UNIVERSITÀ DEGLI STUDI DI PADOVA

BACHELOR'S DEGREEE - TESI TRIENNALE

Machine Learning for transient selection in wide-field optical surveys

Author:
Margherita GRESPAN

Supervisor:
Prof. Stefano CIROI
Assistant Supervisor:
Dr. Enrico CAPPELLARO

Corso di Laurea Triennale in Astronomia
Dipartimento di Fisica e Astronomia "Galileo Galilei"

Academic Year 2017/2018

"Big Data is like teenage sex: everyone talks about it, nobody really knows how to do it, everyone thinks everyone else is doing it, so everyone claims they are doing it."

Dan Ariely, Duke University

UNIVERSITÀ DEGLI STUDI DI PADOVA

Abstract

Bachelor Degree in Astronomy
Department of Physics and Astronomy "Galileo Galilei"

Thesis

Machine Learning for transient selection in wide-field optical surveys

by Margherita GRESPAN

Modern-time domain astronomical surveys are able to monitor large swaths of sky registering the variability of celestial sources. I will focus on the search of transients possibility related to gravitational waves (GW) events. Until recently astronomers need to visually select promising candidates from surveys containing a large number of *false positives*, a procedure that is very time consuming. In a wide-field observations the number of detections can easily grow to 10^4 -10^5 and their visual inspection would require days of work. In this thesis I will present a transient evaluation thorough a ranking approach used by the INAF Padua GRAWITA group which strongly reduce the need for visual inspection though it is not yet optimal.
We then explored an alternative approach using a machine learning algorithm (The Random Forest Classifier) providing an automate probabilistic statement about the nature of an astrophysical source as a real transient or as an artifact.
I will describe how I prepared the *training set*, the *test set* and the cross-check of the machine learning detection results. With the internal validation the algorithm secured a missed detection rate of 10% that in the best case of the external validation corresponds to false positives rate of 14%.

UNIVERSITÀ DEGLI STUDI DI PADOVA

Sommario

Corso di Laurea Triennale in Astronomia
Dipartimento di Fisica e Astronomia "Galileo Galilei"

Tesi di Laurea

"Machine Learning": selezione di transienti nelle ricerche a grande campo

di Margherita GRESPAN

Le survey astronomiche di ultima generazione sono in grado di monitorare grandi aree di cielo registrando la variabilitá delle sorgenti celesti. Questa tesi é incentrata sulle modalitá di ricerca degli oggetti transienti che potrebbero avere delle relazioni con l'emissione di onde gravitazionali (GW). Fino a poco tempo fa gli astronomi hanno dovuto classificare visivamente dalle survey, contenenti un grande numero di falsi positivi, i candidati promettenti. Tale procedura richiede un gran dispendio di tempo. In un'osservazione a grande campo il numero di rilevazioni puó arrivare a 10^4 -10^5 oggetti e la loro ispezione visuale puó richiedere giorni di lavoro.
In questa tesi presenteró la valutazione degli oggetti transienti attraverso il metodo del ranking, usato dal gruppo GRAWITA di Padova, che riduce in maniera sostanziale il bisogno di un'ispezione visiva, nonostante il risultato ottenuto non sia ancora ottimale.
Allo scopo di velocizzare la valutazione di questi oggetti ho utilizzato un algoritmo di machine learning (il Random Forest Classifier) che fornisce una classificazione probabilistica automatica sulla natura delle sorgenti astrofisiche, determinando se queste siano oggetti reali oppure artefatti.
Descriveró come ho preparato il training set, il test set e il cross check delle rivelazioni del machine learning. Con la validazione interna l'algoritmo ha assicurato un numero di candidati non rivelati (MD) del 10% che, nel miglior caso, nella validazione esterna é corrisposto ad un numero di falsi positivi (FP) del 14%.

Acknowledgements

Dopo tre anni di esami, lezioni, giornate passate in aula studio ecco che il mio ultimo sforzo all'interno della triennale si sta portando a termine. Sono stati tre anni difficili, emozionanti, stancanti, ma anche felici e pieni di apprendimento. Non credevo di poter cambiare così tanto, tutte le esperienze fatte all'interno dell'Università di Padova mi hanno fatto crescere come studente, come futura lavoratrice ma soprattutto come persona.

Vorrei prima di tutto ringraziare il mio correlatore Enrico Cappellaro che nella preparazione e stesura di questo lavoro mi ha seguito costantemente, dimostrandosi sempre disponibile e capace, soprattutto di farmi capire molti concetti per me nuovi. Grazie anche a Sheng che mi ha aiutato svariate volte sempre con il sorriso. Ringrazio anche il mio relatore Stefano Ciroi, è solo grazie a Lei se ho avuto l'occasione di lavorare con delle persone così. Grazie a tutti voi, senza il vostro contributo questa tesi non sarebbe stata possibile.

Un ringraziamento va anche ai miei genitori, mi siete sempre stati vicino e con i vostri contributi mi avete permesso di vivere serenamente questi tre anni da fuori sede. Ringrazio tutta la mia famiglia, grazie di avermi incoraggiato in questa esperienza che a me pareva impossibile.

É arrivato il momento di ringramziare le persone che hanno vissuto questi anni in stretto contatto con la mia persona che a volte poteva diventare insopportabile. Vi siete dovuti subire ogni mio singolo cambio di umore e tutte le mie inutili filippiche. Mi riferisco soprattuto ai miei coinquilini e ai grandi frequentatori di casa Dessole. E' stato un anno di convivenza tanto difficile quanto divertente, assieme siamo riusciti a invogliarci l'un l'altro e a vincere assieme questa "corsa all'astronomo". In particolare grazie Marco e Alberto, mi siete stati vicini più di qualunque altro.

Ultimi, ma non ultimi ringrazio i miei amici fuori sede. Mi riferisco a voi tre che siete stati in questi ultimi anni sparsi in giro per l'Italia. Sapere che la nostra amicizia è rimasta la stessa anche se noi stiamo cambiando di giorno in giorno mi rassicura. Il Natale assieme fa capire quanto siamo una vera famiglia.

Grazie a tutti,

Margherita

Contents

Abstract **iii**

Sommario **v**

Acknowledgements **vii**

1 Introduction **1**
1.1 The Gravitational Waves 1
1.1.1 Importance of GWs 2
1.2 The working group — GRAWITA 3
1.3 The new generation surveys and transients 3

2 The GRAWITA Data Processing **5**
2.1 Observations 5
2.1.1 VSTTube 5
The `diff-pipe` pipeline 5
The difference of images 6
`SExtractor` 7
2.1.2 The Ranking approach 7
2.1.3 Stamps 8
2.2 The visual classification 8
2.2.1 Candidates informations 9
Panel 2.6 description 9
2.2.2 Classification 11
2.2.3 Artefacts in Difference Images 11
2.2.4 Results 12

3 The training set **17**
3.1 A brief introduction to Machine Learning 17
3.1.1 Decision Trees 18
Random Forest 19
3.1.2 Classification 20
3.1.3 Evaluating Models 20
The Confusion Matrix 20
Accuracy 21
3.2 Definition of the training set 21
The `read_negativi` function 21
The `creazione_tabelle` function 23
3.2.1 The stamps creation — `produzione_stamps` function 23
3.3 The Random Forest implementation 24
3.3.1 The `machine_learning` function 24
3.3.2 `Fit` results 25
Testing dependence on stamp size 25

Test of effect of estimator number 25

4 Application of the classifier to the GW170814 survey **29**

4.1 External check 29

4.1.1 Classifier implementation 29

4.1.2 Stamps preparation 30

4.1.3 `Clf` implementation — The `model_implementation` function . . 30

Survey with no ranking limitations 30

Survey with ranking limitations 31

4.1.4 Cross-check — The `cross_check` function 31

4.1.5 Further analysis 33

Limitation in magnitude 33

Changing of the *hypothesis* value 35

4.2 Conclusions 35

A Python Scripts **37**

A.1 Script for `clf` creation 37

A.2 Script for plots creation 43

A.3 Script for model evaluation 44

B Tables **49**

Bibliography **51**

List of Figures

1.1 An artist's impression of gravitational waves generated by binary neutron stars. Credits: R. Hurt/Caltech-JPL . . . 1
1.2 Two-dimensional illustration of how mass in the Universe distorts space-time. It can be easily seen how the space time is represented as a stretched sheet of rubber deformed by the mass of the body. Credits: NASA . . . 2
1.3 Ground-based facilities available for GRAWITA. Credits: Brocato . . . 4

2.1 Scheme of the images analysis. . . . 6
2.2 Example of negative source. The reference image is the one on the right and the difference is in the middle. . . . 6
2.3 Example of difference images. . . . 8
2.4 Screen of the GRAWITA site. . . . 9
2.5 A zoom in of the possible candidates classifications. There are the macrocategories Real/Bogus and even a more specific one. For my purpose the Real or Bogus classification is enough. . . . 10
2.6 A zoom in of the "header" of the interface. The name of the survey exposed is G211117, is a consequential number indicating a survey. When astronomers are sure that a gravitational trigger is detected in a survey they change the name with the initial letters GW and the date. Basically G211117 and GW151226 are the same thing. . . . 11
2.7 An example of SN. . . . 12
2.8 An example of AGN. . . . 12
2.9 An example of VAR object, i.e. objects showing a record of long-term variability. This one is saved in the catalog as an eclipsing binaries. . . . 13
2.10 An example of bright star. . . . 13
2.11 An example of dipole; they are residuals with roughly equal amounts of positive and negative flux, caused by errors in image registration. . . . 13
2.12 bogus . . . 14
2.13 CCD error. . . . 14

3.1 A schematic flowchart of how supervised machine learning works. In 2 is presented the data pre-processing, in this chapter is described the definition of the training set and in the next one (4) the algorithm selection, the training and the evaluation. Credits: Kotsiantis, 2007. . . . 18
3.2 A schematic decision trees example. Credits: Mayur Kulkarnix . . . 19
3.3 Preparation of the training set script flowchart. . . . 22
3.4 Stamp creation. . . . 23
3.5 Example of a stamp. . . . 23
3.6 Histogram that shows the RF predictions. The blue dotted line is the threshold at MD=10%. . . . 24
3.7 Missed Detection Rate and False Positive Rate plot. The dotted line indicate the selected threshold at 0.35, it coincides with a MD=10%. . . . 24

3.8 ROC curves of different stamp sizes. 26
3.9 ROC curves made by `fit` with a different number of estimators. 26

4.1 Stamps of the 9 non detected objects. 30
4.2 Stamp examples of 4 detected objects. 30
4.3 Histogram of the detected sources of all scores objects in GW170814 survey. 31
4.4 Detected objects by the `clf` in GW170814. The dotted blue line is at h=0.35. 32
4.5 Magnitude trend of the GW151226 survey. The bins have range 1 mag and are from the lower magnitude found in the survey from the higher. 34
4.6 Limited magnitude trend in the GW151226 survey. The bins have a range of 1 mag and they start from 20 mag. 34
4.7 Histogram that shows the trend of the RF predictions for the survey with limited magnitude. The bins have a range of 0.025. 34
4.8 ROC curve for the survey with limited magnitude. The blue dotted line is at h=0.30 that correspond at MD=10%. 34
4.9 Trend of the objects in the GW170814 survey with a `clf` trained in a survey limited in magnitude. The dotted blue line is at h=0.30. 35

Chapter 1

Introduction

Gravitational waves (GWs) have been predicted by theory of general relativity (Einstein 1916) as a perturbation of space-time metric. In many cases, the events that produce GWs are expected to produce also electromagnetic radiation. My work is focused in the search of optical counterparts search of GW signals.
Given the typical size of the GW localization error-box, searching for counterpart requires telescopes with large FoV, yet the new generation telescopes are able to cover a large span of sky i.e. thousand of square degrees (Singer et al., 2014).The larger is the sky region, the higher is the number of detected celestial objects and hence the time taken to find those which are related to GWs.
In this thesis I evaluate the efficiency of the `Random Forest` method for the selection of transient objects.

1.1 The Gravitational Waves

This thesis is focused in the detection of transients that are possible EM counterparts of gravitational waves, exploiting wide-field optical surveys.
Gravitational waves are "ripples" in the space-time caused by some of the most catastrophic and energetic processes in the Universe.
Einstein's general relativity theory showes that massive accelerating objects (such as neutron stars or black holes orbiting around each other) create "waves" perturbating the space-time fabric radiating from the source.
GWs can be visualized as waves on the surface of the water moving away from a stone thrown into a pond. These ripples travel at the speed of light through the Universe, carrying with them informations about their cataclysmic origins as well

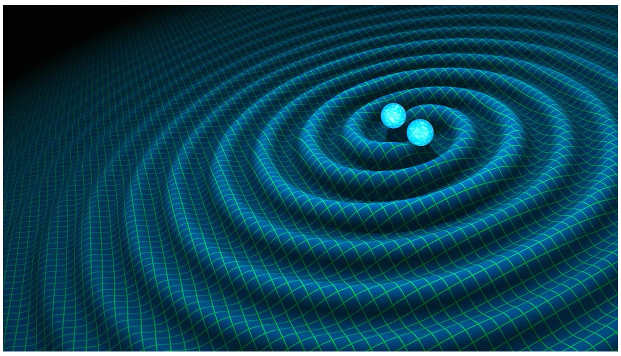

FIGURE 1.1: An artist's impression of gravitational waves generated by binary neutron stars. Credits: R. Hurt/Caltech-JPL

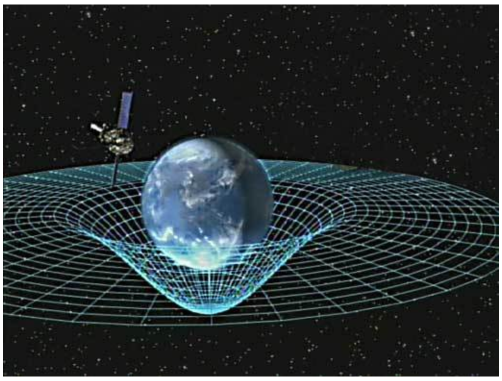

FIGURE 1.2: Two-dimensional illustration of how mass in the Universe distorts space-time.
It can be easily seen how the space time is represented as a stretched sheet of rubber deformed by the mass of the body.
Credits: NASA

as invaluable clues to the nature of gravity itself (*What are Gravitational Waves?*).

GWs are produced by a large variety of astrophysical phenomena. For example from coalescence of Binary systems of compact objects like two Neutron Stars (BNS), of a Neutron Star and a stellar-mass Black Hole (NSBH) or two black holes (BBH). Also can originate from the slightly wobbly rotation of neutron stars that are not perfect spheres but also from the same event that created the Universe (the Big Bang) create GWs that in future it may be possible to detect.
Current detectors are indeed most sensitive to GWs from nearby compact object coalescence.

1.1.1 Importance of GWs

Gravitational radiation has been detected indirectly since the seventies in the context of binary systems. Only one century after Einstein's theoretical prediction an international collaboration of scientists (LIGO Scientific Collaboration and Virgo Collaboration) reported the first direct observation of gravitational waves.
GWs are detected by the two most sensitive interferometers: Laser Interferometer Gravitational Wave Observatory LIGO (Aasi et al., 2015) and european advanced VIRGO (Acernese et al., 2014).

