Advanced Laboratory course

Data Analysis with IceCube Monte Carlo Simulation Data

Theodor Zies
theodor.zies@tu-dortmund.de

Can Toraman
can.toraman@tu-dortmund.de

Colloqium: 21.05.2024

Hand-in: 12.06.2024

TU Dortmund – Physics department

Contents

1 Goals

The goal of this analysis is the classification of neutrino measurements in Monte Carlo simulations from the IceCube experiment. First, the minimum redundancy, maximum relevance (mRMR) selection is used to determine the best features. This is directly followed by the application and comparison of three different machine learning algorithms for the classification.

2 Theory

Before starting the data analysis, it is important to investigate the origin of the measured particles and understand the applied algorithms.

2.1 Cosmic Rays

So-called "Cosmic Rays" consist of highly energetic particles, for examples protons, electrons, different nuclei as well as neutrinos. The study of these particles has been a point of interest for many centuries but the term "Cosmic rays" was established in the early 20-th century. Incoming cosmic rays usually interact with the matter in the earths atmosphere creating particle showers. These incoming particles originate from a variety of different sources in the cosmos, some examples are active galactic nuclei or supernovae. Their energy spectrum is described by

$$\frac{\mathrm{d}\Phi}{\mathrm{d}E} = \Phi_0 E^{\gamma},$$

where $\gamma \approx -2.7$ is the so-called spectral index. The energy carried by these particles can range up to $10^{20}\,\mathrm{eV}$.

2.2 Atmospheric and astrophysical leptons

The showers caused by cosmic rays create a range of different particles. High energetic muons and neutrinos originate mostly from the decay of lighter mesons in atmospheric particle showers. In the IceCube experiment these are referred to as *conventional* and form a large background in the data. As the lighter mesons loose some of their energy the resulting energy spectrum for the muons and neutrinos is lower and follows a $E^{-3.7}$ proportionality. If the muons and neutrinos originate from heavier hadrons (e.g Λ-baryons or D-mesons), their energy spectrum will much more resemble those from astrophysical ones. This is caused by the short lifetime of the heavier hadrons, leaving them less time to loose energy. When a particle originating from the decay of a short lived particle is detected it is called *prompt*. From the prompt particles, the ones originating form the decay of a neutrino form the signal.

2.3 mRMR Selection

The mRMR selection algorithm is implemented using python library `mrmr-selection`. This algorithm works by maximizing the correlation to the target and tries to minimize the correlation between the individual features. This is achieved using the joint information

$$I(x,y) = \int p(x,y) \log\left(\frac{p(x,y)}{p(x)p(y)}\right) \mathrm{d}x\mathrm{d}y,$$

where $p(x), p(y), p(x,y)$ are the probability densities of the different features. It is a feature selection algorithm independent of the type of learner applied.

2.4 Naive Bayes Classifier

The Naive Bayes Classifier uses a Bayesian approach in order to classify candidates as signal A or background $\bar{A}$. It assumes the probability $p(B|A)$ to be background/signal with the given feature V is only dependent on B (naive). The classifier calculates

$$Q = \prod_{i=1}^{n} \frac{p(B_i|A)}{p(B_i|\bar{A})},$$

which is > 1 for a signal and < 1 for a background prediction.

2.5 Random Forest Classifier

A random forest classifier uses k binary decision trees, which try to determine ideal cuts on the given features by minimizing a gini-impurity function

$$\mathrm{Gini}(p_i) = 1 - \sum_{i=1}^{k} {p_i}^2,$$

where p_i is the probability of a candidate belonging to a class i. Each tree is trained on a random subset of the data in order to avoid over training. The determined result of each tree is then averaged or the majority voting is used depending on the implementation.

2.6 Multi-Layer Perceptron Classifier

The Multi-Layer Perceptron Classifier (MLPClassifier) is a classic neural network with Limited-memory Broyden–Fletcher–Goldfarb–Shanno algorithm (LBFGS) loss function [2]. It is trained using labeled data. As the MLPClassifier usually is implemented with multiple hidden layers it allows classification of non linear problems.

