https://doi.org/10.48048/tis.2022.5373

Application of Machine Learning Algorithms for Searching BSM Higgs Bosons Decaying to a Pair of Bottom Quarks

Jinna Waiwattana[1], Chayanit Asawatangtrakuldee[2,*], Pakorn Saksirimontri[1], Vichayanun Wachirapusitanand[2] and Natha Pitakkultorn[1]

[1]Mahidol Wittanyanusorn School, Nakorn Pathom 73130, Thailand
[2]Department of Physics, Faculty of Science, Chulalongkorn University, Bangkok 10330, Thailand

([*]Corresponding author's e-mail: Chayanit.Asawatangtrakuldee@cern.ch)

Received: 25 July 2022, Revised: 26 August 2022, Accepted: 30 August 2022

Abstract

Although the discovery of the 125 GeV Higgs boson confirms the Higgs mechanism of the Standard Model (SM), many theories beyond the SM have been introduced to address several phenomena yet to be explained by the SM. For instance, the 2-Higgs Doublet Models is the simplest extension of the SM Higgs sector and predicting the existence of additional Higgs bosons at different states. The aim of this study is to search for machine learning (ML) algorithms which have been widely used in High Energy Physics. This will improve the sensitivity of the search for BSM Higgs bosons produced in association with a bottom quark ($b\bar{b}H/bH$) that then decays into a pair of bottom quarks ($H \rightarrow b\bar{b}$); the predominant decay channel of the Higgs boson, though, buried by a large multi-jet background process. In this study, we train 2 different ML algorithms: Tree-based models and Neural Networks, to classify signal and background events collected by the Compact Muon Solenoid detector from proton-proton collisions at 13 TeV. The evaluation metrics are calculated to provide classification efficiencies from different models. The results show that the classification of signal and background processes can be improved using ML techniques. Neural Networks reported the highest AUC score of 0.951 which is comparable with Adaptive Boosting model, while Decision Trees (DTs) and Random Forest models slightly underperformed by 2 - 3 %. We therefore can make use of the trained models as signal vs background classifiers to perform further statistical analysis searches for BSM Higgs bosons.

Keywords: Beyond the Standard Model, Higgs bosons, Compact Muon Solenoid, Neural Networks, Decision Tree, Random Forest, Adaptive Boosting

Introduction

Recently, the discovery of a new particle at a mass of 125 GeV was announced by A Toroidal LHC Apparatus (ATLAS) and Compact Muon Solenoid (CMS) experiments [1-3] from the Large Hadron Collider (LHC) at CERN. They have put forward a wide-ranging program to understand the properties and interactions of this newly discovered boson. The current experimental measurements, including its spin, charge-parity transformation (CP) properties and coupling strengths to SM particles [4,5], have shown the compatibility within the uncertainties with the SM Higgs boson. Through Yukawa interactions [6], the SM Higgs boson has a large coupling to b-quarks, and the $H \rightarrow b\bar{b}$ channel is the predominant decay mode with a branching fraction of approximately 0.58 [7]. Despite its large branching fraction, such a decay channel was experimentally observed only a few years later after its discovery with LHC Run-2 data [8,9] due to the sensitivity limited by a sizable background, known as heavy-flavor multi-jet production, as well as the detector resolution. The sensitivity of a heavier Higgs boson is suppressed due to the small Yukawa coupling for large values of Higgs mass and also challenged by the same background process.

Many beyond the Standard Model (BSM) theories have been introduced to address several phenomena left unexplained by the SM. For example, hierarchy problems concerning the particle masses, the nature of dark matter and dark energy, etc. Precise measurements of the couplings of the observed Higgs boson to SM particles also provide indirect constraints on additional contributions to the Higgs boson width from BSM decays ($H \rightarrow BSM$). Based on the results presented in [7], the 95 % confidence level (CL) on the indirect upper limit of $H \rightarrow BSM$ is measured to be 0.34. Furthermore, the CMS

experiment has recently published results on the direct search for invisible decays of the Higgs boson via vector boson fusion at 13 TeV [10], providing the best observed (expected) upper limit on the invisible branching fraction of the Higgs boson of 0.18 (0.10) at the 95 % CL, assuming the SM production cross section. Therefore, the existence of the Higgs boson decaying to BSM particles remains an interesting possibility, which strongly motivates the direct BSM Higgs boson searches.

In the Higgs sector of the Minimal Supersymmetric Standard Model (MSSM) [11], each SM field is promoted to a superfield which connects the SM particles to their respective super-partners. Two doublets of complex scalar fields with opposite hypercharge are also introduced to protect the Higgs boson mass from being close to the very high scale in hierarchy and naturalness problems. In this theory, the 2 scalar doublets are required to enable Yukawa interactions with up-type and down-type fermions. After electroweak symmetry breaking, the 2 doublet fields lead to 5 physical states of Higgs bosons: Two CP-even h, H bosons, a pseudoscalar A boson and 2 charged Higgs bosons, $H^{\pm}$. The lightest CP-even Higgs boson, h, has properties similar to the SM Higgs boson. Alternatively, the 2-Higgs Doublet Models (2HDM) [12] offer a more generic extension of the Higgs sector in the SM by introducing only an additional scalar doublet field without assuming supersymmetry. This means the fermionic sector of the SM is not changed, while the Higgs sector is structured the same way as in the MSSM. There are 4 types of 2HDM that conserves charge-conjugation parity (CP) symmetry and avoids flavor-changing neutral currents (FCNC) at the tree level. The type-II (similar to MSSM) and flipped of 2HDM are particularly interesting in connection to $b\bar{b}H/bH$ production and $H \to b\bar{b}$ final state because the couplings to b-quark are enhanced for this study.

Based on data collected by the CMS at the LHC, searches for neutral Higgs bosons in the context of MSSM and 2HDM have been studied since LHC Run-1 during 2010 - 2012 at a center-of-mass energy of 7 and 8 TeV [13,14] and continued in Run 2 (2015 - 2018) with increased energy of 13 TeV. The latest published results using 2016 data collected by the CMS and ATLAS detectors [15,16], corresponding to an integrated luminosity of 35.7 fb^{-1} and 27.8 fb^{-1} respectively, were performed with the traditional cut-based method. No evidence of signal with additional neutral Higgs bosons was observed. One would improve the sensitivity of this analysis by increasing the amount of data, while developing an analysis method using more sophisticated tools, namely ML techniques, as introduced in this study.

ML has had a critical role in the study of high energy physics for almost a decade and has been explored in many aspects, including data detection, data quality monitoring, physics object identification such as jet-flavor tagging and event classification in physics analyses. The most popular example of physics analysis can be seen from the discovery of the SM Higgs boson by the CMS experiment at the LHC in 2012 with multivariate analysis using Boosted Decision Trees [2]. Another example is the recently published results on the search for SM Higgs boson decaying into a pair of charm quarks, where charm-flavored jets can be distinguished from jets originating from other flavor quarks using Graph Neural Networks [17]. Baldi's study [18], finding exotic particles by solving difficult signal versus background classification problems, also demonstrates that Deep Learning can improve the power of high energy physics searches.

