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, Università 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: February 12, 2023)

Abstract goes here.

I. INTRODUCTION

The anticipated fourth observing run (O4) by LIGO, Virgo and KAGRA (LVK) gravitational wave (GW) detectors is expected to contribute significantly in the already established field of multi-messenger astronomy. The simultaneous detection of a binary neutron star (BNS) merger in gravitational and electromagnetic (EM) waves in 2017 (GW170817) [**?**] gave answers to long-lasting problems, such as the origin and production of short Gamma Ray Bursts (GRBs) [**?**], the production of heavy elements during a merger of BNS [**?** **?** **?** **?**], ruled out Equations of State (EoS) [**?**] and alternative theories of gravity [**?**]. Such simultaneous observations are crucial for a variety of scientific fields, like astrophysics, particle physics, cosmology, and therefore the necessity to coordinate them successfully emerges. Among the challenges that this attempt brings is the early alert of the EM telescopes/facilities from the GW detectors.

Some of the expected sources to produce an EM counterpart are the compact binary coalescences (CBC), such as a neutron star with another neutron star (NSNS) or with a black hole (NSBH). In the case of NSBH, tidally disrupted matter could form an accretion disk, with high temperature, around the BH where processes like pair production and annhilation could power a short GRB. If the ejecta is unbound, r-process neucleosynthesis could power a kilonova [**?** **?** **?** **?** **?** **?**]. In a BNS, even if tidal forces are not strong enough, the expulsion of neutron rich material after the physical contact during merger can result in a kilonova, as seen in the case of GW170817 [**?** **?** **?** **?** **?** **?** **?**]. There could also be relativistic outflows producing a GRB, like in the case of GW170817, or a prompt collapse which could supress the GRB [**?**]. Therefore, many complex physical processes could lead to EM emission and the presence of post-merger mass remnant is a common factor.

Numerical relativity (NR) simulations can provide accurate information about the coalesence of the binary system and its aftermath/remnant. However, these simulations are computationally expensive and require a lot of time, making them inapropriate to alert follow-up EM searches. Another approach to computing remnant matter is via emperical fits of NR results, which provides an expression for the remnant mass as shown in [**?** **?**]. These fits, however, demand more information (like the EoS) than what is available at real-time in detector data [**?**]. Another factor adding to the difficulties of estimating the post-merger remnant matter is the inaccuracies of the source parameters calculated by modeled search pipelines in real-time. Consequently, it is important to alert the EM ground and space telescopes quickly and accurately in order to have synchronized EM observations.

In the previous observing runs in LIGO and Virgo there were used different matched-filtering pipelines in low latency in order to detect GWs from CBC, such as the GstLAL-based inspiral pipeline (GstLAL) [**?**], the PyCBCLive [**?**], the MBTAOnline [**?**]. These searches are based on a discrete template bank of CBC waveforms that provide the masses, the dimensionless aligned/anti-aligned spins of the objects along the orbital angular momentum. The best matching template was performed through a detection statistic during the second observing run (O2), which was the first time that real-time data was made available to EM observatories to support follow-up observations. Classification of the binary systems was performed according to the following probabilities: the probability that at least one of the objects in the binary is a NS, p(HasNS) and the probability that there will be a matter remnant p(HasRemnant). For instance, in a BNS merger both probabilities will be equal to one, while in a binary black hole (BBH) merger, both probabilities will be zero. In [**?**] the authors approached the problem as binary classification and implemented machine learning (ML) supervised algorithms, such as KNeighborClassifier (KNN) [**?**] and Random Forest (RF) classifiers. These methods were used during the O3 to calculate the probability that an EM-bright progenitor is present in the

* 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

system.

In this work, we revisit and extend the problem of classification in low-latencyby exploring novel algorithms and new classification schemes . First, we propose the use of a combined probability of whether there is a NS in the event and whether there is a remnant. By doing so, we exclude the unphysical situation of having a larger probability of having a remnant than having a NS. In addition, we perform a thorough study of different algorithms, perform several tests and account for different EoSs. We provide a direct comparison between the methods we used, as well as the ones already implemented e.g. in [**?**].

Different methods of classification were considered, such as KNN, RF, genetic programming Support Vector Machine and were tested for the same data. Here, we present the ones that performed best, the RF, genetic programming and KNN. The last one is also used in [**?**] and thus we provide a direct comparison.

The structure of the paper is as follows. In Section 2 there is a detailed description of the data used, as well as the labeling introduced. In Section 3, the classification methods, namely RF, KNN, genetic programming are presented. The 4th Section is dedicated to the results of each method and their comparison. Lastly, there is a summary of the work carried out in this paper and the suggestion to use these methods in the next observing runs.

II. DATASET AND LABELING

In this section, we discuss the data conditioning process in our study. The data we use for training and testing are the same injections during O2 as in [REF Deep's paper], also aiming to predictict the probability of a detected event of having a neutron star (HasNS) and/or a remnant object (HasREM) that could emit an electromagnetic counterpart. As we utilize supervised ML techniques, we first label the data accordingly.

HasNS is considered true if $m2_{inj} < 3M\odot$ as in [REF]. For HasREM and following [REF] we use the Focault formula for the remnant mass and if the result is greater than 0 we label the event as true . However, this formula depends on the Equation of State (EoS) as it utilizes the compactness of the NS. The injections used in our study utilize the 2H EoS, but we relabel the data with a different one as explained later, as the EoS used for labelling the training data will affect the ability of predicting correctly a real event.

