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: December 16, 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 [?].

The breakthrough observation of the binary neutron star (BNS) merger simultaneously in gravitational and electromagnetic (E/M) waves [?] marked a new era of multi-messenger astronomy including gravitational waves (GWs). The event GW170817 gave answers to open questions, such as the origin and production of Gamma-ray bursts (*cite), the production of heavy elements during a merger of BNS (*cite), rule out Equations of States (*cite) and alternative theories of gravity (*cite), +?. Among the challenges that it brings is the early alert of the E/M telescopes. The sources that can produce an E/M counterpart are the coalescences of a neutron star with another neutron star (NSNS) or with a black hole (NSBH). In the case of NSNS: GRBs, neutron star. For NSBH: accretion disk –> short GRBs.

- Mention empirical fits, Foucart?
- Real time inference: What was done in O2, O3 (Deep)
- Probabilities HasNS, HasRemnant
- What we suggest, Random Forest classification (what is it, what are the advantages)

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.

III. RANDOM FOREST (RF)

A RF is an ensemble of decision trees. One of its major strengths is that every train trains and classifies independently from the rest, while the RF classification joints all the results and assign as category the mode from the trees. The probability of belonging to a category is therefore straightforward, being the number of trees that chose it divided by the total number of trees. Notice that the training and evaluation of a RF can be accelerated by parallelization, as computations inside each tree are independent from the rest.

The training of a RF is usually done with bootstrap, a technique that assigns a random subset of the training

* 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

dataset to each tree. This prevents overfitting as every individual classifier is not exposed to the same data, and encourages pattern recognition by studying the same data from different subsets. Every decision tree is composed by nodes, where data its splitted until the different categories are separated. At each node, a subset of the features of the data is selected along threshold values that maximize the information gain at the separation. The binary splitting at each node gives the tree its name, as it can be visualized as roots going deeper at separations.

We use the RK implementation in scikitlearn [REF AND DETAILS]. The main hyperparameters to tune in this module are the number of trees, the maximum depth allowed and the information gain criteria used at splitting (two are offered). We have observed that the maximum number of features to be considered in a node can be kept fix as the square root of the total number of features. Given that the aiming of this work is to improve the current low latency classification, the model once trained can occupy a restricted amount of memory. Therefore before searching the optimum hyperparameters for our dataset, we restrict those which make heavier models: the number of trees and their depth. We set to 100 the maximum number of trees the forest may have, and 25 their possible maximum depth.

For the RF we use events with 5 features: the two masses, their corresponding spins and the SNR of the detection. In the tuning of the hyperparameters we measure the performance by its score: the number of events correctly classified against the total number of events in the testing dataset, if threshold is taken as 0.5. As all categories are balanced, this approach is enough to roughly compare models. The best model found achieves a score of *whatever* using *number* of trees, *number* maximum depth and *name* criteria for the information gain.

INFORMATION TO SAY, JUST HERE AND NOT WELL WRITTEN:

For the 23 EoS we try a crossvalidation with:

trees in the forest = [10, 30, 50, 80, 100, 300] (more trees take too much memory) criterion = ‘entropy’ (in all tests, criterion ‘gini’ gives very similar results)

max_features = ‘sqrt’ (in all tests, using a portion of the features instead of all of them in each node gives better results)

max_depth = [15, 25, 35, 45, None] (we would prefer more shallow trees)

Using the complete dataset, and not Sushant partitions, Use first 30

We obtain the following:

And therefore we select 50 trees, 15 depth for all EoS and present the results with that.

IV. DATASET AND LABELING

HasNS true if m2inj<3Ms

HasREM Focault remnant mass >0. Requires compactness so it is model dependent. The O2 dataset used

	Best					S	
EOS	**Trees**	**Depth**	**Score**	**MB**	**MB (c)**	**Trees**	**Depth**
APR4_BB	300	15	0.9683018	94.7	19.7	50	15
BHF_BBB2	80	15	0.9685127	24.4	5.1	300	15
H4	80	15	0.9618587	29.6	6.1	300	15
HQC18	300	15	0.9673755	93.7	19.6	100	15
KDE0V	300	15	0.9673295	92.0	19.3	80	15
KDE0V1	100	15	0.96704954	30.9	6.5	80	15
MPA1	80	15	0.96601225	27.2	5.6	300	15
MS1_PP	300	15	0.96563534	113.5	23.2	80	15
MS1B_PP	300	15	0.96555340	114.2	23.3	100	15
RS	300	15	0.96447350	103.8	21.6	80	15
SK255	300	15	0.96472405	105.8	22.0	100	15

