Fabio Cufino - Angelo Arisi

Advanced Laboratory course:
Particle physics

Data Analysis with IceCube Monte Carlo Simulation Data

October 15, 2024

technische universität
dortmund

Abstract

This analysis employs various machine learning techniques and statistical methods to achieve binary classification of signal and background neutrino events. The data, derived from Monte Carlo simulations of the IceCube experiment, serves to optimize the distinction between true neutrino signals and background noise, crucial for extracting meaningful information from detected events. The methodologies explored include the mRMR feature selection process, Naive Bayes classifier, k-Nearest Neighbors, and Random Forest classifier, each evaluated for their effectiveness in enhancing the accuracy of neutrino event classification.

Table of contents

1 Introduction

The IceCube Neutrino Observatory at the South Pole, allows a detailed exploration of the mass composition of primary cosmic rays in the energy range from about 100 TeV to 1 EeV. It is designed to look at point-like sources of neutrinos which, considering that high energy cosmic rays are not sensitive to magnetic fields, could contain information on the direction of the original source. However, the neutrino signals are swamped by the background of atmospheric cosmic ray muons. The distinction of background and signal events is then rather fundamental in order to extract information from the detected physical quantities. A useful tool to optimize the classification techniques are MonteCarlo simulations: simulations of an experiment, using random number generators, which provide pseudo-data based on known properties of signal and background events. Then, since the label is known, these simulations are useful to produce signal-background separation analyses.

2 ICE CUBE experiment

The IceCube Neutrino Observatory, situated at the geographic South Pole, is a highly specialized experiment for the detection of high-energy neutrinos and muons through the measurement of Cherenkov light in deep Antarctic ice. The detector comprises three main components: the in-ice array, DeepCore, and IceTop. Its structure is shown Figure 1.

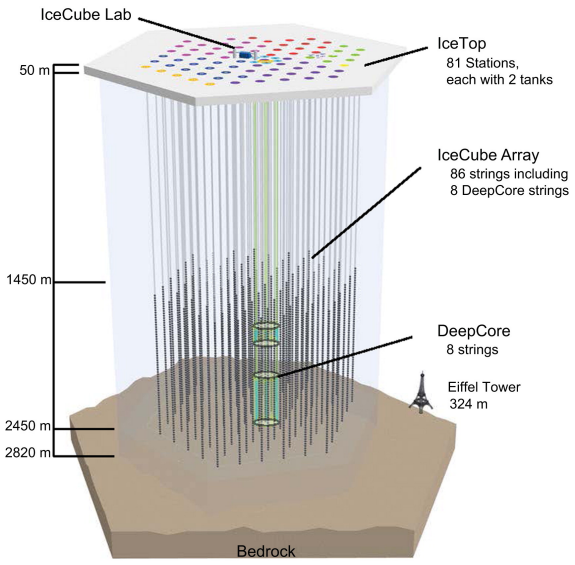

Figure 1: Structure of the IceCube detector, highlighting its 3 subdetectors.

The in-ice array, located at depths between 1450 and 2450 meters, includes 5160 photo-multipliers arranged along 86 strings. The photo-multipliers capture Cherenkov light emitted when high-energy charged particles exceed the phase velocity of light in the medium. The velocity c in the medium is given by $c = c_0/n$, where c_0 refers to the speed of light in vacuum and n to the refractive index of the medium.
The DeepCore, a more densely packed section of the array with a lower threshold of 10 GeV

compared to the approximately 100 GeV threshold of the in-ice array, allows for enhanced detection of lower-energy neutrinos due to its closer photo-multiplier spacing and higher sensitivity. The IceTop, positioned at the surface, detects cosmic ray air showers and serves as a veto for the in-ice array to distinguish between cosmic ray and neutrino-induced events. Neutrinos are detected through their interactions with ice nuclei, which occur via charged current (CC) or neutral current (NC) interactions as described by:

$$\begin{aligned} \nu_l(\overline{\nu_l}) + A &\rightarrow l^{\pm} + X(CC) \\ \nu_l + A &\rightarrow \nu_l + X(NC) \end{aligned} \tag{1}$$