Gravitational waves carry information about their dramatic origin and about the nature of gravity that cannot otherwise be obtained.
In the LIGO observations (Abbott et al., 2016) the ripples detected were occouring from the fraction of second when two black holes collided into each other at nearly one-half of the speed of light and formed a single more massive black hole, converting a portion of the combined black holes' mass to energy. This energy was emitted as a final strong burst of gravitational waves[1].
Detecting this type of events allows us to observe the Universe in a way never before possible. It gives us a deeper understanding of cataclysmic events. They give us information about objects like black holes otherwise invisible in the EM range (*Why Detect Them?*).

[1]https://www.jpl.nasa.gov/news/news.php?feature=5137

1.2 The working group — GRAWITA

For my work I used the data of the GRAvitational Wave INAF TeAm (GRAWITA[2]), an Istituto Nazionale di Astrofisica (INAF) collaboration. The team is performing photometric and spectroscopic follow-up in the EM domain of the GW trigger, in particular in the optical/near infrared (NIR).
GRAWITA has the access to conspicuous ground-based facilities like VST, VLT, LBT, TNG, REM (Fig. 1.3.)
The GRAWITA team based in Padova developed a transients identification pipeline (`SUDARE` Cappellaro et al., 2015) based on images difference method that rely on different astronomical tools e.g. `Hotpants, Sextractor, astropy, pyraf, Mysql`, etc.
The candidates list is extracted from the subtracted images (using the `SExtractor` tool, Bertin and Arnouts, 1996). A ranking algorithm instead help astronomers to decrease the number of candidates that needs visually check (see Chapter 2).

1.3 The new generation surveys and transients

Depending on the process of compact object merging, GWs event can be accompanied by EM radiation emission. GWs observations would help to know a lot more about the cosmos. After the first detection in 2015 by the LIGO interferometer (Abbott et al., 2016)
The potential gain of detecting the EM counterparts of GWs motivated a world-wide effort of the whole astronomical community, employing many telescopes and instruments, ground and space-based, ranging from high energy through optical to radio wavelengths, each contributing the monitoring of a portion of the sky localization area with different depth and cadence[3] (Brocato et al., 2017). In GWs follow-up the optical counterparts are transient sources.

Optical transients, our objects of interest, are the astronomical sources showing a significant change in brightness variation, either raising or declining flux, that can be associated to extra-galactic events.
The most common detection procedure is based on subtracting images. The field of view is observed in different epochs and then every image is compared with a reference one that could be the older or more recent that those to be searched. In an ideal case the template image is taken before the actual search epochs. Data acquisition is described in Ch.2.
This procedure creates a difference image that in the ideal case is just a pure noise image but for real transients such as a supernovaw, asteroids or variable stars. In a real case we find instead a large number of spurious sources (bogus) due to defects in the detectors, poorly removed cosmic rays contamination, faint residual of bright star subtraction, small misalignment of the images, etc.
In a wide-field observations the number of detections can easily grow to 10^4 -10^5. That leads the astronomer to search for quick methods of transients classification.
The GRAWITA team (see subsection 1.2) uses Ranking approach (described in 3). This approach however leaves a large fraction of events still requiring visual inspection. It is interesting to test a different approach to object classification based on

[2]https://www.grawita.inaf.it/ gwpadova/home.php
[3]LIGO-VIRGO observing plans at https://www.ligo.org/scientists/GWEMalerts.php

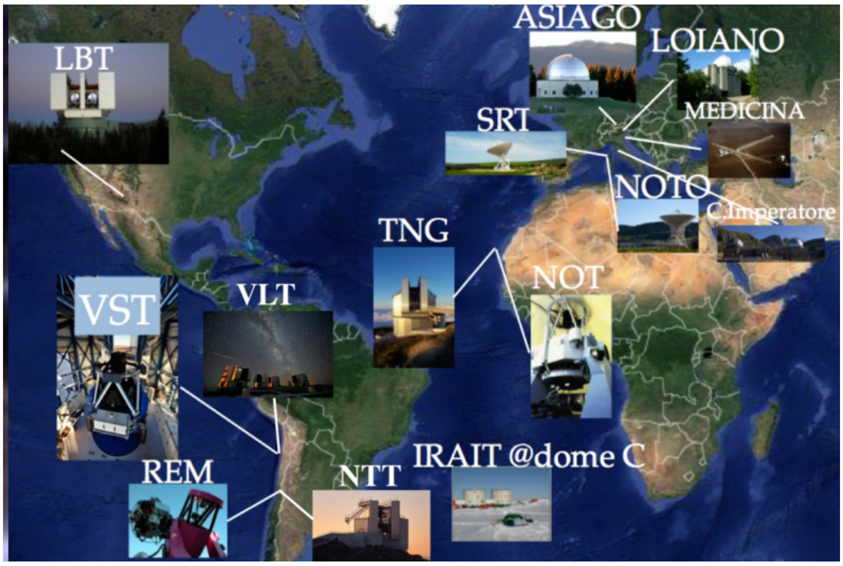

FIGURE 1.3: Ground-based facilities available for GRAWITA. Credits: Brocato

Machine Learning.
There is also interesting in view of the next-generation surveys, such as GAIA, the Dark Energy Source (DES), LSST and SKA will lead to an era of exascale astronomy requiring new machine learning and statistical interference tools (Buisson et al., 2015).
Machine Learning needs a training set from which learn to how to make transients classification, the preparation of the training set is explained in Chapter 3. Finally in Chapter 4 I summarize the Random Forest approach results and I draw my conclusions.

Chapter 2

The GRAWITA Data Processing

In this work I used the observations of GW151226 already reduced, calibrated and the extracted stamps for each candidate. Hereafter I will briefly describe the image acquisition and reduction method made by the GRAWITA team.

2.1 Observations

The VLT Survey Telescope (VST; Capaccioli and Schipani, 2011) is located at Cerro Paranal Observatory; it is a joint venture between the European Southern Observatory (ESO) and the INAF-Osservatorio Astronomico di Capodimonte (OAC) in Napoli (Cicco et al., 2015). The telescope has a size of 2.6 m. Tha camera (Omegacam) has a field of view of 1 sq deg with a pixel size of 0.21 arcsec /pixel. In 40 second exposure time (we use a short exposure time to be able to monitor a large area) we can reach mag ~ 22 with S/N ~ 3.

2.1.1 VSTTube

After acquisition, the raw images are mirrored to ESO data archive, then transferred through an automatic procedure from ESO Headquarters to the VST Data Center in Naples (Brocato et al., 2017).
The first part of the data reduction is performed using the `VSTTube` pipeline (Grado et al., 2012), developed exclusively for the VST-OmegaCAM data. It performs in mainly tree steps: prereduction, astrometric and photometric calibration, mosaic production.
First `VSTTube` reduces individuals exposure and then combines single epochs images to produce the final mosaic.VST tube implements the instrumental signature removal: overscan, bias, flat-field correction, CCD gain harmonization of the 32 CCDs, illumination correction and, for the i band, defringing (Wright et al., 2015).
The dithered images for one epoch are median averaged, producing one stacked image for each pointing. The pipeline also creates a weight pixel mask tracking, for each pixel, the number of dithered exposures contributing to the combined image after accounting for CCD gaps, bad pixels and cosmic rays rejection.
The pipeline can also create a stacked image by combining exposures taken at different time, in a given filter, with the best image quality (Cappellaro et al., 2015).

The `diff-pipe` pipeline

The GRAWITA team usually uses two independent but equivalent procedures for transient detection:

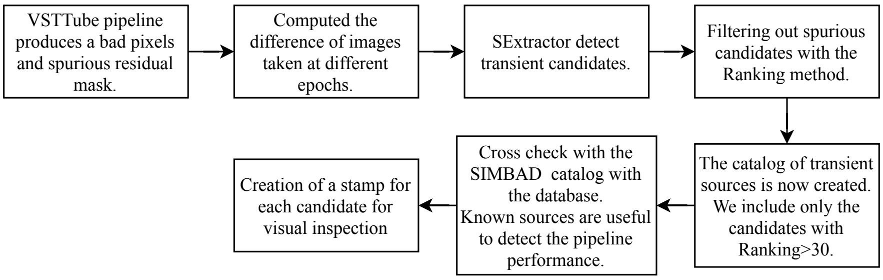

FIGURE 2.1: Scheme of the images analysis.

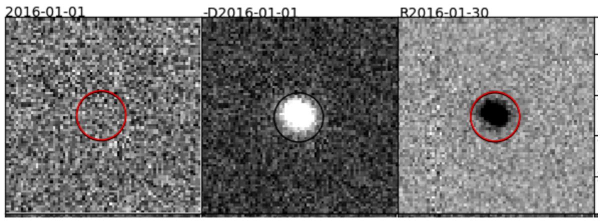

FIGURE 2.2: Example of negative source. The reference image is the one on the right and the difference is in the middle.

1. the photometric pipeline `ph-pipe` (for further information see (Brocato et al., 2017))
2. the images difference pipeline `diff-pipe` whose results I used for the ensemble learning method `Random Forest` implementation.

The `diff-pipe` is based on the analysis of the images subtraction using the approach of the supernova search program (Botticella et al., 2016), (Cappellaro et al., 2015).
Basically the pipeline includes `Python` scripts with specialized tools for the data analysis e.g. `SExtractor`[1] (Bertin and Arnouts, 1996) and `topcat`[2]/`stilts`[3]. The first one is dedicated to sources extraction and the latter to catalogs handling.
Fig.2.1 shows a simple scheme of what `diff-pipe` does.
I will highlight the main points in the following paragraphs.

The difference of images

Having observations at the different epochs, the most intuitive way of proceeding for transient search is images subtraction.
In this first step different epochs observations are compared with a reference one usually taken before the actual search epochs. Not always a template image is available so is left to the user the decision to took the last or the first observed epoch as reference.
I used for this thesis the GW151226 survey where the reference image was the last.
We searched both for positive sources (sources that at the latest epoch disappeared or are fainter than in the previous epochs) or negative ones (objects that where brighter at the latest epoch, ex. fig. 2.2) (Brocato et al., 2017).

[1]https://www.astromatic.net/software/sextractor
[2]http://www.star.bris.ac.uk/ mbt/topcat/
[3]http://www.star.bris.ac.uk/ mbt/stilts/

SExtractor

SExtractor is a software for sources extraction, that we use to detect sources in the difference images. It is able to work very rapidly in large images.
The complete analysis of an image is done in six steps: estimation of the sky background, thresholding, deblending, filtering of the detections, photometry, and star/-galaxy separation (Bertin and Arnouts, 1996).

2.1.2 The Ranking approach

As mentioned before the list of objects produced by the SExtractor algorithm contains a large number of spurious objects due to different factors related to the images creation: improper flux-scalings, incorrect PSF convolution, mis-alignment of the images etc.
In order to filter out bogus from the candidates list GRAWITA currently uses a ranking approach (Cappellaro et al., 2015) which assigns a score to every candidate and based on the value of that score an object is classified. The score is increased/decreased from an initial value (equal for all the candidates) on the basis of different measured parameters[4] (described in Brocato et al., 2017) in particular:

- FWHM
- ISOAREA
- FLUX_RADIUS
- CLASS_STAR
- Proximity to bright sources (penalized)
- Proximity to galaxies (promoted)
- Ratio positive/negative pixels in the defined aperture (penalized if it's below a specific threshold)

In the end of the ranking selection a threshold is chosen above which the candidates are visually inspected. Usually under score 30 all the objects are bogus i.e. non transient object but artefacts.
In my case I used as threshold a score=30.
The use of the Ranking reduces the order of candidates to manually classify. For example in the GW151226 the number of objects selected with a score greater than 30 is about 6300; with a score greater than -60 are 190042 and without score selection are 352268.
Those numbers help to understand how much this approach reduces the quantity of candidates but still a manual classification of 6000 objects requires a lot of time (we can estimate that the time needed for visual inspection is about 5 sec per object) to astronomers and does not permit a real time classification and a quick response to GW triggers.
In the following paragraph I will describe in details how manual classification works.

[4]for further explanation see the complete guide: https://www.astromatic.net/pubsvn/software/sextractor/trunk/doc/sextra

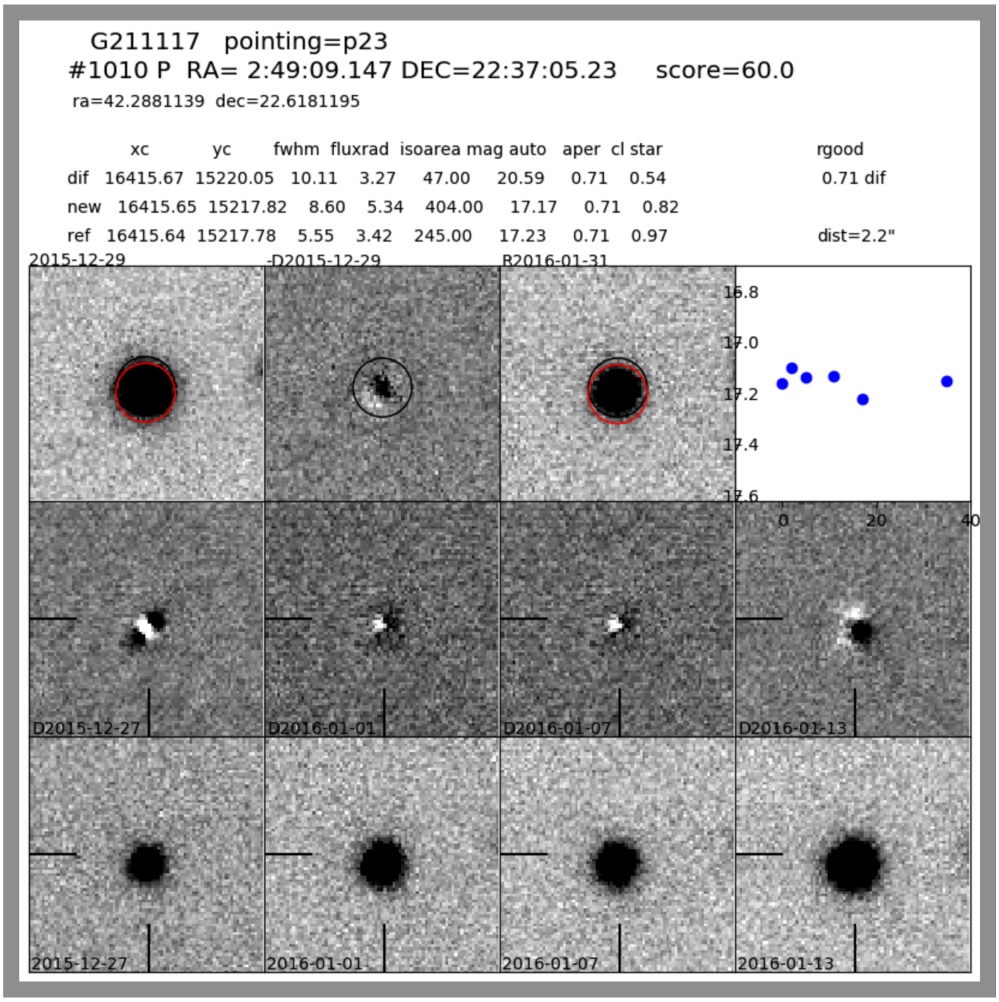

FIGURE 2.3: Example of difference images.

