Friedrich-Alexander-Universität
Naturwissenschaftliche Fakultät

Master's Thesis
in Physics

A Hybrid Approach for Optimizing Muon Background Simulations of the IceCube Neutrino Observatory

Simon Koch

Supervisor: Prof. Dr. Claudio Kopper

Erlangen Centre for Astroparticle Physics

Submission date: 18 November 2025

Abstract:

The IceCube Neutrino Observatory detects high-energy cosmic neutrinos by observing Cherenkov radiation emitted from secondary particles, such as muons, that are produced in neutrino interactions. A key challenge in detecting cosmic neutrinos is the large background of cosmic-ray-induced muons from the atmosphere, which has to be reduced by several orders of magnitude.

Thus, a large sample of background events has to be simulated in order to accurately estimate the background reduction efficiency. The computationally most expensive part of the IceCube simulation chain is the propagation of Cherenkov photons.

In this work, a hybrid simulation approach is presented that combines traditional simulation methods with a machine learning model in the form of a boosted decision tree classifier. Based on the energy loss information of each event of a given cosmic-ray-induced muon sample, the classifier predicts the probability for each muon to remain in the sample after the background reduction process. Events with a low chance of remaining in the sample are excluded from photon propagation and further simulation steps. This approach ensures that computational resources are better spent on statistically rare events, which have a high chance of surviving the background reduction process. Depending on the initial energy range of the muon sample, a computational gain factor of up to 2.2 ± 0.4 was obtained. As a consequence, more background events can be simulated within the same amount of time. Therefore, employing such a classifier in future simulations might reduce the statistical uncertainty of the estimated background reduction efficiency, without increasing the budget spent on simulation.

Table of Contents

1. Introduction

The identification of ultra-high-energy cosmic ray sources and understanding the acceleration processes of these particles are fundamental tasks in the field of astroparticle physics. However, locating potential sources based on measurements of cosmic rays is difficult. Since cosmic rays are charged particles, they get deflected by magnetic fields along their way to Earth. Fortunately, there are other messenger particles like neutrinos, which are produced by hadronic interactions of cosmic rays with the medium at respective acceleration sites. Due to their physical properties, neutrinos are unlikely to interact with matter or fields along their way to Earth and point directly back to their sources. Therefore, the detection of cosmic neutrinos is of great interest in astroparticle physics, as it provides valuable information on distant astrophysical particle sources [20].

Detecting neutrinos on Earth involves several challenges. Due to the low interaction probability of neutrinos with matter, huge detectors have to be built to detect cosmic neutrinos in statistically significant numbers. An example of such a detector is the IceCube Neutrino Observatory located at the South Pole. It has a volume of about $\sim 1\,\mathrm{km}^3$ deep within the ice, instrumented with 5160 digital optical modules (DOMs). These modules detect Cherenkov light emitted by secondary charged particles like muons, produced by neutrino interactions with nuclei in the ice. A challenge in detecting such muons is the large background of cosmic-ray-induced muons from the atmosphere, which has to be reduced by several orders of magnitude. During data analysis, this is done by applying event filters to obtain neutrino event samples with a reduced signal-to-noise ratio. One of those is the medium-energy starting event (MESE) sample [21],[29].

The related background reduction efficiency and its uncertainty can be estimated based on extensive simulations of the muon background of IceCube. In the current state of the IceCube simulation, each background muon has to be fully simulated from its generation up to the signal processing and application of event filters. This includes the simulation of the Cherenkov photons induced by stochastic energy losses of the muon, which is one of the computationally most expensive steps of the IceCube simulation [10],[18],[32]. However, a large number of muons is removed by the event filters in the end. Therefore, a significant amount of computational resources is spent on photon propagation unnecessarily.

This thesis aims to present a possible solution to this efficiency problem by introducing a hybrid simulation approach that combines traditional simulation methods with a machine learning model in the form of a boosted decision tree (BDT) classifier. Based on the energy loss information of each muon event of a given simulation sample, the classifier predicts the probability for each muon to remain in the sample after applying the MESE event filter. Events with a low chance of remaining in the final sample are excluded from photon propagation and further simulation steps. With that, computational resources can be saved. The results presented in this work show that, depending on the initial energy range of the muon sample, a computational gain factor of up to 2.2 ± 0.4 can be obtained using the hybrid simulation approach. Therefore, about twice as many background events can be simulated within the same amount of time. Consequently, the statistical uncertainty of the estimated background reduction efficiency might be reduced by this simulation approach, without increasing the budget spent on simulation.

In section 2, information on several topics related to this work is provided. A proof of concept of the hybrid simulation approach can be found in section 3, along with the results regarding the training and optimization of the BDT classifier. In section 4, a testing framework for optimized muon simulations as described above is introduced, which provides full access to the IceCube simulation parameters and was used to determine the computational gain [19].

2. Theoretical Background

This section provides information on different topics that are related to this thesis. Details on neutrinos and how to detect them can be found in section 2.1, which is mainly based on [20],[21]. Further information on the IceCube simulation is given in section 2.2, primarily referring to [32]. In section 2.3, a brief overview of machine learning, boosted decision trees, and tools to evaluate the performance of classification models is provided, mostly based on [35],[42],[43].

2.1. Neutrino Astronomy

In the field of astroparticle physics, different messenger particles are detected on Earth to gather information on astrophysical objects. These particles include cosmic rays, which are ionized nuclei, predominantly composed of protons ($\sim 90\,\%$), alpha particles ($\sim 9\,\%$), and heavier nuclei. Cosmic rays are accelerated to relativistic energies of up to $10^{20}\,\mathrm{eV}$ (ultra-high-energy cosmic rays, UHECRs). The fraction of UHECRs is quite small, but understanding how they reach such extreme energies is particularly interesting and a fundamental question in cosmic ray physics [20].

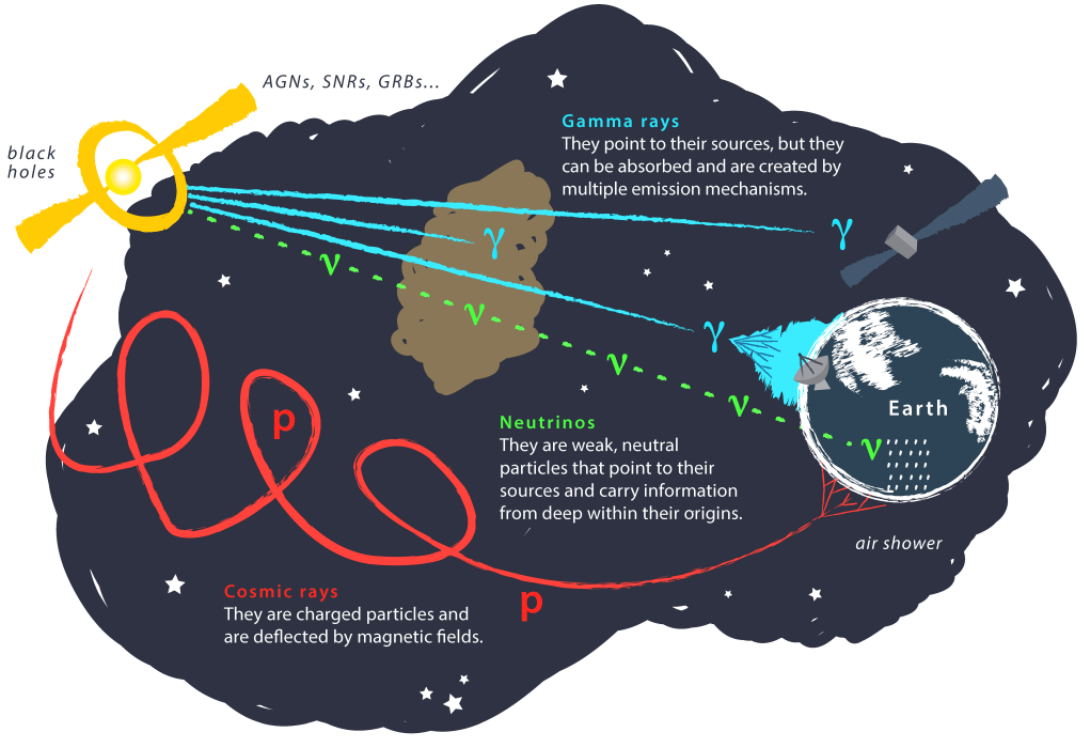

Figure 1: Schematic path of different particles from their astrophysical source to Earth. This image was adapted from [22].

Potential sources of cosmic rays are, among others, supernova remnants (SNR), active galactic nuclei (AGNs), and gamma-ray bursts (GRBs). Still, the origin and acceleration mechanisms of cosmic rays are not yet fully understood. Since cosmic rays are charged particles, they are deflected by magnetic fields along their path to Earth. This makes the direct identification of their sources quite difficult. However, cosmic rays produce secondary particles when they interact with the medium at their acceleration sites, including highly energetic photons (γ-rays) and neutrinos. Both are neutral particles travelling along straight paths from their sources to Earth (see Figure 1). In contrast to γ-rays, neutrinos are very unlikely to be absorbed along their way to Earth due to their physical properties. This makes them particularly interesting for studying distant astrophysical particle sources. Furthermore, they can only be produced through hadronic processes. Therefore, observing them provides independent information on the contribution of hadronic and electromagnetic processes to the γ-ray flux from the galactic plane [20]. Further details on neutrinos and their detection on Earth are presented below.

2.1.1. Neutrinos

Neutrinos are elementary particles of the Standard Model of particle physics. Having a spin of 1/2, they belong to the particle class of fermions. There are three different neutrino flavors, resulting in the electron neutrino ν_e, the muon neutrino ν_μ, and the tau neutrino ν_τ. The corresponding charged leptons are the electron e^-, the muon μ^-, and the tauon τ^-. Together, they form the three lepton pairs of the Standard Model. Neutrinos can couple to the charged $W^\pm$ and the neutral Z^0 gauge bosons, resulting in charged and neutral current interaction vertices (see Figure 2) [20].

Figure 2: Examples of interaction vertices for neutrinos of flavour l. **Left:** Charged current interaction. **Right:** Neutral current interaction.

Since neutrinos are leptons with no electrical charge, they only interact via the weak interaction. Due to their low cross section, their interaction probability with matter is quite small. Therefore, they can travel from their astrophysical source to Earth without being deflected or absorbed. On the other hand, this makes detecting neutrinos quite challenging (see section 2.1.2) [20].

The standard production scenario of high-energy neutrinos at astrophysical sources like AGNs is the decay of charged pions. When UHECRs interact with the background radiation at the acceleration site, neutrons n and positively charged pions π^+ are produced [20]:

$$p + \gamma \longrightarrow n + \pi^+ \tag{2.1}$$

The pions decay into positively charged muons μ^+ and muon neutrinos ν_μ. Since muons are also unstable, they further decay into positrons e^+, electron neutrinos ν_e, and muon antineutrinos $\bar{\nu}_\mu$ [20]:

$$n + \pi^+ \longrightarrow n + \mu^+ + \nu_\mu \longrightarrow n + e^+ + \nu_e + \bar{\nu}_\mu + \nu_\mu \tag{2.2}$$

This results in a neutrino flux composition of about $\phi^0(\nu_e) : \phi^0(\nu_\mu) : \phi^0(\nu_\tau) = 1 : 2 : 0$ at the source [20].

In the Standard Model, neutrinos are assumed to be massless. However, it was observed that neutrinos can change their flavor while they propagate. This effect is called neutrino oscillation and can be described by introducing a nonzero neutrino rest mass [20]. Recent measurements by the KATRIN experiment put an upper limit of $m_\nu < 0.45\,\text{eV}$ (90 % CL) to the neutrino mass [23]. Due to neutrino oscillations, the neutrino flux composition at Earth, originating from the standard case of charged pion decay, is approximately $\phi^0(\nu_e) : \phi^0(\nu_\mu) : \phi^0(\nu_\tau) \approx 1 : 1 : 1$. The flux composition is expected to differ from the generic $1 : 1 : 1$ case, depending on the production mechanism at the source (see [20] for more details). Therefore, being able to differentiate neutrinos by their flavor in detectors is important to obtain information in this regard.

2.1.2. IceCube Neutrino Observatory

Due to the weak interaction of neutrinos with matter, huge detectors have to be built to detect statistically significant numbers of cosmic neutrinos [21]. One example of such a detector is the IceCube Neutrino Observatory located at the South Pole, which has a detector volume of about $\sim 1\,\text{km}^3$ (see Figure 3). IceCube consists of 86 cables called strings, that were deployed deep into the ice. Each of the strings is equipped with 60 Digital Optical Modules (DOMs), evenly distributed along the strings at a depth of 1450 m – 2450 m [20],[21]. Each DOM consists of a glass sphere containing a 10-inch photomultiplier tube (PMT) and some electronics for signal processing [21]. PMTs are light detection devices that convert incident photons into an analogue voltage signal based on the photoelectric effect [39].

The detection principle of IceCube relies on the measurement of Cherenkov radiation. This kind of light is emitted if charged particles propagate through a medium faster than the speed of light in the medium [21]. The setup described above enables the indirect detection of neutrinos by measuring the Cherenkov light emitted from secondary charged particles produced in neutrino interactions with the ice. Depending on the neutrino flavor and the particle interaction, different light patterns can be observed in the detector [21]. High-energy neutrinos typically interact via deep inelastic scattering with nucleons of the ice. In more detail, the incoming neutrino interacts with quarks of a target nucleus via the exchange of a $W^\pm$ or Z^0 boson. In both cases, the hit nucleus is split into fragments, typically resulting in a hadronic particle shower. Secondary charged particles of such a shower emit Cherenkov light in a spherical pattern referred to as "cascade" (see left plot of Figure 4) [21]. As illustrated in Figure 2, a charged lepton is produced in charged current interactions, carrying about 80 % of the initial neutrino energy [20]. In the case of a produced electron, the hadronic cascade of nuclear fragments is superimposed by the electromagnetic cascade induced by the rapid scattering of the electron in ice [21].

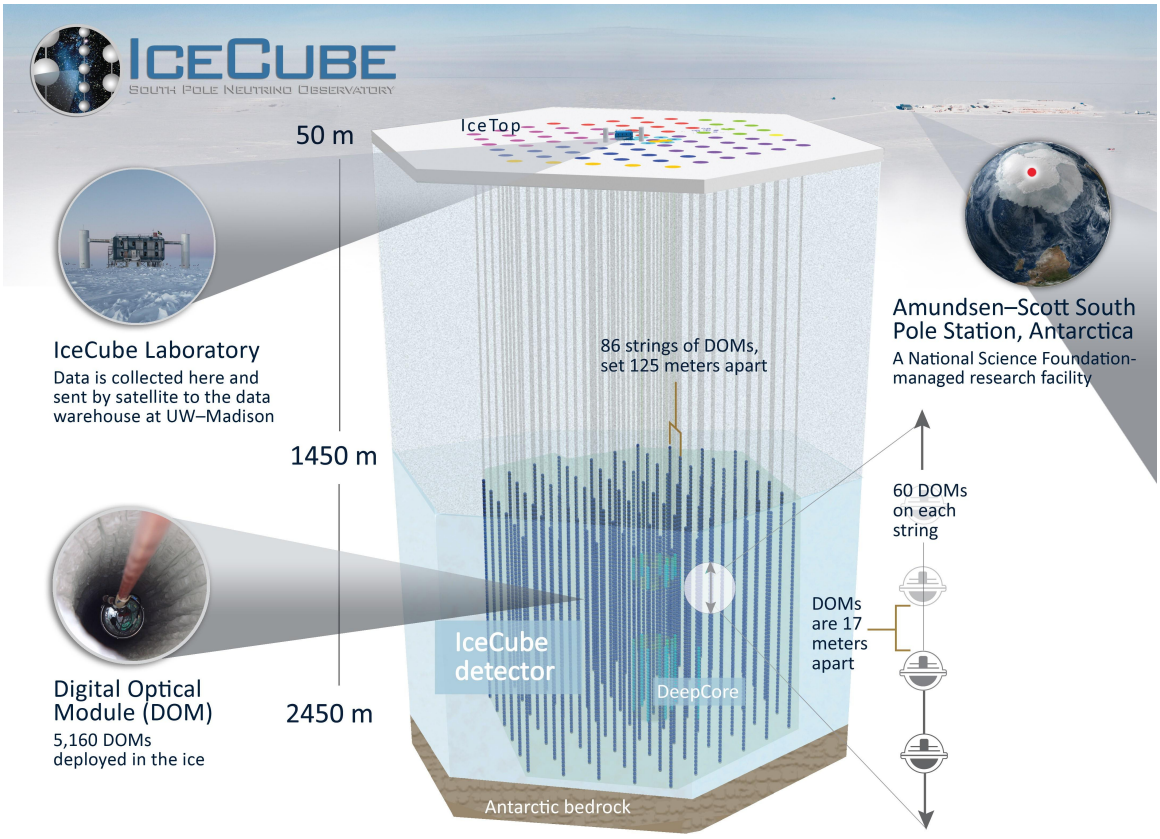

Figure 3: Schematic view of the IceCube Neutrino Observatory. A detector volume of ~1 km^3 was instrumented with 5160 Digital Optical Modules (DOMs). A total of 86 strings with 60 DOMs per string was deployed into the ice at a depth of 1450 m – 2450 m. This image was adapted from [24].