2.7 Evaluation of the Models

All the above mentioned models are evaluated using a range of metrics. During the classification, a subsample of the training data is used to check the models for overtraining.

This procedure is further described in subsubsection 5.3.1.After the training the accuracy a and the precision p are determined on a test data sample using

$$\begin{aligned} a &= \frac{\mathrm{TP} + \mathrm{TN}}{\mathrm{TP} + \mathrm{TN} + \mathrm{FP} + \mathrm{FN}}, \\ p &= \frac{\mathrm{TP}}{\mathrm{TP} + \mathrm{FP}}, \\ r &= \frac{\mathrm{TP}}{\mathrm{TP} + \mathrm{FN}}, \end{aligned}$$

where TP, TN, FP and FN are the counts of correctly classified signal (TP) and background (TN) and respectively falsely classified signal (FP) and background (FN). Ideally both values are close to 1. Additionally , the f_β score

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

is calculated using the Recall r. By using a $\beta \neq 1$, it is possible to give a higher importance to the precision p or recall r, respectively. Lastly the Receiver Operating Characteristic (ROC) curve is determined by plotting the True Positive Rate

$$\mathrm{TPR} = \frac{\mathrm{TP}}{\mathrm{TP} + \mathrm{FN}}$$

against the False Positive Rate

$$\mathrm{TPR} = \frac{\mathrm{FP}}{\mathrm{FP} + \mathrm{TN}}$$

for different cut values τ_c on the prediction of the classifier. A metric for the quality of the classification is the area under the aforementioned curve. A value of 0.5 results from random guessing whereas a value of 1 is ideal.

3 The IceCube Experiment

The IceCube experiment is located in the antarctic and consists of a huge array of photomultipliers inside the ice. The primary goal is the measurement of the origin and energy of astrophysical neutrinos. The experiment is subdivided into the in-ice array, DeepCore, and IceTop. This subdivision is due to the different densities of photomultipliers on each cable. The DeepCore for example has the highest photomultiplier density and allows for measurements in a lower energy range (10 GeV), where as the in-ice array limit lays around 100 GeV. Additionally the IceTop, positioned above the DeepCore and the in-ice array, is used as a shower detector and simultaneously as a veto for Conventional neutrinos. In total 5160 photomultipliers are placed at a depth of 1450 - 2450 m. These are used in order to measure the Cherenkov light emitted by charged particles exceeding the speed of light inside the medium they are traversing.

Since neutrinos only interact via the weak interaction, their direct measurement is impossible. Therefore the IceCube experiment looks for particles created by weak interactions of neutrinos with water molecules. These interactions are mediated through charged and neutral currents (CC and NC) as seen in Equation 2 (CC) and Equation 3 (NC).

$$\nu_l(\bar{\nu}_l) + A = l^{\mp} + X \tag{2}$$

$$\nu_l + A = \nu_l + X \tag{3}$$

Leptons created in CC are detected using the shape of the signature they leave in the detector. Electronic signals appear round whereas the signal generated by muons are longer as they loose their energy slower than an electron. Tau signatures are harder to detect as they generate two round signals, one from the fast decaying tau, the second one from the electron created during the tau decay. Particles from NC are detected by cascades created from hadronic particles, which leave a signature similar to the electronic one.

The different types of signatures also impact the quality of the measurement. Muons that are created outside of the detector but still leave a track have a high energy uncertainty. The measurement of the angle however is much more precise compared to the electron. On the other hand, the electrons energy measurement carries a much smaller uncertainty.

The data samples used in this analysis are generated in Monte Carlo Simulations of the experiment. This allows for labeling of the so-called pseudo data. This data is used in the training of the applied algorithms.

