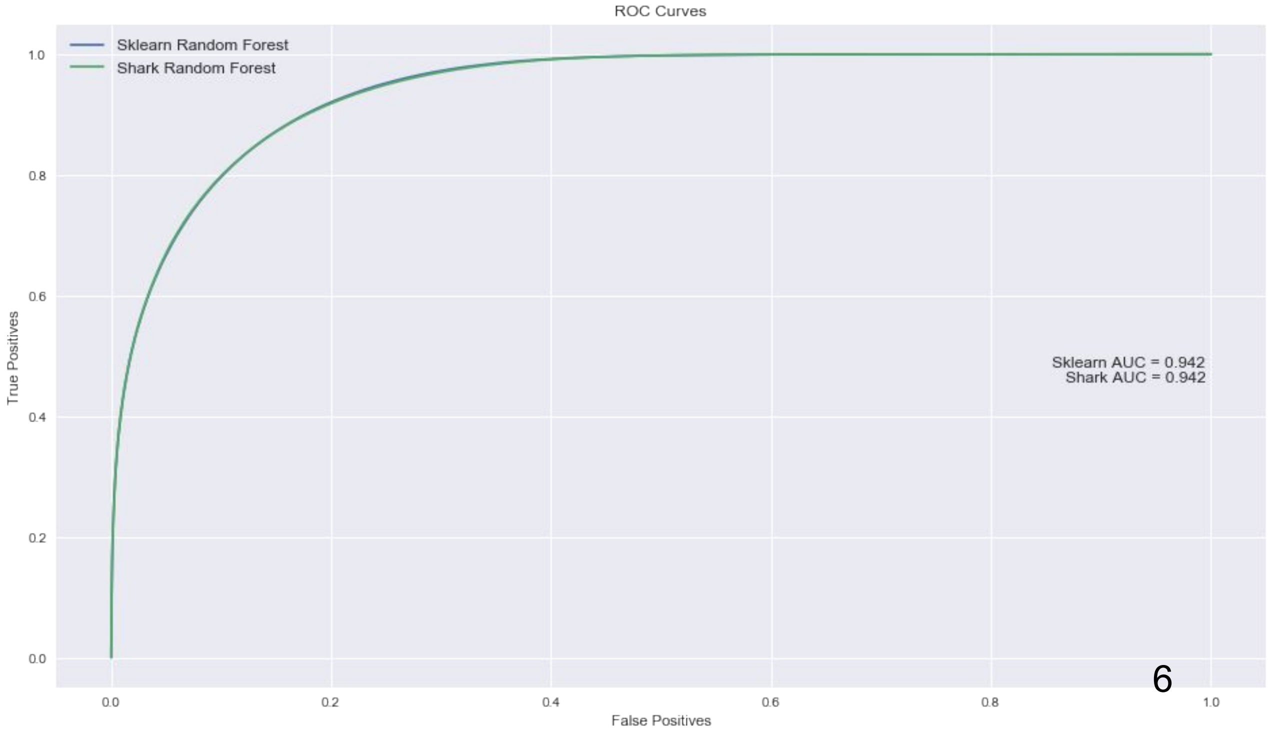
Results

ROC curve and area

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.



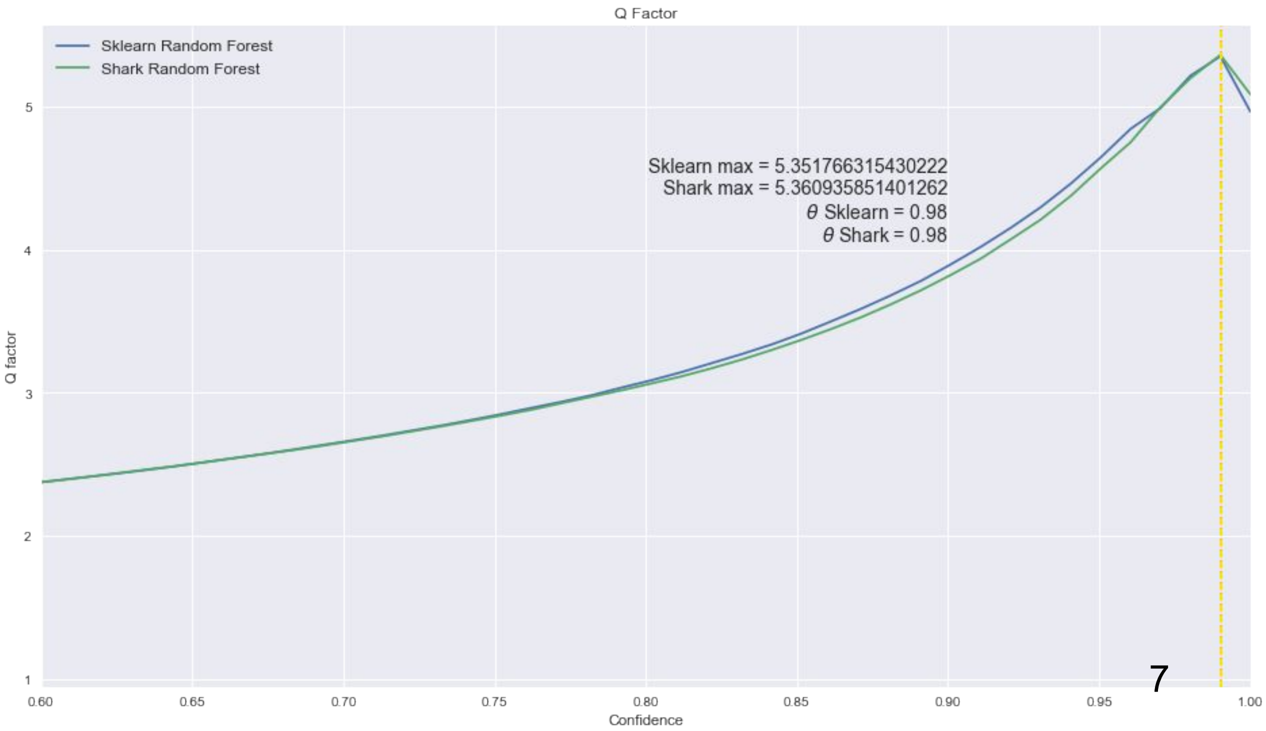
Results

<u>Q-factor</u>

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 εγ 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