A similar light pattern is observed for the charged current interaction of a ν_τ. Since the resulting τ decays quite fast, this also leads to a particle cascade [21]. Depending on the energy of the τ, the position of the ν_τ and the τ decay can be spatially distinct, showing a "double bang signature" of two separate cascades [20]. A very different kind of light pattern is observed for the charged current interaction of a ν_μ, resulting in the production of a μ. The muon is a long-lived particle that can travel relatively long distances through the detector before it decays [21]. The resulting light pattern is referred to as "track" (see right plot of Figure 4). Muons lose their energy by ionization, bremsstrahlung, pair production, and photo-nuclear interactions [21]. Above ~ 1 TeV, the Cherenkov light emission of muons in ice is dominated by charged particles produced in stochastic energy losses along the track, including bremsstrahlung, pair production, and photo-nuclear interactions [36]. These stochastic energy losses play an important role in this thesis, as explained in the ongoing sections.

Generally, the neutrino energy can be estimated from the detected amount of Cherenkov photons. This estimate is more accurate for cascades. There, the measured number of photons is proportional to the neutrino energy transferred to the cascade, which can be completely contained in the detector volume [21].

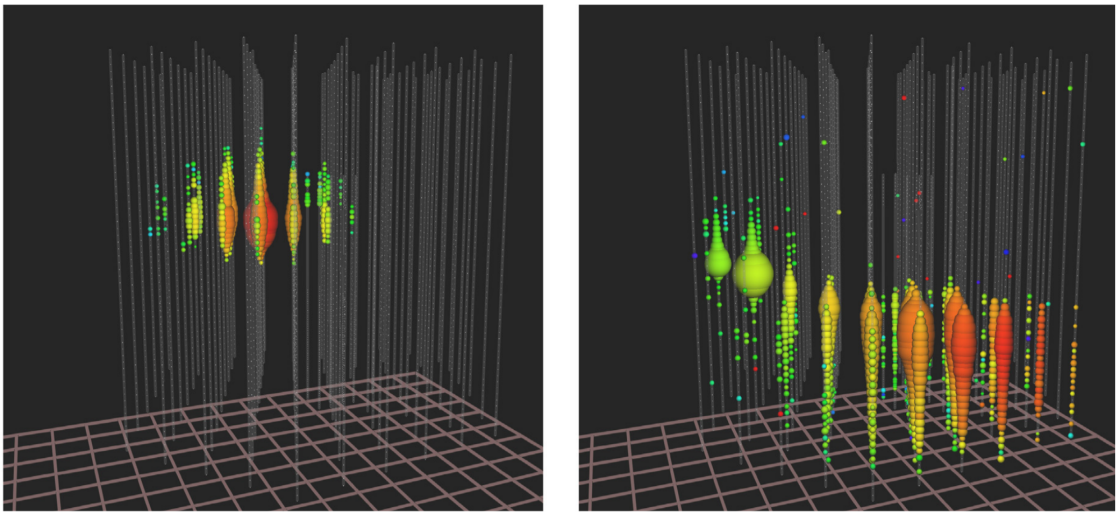

Figure 4: Signals of neutrino events as observed by the IceCube detector. White dots correspond to DOMs without a signal. The colour encodes the arrival time of the light signal (red - early, purple - late), while the size of the coloured DOMs indicates the number of detected photons. **Left:** Cascade-like event as observed for interactions of ν_e and ν_τ. **Left:** Track-like event as observed for interactions of ν_μ. This image was adapted from [21].

For tracks, this is not necessarily the case, which makes the energy reconstruction of ν_μ less precise. On the other hand, the direction of the initial neutrino can be reconstructed more accurately for ν_μ, due to the related light pattern [21].

Since its completion of construction in 2011, the IceCube Neutrino Observatory has made several discoveries. Some of the most prominent ones are listed below. In 2013 of a flux of cosmic neutrinos with energies in the range of 30 TeV – 1.2 PeV was observed [25]. Furthermore, two AGNs were identified as point sources of astrophysical neutrinos: TXS 0506+056 in 2018 and NGC 1068 in 2022 [26],[27]. Another recent achievement was the discovery of a diffuse neutrino flux from the galactic plane in 2023 [28].

2.1.3. Background Events of the IceCube Neutrino Observatory

The IceCube Neutrino Observatory naturally has a large background of neutrinos and muons produced in extensive air showers that are induced by cosmic rays hitting the Earth's atmosphere (see Figure 5). Track-like events as generated by muons induced by cosmic neutrinos, also occur for atmospheric muons and muons originating from interactions of atmospheric neutrinos with the ice [21]. The recorded event rate of IceCube is about ~ 3 kHz, which is dominated by atmospheric muons [37]. Only one event in a million is a neutrino, resulting in a neutrino event rate of about ~ 10^5/year [37]. A majority of these are atmospheric neutrinos, resulting in an actual signal of a few hundred astrophysical neutrinos per year [37]. From data analysis, only about 10 neutrinos per year are identified as astrophysical neutrinos with high confidence [37]. This shows the importance of an efficient background reduction for IceCube, which can be achieved in different ways. One possible approach is to use the Earth as a natural shield against atmospheric muons. This limits the observation of cosmic neutrinos to up-going events from the Northern sky. The remaining background of atmospheric neutrinos can be distinguished from the signal based on the measured energy spectrum, since the atmospheric neutrino flux is expected to be relatively small above ~ 300 TeV [21].

For all sky observations, a different method can be used to distinguish cosmic neutrinos from the background. In this method, one tries to identify high-energy neutrinos that interacted inside the detector volume. These events are called high-energy starting events (HESE) [21].

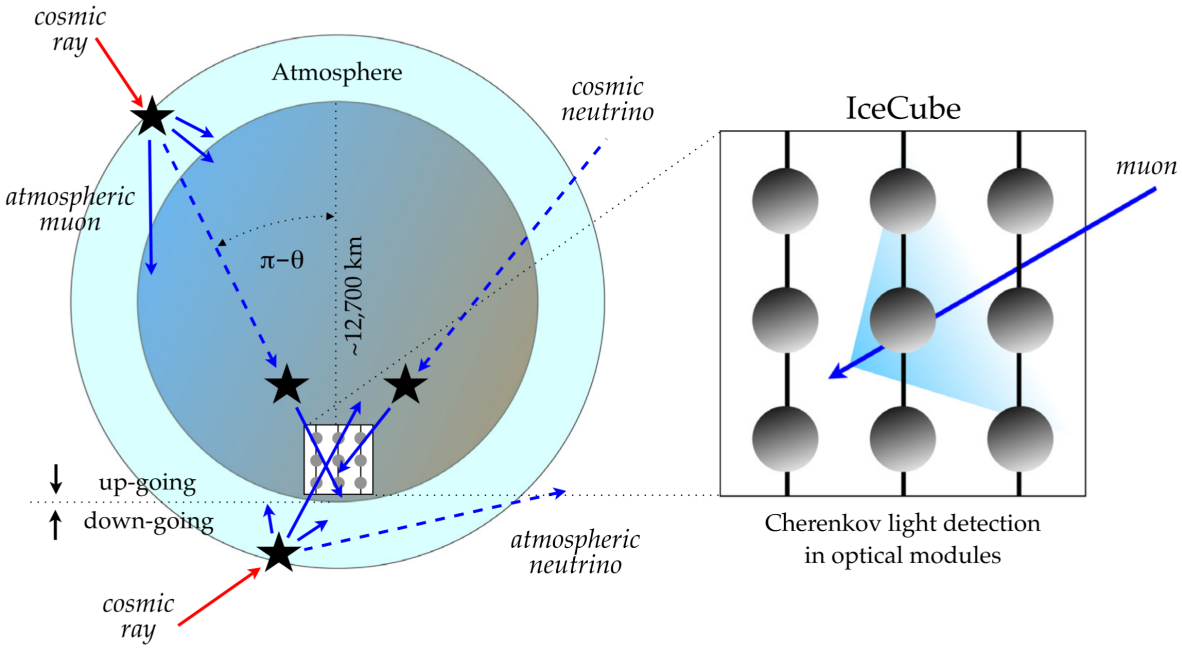

Figure 5: Illustration of the background of the IceCube Neutrino Observatory. Blue dashed arrows indicate neutrinos, which are either produced in the atmosphere (background) or by astrophysical sources (signal). Solid blue lines denote muons. The atmospheric muon background is suppressed for up-going events since the Earth acts as a shield. This image was adapted from [21].

The HESE event sample is generated by introducing a veto region that is excluded from a fiducial volume within the detector (see Figure 6). If an event induces a signal in the veto region with a signal strength above a certain threshold, it is excluded from the HESE sample. By selecting only high-energy events with an initial interaction vertex observed inside the fiducial volume, the atmospheric muon background can be suppressed for the whole sky, since an atmospheric muon would also induce a significant signal in the veto region [21],[30]. Furthermore, the background of atmospheric neutrinos is reduced for the southern sky, since these events are generally accompanied by atmospheric muons from the same air shower (self-veto effect) [29],[46]. Further details on the HESE sample can be found in [30].

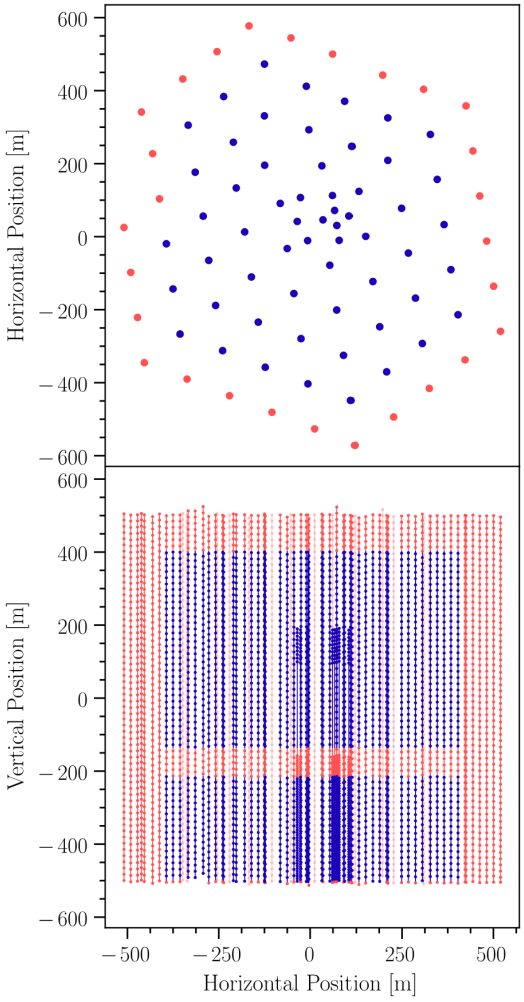

Figure 6: Schematic view of the IceCube veto region used to define the HESE event sample. **Top:** Top view of the IceCube detector array, indicating the strings completely belonging to the veto region (red) and the strings including at least one non-veto DOM (blue). **Bottom:** Corresponding side view, indicating the DOMs of the veto region (red), and the DOMs of the fiducial volume (blue). This image was adapted from [30].

An extension of the HESE sample to lower energies is the sample of medium-energy starting events (MESE). Compared to the HESE sample, further event selections based on consecutive selection levels are applied to generate the MESE sample. This also includes an event classification into cascades and tracks using a neural network. Overall, this leads to an increase in the event statistics towards lower energies ($\sim 1\,\mathrm{TeV}$). The rate of atmospheric muons at the final event selection level of the MESE sample is reduced by ten orders of magnitude compared to the initial sample, improving the signal-to-noise ratio [29],[31].[1] Based on simulations, one can determine the ratio of background events that are left in the MESE sample and background events that were excluded. The resulting background reduction efficiency and its associated uncertainty are crucial parameters to quantify how effectively background events are filtered out during data processing. A large number of background events have to be simulated to accurately estimate these quantities. More details on the IceCube simulation are provided in the following section.

2.2. IceCube Simulation

Neutrino detection and background processes in IceCube can be modeled using Monte Carlo simulations, which are essential for interpreting the measured signal of the IceCube detector and event reconstruction. As illustrated in Figure 7, the IceCube simulation can be understood as a consecutive execution (simulation chain) of different simulation steps. This includes all processes from the simulation of different particles arriving on Earth, including whole cosmic ray induced air showers, up to the simulation of the detector electronics. Furthermore, all levels of data processing used in the analysis of real data can be applied equivalently to the simulated data [32].

The code of the IceCube simulation is implemented within the simulation framework IceTray, which is mainly based on C++ and Python [17]. IceTray provides a modular structure, enabling the construction of large simulation chains by linking consecutive simulation steps through the input and output files of their corresponding scripts. In this thesis, this modular system was used to define a simulation chain specifically designed for testing optimized muon simulations (see section 4.1). The corresponding simulation steps are described below.

MuonGun:

MuonGun is the IceCube implementation of the simulation tool MUPAGE [33]. As one of the available software modules for particle generation (see red box in Figure 7), it can be used to generate and parametrize a flux of muons under the ice [32]. More specifically, muons are injected on a cylinder surface surrounding the IceCube detector array. The spectrum of the muon energy E_μ is modeled by a power law $\propto (E_\mu + E_{off})^{-\gamma}$ (see *MuonGun/private/MuonGun/EnergyDistribution.cxx* in [17]) where E_{off} is an adjustable energy offset and γ the spectral index. The corresponding energy range and value of γ defining the muon flux can be freely adjusted along with other parameters regarding the detector geometry and the surrounding ice [17].

[1] Applying the selection cuts to obtain the MESE sample is referred to as applying the MESE event filter in this thesis.