TABLE I. table1 comparison forests

	Best					Seco		
EOS	**Trees**	**Depth**	**Score**	**MB**	**MB (c)**	**Trees**	**Depth**	**S**
SK272	300	15	0.96401816	109.0	22.6	100	15	0.96
SKI2	50	15	0.96242338	18.8	3.9	300	15	0.96
SKI3	50	15	0.96174537	19.0	3.9	100	15	0.96
SKI4	300	15	0.96598969	100.6	20.9	30	15	0.96
SKI5	100	15	0.96343381	38.2	7.8	80	15	0.96
SKI6	300	15	0.96586928	101.7	21.1	30	15	0.96
SKMP	300	15	0.96544567	100.2	20.9	80	15	0.96
SKOP	100	15	0.96610459	32.3	6.8	300	15	0.96
SLy	80	15	0.96728884	25.3	5.3	300	15	0.96
SLY2	100	15	0.96745868	31.8	6.6	80	15	0.96
SLY9	300	15	0.96605993	101.6	21.1	100	15	0.96
SLY230A	300	15	0.96714915	95.5	20.0	100	15	0.96

TABLE II. table2 RF comparison

2H EoS. Later we do the training with 23 EoS and do a weighted average according to the Bayes factor of each EoS, obtaining analogous performance.

Until here is same as Deep.

NEW: Instead of training two different classifiers, one for each feature, as they are in part related (an event cannot have remnant if there is no NS) we build 3 mutually exclusive categories: label 0 if no NS and no remnant, label 1 if NS but no remnant, label 2 if both. This labeling eliminates the posibility of an unphysical classification of the event, where pHasRem>pHasNS, as there is no category for hasREM but no NS. As the categories are mutually exclusive $p(0)+p(1)+p(2) = 1$. Trivially p(HasREM)=p(2). And for the NS p(hasNS)=p(hasNS and hasREM ∪ hasNS and No hasREM)=p(2∪1)=p(2)+p(1)=1-p(0)

HasNS	HasRem	Our label
0	0	0
1	0	1
1	1	2

TABLE III. Labelling adopted for classification of having a NS and having a remnant with the same classifier

We test the performance using random forest (RF) and compare the results with k-nearest neighbors (KNN) [paper Deep] and genetic programming (GP) [paper GP].

140k training samples. 60k for testing. Training/testing dataset: injections in O2 data stream. (Injections time and waveform approximants detailed in Chatterjee et al 2020). Labeling with the 3 categories the classes are pretty even (30.38% for label 0, 36.30% for label 1 and 33.32% for label 2), which is needed for good accuracy on classification. If not, a classifier would always be biased to the category that it has seen the most.

V. 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 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*.

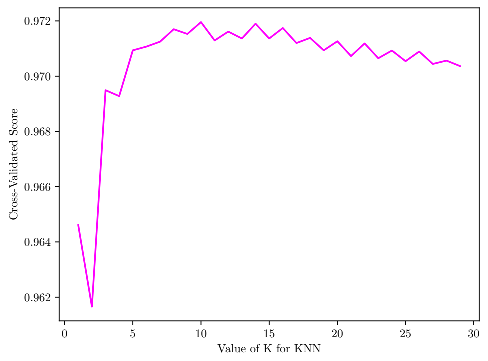

FIG. 1. Score of our KNN model as a function of the number of neighbors. We are considering *HasNS*.

This means that the method will never give false alarms.

Finally, the F1 score $F1 = 2(\text{precision} \times \text{sensitivity})/(\text{precision} + \text{sensitivity})$ is a type of score that takes into consideration how precision and sensitivity compensate each other. A perfect classifier would have a F1 score of 1.

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 $\rm s_m = 0 : 9718355224352762$ and a testing score $\rm s_t = 0.9723828730478842$. 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 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.

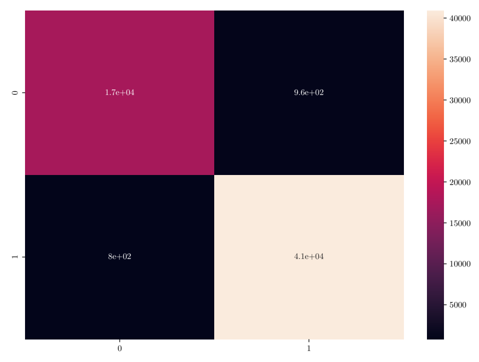

FIG. 2. Confusion matrix for our model for *HasNS*, using the independent recovered values.