This paper is aimed at studying and developing ML classifiers which could help discriminate signal over background processes and improve the sensitivity of the search. The following sections will be discussed: Datasets and input features, Machine learning models, Evaluation methods, Results and discussion, and Conclusions. In the next section, signal and background datasets generated for this study and input features will be described, followed by the ML models and their evaluation methods on how to compare and present the performance among different algorithms. Results and discussion on this study will be provided, and a conclusion will be given in the last section.

Materials and methods

Datasets and input features

Signal and background processes

To search for additional Higgs bosons predicted by the MSSM and 2HDM, Monte Carlo simulations are used to generate hard scattering processes from proton-proton collisions assuming 13 TeV at the LHC by MadGraph_aMC@NLO generator [19] and passing parton showering and hadronization using Pythia generator [20]. The particles from the collisions are then detected by the CMS detector using Delphes [21], a framework for fast detector response simulation. In this study, we are interested in searching for a neutral Higgs boson with a mass of 300 GeV produced from proton-proton collisions in association with bottom quark(s) ($b\bar{b}H/bH$) that then decays to bottom anti-bottom quark pairs ($H \to$

$b\bar{b}$). Note that the choice of the Higgs boson mass is adequate since we can repeat the whole training procedure using different masses for the search.

The characteristic signature of the signal process is the final state of 2 highly energetic b-tagged jets, initiated from decay products of the Higgs boson, together with at least 1 associated b-tagged jet as shown in **Figure 1**. Background contributions arise from the SM processes, mainly the heavy-flavor multi-jet production, providing a similar triple b-tagged jets signature as the signal process while at least 1 light-flavored jet is misidentified as a b-quark initiated jet as displayed in **Figure 2**. Note that jets are sprays of particles produced by the hadronization of a quark or gluon and reconstructed as a single composite object by clustering and recombination algorithms. A distance parameter of 0.4 and anti-k_T algorithm is used for jet clustering technique in this study. The b-tagged jets refer to jets that arise from the decay of b-hadrons.

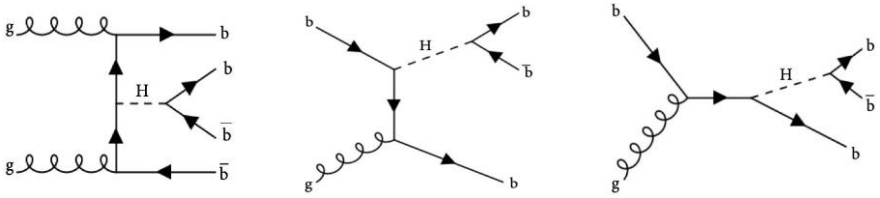


Figure 1 Feynman diagrams of the signal process ($b\bar{b}H/bH$ production and $H \rightarrow b\bar{b}$ decay channel).

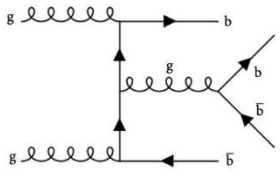


Figure 2 Feynman diagram of background process (heavy-flavor multi-jet production).

Signal and background processes are generated separately in a total of 200,000 and 1,500,000 events, respectively. After parton showering (the cascades of radiation produced from quantum chromodynamics processes) on heavy-flavor multi-jet events, the number of generated background decreases to 279,728 corresponding to the jet matching efficiency of around 0.20 and makes the final statistics on both signal and background datasets to be comparable for the ML training. In addition, signal and background events are required to have at least 3 jets in each event before using them in the next step. No event is discarded.

Low-level and high-level features

The final datasets include 35 features as independent variables. We categorize all input features into 2 categories: Low-level and high-level features. Low-level features represent basic measurements that can be provided by the detector, while high-level features are further constructed from low-level features by assuming a particular pair of jets decayed from the Higgs boson. In this study, a pair of jets is taken from all possible combinations of 3 leading jets ordered by transverse momentum, and then the 4-momentum vector is calculated for each di-jet pair written as j12, j13 and j23. **Tables 1** and **2** show the lists of low-level and high-level features in simulated datasets.

To perform a supervised ML study, labels 0 and 1 are added to all simulated events corresponding to background and signal processes, respectively. These labels are responsible for classification tasks in the ML training as target variables (output). Before moving towards the training, we draw the kinematic distributions of all the inputs to visualize the separation power of each input feature to be used for signal and background classification. **Figure 3** shows the comparison of low-level inputs and **Figure 4** shows the comparison of high-level inputs. These histograms are normalized to have the same total count of 100 events in both processes for shape comparison.

Furthermore, we also check linear correlation among input features by calculating the Pearson's correlation coefficients of each pair of input features. If any high-level feature shows strong correlation

with the low-level features, they will not be considered in the training. The reason is that any 2 strongly correlated input features do not provide additional useful information to the ML training, nonetheless, they unnecessarily consume computing resources.

Table 1 Definition of low-level features in the datasets. Note that each variable is counted per event basis.

Feature name	Definition
njets	Number of jets
nbjets	Number of b-tagged jets
ptj1, ptj2, ptj3	Transverse momentum of the 1st, 2nd and 3rd leading jets in order. Transverse momentum is defined as momentum along the plane orthogonal to the axis of collision (beam axis).
etaj1, etaj2, etaj3	Pseudorapidity of the 1st, 2nd and 3rd leading jets is defined as $\eta = -\ln(\tan(\theta/2)), \eta \in (-\infty, \infty)$ where θ is the polar angle between the particle 3-momentum $\vec{p} = (p_x, p_y, p_z)$ and the positive direction of the beam axis.
phij1, phij2, phij3	Azimuthal angle (ϕ) of the 1st, 2nd and 3rd leading jets. This angle is the transverse momentum's angle measured on the plane orthogonal to the beam axis.
mj1, mj2, mj3	The invariant mass of the 1st, 2nd and 3rd leading jets

Table 2 Definition of high-level features in the dataset. Note that each variable is counted per event basis.

Feature name	Definition
pt12, pt13, pt23	Transverse momentum of each di-jet pair assuming each pair could be the decay product of the Higgs boson and calculated from 4-momentum
m12, m13, m23	The invariant mass of each di-jet pair assuming each pair could be the decay products of the Higgs boson. The invariant mass of the Higgs boson can be calculated as: $m_{ij}^2 = (E_i + E_j)^2 - \left\|(\vec{p}_i + \vec{p}_j)\right\|^2$ where i, j = 1,2,3
eta12, eta13, eta23	Pseudorapidity of each di-jet pair assuming each pair could be the decay product of the Higgs boson and calculated from 4-momentum
phi12, phi13, phi23	The polar angle of each di-jet pair assuming each pair could be the decay product of the Higgs boson and calculated from 4-momentum
dEta12, dEta13, dEta23	Pseudorapidity difference ($d\eta$) between 2 particles
dPhi12, dPhi13, dPhi23	Polar angle difference ($d\phi$) between 2 particles
dR12, dR13, dR23	A measurement of the angular separation between 2 particles defined as: $dR = \sqrt{(d\eta)^2 + (d\phi)^2}$