4 Tasks

As already mentioned, the goal is the classification of events into ones consisting of either signal or background muons. This is archived using the mRMR selection algorithm and three learners. Their performance is determined and compared. The learners produce a probability for each candidate to be prompt or conventional, the cut τ_c on the probability value is determined by maximizing the f_β score, with $\beta = 0.1$.

5 Data analysis

In total, three different data sets are available for this analysis. The training of the classifiers will be performed using the two datasets `signal_train.csv` and `background_train.csv`, while the `test.csv` file will be used to test the created model.

5.1 Data preprocessing

The events in the signal and background training datasets were produced via Monte Carlo simulations. Consequently, the datasets contain Monte Carlo truths, event ids and weights which should not be used during the training process, as they are not present in

actual data. All attributes with a name containing the key words `Weight`, `MC`, `Corsika`, or `I3EventHeader` are removed from the datasets. The training data is already labeled with 0 for background and 1 for signal. These labels are removed and stored, as they should only be used for classifier training and validation purposes. Some entries contain `Nans` or `Infs`, which have to be removed. As a first step, all attributes where more than 10 % of entries are `Nans` or `Infs` are removed. The remaining invalid values are adressed by deleting all events where any attribute is `Nan` or `Inf`. Lastly, the data is rescaled by removing the mean and scaling to unit variance.

5.2 Attribute selection

Before the multivariate section can be performed, a selection of suitable attributes is needed, which the classifier can use. The goal is to choose attributes which contain the most information about wether an event is signal or background. Using all available attributes is avoided as this can lead to overtraining and increases computational time. Additionally, too many features result in a large parameter space, this can lead to problems later on depending on the type of classifier used. The selection is performed using the mRMR method described in subsection 2.3. Using the package `mrmr_selection` [1], the 100 best features are determined in ascending order. In Figure 1, the distributions of the four best features for signal and background events are shown.

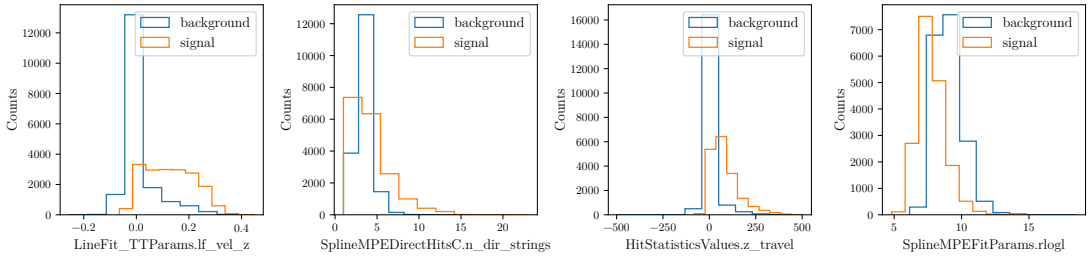

Figure 1: Distributions of the four best features determined via the mRMR method, for signal and background events.

All distributions show a good seperation of signal and background, which is excpected from the best features determined by the mRMR method.

5.3 Multivariate selection

In this section, three different models are trained and compared. All models are implemented from the `scikit-learn` libary [3].

5.3.1 Naives Bayes

The first model used is the Naives Bayes classifier. To determine the optimal amount of features, the model is trained and tested multiple times with different numbers of attributes. For each case, the best attributes determined in subsection 5.2 are used,

e.g. at first only the best five features are selected, then the best ten features and so on. Quality parameters are computed for each case and plotted against the number of features in Figure 2. The accuracy, precision and ROC score are chosen as the quality parameters for all three models tested in this analysis.