2.1.3 Stamps

Once we have the list of transient candidates it is possible to visualize every object in a stamp i.e. a cropped square containing the transient. In order to make the classification easier the GRAWITA group created an interface like Fig.2.3.
For every candidate it is showed the search and the subtracted image for every epoch and the reference one.
Now is possible to start the eyeballing process. The user based on experience or on some measured parameters (see. following chapters) has to decide if the represented object is real or bogus. Here is where my effort starts.

2.2 The visual classification

In the GRAWITA site [5] there is an interactive interface that permits the astronomer to visual inspect candidates displaying all useful information.
Fig.**??** shows the interface otuput.

[5] https://www.grawita.inaf.it/g̃wpadova/

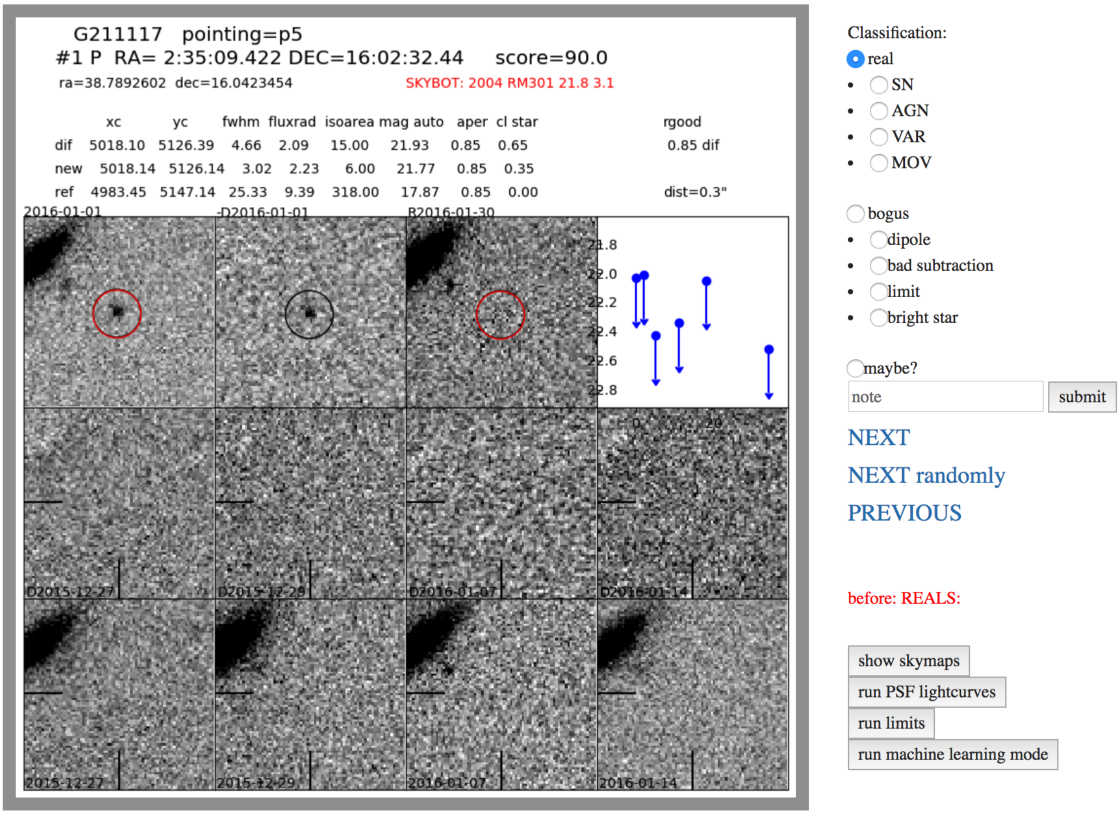

FIGURE 2.4: Screen of the GRAWITA site.

2.2.1 Candidates informations

Panel 2.6 description

In the upper left of Fig. **??** (see Fig. 2.6) there are:

- *GW151226* is the LIGO name of the GW event where:
  - *26* is the day of the first observation,
  - *12* is the month,
  - *15* is the year (2015)
- *pointing* is the VST pointing where the candidate has been found
- *#1* is the number of the detected object. It permits to univocally identify a candidate. In Chapter 4 it will be called `id`.
- *P* means that the image difference is positive.
  Since we used as reference the last image taken if the object has increased in brightness the difference will be positive, otherwise it will be negative.
  In Chapter 4 it will be identified under the name `search`.
- *RA* is the right ascension coordinate.
- *DEC* is the declination.
- *score* is the score that the object "achieved" after the `Ranking method`. Usually 90 is the maximum score and many (30%) of the candidates with this value are real transients. Under 30 all the objects are Bogus.

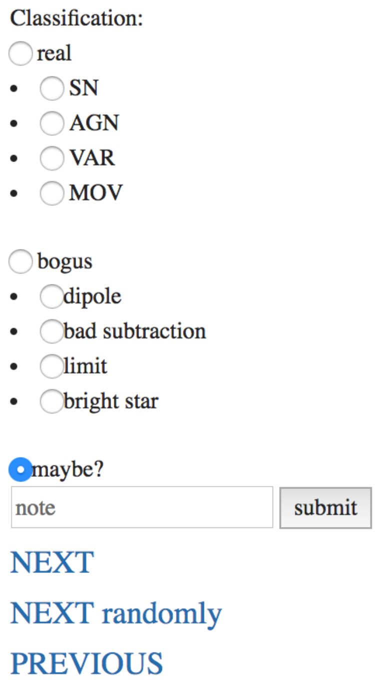

FIGURE 2.5: A zoom in of the possible candidates classifications. There are the macrocategories Real/Bogus and even a more specific one. For my purpose the Real or Bogus classification is enough.

- with a cross-check with public source catalogues is possible to find if the object is already known:
  - *SKYBOT* [6] permits to identify Solar System objects, usually it means that the object visualized is an asteroid.
  - *NED* [7] is a database of galaxies and in general extragalactic objects.
  - *SIMBAD* [8] provides a basic data and measurements database for extrasolar objects.

Other informations are written immediately above the stamps:

- *xc* is the abscissa coordinate of the centre of the target
- *yc* is the vertical axis coordinate of the centre of the target
- *fwhm* is the full width half maximum of the candidate measured by `SExtractor`
- *fluxrad* is the flux
- *isoarea* number of pixels with values exceeding some predefined threshold
- *mag auto* is the apparent magnitude of the object
- *aper* aperture magnitude of the transient

[6] http://vo.imcce.fr/webservices/skybot/
[7] https://ned.ipac.caltech.edu/
[8] http://simbad.u-strasbg.fr/simbad/

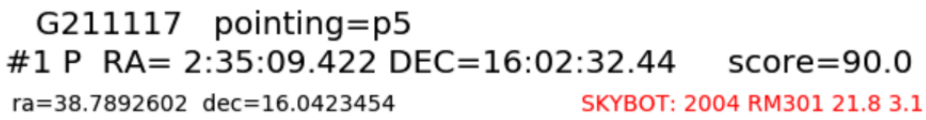

FIGURE 2.6: A zoom in of the "header" of the interface. The name of the survey exposed is G211117, is a consequential number indicating a survey. When astronomers are sure that a gravitational trigger is detected in a survey they change the name with the initial letters GW and the date. Basically G211117 and GW151226 are the same thing.

- *cl star* is the `SExtractor` classification object. If it is close to 1 that object is a star, near 0 means that is a galaxy.

Also the stamps are organized with a precise criteria. From the upper left to the right one can find the search image, the difference image (its name has the capital D letter before the date) and the reference one (indicated with the letter R before the date of the acquisition) and the light curve.
Other stamps are search images in different epochs all subtracted with the reference one.

2.2.2 Classification

At the left of Fig.**??** (see 2.5) there is the interactive portion of the interface where the user can input if the object represented is real or bogus and also input a tentative sub-type classification:

- *SN*: for the supernovae.
- *AGN*: for the nuclear active nuclei.
- *VAR*: for the variable objects.
- *MOV*: for the moving objects, anything showing signs of motion.
- dipole: is a type of bad subtraction where in the image difference a part of the object is positive (black) and a part negative (white).
- *bad subtraction*: residual profile inconsistent with a real source.
- *limit*: object is in the edge of a stamp.
- *bright star*: when the candidates is near a bright source faint residuals often remain.

2.2.3 Artefacts in Difference Images

We call bogus (following Bloom et al., 2012) artefacts of no astronomical interest.
Typically, thousand of variable transient candidates are detected on each subtraction image, the vast majority of which are subtraction artefacts, which can occur for a plethora of reasons, including improperly reduced images, edge effects on the reference or new image, misalignment on the images, improper flux scalings, incorrect PSF convolution, CCD array defects, and cosmic rays (Brink et al., 2013).
In figs. 2.10, 2.11, 2.12, 2.13 are represented some types of bogus.
In difference images reals objects have point-like residuals, artefacts have diffraction

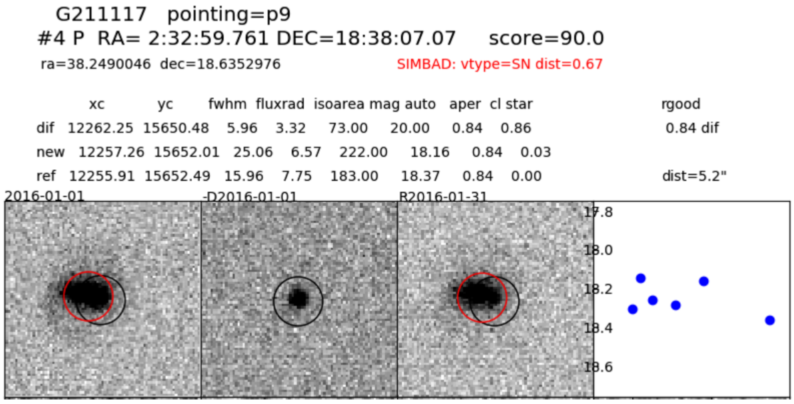

FIGURE 2.7: An example of SN.

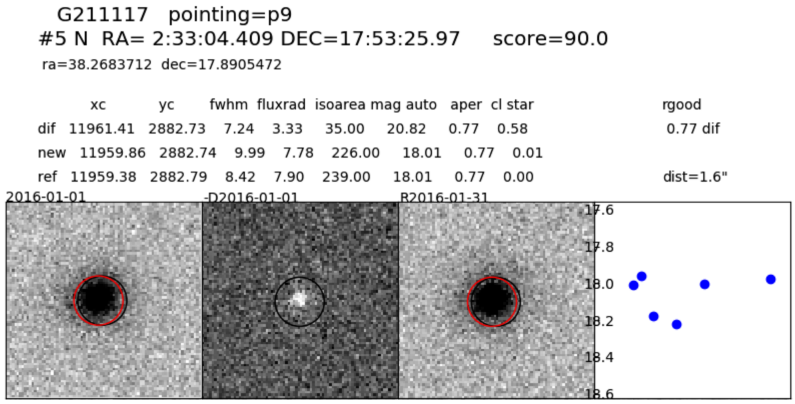

FIGURE 2.8: An example of AGN.

spike-like residuals while the dipoles/saturated class have residuals almost point-like with a part that has negative flux arising from registration errors or saturated CCD effects.

2.2.4 Results

In the 72 pointing of GW151226 the total amount of detected objects by the `SExtractor` was 352268. After the score selection they became 6300.
I visually inspected about 3000 objects, It is about half of the total number of selected candidates with ranking > 30 that for my purpose it is enough.
After this process one can really appreciate why the implementation of the machine learning would be a great help for the transient search.
In the modern wide-field surveys, where the number of candidates is huge, the visual inspection made by the astronomer is indeed the bottleneck in the target identification. For the classification procedure (see next chapter) it is important that the training set contains at least a few thousand objects with a comparable fraction of real and bogus.
In a real world the number of real candidates in a typical field is very small, less that few tens per square degree. This is a limitation when producing and adequate training set.
One solution is generate artificial stars in the image that however has a number of complications . For the classification procedure (see next chapter) it is important that the training set contains at least a few thousand objects with a comparable fraction

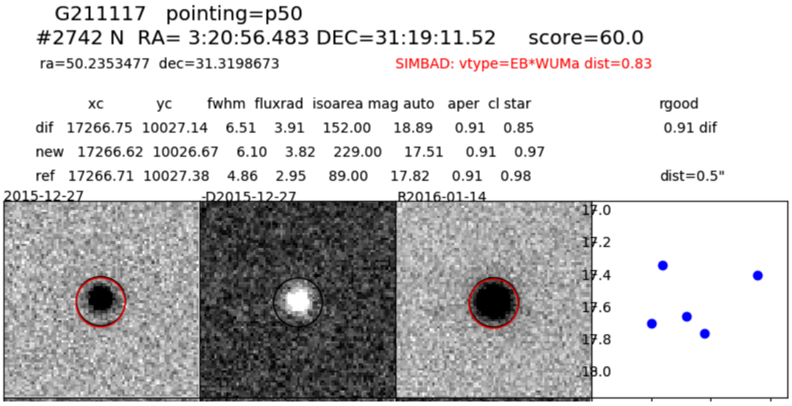

FIGURE 2.9: An example of VAR object, i.e. objects showing a record of long-term variability. This one is saved in the catalog as an eclipsing binaries.

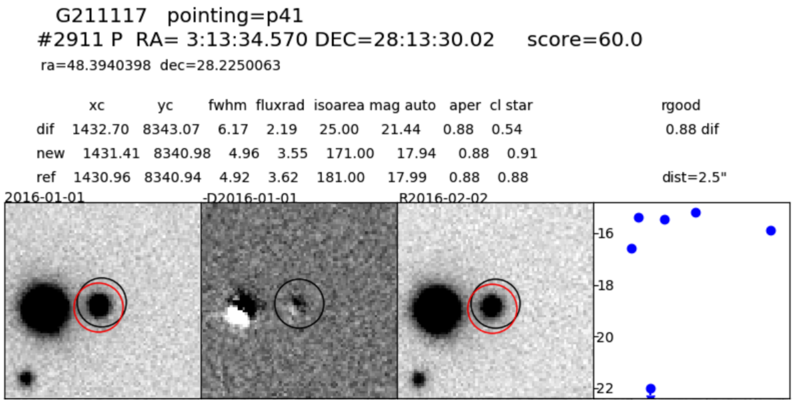

FIGURE 2.10: An example of bright star.