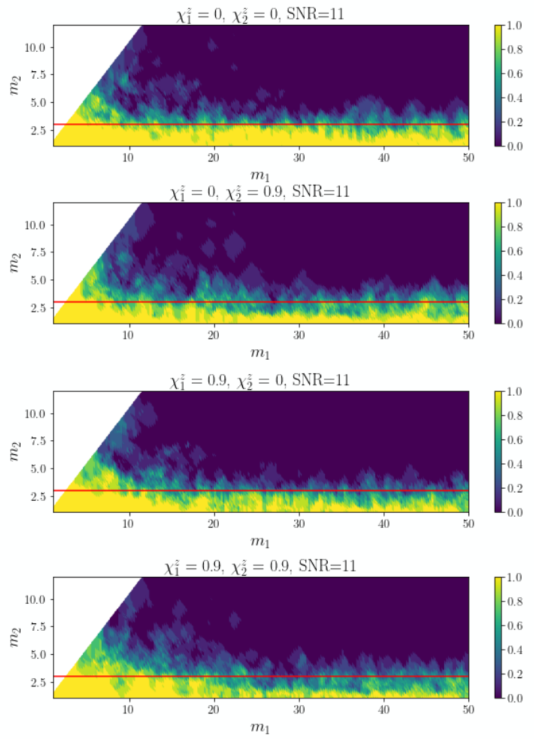

FIG. 3. Probability of having a neutron star in the binary system as a function of the values of the masses. The different panels show the results for different spins. The solid red line depicts the threshold mass for m_2.

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 $K_{\text{REM}} = 6$. Fig. 5 shows the dependence of the mean score with K for this label. In this case, it is clearer that the score peaks at a certain value of K. Another hyperparameter that

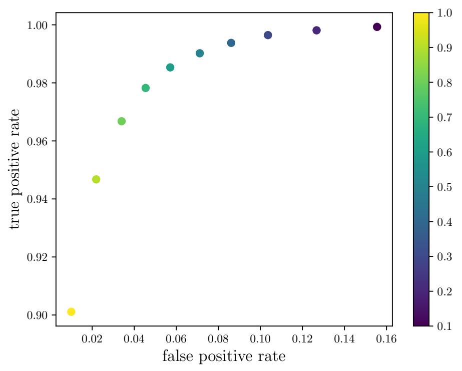

FIG. 4. Relation of the true and false positive rates as a function of the threshold applied to make the decision between having or not having a neutron star.

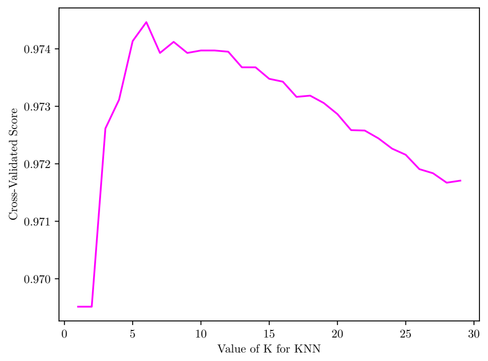

FIG. 5. Score of our KNN model as a function of the number of neighbors. We are considering *HasREM*.

we change is the way the neighbors are weighted; in this case, we weight them with the inverse of the distance.

The confusion matrix depicted in Fig. 6 shows that the algorithm classifies correctly those events that do not have any remnant after the merger. On the other hand, it has more problems when classifying events with a post-merger remnant, i.e., with at least a neutron star with a mass $M_{\text{NS}} \lesssim 2.83\ M_\odot$ [MMT: Maybe this should go somewhere else]. There is also a dependence on the value of the spin that can be seen in Fig. 7 [MMT:it would be nice to use the plots with the red line that Marina got].This dependence is correct, since the probability of having a remnant increases for large values of m_1 at larger spin, but this dependence is given by the EOS. We show in Fig. 8 the ROC curve for this new case.

[MMT: Nothing more to show here. These are the results I got for our KNN. It would be nice to plot the ROC curves for KNN and RF all together, and also the confusion matrices. I would like also to put a table comparing the different values of precision, sensitivity and score for both algorithms. More tests are needed with new datasets for KNN if we want an updated comparison.]

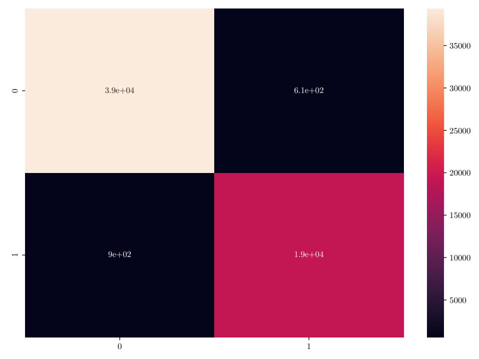

FIG. 6. Confusion matrix for our model for *HasREM*, using the independent recovered values.