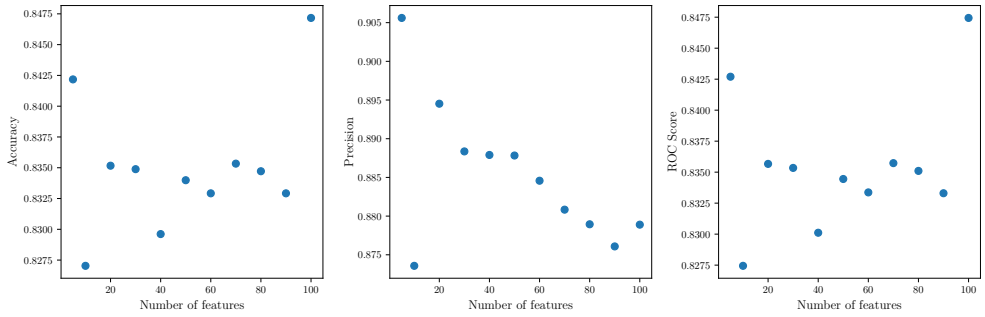

Figure 2: Quality parameters of the Naive Bayes classifier for a varying number of features.

Since the Naive Bayes is a very simple model, adding more then five features only leads to random fluctuations in the model performance. This is excpected, as the model has a low capacity and cannot make use of the additional information provided when more features are given. This is why only the five best attributes are used to train this classifier. To validate the classifiers performance, a k-folding algorithm is used with five splits. This means that the data is split in five parts, the classifier is then trained on four parts and tested using the remaining part. In this fashion, five classifiers are each trained and tested on different subsets. If the model is working good the results should be similar for all choices of subsets. The scores of the Naive Bayes classifier are determined to be

$$\begin{aligned} \text{Accuracy} &= 0.8432 \pm 0.0019\,, \\ \text{Precision} &= 0.907 \pm 0.004\,, \\ \text{Recall} &= 0.7660 \pm 0.0022\,, \\ \text{ROC Score} &= 0.9278 \pm 0.0017\,. \end{aligned}$$

The errors are very small, meaning the model performs independently of the specific subset of data it was trained on. The optimal decision threshold for the predicted probabilities is calculated using the f_β score. Here, $\beta = 0.1$ is used, meaning that much more importance is placed on the precision of the classification rather then the accuracy. The f_β score is computed for each possible threshold τ_c, then the maximum value is determined and consequently the best value for τ_c. The score and its maximum are displayed in Figure 3.

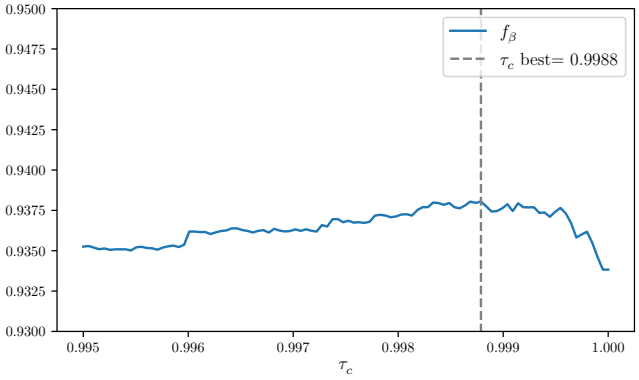

Figure 3: The f_β score of the Naive Bayes classifier and its maximum value.

At the maximum, the best value of the cut $\tau_{c,\,\text{best}}$ is also given in Figure 3. After applying the determined cut on the predicted probabilities of the Naive Bayes model, a confusion matrix shown in Figure 4 is created.

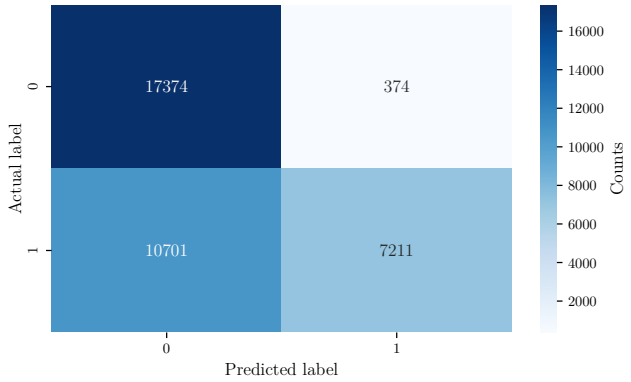

Figure 4: Confusion matrix for the Naive Bayes classifier.