Predicting the presence of a NS and the creation of a remnant object like this requires a binary classifier for each of the two tasks. We employ a new approach that utilizes the relationship between HasNS and HasREM to train a single multilabel classifier. We relabel the data into 3 mutually exclusive categories: label 0 if there is no NS and no remnant, label 1 if there is a NS but no remnant, and label 2 if there are both. Our labelling is summarized in table I. This labeling eliminates the possibility of an unphysical classification of the event, where $p(HasREM) > p(HasNS)$, as there is no category for HasREM but no NS.

As the categories are mutually exclusive, $p(0) + p(1) + p(2) = 1$. Therefore, p(HasREM)=p(2) and for the NS, p(hasNS)=p(hasNS and hasREM $\cup$ hasNS and No hasREM)=p(2$\cup$1)=p(2)+p(1)=1-p(0). This approach allows us to use a single classifier while avoiding any unphysical classifications without further treatment of the data or the output.In this work, RF and KNN were trained on data labeled as in I, however, genetic programming algorithm used in this work was trained on binary labels for hasNS and hasRemnant separately because of its limitations in performing multi-label classification.

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

TABLE I: Labelling adopted for classification of having a NS and having a remnant with the same classifier

In order to avoid conditioning the results of HasREM too heavily to the equation of state (EOS) selected for training, we labeled the dataset with the 23 EOS selected by the LIGO-Virgo collaboration in [ref] for studies of neutron stars. These EOS cover a wide parameter space, making together a robust estimation for the neutron star composition.

Each algorithm presented here was trained for each of the 23 EOS and predictions were obtained. The final prediction provided is a weighted average between the results of all of them, with the weights determined by the Bayes factor of each EoS, as explained in [ref2]. This approach allows for the consideration of multiple EOS and their relative likelihoods, resulting in a more robust and accurate prediction.

III. CLASSIFICATION ALGORITHMS

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 does not 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)

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 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 [**?**]. 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 300 the maximum number of trees the forest may have, and 45 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.

C. Genetic Programming

Genetic Programming(GP) is a supervised ML algorithm [**?**].It is an evolutionary computation that employs naturally occurring genetic operations, a fitness function, and multiple generations of Darwinian evolution to resolve a user-defined task [**?**]. GP can be used to discover a mathematical relationship between the input variables, also called features, in data (regression) or group data into categories (classification). Similar to biological natural selection, GP learns how well the program functions by comparing the output of its multivariate expression to a *fitness score*. Programs with high fitness score are most likely, but not certain, to propagate to next generation. With subsequent generations, programs are more likely to get better at solving the task at hand [**?**].

GP multivariate expressions are classically represented as a syntax tree, where the trees have a root (top center), nodes (mathematical operators), and leaves (operands). Operators can be arithmetic, trigonometric, boolean, etc. and the operands are place-holder variables related to the problem. The tree depth can be varied aiding in evolution of more complex multivariate expression. GP intitially generates a stochastic poplulation of trees, computes fitness scores and randomly selects trees for comparison. The lead scoring trees are selected and one of the genetic operators (reproduction, mutation, crossover) is randomly applied to carry them forward to next generation. The process repeats until user-defined conditions are met. The run-time parameters like intial population size, tree depth, tournament size for competing trees, number of generations, and termination criterion can be tuned for better algorithmic performance.

The unique aspect of GP algorithm used here is the transparency of its internal workings. Unlike many black box ML algorithms, the evolutionary process can be reviewed at each step which might be quite important to many researchers.[**?**]

For our analysis, we used an open source python code, Karoo GP [**?**]. Karoo GP package is scalable, with multicore and GPU support enabled by the TensorFlow library and has capacity to work with very large datasets. The latest version of Karoo GP can be found in PyPI [**?**].

IV. RESULTS

In order to decide which method gives a better performance in classifying this kind of events, we can apply them over testing data and finally do a comparison between both. A way to see how data is classified we can construct histograms where the number of events that are classified with a label (`HasNS/HasRemnant`) `True` or `False` will change with a given threshold of the probability. For an algorithm with perfect performance, all the events with label `True (False)` should be at $p(\texttt{label}) = 1$ ($p(\texttt{label}) = 0$).

Another way to check the algorithm's performance is by building the so-called *Receiver Operating Characteristic (ROC) Curve*. They show the variation of the true-positive rate (or efficiency) with the false-positive rate given a certain threshold for the probability. An algorithm with a proper performance will give a steeper ROC curve, or in other words, will have a higher eficiency with a lower false-positive rate.

In the ROC curves that we will present in the following subsections, we highlight three reference EoS in color, from which we show results in more detail. We select BHF_BBB2 because is the model that give the lowest maximum mass, MS1_PP as the model with the bigger maximum mass, and we also include SLy because is the most accepted EoS for NS modeling (reference), and is the one that was used in the injections that are our dataset.

Here are the results for both methods:

A. KNN Results

We apply cross validation in order to fix the different hyperparameters of the algorithm. To do so, we compute the score over a range of different parameters. We consider a number of neighbors between 1 and 20; the different metrics we test are the `euclidean`, `manhattan` and `cityblock`; the algorithms to compute the nearest neighbors can be `BallTree`, `KDTree` and the brute-force search, and the possible weight functions are the uniform weights (all points are weighted equally) and the *distance* weights (points are weighted by the inverse of their distance).

Considering all these possibilities, we apply the `cross val score` function from `scikit-learn` using a 10-fold cross validation to all the 23 datasets with different EOS. We find that the optimal metric, algorithm and weights are the same for all the EOS, being the `manhattan` metric, the `BallTree` algorithm (with a leaf size of 30) and the *distance* weights. The only parameter that differs is the number of neighbors, that goes from 8 to 12. One can find the optimal hyperparameters and the corresponding score for each EOS in Table II. Doing an average weighting by the Bayes factor of each EOS, we finally get an optimal number of neighbors of 10. The mean score from the cross validation goes from 0.941 (for H4 EOS) to 0.953 (for APR4 EPP), as one can see in Table II.