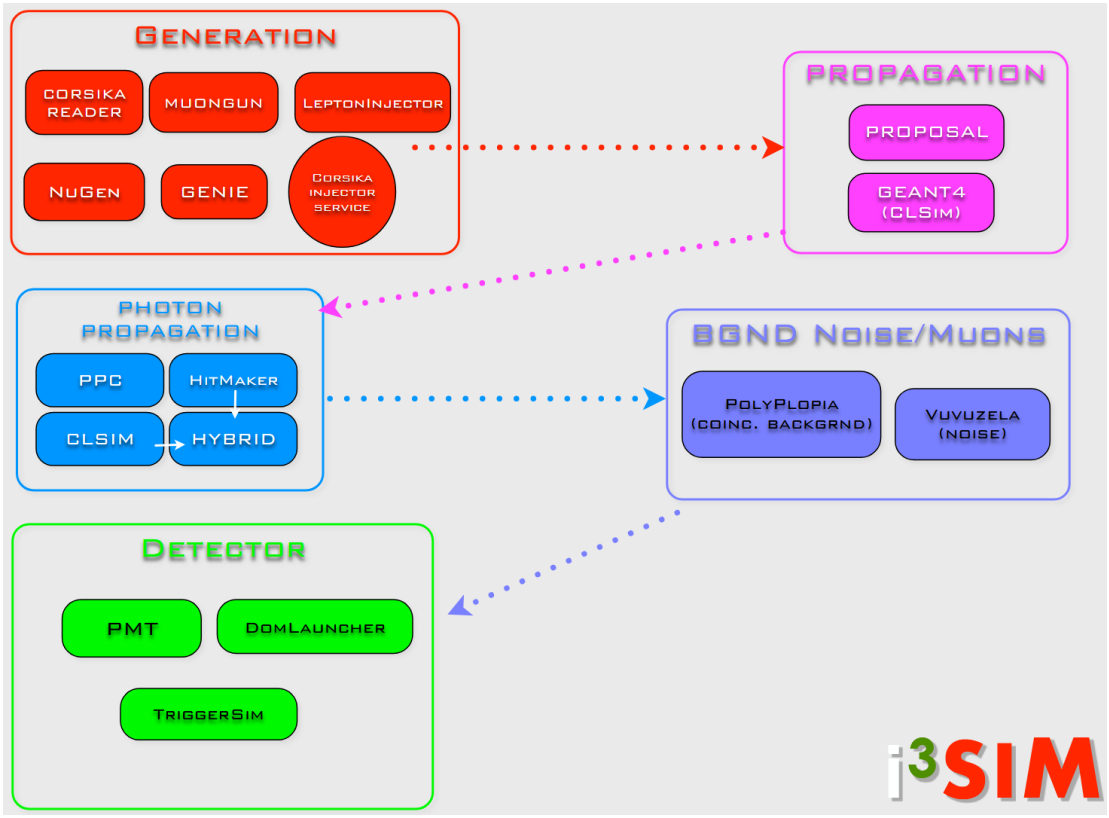

Figure 7: Scheme of different parts of the IceCube simulation chain. Generation, propagation, and interaction of different particles, as well as the detector simulation, are performed based on different simulation steps. The ones relevant in the context of this thesis are explained in the text. This image was adapted from [32].

CORSIKA:

Another software module for particle generation used in the IceCube simulation is CORSIKA, which is a program originally developed at the Karlsruhe Institute of Technology (KIT) [34]. It is used for simulations of cosmic ray induced air showers. This involves the propagation and decay of various particles and their hadronic and electromagnetic interactions with nuclei of the atmosphere [32].

Polyplopia:

Polyplopia is a module of the IceCube simulation that can be used to create coincident events, which refers to events that are detected within a certain time window Δt. In the context of this thesis, it is used together with MuonGun and CORSIKA. Primary muon events generated by MuonGun are combined with coincident background events from the CORSIKA air showers by sampling from a Poisson distribution with respect to the time window Δt [32].

Muon Propagation (PROPOSAL):

The in ice propagation of charged leptons generated by MuonGun and CORSIKA is based on the PROPOSAL simulation tool [5]. This includes the simulation of stochastic energy losses, as described in section 2.1.2 in the context of muon tracks in ice. Furthermore, particle decays are simulated [32].

Photon Propagation (CLSim):

The in ice propagation of Cherenkov photons, induced by charged particles generated in previous simulation steps, is simulated with a tool called CLSim. In this thesis, CLSim was used in combination with the SnowStorm module.[2] SnowStrom can be used to continuously perturb the ice model used for photon propagation. This is useful for studies of the systematic uncertainties of IceCube. In contrast to the aforementioned simulation steps, photon propagation is run almost exclusively on GPUs [32]. It is one of the computationally most expensive parts of the IceCube simulation (see average computation time of photon propagation for muon simulation datasets used in the context of this thesis [10],[18]).

Detector Simulation:

The modeling of the PMT response and DOM electronics is based on the DOMLauncher module of the IceCube simulation. The overall detector simulation further includes the generation of random noise induced thermally and by radioactive decays in the DOM glass, using the software tool Vuvuzela. Additionally, different detector triggers are simulated to accurately model the data acquisition [32].

Filter Level 1 and 2:

Data processing steps used in the analysis of real data can be applied equivalently to the simulated data, after simulating the detector response. In this work, filter level 1 and 2 were used, which are filter stages that process the simulated data, based on the corresponding IceTray scripts listed in Table 20. This includes the event selections applied for the MESE sample mentioned in section 2.1.3 [17],[32].

The simulation steps described above represent only a part of the simulation capabilities available in the IceTray software framework. This demonstrates the versatility of the IceCube simulation. On the other hand, extensive simulations for estimating quantities like the background reduction efficiency require a lot of computational resources. According to the computing report 2025, IceCube spent about $\sim 2.4\,\text{Mhours}$ on total GPU computing during the previous year [40]. Assuming a reasonable cost of $\sim 0.30\,\text{cts/GPU-hour}$ yields an estimated budget of $\sim 7.2 \cdot 10^5\,\$/\text{year}$ dedicated to GPU computation. As mentioned previously, photon propagation is one of the computationally most expensive simulation steps and relies mostly on GPU computing. As a consequence, it contributes significantly to this cost. Therefore, reducing the computational resources required for

[2] The photon propagation of the testing framework introduced in section 4.1 is based on SnowStorm. However, the systematic perturbations were defined by a delta distribution. Therefore, the functionality of SnowStorm was not used in the context of this thesis effectively.

muon background simulations with respect to photon propagation is a main motivation of this thesis.

2.3. Machine Learning

In general, machine learning is a scientific discipline focused on developing algorithms that enable computers to evolve certain behaviours based on (training) data [35]. A resulting machine learning model can be understood as a mathematical mapping from an input space X to an output space Y, learned from the data [38]. A subcategory of machine learning is supervised learning. There, the data consists of pairs of elements $(x \in X, y \in Y)$, and the goal is to develop (train) a predictive model $y = F(x)$. This is achieved by minimizing the expected value of a loss function $L(y, F(x))$, which is a measure of the disagreement between the true and predicted values of the elements in Y [35]. Two major tasks of supervised learning are regression and classification. A model that makes predictions on an output space consisting of categorical values is referred to as a classifier. If the output space consists of continuous values, one speaks of a regressor [38]. The underlying model can be based on a variety of algorithms. In this thesis, boosted decision tree (BDT) models were trained for classification and regression tasks. The corresponding algorithm is referred to as gradient boosting, which is explained in section 2.3.1 below.

2.3.1. Gradient Boosted Decision Trees

The BDT models trained in this thesis are based on the software of *scikit-learn* (v.1.5.2) [1],[8],[9]. According to the references of section 1.11.1.2. in the *scikit-learn* user guide [41], the fundamental gradient boosted decision tree algorithms implemented by *scikit-learn* are closely related to the work of J.H. Friedman. Therefore, this section aims to convey a general understanding of the gradient boosting algorithm based on [42],[43]. Before going into detail on that, the concept of decision trees is introduced.

Decision Trees:

A decision tree is an example of a supervised learning model. It is based on the partitioning of the input space X into disjoint regions by building a tree-like structure using consecutive binary decisions. The data regions resulting from the binary decisions are referred to as nodes. An example of a single binary decision is splitting the elements x of a one-dimensional input space X into two regions $L \subset X$ and $R \subset X$, which are below and above a reference value $x_s \in X$ [35]:

$$x < x_s \Rightarrow x \in L \tag{2.3}$$

$$x \geq x_s \Rightarrow x \in R \tag{2.4}$$

Starting with a root node that includes all elements of the input data sample (training data), the decision tree is grown up to a certain level of consecutive binary decisions referred to as tree depth. The final nodes of the tree are called leaf nodes or leaves and define the final partitions of the input space [35].

A tree is grown until a certain stopping criterion is reached. Examples for stopping criteria of splitting individual nodes are reaching a maximum tree depth or reaching the minimum number of samples included in a node. Further details on stopping rules can be found in [35]. The optimal splitting of a node is learned by the model, based on an impurity function. An example of such a function is the Gini diversity index [35]:

$$\phi(p,q) = 1 - p^2 - q^2 \tag{2.5}$$

Assuming a classification problem as an example, where each element of the input data corresponds to one of the two classes A or B, p and q can be interpreted as posterior probabilities $P(A|t)$ and $P(B|t)$ of a sample to belong to class A or B with respect to node t. The probability of a sample to belong to the node t is given by $P(t)$ [35]:

$$P(A|t) = N_A/N_t \tag{2.6}$$
$$P(B|t) = N_B/N_t \tag{2.7}$$
$$P(t) = N_t/N \tag{2.8}$$

Here N is the total number of training samples, N_t is the number of samples included in node t, and N_A and N_B are the respective numbers of samples of the node belonging to class A or B. Based on these definitions, a minimal impurity is reached if a node only contains samples belonging to a single class. Referring to the example from above, the optimal splitting of a node can be found by maximizing the impurity gain ΔI over all possible values of x_s and all elements of the training sample. The impurity gain with respect to the parent node t_0 can be defined by the following expression [35]:

$$\Delta I = I(t_0) - I(t_L) - I(t_R) \tag{2.9}$$

Here $I(t) = P(t)\phi(p,q|t)$ is the weighted node impurity. The overall partitioning of the input space is optimized by growing the whole tree with an algorithm based on this mechanism. The prediction y of a trained decision tree classifier for a given input x is the class with the highest posterior probability with respect to the leaf node into which the sample is sorted. Further details on decision trees can be found in [35].

Gradient Boosting:

The fundamental gradient boosting algorithm, as presented by J.H. Friedman can be explained by considering a general supervised learning problem of finding a function $F^*(\mathbf{x})$ that maps a set of input variables $\mathbf{x} = \{x_1, ..., x_n\}$, also referred to as input vector in this thesis, to an respective output (target) y. For a given training set $\{y_i, \mathbf{x}_i\}_{i=1}^N$ of known pairs $(y, \mathbf{x})$, this function is generally defined by the minimization of the expected value $E_{y,\mathbf{x}}$ of a loss function $L(y, F(\mathbf{x}))$ [42]:

$$F^*(\mathbf{x}) = \arg\min_{F(\mathbf{x})} E_{y,\mathbf{x}} L(y, F(\mathbf{x})) \tag{2.10}$$

The goal of the gradient boosting algorithm is to construct a larger regression model consisting of multiple simple functions $h(\mathbf{x};\mathbf{a})$ with parameters $\mathbf{a} = \{a_1, a_2, ...\}$, referred to as "weak"- or "base learners", which are added together in an iterative process [42],[43].

With that, the function $F^*(\mathbf{x})$ is approximated by an additive expansion after M (boosting) iterations [42]:

$$F(\mathbf{x}) = \sum_{m=0}^{M} \beta_m h(\mathbf{x}; \mathbf{a}_m) \tag{2.11}$$

Starting with an initial guess $F_0(\mathbf{x})$, the approximation of $F(\mathbf{x})$ is updated based on the following rule [42]:

$$F_m(\mathbf{x}) = F_{m-1}(\mathbf{x}) + \beta_m h(\mathbf{x}; \mathbf{a}_m) \tag{2.12}$$

In each iteration m the parameters $\mathbf{a}_m$ are determined by fitting $h(\mathbf{x}; \mathbf{a})$ by least squares to the "pseudo"-residuals $\tilde{y}_{im}$ based on the training set $\{y_i, \mathbf{x}_i\}_{i=1}^N$ and the parameter ρ [42]:

$$\mathbf{a}_m = \arg\min_{\mathbf{a},\rho} \sum_{i=1}^{N} (\tilde{y}_{im} - \rho h(\mathbf{x}_i; \mathbf{a}))^2 \tag{2.13}$$

$$\tilde{y}_{im} = -\left[\frac{\partial L(y_i, F(\mathbf{x}_i))}{\partial F(\mathbf{x}_i)}\right]_{F(\mathbf{x})=F_{m-1}(\mathbf{x})} \tag{2.14}$$

The "pseudo"-residuals are defined by the negative gradient of the loss function, which is why the algorithm is referred to as gradient boosting [42],[43]. The obtained weak learner $h(\mathbf{x}; \mathbf{a}_m)$ is then used to find the optimal value of the expansion coefficient β_m of the current boosting iteration m by minimizing the loss function [42]:

$$\beta_m = \arg\min_{\beta} \sum_{i=1}^{N} L(y_i, F_{m-1}(\mathbf{x}_i) + \beta h(\mathbf{x}_i; \mathbf{a}_m)) \tag{2.15}$$

The resulting algorithm is quite general. It can be combined with a variety of weak learners $h(\mathbf{x}; \mathbf{a})$ and arbitrary differential loss functions $L(y, F(\mathbf{x}))$ for classification or regression tasks [42],[43]. One example is given below to further illustrate the concept of gradient boosting.

Gradient Boosted Decision Tree Regressor:

One possible choice of a weak learner is a decision tree regressor, which partitions the input space into J disjoint regions $\{R_{jm}\}_{j=1}^J$ and predicts a separate constant value for each of them given by the mean of the corresponding "pseudo"-residuals $\bar{y}_{jm} = mean_{\mathbf{x}_i \in R_{jm}}(\tilde{y}_{im})$ [42]:

$$h(\mathbf{x}; \{R_{jm}\}_{j=1}^J) = \sum_{j=1}^{J} \bar{y}_{jm} \cdot 1(\mathbf{x} \in R_{jm}) \tag{2.16}$$

The function $1(\mathbf{x} \in R_{jm})$ yields 1 if $\mathbf{x} \in R_{jm}$ else it yields 0. Based on this base learner, Equation 2.12 can be rewritten with the expansion coefficients γ_{jm} of the j^{th} leaf node at the m^{th} boosting iteration [42],[43]:

$$F_m(\mathbf{x}) = F_{m-1}(\mathbf{x}) + \nu \cdot \gamma_{jm} \cdot 1(\mathbf{x} \in R_{jm}) \tag{2.17}$$

The parameter $0 < \nu \leq 1$ is introduced to control how much each boosting iteration contributes to the overall model. It is referred to as the learning rate. If the absolute error is used as a loss function $L(y, F(\mathbf{x}) = |y - F(\mathbf{x})|$, one finds that the expansion

coefficients are equal to the median of the actual residuals in the j^{th} leaf node at the m^{th} boosting iteration $\gamma_{jm} = \text{median}_{\mathbf{x}_i \in R_{jm}}(y_i - F_{m-1}(\mathbf{x}_i))$ [43]. A corresponding gradient boosted decision tree regression model using this loss function is illustrated in Figure 8 for different settings of the number of boosting iterations M. For that, the *GradientBoostingRegressor()* class of *scikit-learn* (v1.5.2) was used [44]. The learning rate was set to $\nu = 0.1$ and the maximum tree depth per boosting iteration was configured with a value of 2. The model was trained on data generated by adding random noise to a sine function. One can see that the model converges against this function with a rising number of boosting iterations. In this example with one-dimensional input data, training a BDT is visually equivalent to fitting an extensive step function to the data. The effect of the maximum tree depth can also be observed in Figure 8 by the black curve corresponding to a single boosting iteration. At the maximum tree depth of 2, the decision tree can have up to four leaf nodes. Therefore, the black curve shows four "steps" after adding one of these trees to the overall model.