Due to the low capacity of the model and the high focus on precision, a lot of signal events are missed.

5.3.2 Random Forest

The steps performed for the Random Forest are identical to the ones from the Naive Bayes model. The Random Forest is initialized with 100 trees. The optimal number of features is again chosen by evaluating the scores in Figure 5.

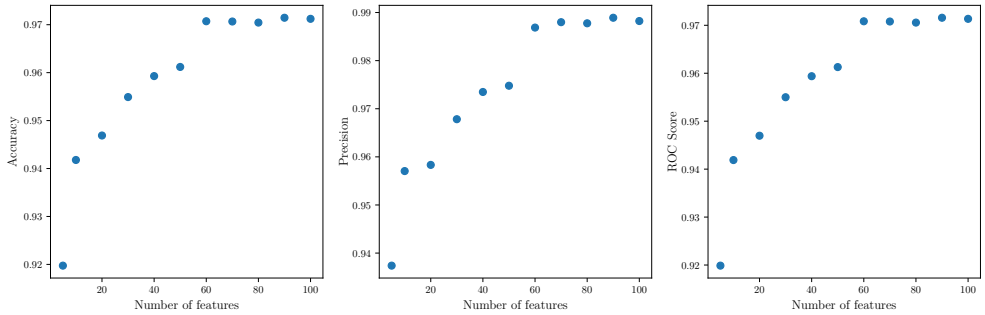

Figure 5: Quality parameters of the Random Forest for a varying number of features.

The model stops improving when using more than 60 features, so this number is chosen for the Random Forest. As before, the performance of the Random Forest is determined via k-folding the training data with five splits, the results are

$$\begin{aligned}\text{Accuracy} &= 0.9729 \pm 0.0023\,,\\ \text{Precision} &= 0.9859 \pm 0.0014\,,\\ \text{Recall} &= 0.960 \pm 0.004\,,\\ \text{ROC Score} &= 0.9956 \pm 0.0006\,.\end{aligned}$$

The scores are highly improved compared with the Naive Bayes approach, the cross validation erros are again very low. The optimal decision threshold is determined in the same way as before, Figure 6 shows the f_β score and its maximum.

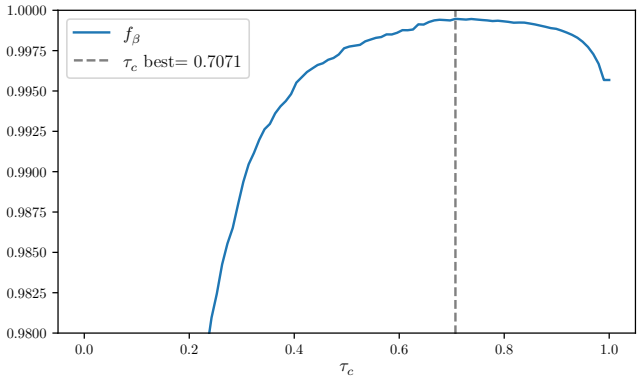

Figure 6: The f_β score of the Random Forest classifier and its maximum value.

The resulting best cut $\tau_{c,\,\text{best}}$ is also given in Figure 6. The resulting confusion matrix when using this threshold is given in Figure 7

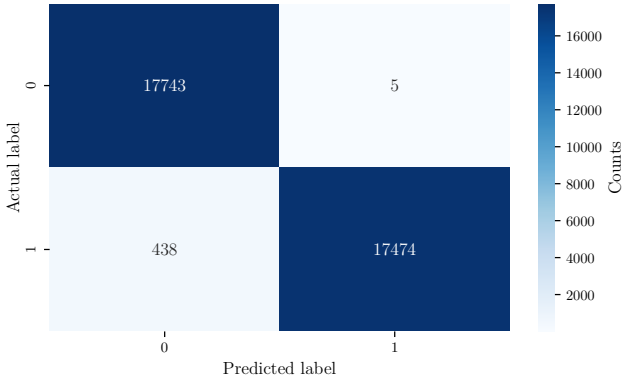

Figure 7: Confusion matrix for the Random Forest classifier.