Considering now the SLy EOS, the model with these parameters gives a confusion matrix that is shown in Fig. **??**. The probability of having a NS as a function of m_1 and m_2 is shown in Fig. **??**. 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. There is also a dependence of the probability of having a remnant on the value of the spin that can be seen in Fig. **??**.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.

EOS	Score	$K_{\text{neighbors}}$	Bayes Factor
APR4 EPP	0.953	10	1.526
BHF BBB2	0.951	8	1.555
H4	0.941	10	0.056
HQC18	0.950	10	1.422
KDE0V	0.951	8	1.177
KDE0V1	0.949	10	1.283
MPA1	0.951	14	0.276
MS1 PP	0.948	12	0.001
MS1B PP	0.947	11	0.009
RS	0.945	10	0.176
SK255	0.946	10	0.179
SK272	0.945	10	0.156
SKI2	0.943	12	0.108
SKI3	0.943	12	0.107
SKI4	0.949	10	0.330
SKI5	0.945	9	0.025
SKI6	0.949	10	0.288
SKMP	0.948	10	0.290
SKOP	0.948	10	0.618
SLy	0.950	10	1.000
SLY2	0.950	10	1.028
SLY9	0.949	10	0.370
SLY230A	0.950	10	0.932

TABLE II

In Figs. 1, 2 and 3 we depict the histograms of the probabilities p(*HasNS*) (p(*HasRemnant*)) for injections of binaries that had a NS (EM counterpart) and for those that not, for the EOS BHF BB2, MS1 PP and SLy, respectively.

We show in Fig. 4 the ROC curves of the classifier for the two different probabilities we consider. All the EOS are depicted in this figure, but we highlight three cases: BHF BBB2, MS1 PP and SLy.

B. RF Results

We apply crossvalidation in the number of trees and depth of the forests for the 23 EoS, fixing the information gain criteria to `entropy`. We also save the second best option for comparison, and save both forests for each EoS in order to compare the file size. As the goal is to provide a model that can run in a low latency pipeline the amount of memory it can take is limited, even more when there will be 23 different model for the EoS generalization.

In table III we present a summary of best and second best hyperparameters found in the crossvalidation for each EoS, along the memory the model occupies and the difference in the score. As we can see, usually a forest with many trees has a second best option with far less that is lighter in memory and achieves a similar performance. The optimum maximum depth is always 15. Also the score achieved for every EoS is similar, and so we check that our accuracy is not NS model dependent.

To simplify the model and according to the results of crossvalidation, we train the final forests for all EoS

	Best				Second best			
EOS	Trees	Depth	Size(MB)	Score	Trees	Depth	Size(MB)	Δscore
APR4_BB	300	15	94.7	0.9683018	50	15	15.7	3.35e-5
BHF_BBB2	80	15	24.4	0.9685127	300	15	91.6	5.16e-5
H4	80	15	29.6	0.9618587	300	15	111.4	1.19e-4
HQC18	300	15	93.7	0.9673755	100	15	31.3	3.06e-4
KDE0V	300	15	92.0	0.9673295	80	15	24.5	2.06e-4
KDE0V1	100	15	30.9	0.96704954	80	15	24.5	3.43e-5
MPA1	80	15	27.2	0.96601225	300	15	102.1	8.19e-5
MS1_PP	300	15	113.5	0.96563534	80	15	30.2	1.15e-4
MS1B_PP	300	15	114.2	0.96555340	100	15	38.0	1.97e-4
RS	300	15	103.8	0.96447350	80	15	27.6	2.36e-4
SK255	300	15	105.8	0.96472405	100	15	35.5	3.69e-4
SK272	300	15	109.0	0.96401816	100	15	36.4	1.99e-4
SKI2	50	15	18.8	0.96242338	300	15	112.8	8.37e-5
SKI3	50	15	19.0	0.96174537	100	15	38.1	6.62e-5
SKI4	300	15	100.6	0.96598969	30	15	9.8	8.37e-5
SKI5	100	15	38.2	0.96343381	80	15	30.4	1.16e-4
SKI6	300	15	101.7	0.96586928	30	15	10.0	2.17e-4
SKMP	300	15	100.2	0.96544567	80	15	26.9	1.69e-4
SKOP	100	15	32.3	0.96610459	300	15	96.2	6.85e-5
SLy	80	15	25.3	0.96728884	300	15	95.2	8.49e-5
SLY2	100	15	31.8	0.96745868	80	15	25.4	2.38e-4
SLY9	300	15	101.6	0.96605993	100	15	34.1	1.51e-4
SLY230A	300	15	95.5	0.96714915	100	15	31.9	2.53e-4

TABLE III: Comparison of the best and second best RF models obtained during crossvalidation for all EoS. We show the file size in MB of the forest, and the difference in score between the two options.