where, ν_l represents a neutrino of type l, $l^{\pm}$ is the corresponding charged lepton and $\overline{\nu_l}$ is the antineutrino. A is the target nucleus, and X a hadronic final state.
In the detector, electrons produce approximately spherical cascades, whereas muons, due to their lower energy loss rate, form extended tracks of Cherenkov light. Tau leptons, due to their short lifetime, generally produce signatures similar to electrons unless at very high energies. Neutral current interactions result in hadronic cascades, which are also observable through their Cherenkov light emissions. The detection focuses on Cherenkov light generated by secondary particles like electron-positron pairs and photons from these interactions, allowing for detailed study of high-energy astrophysical phenomena.
The use of events interacting within the detector, known as starting events, is an analytical method where the outer sections of the detector function as a veto to eliminate atmospheric muons. In this approach, the effective volume considered for analysis is smaller than the total detector volume. All neutrino flavors contribute similarly, resulting in a significant number of cascade events from neutral current interactions or charged current interactions involving electron and tau neutrinos. These cascade events provide good energy resolution but poor angular resolution.
Conversely, traces of through-going muons exhibit poor energy resolution but good angular resolution, and depending on their energy, they can travel over long distances. This characteristic allows the analysis to extend the effective volume beyond the physical detector volume by utilizing the Earth as a shield against atmospheric muons. Muons arriving from below the detector must originate from neutrino interactions, as they penetrate through the Earth. By applying a cut based on the reconstructed zenith angle, it is possible to distinguish between atmospheric neutrinos and muon neutrinos, assuming perfect directional reconstruction. However, due to imperfect reconstruction for a small fraction of events, this cut only enhances the signal-to-noise ratio from $1 : 10^6$ to $1 : 10^3$.

To further separate incorrectly reconstructed muons from muon neutrinos, machine learning techniques are employed. Improving the signal-background separation using this analysis approach is the primary objective of this exercise.

3 Statistical methods

The goal of this analysis is to perform a binary classification of signal and background events, from Monte Carlo simulated IceCube data. The simulated dataset contains hundreds of features with varying information content. In order to reduce the complexity and the computation time it is necessary to select only a subset of features, which ideally maximizes the information content regarding the target. This process is called attribute selection.

3.1 Attribute selection

There are several different methods to perform attribute selections which are based on different algorithms, but all aimed at maximazing the information gain. In this analysis specifically, the mRMR selection was used.

3.1.1 mRMR

The minimum Redundancy, Maximum Relevance selection method , or mRMR, aims to select features that are highly relevant to the target variable while minimizing redundancy among themselves.

The relevance $\mathcal{R}$ of a feature x_i to the target variable y is quantified using mutual information $I(x_i, y)$:

$$I(x, y) = \int p(x, y) \log \left(\frac{p(x, y)}{p(x)p(y)} \right) dxdy, \tag{2}$$

while the redundancy $\mathcal{D}$ among a set of selected features S is defined as the average mutual information between pairs of features x_i and x_j:

$$\mathcal{D} = \frac{1}{|S|^2} \sum_{x_i, x_j \in S} I(x_i, x_j) \tag{3}$$

The objective is to maximize $\mathcal{R} - \mathcal{D}$, which is formally expressed as:

$$\text{mRMR}(x_i) = I(x_i, y) - \frac{1}{|S|} \sum_{x_j \in S} I(x_i, x_j) \tag{4}$$

This balance ensures that the selected features provide the most informative representation of the target variable while avoiding redundancy, then improving the efficiency and performance of the model.

3.1.2 Jaccard Index

The Jaccard index assess the validity of an attribute selection algorithm on a specific dataset. In particular, it checks the stability of the attribute selection against statistical fluctuations in the dataset. If the attribute selection is performed for two different subsets of the data,

identifying with F_a the set of features for the first selection and with F_b the set of features for the second selection, the Jaccard index is defined as:

$$J(F_a, F_b) = \frac{|F_a \cap F_b|}{|F_a \cup F_b|} \tag{5}$$

where $0 < J(F_a, F_b) < 1$. Values close to 1 indicate a similar selection. The process can be then performed for a number l of subsets and the Jaccard score can be computed as follows:

$$J = \frac{1}{l(l-1)} \sum_{i=1}^{l} \sum_{j=i+1}^{l} J(F_a, F_b) \tag{6}$$

A stable attribute selection has a Jaccard score close to 1.

3.2 Classification

For the binary classification, three different methods were used: the Naive Bayes classifier, the K Nearest Neighbours and a Random Forest classifier.