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

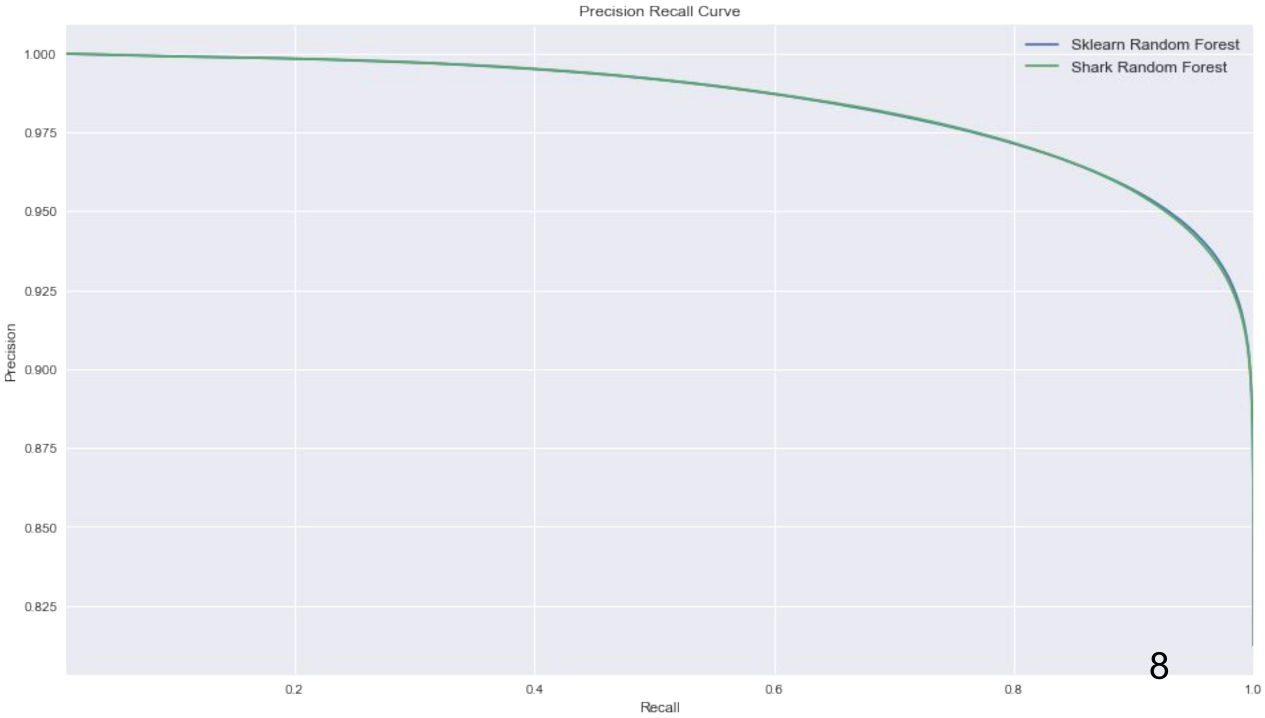
Where

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

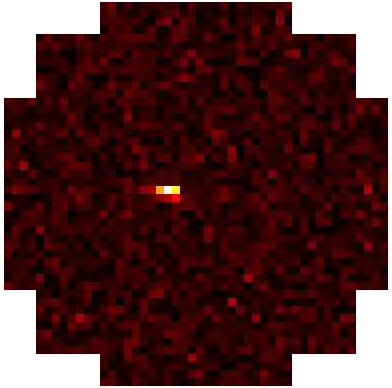
It can be said Shark and scikit-learn implementations of Random Forest seem to have the same performances on this dataset.

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.

Convolutional Neural Network

A five layers **CNN** has been trained with ASTRI calibrated data (*lv1a*), and then used to predict.



Network arch was taken from an example on keras blog to perform classification on dog and cats pictures:
https://blog.keras.io/building-powerful-image-classification-models-using-very-little-data.html

The aim was to fast prototype, but very nice results anyway!

```
In [13]: model.summary()
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
conv2d_1 (Conv2D)            (None, 54, 54, 32)        320
_________________________________________________________________
batch_normalization_1 (Batch (None, 54, 54, 32)        128
_________________________________________________________________
activation_1 (Activation)    (None, 54, 54, 32)        0
_________________________________________________________________
max_pooling2d_1 (MaxPooling2 (None, 27, 27, 32)        0
_________________________________________________________________
conv2d_2 (Conv2D)            (None, 25, 25, 32)        9248
_________________________________________________________________
batch_normalization_2 (Batch (None, 25, 25, 32)        128
_________________________________________________________________
activation_2 (Activation)    (None, 25, 25, 32)        0
_________________________________________________________________
max_pooling2d_2 (MaxPooling2 (None, 12, 12, 32)        0
_________________________________________________________________
conv2d_3 (Conv2D)            (None, 10, 10, 64)        18496
_________________________________________________________________
batch_normalization_3 (Batch (None, 10, 10, 64)        256
_________________________________________________________________
activation_3 (Activation)    (None, 10, 10, 64)        0
_________________________________________________________________
max_pooling2d_3 (MaxPooling2 (None, 5, 5, 64)          0
_________________________________________________________________
flatten_1 (Flatten)          (None, 1600)              0
_________________________________________________________________
dense_1 (Dense)              (None, 64)                102464
_________________________________________________________________
batch_normalization_4 (Batch (None, 64)                256
_________________________________________________________________
activation_4 (Activation)    (None, 64)                0
_________________________________________________________________
dropout_1 (Dropout)          (None, 64)                0
_________________________________________________________________
dense_2 (Dense)              (None, 1)                 65
_________________________________________________________________
batch_normalization_5 (Batch (None, 1)                 4
_________________________________________________________________
activation_6 (Activation)    (None, 1)                 0
=================================================================
Total params: 131,365
Trainable params: 130,979
Non-trainable params: 386
_________________________________________________________________
```

Convolutional Neural Network (keras + tensorflow)

Input

I extracted 16 bit grayscale png images from ASTRI lv1a (calibrated) data; this allowed to store full photo electron equivalent values.