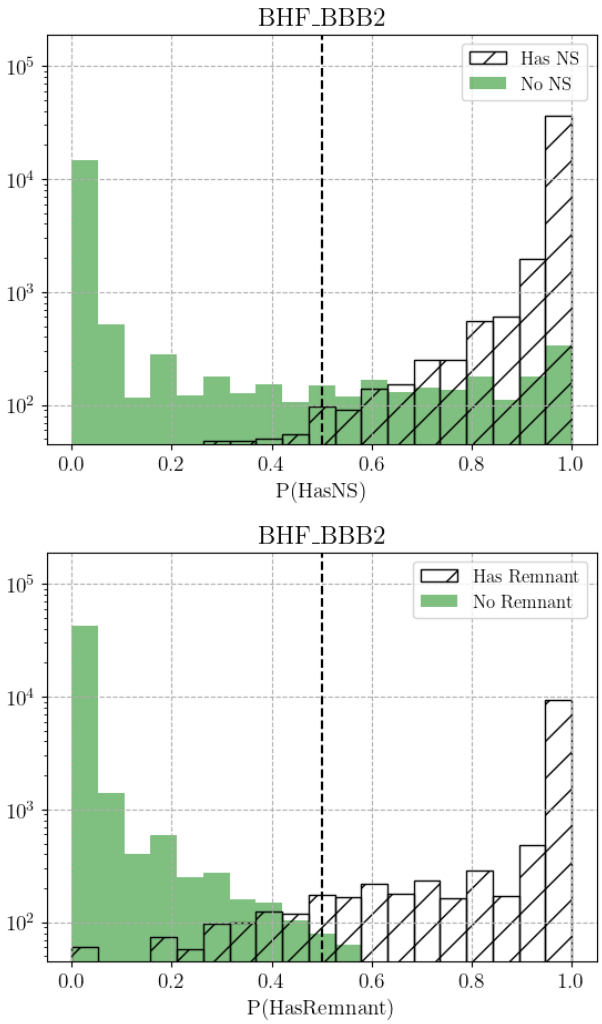


FIG. 1: Histograms BHF BBB2 KNN

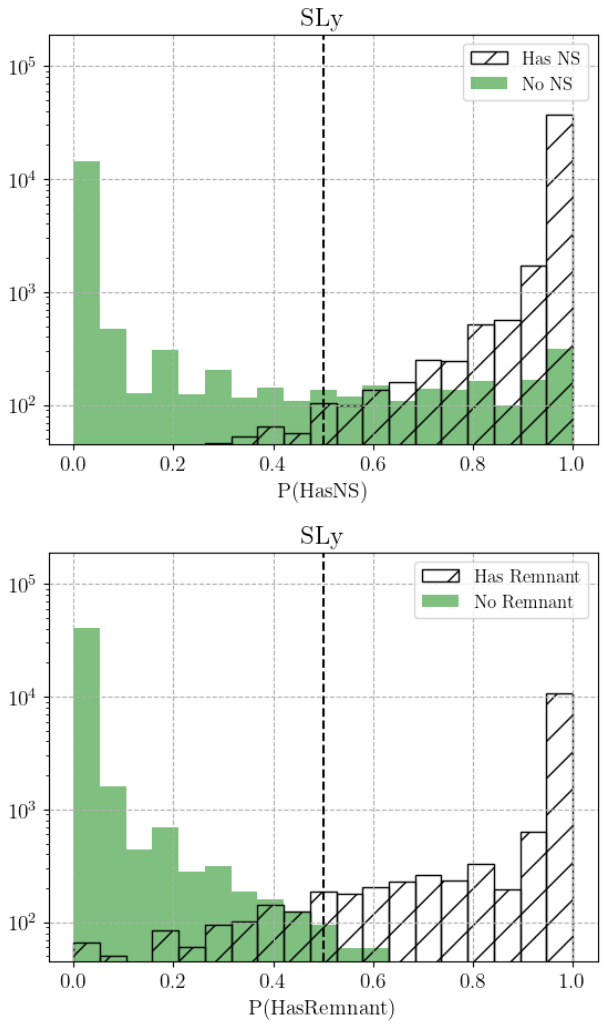


FIG. 2: Histograms SLy KNN

with 50 trees and 15 maximum depth. In figure 5 we show the ROC curves for all models to give an idea of the performance. Notice that HasREM performs better than HasNS. The ourperformance of HasREM against HasNS in RF is even more noticeable in the histograms in figures 6, 7 and 8 for the highlighted EoS, where the bars of asigned probabilities do not intersect each other and therefore there exists a threshold value for perfect classification in the testing dataset.

C. GP Results

The end of each Karoo GP training produces a multivariate exression with heighest fitness score. We repeat the training process 200 times to get 200 expressions for each label (hasNS and hasRemnant). Due to the stochastic nature of the GP algorithm, the expressions mostly tend to be unique depending upon the complexity of the classification problem. Each expression can then be used to classify the reconstructed source parameters to determine the labels hasNS and hasRemnant by substituting the values in the expression. If the result is greater or equal to zero, we classify the reconsturcted set of prameter as true and false otherwise. The performace of each of the expressions against the testing set for one EoS is shown in Fig 9. As seen in the plots, the average performace of hasRemnant expressions is better than hasNS expressions. Also the winning expressions are more dispersed in case of hasRemnant. Shall we explain the reasons: hasNS being simpler classification but contaminated by poor reconstructions by the pipeline. HasRemnant is more complex leading to more variable winning trees.

Calculation of Probability: Using testing set, we compute probabilities using bayesian method. HasNS and hasRemnant are not inherintly independent quantities since there can be no hasRemnant without hasNS. Hence, we utilize both hasNS winning expressions and hasRemnant winning expressions for quantifying probabilities $p(hasNS)$ and $p(hasRemnant)$. If XREM denotes the number of hasRemnant winning trees which classifies