3.2.1 Naive Bayes Classifier

The Naive Bayes classifier is based on Bayes' theorem, which describes the posterior probability of an hypothesis given the result of a measure, based on a prior probability of the hypothesis:

$$P(C \mid x) = \frac{P(C) \cdot P(x \mid C)}{P(x)} \tag{7}$$

In a binary classification problem, where for example C refers to the signal and $\overline{C}$ to the background, the value of the coefficient Q:

$$Q = \frac{P(C) \cdot P(x \mid C)}{P(\overline{C}) \cdot P(x \mid \overline{C})} \tag{8}$$

is an indicator of the likelihood of an event being signal ($Q > 1$) or background ($Q < 1$). For multiple features $(x_1, x_2, \ldots, x_n)$, under the key assumption that the features in the dataset are conditionally independent given the class label, the global probability is simply the product probability. The coefficient Q is then expressed as:

$$Q = \frac{P(C \mid x_1 \ldots x_n)}{P(\overline{C} \mid x_1 \ldots x_n)} = \frac{\prod_{i=1}^{n} P(C \mid x_i)}{\prod_{i=1}^{n} P(\overline{C} \mid x_i)} \tag{9}$$

3.2.2 kNN

The k-Nearest Neighbors (kNN) classifier is a non-parametric, supervised learning algorithm used for classification and regression tasks. It works by identifying the k closest training examples in the feature space to a given query point. Secondly, it identifies the class of said

point as the most present between the k neighbours, or it outputs a probability which is the mean of the classes of the k neighbours. In the case, to get an actual label, a threshold value has to be chosen. The distance between the query point and the training examples depends on the metric chosen. The most commonly used is the euclidean distance:

$$d(x, y) = \sqrt{\sum_{i=1}^{n}(x_i - y_i)^2} \tag{10}$$

Other than the metric, the kNN algorithm can depend highly on the number of nearest neighbours k.

3.2.3 Random Forest

The Random Forest classifier is a learning algorithm based on decision trees. A single tree operates by recursively partitioning the feature space into subsets based on the value of the input features, until a certain depth is reached, creating a tree-like model of decisions. In order to minimize over-training, each tree in the forest is trained on a different subset of the data, created through a process known as bootstrapping, where random samples with replacement are taken from the original dataset. Furthermore, in order to increase the model's ability to generalize, the decision trees could also be built using a random subset of features at each split. The final classification is determined by aggregating the predictions from all the individual trees. Specifically, in the Breiman implementation, the final decision c is the arithmetic mean of the different decision from the N trees:

$$c = \frac{1}{N}\sum_{i=1}^{N} P_i \tag{11}$$

3.2.4 Quality parameters

There are different methods to asses the goodness of a classification. From a binary classification problem there a only four types of outcome for a single event: true positive (tp), false positive (fp), true negative (tn) and false negative (fn). Using these, the following indexes can be calculated:

$$\text{precision} = \frac{tp}{tp + fp} \tag{12}$$

$$\text{recall} = \frac{tp}{tp + fn} \tag{13}$$

$$f_\beta = (1 + \beta^2)\frac{p \cdot r}{\beta^2 p + r} \tag{14}$$

The different types of outcome strongly depend on the chosen value τ_c of the threshold. Another useful tool to check the performance of classification algorithms are the ROC curve and the precision-recall curve. The ROC curve plots the true positive rate against the false positive rate for different values of the threshold τ_c. They are defined as:

$$TPR(\tau_c) = \frac{tp(\tau_c)}{tp(\tau_c) + fn(\tau_c)} \tag{15}$$

and

$$FPR(\tau_c) = \frac{fp(\tau_c)}{fp(\tau_c) + tn(\tau_c)} \tag{16}$$

The area under the ROC curve A_{ROC} is a quality parameter, which is equal to 1 for a perfect classification model and equal to 0.5 for a random guess. Values of the area $A_{ROC} < 0.5$ mean that the model inverts the classes which can be corrected by simply inverting the labels.

4 Analysis

The analysis starts by importing the three different datasets: the signal dataset, the background dataset, and the test dataset. These datasets were sourced from CSV files and imported into the analytical environment.
The signal and background datasets contain the 'label' feature that ensure the distinction between the nature of a single event. The test dataset instead does not contain any labels at all.

4.1 Data preparation

A thorough data cleaning process was executed in order to ensure that the datasets were adequately prepared for accurate and unbiased analysis. The steps followed are here briefly described in chronological order.