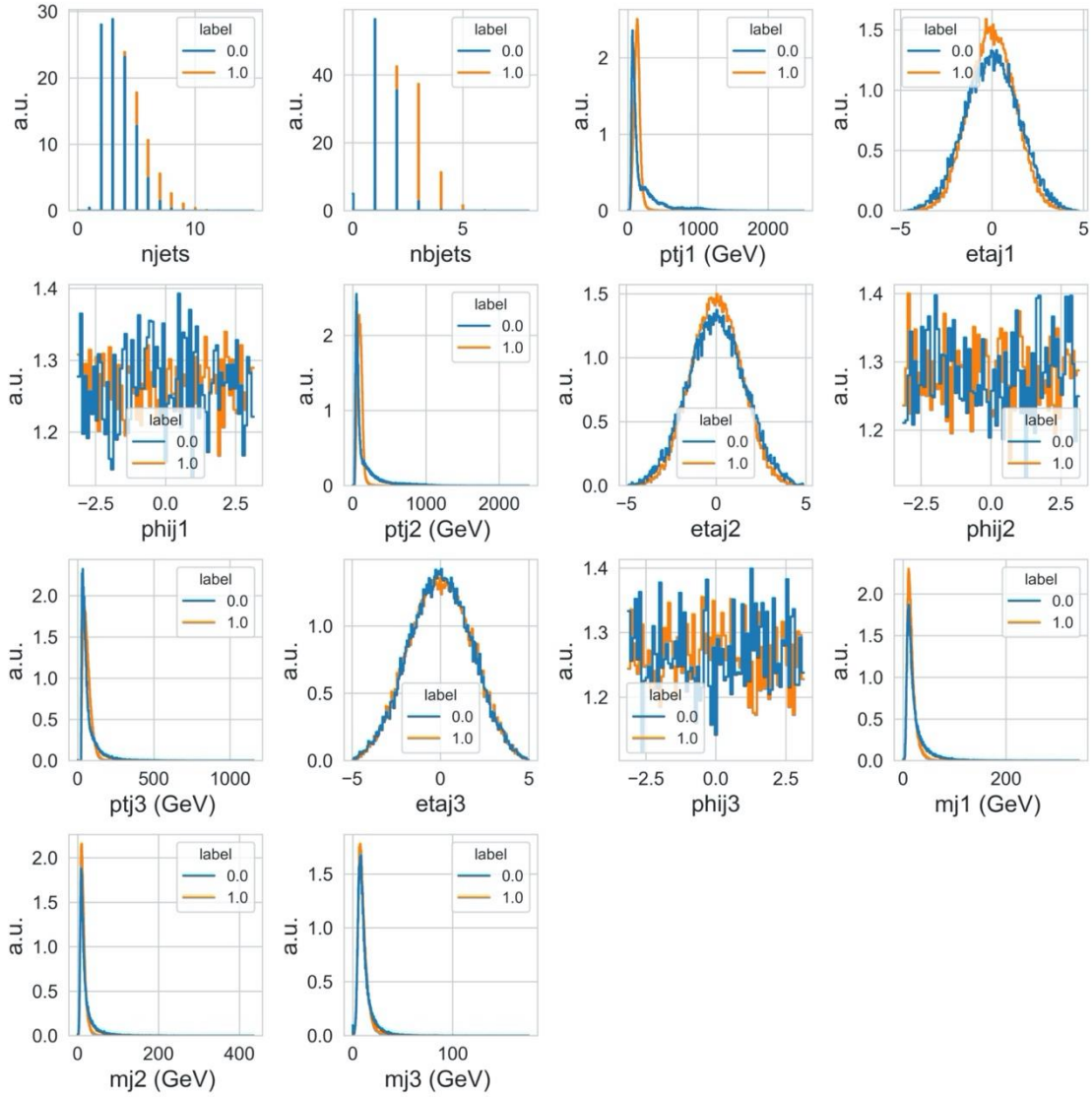


Figure 3 Low-level features: Distributions of low-level input features comparing signal (orange) and background (blue) processes.

Correlation matrix and standardization

Pearson's correlation coefficient is a measure of linear correlation between 2 variables and is generally defined as the covariance of the 2 variables divided by the product of their standard deviations, as referred to [22], resulting in the value range between –1 to 1. In this study, the Pearson's correlation coefficient of any 2 input features can be written as r_{xy}, and given paired data $\{(x_1, y_1), \ldots, (x_n, y_n)\}$ consisting of n pairs (n events), r_{xy} can be evaluated using the following expression:

$$r_{xy} = \frac{\sum_{i=1}^{n}(x_i-\bar{x})(y_i-\bar{y})}{\sqrt{\sum_{i=1}^{n}(x_i-\bar{x})^2 \sum_{i=1}^{n}(y_i-\bar{y})^2}} \quad (1)$$

where $\bar{x}$ and $\bar{y}$ represent the mean value of input feature x and y, respectively. If the value of r_{xy} is close to ±1, it implies that these 2 input features are nearly well explained by a linear equation, therefore they are strongly correlated. In contrast, 2 input features are relatively uncorrelated when the value of r_{xy} is close to 0. Any 2 input features are considered to be highly correlated if the absolute value of Pearson's correlation coefficient r_{xy} is greater than 0.9.