Dataset was splitted this way:

- 140000 events for training (70000 each population, gammas and protons);
- 60000 events for testing (30000 each);
- 200000 events for prediction (200000 each).

Convolutional Neural Network (keras + tensorflow)

General results

- ~~Training (and testing) took less than two hours on a Nvidia Tesla K20 GPU~~
- Prediction took 1 hour (30 epochs, stopped after 11 checking on val_acc, with patience=5 and min_delta=0.001), and gave this performance:
  - val_acc = **0.967**
  - val_loss = 0.096
  - val_rmse = 0.045

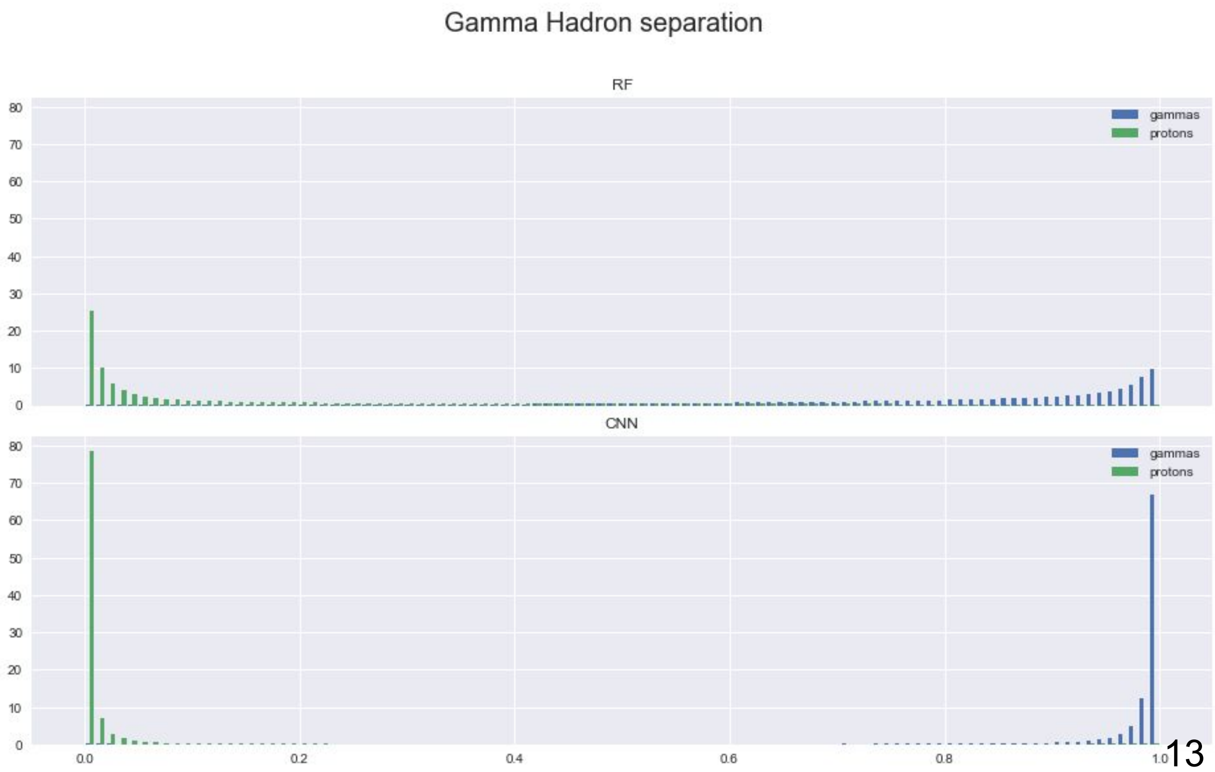
<u>As a first comparison, RF (optimized via GridsearchCV) best oob_scores was **0.86**</u>

Comparison with CNN

Gamma - Hadron separation

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

CNN better separates populations.



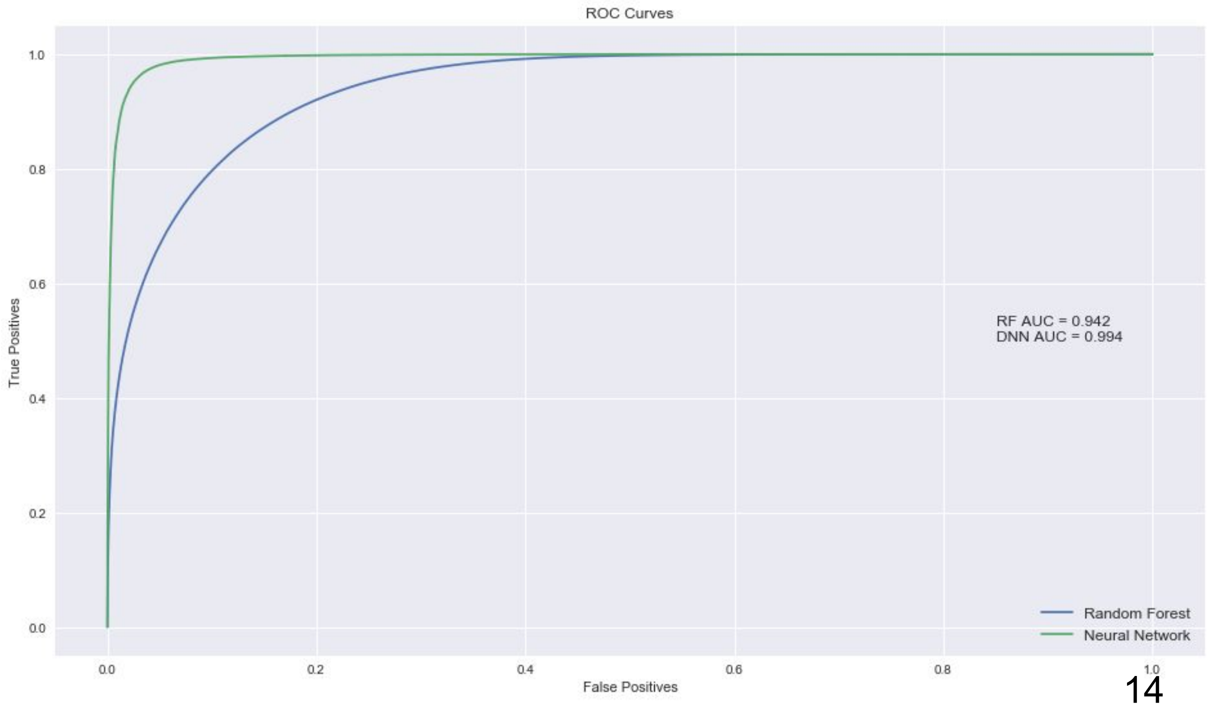
Comparison with CNN

<u>ROC curve and area</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.