The initial step involved verifying the dimensions of each dataset. The signal dataset contained 17,933 rows and 283 features; the background dataset comprised 18,067 rows and 283 features; and the test dataset included 4,000 rows and 282 features.
We then performed **features intersection** to maintain consistency between the signal and background datasets. Only the features that were included in both datasets were retained. This essential step ensured that both datasets were comparable and compatible for the analysis.
Subsequently, features related to Monte Carlo simulations were removed to avoid introducing bias into the analysis. Columns containing the terms 'Corsika', 'Weight', 'MC' and 'I3EventHeader' were excluded from both the signal and background datasets.
It was then necessary to handle **missing values** such as the presence of NaN (Not a Number) and infinite values in the datasets. Features from the signal and background datasets that had more than 95% of their values as NaN were removed. This decision was made to eliminate features that were predominantly missing data, which could skew the analysis. For the remaining incomplete features, the percentage of missing values was below 1%, so the missing values were inputted using the mean value of the feature. This input method ensured that the datasets remained as complete as possible without introducing significant bias.
Some features contained only a single unique value across all rows. Those feature were removed from the dataset since they do not provide any information for the analysis or for training machine learning models. Finally, **index resetting** was necessary. The indices of the signal and background datasets were reset. This step facilitated seamless data manipulation and analysis by ensuring a clean and consecutive ordering of rows. Proper indexing is crucial for maintaining the integrity of data during subsequent analyses and manipulations.
This pre-processing was critical for maintaining the integrity and reliability of the analytical results. All the features removed from signal and background datasets were also removed from the test dataset.
The process ends with the concatenation of the signal and background datasets, performing a consequent randomization of the entire dataset. This key passages assured the impossibility

of recognizing, a priori, a background event from a signal event, thereby making the dataset suitable for training.

4.2 Feature Selection

The feature selection was performed using the mRMR method described in Section 3.1.1, using 15 as the number of features to select from all the columns of the dataset. This number was chosen as a compromise between having a high performance for the classification algorithms and reducing the computation time and redundant information as much as possible. To evaluate the stability of the feature selection method on the dataset, the Jaccard score described in Section 3.1.2 was used. The dataset was split into 10 non-overlapping subsets, and for each one the mRMR feature selection algorithm was applied to identify and record the chosen features. The different sets of feature were used to compute the Jaccard score according to equation 6 obtaining a result of J=1. This means that the same exact features were selected for each subset, assessing the stability of the method used.
Figure 2 shows an overview of all selected features and their distributions highlighting signal and background events.