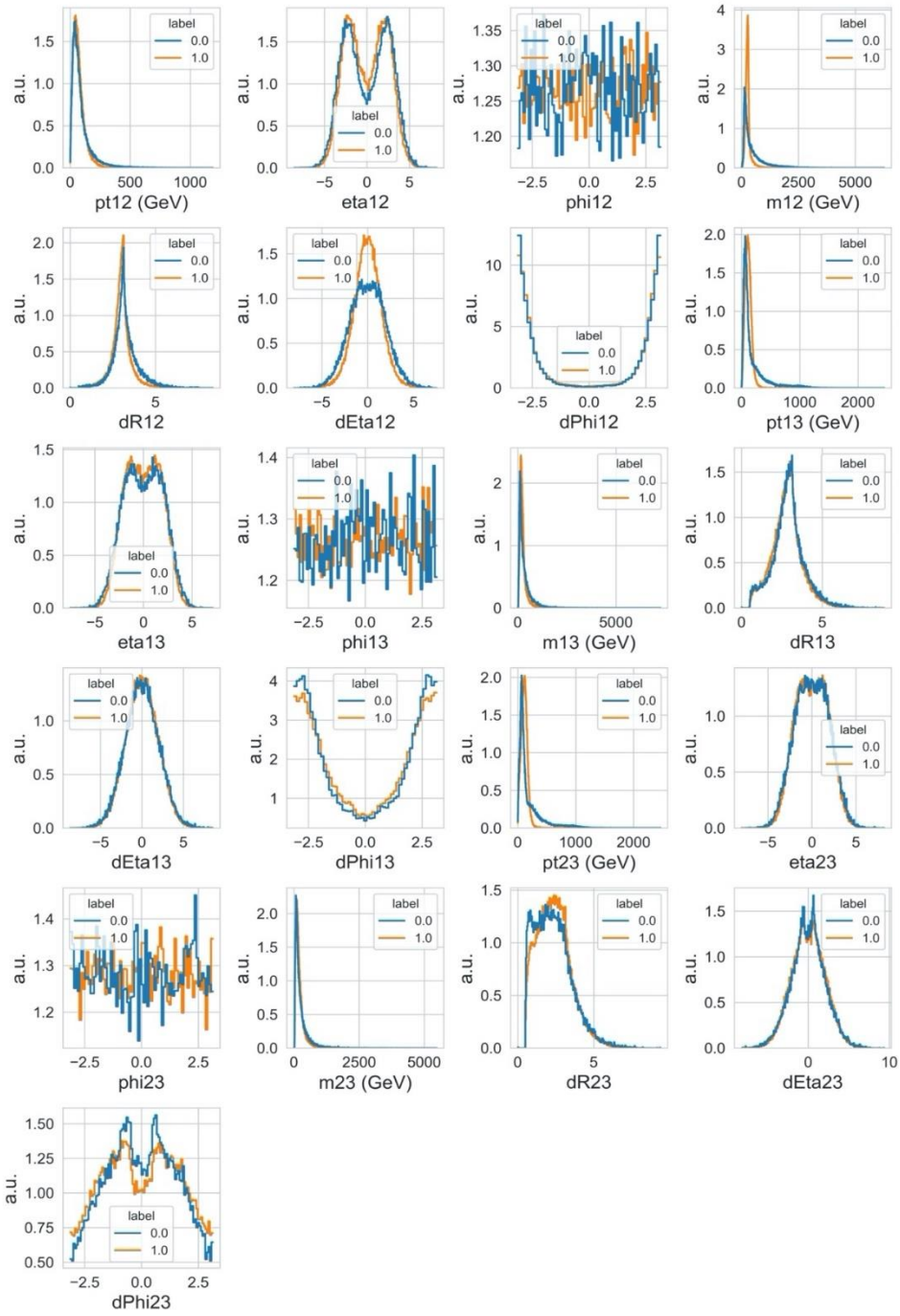


Figure 4 High-level features: Distribution of high-level input features comparing signal (orange) and background (blue) processes.

Finally, we separate both signal and background datasets into training and testing datasets corresponding to 70 and 30 %, respectively. The number of events in the training dataset is also balanced using an undersampling method to overcome the risk of majority-biased prediction in our ML models. The final number of events in the training datasets for both signal and background classes is consistent at 140,000 events.

In the case of Neural Networks, each input feature is further standardized by rescaling the mean value to 0 and standard deviation to 1. The standardization is necessary because Neural Networks use an optimization method that would work inefficiently when input features have wide ranges of value.

Machine learning models

In this study, Tree-based models, including DTs, Random Forest and Adaptive Boosting and Neural Networks, the most famous supervised learning models, are considered to determine which model provides the best discriminatory performance for further statistical analysis. This section explains the general concept and techniques behind these 4 models. Concerning the tools, we implement each model in our study using the Scikit-learn and Keras libraries in python, as referred to [23,24].

Tree-based models

Tree-based model is a class of learning algorithms that generates predictions by utilizing a series of if-then rules. It can perform both classification and regression tasks by creating a tree-like structure and determining the target variable using the desired features. Particularly, DTs are the main component for all Tree-based ML models. The tree structure consists of decision nodes for making decisions and leaf nodes for displaying the output. In this study, to build the Tree-based models for ML, we import the Scikit-learn library to generate the model and to support other ML-related libraries such as Pandas, NumPy and Seaborn. Then, we use classification tree algorithms for all the Tree-based models.

Decision Trees (DTs)

DTs are a non-parametric supervised learning method. It can create a model to predict the value of the target variable by learning simple decision rules inferred from input features of the training datasets. DTs are composed of 2 steps: Induction and pruning processes. Induction is the process where the trees are built and determined which features to split on by selecting the feature that will produce the most homogenous resulting datasets. Since our study is a classification criteria task, we select the Gini index function as a cost function to evaluate splitting in feature selection, which can be calculated using the following equation [23]:

$$\text{Gini index} = 1 - \sum_{i=1}^{n} p_i^2 \quad (2)$$

where p_i stands for a proportion of the sample that belongs to class n for a particular node.

The Gini index function creates binary splits, thus producing the hierarchical tree structure. When the trees are prone to majority overfitting, the pruning process is followed to get the optimal decision tree by deleting unnecessary nodes from the trees.

Random Forest

Random Forest, another concept of the Tree-based models, is a bagging ensemble method by creating multiple DTs and randomly sampling with bootstrapping technique, a method of estimating quantities by averaging estimates from multiple samples. When bootstrapping is performed, there will be some data events not being chosen in the sampling process called out-of-bag data which could be further used in the validation of the model. Each node is only split on the feature's random selection. Finally, the algorithm will select the majority vote from the trees to gain the final predictions.

Adaptive Boosting

Adaptive Boosting, on the other hand, implements a boosting method. This model builds continuous small trees with each tree aimed at correcting the net error from the preceding trees. The subsequent tree will focus on correctly classifying errors from the previous tree and so on. The final prediction is calculated as a weighted sum of all individual predictions.

Neural Networks

Multi-Layer Perceptron is 1 type of artificial neural network that requires learning to be supervised. It is made up of an input layer, multiple hidden layers and an output layer in which each layer has a specific number of nodes. Each node in 1 layer fully connects to another node in the next layer and is initialized with weight and bias. Neural Networks can learn the non-linear relationship between input features and the target variable using activation functions which are functions that transform the output from a node given input. The key algorithm that is responsible for weight and bias updating after computing the difference between predicted results and the true labels is the optimizer, a mathematical method used to minimize the losses and provide accurate results. The central concept behind optimization algorithms is called backpropagation.

Evaluation methods

Evaluation metrics

To quantify the predictive performance of ML models on seen and unseen data, the following metrics are used [23]:

$$\text{Accuracy} = \frac{TP + TN}{TP + FP + TN + FN} \quad (3)$$

$$\text{True positive rate} = \frac{TP}{TP + FN} \quad (4)$$

$$\text{False positive rate} = \frac{FP}{TN + FP} \quad (5)$$

where, in our binary classification case, true positive (TP) is the number of events correctly predicted to be signal, true negative (TN) is the number of events correctly predicted to be background, false positive (FP) is the number of events incorrectly predicted to be signal, and false negative (FN) is the number of events incorrectly predicted to be a background.

Accuracy metric measures a fraction of correct predictions, while TP rate measures a fraction of events correctly predicted as signals out of total actual signal events, and FP rate measures a fraction of events wrongly predicted as signals out of total actual background events. Therefore, Receiver Operating Characteristic (ROC) curves can be created by plotting TP rate versus FP rate at various thresholds (or cutoff points) which helps to illustrate the performance of a binary classifier system as its discrimination threshold is varied. AUC score is the area under the ROC curve; AUC closer to 1 indicates good predictive performance, whereas the AUC equal to 0.5 suggests that a model has similar predictive performance as random guessing.

Feature selections

To improve ML classifier scores and boost their predictive performance on very high-dimensional datasets, once the models are trained on the training datasets, we quantitatively calculate the importance of each input feature for the model to classify events into signal or background classes. Feature selection based on model performance is a process where we select input features that contribute the most to the target variable.

Feature Importance based on Mean Decrease in Impurity (MDI) is an embedded method of feature selection specifically for the Tree-based models. We can obtain the importance of each feature right after training the model on the datasets. The Scikit-learn implements an optimized version of the classification and regression trees (CART) algorithm which creates a binary tree (each node has exactly 2 outgoing child nodes). The algorithm finds the best feature and splits the value by using the impurity criterion. Gini impurity is used for classification trees. Therefore, the feature importance can be computed along the way as the decrease in Gini impurity is weighted by the proportion of samples reaching that node. The higher value the more important such input features are.