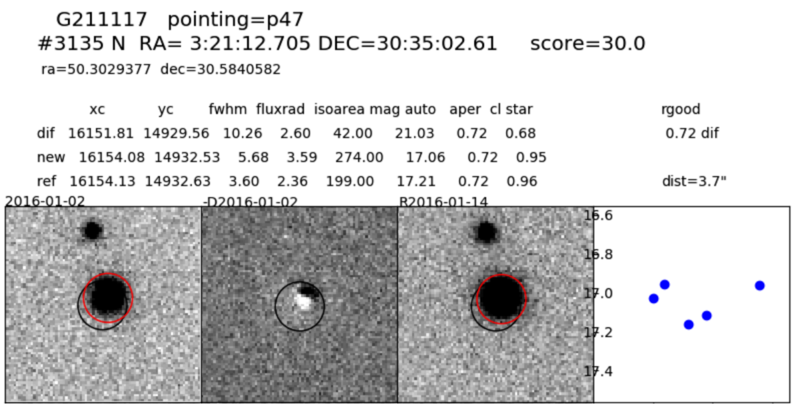

FIGURE 2.11: An example of dipole; they are residuals with roughly equal amounts of positive and negative flux, caused by errors in image registration.

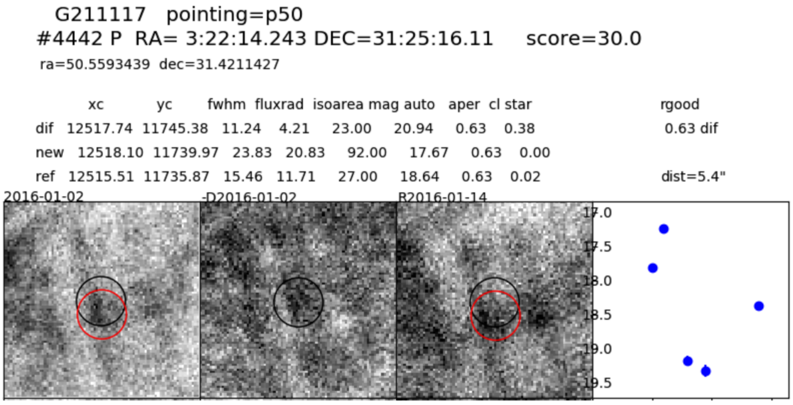

FIGURE 2.12: bogus

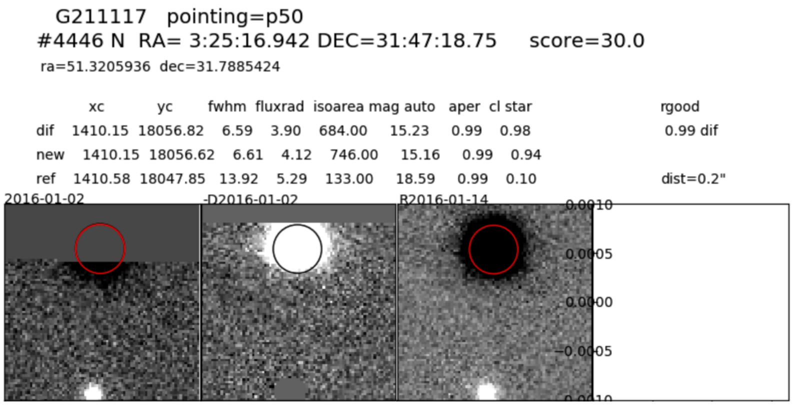

FIGURE 2.13: CCD error.

of real and bogus.
In a real world the number of real candidates in a typical field is very small, less that few tens per square degree. This is a limitation when producing and adequate training set.
One solution is generate artificial stars in the image that however has a number of complications .
Therefore we have a good fraction of real transient representatives. To match with a similar number of bogus we randomly selected from our database a sample of objects with low score.

Chapter 3

The training set

3.1 A brief introduction to Machine Learning

Arthur Samuel, a pioneer in artificial intelligence at IBM and Stanford, defined machine learning as *"The field of study that gives computers the ability to learn without being explicitly programmed"*.

Machine learning consists of using algorithms to extract informations from raw data and represent it in some type of model. We use this model to infer things about other data we have not yet modeled (Gibson and Patterson, 2017).
In my case the raw data are the subtracted images, the model is the ability of the machine to assign a number between 0 (bogus) and 1 (real) to the extracted candidates not yet classified.
Talking about "learning" for a machine could sound strange but its deeper meaning is that the machine uses algorithms for acquiring structural descriptions from data examples. A computer learns something about the structures that represent the information in the raw data.

There are two principal types of machine learning algorithms: the supervised and the unsupervised. If instances are given with known labels (the corresponding correct outputs, see table 3.1) then the learning is called supervised, in contrast to unsupervised learning, where instances are unlabeled (Gibson and Patterson, 2017).
We can also call the models built for information extraction as structural description.
Structural descriptions contain the information extracted from the raw data, and we can use those to predict unknown data. Models can take many forms such as: decision trees, linear regression, neural network weights.
Each model works in a different way. They apply different rules to known data to predict unknown data. Decision trees are so called because they create a set of rules in the form of a tree structure (as represented in 3.2). Linear models create a set of parameters to represent the input data (Gibson and Patterson, 2017).

Data in standard format					
case 1	Feature 1	Feature 2	...	Feature n	Label
1	xxx	x	...	xx	good
2	xxx	x	...	xx	bad
3	xxx	x	...	xx	bad
...	...	...	...	...	...

TABLE 3.1: Example of instances with known value

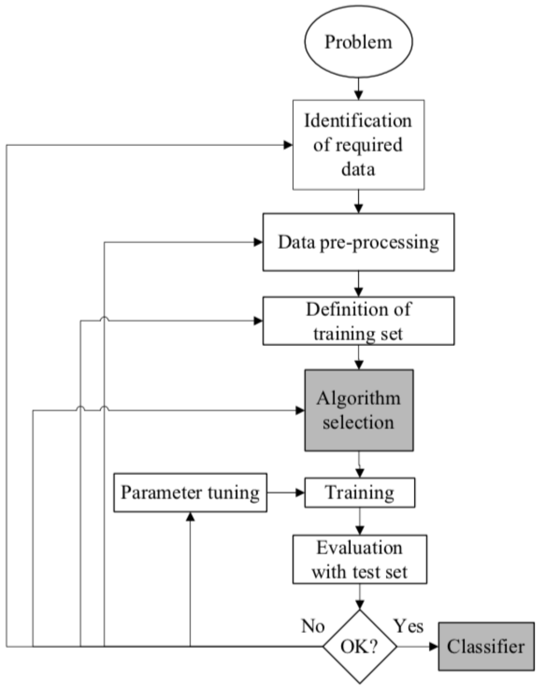

FIGURE 3.1: A schematic flowchart of how supervised machine learning works. In 2 is presented the data pre-processing, in this chapter is described the definition of the training set and in the next one (4) the algorithm selection, the training and the evaluation. Credits: Kotsiantis, 2007.

3.1.1 Decision Trees

Decision trees is a type of supervised algorithm where the data is continuously split according to a certain parameter. This is based on trees (transient candidates in our case) that classify instances by sorting them based on feature values (Kotsiantis, 2007).
A decision tree is drawn upside down with its roots at the top (see fig 3.2[1]).
The tree is composed by two entities: decision nodes and leaves. The leaves are the decisions or the final outcomes, it represents a value that the node can assume. The decision nodes are where the data is split, each node in a decision tree represents a feature of an instance to be classified.
Thus instances are classified starting at the root node and sorted based on their feature values.
It is easy to understand why a decision tree classifies an instance as belonging to a specific class, the feature importance is clear and relations can be viewed clearly.
There is a huge variety of learning algorithms based on decision trees. For my work I used the `Random Forest Classifier` as made available into the `scikit-learn` python package[2].

[1]https://www.xoriant.com/blog/product-engineering/decision-trees-machine-learning-algorithm.html
[2]https://scikit-learn.org/stable/modules/generated/sklearn.ensemble.RandomForestClassifier.htmlsklearn.ensemble.I

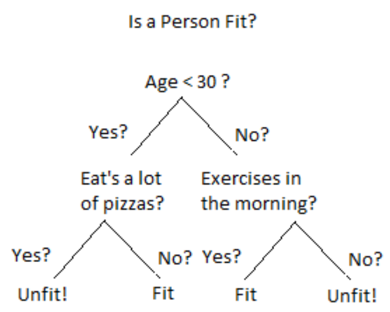

FIGURE 3.2: A schematic decision trees example. Credits: Mayur Kulkarnix

Random Forest

Random Forest (RF) operates by constructing a multitude of decision trees as ensemble of classifier. RF classify instances by combining prediction of their trees together (Buisson et al., 2015). The first algorithm for random decision forests was created by Tin Kam Ho (Ho, 1995) and introduced by Breiman, 2001.
RF as a supervised learning entails learning a model from a training set of data for which we provide the desired output for each training example (Wright et al., 2015). For that purpose a previous object classification is needed.
Random forests is a combination of tree predictors such that each tree depends on the values of a random vector sampled independently and with the same distribution for all trees in the forest. RF aims to classify examples by building many decision trees from bootstrapped (sampled with replacement) versions of the training data (Breiman, 2001).
In this work I use the `Random Forest classifier` `Python` implementation because it has been used in many surveys and compared with other algorithms it had the best efficiency. The RF classifier is superior to other methods in terms of accuracy, speed, and relative immunity to irrelevant features; the RF can also be used to estimate the importance of each feature in classification. It has shown high levels of performance in the astronomy literature (Wright et al., 2015; Bloom et al., 2012; Brink et al., 2013; Carliles et al., 2010; Richards et al., 2011).
The algorithm needs in input the labelled objects and a 1-dimensional (1D) vector (the flattened image) for each one. In order to built a training set of candidates with known class labels one rely on data from previous surveys; I used the GW151226 survey analyzed by me (as described in Ch. 2).
Each element of the vector correspond to some numeric data that lets the algorithm distinguish examples belonging to each class. Is it possible to see the feature importance for every image but for a stamp trivially this would be the pixels of the source. We decided to select the 20% of the elements in the list prepared adter GW151226 survey as a *test set* and the remaining 80% as *training set*. The test set is used for method comparison after the learner has been trained, for further discussion see Sect. 3.1.3.

3.1.2 Classification

Classification is the process of predicting the class of given data points. In my case stamps, or more precisely the pixels that compose the image, are the input features Classification belongs to the category of supervised learning where the targets also provided with the input data.
There are various type of classification. The most basic form is a binary classifier that only has a single output with two labels (two classes: 0 and 1). These classes are conventionally referred, in the literature, as positive classification (1) and negative (0). I have presented in chapter 2 the candidates classification. There are a lot of different type of real transients (1) and bogus (0) in subtracted candidates but all amenable in the two broad classes.
The output can also be a floating-point number between 0.0 and 1.0 to indicate the probability to belong to a certain class. We will see (chapter 4) that in most cases the precise natural number is quite impossible to receive as output.
The user needs to determine a threshold to delineate the boundary between the two classes. But, how to choose the threshold is discussed in the next sections.

3.1.3 Evaluating Models

Evaluating the machine learning algorithm models is an essential part of the project. Understanding how well an algorithms worked is possible comparing the known correct classification with the value of the prediction.
Thank to that process we understand how many real objects the machine has recognized, how many reals have been detected as bogus (*missed detection rate*, MDR) and how many bogus have been classified as reals (*false positive rate*, FPR).
Those parameters are easily visualized in the confusion matrix (3.2) and in a plot; in particular they are calculated with the following expressions:

$$f_p = false\,positive \quad f_n = false\,negative \quad t_n = true\,negative$$

$$FPR = \frac{f_p}{f_p + t_n}$$

$$MDR = \frac{f_n}{f_n + t_p}$$

The plot created plotting the false positive rate against the true positive rate at various threshold settings is called ROC (*receiver operating characteristic*) curve. It provides tools to evaluate the models and select the optimal model and threshold. For example if the astronomer can only accept to loose less than 5% of the real transients it takes a value of the threshold which corresponds to 5% of MDR. The threshold can be seen as a line after which objects change classification. In an astronomer work of transient discovery it's important to not loose many candidates but also it is no sense to choose a very low threshold because it will cause an high number of false positive objects and the use of machine learning to reduce the number of objects that need visual inspection will loose it's meaning.

The Confusion Matrix

The confusion matrix (see 3.2) is a table with rows and columns that represents the predictions and the actual outcomes (labels) for a classifier. We use this table to

		True condition	
		condition positive	condition negative
Predicted condition	Predicted condition positive	True Positive	False positive
	Predicted condition negative	False negative	True negative

TABLE 3.2: An example of confusion matrix

better understand how well the model or classifier is performing. It make possible to describes the complete performance of the model.

Accuracy

Accuracy is the degree of closeness of measurements of a quantity to true value of that quantity, it can be also seen as the average of the values lying across the "main diagonal" of the confusion matrix.

$$Accuracy = \frac{(TP + TN)}{(TP + FP + FN + TN)}$$

Accuracy is the ratio of number of correct predictions to the total number of input samples so it has a good result only if the number of samples belonging to each class are in equal number. For that is better to create a training set with the equal number of bad and good candidates.

3.2 Definition of the training set

The classification results of the Ranking approach (see chapter 3) are automatically saved in `MySQL` tables.
The first step of the preparation of the machine learning input is to handle those tables:

- `GW_classify_margherita` is the file where the classification parameters are saved, like the id or the candidates type (see Fig. 2.5)
- `candidates_GW151226` is the file where the candidates informations are saved, like the id, the pointing, the search date, the reference date, center coordinates x, y, the magnitude, the image type (positive or negative subtraction) and the ranking. The majority of those parameters are explained in paragraph 2.2.1.

In the Appendix A it is reported a part of the script I wrote for this thesis. For now I will focus on the first two functions of the script.

The `read_negativi` function

The `Random Forest Classifier` requires a number of examples of real and bogus of equal value.
Usually in one survey those numbers show a huge difference: even after ranking the number of bogus is two order of magnitude higher than the real transients. Astronomers solved this problem creating artificial stars.