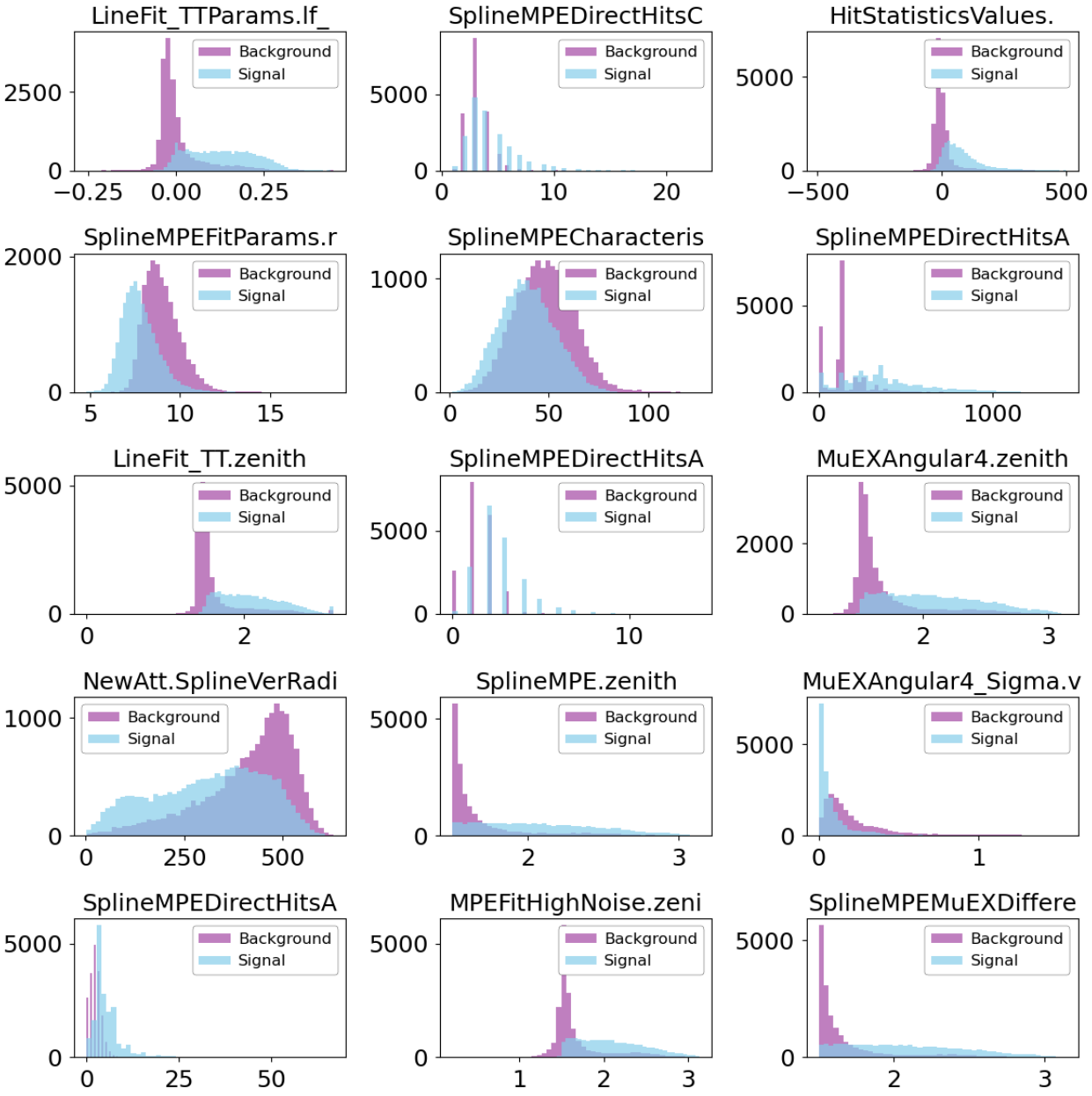

Figure 2: Distribution of the 30 features selected using the mRMR method. Light blue for signal events and purple for background events.

4.3 Multivariance analysis

The multi-variance analysis was then performed on the dataset with the selected features. The dataset was scaled with Standard Scaler and then split randomly between train and test, with the proportion of 80% training and 20% test. Three different classification methods were then used.

4.3.1 Naive Bayes Classifier

The first method applied was the Naive Bayes Classifier described in section 3.2.1. Here specifically, the Gaussian Naive Bayes algorithm was used, where the likelyhood of the features is a priori assumed to be Gaussian. This simple algorithm gives an accuracy score of $a = 0.85$ for a $\tau_c = 0.5$. Figure 3 and Figure 4 show the ROC curve and the confusion matrix.

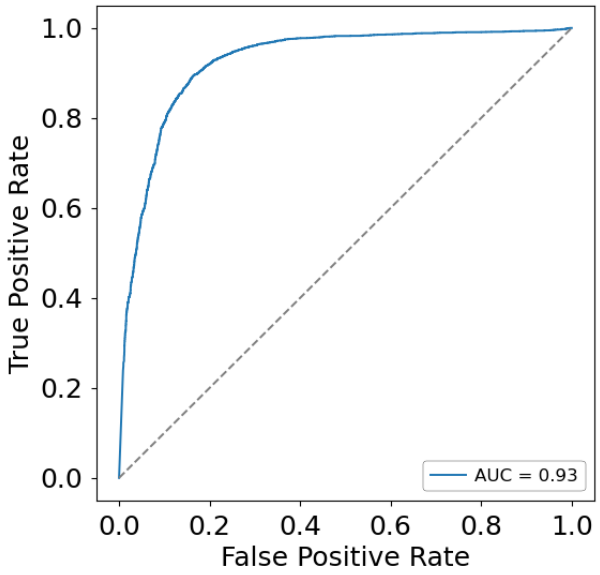

Figure 3: ROC curve of the model. It rapresents the ture positive rate against false positive rate for every threshold value τ_c.