Considering the simplest case as DTs, when we assume only 2 child nodes, the importance of a particular node can be calculated by [23]:

$$ni_j = (N_j/N)\, I_j - (N_{\text{left}(j)}/N)\, I_{\text{left}(j)} - (N_{\text{right}(j)}/N)\, I_{\text{right}(j)} \quad (6)$$

where ni_j represents the importance of node j, N_j/N is the proportion of samples reaching node j, I_j is the Gini impurity of node j, and left(j) or right(j) subscription means the child node from the left or right split on node j. Then, the importance of feature i (fi_i) is evaluated as:

$$fi_i = (\textstyle\sum_j ni_j \,/ \sum_k ni_k) \quad (7)$$

where $\sum_j ni_j$ represents the sum of all nodes splitting on input feature i, while $\sum_k ni_k$ is the sum of all nodes. In addition, we also normalize this value by dividing by the sum of all feature importance values ($\sum fi_i$) to get the value range from 0 to 1.

Recursive Feature Elimination (RFE) is one of the wrapper methods of feature selection implemented in Scikit-learn. In principle, the feature importance values of the remaining features are subject to change whenever 1 feature is removed from the training dataset. As a result, we have to repeatedly train the model until it obtains the correct feature importance values of all features. Therefore,

the RFE algorithm has a role to select features by recursively removing the least important features until the specified number of final features reach. Given any ML model that possesses calculation of feature importance as its attribute, such as MDI of Tree-based models, the model is trained on the initial training dataset to acquire feature importance. Subsequently, the least important features are removed from the training dataset. These steps are repeated until the specified number of final features is obtained.

Permutation Feature Importance, on the other hand, measures the reduction of the model score when the value of a particular input feature is randomly rearranged. Given the fitted predictive model m and tabular data D. The algorithm will first compute the baseline score s of the model m on data D, in this case, we use AUC as the model score. For each input feature j (column of D), the algorithm randomly shuffles column j of data D to generate a corrupted version of the data named $D_{k,j}$ for each repetition k in $1,\ldots,K$ and computes the score $s_{k,j}$ of model m on corrupted data $D_{k,j}$. Finally, importance i_j for input feature f_j can be calculated using the following equation [23]:

$$i_j = s - \frac{1}{K}\sum_{k=1}^{K} s_{k,j} \tag{8}$$

Bear in mind that the permutation importance of each input feature does not review the intrinsic predictive value of that input feature in general, however, it only represents how important each input feature is for 1 particular model. Therefore, the input feature that obtains low importance for a bad model (low evaluation scores) could turn out to be very important for another good model. Besides, a pair of strongly correlated input features could cause an underestimation of importance since the model will still have access to the permuted feature through its correlated feature resulting in a lesser reduction in AUC than it should be. In this study, we implement the permutation feature importance on the Neural Network model.

Results and discussion

Results on feature visualization and correlation matrix

There are 35 input features with 140,000 events per process in the training datasets after passing the undersampling method. As mentioned, training the ML models with too many input features could cause the overfitting and performance reduction of the models. Also, large datasets require long computational time and resources. Initially, the visualization of **Figures 3** and **4** suggested that the low-level features including njets, nbjets, pt1, pt2 and pt3, together with several high-level features such as mj1, mj2, mj3, m12, pt12, pt13, pt23 and dEta12 could be good candidates to provide a strong separation power for classification. We took them in the 1st list of input features for the ML training.

For the next step, we applied the method of input feature filtering by calculating the Pearson's correlation coefficient across 35 input features using Eq. (1). The values of Pearson's correlation coefficients of all pairs of input features are illustrated as a correlation matrix in **Figure 5**. In case the absolute value of Pearson's correlation coefficient was greater than 0.9, the 2 input features will have been considered to be strongly correlated; therefore, the high-level features were excluded from the list. From the correlation matrix, 2 pairs of input features: pt13 with ptj1 and pt23 with ptj2, reported the value of Pearson's correlation coefficient of 0.96 and 0.95, respectively. Therefore, we excluded pt13 and pt23 from the list of input features of the training datasets to reduce dimensional dependent variables in the ML training.

Results on hyperparameter optimization of each model

When creating the ML models, each model architecture is designed by a default set of parameters, and we usually do not know what the optimal model architecture should be for a given model at the beginning. Fortunately, we can search for the range of possibilities using the machine itself to explore and select the optimal model automatically. The set of model parameters defining the model architecture is referred to as hyperparameters and the process of searching for the ideal ML model architecture is called hyperparameter tuning. Note that hyperparameters are not the model parameters and cannot be trained directly from the datasets.

In this study, we performed hyperparameter tuning on each Tree-based model and Neural Network to improve the model architectures and their performances concerning our previous work shown in [25]. The results of different Tree-based models on hyperparameter tuning are presented in **Figure 6**.

