Work in Classification

Simone Albanesi,[1,*] Marina Berbel,[2,†] Marco Cavaglià,[3,‡] Lorena Magaña Zertuche,[4,§] Miquel Miravet-Tenés,[5,¶] Dimitra Tseneklidou,[6,**] and Yanyan Zheng[3,††]

[1] *Dipartimento di Fisica, Universit'a di Torino & INFN Sezione di Torino, via P. Giuria 1, 10125 Torino, Italy*
[2] *Departament de Matemàtiques, Universitat Autònoma de Barcelona, 08193 Bellaterra, Spain*
[3] *Institute of Multi-messenger Astrophysics and Cosmology & Physics Department, Missouri University of Science and Technology, Rolla, MO 65409, USA*
[4] *Department of Physics and Astronomy, University of Mississippi, University, Mississippi 38677, USA*
[5] *Departament d'Astronomia i Astrofísica, Universitat de València, Dr. Moliner 50, 46100, Burjassot (València), Spain*
[6] *Theoretical Astrophysics Department, Eberhard-Karls University of Tübingen, Tübingen 72076, Germany*

(Dated: October 14, 2022)

Abstract goes here.

I. INTRODUCTION

In our introduction we want to start from big picture (LIGO, ML, pipelines) and narrow down to what work we are presenting here and why it is important. We also describe in which ways it is novel and how it compares to previous works like in [**?**]. We may also want to cite [**?**].

Maybe starting from the events that could lead to an E/M counterpart Introduce the probabilities HasRemnant, HasNS...

II. IMPROVING BINARY CLASSIFICATION THROUGH ML

In this section we may want to expand on the idea of using classification itself. We should mention which algorithms we attempted to use and why they did not work. Then we can talk more in detail about the ones that did work in the following subsections.

A. K-Nearest Neighbors (KNN)

One of the non-parametric algorithms we have used for classification is the `KNeighborsClassifier` (KNN). This algorithm assumes that similar things are near to each other. It captures the idea of similarity by computing the distance between points in a graph. It is also a lazy learning algorithm, because it doesn't have a training phase, it uses all the data for training while classification -the dataset is stored and at the time of classification, it acts on the dataset.

The workflow of the KNN algorithm is the following: first, after splitting the data into training and testing sets, we need to select the number of neighbors, K, which can be any integer. Then, for each point of the testing dataset we compute the distance between the point and each row of the training data with a chosen metric (Euclidean, Manhattan, Chebysev, Mahalanobis), and we sort the points in ascending order based on the distance value. By choosing the top K neighbors from the sorted array, we assign a class to the test point, which is the most frequent class of the chosen neighbors.

B. Random Forest (RF)

III. DATA CONDITIONING

We should in principle condition the data in the same way but any differences can also be described here. Talk here about the injection campaign and how the data is splitted into training and testing datasets (?).

IV. RESULTS

Here we talk about overall results and specifically from each algorithm in the subsections below. [MMT: Below I described how to check the performance of the algorithms. Maybe a table with all the scores/sensitivities/precisions from both KNN and RF would be useful (already got it in a google doc)]

To measure the performance of the classifiers we use some common statistical quantities. The score is the number of correctly predicted events over the number of total events (a perfect classifier has a score of 1). It works best when there is an equal number of events for each label in the training set. It does not consider the importance of misclassification, or that the training data can be biased towards one specific label.

The mean score is computed by training the algorithm on the 90% of the dataset and testing it on the remaining 10%, cycling the train/test combination over the full dataset. To do that, we are going to use the training dataset, since it's the larger one. In order to train and

* simone.albanesi@edu.unito.it
† mberbel@mat.uab.cat
‡ cavagliam@mst.edu
§ lmaganaz@go.olemiss.edu
¶ miquel.miravet@uv.es
** dimitra.tseneklidou@uni-tuebingen.de
†† zytfc@umsystem.edu

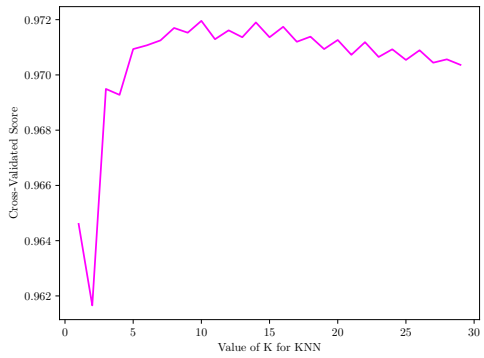

FIG. 1. Score of our KNN model as a function of the number of neighbors. We are considering *HasNS*.

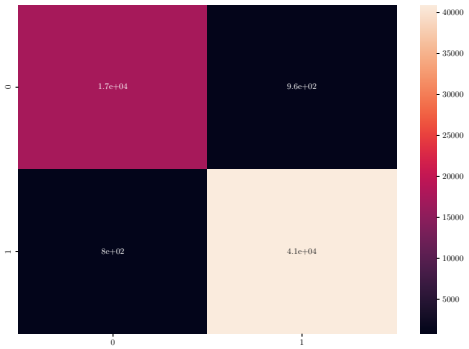

FIG. 2. Confusion matrix for our model for *HasNS*, using the independent recovered values.

test the model and create the different plots, we are going to use the training and testing files.

Another useful quantity is the sensitivity. It is the ratio between the true positives and the sum of the true positives and false negatives. It measures how much the algorithm predicts *true* (in our case it would be that the event has NS or has REM), when it is actually *true*.Having a sensitivity equal to 1 would mean that our method predicts *true* for every event. Therefore, a method with high sensitivity will barely miss true alarms.