CNN performs significantly better.



INAF ISTITUTO NAZIONALE DI ASTROFISICA NATIONAL INSTITUTE FOR ASTROPHYSICS

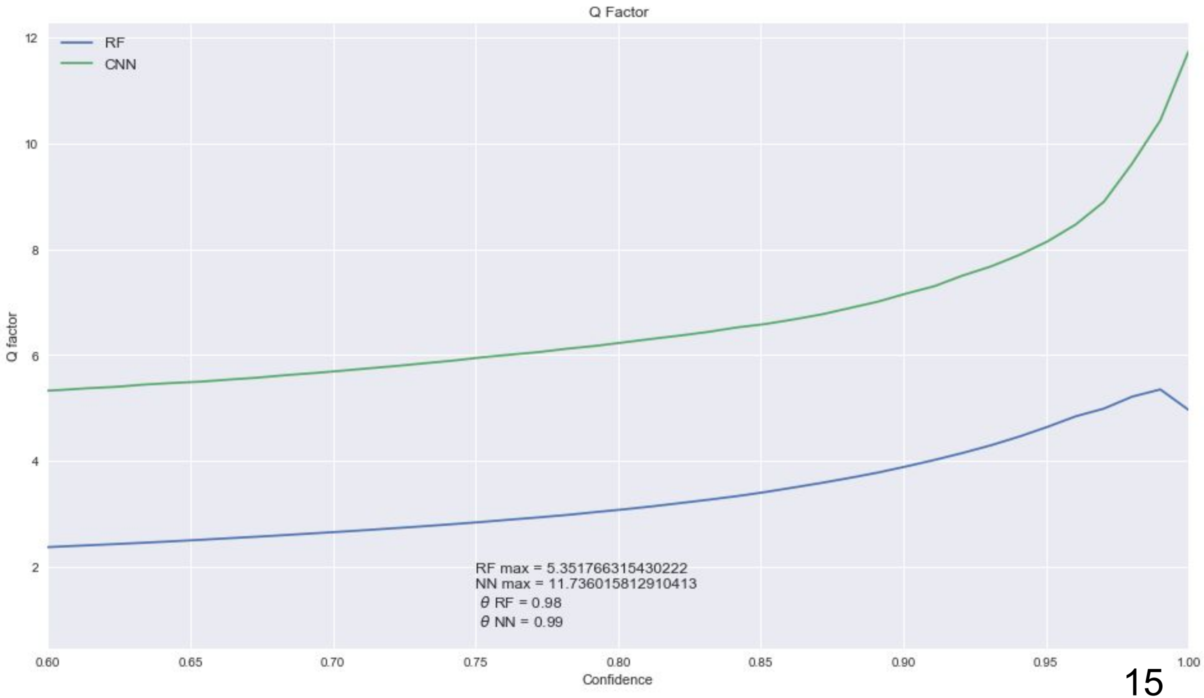
Comparison with CNN

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 ε_γ is the fraction of **well classified gammas**, and ε_p is the fraction of **badly classified protons** (false positives). This is usually plotted against the *acceptance* (or *threshold*) for the probabilities.



Comparison with CNN

Precision - Recall

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

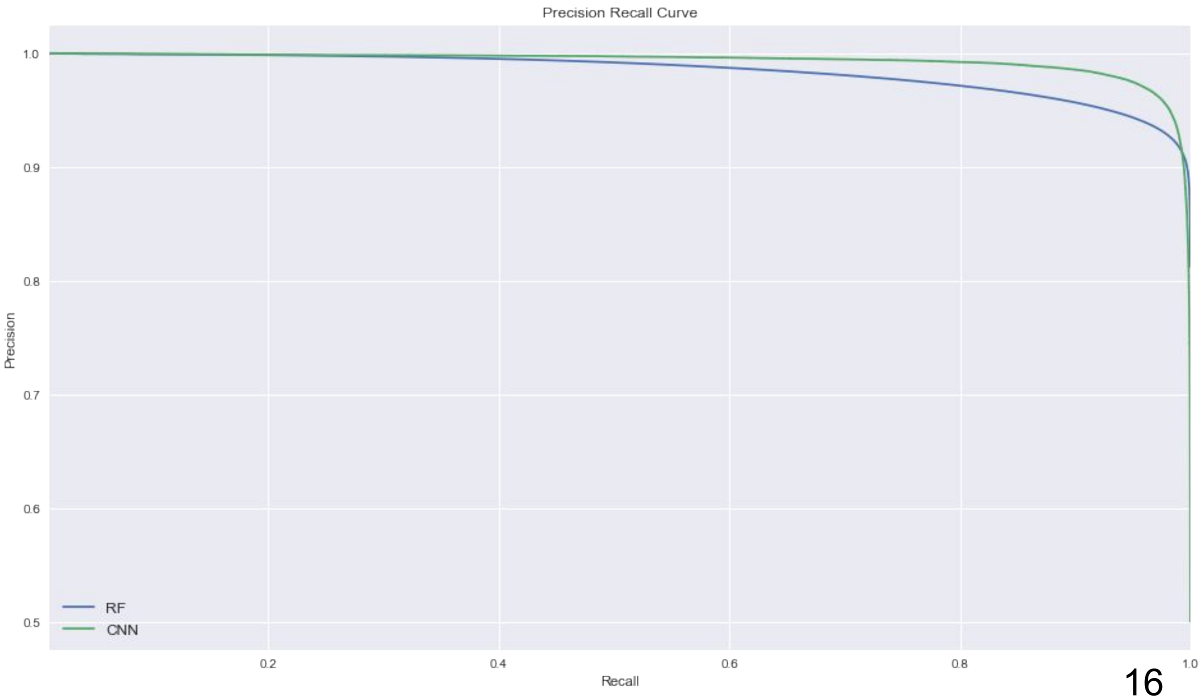
Where

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!)