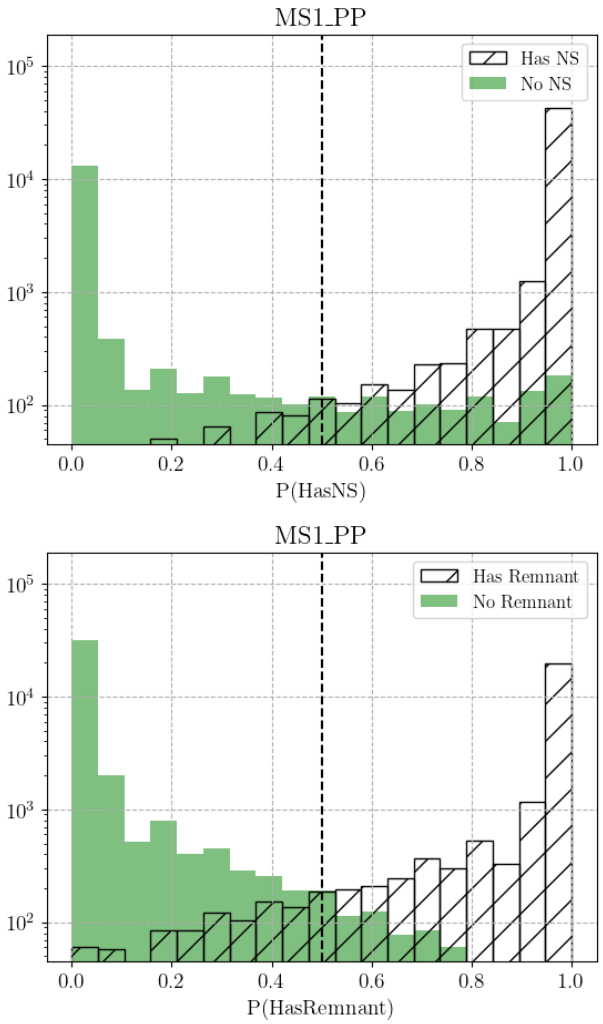


FIG. 3: Histograms MS1 PP KNN

given set of parameter as true and XNS denotes the number of hasNS winning trees which classifies as true, we compute the probability on the testing set as:

$$p(HasNS|XREM \cap XNS) = \frac{p(XREM \cap XNS|HasNS).p(HasNS)}{p(XREM \cap XNS)} \quad (1)$$

$$p(HasRemnant|XREM \cap XNS) = \frac{p(XREM \cap XNS|HasRemnant).p(HasRemnant)}{p(XREM \cap XNS)} \quad (2)$$

There as some caveats to this calculation. Given a finite testing set, we cannot fully compute the entire parameter space (201 x 201 cases) of the probability since there are some cases which are highly unlikely to occur, like $p(hasNS|XNS = 0 \cap XREM = 200)$. Also, our probability computations might also suffer due to the