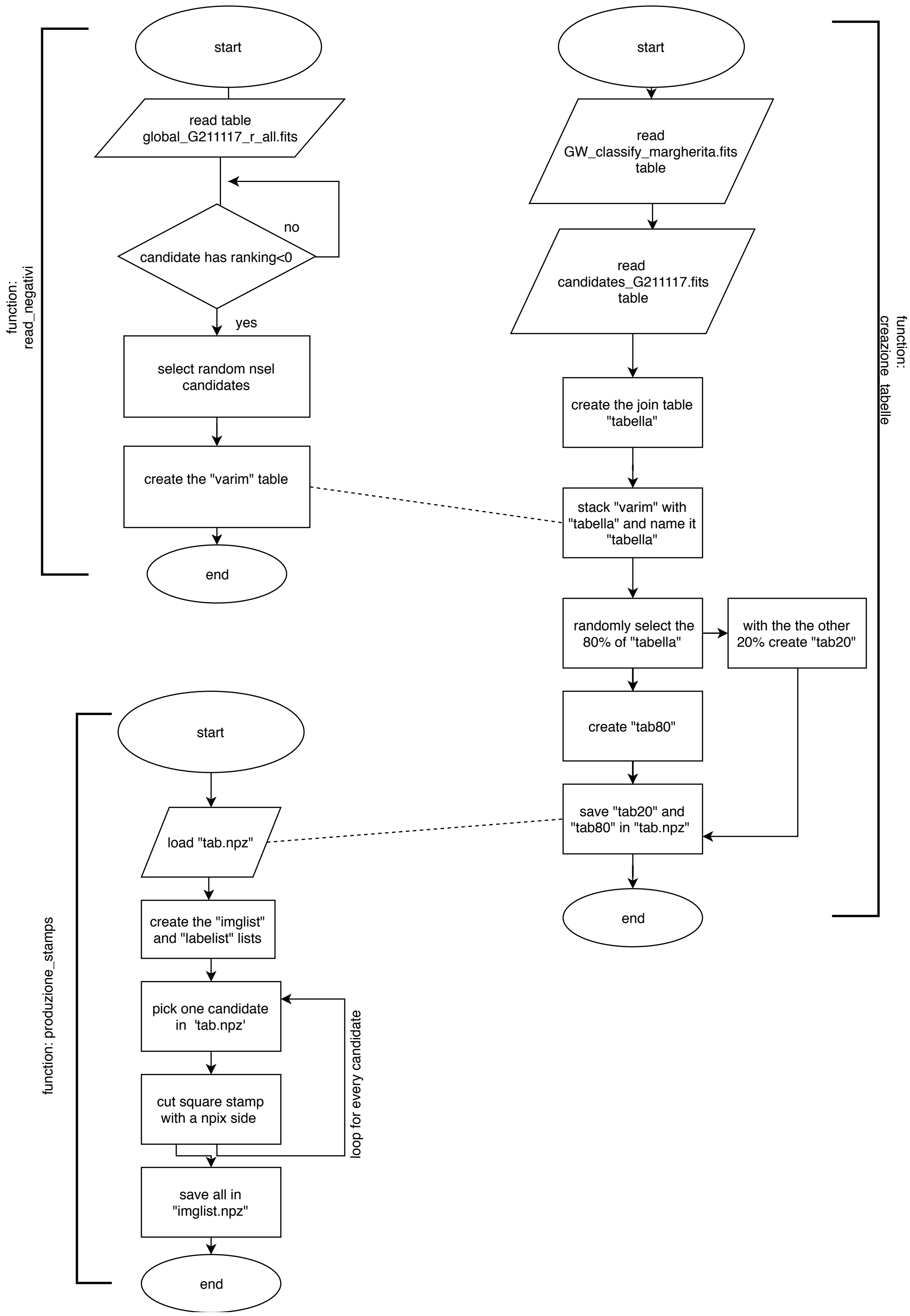

FIGURE 3.3: Preparation of the training set script flowchart.

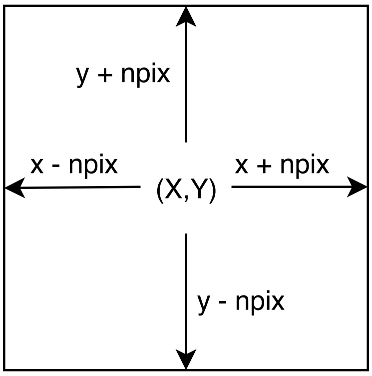

FIGURE 3.4: Stamp creation.

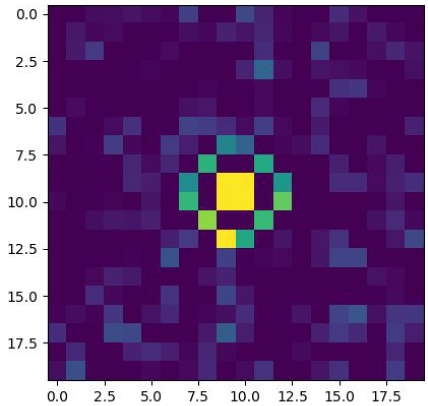

FIGURE 3.5: Example of a stamp.

In the GW151226 survey this is not required since the observation was made with a low declination, near the ecliptic, that leads to a huge asteroids observation. Those are not real transients since they not change in magnitude, instead they appear in one epoch and in the next one they are gone, but those are a good input for the machine learning process.
Actually we need more bogus than those I classified. The task of `read_negativi` is, precisely, to read a number of candidates with negative ranking. We choose to use negative score objects that we are sure are almost all bogus.
The function requires a parameter in input (`nsel`), that is the number of negative score objects that the user needs.
`Read_negativi` reads from the fits table `global_G211117_r_all` ,which contains score values and objects informations, and give an output catalog with the informations that we need such the `id, search, coox, cooy, date_new, Ranking, pointing` (see Appendix A).

The `creazione_tabelle` function

The `creazione_tabelle` function is made for the table handling.
The first step consists in reading the `MySQL` tables: `GW_classify_margherita` and `candidates_G211117`. The two tables are joined with the parameters we need. In particular the candidates type are merged in the two macro-categories that we already know: reals and bogus. The table is then saved as `numpy` table for an easier manipulation. Then the final table including the negative ranking is divided in two different tables: one that contains the 20% (identified as `tab20`) of the total amount of candidates and one with the 80% (`tab80`). Table splitting in performed by random selection.

3.2.1 The stamps creation — `produzione_stamps` function

The `produzione_stamps` function takes as input the `npix` value i.e. the number of pixel of the square stamp side.
When the script runs the user has to input the length of the stamp size. `Npix` is half of the input value because the script takes (from the tables created in the function 3.2) the coordinates of the source center and then adds and subtracts the `npix` value creating a stamp around the source center.
Fig.3.4 shows how the stamps are constructed.

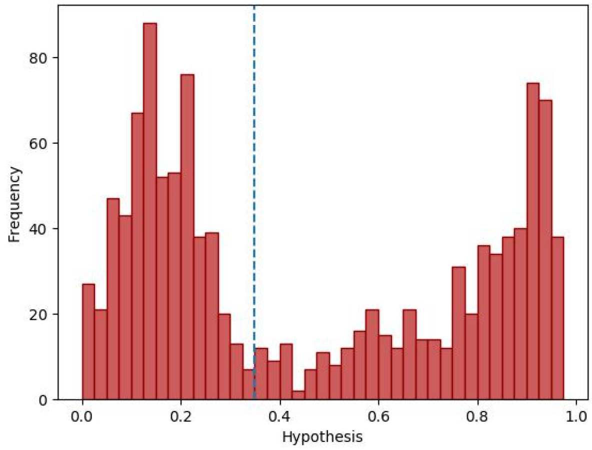

FIGURE 3.6: Histogram that shows the RF predictions. The blue dotted line is the threshold at MD=10%.

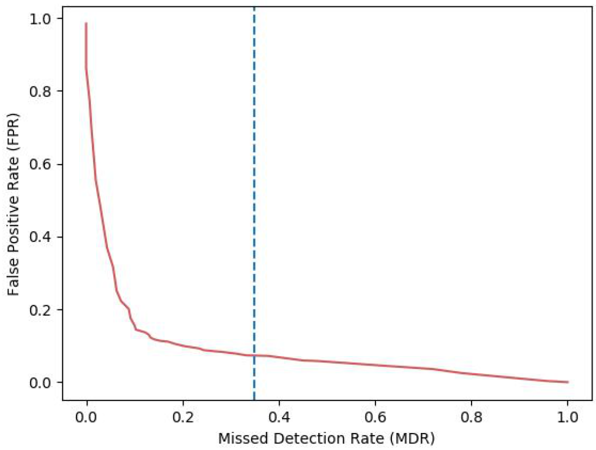

FIGURE 3.7: Missed Detection Rate and False Positive Rate plot. The dotted line indicate the selected threshold at 0.35, it coincides with a MD=10%.

In this part of the candidate handling is important to avoid those lying near the the edge of the image. At the end of this script all the candidates labels and their cropped stamp are saved. The labels are a number indicating the "nature" of the object, in other words if is real or bogus. We adopted that real objects are indicated with the number 1 and bogus one with 0. This convention is used in the whole thesis. The labels list and the stamps are the input for the machine learning. I will explain how to implement those in the next paragraph.

3.3 The Random Forest implementation

3.3.1 The `machine_learning` function

This function permits to create a model able to classify further surveys (like the GW171408 one, see Ch4).
`RandomForestClassifier` meta estimator, from `sklearn.ensemble` library, requires

various input hyperparameters[3]; all of them are optional and if they are not specified a default value[4] is adopted. The only parameter I specified is the number of estimators, i.e. the number of trees in the forest.
The method `fit(X,Y)` builds a forest of trees from the training set (X, Y) in which X has to be the flattened stamps and Y the list of the labels for each object.
The `predict(X)` method predicts classes for X, i.e. it is composed by a list of 0 and 1. The `predict_proba(X)` method predicts classes probabilities for X, i.e. it is composed by list of float value between 0 and 1. In the `sklearn` jargon this prediction is named *hypothesis*, it can be seen as the probability that an image has of belonging to the class of real images. In these two methods the input X is the test set, namely the set composed by the 20% of the entire dataset.
Thanks to the *test set* is possible to visualize the probability predictions in a histogram (fig. 3.6) and evaluate the performance of the classifier algorithm. This plot is created with bins of 0.025 of the *hypothesis* value.
In fig. 3.6 we see that the picks are near the limits 0 and 1. This is the first sign that machine learning is working efficiently because is clear the division between the two classes.

3.3.2 `Fit` results

Using the `joblib` method[5] made available by `scikit-learn` is possible to store a model after calculation.
The performance of random forest depends on the choice of proper parameters for the algorithm. For instance it is known (Wright et al., 2015) that changing `n_estimators` or stamps size change significativly the algorithm performance.

Testing dependence on stamp size

I ran the machine learning with different input images sizes and I plotted for each selected value the ROC curve(fig 3.8). For the python code see A.2.
The performance changes is evident with very small stamps like the 2x2 or even 4x4 pixels. When: the input stamps are too small so not all the relevant objects pixels are included in the image and it become difficult to detect real sources.
We decided to take 20x20 pixels stamps because they have the right amount of pixels (400) so the source is all contained in the stamp and there are not too many background pixels in order to avoid contamination by nearby sources.

Test of effect of estimator number

In fig. 3.9 are shown the ROC curves made with different numbers of trees (`n_estimators`). The performance gets worse when we use a small number of estimators (less than 50). We decided to use `n_estimators=100` because it gives an efficiency very similar to the maximum value but it takes a lot less time to run.
The threshold (also called hypothesis, h) in binary classification is 0.5 by default. In general, we want to select a value of h that guaranties some specific performance.

[3]In machine learning, a hyperparameter is a parameter whose value is set before the learning process begins.

[4]see the full list of hyperparamethers here: https://scikit-learn.org/stable/modules/generated/sklearn.ensemble.RandomForestClassifier.html

[5]https://scikit-learn.org/stable/modules/model_persistence.html

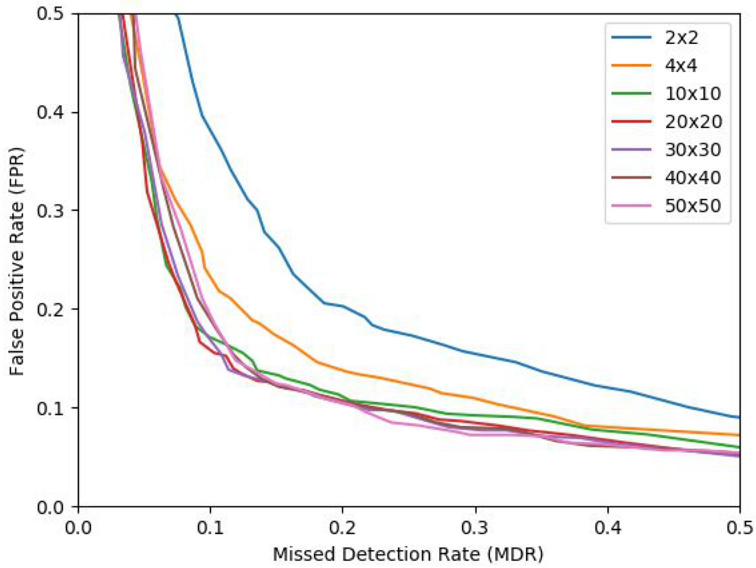

FIGURE 3.8: ROC curves of different stamp sizes.

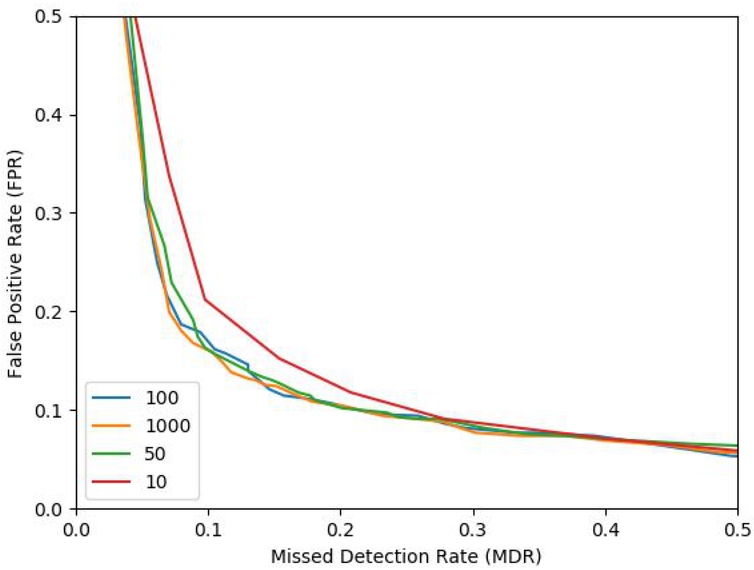

FIGURE 3.9: ROC curves made by `fit` with a different number of estimators.

To this aim we printed in tab B.1 the values of the *hyphotesis*, and the corresponding *missed detection rate* and the *false positive rate*.
With any classifier the largest h is the lower the MDR but higher the FPR and vice versa. We decided to take as reference a MD equal to 10% so our reference value of h was 0.35 (see tab.B.1). Now we have all the elements for applying the derived random forest model `clf` to a new survey.

Chapter 4

Application of the classifier to the GW170814 survey

4.1 External check

The Random Forest algorithm after learning from the training set returns a model that describes the relationship between the observed data and the class of the source. Once the classification function (`clf`) is estimated, it can be employed to predict the class of any future object from its observed data (Brink et al., 2013).
In Ch.3 I described how I prepared and implemented the *training* and *test* sets. In this chapter I will describe how I implemented the `clf` created with the GW151226 training set to a new data set obtained for a different GW trigger (GW170814).

4.1.1 Classifier implementation

The transients found during the GW170814 follow-up survey have been already classified by other members of the GRAWITA team using the ranking algorithm and visual inspection. Hence we are able to make a cross check with the ones found by the *classifier*. For my model evaluations I will change some parameters in the script. Those are global variables declared at the beginning of the script (see A.3) and are:

- `score_limit`: the minimum score of the candidates. For example, if I use the value 30 the `clf` classifies only the candidates with score greater than 30. The minimum possible value to take is -60.
- `use_clf`: the type of the *classifier*. I computed different `clf` based on different training sets. didates over a certain magnitude and with `n_estimator`=100, see 4.1.5
- `show_plot`: a boolean value True/False in order to visualize or not the histogram plot.
- `hlimit`: the minimum value of the threshold above which the candidates are detected as reals. It makes possible to change the *hyptohesis* value hence the MDR and the relative FPR (see B.1). This means that if I chose h=0 the `clf` would detect all the objects in the survey as real transients; if I chose h=0.35 (i.e. MD=10%) the `clf` would detect as reals only the candidates above that threshold value.