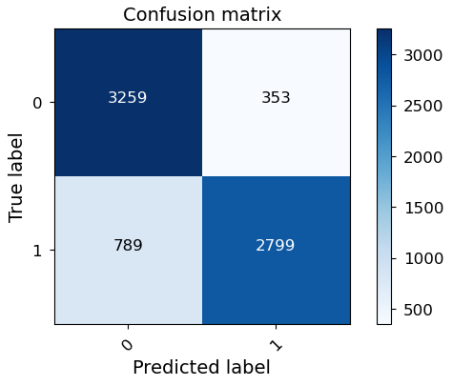

Figure 4: Confusion matrix using a threshold value of $\tau_c = 0.5$.

4.3.2 KNN

The kNN method explained in section 3.2.2 was then applied. The number of nearest neighbours was chosen to be $k = 5$ and euclidean metric was used to calculate the distance. In order to avoid the bias due to the specific train-test split, cross validation was performed with a number of subset equal to 5. The total accuracy was then computed as the mean of the 5 different accuracy values with a final value of $a = 0.96$. The threshold value to assign the label was chosen at $\tau_c = 0.5$. Figure 5 and Figure 6 show the ROC curve and the confusion matrix.

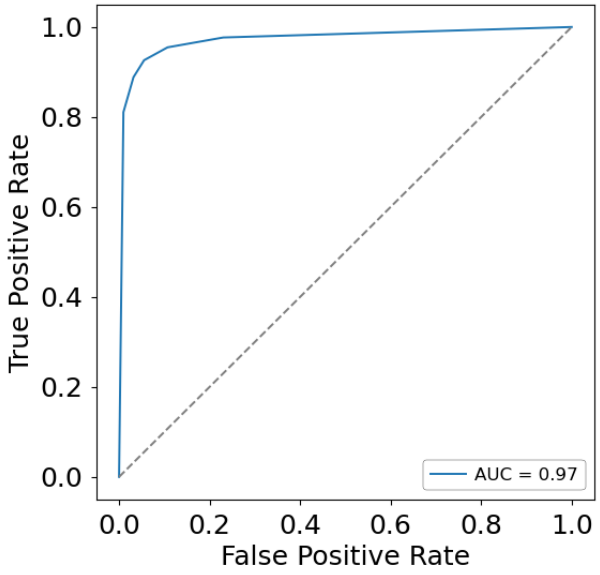

Figure 5: ROC curve of the model. It rapresents the ture positive rate against false positive rate for every threshold value τ_c.

The high value of the area under the curve $A_{ROC} = 0.98$ indicates a very good performance for the classifier.

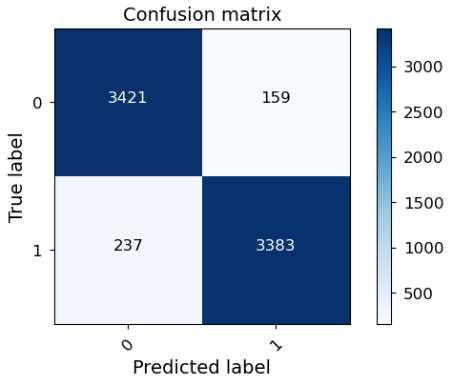

Figure 6: Confusion matrix using a threshold value of $\tau_c = 0.5$.

4.3.3 Random Forest

The Random Forest Classifier was also applied to the dataset. The number of estimators was set to 100, while the same number of subset $n = 5$ was used for cross validation. The accuracy obtained was $a = 0.96$ using $\tau_c = 0.5$. Figure 7 and Figure 8 are show the ROC curve and the confusion matrix.

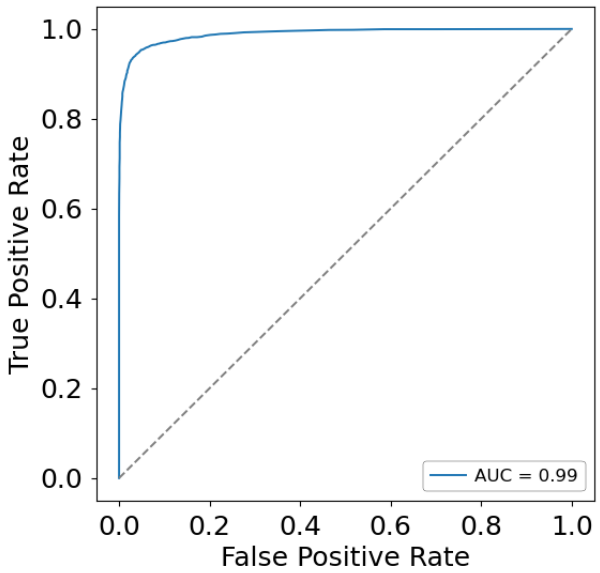

Figure 7: ROC curve of the model. It rapresents the ture positive rate against false positive rate for every threshold value τ_c.