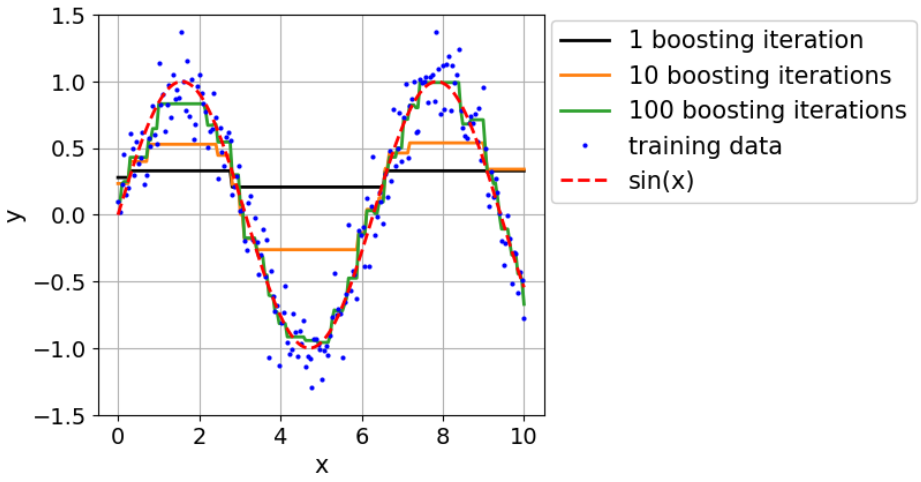

Figure 8: Results of a trained regression model based on the *GradientBoostingRegressor()* class of *scikit-learn* (v1.5.2) [44]. The training data (blue dots) was generated by adding random noise to a sine function (red dashed line). The model was trained for three different settings of the number of boosting iterations. Solid lines of different colours show the corresponding approximations of $\sin(x)$.

In the context of this thesis, slightly more advanced BDT models were applied to high-dimensional input spaces (see section 3) [8],[9]. However, at the core, they are based on the same algorithm as explained above. Therefore, they share the same parameters, which were mentioned throughout this section, including the learning rate and the maximum tree depth. These kinds of external model parameters are referred to as hyperparameters.

2.3.2. Evaluation of Model Predictions

This section explains the main tools and concepts used to evaluate the model predictions throughout section 3 and section 4.

Data Splitting and Model Validation Concepts:

In Figure 8, one can see that the trained BDT regressor approximates the underlying sine function of the training data with rising model complexity. However, the training data carries additional noise on top of the underlying sine function. Therefore, it is unavoidable that the BDT learns to model the noise component to some extent during the training process. A complex BDT might approximate this noise quite well, which results in a high prediction accuracy with respect to the training data. On the other hand, this leads to lower prediction accuracy on general data based on the same distribution around the respective predicted function, but with a different random noise component. This effect of over-fitting the training data is called overtraining [35]. To estimate the model performance independent of this effect, it is common practice to split the training data into a training set, only used for training, and a separate test set, only used for evaluating the model. A possible measure for the final accuracy of the model after training is the (test) loss obtained for the test set based on the respective loss function $L(y, F(x))$ used during training [35]. In this thesis, the loss is referred to as training loss or test loss, depending on which dataset is used to calculate it.

Having a fixed split of training and test data might also introduce a bias for the model evaluation, especially for smaller datasets. A method to estimate the uncertainty of the used measure for the prediction accuracy, depending on the applied split of training and test data, is cross-validation. Cross-validation works by splitting the data into k disjoint subsets (folds). Training is performed based on the fraction $(1 - 1/k)$ of the data, while testing is based on the remaining $1/k$ fraction. By repeating the training and testing process for k realizations of these fractions based on the disjoint folds, one can determine the average of the prediction accuracy measure and its related uncertainty [35].

In this work, a separate validation set is split from the training data in addition to the test set. This dataset is used to evaluate the early stopping algorithm of the used BDT models [8],[9]. The validation loss is calculated after each boosting iteration of the related gradient boosting algorithm. At first, this loss is minimized, similar to the training loss. With a rising number of boosting iterations, the BDT model possibly starts to overtrain, which leads to an increase in the validation loss. Early stopping means that the gradient boosting algorithm is aborted as soon as a significant increase in the validation loss is observed during training. This prevents the BDT model from over-fitting to the training data.

Binary Classification Methods:

The BDT classifier trained and optimized as shown in section 3.2.1 is used to sort simulated muon events of the IceCube Neutrino Observatory into two classes referred

to as the positive class and the negative class throughout this work.[3] Therefore, the elements of the input data of the BDT are often called events. The BDT classifier predicts the probability for a muon event to belong to the positive class. Based on this probability obtained for each muon event, a binary classification can be performed based on different methods. In this work, two different methods are used, which are explained below.

The predictions of the BDT classifier result in a certain distribution within the output space given by the interval $[0, 1]$. One possible approach for a binary classification is to define a probability threshold that divides the predicted probability distribution into two sections. Elements of the input space (muon events) with a corresponding prediction above this threshold are considered as *positive* events. This classification can be $true$ or $false$ depending on whether the event actually belongs to the positive class. The same holds for elements of the input space with a prediction below the threshold, which are considered as *negative* events. Again, the classification can be $true$ or $false$ depending on whether the event actually belongs to the negative class. This results in four prediction categories: *true positive* (TP), *false positive* (FP), *true negative* (TN), and *false negative* (FN) [35]. The probability threshold is typically set at a fixed value that minimizes the number of false predictions. Based on the number of events in each category, several quantities for measuring the performance of the classifier can be defined (see later in this section).

A second approach to separate elements of the input space into two classes is using a rejection sampling method. In contrast to the aforementioned fixed-threshold approach, this method allows for dynamic event classification based directly on the predicted probabilities. In the context of this thesis, it is defined as follows. Each element i of the input sample has a predicted probability p_i and a corresponding weight $w_i = 1/p_i$ assigned to it. If p_i is smaller than a minimum accepted probability of $p_{min} = 0.1\,\%$ it is redefined to be equal to this probability ($p_i = p_{min}$). This is done to avoid overly large weights for predictions very close to zero. An event i is classified as accepted (positive) or rejected (negative) by comparing p_i to a random number $n_{rand,i}$ within the interval $[0, 1]$:

$$n_{rand,i} < p_i \Rightarrow \text{event } i \text{ is accepted (positive)} \tag{2.18}$$

$$n_{rand,i} \geq p_i \Rightarrow \text{event } i \text{ is rejected (negative)} \tag{2.19}$$

Consequently, events with a probability $p_i \to 1$ are most probably classified as positive, while events with $p_i \to 0$ are most probably classified as negative. The advantage of this approach is that one does not have to search for an optimal probability threshold that minimizes the number of false predictions. On the other hand, this approach reduces transparency regarding whether a given prediction is $true$ or $false$, due to the probability-based sampling. The accepted (positive) event sample obtained by this approach is weighted based on the weights w_i. The effective sample size $N_{eff} \leq n$ equivalent to an unweighted sample is given by the following formula, where n is the number of positive events obtained by the rejection sampling [45]:

$$N_{\text{eff}} = \left(\sum_{i=1}^{n} w_i\right)^2 \Big/ \left(\sum_{i=1}^{n} w_i^2\right) \tag{2.20}$$

[3] The actual meaning of these classes in the context of this work is explained along with the results presented in ongoing sections (see Table 3).

Measures of Classifier Performance:

Related to the prediction categories of the fixed-threshold approach, several quantities can be defined, which help to measure the classifier's performance. The value of each of these quantities depends on the applied probability threshold [35]:

$$\text{accuracy} = \frac{\text{TN} + \text{TP}}{N} \tag{2.21}$$

$$\text{error rate} = \frac{\text{FN} + \text{FP}}{N} \tag{2.22}$$

$$R_{\text{pos}} = \frac{\text{FP} + \text{TP}}{N} \tag{2.23}$$

$$R_{\text{neg}} = \frac{\text{FN} + \text{TN}}{N} \tag{2.24}$$

$$\text{FPR} = \frac{\text{FP}}{\text{TN} + \text{FP}} \tag{2.25}$$

$$\text{TPR} = \frac{\text{TP}}{\text{TP} + \text{FN}} \tag{2.26}$$

$$\text{FNR} = \frac{\text{FN}}{\text{TP} + \text{FN}} \tag{2.27}$$

These quantities are typically determined with respect to the test set. The total number of events in the test set is given by N. The variables TP, FP, TN, and FN are the number of events in the respective prediction category. The accuracy is the fraction of events with a *true* prediction independent of the class. The error rate is the inverse of the accuracy and defines the fraction of events with a $false$ prediction. A high accuracy (low error rate) is an indicator of a well-performing classifier. The rates R_{pos} and R_{neg} are the fractions of events classified as positive or negative, respectively, independent of whether the predictions are correct or not. R_{neg} is used as a metric during the optimization of the BDT classifier as presented in section 3.2.1. Further information in this regard can be found at the beginning of section 3.2. The $false$ *positive rate* FPR and the *true positive rate* TPR are used to define the Receiver Operating Characteristic (ROC) curve (see Figure 9), which is given by plotting TPR against FPR for different values of the applied probability threshold [35].[4] The shape of the ROC curve is a qualitative measure of the classifier's performance. The closer the curve is to the upper left corner of the plot, the better the underlying model. If the predictions of the model are random, as for a coin toss, the ROC curve is equal to the angle bisector of the plot (see "no skill" line in Figure 9). In that way, the performance of different models can be compared, which is useful for optimizing hyperparameters (see section 3.2.1). A quantification can be achieved by calculating the area under the ROC curve ($0 \leq \text{AUC} \leq 1$). The larger this area is, the better the underlying model [35]. In this thesis, a variation of the ROC curve is used, where the $false$ *negative rate* FNR is plotted against FPR, referred to as the FNR-FPR curve. Since FNR = 1 – TPR, the area under the FNR-FPR curve has to be

[4] Note that the values of TP, FP, TN, and FN are often normalized by N and therefore referred to as normalized rates in this thesis. They should not be confused with the $false$ *positive rate* FPR, the *true positive rate* TPR, or the $false$ *neagtive rate* FNR.

minimized for an optimal model, and the curve itself has to converge to the lower left corner of the plot instead. The "no skill"-line goes from the upper left to the lower right corner for the FNR-FPR curve.

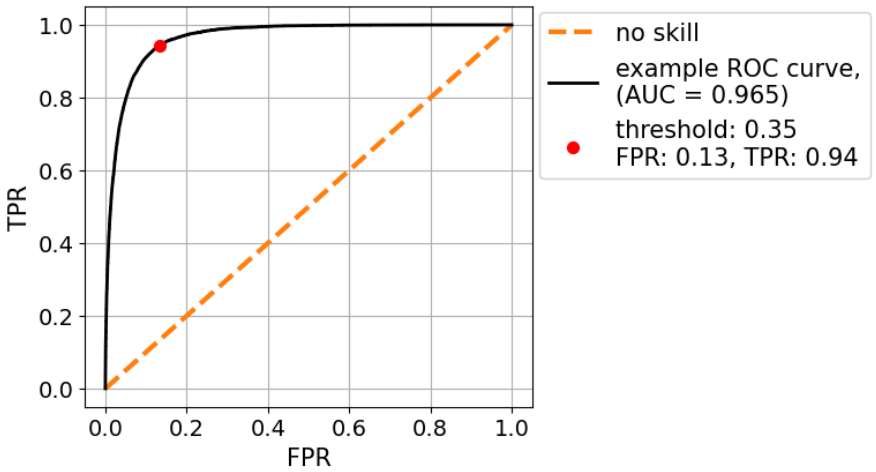

Figure 9: Example of a ROC curve (TPR against FPR) of a binary classifier. The red dot corresponds to a probability threshold of 0.35 applied to the underlying probability distribution. The orange dotted line marks the expected ROC curve of a model with random predictions corresponding to a fair coin toss.

3. Development of a Classifier

The IceCube simulation, as described in section 2.2, is a crucial tool for modelling the signal and background processes for the IceCube Neutrino Observatory. As was pointed out in section 2.1.3, the background reduction efficiency is an important parameter that has to be estimated to quantify how effectively background events, like cosmic-ray induced muons, are filtered out during data processing. Therefore, large samples of background muons have to be simulated. In section 2.2 it was also explained that these simulations are of high computational and, with that, financial cost. In this context, a large amount of computational resources is consumed by the propagation of Cherenkov photons, which are induced by the muon energy losses. Therefore, reducing the calculation time that has to be spent on simulation steps like the photon propagation is advantageous for future simulations of background muons.

In the current state of the simulation chain, photon propagation is performed for all muon events that were generated and propagated in previous simulation steps. In later simulation steps, background reduction is performed by applying multiple filters to those events. In that way, a large amount of background muons is removed from the final event sample, although a lot of computational resources were spent in the first place.

In order to contribute to the solution of this efficiency problem, a BDT classifier (see section 2.3.1), was trained as part of this thesis. As shown below in the current section, the classifier is able to differentiate between muon events that pass the MESE filter (see section 2.1.3) and events that are rejected by it, based on the muon energy loss information. By inserting this model right before the simulation step of photon propagation (see section 2.2), it is possible to remove events from the simulation chain that would be discarded with a high chance by the MESE filter in later simulation steps. With this hybrid simulation approach of combining traditional simulation methods with a machine learning model, computational resources can be saved.

The following section 3.1 shows a proof of concept for this idea. It shows the evaluation and results of a BDT regression model that was trained on data generated by a toy simulation with several input vector and hyperparameter settings. In section 3.2, the optimization and results of a BDT classification model that was trained on data from the IceCube simulation are discussed. In section 3.2.2, the resulting efficiency of the optimized BDT is compared with the results of a recurrent neural network (RNN) model that was trained for the same purpose [4].

3.1. Proof of Concept with Data of the ToyCube Simulation Tool

In order to show that a hybrid simulation approach, as proposed in the beginning of section 3, conceptually works, a BDT regression model of the python library *scikit-learn* [1] was trained on data that were generated by the ToyCube simulation tool [2],[4]. Different hyperparameter settings as well as different construction methods of the BDT input vector were analysed. Details on the generation of training data, training of the model and the resulting model predictions are presented below.

3.1.1. ToyCube Simulation Tool

Simple access to training data is generally useful for the development of machine learning models. Therefore, the ToyCube simulation tool was developed in cooperation with a Bachelor student at ECAP (B. Mayer), in order to simulate the propagation and energy losses of muons in ice and to estimate the number of induced photons that are detected in the IceCube veto DOMs (see section 2.1.3). ToyCube was implemented in Python, specifically using the PROPOSAL library for muon propagation in ice [5]. As section 3.1.2 shows, the stochastic energy losses of the respective muon event serve as input of the BDT regression model, while the respective number of resulting photons detected in the veto region of IceCube is used as the respective training target. Figure 10 illustrates the veto region within the IceCube detector array, as it is defined in the context of this simulation tool. A detailed review of ToyCube and the development of a RNN regression model can be found in the Bachelor's thesis of in B. Mayer [4]. The related source code to that thesis can be found in the *main* branch of the ToyCube git repository [2]. The source code related to this Master's thesis and the development of the BDT regression model can be found in the side branch of the same repository [3].[5] In the following, information on how ToyCube generates training data is provided.

Each muon event simulated by ToyCube has a random initial position and initial propagation direction within the IceCube coordinate system. The initial position is defined by assigning a random position $\vec{r}_{\text{disc}}$ on a disc of radius $r = 800\,\text{m}$ to each muon of a given sample. The disc is centered at $(x = 0\,\text{m}, y = 0\,\text{m}, z = 930\,\text{m})$ and perpendicular to the z-axis. Each position vector $\vec{r}_{\text{disc}}$ is rotated by a random normalized direction vector $\vec{d}$ as defined by equation (4) of [4]. The initial position is defined as the point in space to which $\vec{r}_{\text{disc}}$ points after the rotation is applied. The (initial) propagation direction is defined as $-\vec{d}$. Initial positions and propagation directions of a sample of 200 muons are illustrated in Figure 11 before and after the rotation is applied. Furthermore, each muon is generated with a random initial energy between 0.1 TeV and 10^3 TeV. With this assignment of muon properties, it is possible to simulate the propagation of muons along random particle tracks through the detector.

[5] Whenever ToyCube is mentioned in this work, it refers to the code and implementation given by that side branch.