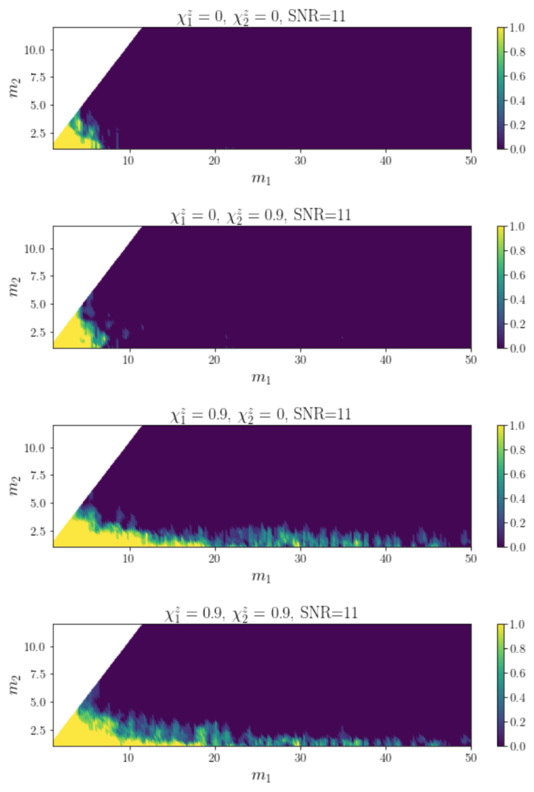

FIG. 7. Probability of having a remnant as a function of the values of the masses. The different panels show the results for different spins.

B. RF Results

C. Comparison between RF and KNN

In order to decide which classification algorithm works better, we compute the different quantitites presented at the beginning of this section, for both *Has NS* and *Has REM* and the two different possible values of the labels: 0 (no NS/no remnant) and 1 (has NS/has remnant). We fix the threshold to 0.5.

We also show in Table x the true positive (TP) and false positive (FP) rates for different choices of the threshold.

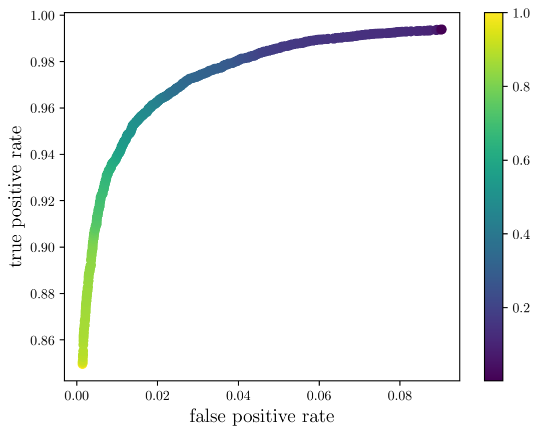

FIG. 8. As in Fig. 4, we show the relation of the true and false positive rates, but in this case as a function of the threshold applied to make the decision between having or not having a remnant object.

Label: 0	*Has NS*		*Has REM*	
	RF	KNN	RF	KNN
Score	0.974	0.972	0.988	0.975
Sensitivity	0.952	0.943	0.991	0.985
Precision	0.964	0.966	0.991	0.978
F1	0.958	0.954	0.991	0.981

TABLE IV. cap

VI. 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

Label: 1	*Has NS*		*Has REM*	
	RF	KNN	RF	KNN
Score	0.974	0.972	0.988	0.975
Sensitivity	0.984	0.985	0.982	0.955
Precision	0.979	0.975	0.982	0.969
F1	0.982	0.954	0.982	0.962

TABLE V. cap

	Has NS				*Has REM*			
	RF		KNN		RF		KNN	
Threshold	TP	FP	TP	FP	TP	FP	TP	FP
0.07	0.999	0.118	1.000	0.000	0.998	0.033	0.994	0.090
0.27	0.994	0.076	0.998	0.134	0.992	0.015	0.981	0.040
0.51	0.984	0.048	0.990	0.072	0.981	0.009	0.953	0.014
0.80	0.961	0.020	0.981	0.053	0.959	0.004	0.904	0.004
0.94	0.925	0.007	0.965	0.034	0.925	0.002	0.850	0.001

TABLE VI. cap

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)

The work of M.B. was supported by the Spanish Agencia Estatal de Investigación (Grants No. PID2020-118236GB-I00).

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

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