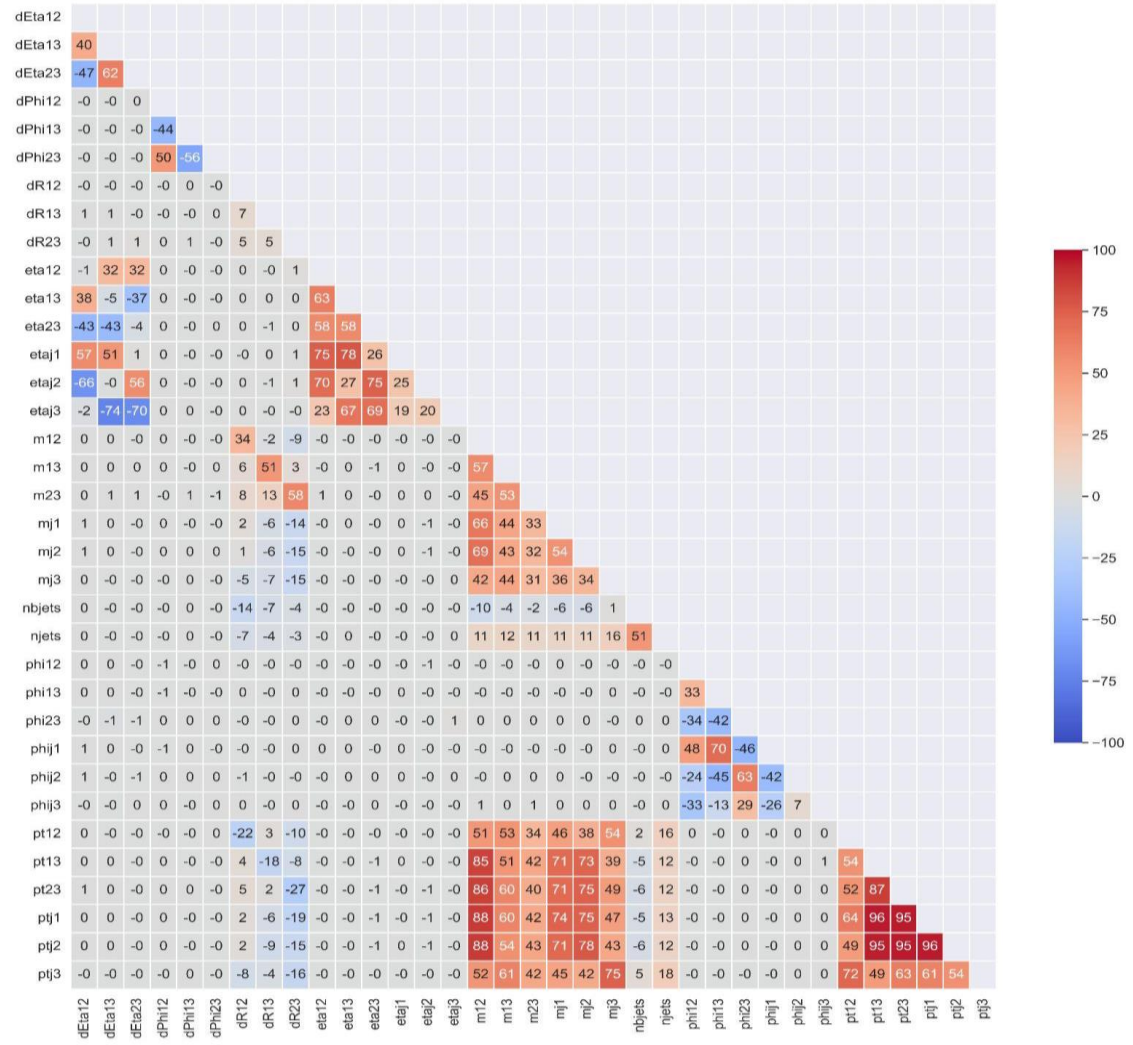
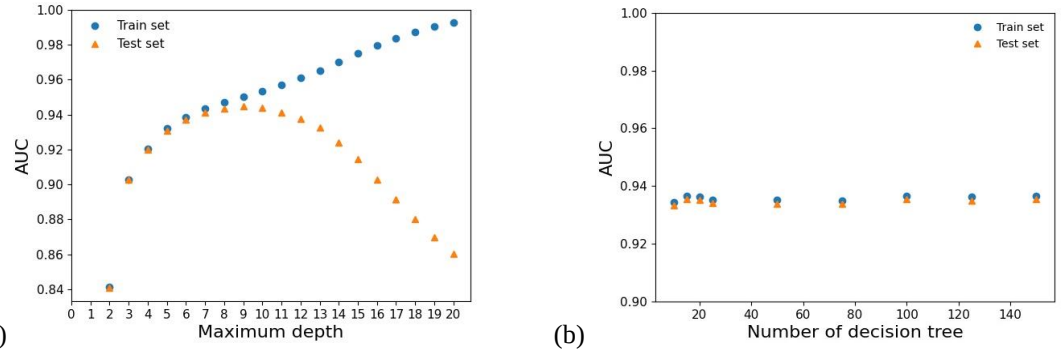
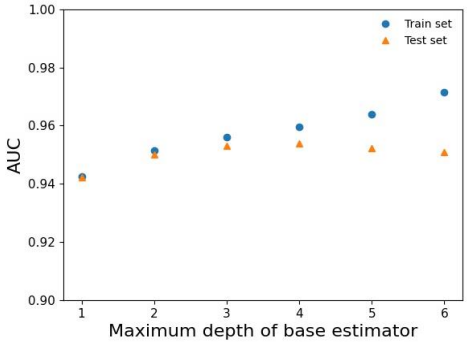


Figure 5 Correlation matrix: The absolute values of Pearson's correlation coefficient among 35 input features which are ordered alphabetically and scaled to 100 % for legibility.



(a) (b)



(c)

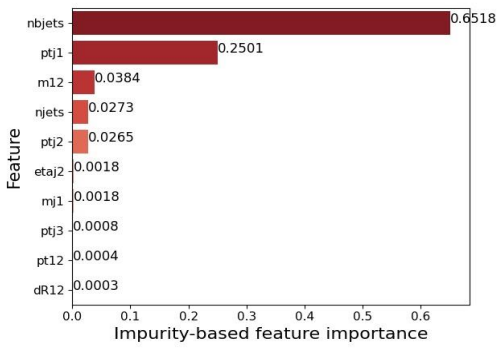
Figure 6 Hyperparameter tuning: A comparison of AUC scores between the training (blue circle) and testing (orange triangle) datasets at different hyperparameter setups for each model (a) DTs (b) Random Forest and (c) Adaptive Boosting.

The DTs model was found to have optimal architecture at the maximum depth of 6 where the AUC scores reached the plateau and agreed well between the training and testing datasets. The Random Forest model showed similar performance in AUC scores across the number of DTs from 10 to 150, therefore, we simply selected 50 DTs as hyperparameters of this model. Concerning the Adaptive Boosting model, the depth of base estimator of 3 was selected to keep high performance and good agreement between the training and testing datasets at the same time.

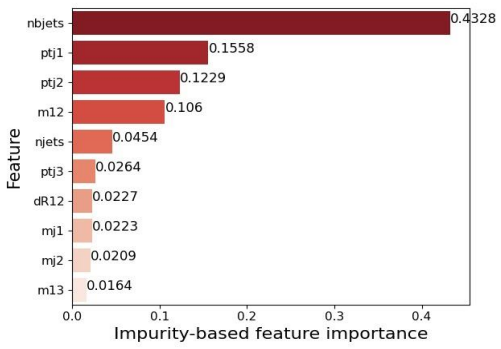
In the case of the Neural Network model, the hyperparameters were also adjusted to provide the optimal model architecture. Finally, we selected a 4-layer Neural Network model including an input layer with 33 input nodes, an output layer with a single node and a sigmoid activation function, together with double hidden layers having 100 and 50 hidden nodes, respectively. The rectifier linear unit (RELU) activation function, defined as the positive part of $f(x) = \max(0, x)$, was selected in both hidden layers. The chosen optimizer was stochastic gradient descent [23]. The loss function to minimize was binary cross entropy [23]. Weights of all layers were initialized from a uniform distribution within [-limit, limit], where limit $= (0.6/(fan_{\text{in}} + fan_{\text{out}}))^{1/2}$ and fan_{in} is the number of input nodes in the weighted tensor and fan_{out} is the number of output nodes. Biases of all layers were initialized to 0.

Results on feature importance

Input features of the ML models are supposed to take only important features. In this study, we evaluated the importance of 33 input features using MDI and RFE for the Tree-based models, and Permutation Feature Importance for Neural Network. The results on the feature importance of each model are illustrated in **Figure 7**. The plots of Tree-based models (a), (b) and (c) show the top 10 input features with the highest importance values given in fractional numbers, while Neural Network provides the top 21. One could see that the model with higher complexity, namely the Neural Network, resulted in more input features with higher importance value than the Tree-based models. This is because the Neural Network makes use of all input features with different weights adjusted, while the Tree-based models only select a subset of input features in the given trees per training.