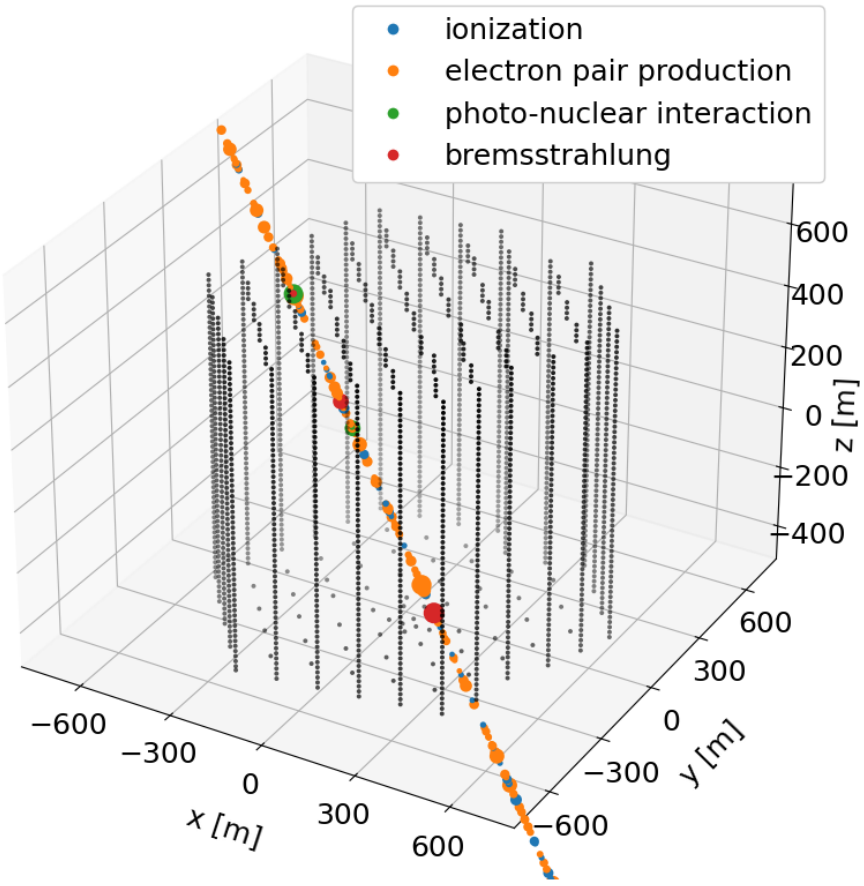

Figure 10: Positions of the IceCube veto DOMs (black dots) and energy losses along the track of a muon, as defined and generated by ToyCube. Different energy loss types have different colours. The size of the energy loss dots is scaled with respect to the deposited energy within the ice.

The actual simulation of muons is based on the aforementioned PROPOSAL python library, which generally can be used to simulate the propagation of charged leptons and gamma rays through different media. In the specific case of the ToyCube simulation, charged muons are propagated through a homogeneous ice sphere, which is defined by the *config_minimal.json* file [3]. Four different types of discrete particle interactions of muons within ice are simulated: ionization, electron pair production, photo-nuclear interactions and bremsstrahlung. Based on the respective cross section, PROPOSAL is able to calculate the energy losses along the muon particle track. Each energy loss has a unique position $\vec{r}_{\text{loss}}$ and amount of deposited energy E_{loss} assigned to it. Figure 10 shows the particle track of a muon propagating through the IceCube detector array, as generated by ToyCube.

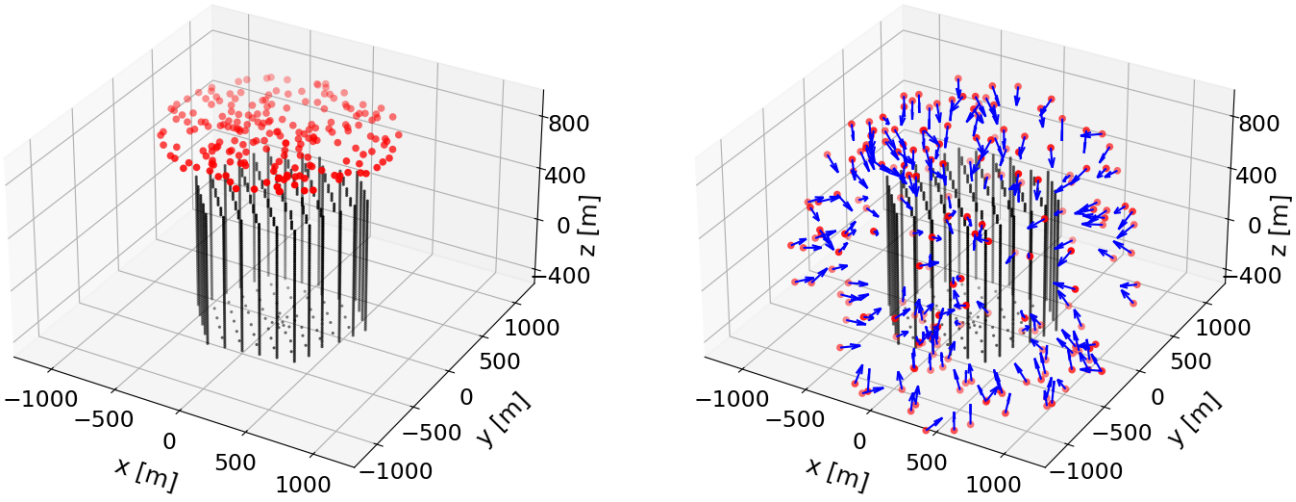

Figure 11: **Left plot:** Random positions (red dots) of 200 muons on a disc at $z = 930\,\mathrm{m}$ before the random rotation is applied, compared to the positions of the IceCube veto DOMs (black dots). **Right plot:** Initial positions (red dots) and propagation directions (blue arrows) of 200 muons as simulated by ToyCube, compared to the positions of the IceCube veto DOMs (black dots).

For a given energy loss and veto DOM, the number of photons induced by the energy loss and detected by the veto DOM is estimated by using the following formula, which depends on E_{loss} and the distance r between $\vec{r}_{\mathrm{loss}}$ and the position of the veto DOM [4],[6]:

$$N_{\mathrm{ph}}(E_{\mathrm{loss}}, r) = \frac{E_{\mathrm{loss}}}{300\,\mathrm{TeV}} \cdot \mu(r) = \frac{E_{\mathrm{loss}}}{300\,\mathrm{TeV}} \cdot n_0 A \cdot \frac{e^{-\frac{r}{\lambda_p}}}{4\pi\lambda_c r \cdot \tanh\left(\frac{r}{\lambda_c}\right)} \tag{3.1}$$

The propagation length λ_p is defined as $\lambda_p = \sqrt{\lambda_a \lambda_e/3}/1.07$, with the absorption length $\lambda_a = 98\,\mathrm{m}$ and effective scattering length $\lambda_e = 24\,\mathrm{m}$. The constant λ_c is defined as $\lambda_c = \lambda_e/3\zeta$, with $\zeta = \exp(-\lambda_e/\lambda_a)$ [6]. The product of the number of emitted photons n_0 by a point source and the effective photon collection area A of the receiving sensor was determined by fitting $\mu(r)$ to simulation data obtained with Monte-Carlo photon propagation [4],[7]. The resulting value is given as $n_0 \cdot A = (6.4 \pm 0.5) \cdot 10^7\,\mathrm{m}^2$ [7].

The total number of photons $N_{\mathrm{ph}}^{\mathrm{tot}}$ that are induced by a single muon event and detected by the respective veto DOMs is calculated in the following way: At first, N_{ph} is calculated for each loss of a muon event with respect to each veto DOM. This leads to *number of losses* times *number of veto DOMs* values of N_{ph}. In a second step $N_{\mathrm{ph}}^{\mathrm{tot}}$ is obtained by summing over all the resulting values of N_{ph}.

For some muon events, no energy loss is generated during simulation. These events are excluded from the datasets produced by ToyCube, since the BDT needs at least one energy loss as input to make a prediction. In order to minimize the size of the resulting training data files and save computational time during training, a filter condition is

applied to every energy loss of each simulated muon, based on the number of detected photons per loss with respect to all veto DOMs $N_{\text{ph}}^{\text{veto}}$. The filter condition is that all energy losses which yield $N_{\text{ph}}^{\text{veto}}$ <= 0.01 are excluded from the training data. Since all the excluded energy losses hardly contribute to the total number of detected photons $N_{\text{ph}}^{\text{tot}}$, it is assumed that no relevant information is lost by applying this filter. If a muon event has no energy losses after applying the filter, it is removed from the simulated dataset completely. Figure 12 shows the same muon track as Figure 10 but after the filter condition is applied.

In the context of ToyCube, the BDT regression model is supposed to predict $N_{\text{ph}}^{\text{tot}}$ for each muon event, based on the filtered energy losses of the respective muon. By comparing the predicted number of photons in the IceCube veto region to a threshold, an event filter can be modeled. If the predicted value is above the threshold the event passes the filter, if the predicted value is below the threshold it is rejected by the filter. In that way, a first impression on the performance of a classifier model can be obtained, as discussed in the following subsections of section 3.1.

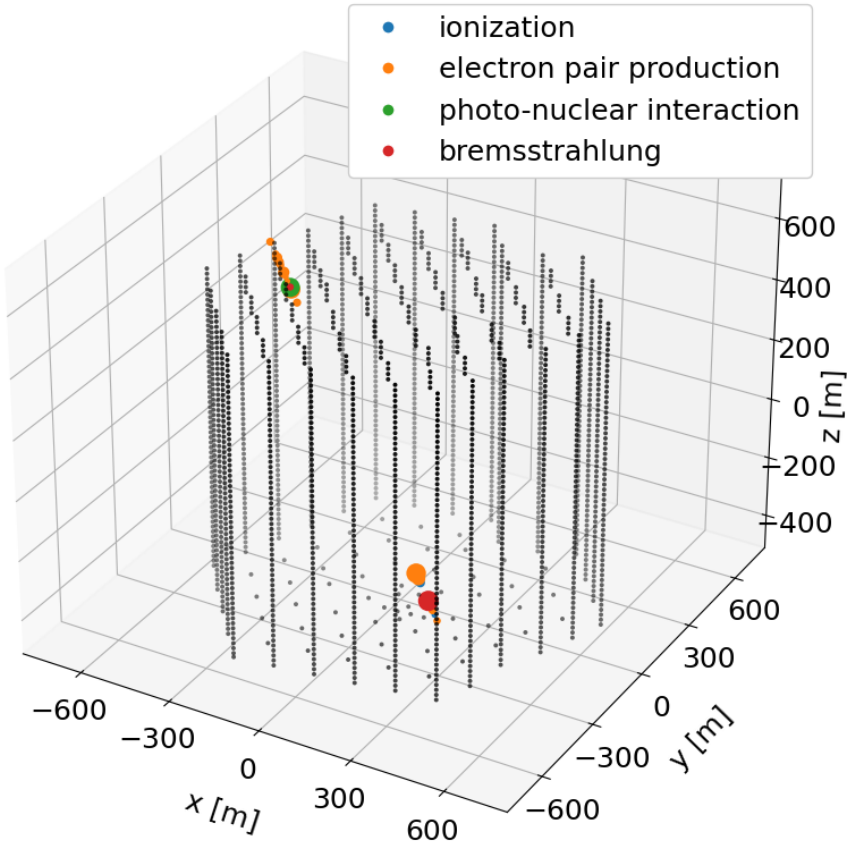

Figure 12: Positions of the IceCube veto DOMs (black dots) as defined in ToyCube. Compared to Figure 10, this plot shows the energy losses along the track of a muon, after the filter condition mentioned in the text was applied. Different energy loss types have different colours. The size of the energy loss dots is scaled with respect to the deposited energy within the ice.

3.1.2. Implementation of a BDT Regression Model

The BDT regression model that was trained on ToyCube data is defined by the *sklearn.ensemble.HistGradientBoostingRegressor()* class of *scikit-learn* (v1.5.2) [1],[8].[6] In general, this model is related to a gradient-boosted decision tree as described in section 2.3.1. In addition, it preprocesses the input vectors before training. In particular, it sorts each feature of a given set of input vectors into up to 256 integer-valued bins. For large datasets (10000 samples and above), this leads to a much faster training algorithm, since the input consists of a fixed histogram-like structure instead of continuous values. This is advantageous for the use case of this work, since the amount of background muons produced in the IceCube simulation is typically larger than 10000 events. This also applies to the ToyCube simulation tool.

Before the BDT regressor was trained, the training input and target had to be defined. As described in section 3.1.1, the training data generated by ToyCube provides the following information for each muon event: The position $\vec{r}_{\text{loss}}$ and deposited energy E_{loss} of each energy loss that passes the filter condition $N_{\text{ph}}^{\text{veto}} > 0.01$ and the number of photons $N_{\text{ph}}^{\text{tot}}$ that are induced by those losses and detected by the IceCube veto DOMs. The latter is supposed to be predicted by the BDT regressor based on the filtered set of values of $\vec{r}_{\text{loss}}$ and E_{loss} as input. Thus, the training target was defined as the respective value of $\log_{10}(N_{\text{ph}}^{\text{tot}})$ for each sample. The logarithm was applied to restrict the range of the target values, since $N_{\text{ph}}^{\text{tot}}$ covers several orders of magnitude. By definition, the BDT regressor expects an input vector of the same length for each sample. Since the number of energy losses can be different for each muon, $\vec{r}_{\text{loss}}$ and E_{loss} can not be used as input directly. Instead, the energy losses of each sample were mapped to a fixed number of values to create input vectors of a suitable length. This was achieved by defining a binning grid that splits the IceCube detector volume into a fixed number of three-dimensional spatial bins, with energy losses assigned according to their positions. Different binning grid versions were tested on ToyCube data. They are explained in more detail below.

The first version is based on two concentric cylinders of different radii ($r_{\text{inner}} = 380\,\text{m}$, $r_{\text{outer}} = 700\,\text{m}$) that are centered at $(x = 0\,\text{m}, y = 0\,\text{m}, z = 0\,\text{m})$ within the IceCube coordinate system (see Figure 13). The top and bottom surfaces of both cylinders are oriented perpendicular to the z-axis. Both cylinders are of the same height, which means both reach from $z = -600\,\text{m}$ to $z = 600\,\text{m}$. The actual bins are defined by dividing the volume between the inner and outer cylinder into n_{seg} circular segments. Each of those segments has the height of the cylinders and an angular range of $\alpha = 360°/n_{\text{seg}}$. The volume of the inner cylinder is used as an additional bin.

For each muon sample the position $\vec{r}_{\text{loss}}$ of each corresponding energy loss is compared against the following condition for the i^{th} bin ($i \in \{0, ..., n_{\text{seg}}-1\}$) in cylindrical coordinates (ρ, ϕ, z):

$$(-600\,\text{m} < z < 600\,\text{m}) \wedge (r_{\text{inner}} \leq \rho < r_{\text{outer}}) \wedge (i \cdot \alpha \leq \phi < (i+1) \cdot \alpha) \tag{3.2}$$

And against the following condition if $i = n_{\text{seg}}$:

$$(-600\,\text{m} < z < 600\,\text{m}) \wedge (\rho < r_{\text{inner}}) \tag{3.3}$$

[6] For simplicity the *sklearn.ensemble.HistGradientBoostingRegressor()* model is referred to as BDT regressor in this thesis.

An energy loss is located within the i^{th} bin if the corresponding condition is met. The information assigned to the i^{th} bin is given by the summed energy $E_{\text{bin}_i} = \sum_{\text{losses}_i} E_{\text{loss}}$ with respect to all energy losses that are located within the respective bin.

The i^{th} component of the corresponding input vector of a muon event is defined as $\log_{10}(E_{\text{bin}_i})$. The logarithm was applied for the same reason as for the training target. Different variations of the cylindrical binning grid were used to train the BDT regressor. At first, the number of circular segments was set to $n_{\text{seg}} = 8$, resulting in $n_{\text{bins}} = 9$ bins in total, but also versions of the cylindrical binning grid with $n_{\text{seg}} = 16$ ($n_{\text{bins}} = 17$) and $n_{\text{seg}} = 72$ ($n_{\text{bins}} = 73$) were tested (see section 3.1.3). Figure 13 shows the position of the cylinders with respect to the IceCube veto DOMs in the case of $n_{\text{seg}} = 8$.