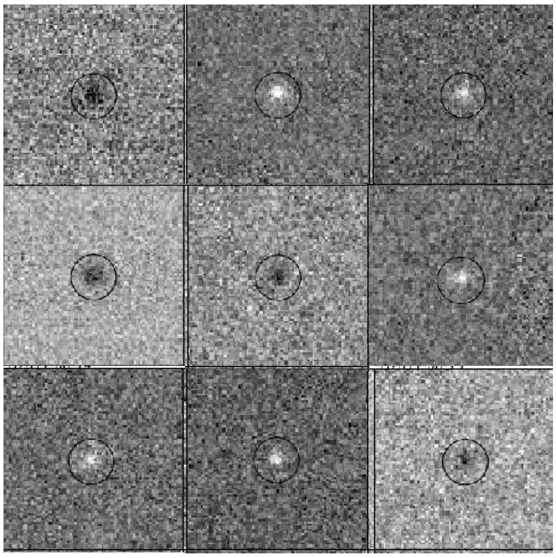

FIGURE 4.1: Stamps of the 9 non detected objects.

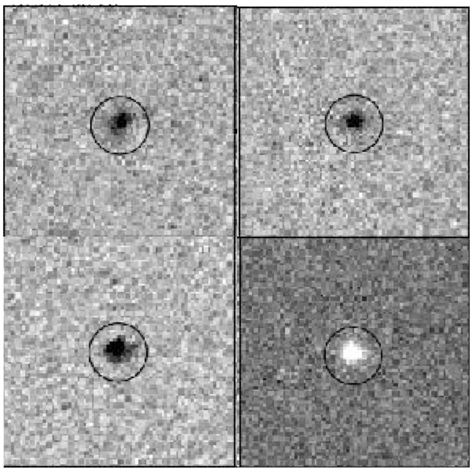

FIGURE 4.2: Stamp examples of 4 detected objects.

4.1.2 Stamps preparation

The process of stamps creation is the same as the one done for training set, see Ch.3.2.1. We need to create the stamps of all the candidates in the survey.
In GW170814 without score limitation there are 248265 candidates, instead those with score greater than 30 are 9362. All values are shown in tab 4.1.
My intent is to evaluate the performance of Random Forest, without the application of any prior method, so first of all I implemented the survey without score limitation.

4.1.3 `Clf` implementation — The `model_implementation` function

Survey with no ranking limitations

The input in the methods `predict()` and `predict_proba()` (both described in 3.3.1) are the flattened images of the candidates. It is possible to create an histogram (fig.4.3) that shows the trend of the predictions probabilities, like the one in fig.3.6.

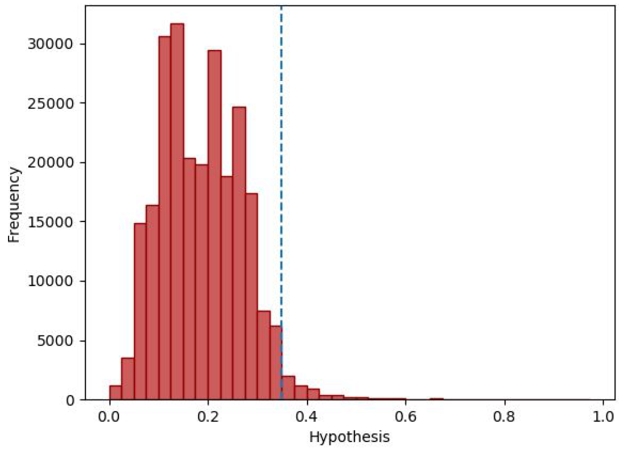

FIGURE 4.3: Histogram of the detected sources of all scores objects in GW170814 survey.

This histogram is quite different from the one in the previous chapter. It has an higher candidates frequency near the 0. This is due to the fact that there is a different real/bogus ratio between the two set of data. The number of detected transients by the `clf` is 5882.

Having selected 5882 candidates in a survey of 248256 objects is a good result, the classification made by the RF has diminished by the 98% the initial number of candidates. But still the ∼6000 transient candidates found by the algorithm should be manually checked. It is important to consider that in the discovery of transient events the timing is a vital aspect. How early a GWs optical counterpart is detected and how soon a potentially crucial follow-up can be deployed are the the main reasons why astronomers needs to implement a machine learning algorithm that speeds up the process.

Survey with ranking limitations

I implemented the ranking limited survey. The total number of objects with score>30 is 9362. The number of the detected objects by the `clf` is 743 (the relative histogram is in fig. 4.4). This is much smaller number than 5882 and it is feasible to visually inspect those candidates in a short time.
We conclude that the best strategy is first to make a ranking selection and then implement the *classifier*.

4.1.4 Cross-check — The `cross_check` function

The ∼750 candidates found by the machine learning algorithm need to be cross checked with the ones found by the GRAWITA team.
The list of the already classified objects contains only plausible extragalactic transients which are possible optical counterparts of gravitational waves so are transient like asteroids and variable stars have been removed. However, with a cross check of only those objects, the `clf` evaluation would not be fair because we trained the algorithm with all the types of transients. The solution devised is to visually check the ∼750 candidates. I found 103 real transients in the *classifier* detected objects. This is equal to a *FPR=14%* (marginally lower than the one we expected of the 17%, see

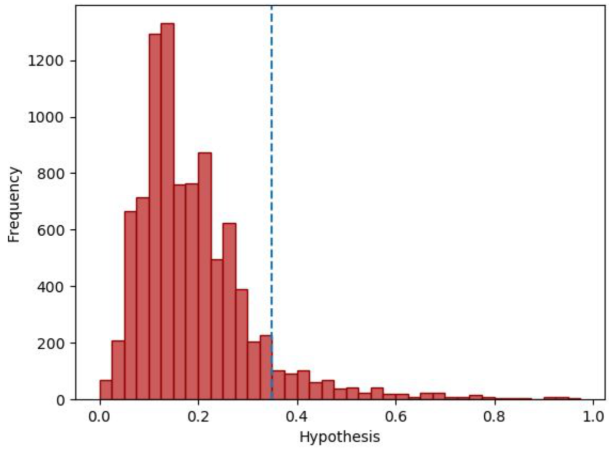

FIGURE 4.4: Detected objects by the `clf` in GW170814. The dotted blue line is at h=0.35.

clf	score	h	*MDR (clf)*	*FPR (clf)*	Matched (GRAWITA)	Matched (by me)	Detected	Full table
G211117	-60	0.35	10%	17%	39		5882	248265
G211117	30	0.35	10%	17%	39	103	743	9362
G211117	30	0.23	5%	32%	44		2418	9362
G211117	60	0.35	10%	17%	34		309	1688
G211117	90	0.35	10%	17%	8		30	115
G211117 mag limited	30	0.30	10%	8%	41		1172	9362

TABLE 4.1: This is the table with the final results of the cross checks. In the first column is written the name of the *classifier*, G211117 means that it has been trained in the GW151226 survey. In the second column is specified the value of the `score_limit`, in the third column is written the threshold value with the relative FPR and MDR. Then there are the number of matched candidates: firstly with the GRAWITA list (composed of 48 objects) and secondly my classification. In the last two columns there are the detected objects by the `clf` and with the number of candidates given in input.

Confusion Matrix

		True Condition		
		Real	*Bogus*	
Predicted	*Real*	*14%*	87%	cross check values
Condition	*Bogus*	*10%*	90%	input values

TABLE 4.2: This is the confusion matrix based on my classification of the 20x20 pixels stamps of ranking limited objects of the GW170814 survey, with a *classifier* trained in the GW151226 survey with a threshold=0.35.

B.1) and a *MD=10%* corresponding to *h*=0.35.
Of the about 750 objects:

- 39 were in the list of the 48 objects given by the GRAWITA team, that is the 82% of the candidates.
  In fig.4.1 are shown the stamps of the 9 non detected objects, they are fainter than the detected objects, see an example of detected objects in fig. 4.2). Likely this is the reason why they are not in the list of the ∼750 candidates.
  Of the ∼6000 objects detected by the *classifier* regardless of ranking I found the same 39 objects. This means that there are no possible real transients with score lower than 30 and that the ranking approach is reliable.
- in my classification there are:
  - 19 are plausible extragalactic transients.
  - 84 are consistent with variable stars or asteroids
  - 139 are unclear cases that could be either real or bogus, like a faint residual of the image subtraction.

Tab 4.1 shows the numbers of detections and matching for different score limits. We see that a `score_limit` equal to 60 or 90 is too high and we lose a lot of candidates. The conclusion is that the best way of proceeding is using the ranking approach, selecting the objects with score greater than 30, and then implementing the machine learning *classifier*.

4.1.5 Further analysis

Limitation in magnitude

I described the peculiarity of the GW151226 survey in paragraph 3.2. In the survey there are a lot of candidates that appear only at one epoch this is due to a large contamination from asteroids. Asteroids are usually brighter than other sources because are relatively nearby. Extragalactic transients possible GW counterparts are tipically faint, then we want to study what happens when we train the RF only with faint objects.

In fig.4.5 I divided the *training set* (sources with score>30) into bins of range 1 mag, from the lower magnitude to the higher. Evaluating the histogram I restricted the survey to the candidates with magnitude greater than 20 mag, see 4.6. I created a *training set* only with the fainter objects, the MD and FP values are in tab. B.2. The MD=10% corresponds to a *hypothesis* value of 0.30, the ROC curve is shown in 4.8.

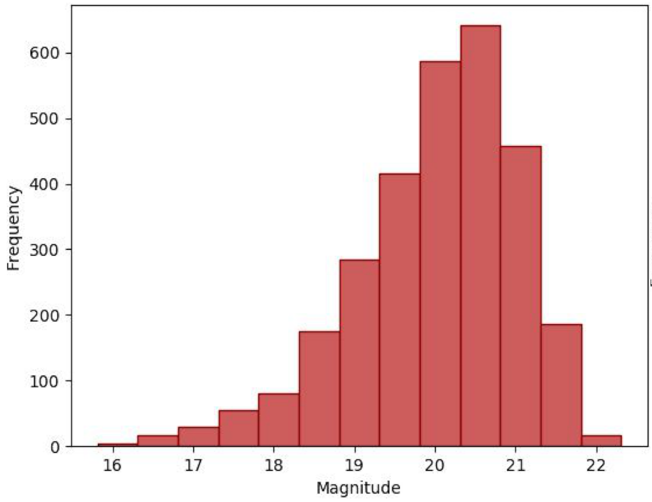

FIGURE 4.5: Magnitude trend of the GW151226 survey. The bins have range 1 mag and are from the lower magnitude found in the survey from the higher.

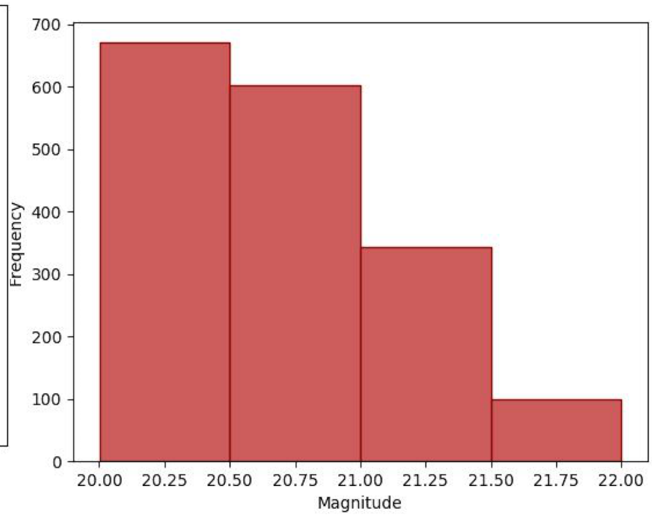

FIGURE 4.6: Limited magnitude trend in the GW151226 survey. The bins have a range of 1 mag and they start from 20 mag.

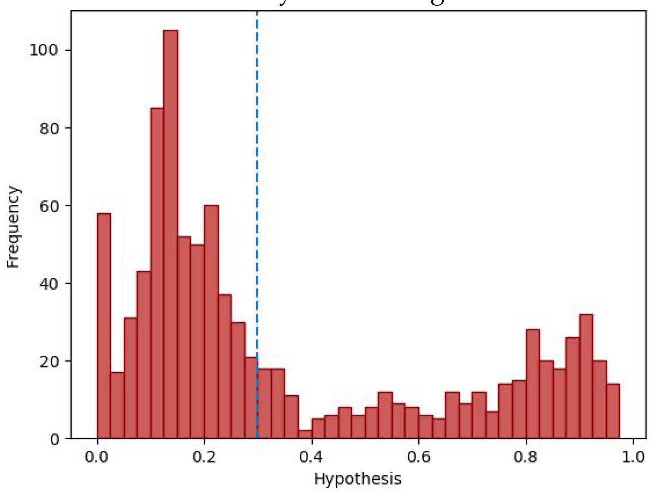

FIGURE 4.7: Histogram that shows the trend of the RF predictions for the survey with limited magnitude. The bins have a range of 0.025.

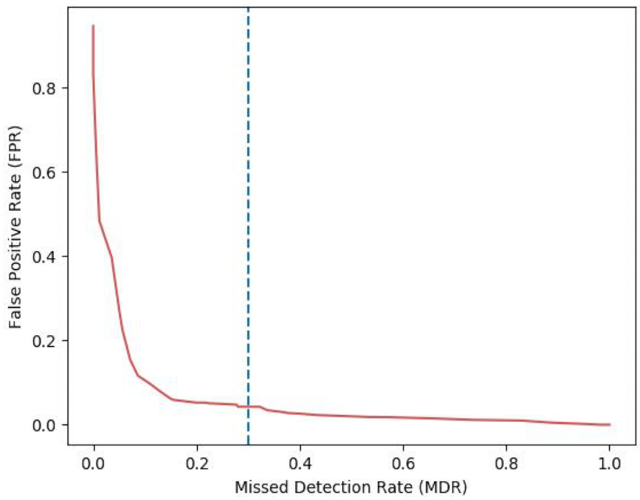

FIGURE 4.8: ROC curve for the survey with limited magnitude. The blue dotted line is at h=0.30 that correspond at MD=10%.

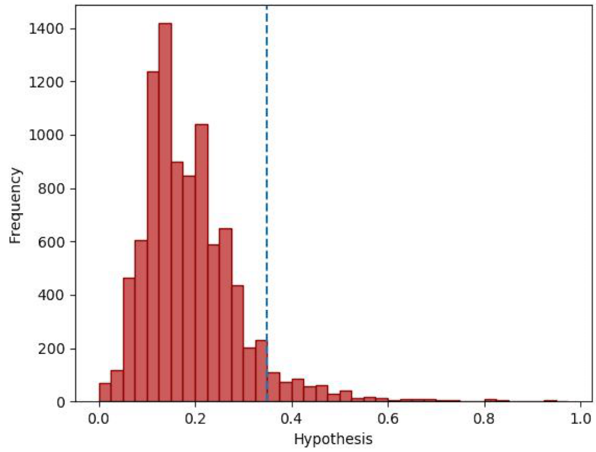

FIGURE 4.9: Trend of the objects in the GW170814 survey with a `clf` trained in a survey limited in magnitude. The dotted blue line is at *h*=0.30.