(a)



(b)

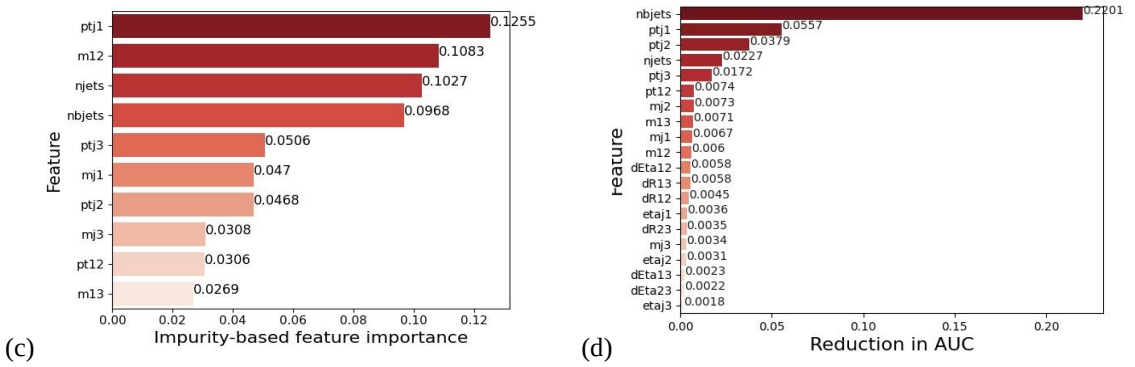


Figure 7 Feature importance: The rank of input features ordered by the values of importance and given in fractional numbers for (a) DTs, (b) Random Forest, (c) Adaptive Boosting and (d) Neural Network models.

Meanwhile, we also performed a test where the ML models were trained with different numbers of input features to determine the elbow point that gives the best performance but the least number of input features. The configuration giving a constant performance with the least number of input features was chosen for each model, provided the performance on the training and testing datasets are identical indicating that there is no overfitting, as shown in **Figure 8**.

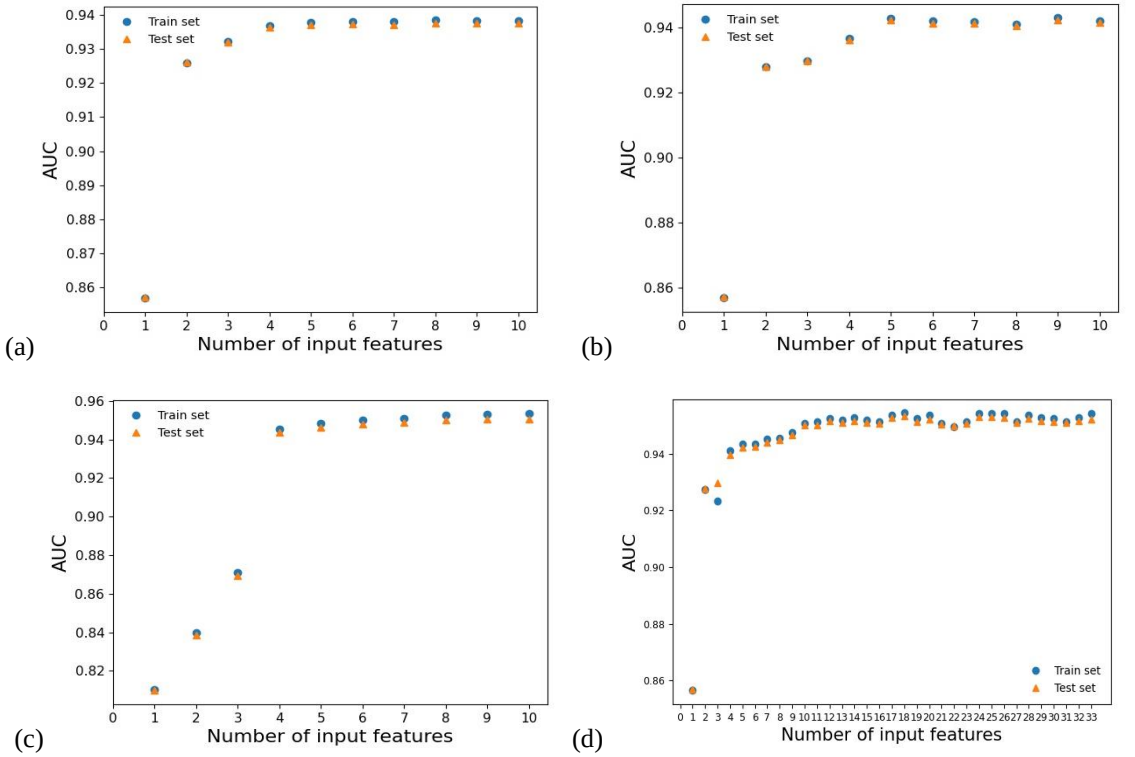


Figure 8 Performance of (a) DTs, (b) Random Forest, (c) Adaptive Boosting and (d) Neural Networks for different numbers of input features and comparing between the training and testing datasets reported by the AUC scores.

Results of evaluation metrics

After passing hyperparameter optimization and feature importance steps, the final results on model performances and corresponding numbers of input features are presented in **Table 3**. The results show both the accuracy and AUC scores on different ML models and compare the training and testing datasets

with the best number of input features. Neural Network and Adaptive Boosting models achieved the highest performance with the best number of input features of 10 and 8, respectively. On the other hand, DTs and Random Forest models obtained slightly lower performance with 5 numbers of input features. Since the hyperparameters of each ML model are optimized and the models are retrained accordingly, these results obtain reliability and improvement with respect to the previous work [25]. Without the optimization of model architectures, Neural Network underperformed slightly more than Adaptive Boosting model, while DTs and Random Forest models remain unaffected. Therefore, the hyperparameter tuning will be crucial for more sophisticated models such as Neural Networks.

The ROC curve comparing 4 ML models is illustrated in **Figure 9** which can be very useful to understand the performance of each binary classifier and provide a selection of discrimination thresholds for further statistical analysis.

Table 3 Evaluation metrics.

Model	Best Number of Input Features	Name of Input Features	Accuracy		AUC score	
			Train	Test	Train	Test
Decision Tree	5	nbjets ptj1 m12 njets ptj2	0.863	0.861	0.938	0.937
Random Forest	5	nbjets ptj1 ptj2 m12 njets	0.865	0.863	0.943	0.942
Adaptive Boosting	8	ptj1 m12 njets nbjets ptj3 mj1 ptj2 mj3	0.877	0.875	0.952	0.949
Neural Network	10	nbjets ptj1 ptj2 njets ptj3 pt12 mj2 m13 mj1 m12	0.878	0.876	0.953	0.951

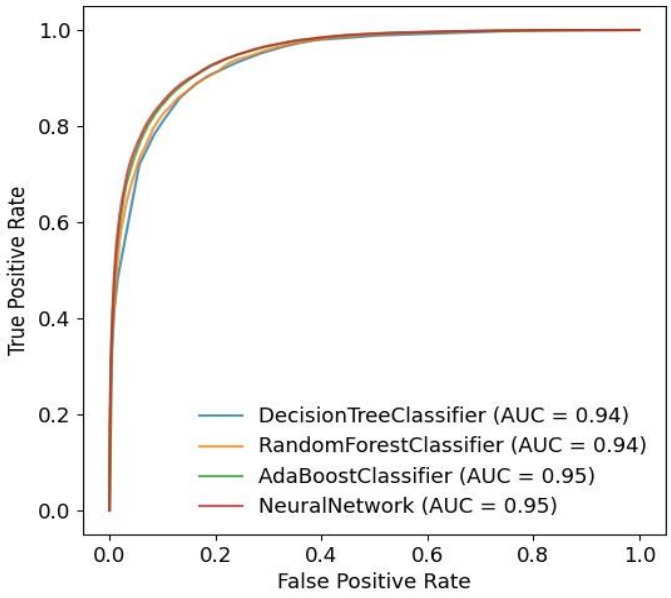


Figure 9 The ROC curves plotting TP rate versus FP rate at various thresholds for different models: DTs (blue), Random Forest (orange), Adaptive Boosting (green) and Neural Network (red).