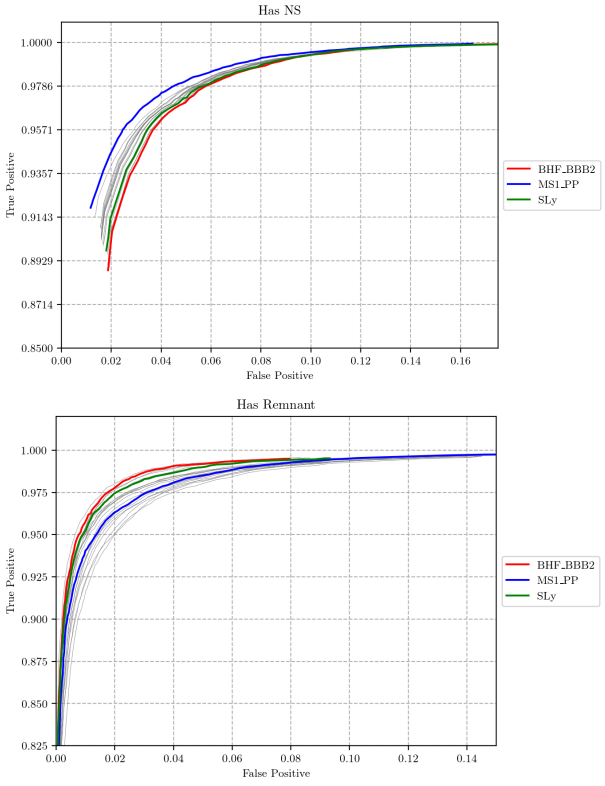


FIG. 4: ROC curves KNN

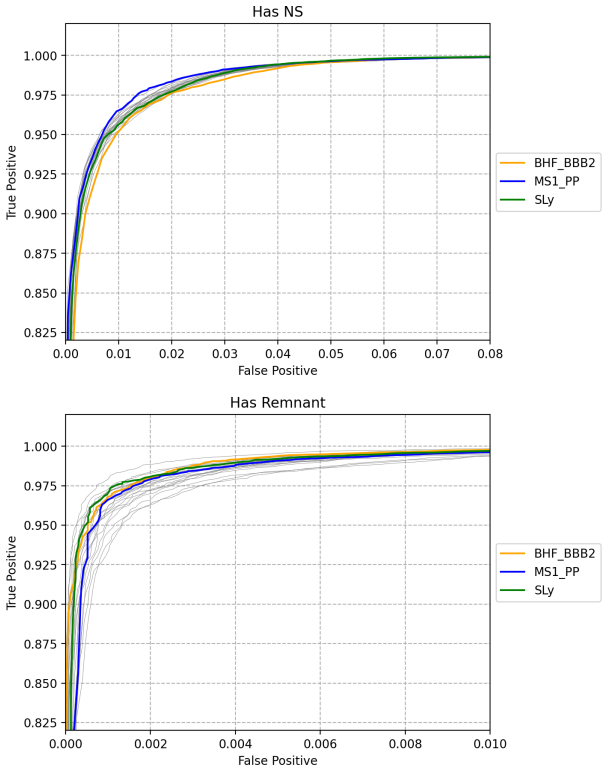


FIG. 5: ROC curves

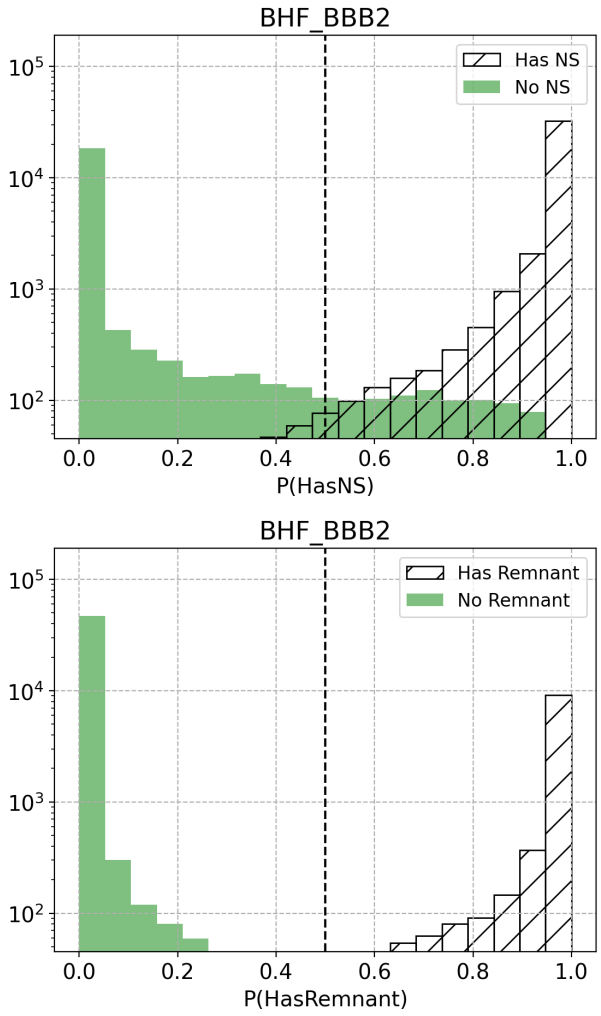


FIG. 6: Histograms BHF BBB2

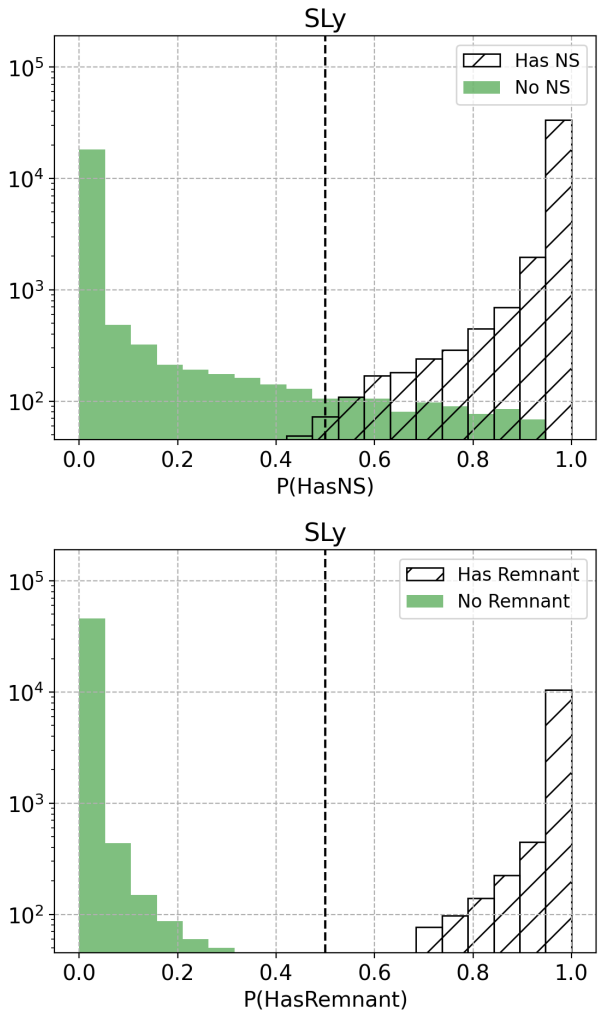


FIG. 7: Histograms SLy

finite testing set size. The resulting probability distribution is highly rugged and discontinuous or incomplete. Hence, in order to span the entire parameter space of the probability, we use gasussian process regression (GPR).

Gaussian Process Regression: We implement GPR using a python package gpytorch [**?**]. Using the Rational Quadratic(RQ) kernel of the gpytorch , we compute a smooth probability distribution over all possible combination of XREM and XNS as shown in Fig 10. The selection of the parameters for GPR was motivated by two major factors. The first being that probability had to be confined between interval [0,1] and the case $P(hasNS)$ or $P(hasRemnant)$ 1 when XNS and $XREM$ approaches maximum number of trees.The second being the condition $p(hasNS|XREM \cap XNS) \geq p(hasRemnant|XREM \cap XNS)$. After multiple trials, we found that using 200 winning expression and using length scale interval $[1X10^{-5}, 1X10^{5}]$ and alpha parameter interval $[1X10^{-5}, 10]$ for GPR, we can obtain desired probability distribution.

Using this probablity, we make the ROC curve using the testing set for all the EoS as shown in 11.

D. Algorithm comparison

To compare quantitatively the results from RF and KNN we compute the true positive and false positive rate for several threshold values, for both HasNS and HasREM, for the three selected EoS. These are tables IV, V and VI. For HasNS the two algorithms perform similarly, with almost the same TP for all threshold values and accross EoSs, although the false positive is smaller always in the RF. For HasREM we obtain that RF performs better than KNN in every case, with not only a smaller false positive rate, but a greater true positive rate.