The implementation of new `clf` in the GW170814 survey created a trend of the predictions like the one shown in 4.9. The detected objects are 1172, a number of detections higher than the one made by the *classifier* with no magnitude limitations. The cross-check with the 48 objects of the GRAWITA team generated an higher number than the previous. 41 GWs optical counterparts were detected. It is not a great improvement: we detected 2 more objects but we should manually check twice the objects of the previous `clf`.
The conclusion is that a magnitude limitation is not necessary.

Changing of the *hypothesis* value

I tried to change the *hypothesis* value to 0.23, corresponding to a MD=5% (tab 4.1). I matched 44/48 objects of the GRAWITA list, that is the 92%.
It seems a great achievement but the number of detected objects rapidly increased and so the `false positives`. From the tab B.1 we see that with a MD=5% we should have a FP=32%, we have a FPR=98%. We can not inspect such an amount of objects, so MD=10% is the right compromise.

4.2 Conclusions

In this thesis I motivated the need for a machine learning approach in the search for GW optical counterparts.To set the context I first describe the tools used by the GRAWITA group: the `VSTtube`, the `SExtractor` and the ranking approach that provides a score for all candidates, higher for plausible true transients and low for .
I visually classified about 3000 objects of the GW170814 survey, all those with high score. These objects, integrated with a similar fraction of bogus extracted randomly from low score candidates was used to construct a *training set* and a *test set* for the machine learning implementation. I used the first one to train the `Random Forest classifier`. Test set is used to evaluate the performance of the method, and in particular a threshold, i.e a *MDR* and *FPR* value.
As the feature representation of the detections I decided after some testing, to use 20x20 pixels stamps. Also as the number of trees in the forest, I adopted `n_estimators`=100.
I evaluated the performances with the decided *hyperparameters* and I decided to take

a MD=10%. I trained the algorithm with the decided *hyperparameters*, I evaluated the performances with the *test set*, I decided to take a MD=10% and I saved the classifier that was created. As an external check I applied the `clf` in another survey, the GW151226 one. Firstly the `clf` classified the GW151226 objects without ranking limitation. I made a cross check of the machine learning detected objects with the list of the 48 known GWs counterparts in GW151226. The `clf` detected about 6000 objects, but only 39 matched the list of real transients.
Then I applied the *classifier* to the list of $\sim$9400 with high ranking of the GW151226 survey. With MD=10% I recovered the same previous 39 objects; with MD=5% I found 44/48 objects. On the other side with a MD=10% the number of detected objects is 743 while with a MD=5% they are 2418 that is the number of false positive visibly increased.
My conclusion is that in order to reduce the number of candidates left for visual inspection the best strategy is implement the machine learning to the ranking limited sample of objects.
All together we concluded that while further testing are certainly required the current implementation is already a valuable aid and indeed will be implemented in the incoming observing season .

Appendix A

Python Scripts

A.1 Script for `clf` creation

```
import os,sys,glob
from astropy.io import fits
from astropy.table import Table, Column,vstack
import numpy as np
import sqlconn
import random
import matplotlib
matplotlib.use ('TkAgg')
import pylab as plt
from sklearn.ensemble import RandomForestClassifier
from sklearn.externals import joblib
from scipy import interpolate
from sklearn.externals import joblib

nsel = 3000 # number of negative ranking candiates

#function that reads a number of negative ranking candidates equal to nsel
def read_negativi (nsel):

    #path
    dataou = os.path.expandvars("$gw_dataou")
    fdir_n = dataou+'G211117/global/'
    cdiff = 'global_G211117_r_all'
    #read the fits table
    catalog = Table.read(fdir_n+cdiff+'.fits')
    #I select only negative ranking obj
    ineg =np.where(catalog['Ranking']<=0)[0]
    #random selection
    isel = np.random.choice(ineg,nsel)
    #I create the table with the paramaters of interest
    catalog['NUMBER_1','search','X_IMAGE_1','Y_IMAGE_1','epoch','Ranking',
    'pointing'][isel]


    catalog.rename_column ('X_IMAGE_1', 'coox')
    catalog.rename_column ('Y_IMAGE_1', 'cooy')
    catalog.rename_column ('epoch', 'date_new')
    #id make possible to univocally identify the candidates
    catalog.rename_column ('NUMBER_1', 'id')


    catalog['date_new']=catalog['date_new'].astype(str)

    #date:dd-mm-yyyy
    for i in range(len(catalog)):
        catalog['date_new'][i] = catalog['date_new'][i][:4]+'-'+\
```

```
            catalog['date_new'][i][4:6]+'-'+catalog['date_new'][i][6:]

    #per i candidati con ranking negativo impongo un id minore di 0 in modo tale da potrerli riconoscere subito
    catalog['id']=catalog['id']*(-1)

    return catalog['id','search','coox','cooy','date_new','Ranking','pointing'][isel]





#function that join the table with the eyeballed candidates with the negative ranking one. Two tables, one with 20% of the total candidates and one with the 80% are the function final result.

def creazione_tabelle():

    #I make an sql table with all the type of real or bogus merged in the two macrocategories.In particular:
   #                                  -join between: GW_classify_margherita and candidates_G211117.
    #GW_classify_margherita is the table with the eyeballed obj, candidates_G211117 is the one that contains all the parameters of the obj with score > 30.

    command='select REALS+SN+AGN+VAR+MOV as reals ,BOGUS+BAD+LMT+BRIGHT+MB as bogus, can.number, can.coox, can.cooy, can.pointing, can.magauto, can.ranking, can.date_new, can.search from GW_classify_margherita as gwc inner join candidates_G211117 as can on can.number=gwc.id;'

    tab=sqlconn.query(command,sqlconn.conn2)

    #creation of the sql table

    number,coox,cooy,pointing,reals,bogus,magauto,ranking,date_new,search = [],[],[],[],[],[],[],[],[],[]

    for t in tab:
        number.append(t['number'])
        coox.append(t['coox'])
        cooy.append(t['cooy'])
        pointing.append (t['pointing'])
        ranking.append(t['ranking'])
        reals.append(t['reals'])
        bogus.append(t['bogus'])
        magauto.append (t['magauto'])
        date_new.append (t['date_new'])
        search.append (t['search'])

    number = np.array(number)
    coox = np.array(coox)
    cooy = np.array(cooy)
    pointing = np.array(pointing)
    ranking = np.array(ranking)
    reals = np.array(reals)
    bogus = np.array(bogus)


    #id -> unique identificator (id positive for the classified objects and negative for the ones with score<0)
    #coox-> x coordinate of the centre of the obj
```

```
    #cooy-> y coordinate of the centre of the obj
    #pointing-> number of the pointing where the source is included
    #ranking -> rankingof the obj
    #reals -> it is equal to 1 if the obj has been classified as real
    otherwise is 0
    #bogus -> it is equal to 1 if the obj has been classified as bohus
    otherwise is 0
    #magauto -> magnitude of the obj
    #date_new -> date of the observation
    #search -> you find P if the diff image is Positive or N if it's
    negative

    tabella = Table([number,coox,cooy,pointing,ranking,reals,bogus,magauto
    ,date_new,search],names=['id','coox', 'cooy','pointing','Ranking', '
    reals', 'bogus','magauto', 'date_new','search'])


    varim = read_negativi(nsel)
    tabella = vstack([varim,tabella])

    #I create the table with the 80% of the total amount of obj taken
    randomly
    ii = random.sample (range(0,len(tabella)),int(len(tabella)*.8))
    bb= Column (length=len(tabella), dtype=bool)
    tabella.add_column(bb,name='bb')
    tabella['bb'][:]= False
    tabella['bb'][ii]=True


    #creation of the two tables: tab20, tab80
    tab80=tabella[np.where(tabella['bb'])]
    tab20=tabella[np.where(tabella['bb']==False)]

    print (tab20)
    print (tab80)

    np.savez('tab.npz',tab80=tab80, tab20=tab20)


#function for the creation of the stamps of a precise size that the user
    can decide
def produzione_stamps (npix):

    #load the tables
    npztab = np.load('tab.npz')
    tab80 = npztab['tab80']
    tab20 = npztab['tab20']

    #find the path
    data = os.path.expandvars("$gw_dataou")
    _gwdir = data+'G211117/'

    imglist,labelist = {},{}

    #In this loop I create the stamps (with the size indicated by the user
    )
    for t,l in zip([tab80,tab20],['tab80','tab20']):
        imglist[l],labelist[l]= [],[]
        for i in range(len(t)):

            pointing = t['pointing'][i]

```

```python
            lista = glob.glob(_gwdir + t['date_new'][i]+'/diff_G211117_r_'+\
                    t['date_new'][i].replace('-','')+'*'+pointing.replace(' ','')+'.fits')
            #print (lista)
            if len(lista)>1:
                print ("ERROR:  more than one file")

            hdr = fits.open(lista[0])

           #I read from the table the coordinates of the obj center
            xcentre=t['coox'][i]
            ycentre=t['cooy'][i]

            xplus= int(xcentre) + npix
            xminus= int(xcentre) - npix
            yplus= int(ycentre) + npix
            yminus= int(ycentre) - npix

            limy,limx = hdr[0].data.shape

            if xminus>=0 and xplus<=limx-1 and yminus>=0 and yplus<=limy-1:
                imglist[l].append(hdr[0].section[yminus:yplus,xminus:xplus])
                labelist[l].append(t['reals'][i])
                print (len(imglist[l]))

            #plt.imshow(imglist[l][i], vmin=0, vmax=200)
            #plt.show()


        imglist[l] = np.array(imglist[l])
        labelist[l] = np.array(labelist[l])

    np.savez('imglist',imglist80=imglist['tab80'],imglist20=imglist['tab20'],labelist80=labelist['tab80'], labelist20=labelist['tab20'])



#function that runs the machine learning
def machine_learning():

    flist = np.load('imglist.npz')
    #I load obj lists (made by only 0 and 1, 1 stays for real and 0 for bogus)
    labelist80 = flist['labelist80']
    labelist20 = flist['labelist20']
    #I load the stamsps
    imglist80 = flist['imglist80']


############### tab 80  #########
    rimglist80=[]
    for img in imglist80:
        #The Random Forest Classifier requires the stamps with a flattened shape
        rimglist80.append(img.flatten())

    rimglist80 = np.array(rimglist80)

    print('Shape of the inputs for the machine learning: ', rimglist80.shape,len(labelist80))
```

```
    #I calculate the dimension of the stamps without asking again to the
    user
    a =rimglist80.shape[1]
    n=int(np.sqrt(a))
    n=str(n)
    print('You are using '+n+'x'+n+' stamps')


############### tab 20 #########


    imglist20 = flist['imglist20']
    rimglist20= []

    for img in imglist20:
        rimglist20.append(img.flatten())


    rimglist20 = np.array(rimglist20)

#I ask to the user the number of estimators
    estimators=input('input n_estimators:  ')
    estimators=int(estimators)

#Run the machine learning
    clf = RandomForestClassifier(n_estimators=estimators)
    clf_fit=clf.fit(rimglist80,labelist80)
#make the predictions
    clf_predict = clf.predict(rimglist20)
#make the probability
    clf_proba = clf.predict_proba(rimglist20)
#I save in the database the fit
    joblib.dump(clf,'clfG211117')

    print('score', clf.score(rimglist20,labelist20, sample_weight=None))
    print ('prediction ', clf_predict)
    print ('proba ', clf_proba)
    print ('fit',clf_fit)
    print ('feature importance', clf.feature_importances_)

    plt.ion()

########## HISTOGRAM  ############

#histogram for the visual representation of real and bogus

    plt.hist(clf_proba[:,1],bins=np.arange(0,1,.025), edgecolor='darkred',
     color='indianred')
    plt.xlabel('Hypothesis')
    plt.ylabel('Frequency')
    plt.legend()
    input('return to quit')

#I calculate the number of false positive and the missed detection rate

    fp,md =[],[]

    for h in np.arange(0,1.01,.025):
        hvect = np.zeros(len(labelist20))
        ii = np.where(clf_proba[:,1]>h)
        hvect[ii] = 1
        jj=np.where((hvect==1)&(labelist20==0))
        fp.append(len(jj[0]))
```

```
            jj=np.where((hvect==0)&(labelist20==1))
            md.append(len(jj[0]))


    fp,md = np.array(fp),np.array(md)
    jj=np.where(labelist20==0)
    fp = fp/float(len(jj[0]))
    jj=np.where(labelist20==1)
    md = md/float(len(jj[0]))

    estimators=str(estimators)
    np.savez(n+'x'+n+'est'+estimators,fp=fp,md=md,clf_proba=clf_proba)

    for i in range(len(fp)):
        print('{:.2f} {:.2f} {:.2f}'.format(np.arange(0,1.01,.025)[i],md[i],fp[i]))

######### PLOT MDR AND FPR ########

    plt.plot(md,fp,'-',color='indianred')
    plt.xlabel('Missed Detection Rate (MDR)')
    plt.ylabel('False Positive Rate (FPR)')
    plt.legend()

    input('return to quit')





######## FUNCTION CALLING ################

answ=input('''
Tables creation              -------> t
Create the stamps          -------> s
Run the machine learning -------> m
Cross check                   -------> c      ''')


if answ=='t':
    read_negativi(nsel)
    creazione_tabelle()
    n=input('How many pixel for each side do you want for your (square) stamps?')
    n=int(n)
    npix=int(n/2)
    print (npix)
    produzione_stamps(npix)
    machine_learning()
elif answ=='s':
    n=input('How many pixels for each side do you want for your (square) stamps?    ')
    n=int(n)
    npix=int(n/2)
    produzione_stamps(npix)
    machine_learning()

elif answ=='m':
    machine_learning()

else:
    print ('ERROR: wrong value')
```