The ROC curve gives an area of $A_{ROC} = 0.99$. It is clear that the random forest classifier performs better than the previous algorithms.
In the context of classification algorithms, a common practice has been to utilize a threshold of $\tau_c = 0.5$ to translate predicted probabilities into definitive class labels. Although intuitive, a value of 0.5 is not always the best choice depending on the objective of the analysis.

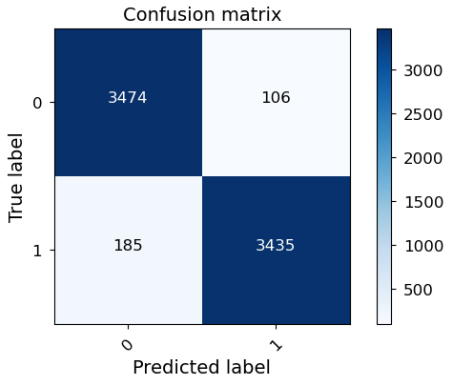

Figure 8: Confusion matrix using a threshold value of $\tau_c = 0.5$.

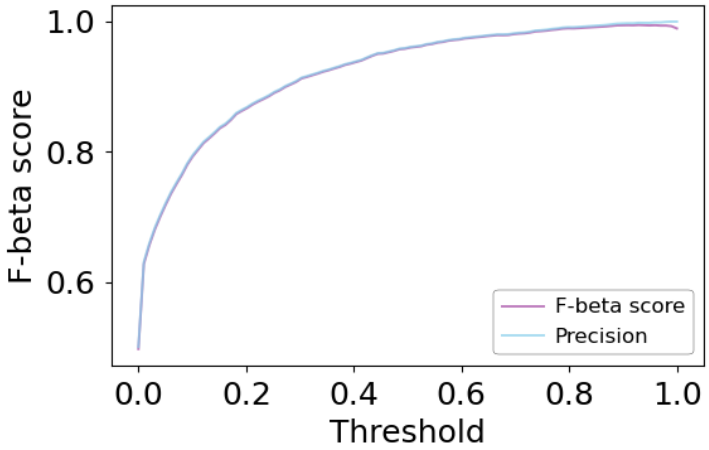

Figure 9: Plot of the precision curve (blue) and the f_β curve (purple).

A useful way to pick the most functional threshold is to choose the one which minimizes the f_β index defined according to the Equation 14. For a the fixed value of $\beta = 0.1$ the behaviour of the index as a function of the threshold value, is almost identical to the one of the precision index, as shown in Figure 9.

As shown in the plot, the value that maximizes the f_β is $\tau_c = 0.92$. Figure 10 shows the behavior for precision and recall as a function of the threshold.

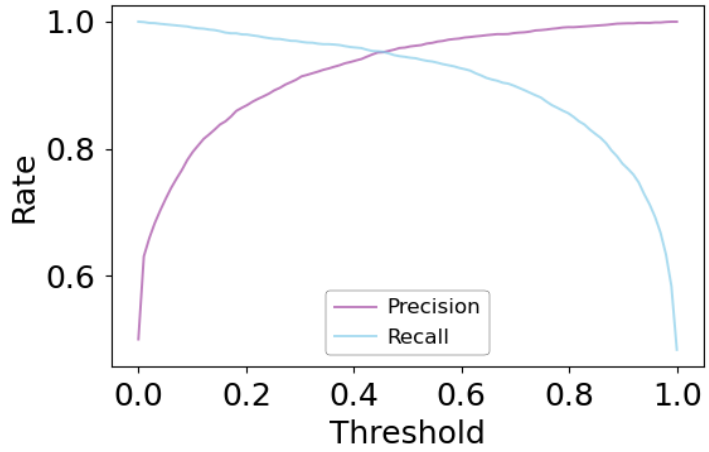

Figure 10: Plot of the precision index (purple) and recall index (blue) as a function of threshold value τ_c.

We see how $\tau_c = 0.92$ guarantees a very high precision without decreasing significantly in terms of recall. Finally, Figure 11 and Table 1 show the confusion matrix and the values for precision, recall and accuracy for $t_c = 0.92$.