V. CONCLUSIONS

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

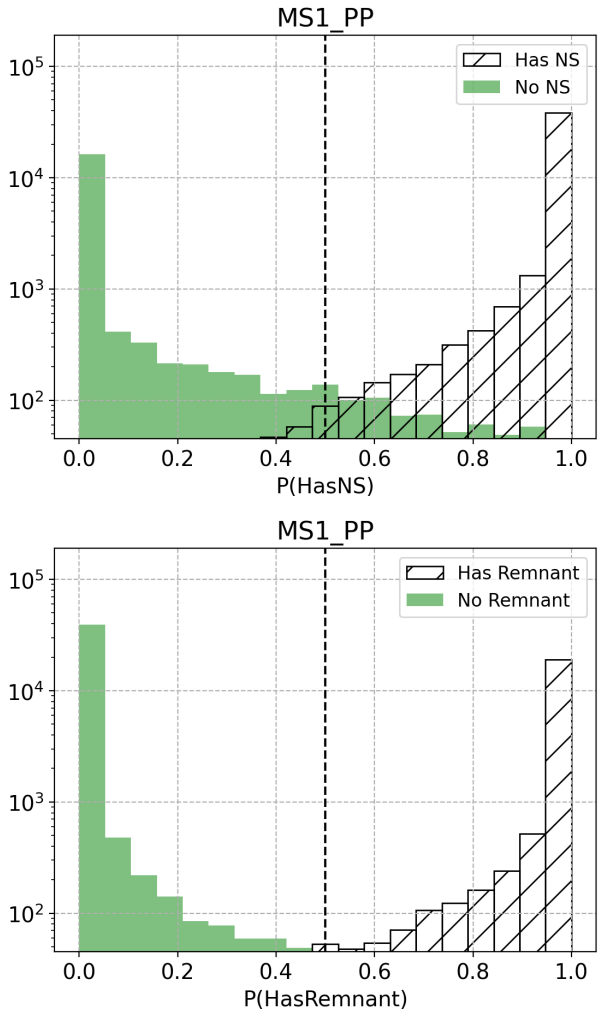


FIG. 8: Histograms MS1 PP

	Has NS				Has REM			
	RF		KNN		RF		KNN	
Threshold	TP	FP	TP	FP	TP	FP	TP	FP
0.1	0.999	0.107	0.999	0.156	0.998	0.011	0.992	0.051
0.3	0.998	0.068	0.996	0.117	0.993	0.005	0.974	0.017
0.5	0.994	0.042	0.991	0.088	0.985	0.003	0.937	0.006
0.8	0.967	0.014	0.966	0.043	0.957	0.001	0.845	0.001

TABLE IV: BHF_BB2

	Has NS				Has REM			
	RF		KNN		RF		KNN	
Threshold	TP	FP	TP	FP	TP	FP	TP	FP
0.1	1.000	0.114	0.999	0.138	0.999	0.023	0.995	0.103
0.3	0.998	0.065	0.995	0.097	0.996	0.010	0.983	0.044
0.5	0.994	0.036	0.989	0.068	0.990	0.004	0.961	0.019
0.8	0.968	0.011	0.967	0.031	0.967	0.001	0.899	0.004

TABLE V: MS1_PP

	Has NS				Has REM			
	RF		KNN		RF		KNN	
Threshold	TP	FP	TP	FP	TP	FP	TP	FP
0.1	1.000	0.107	0.999	0.155	0.998	0.013	0.992	0.059
0.3	0.999	0.064	0.996	0.112	0.993	0.005	0.974	0.020
0.5	0.994	0.038	0.990	0.084	0.986	0.003	0.940	0.007
0.8	0.965	0.012	0.965	0.040	0.958	0.001	0.848	0.001

TABLE VI: SLy

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.

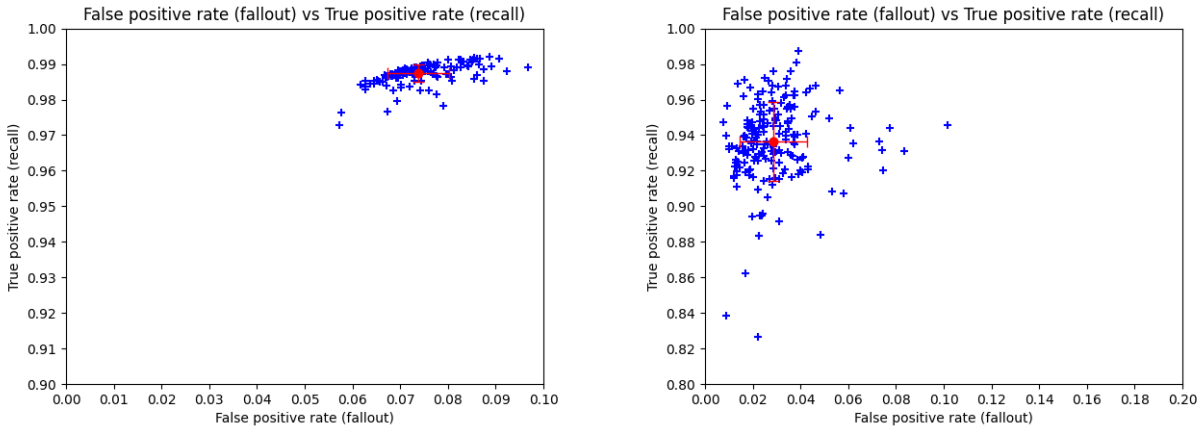
The authors are grateful for computational resources provided by the LIGO Laboratory and supported by the U.S. National Science Foundation Grants PHY-0757058 and PHY-0823459, as well as resources from the Gravitational Wave Open Science Center, a service of the LIGO Laboratory, the LIGO Scientific Collaboration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