Conclusions

The study shows that Tree-based and Neural Network algorithms can be applied to classify unique characteristics of the signal ($b\bar{b}H/bH$ production and $H \to b\bar{b}$ channel) from the background (heavy-flavor multi-jet production) process. Among the 4 models, Neural Network and Adaptive Boosting yield the highest classification scores of around 0.95. This is because both models can optimize their model parameters by learning from the error of prior prediction. The accuracy and AUC scores of the Adaptive Boosting model are very close to those of Neural Networks. This could be due to the characteristics of our signal process being distinguishable from the background process. When the ML models are applied to signal processes with very similar signatures to background processes, Neural Networks may show better discrimination power than the Tree-based models. Therefore, a further detailed study is still mandatory for the choice of algorithms, as it may change on a case-by-case basis. In any case, with some modifications, we finally can make use of our trained classifiers to perform statistical analysis on real collision data and search for BSM Higgs bosons at a wide mass range.

Acknowledgments

First of all, we would like to thank our project advisors; Dr. Chayanit Asawatangtrakuldee and Mr. Jatuporn Puntree who always supported and encouraged us and made this project possible, as well as Mr. Vichayanun Wachirapusitanand who helped answer many technical issues and gave some useful advice. Furthermore, we thank our beloved teammates for their patience and all hard work. It would not have been possible without one of us. We would also like to thank our friends who supported us all throughout the journey. Finally, our deepest gratitude to our parents for their constant motivation and encouragement, without their help and support it would not have been possible to complete this project.

References

[1] ATLAS Collaboration. Observation of a new particle in the search for the Standard Model Higgs boson with the ATLAS detector at the LHC. *Phys. Lett. B* 2012; **716**, 1-29.
[2] CMS Collaboration. Observation of a New Boson at a mass of 125 GeV with the CMS experiment at the LHC. *Phys. Lett. B* 2012; **716**, 30-61.
[3] ATLAS and CMS Collaborations. Combined measurement of the Higgs boson mass in pp collisions at $\sqrt{s}$ = 7 and 8 TeV with the ATLAS and CMS experiments. *Phys. Rev. Lett.* 2015; **114**, 191803.
[4] ATLAS Collaboration. Measurements of the Higgs boson production and decay rates and coupling strengths using pp collision data at $\sqrt{s}$ = 7 and 8 TeV in the ATLAS experiment. *Eur. Phys. J. C* 2016; **76**, 6.
[5] CMS Collaboration. Study of the mass and spin-parity of the Higgs boson candidate via its decays to Z boson pairs. *Phys. Rev. Lett.* 2013; **110**, 081803.
[6] H Yukawa. On the interaction of elementary particles. *Proc. Phys. Math. Soc. Jpn*. 1935; **17**, 48-57.
[7] ATLAS and CMS Collaborations. Measurements of the Higgs boson production and decay rates and constraints on its couplings from a combined ATLAS and CMS analysis of the LHC pp collision data at $\sqrt{s}$ = 7 and 8 TeV. *J. High Energy Phys.* 2016; **2016**, 45.
[8] ATLAS Collaboration. Observation of $H \to b\bar{b}$ decays and VH production with the ATLAS detector. *Phys. Lett. B* 2018; **786**, 59-86.
[9] CMS Collaboration. Observation of Higgs boson decay to bottom quarks. *Phys. Rev. Lett.* 2018; **121**, 121801.
[10] CMS Collaboration. Search for invisible decays of the Higgs boson produced via vector boson fusion in proton-proton collisions at $\sqrt{s}$ = 13 TeV. *Phys. Rev. D* 2022; **105**, 092007.
[11] A Djuadi. The anatomy of electro-weak symmetry breaking. Tome II: The Higgs bosons in the Minimal Supersymmetric Model. *Phys. Rep.* 2008; **459**, 1-241.
[12] GC Branco, PM Ferreira, L Lavoura, MN Rebelo, M Sher and JP Silva. Theory and phenomenology of two-Higgs-doublet models. *Phys. Rep.* 2011; **516**, 1-102.
[13] CMS Collaboration. Search for a Higgs boson decaying into a b-quark pair and produced in association with b quarks in proton-proton collisions at 7 TeV. *Phys. Lett. B* 2013; **722**, 207-32.
[14] CMS Collaboration. Search for neutral MSSM Higgs bosons decaying into a pair of bottom quarks. *J. High Energy Phys.* 2015; **2015**, 71.
[15] CMS Collaboration. Search for beyond the standard model Higgs bosons decaying into a $b\bar{b}$ pair in pp collisions at $\sqrt{s}$ = 13 TeV. *J. High Energy Phys.* 2018; **2018**, 113.

[16] ATLAS Collaboration. Search for heavy neutral Higgs bosons produced in association with *b*-quarks and decaying to *b*-quarks at $\sqrt{s}$ = 13 TeV with the ATLAS detector. *Phys. Rev. D* 2020; **102**, 032004.
[17] CMS Collaboration. Search for Higgs boson decay to a charm quark-antiquark pair in proton-proton collisions at $\sqrt{s}$ = 13 TeV. *Phys. Rev. Lett*. 2022. DOI: 10.48550/arXiv.2205.05550.
[18] P Baldi, P Sadowski and D Whiteson. Searching for exotic particles in high-energy physics with deep learning. *Nat. Commun*. 2014; **5**, 4308.
[19] J Alwall, R Frederix, S Frixione, V Hirschi, F Maltoni, O Mattelaer, HS Shao, T Stelzer, P Torrielli and M Zaro. The automated computation of tree-level and next-to-leading order differential cross sections, and their matching to parton shower simulations. *J. High Energy Phys.* 2014; **2014**, 79.
[20] T Sjöstrand, S Ask, JR Christiansen, R Corke, N Desai, P Ilten, S Mrenna, S Prestel, CO Rasmussen and PZ Skands. An introduction to PYTHIA 8.2. *Comput. Phys. Commun*. 2015; **191**, 159-77.
[21] JD Favereau, C Delaere, P Demin, A Giammanco, V Lemaître, A Mertens and M Selvaggi. DELPHES 3: A modular framework for fast simulation of a generic collider experiment. *J. High Energy Phys.* 2014; **2014**, 57.
[22] DE Hinkle, W Wiersma and SG Jurs. *Applied statistics for the behavioral sciences*. Rand McNally College Publishing, Chicago, Illinois, 1998.
[23] F Pedregosa, G Varoquaux, A Gramfort and V Michel, B Thirion, O Grisel, M Blondel, P Prettenhofer, R Weiss, V Dubourg, J Vanderplas, A Passos and D Coutnapeau. Scikit-learn: Machine learning in python. *J. Mach. Learn. Res*. 2011; **12**, 2825-30.
[24] F Chollet and others. *Keras*. Available at: https://keras.io, accessed 2015.
[25] J Waiwattana, P Saksirimontri, N Pitakkultorn, C Asawatangtrakuldee, V Wachirapusitanand and J Puntree. Search for BSM Higgs bosons with machine learning techniques. *In*: Proceedings of the Siam Physics Congress 2022, Nakhon Ratchasima, Thailand, 2022.