A.2 Script for plots creation

```python
import os,sys,glob
from astropy.io import fits
from astropy.table import Table, Column,vstack
import numpy as np
import sqlconn
import random
import matplotlib
matplotlib.use ('TkAgg')
import pylab as plt
from sklearn.ensemble import RandomForestClassifier
from sklearn.externals import joblib
from scipy import interpolate
from sklearn.externals import joblib

####### SCRIPT FOR THE CREATION OF THE PLOTS  #######

#plot missed detection rate and false positive rate with different stamp sizes
def plot_mdr_fpr():

    #load the value of fp and md for stamps with 2 px for each side
    two= np.load('2x2.npz')
    fp2 = two['fp']
    md2 = two['md']

    #load the value of fp and md for stamps with 4 px for each side
    four= np.load('4x4.npz')
    fp4 = four['fp']
    md4 = four['md']

    #load the value of fp and md for stamps with 10 px for each side
    ten= np.load('10x10.npz')
    fp10 = ten['fp']
    md10 = ten['md']

    #load the value of fp and md for stamps with 20 px for each side
    twenty= np.load('20x20.npz')
    fp20 = twenty['fp']
    md20 = twenty['md']

    #load the value of fp and md for stamps with 30 px for each side
    thirty= np.load('30x30.npz')
    fp30 = thirty['fp']
    md30 = thirty['md']

    #load the value of fp and md for stamps with 40 px for each side
    fourty= np.load('40x40.npz')
    fp40 =fourty['fp']
    md40 = fourty['md']

    #load the value of fp and md for stamps with 50 px for each side
    fifty= np.load('50x50.npz')
    fp50 = fifty['fp']
    md50 = fifty['md']

    plt.ion()
    plt.plot(md2,fp2,'-', label='2x2')
    plt.plot(md4,fp4,'-', label='4x4')
    plt.plot(md10,fp10,'-', label='10x10')
    plt.plot(md20,fp20,'-', label='20x20')
```

```
    plt.plot(md30,fp30,'-', label='30x30' )
    plt.plot(md40,fp40,'-', label='40x40')
    plt.plot(md50, fp50,'-', label='50x50')
    plt.axis([0.,.5,0.,.5])

    plt.xlabel('Missed Detection Rate (MDR)')
    plt.ylabel('False Positive Rate (FPR)')
    plt.legend()

    input('return to quit')

#script for the creation of md fp plot with different value of n_estimators
def changing_estimators():

    flist = np.load('imglist.npz')
    #load different values
    clf100 = np.load('20x20est100.npz') #load n_est=100
    md100 = clf100['md']
    fp100 = clf100['fp']
    clf1k = np.load('20x20est1000.npz') #load n_est=1000
    md1k = clf1k['md']
    fp1k = clf1k['fp']
    clf10 = np.load('20x20est10.npz') #load n_est=10
    md10 = clf10['md']
    fp10 = clf10['fp']
    clf50 = np.load('20x20est50.npz') #load n_est=50
    md50 = clf50['md']
    fp50 = clf50['fp']

    plt.ion()
    plt.plot(md100,fp100,'-', label='100' )
    plt.plot(md1k,fp1k,'-', label='1000')
    plt.plot(md50, fp50,'-', label='50')
    plt.plot(md10, fp10,'-', label='10')
    plt.axis([0.,.5,0.,.5])

    plt.xlabel('Missed Detection Rate (MDR)')
    plt.ylabel('False Positive Rate (FPR)')
    plt.legend()




######## FUNCTION CALLING ################
plot_mdr_fpr()
changing_estimators()
```

A.3 Script for model evaluation

```
import os,sys,glob
from astropy.io import fits,ascii
from astropy.table import Table, Column,vstack
import numpy as np
import sqlconn
import random
import matplotlib
matplotlib.use ('TkAgg')
import pylab as plt
from sklearn.ensemble import RandomForestClassifier
from sklearn.externals import joblib
from scipy import interpolate
```

```python
from sklearn.externals import joblib
import collections

#parameters used in all the script
npix = 10
score_limit = 30
use_clf = 'clfG211117'
show_plot = True
hlimit = 0.35

def produzione_stamps():

    #path
    _ff = '/data01/padova/PDdiff/GW170814/global/global_GW170814_r-60.fits'
    tab = Table.read(_ff,format='fits')

    tab['epoch']=tab['epoch'].astype(str)
    for i in range(len(tab)):
        tab['epoch'][i] = tab['epoch'][i][:4]+'-'+\
            tab['epoch'][i][4:6]+'-'+tab['epoch'][i][6:]


    data = os.path.expandvars("$gw_dataou")
    _gwdir = data+'GW170814/'

    imglist = []
    #selection of obj with score>score limit
    ii = np.where(tab['Ranking']>=score_limit)[0]
    for i in ii:
        pointing = (tab['pointing'][i])
        lista = glob.glob(_gwdir+tab['epoch'][i]+'/diff_GW170814_VST_r_'+\
                tab['epoch'][i].replace('-','')+'*'+pointing+'.fits')

        if len(lista)>1: print ("ERROR:  more than one file")
        hdr = fits.open(lista[0])

        #coordinates selection
        xcentre=tab['X_IMAGE_1'][i]
        ycentre=tab['Y_IMAGE_1'][i]

        #stamps creation
        xplus= int(xcentre) + npix
        xminus= int(xcentre) - npix
        yplus= int(ycentre) + npix
        yminus= int(ycentre) - npix

        limy,limx = hdr[0].data.shape

        #check if the obj are in the edge
        if xminus>=0 and xplus<=limx-1 and yminus>=0 and yplus<=limy-1:
            imglist.append(hdr[0].section[yminus:yplus,xminus:xplus])

    imglist = np.array(imglist)
    print("####  selected ",len(imglist),'stamps')
    np.savez('imglistGW'+str(score_limit)+'.npz',imglist=imglist)


def model_implementation():

    #load stamps
    flist = np.load('imglistGW'+str(score_limit)+'.npz')
    imglist = flist['imglist']
```

```python

    _ff = '/data01/padova/PDdiff/GW170814/global/global_GW170814_r-60.fits'
    tab = Table.read(_ff,format='fits')
    #canidates selection by ranking value
    ii = np.where(tab['Ranking']>=score_limit)
    stab = tab[ii]

    #images flattened
    rimglist= []
    for img in imglist:
        rimglist.append(img.flatten())
    rimglist = np.array(rimglist)

    #load clf
    clf = joblib.load(use_clf)
    #predictions
    clf_predict = clf.predict(rimglist)
    clf_proba = clf.predict_proba(rimglist)

    #histogram
    if show_plot:
        plt.ion()
        plt.hist(clf_proba[:,1],bins=np.arange(0,1,.025), edgecolor='darkred', color='indianred')
        plt.xlabel('Hypothesis')
        plt.ylabel('Frequency')
        plt.legend()
        input('return to quit')

    stab['h'] = clf_proba[:,1]
    stab.write('classif'+str(score_limit)+'.fits',format='fits')


def cross_check():

    _ff = 'classif'+str(score_limit)+'.fits'
    stab = Table.read(_ff,format='fits')
    stab.sort('Ranking')
    stab.reverse()
    stab['ordine'] = np.arange(len(stab))+1

    #selection on the candidas by the h value
    ii = np.where(stab['h']>hlimit)
    htab = stab[ii]

    #loading of the tables with the already classified objects. It contains the list of only the real transients
    ff = open('full_list_paper.asc')
    righe = ff.readlines()
    ras,decs = [],[]
    for r in righe[1:]:
        ras.append(float(r.split()[0].split('-')[0]))
        decs.append(-float(r.split()[0].split('-')[1]))

    n =0
    #cross check of the candidates selected by the RF with the ones manual selected
    for i in range(len(ras)):
        ram,decm = htab['X_WORLD_1'],htab['Y_WORLD_1']
        #comparing the object by the distance, when this is <1 arcsec the object is the same
```

```
            dist = np.sqrt(((ras[i]-ram)*np.cos(decs[i]*np.pi/180.))**2+(decs[i]-decm)**2)
            imin = np.argmin(dist)
            if dist[imin]< 1/3600.:
                print (htab['ordine'][imin],ras[i],decs[i],htab['h'][imin])
                n += 1
            else:  print('missed',ras[i],decs[i])
    print('matched={} total={} with h>{} (full table={})'.format(n,len(htab),hlimit,len(stab)))

####### FUNCTION CALLING #########

answ ='q'
while answ not in 'smc':
    answ=input('''
Create the stamps         ------> s
Run the machine learning ------> m
Cross check               ------> c  ''')

if answ=='s':    produzione_stamps()
elif answ=='m':  model_implementation()
elif answ=='c':  cross_check()
```

Appendix B

Tables

h	*MD*	*FP*
0.00	0.00	0.98
0.03	0.00	0.95
0.05	0.00	0.91
0.08	0.00	0.86
0.10	0.01	0.75
0.12	0.01	0.67
0.15	0.03	0.57
0.18	0.03	0.49
0.20	0.04	0.38
0.23	0.05	0.32
0.25	0.07	0.25
0.28	0.08	0.21
0.30	0.08	0.18
0.33	0.09	0.17
0.35	0.10	0.17
0.38	0.11	0.16
0.38	0.11	0.16
0.40	0.13	0.14
0.43	0.13	0.14
0.45	0.14	0.13
0.48	0.15	0.13
0.50	0.16	0.12
0.53	0.17	0.11
0.55	0.18	0.11
0.58	0.20	0.10
0.60	0.23	0.09
0.62	0.24	0.09
0.65	0.27	0.09
0.68	0.28	0.08
0.70	0.30	0.08
0.73	0.32	0.08
0.75	0.36	0.07
0.78	0.40	0.07
0.80	0.44	0.06
0.83	0.48	0.06
0.85	0.56	0.05
0.88	0.62	0.04
0.90	0.72	0.03
0.93	0.79	0.02

0.95	0.90	0.01
0.98	0.97	0.00
1.00	1.00	0

TABLE B.1: Values of MD, FP in function of the threshold h.

h	*MD*	*FP*
0.00	0.00	0.96
0.03	0.00	0.91
0.05	0.00	0.87
0.08	0.00	0.83
0.10	0.01	0.74
0.12	0.02	0.65
0.15	0.02	0.49
0.18	0.03	0.40
0.20	0.04	0.27
0.23	0.05	0.22
0.25	0.07	0.16
0.28	0.08	0.11
0.30	0.10	0.08
0.33	0.13	0.07
0.35	0.15	0.06
0.38	0.16	0.06
0.40	0.18	0.06
0.43	0.21	0.05
0.45	0.21	0.05
0.48	0.23	0.05
0.50	0.26	0.05
0.53	0.28	0.05
0.55	0.29	0.05
0.58	0.30	0.04
0.60	0.33	0.04
0.62	0.35	0.03
0.65	0.37	0.03
0.68	0.39	0.03
0.70	0.42	0.02
0.73	0.45	0.02
0.75	0.48	0.02
0.78	0.51	0.02
0.80	0.56	0.02
0.83	0.61	0.02
0.85	0.67	0.01
0.88	0.73	0.01
0.90	0.81	0.01
0.93	0.87	0.00
0.95	0.95	0.00
0.98	0.98	0.00
1.00	1.00	0.00

TABLE B.2: Magnitude limited *classifier*: values of MD, FP in function of the threshold h.

Bibliography

Aasi, J et al. (2015). "Advanced LIGO". In: *Quantum Grav. 32 074001*. URL: http://iopscience.iop.org/article/10.1088/0264-9381/32/7/074001.

Abbott, P. et al. (2016). "Observation of Gravitational Waves from a Binary Black Hole Merger". In: URL: https://arxiv.org/abs/1412.1488.

Acernese, F et al. (2014). "Advanced Virgo: a second-generation interferometric gravitational wave detector". In: *Quantum Grav. 32 024001*. URL: http://iopscience.iop.org/article/10.1088/0264-9381/32/2/024001.

Bertin, E. and S. Arnouts (1996). "SExtractor: Software for source extraction." In: *Astronomy and Astrophysics Supplement,* URL: https://aas.aanda.org/articles/aas/abs/1996/08/ds1060/ds1060.html.

Bloom, J. S. et al. (2012). "Automating Discovery and Classification of Transients and Variable Stars in the Synoptic Survey Era". In: *Publications of the Astronomical Society of the Pacific* 124.921, p. 1175. URL: http://stacks.iop.org/1538-3873/124/i=921/a=1175.

Botticella, M.T. et al. (2016). "The Pan-STARRS1 Surveys". In: URL: https://arxiv.org/abs/1610.01176.

Breiman, Leo (2001). "Random Forests". In: URL: https://doi.org/10.1023/A:1010933404324.

Brink, Henrik et al. (2013). "Using machine learning for discovery in synoptic survey imaging data". In: *Monthly Notices of the Royal Astronomical Society* 435.2, pp. 1047–1060. DOI: 10.1093/mnras/stt1306. eprint: /oup/backfile/content_public/journal/mnras/435/2/10.1093/mnras/stt1306/2/stt1306.pdf. URL: http://dx.doi.org/10.1093/mnras/stt1306.

Brocato, E. et al. (2017). "GRAWITA: VLT Survey Telescope observations of the gravitational wave sources GW150914 and GW151226". In: URL: https://arxiv.org/abs/1710.05915.

Buisson, L. du et al. (2015). "Machine Learning Classification of SDSS Transient Survey Images". In: *Monthly Notices of the Royal Astronomical Society, 2015, 454 (2): 2026-2038*. URL: https://arxiv.org/abs/1407.4118.

Capaccioli, M. and P. Schipani (2011). "The VLT Survey Telescope Opens to the Sky: History of a Commissioning". In: *The Messenger, vol. 146, p. 2-7*. URL: https://ui.adsabs.harvard.edu/#abs/2011Msngr.146....2C/abstract.

Cappellaro, E. et al. (2015). "Supernova rates from the SUDARE VST-Omegacam search. I". In: URL: https://arxiv.org/abs/1509.04496.

Carliles, Samuel et al. (2010). "Random Forests for Photometric Redshifts". In: *The Astrophysical Journal* 712.1, p. 511. URL: http://stacks.iop.org/0004-637X/712/i=1/a=511.

Cicco, D. De et al. (2015). "Variability-selected active galactic nuclei in the VST-SUDARE/VOICE survey of the COSMOS field". In: *AA 574, A112 (2015)*. URL: https://arxiv.org/abs/1412.1488.

Gibson, Adam and Josh Patterson (2017). *Deep Learning*. O'Reilly Media, Inc.

Grado, A. et al. (2012). “VST processing facility: first astronomical applications”. In: *Memorie della Societa Astronomica Italiana Supplement, v.19, p.362 (2012)*. URL: http://adsabs.harvard.edu/abs/2012MSAIS..19..362G.

Ho, Tin Kam (1995). “Random decision forests”. In: *IEEE Computer Society Washington, DC, USA ©1995*. URL: http://dl.acm.org/citation.cfm?id=844379.844681.

Kotsiantis, S. B. (2007). “Supervised Machine Learning: A Review of Classification Techniques”. In:

Richards, Joseph W. et al. (2011). “On Machine-learned Classification of Variable Stars with Sparse and Noisy Time-series Data”. In: *The Astrophysical Journal* 733.1, p. 10. URL: http://stacks.iop.org/0004-637X/733/i=1/a=10.

Singer, Leo P. et al. (2014). “The First Two Years of Electromagnetic Follow-Up with Advanced LIGO and Virgo”. In: *2014 ApJ 795 105*. URL: https://arxiv.org/abs/1404.5623.

What are Gravitational Waves? https://www.ligo.caltech.edu/page/what-are-gw.

Why Detect Them? https://www.ligo.caltech.edu/page/why-detect-gw.

Wright, D. E. et al. (2015). “Machine learning for transient discovery in Pan-STARRS1 difference imaging”. In: *Monthly Notices of the Royal Astronomical Society* 449.1, pp. 451–466. DOI: 10.1093/mnras/stv292. eprint: /oup/backfile/content_public/journal/mnras/449/1/10.1093/mnras/stv292/2/stv292.pdf. URL: http://dx.doi.org/10.1093/mnras/stv292.