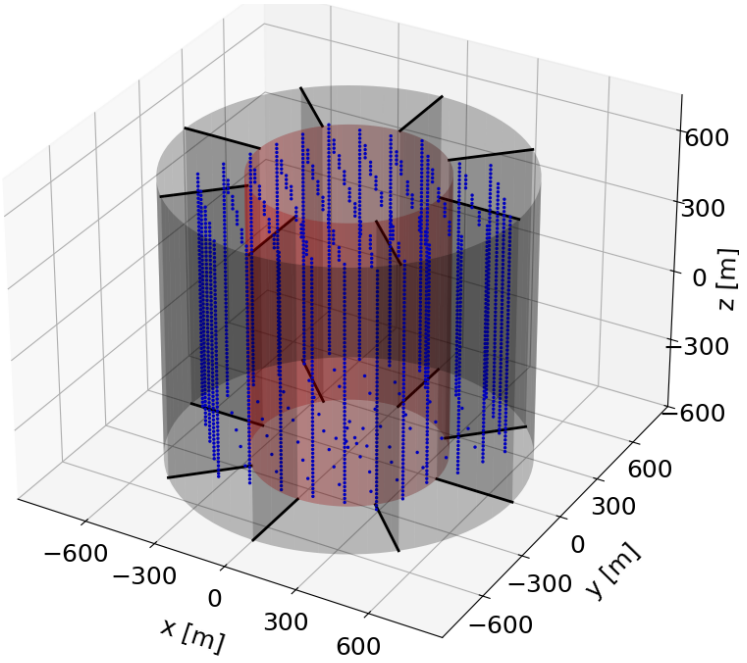

Figure 13: IceCube veto DOMs (blue dots) as well as inner cylinder (red) and outer cylinder (black) of the binning grid, which is used to define input vectors of a fixed length for the BDT regression model. The black lines at the outer cylinders' top and bottom indicate the bins between the cylinders, in the case of $n_{\text{seg}} = 8$. Rotated variations of this plot can be found in section A.1.1.

The second approach to defining the binning grid is based on a hexagonal bin structure, which reflects the hexagonal detector geometry of IceCube. As shown in Figure 14, this grid consists of multiple regular hexagonal prisms (bins) which form a larger hexagonal prism in total. Each bin reaches from $z = -600\,\text{m}$ to $z = 600\,\text{m}$. With that, the hexagonal and the cylindrical binning grids are equal in height. To roughly match the grid size within the xy-plane as well, the side length of the large hexagonal prism is defined by 5 bins with a hexagon radius of $r_{\text{hex}} = 108\,\text{m}$.[7] Additionally, the hexagonal grid is rotated anticlockwise with respect to the z-axis by 8°.

[7] Here r_{hex} is defined in the xy-plane as the radius of the circumcircle of the respective hexagon prism.

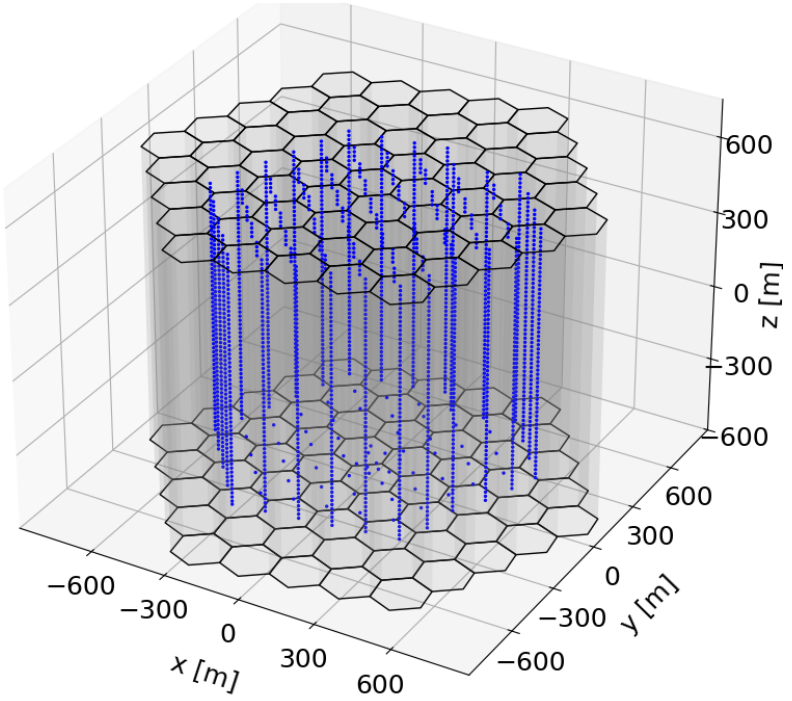

Figure 14: IceCube veto DOMs (blue dots) as well as the hexagonal binning grid (black), which is used to define input vectors of a fixed length for the BDT regression model. Rotated variations of this plot can be found in section A.1.1.

This rotation approximates the tilt of the IceCube string array within the IceCube coordinate system. Based on the resulting grid of 61 bins, the construction of the BDT input vector relies on the same concept as for the cylindrical binning approach. The conditions that must be met for assigning an energy loss to a specific bin were adjusted to account for the hexagonal geometry. For each muon sample, the distances between the projection of $\vec{r}_{\text{loss}}$ onto the xy-plane and each bin centre in the xy-plane are calculated for each energy loss. Then the respective energy loss is assigned to the bin whose centre is closest in the xy-plane. The corresponding distance in the xy-plane is called d_{min}. Each energy loss included in the input vector has to be located within the binning grid. Therefore the following condition must be fulfilled for $\vec{r}_{\text{loss}}(x, y, z)$ and d_{min}:

$$(-600\,\text{m} < z < 600\,\text{m}) \wedge (d_{\text{min}} \leq r_{\text{hex}}) \tag{3.4}$$

For both the cylindrical and the hexagonal binning approach, the region close to the veto DOMs, where most of the filtered energy losses are expected to be located (compare Figure 12), is fully covered by the respective binning grid. Therefore, most of the information on the deposited energy is expected to be preserved in both cases. Having multiple bins ensures that information on the muon direction and location is encoded in the respective input vector to a certain extent. However, due to differences in geometry between the binning approaches, this kind of information is encoded differently depending on the applied binning method. The next subsection shows the impact of these differences and of various hyperparameter configurations on the training process and the performance of the BDT regressor.

3.1.3. Hyperparameter Optimization of the BDT Regression Model

The BDT regressor, as defined in [8], has various hyperparameters that can be adjusted to optimize its performance. In the context of this work, parameter scans were performed with respect to three of those, *max_iter* (*MI*), *learning_rate* (*LR*), and *max_depth* (*MD*). All remaining parameters were set to the respective default setting (see Table 17 in the appendix). With *MI*, the maximum number of boosting iterations applied during training is adjusted. *LR* is equivalent to the learning rate ν described in section 2.3.1. The maximum depth of the decision tree in each boosting iteration is controlled by *MD*. Maximum tree depth means the number of decision splits from the root node to the deepest leaf node of the tree (see decision trees in section 2.3.1). The assigned values during each parameter scan are listed in Table 1. In each scan, the BDT regressor was trained once for each of the 320 resulting hyperparameter combinations. As mentioned in section 3.1.2, different binning grid configurations were compared. This means one parameter scan was performed for each of the following configurations: cylindrical grid with $n_{\text{bins}} = 9$, $n_{\text{bins}} = 17$, or $n_{\text{bins}} = 73$, and hexagonal grid with $n_{\text{bins}} = 61$.

hyperparameter	applied values
max_iter (*MI*)	[250, 500, 750, 1000]
learning_rate (*LR*)	$[1 \cdot 10^{-3}, 1 \cdot 10^{-2}, 5 \cdot 10^{-2}, 1 \cdot 10^{-1}]$
max_depth (*MD*)	$[1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13, 14, 15, 16, 17, 18, 19, 20]$

Table 1: Values of *MI*, *LR*, and *MD* involved in each parameter scan.

For the training, a dataset of 87563 muon samples was generated with the ToyCube simulation tool. Before the parameter scans were performed, a fraction of 10 % of this dataset was separated as a test set. Since *scikit-learn* provides an early stopping algorithm for the BDT regressor, the remaining 90 % subset was randomly split into a 90 % training set and a 10 % validation set. The training/validation split was performed individually for each training run. The default loss function that is applied for the BDT regressor is the '*squared_error*'. In *scikit-learn* (*version 1.5.2*), this function is defined as [8][8]:

$$L(y_{\text{true}}, y_{\text{pred}}) = -\frac{1}{2 \cdot n_{\text{samples}}} \cdot \sum_{i=0}^{n_{\text{samples}}} (y_{\text{true,i}} - y_{\text{pred,i}})^2 \tag{3.5}$$

Applied to this work $y_{\text{true,i}}$ denotes the true value of $\log_{10}(N_{\text{ph}}^{\text{tot}})$ of sample i, as calculated by ToyCube. The respective prediction of that value from the BDT regressor is $y_{\text{pred,i}}$. The variable n_{samples} is the number of samples in the respective dataset. With default settings, early stopping is triggered after a certain boosting iteration, if the calculated validation loss after none of the 10 latest boosting iterations has improved compared to a reference value. The reference value is the 11^{th}-last validation loss plus a tolerance of $1 \cdot 10^{-7}$. A general explanation of the terms training set, test set, validation set, and early stopping can be found in section 2.3.2.

[8] In any figure of section 3.1.3 that shows the training, validation, or test loss, the respective loss corresponds to this function multiplied by −2.

After the training process, each hyperparameter scan was evaluated based on the training, validation, and test loss. The corresponding results of the parameter scan with respect to the cylindrical binning approach ($n_{\text{bins}} = 9$) are illustrated in Figure 15 and Figure 16. Figure 15 shows the training and validation loss over the boosting iteration at $MI = 1000$ for certain combinations of MD and LR. In Figure 16 the test loss L_{test} is plotted over MD for all combinations of MI and LR. The impact of different hyperparameter settings on the training, validation, and test loss is described below.

The training and validation loss curves are almost identical for any given hyperparameter setting (see Figure 15). Therefore, overfitting to the training data was not observed for any of the chosen settings. No peaks or any abrupt changes were observed in the training and validation loss curves. Early stopping is mostly triggered at higher learning rates ($LR \in \{0.05, 0.1\}$) combined with values of $MD \geq 4$ and $MI > 250$. This can be seen in Figure 16 where corresponding loss curves stop before $MI = 1000$ is reached. With rising MD, early stopping is triggered at earlier boosting iterations. Still, the final loss is similar (compare solid curves of the lower subplots of Figure 15). This implies that a BDT model consisting of many shallow decision trees behaves similarly to a BDT model consisting of a few deep decision trees.

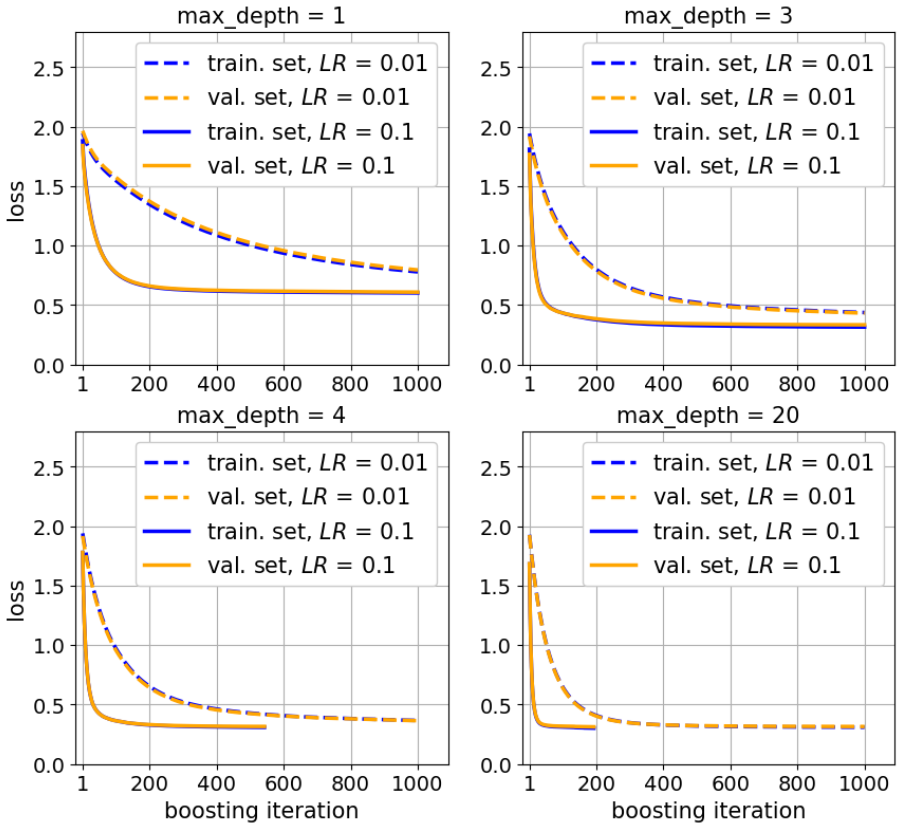

Figure 15: Training loss (blue curves) and validation loss (orange curves) against the boosting iteration ($MI = 1000$) for the cylindrical binning with $n_{\text{bins}} = 9$. Each subplot shows two pairs of curves and corresponds to a certain value of MD. Each pair of curves corresponds to a certain value of LR: $LR = 0.1$ (solid curves), $LR = 0.01$ (dashed curves).

This also shows that adding more or using deeper decision trees does not have an impact on the final loss if the internal structure of the BDT model reaches a certain level of complexity. Figure 15 also shows that models with a lower learning rate (dashed loss curves) need more boosting iterations to reach the same loss as models initialized with a higher learning rate (solid loss curves). This is because the learning rate controls how much each decision tree contributes to the overall BDT model. A lower learning rate results in a smaller contribution to the BDT model of each tree. This leads to a smaller shrinkage of the loss per boosting iteration. At the same time, the used values of *MI* set an upper limit to the number of boosting iterations. Therefore, the final training and validation loss is generally lower for models with higher learning rates.

This is also confirmed by the results of the test loss in Figure 16, since the red data curves corresponding to $LR = 0.1$ generally show the lowest test loss. Comparing the different subplots of Figure 16 shows that an increase of *MI* only leads to a visible difference in the test loss, if learning rate and maximum depth are set to values of about $LR = 0.001$ and $MD < 5$. This is because otherwise early stopping tends to be triggered during training, and with that the maximum boosting iteration given by *MI* is never reached. Therefore, at higher values of *LR* and *MD*, the test loss does not depend on *MI* anymore. Furthermore a saturation of the test loss at $L_{\text{test}} \approx 0.32$ is observed for $LR \geq 0.05$, if *MD* is increased beyond a value of about 10.