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

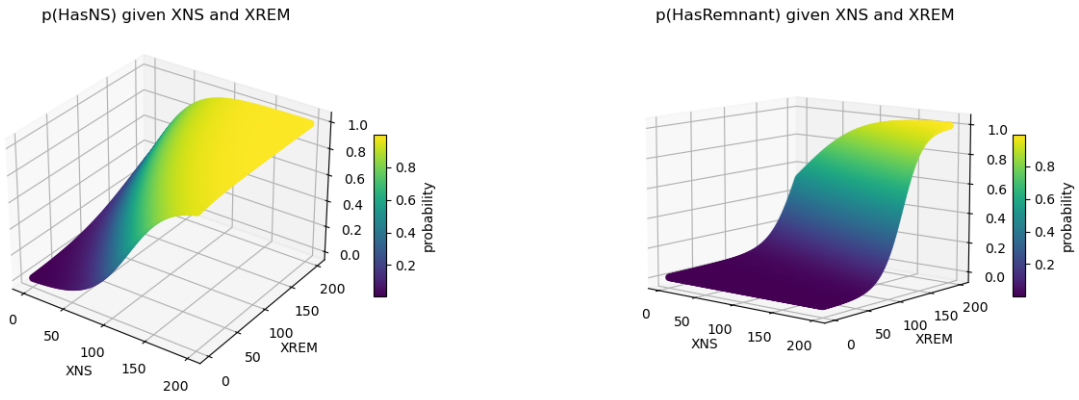
All plots were made using the python package `matplotlib` [**?**].

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



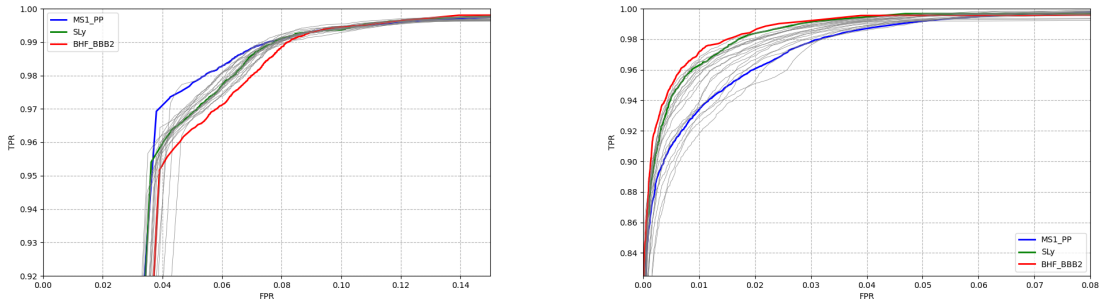
(a) (b)

FIG. 9: The plot on the left is a FPR (False Positive Rate) vs (True Positive Rate) plot for all 200 winning expressions for hasNS classification on MS1-PP EoS. The red dot is the average of TPR and FPR of the expressions and the vetricle and horizontal spread are the one sigma values for TPR and FPR respectively. The plot on the right is similar but for hasRemnant classification.



(a) (b)

FIG. 10: The plot on the left is the probability distrbution of equation 1 after GPR implementation for MS1-PP EoS. The plot on the right is a similar plot of equation 2



(a) (b)

FIG. 11: The plot on the left is the ROC curve for hasHS label evaluated on the testing set whereas the plot in the right is for hasRemnant for all 23 EoS.

	p(HasNS)		p(HasREM)	
event ID	RF	KNN	RF	KNN
GW170823	0.000	0.000	0.000	0.000
GW170817	1.000	1.000	1.000	1.000
GW170814	0.000	0.000	0.000	0.000
GW170809	0.002	0.000	0.000	0.000
GW190408	0.000	0.000	0.000	0.000
GW190412	0.000	0.000	0.000	0.000
GW190413-052954	0.000	0.000	0.000	0.000
GW190413-134308	0.000	0.000	0.000	0.000
GW190421	0.000	0.000	0.000	0.000
GW190425	1.000	1.000	0.999	1.000
GW190426	0.996	1.000	0.009	0.000
GW190503	0.000	0.000	0.000	0.000
GW190512	0.000	0.000	0.000	0.000
GW190513	0.000	0.000	0.000	0.000
GW190517	0.000	0.000	0.000	0.000
GW190519	0.000	0.000	0.000	0.000
GW190521-074359	0.000	0.000	0.000	0.000
GW190602	0.000	0.000	0.000	0.000
GW190630	0.000	0.000	0.000	0.000
GW190706	0.015	0.000	0.000	0.000
GW190707	0.000	0.000	0.000	0.000
GW190708	0.000	0.000	0.000	0.000
GW190720	0.004	0.000	0.000	0.000
GW190727	0.011	0.000	0.000	0.000
GW190728	0.000	0.000	0.000	0.000
GW190814	0.098	0.700	0.000	0.000
GW190828-063405	0.002	0.000	0.000	0.000
GW190828-065509	0.001	0.000	0.000	0.000
GW190915	0.000	0.000	0.000	0.000
GW190924	0.037	0.070	0.000	0.000
GW190930	0.000	0.000	0.000	0.000
GW191109	0.000	0.000	0.000	0.000
GW191129	0.004	0.000	0.000	0.000
GW191204-171526	0.000	0.000	0.000	0.000
GW191215	0.000	0.000	0.000	0.000
GW191216	0.000	0.000	0.000	0.000
GW191222	0.000	0.000	0.000	0.000
GW200112	0.000	0.000	0.000	0.000
GW200115	0.997	0.000	1.000	0.000
GW200129	0.000	0.000	0.000	0.000
GW200219	0.000	0.000	0.000	0.000
GW200224	0.014	0.000	0.000	0.000
GW200225	0.001	0.000	0.000	0.000
GW200302	0.000	0.000	0.000	0.000
GW200311	0.000	0.000	0.000	0.000
GW200316	0.000	0.000	0.000	0.000
GW200322	0.012	0.000	0.000	0.000

TABLE VII: PROBAB TABLE REAL DATA