A quantity that measures how much you can trust a method when it predicts *true* is the precision. It is the ratio between the true positives and the some of the true and the false positives. A precision equal to 1 means that the method never predicts *true* when it is actually *false*. This means that the method will never give false alarms.

A. KNN Results

1. *Has NS*

The metric we use to compute the distance between neighbors is the *Manhattan* metric (or the Minkowski's $L1$ distance), which is the distance between two points measured along axes at right angles. Having $p_1(x_1, y_1)$ and $p_2(x_2, y_2)$ the distance will be

$$d = |x_1 - x_2| + |y_1 - y_2| \,. \tag{1}$$

Moreover, the points are weighted uniformly. After applying cross-validation, we get that the optimal number of neighbors is $K_{\rm NS} = 10$, with a mean score $\mathrm{s_m} = 0 : 9718355224352762$ and a testing score $\mathrm{s_t} = 0.9723828730478842$. In Fig. 1 you can find how the mean score changes with the number of neighbors of the algorithm.

The confusion matrix appears in Fig. 2. It depicts that the number of events that are correctly classified when

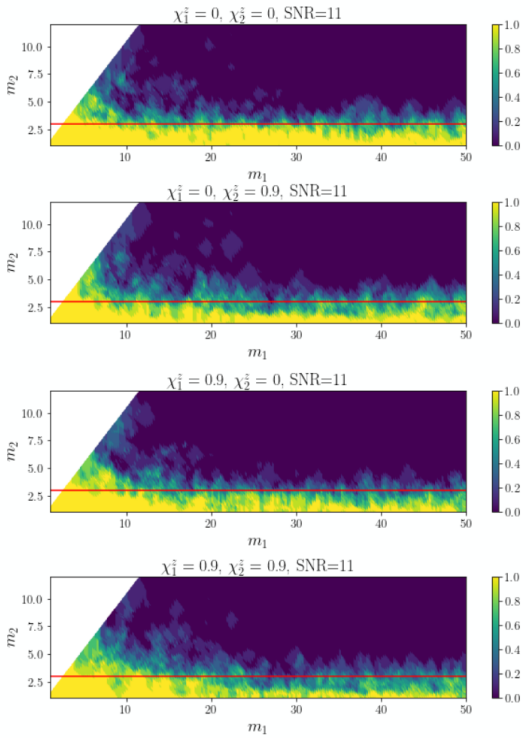

FIG. 3. Probability of having a remnant as a function of the values of the masses. The different panels show the results for different spins. The solid red line depicts the threshold mass for m_2.

there is a NS is much higher than the ones without a NS. The probability as a function of m_1 and m_2 is shown in Figs. 3, There are no big differences with different values of the spins, but the most remarkable one is that the model classifies better for 0-spin values, especially when m_1 is large. The true and false positive rates in terms of the threshold probability (ROC curve) appear in Fig. 4.

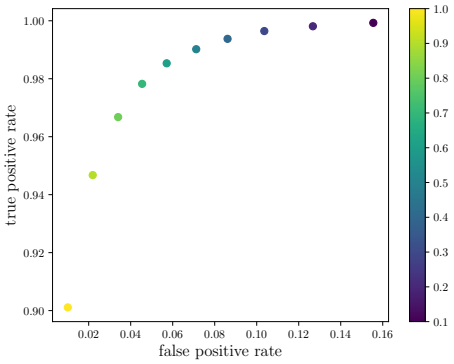

FIG. 4. Relation of the true and false positive rates as a function of the threshold applied to make the decision between having or not having a remnant.

2. *Has REM*

In order to classify the events between having or not a post-merger remnant and therefore a likely EM emission, we apply again the cross validation to get the optimal number of neighbors, which turns out to be $K_{\rm REM} = 6$.

B. RF Results

V. CONCLUSIONS

Reiterate why what we did is important and how it improves current knowledge.

ACKNOWLEDGMENTS

We thank Deep Chatterjee for useful discussions and for sharing his work, which helped us compare our results to those of [**?**]. We also thank Shaon Ghosh for useful discussions and X, Y, and Z for reviewing an earlier version of this manuscript. Part of this research was performed while the authors were visiting the Institute of Pure and Applied Mathematics (IPAM), University of California Los-Angeles (UCLA). The authors would like to thank IPAM, UCLA and the National Science Foundation through grant DMS-1925919 for their warm hospitality during the fall of 2021.

The authors are grateful for computational resources provided by the LIGO Laboratory and supported by the U.S. National Science Foundation Grants PHY-0757058 and PHY-0823459, as well as resources from the Gravitational Wave Open Science Center, a service of the LIGO Laboratory, the LIGO Scientific Collaboration and the Virgo Collaboration.

The work of L.M.Z. was partially supported by the MSSGC Graduate Research Fellowship, awarded through the NASA Cooperative Agreement 80NSSC20M0101. The work of X.Y. was partially supported by NSF Grant No. PHY-20XXXXX. The work of M.M.T. was supported by the Spanish Ministry of Universities through the Ph.D. grant No. FPU19/01750, by the Spanish Agencia Estatal de Investigación (Grants No. PGC2018-095984-B-I00 and PID2021-125485NB-C21) and by the Generalitat Valenciana (Grant No. PROMETEO/2019/071)

All plots were made using the python package `matplotlib` [**?**].

This manuscript has been assigned LIGO Document Control Center number LIGO-P22XXXXX.