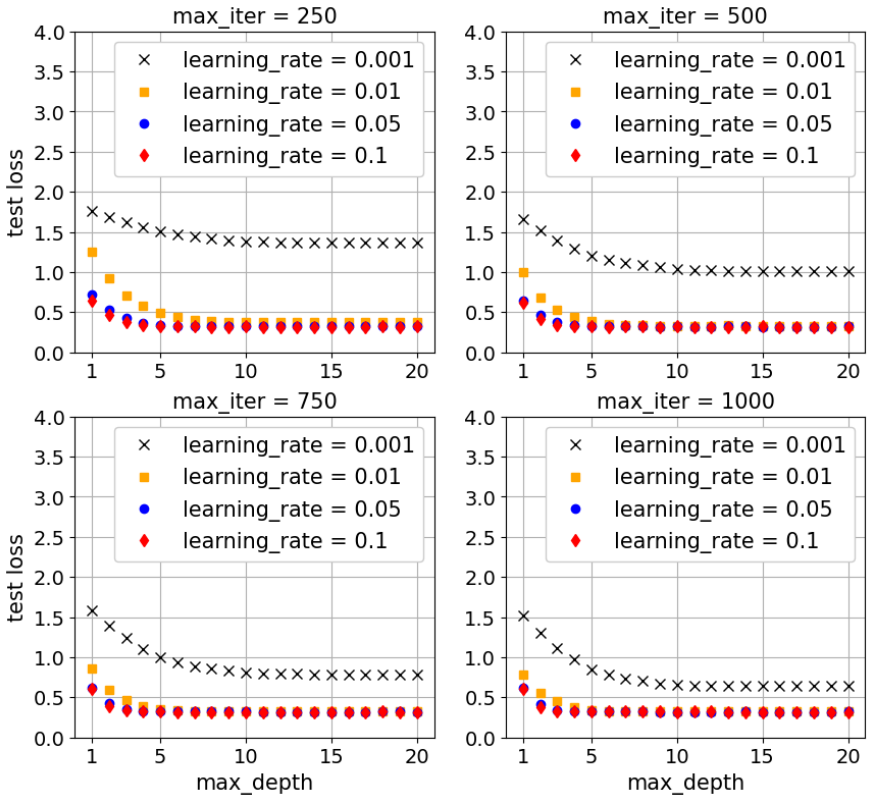

Figure 16: Calculated loss for the test set against *MD* for the cylindrical binning with $n_{\text{bins}} = 9$. Each subplot shows a set of four curves and corresponds to a certain maximum boosting iteration *MI*. Each curve within a subplot corresponds to a certain value of the learning rate *LR*.

This observation is related to the fact that, after a certain point, adding more or deeper trees to the BDT model does not significantly impact the final training and validation loss. This also applies to the test loss, which leads to the aforementioned saturation effect.

In summary, the best results for the test loss in the case of cylindrical binning with $n_{\text{bins}} = 9$ were observed with a value of $L_{\text{test}} \approx 0.32$, for learning rates of $LR \geq 0.05$, and maximum depths of $MD > 10$. Additionally, a maximum boosting iteration of $MI \geq 500$ seems to be a good choice. This ensures that boosting iterations are applied during training until early stopping is triggered and not until MI is reached. With that, a broad range of hyperparameter settings was found that yield approximately equal results for the test loss.

For the parameter scans corresponding to the other three binning grid versions, the general behaviour of the losses with respect to MI, LR, and MD was similar. Equivalent plots to Figure 15 and Figure 16, can be found in section A.1.2 of the appendix. A difference that was observed between the scans is that the value of MD above which the test loss saturates, shifts to larger values for increasing values of n_{bins}. This shows that for larger input vectors, a more complex internal BDT structure is needed to reach the same test loss. Figure 17 illustrates that, by showing a plot of the test loss over MD for the different versions of the binning grid. To ensure comparability, the plot shows the results corresponding to the values of $MI = 1000$ and $LR = 0.1$. With that, the test loss reaches a saturated state at higher values of MD, independent of the binning grid version.

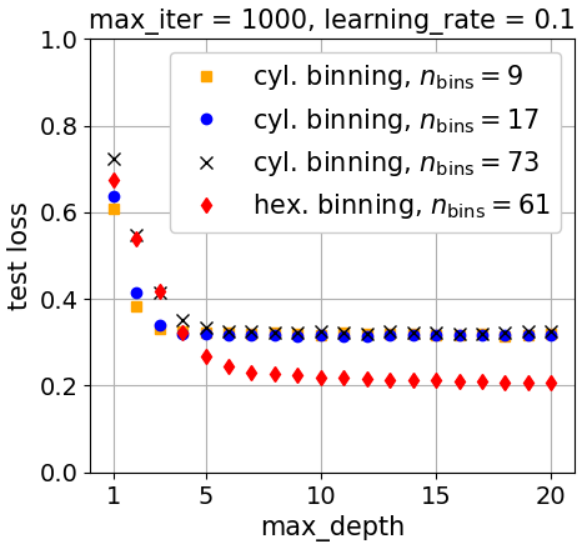

Figure 17: Calculated loss for the test set against MD at $MI = 1000$, $LR = 0.1$. Each curve corresponds to a different binning grid version: cylindrical with $n_{\text{bins}} = 9$ (orange squares), $n_{\text{bins}} = 17$ (blue dots), $n_{\text{bins}} = 73$ (black crosses), and hexagonal with $n_{\text{bins}} = 73$ (red diamonds).

Increasing the number of bins for the cylindrical binning approach from $n_{\text{bins}} = 9$ to $n_{\text{bins}} = 17$ or $n_{\text{bins}} = 73$ did not lead to a significant improvement in the test loss at values of $MD > 10$, where the saturation is observed. A difference was recognized for the hexagonal binning (red curve in Figure 17). Above a maximum depth of $MD = 5$, the test loss of the hexagonal binning is lower than for the cylindrical binning approaches. The lowest value of the test loss across all scans was observed for the hyperparameters $MI = 1000$, $LR = 0.1$, and $MD = 20$, in combination with the hexagonal binning, with a value of $L_{\text{test}} \approx 0.21$.

Therefore, the hyperparameter setting $MI = 1000$, $LR = 0.1$, and $MD = 20$ was chosen to further compare the different binning approaches with respect to the predictions on the test set. Figure 18 shows two different plots for each binning approach. In the left column, the predictions y_{pred} are plotted over the respective true value y_{true} of $\log_{10}(N_{\text{ph}}^{\text{tot}})$ as calculated by ToyCube. A black dashed line marks the angle bisector, where $y_{\text{pred}} = y_{\text{true}}$. The residuals $(y_{\text{pred}} - y_{\text{true}})$ with respect to that line are an indicator of how accurate the predictions of the respective model are. The median and 68 % confidence interval of the residuals were calculated based on all samples of the test set. The respective values for each binning approach are listed in Table 2. The 68 % confidence interval was determined by calculating the 16 % and 84 % quantiles. In addition, a bin-wise calculation of the median and the 68 % confidence interval of the residuals was performed based on 50 bins along the x-axis (y_{true}). The resulting values are shown as orange and red lines in the plots of the left column. In the right column of Figure 18, the residuals are plotted over the binned initial muon energy. This means that the corresponding x-axis is divided into 10 energy bins. The median and 68 % confidence interval of the residuals were calculated per energy bin and are shown as blue data points.

binning approach	cyl., $n_{\text{bins}} = 9$	cyl., $n_{\text{bins}} = 17$	cyl., $n_{\text{bins}} = 73$	hex., $n_{\text{bins}} = 61$
median $\pm\, \sigma$	$-0.02^{+0.50}_{-0.51}$	$-0.01^{+0.50}_{-0.51}$	$0.01^{+0.49}_{-0.52}$	$0.01^{+0.41}_{-0.40}$

Table 2: Median and 68 % confidence interval of the residuals $(y_{\text{pred}} - y_{\text{true}})$ based on the test set. Each value corresponds to a certain binning approach.

The values given in Table 2 show that the overall median of the residuals is quite close to 0 for all binning approaches. The hexagonal binning approach shows the smallest 1σ spread of the residuals around the respective median, which is consistent with the fact that it also achieves the lowest test loss. Based on Table 2, the predictions do not seem to be significantly skewed to larger or smaller values compared to the true values on average. The per bin calculation of the median and confidence interval in the left subplots of Figure 18 confirms these observations for the range of about $y_{\text{true}} = -0.5$ to $y_{\text{true}} = 3$. Below $y_{\text{true}} = -0.5$, which corresponds to events with a low number of photons $N_{\text{ph}}^{\text{tot}}$ in the veto region, the predictions are skewed to comparably higher values. Independent of the binning approach, the model seems to overestimate y_{pred} in that range. Above $y_{\text{true}} = 3$ there seems to be an underestimation of y_{pred} instead. The reason for this underestimation might be that there are fewer training samples above $y_{\text{true}} = 3$ compared to other regions along the x-axis. Therefore, many events above $y_{\text{true}} = 3$ might end up in the same branch of the decision tree and have similar predictions across the range of $y_{\text{true}} > 3$. The origin of the overestimation below $y_{\text{true}} = -0.5$ is discussed below.

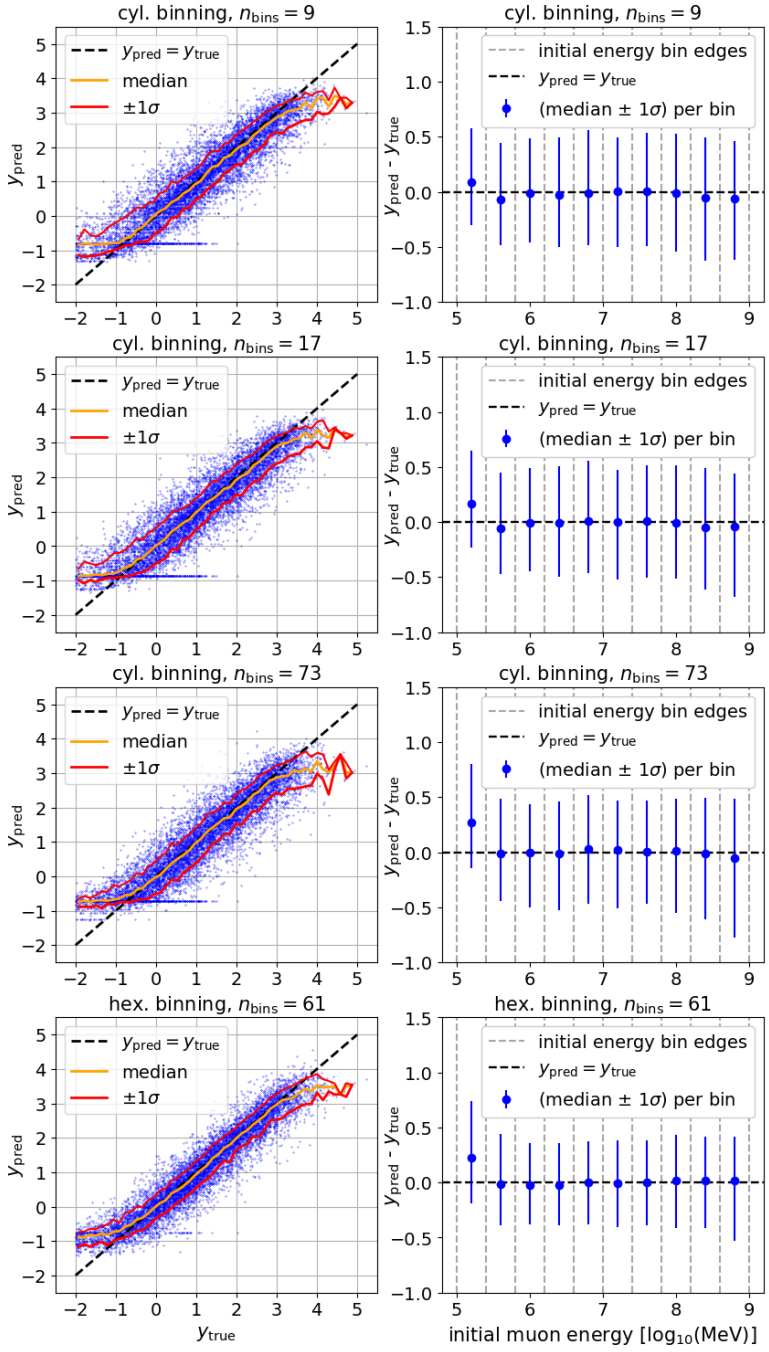

Figure 18: **Hyperparameters:** $MI = 1000$, $LR = 0.1$, $MD = 20$. **Left:** Predictions y_{pred} against respective true values y_{true} of $\log_{10}(N_{\text{ph}}^{\text{tot}})$ based on the test set (blue data points). The red lines show the 68 % confidence interval of the residuals with respect to the black dashed line. The orange line shows the corresponding median of the residuals. Median and confidence interval were calculated per bin based on 50 bins along the x-axis. **Right:** Residuals against binned initial muon energy. **Top to bottom:** Plots with respect to cylindrical grid with 9, 17 and 73 bins and hexagonal grid with 61 bins.

Based on Equation 3.1, low values of $N_{\text{ph}}^{\text{tot}}$ arise if a muon propagates at a large distance to the veto region, or if it has a low energy deposition within the veto region, or if both cases apply. All those cases yield input vectors where most of the entries are 0 or close to 0. These kinds of input vectors seem to be problematic during training and most probably are the origin of the overestimation of y_{pred} below values of $y_{\text{true}} = -0.5$. The subplots in the right column of Figure 18 are consistent with that consideration. The only energy bin that shows a significant shift of the median of the residuals to positive values across all binning approaches is the leftmost energy bin. This shows that there is an overlap between events with a low number of photons in the veto region and events with a low initial energy. If a muon has a low initial energy, it tends to decay after a relatively low number of energy losses. Therefore, the overall energy deposition within the veto region is smaller for those events. This leads to input vectors where most of the entries are zero or close to zero. The subplots in the right column of Figure 18 also show that the hexagonal binning approach exhibits the smallest 1σ spread of the residuals around the median, across different initial muon energies.

Besides the analysis of different binning approaches, training a regression model on ToyCube data can be used to get a first impression of the classification of muon events. As described at the beginning of section 3, a classification model is supposed to be implemented that divides muon events into two classes: The positive class, which corresponds to events that pass a certain event filter and are fully simulated, and the negative class, which corresponds to events that are rejected by that filter and therefore excluded at an early stage of the IceCube simulation.[9] In the context of the regression model, an event filter (compare section 2.1.3) can be mimicked by introducing a photon threshold with respect to the number of photons $N_{\text{ph}}^{\text{tot}}$ detected in the IceCube veto region. If a muon event produces $N_{\text{ph}}^{\text{tot}} \leq 5$ photons, it is classified as a positive event. Events with $N_{\text{ph}}^{\text{tot}} > 5$ are classified as negative events. As Figure 18 shows, the regression model does not make perfect predictions. This leads to four prediction categories: true negative (TN), false positive (FP), false negative (FN), and true positive (TP) as introduced in section 2.3.2. The meaning of these categories is explained in Table 3.

Figure 19 shows the left subplot of Figure 18 for the hexagonal binning approach. Solid green lines and labels of the prediction categories were added to illustrate how events are classified within the prediction space of the regression model. The four binning versions were compared with respect to the event classification by applying the photon threshold, as shown in Figure 19. The resulting classification rates of the prediction categories, normalized with respect to the total number of events N in the test set, can be seen in Figure 20. Furthermore the corresponding values of accuracy, error rate, R_{pos}, and R_{neg} as defined in section 2.3.2 are listed in Table 4.

Independent of the binning approach, an accuracy of $> 89\,\%$ was observed. This proves the concept of a BDT classification model, as it is proposed at the beginning of section 3. With ratios of FP = 5.24 % and FN = 3.12 %, the most accurate classification results were found for the hexagonal binning approach. This is consistent with the prior observations of the hexagonal binning approach showing the smallest 1σ spread of the residuals and the lowest test loss.

[9] In this work "fully simulated" means a simulation up to filter level two stage within the IceCube data analysis.

prediction category	meaning
True Negative (TN)	Events with a correct prediction of being rejected by the event filter. These events are excluded at an early stage of the IceCube simulation. The more TN events, the more computational resources are saved.
False Positive (FP)	Events with a wrong prediction of passing the event filter. These events are fully simulated, although they should have been excluded at an early stage of the IceCube simulation.[9] A high amount of FP events leads to unnecessary event processing.
False Negative (FN)	Events with a wrong prediction of being rejected by the event filter. These events are excluded at an early stage of the IceCube simulation, although they should have been fully simulated.[9] A high amount of FN events leads to a loss of valid background muon simulations. This may distort the estimation of the muon background reduction efficiency.
True Positive (TP)	Events with a correct prediction of passing the event filter. These events are fully simulated and proceed to later stages of the simulation and analysis.[9]

Table 3: Explanation of the prediction categories used for muon event classification.