The results are far better compared with the Naive Bayes approach, almost none of the ≈ 35.000 events have been incorrectly labeled as signal.

5.3.3 Neural Network

The third model used is a Neural Network in form of a multi-Layer Perceptron classifier. The Network consists of two hidden layers with sizes 60 and 10, respectively. The best number of features is again chosen according to the results shown in Figure 8.

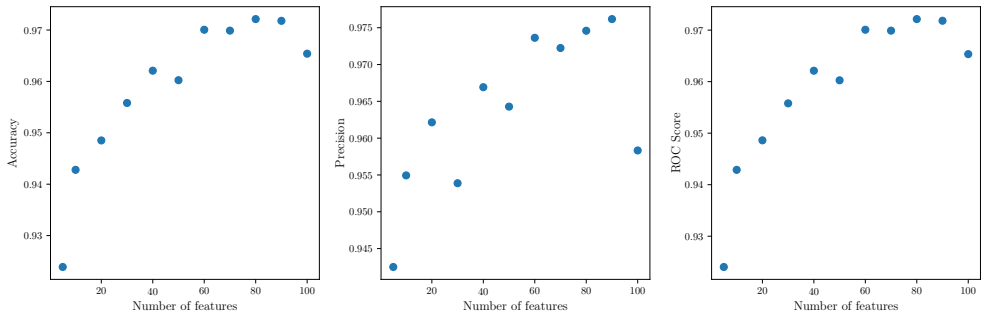

Figure 8: Quality parameters of the Neural Network for a varying number of features.

As with the Random Forest, no improvement can be seen after using more than 60 features, so again 60 is chosen as the number of features here. The results from using k-folding become

$$\begin{aligned} \text{Accuracy} &= 0.9711 \pm 0.0008\,, \\ \text{Precision} &= 0.971 \pm 0.005\,, \\ \text{Recall} &= 0.972 \pm 0.005\,, \\ \text{ROC Score} &= 0.9957 \pm 0.0006\,. \end{aligned}$$

The scores are very similar to the results of the Random Forest. Figure 9 shows the f_β score and its maximum.

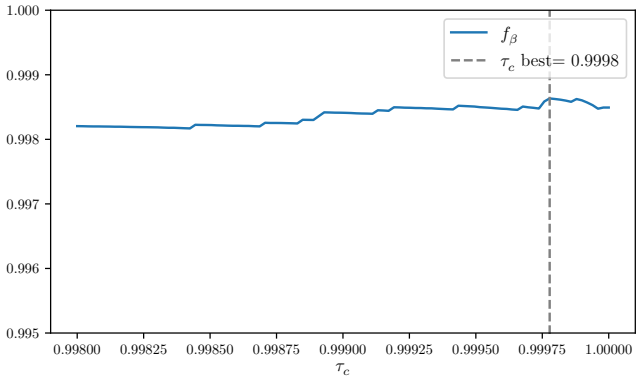

Figure 9: The f_β score of the Neural Network and its maximum value.

Since the Neural Network tends to give predictions either close to zero or one, a very high cut has to be chosen, the $\tau_{c,\,\mathrm{best}}$ value is given in Figure 9. After applying this cut, the confusion matrix shown in Figure 10 results.

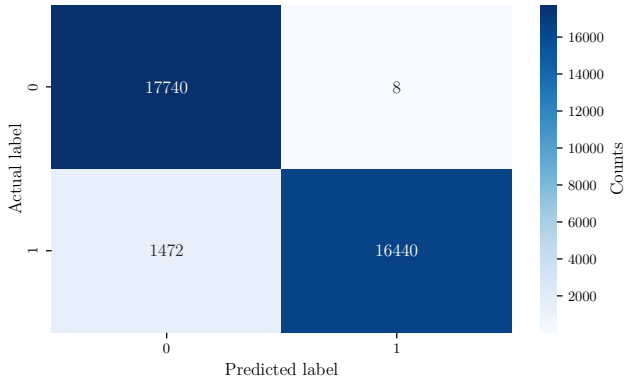

Figure 10: Confusion matrix for the Neural Network.