ISTITUTO NAZIONALE DI ASTROFISICA
NATIONAL INSTITUTE FOR ASTROPHYSICS

Backup slides

ASTRI Data Challenge #1

Main facts on ASTRI DC1:

The events were properly filtered so to follow the experimental energy slope of -2.70, as measured by the BESS Coll. After this procedure, the available statistics resulted in ~4.2×10^{6} triggered events. Since the event rate was calculated to be ~100 Hz, these amount of events corresponded to ~11.6 hours of (single telescope) data taking.

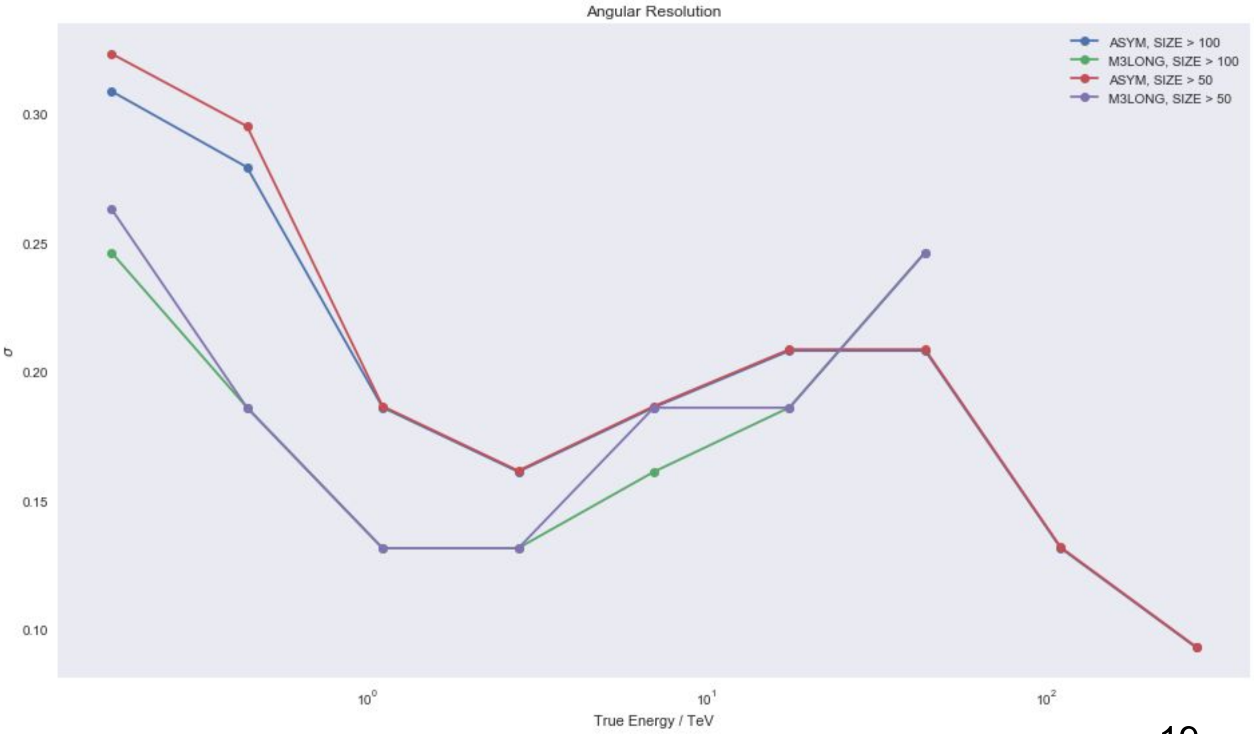
The overall data sample was then split in two subsamples, "ON" and "OFF". We combined the ON sample (~5.8 hours) with a number of randomly selected MC gamma-ray events calibrated to match the flux of the Crab Nebula as measured by the HEGRA Coll. The OFF sample, instead, was slightly reduced from ~5.8 to ~5.5 hours in order to keep a proper amount of independent proton events in their original MC format for the generation of gamma/hadron separation look-up-tables (LUTs)

Future plan: use ON and OFF subsamples for an end to end analysis with CNN.

Details here: https://pos.sissa.it/301/804/pdf, section 5

Angular resolution

We explored performances of two head-tail parameters in angular resolution, finding **M3LONG** performs better than **ASYM** up to ~30 TeV



Confusion Matrix for Sklearn Random Forest Classifier

From wikipedia:

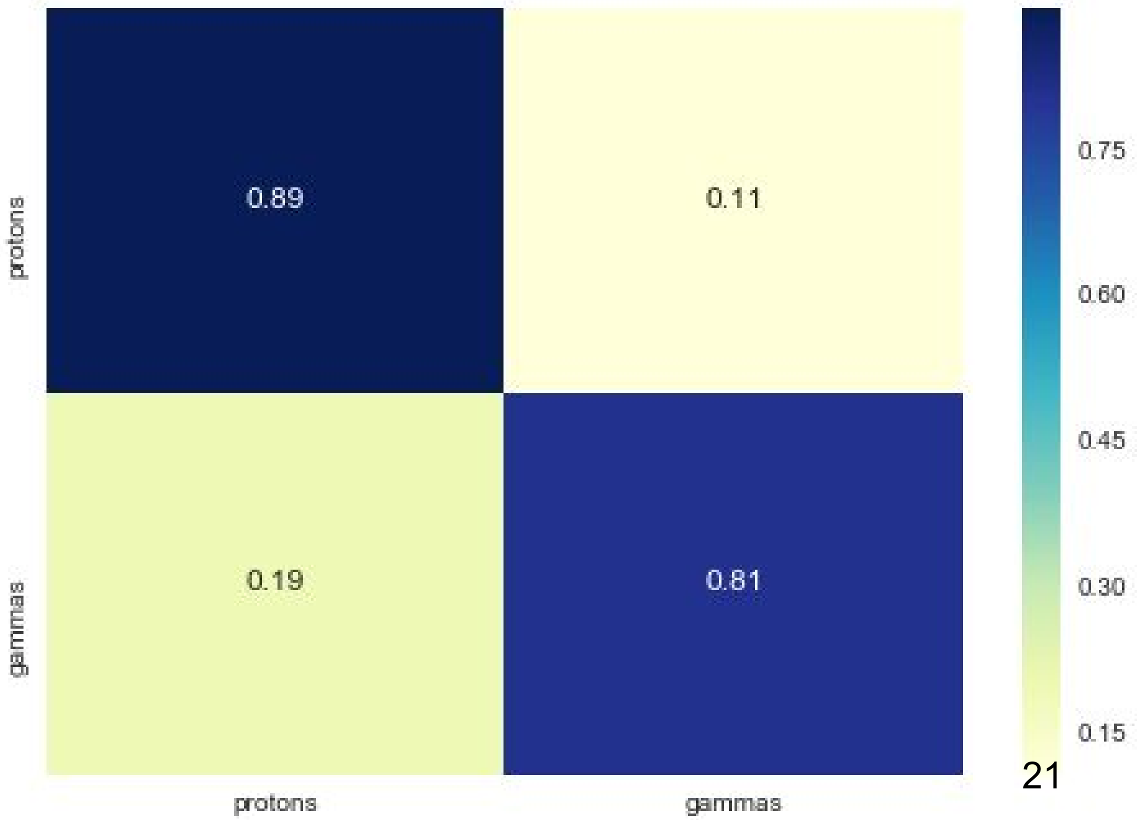
> *In the field of machine learning and specifically the problem of statistical classification, a confusion matrix, also known as an error matrix, is a specific table layout that allows visualization of the performance of an algorithm. The name stems from the fact that* ***it makes it easy to see if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