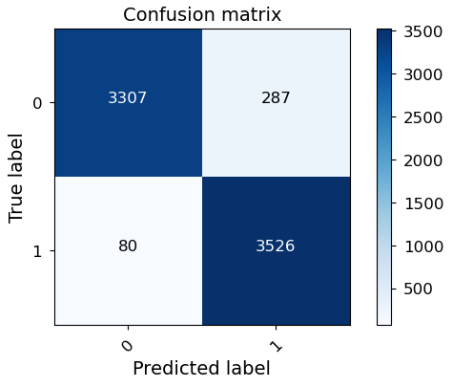

Figure 11: Confusion matrix using a threshold value of $\tau_c = 0.92$

Precision	*Recall*	*Accuracy*
1.00	0.47	0.74

Table 1: Performance metrics of the Random Forest classificator with $\tau_c = 0.92$.

5 Conclusion

This analysis successfully demonstrates the utility of machine learning techniques in classifying neutrino events from the IceCube experiment's Monte Carlo simulation data. By applying the mRMR selection, we effectively reduced the feature space to a manageable subset of highly informative features, ensuring a balance between relevance and redundancy. This selection process was validated through the Jaccard index, which confirmed the stability of our attribute selection approach against statistical fluctuations in the dataset.
The performance of three different classifiers, Naive Bayes, k-Nearest Neighbors, and Random Forest, was evaluated. The Gaussian Naive Bayes classifier achieved an accuracy of 85%, demonstrating the fundamental capability of probabilistic models in this context. The kNN classifier, with an optimal choice of k=5 and cross-validation, improved accuracy to 94%, showcasing the strength of distance-based methods in handling this dataset. Overall, the Random Forest classifier peformed better than the other methods used, with a final accuracy of 96% considering a $\tau_c = 0.5$.
In conclusion, by setting the new threshold at $\tau_c = 0.92$, despite a decrease in recall, we have enhanced the algorithm's ability to detect signal events effectively.

References

[1] R. Abbasi et al. “IceTop: The surface component of IceCube.” In: Nuclear Instruments and Methods in Physics Research Section A: Accelerators, Spectrometers, Detectors and Associated Equipment 700.0 (2013), pp. 188–220. issn: 0168-9002. doi: 10.1016/j.nima.2012.10.067. url: http://www.sciencedirect.com/science/article/pii/S016890021201217X.

[2] R. Abbasi et al. “The design and performance of IceCube DeepCore.” In: Astroparticle Physics 35.10 (2012), pp. 615–624. issn: 0927-6505. doi: http://dx.doi.org/10.1016/j.astropartphys.2012.01.004. url: http://www.sciencedirect.

com/science/article/pii/S0927650512000254. [3] A. Achterberg et al. “First year performance of the IceCube neutrino telescope.” In: Astroparticle Physics 26.3 (2006), pp. 155–173. issn: 0927-6505. doi: 10.1016/j.astropartphys.2006.06.00. url: http://www.sciencedirect.com/science/article/pii/S0927650506000855.

[4] L. Breiman. “Random forests.” In: Machine learning 45.1 (2001), pp. 5–32.

[5] T. Fawcett. “ROC graphs: Notes and practical considerations for researchers.” In: Machine learning 31.1 (2004), pp. 1–38. url: https://www.hpl.hp.com/techreports/2003/HPL-2003-4.pdf(visitedon06/29/2023).

[6] E. Fermi. “On the Origin of Cosmic Radiation.” In: Physical Review 75 (1949), pp. 1169–1174.

[7] IceCube Collaboration. “Neutrino emission from the direction of the blazar TXS0506+056 prior to the IceCube-170922A alert.” In: Science 361.6398 (2018), pp. 147–151. doi: 10.1126/science.aat2890. url: https://www.science.org/doi/abs/10.1126/science.aat2890.

[8] IceCube Collaboration. “Observation of high-energy neutrinos from the Galactic plane.” In: Science 380.6652 (2023), pp. 1338–1343. doi: 10.1126/science.adc9818. url: https://www.science.org/doi/abs/10.1126/science.adc9818.

[9] J.-H. Koehne et al. “PROPOSAL for muon propagation.” In: Computer Physics Communications 184 (9 2013). doi: 10.1016/j.cpc.2013.04.001. url: http://dx.doi.org/10.1016/j.cpc.2013.04.001.