Just as with the Random Forest, almost all signal is labeled correctly. However, a slighlty larger portion of signal events is labeled as background.

5.4 Classification of test data

To make a prediction on the provided test data, one of the three models has to be chosen. The Naive Bayes classifier is ruled out due to its low performance. When comparing the Random Forest with the Neural Network, it becomes apparent that the Random Forest performes slightly better based on the confusion matrices. A comparison of the ROC curves of each classifier is shown in Figure 11.

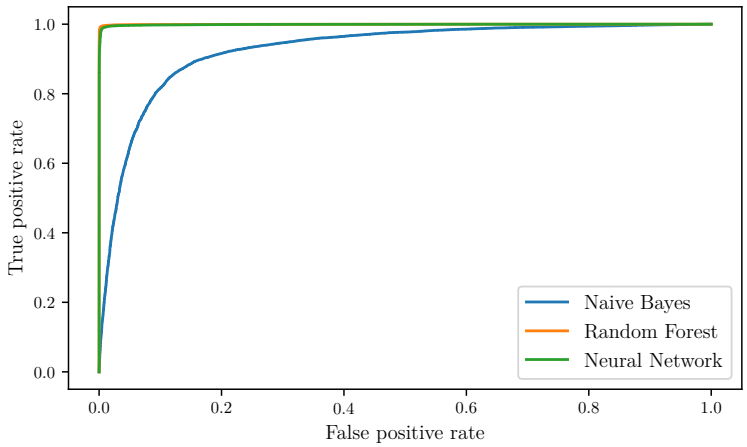

Figure 11: Comparison of the ROC curves of the three different classifiers.

While the Random Forest and Neural Network dont show a large difference here, the lower performance of the Naive Bayes approach is clearly visible. Consequently, the Random Forest is applied to the test sample, in which it identifies approximately half of the events as signal.

6 Discussion

The quality of the feature selection with the mRMR method is difficult to evaluate on its own, however the distributions in Figure 1 show a good seperation between signal and background. This is a good indication that the algorithm has succesfully selected features that contain the highest possible information about wether an event contains a signal component or only background.
As already discussed in the analysis of the Naive Bayes classifier, this model does not perform good enough, based on its limited capacity and low complexity. Requiring a high precision with the f_β score leads to a very large amount of signal events beeing mistaken for background.
In contrast, the Random Forest performs much better. The determination of the optimal amount of input features is stable and shows a clear maximum number of attributes. This helps to ensure that the model is not overtrained and can make use of all the information provided by the 60 different input features. The f_β score gives a clear maximum to optimize the cut on the predicted probabilities.
Compared to the Random Forest, the Neural Network performs slightly worse. This already becomes apparent when determining the best input features, as more fluctuations are already visible here. The optimal cut from the f_β score lies in a region very close to one, making it more sensitive to random changes in the network every time it is trained. Changing the architecture of the network could help improving its performance, however adding more hidden layers or neurons greatly increases the computational time. Since

the training process of the Neural Network already takes much longer than the Random Forest, it follows that the Random Forest model works better for this application, which is why it is chosen in the end.

References

[1] Samuele Mazzanto. *Minimum-Redundancy-Maximum-Relevance algorithm for feature selection*. Version 0.2.8. URL: `https://pypi.org/project/mrmr-selection/`.

[2] Jorge Nocedal. *On the limited memory BFGS method for large scale optimization*. DOI: `10.1007/BF01589116`.

[3] F. Pedregosa et al. *Scikit-learn: Machine Learning in Python*. URL: `https://scikit-learn.org/stable/about.html`.