Sounds useful!

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']

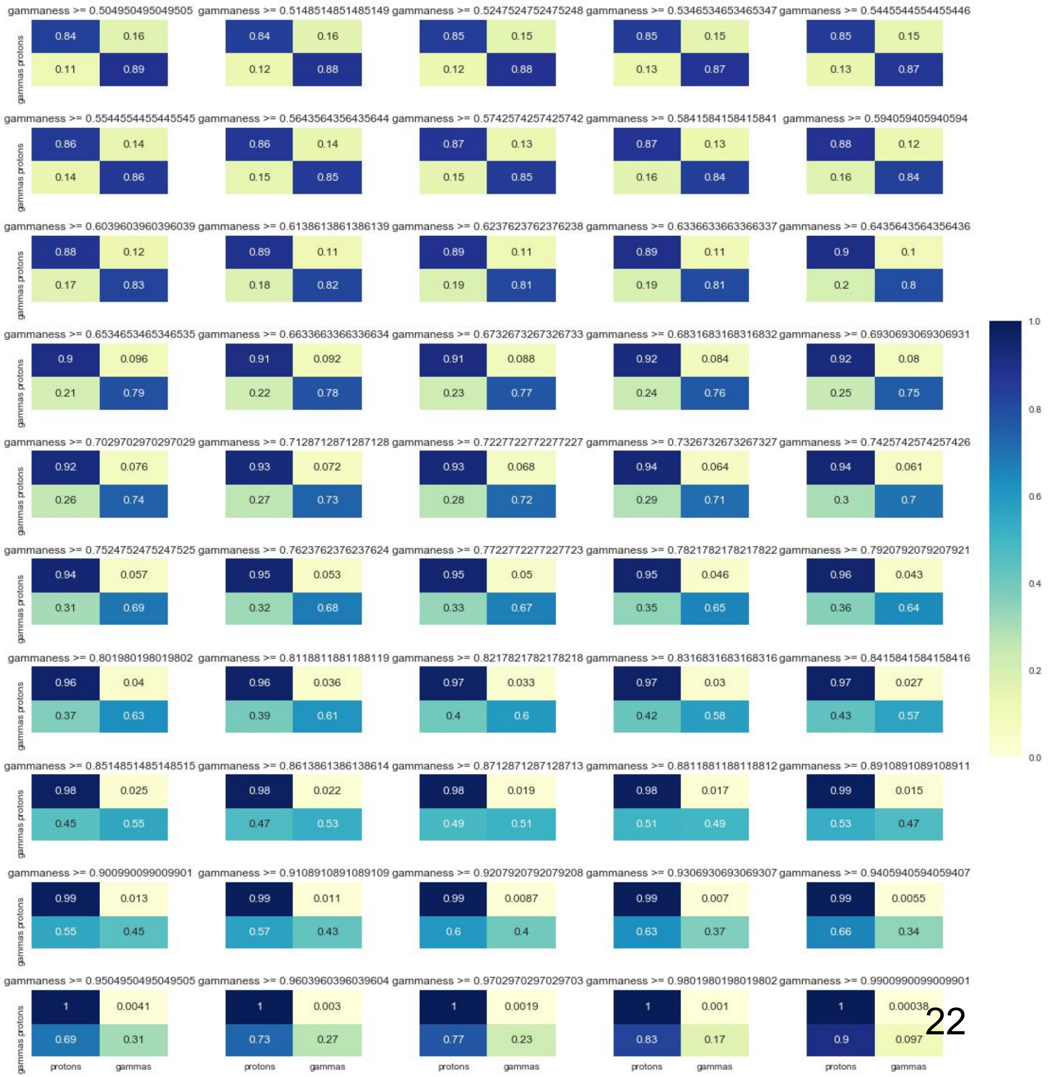


Confusion Matrix

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:

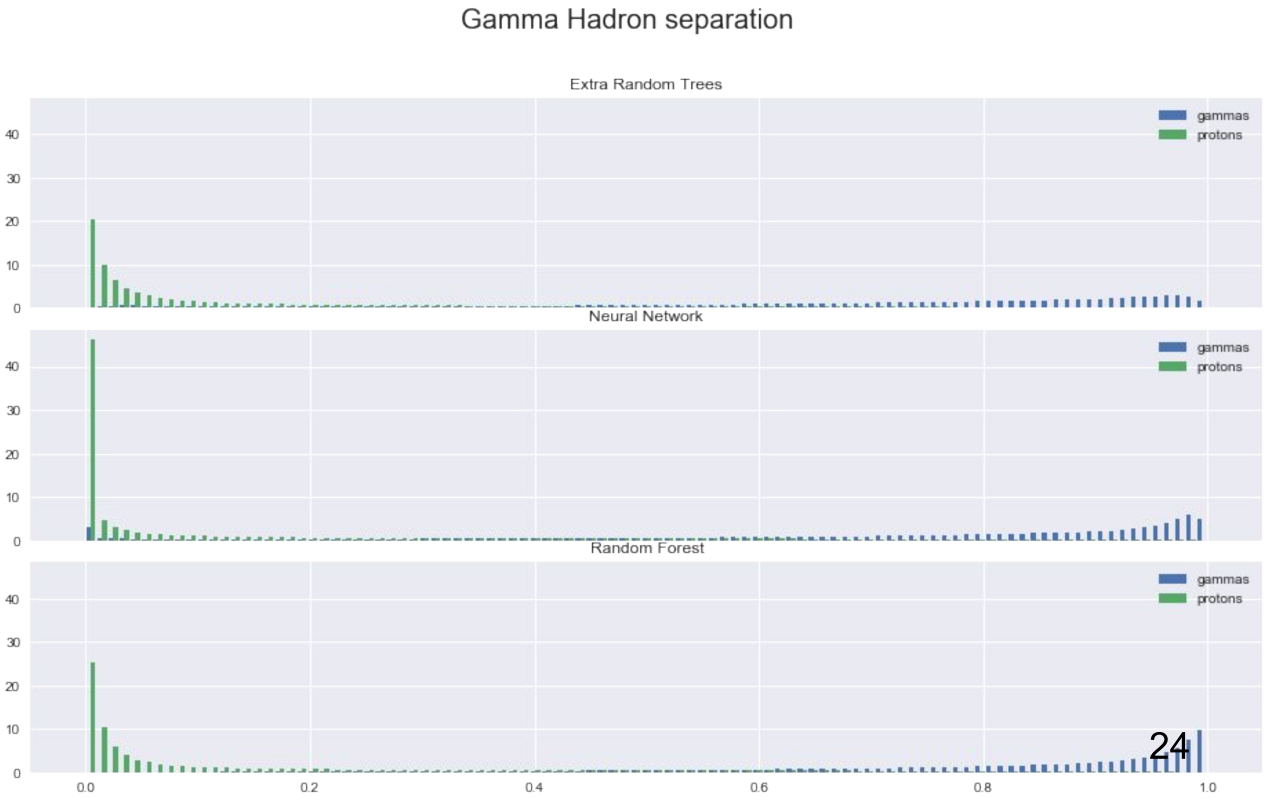
- Multi Layer Perceptron (Neural Network)

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

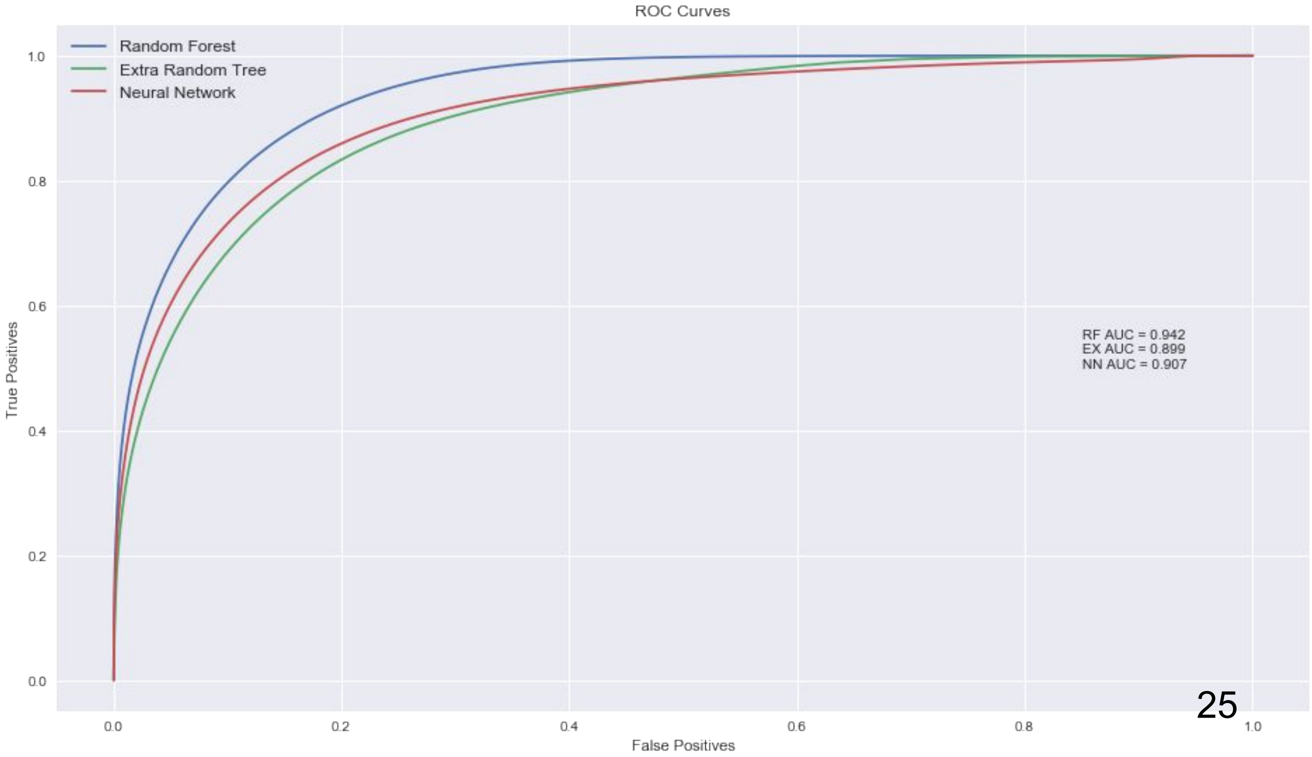
Random Forest leads



Results

ROC curve

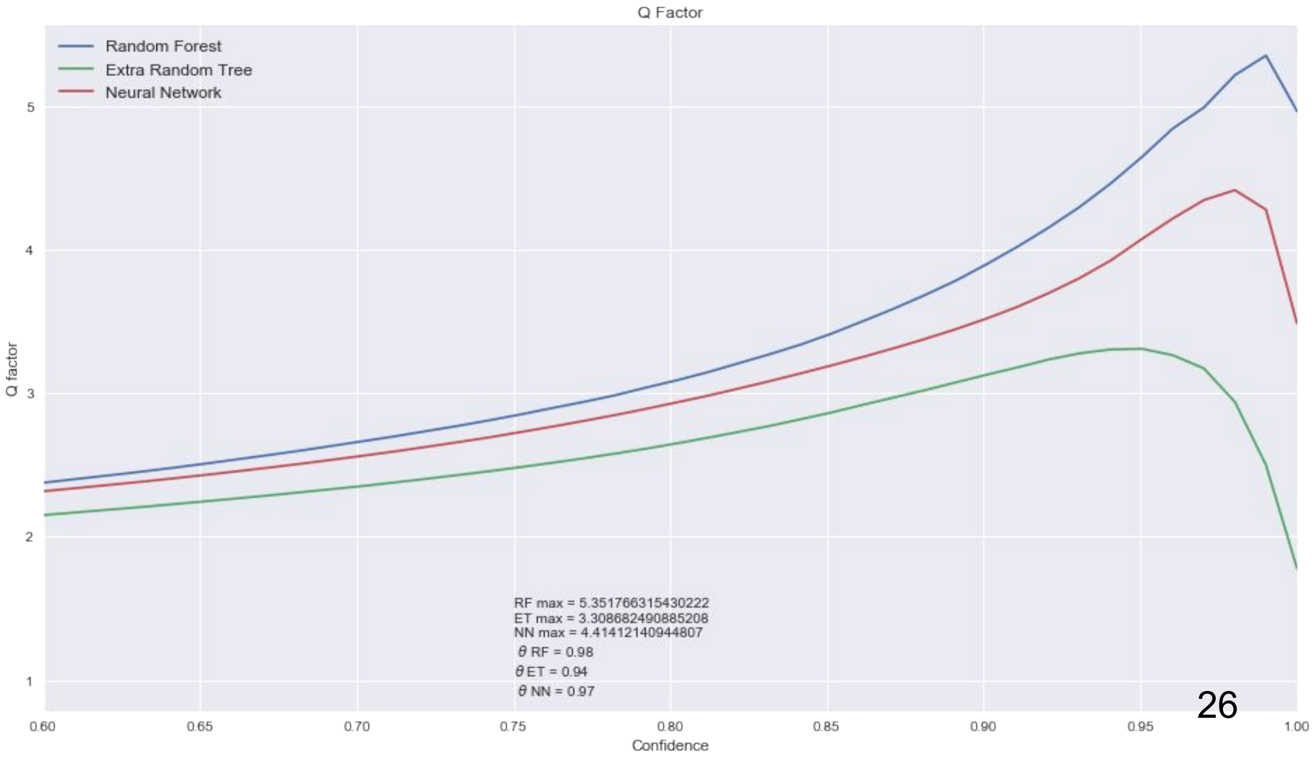
Random Forest leads



Results

<u>Q-factor</u>

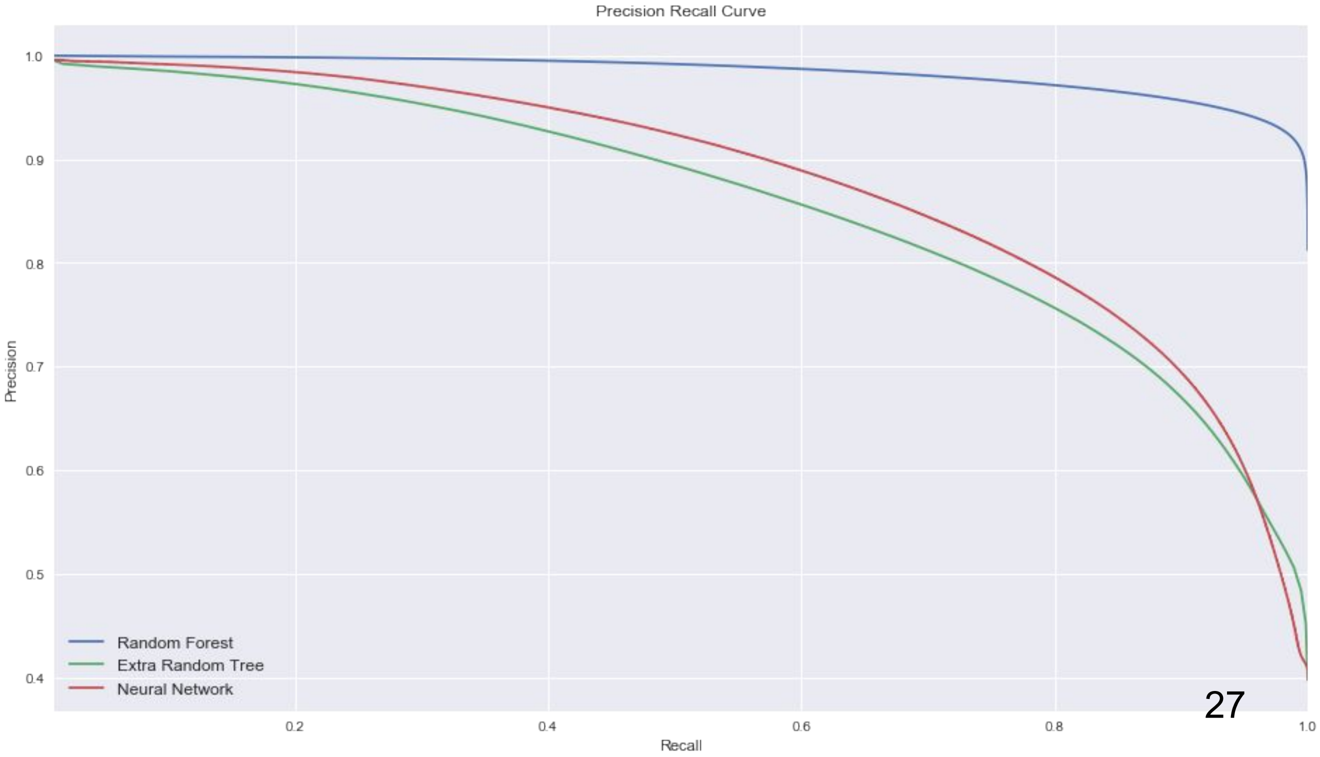
Random Forest leads



Results

Precision Recall curve

Random Forest leads



Comments

Random Forest leads