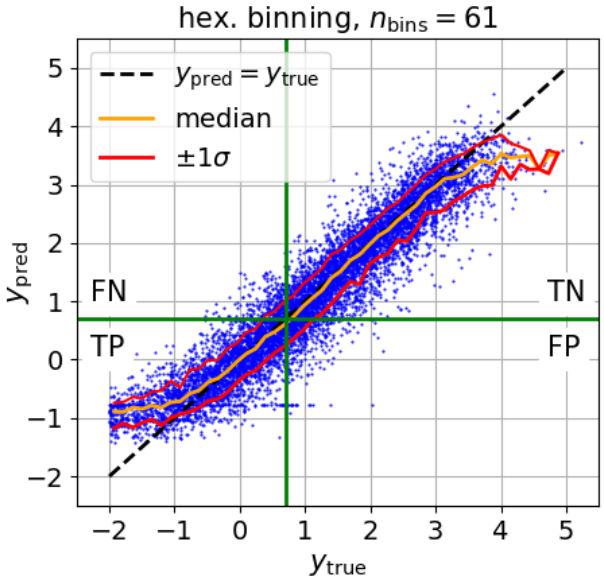

Figure 19: Classification of the muon events into prediction categories, based on the left subplot of Figure 18 for the hexagonal binning as an example. The categories are true negative (TN), false positive (FP), false negative (FN), and true positive (TP). The solid green lines mark the classification threshold of $N_{\text{ph}}^{\text{tot}} = 5$ photons.

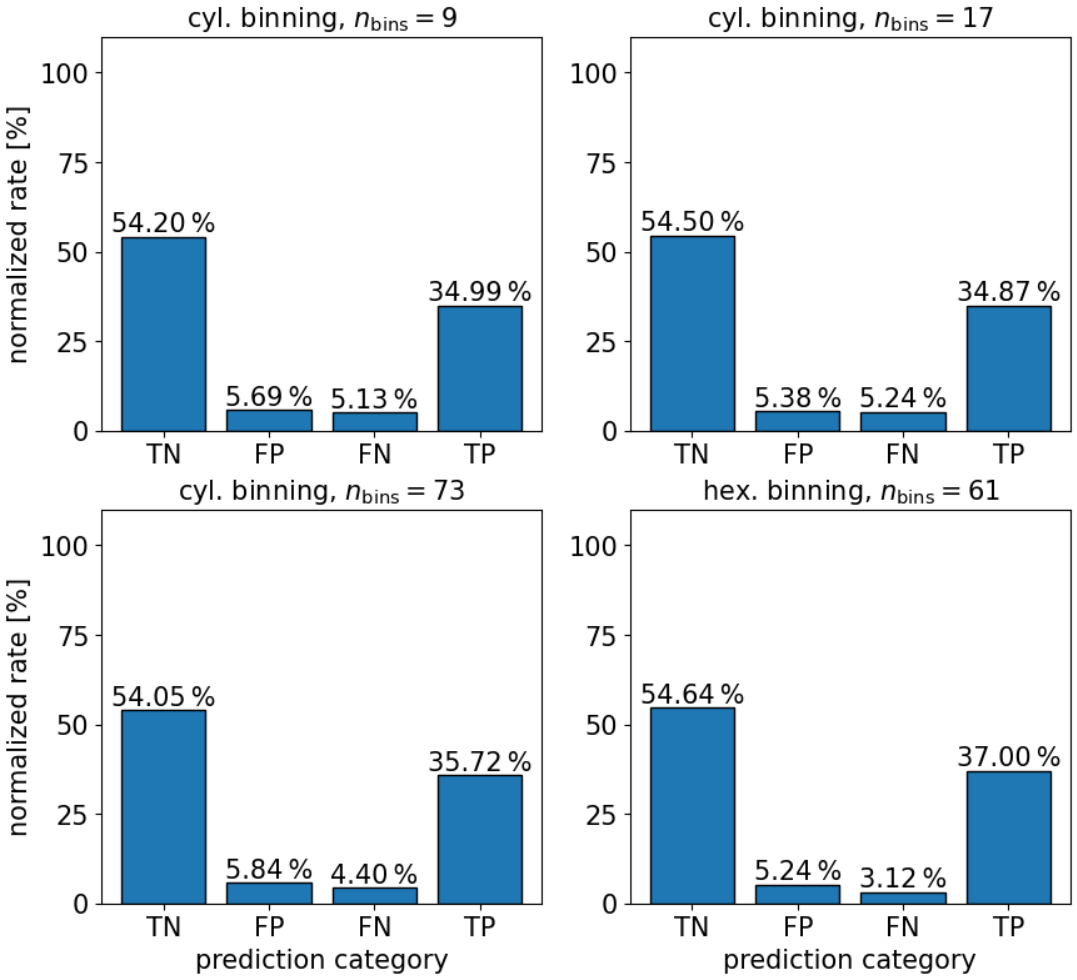

Figure 20: Classification rates normalized with respect to the total number of events in the test set. Prediction categories: true negative (TN), false positive (FP), false negative (FN), and true positive (TP). Each subplot corresponds to a certain binning grid version.

binning approach	cyl., $n_{bins} = 9$	cyl., $n_{bins} = 17$	cyl., $n_{bins} = 73$	hex., $n_{bins} = 61$
accuracy [%]	89.19	89.38	89.77	91.64
error rate [%]	10.81	10.62	10.23	8.36
R_{pos} [%]	40.68	40.25	41.56	42.24
R_{neg} [%]	59.32	59.75	58.44	57.76

Table 4: Values of accuracy, error rate, R_{pos}, and R_{neg} as defined in section 2.3.2, corresponding to the ratios of the prediction categories shown in Figure 20.

In section 3.1, the ToyCube simulation tool was introduced, which was designed to simulate muon tracks within the ice and estimate the number of photons in the IceCube veto region. Based on this toy simulation, a training dataset was generated, which was used to train a BDT regression model [8]. Hyperparameter scans were performed, varying the maximum number of boosting iterations *MI*, the learning rate *LR*, and the maximum depth of the decision trees *MD*. These scans were repeated for different methods of constructing the BDT input vector, each based on a spatial binning of the muon energy losses. In all scans, a saturation effect of the test loss was observed for a broad range of hyperparameter settings. Independent of the binning method, a saturated test loss was observed for the configuration $MI = 1000$, $LR = 0.1$, and $MD = 20$. Using this hyperparameter setting, the binning methods were compared in terms of test loss, the 1σ spread of the residuals $(y_{\text{pred}} - y_{\text{true}})$, and the results of a binary event classification. The best results were found for the hexagonal binning. Therefore, it was selected for the further development of a BDT classification model, which was trained on data from the IceCube simulation. In the following section, the optimization of this classification model is discussed in more detail.

3.2. Training with IceCube Simulation Data

The main goal of this thesis is to develop a machine learning model that can differentiate between muon events that pass the MESE event filter and events that are rejected by it, at an early stage of the IceCube simulation. The previous section 3.1.3 showed that a BDT regression model was successfully trained on data from a toy simulation. It predicts the number of photons detected in the IceCube veto region based on the muon energy losses. By applying a threshold to the predicted number of photons in the veto region, an event filter was simulated. With that, a classification of muon events with an error rate of 8.36 % was achieved.

In the next stage of development, a BDT model was trained on data generated by the IceCube simulation. In particular, parts of the muongun 22552 dataset were provided, including the energy losses and MESE filter decisions of each muon [7],[10].[10] In the IceCube simulation, event filter decisions are represented by a binary value of 0 (negative class, event did not pass the filter) or 1 (positive class, event passed the filter). Therefore, the BDT regression model was replaced by a BDT classification model, which predicts a value between 0 and 1 as output. This output can be interpreted as the probability of a sample to belong to the positive class. The underlying software is the *sklearn.ensemble.HistGradientBoostingClassifier()* class of *scikit-learn* (v1.5.2) [1],[9]. This is the classifier version of the regression model described in section 3.1.2. Both share the fast histogram-based training algorithm and similar hyperparameters. The default parameter settings of the classification model can be found in Table 18 of section A.1.3 of the appendix. The training input of the classifier is constructed in the same way as for the regression model, based on the hexagonal binning grid. The training output is given by the respective MESE filter outcome (0 or 1).

In section 3.2.1, different steps of the optimization process of the BDT classifier are described in chronological order. Previously, the test set loss was used as the main metric to measure improvements in the model predictions. For the optimization process described in section 3.2.1, a different kind of metric was used. Before going into detail on the optimization of the classifier, this metric and some related plots, which appear later within section 3.2, are introduced below.

For each muon event of a given dataset, the classifier predicts the probability of passing the MESE event filter. This leads to a certain distribution of the predicted probabilities for events that belong to the positive class (blue distribution in central plot of Figure 21) and events that belong to the negative class (orange distribution). For a perfect model, one would expect two delta peaks at zero and one, respectively. Since the classifier makes imperfect decisions, these distributions are spread out.

[10] The muon energy losses extracted from the IceCube simulation data include the following loss types: electron pair production, photo-nuclear interaction, bremsstrahlung, and production of delta electrons via ionization.

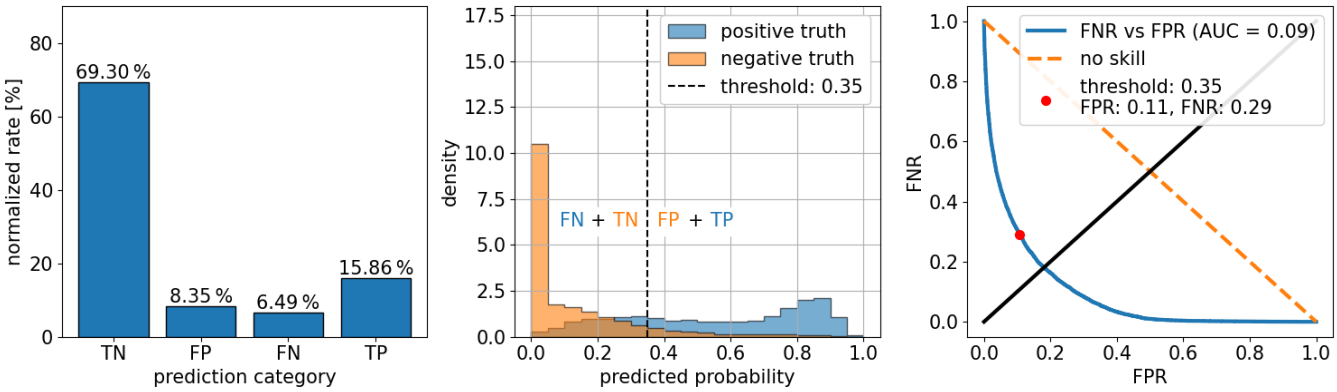

Figure 21: Example plots. **Left:** Classification rates of the prediction categories (see Table 3) normalized with respect to the total number of events of some test set. **Center:** Distribution of the predicted probabilities by the BDT classifier for actually positive (blue) and actually negative (orange) events. The black dashed line marks the probability threshold used for classification. **Right:** FNR-FPR curve, as explained in section 2.3.2. The red dot marks the location of the probability threshold along the curve corresponding to the black dashed line of the central plot.

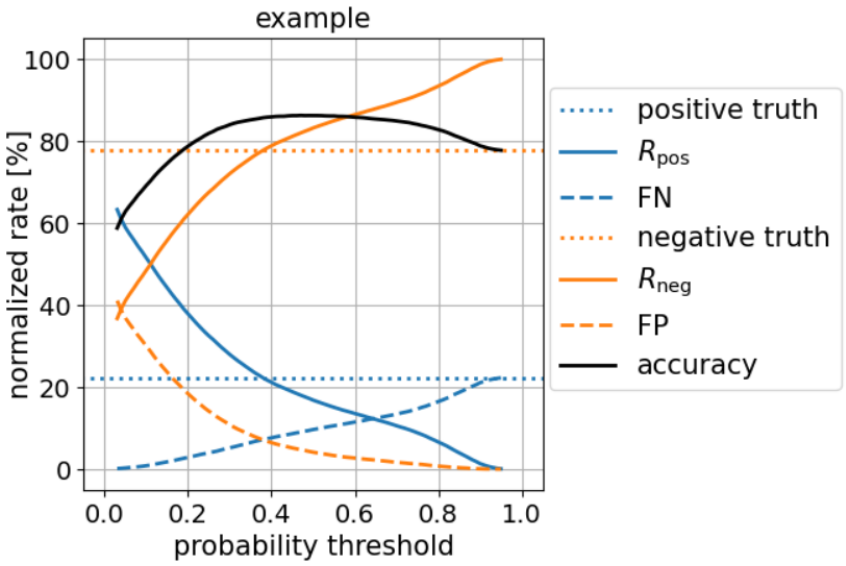

Figure 22: Example plot. Normalized rates of the prediction categories FN and FP (see Table 3), as well as R_{pos}, R_{neg}, and accuracy (see section 2.3.2) plotted against the probability threshold used for classification. The test set fractions of actually positive and actually negative events are shown as horizontal dotted lines. All rates are normalized with respect to the total number of events of the corresponding test set.

A classification is done by applying a probability threshold to the distributions (black dashed line in central plot of Figure 21) as explained for the general case in section 2.3.2. In the context of this section, all events on the left side of the threshold are excluded from the simulation (negative events), while all events on the right side of the threshold are simulated (positive events). This results in the four prediction categories as described in Table 3 (see left plot of Figure 21). While FP events only reduce the efficiency of the optimized simulation approach, FN events eventually lead to a wrong estimation of the muon background reduction efficiency. Therefore, the metric applied during the BDT optimization process is based on a FN limit. The idea is to find a probability threshold where the FN rate does not exceed a certain value, while keeping the rate of negative events R_{neg} (see Equation 2.24) as high as possible. This ensures that as many computational resources as possible are saved, while limiting the distortion of the estimated background reduction efficiency. During the BDT optimization process, improvements of the model performance are measured by comparing the results of R_{neg} for different model parameter settings, at fixed FN limits of 0.5 %, 1.0 % and 5.0 % with respect to a test dataset. Scanning FN, FP, R_{neg}, R_{pos}, and the accuracy with respect to the applied probability threshold (see Figure 22) helps to visualize the impact of certain parameter settings on the model performance. A measure of how accurate the predictions of the BDT model are, independent of the applied probability threshold, is the area under the curve (AUC) of the FNR-FPR curve (see section 2.3.2) shown in the right plot of Figure 21. The smaller the AUC value, the more accurate are the model predictions.

3.2.1. Optimization of the BDT Classifier

At the beginning of the optimization of the BDT classifier, a hyperparameter scan similar to section 3.1.3 was carried out. The hyperparameters of *max_iter* (*MI*), *learing_rate* (*LR*), and *max_depth* (*MD*) are defined equivalently as for the *scikit-learn* regression model used previously. The values applied during the scan are identical to the ones listed in Table 1. Additionally, $MD = \infty$ (unlimited decision tree depth) was added to the scanned values. Apart from that, the default loss function of the classifier is no longer given by Equation 3.5, but the log loss function. For the true values $y_{\text{true}} \in \{0, 1\}$ and classifier predictions within the interval $y_{\text{pred}} \in [0, 1]$ of a sample with size n_{samples}, this function is defined as [11]:

$$L(y_{\text{true}}, y_{\text{pred}}) = \frac{1}{n_{\text{samples}}} \cdot \sum_{i=0}^{n_{\text{samples}}} -(y_{\text{true,i}} \cdot \ln(y_{\text{pred,i}}) + (1 - y_{\text{true,i}}) \cdot \ln(1 - y_{\text{pred,i}})) \quad (3.6)$$

For the training, a dataset of $\sim 4 \cdot 10^5$ muon samples was used, based on events of the muongun dataset 22552 [10]. The fractions of training, validation, and test set were defined in the same way as described for the regression model in section 3.1.3. Furthermore, the early stopping algorithm of the classifier works the same way as mentioned for the regressor in section 3.1.3. The resulting values of the test set loss for the hyperparameter scan are shown in Figure 23. For this scan, the hexagonal binning grid was configured as previously, with $n_{\text{bins,xy}} = 61$ bins along the xy-plane.

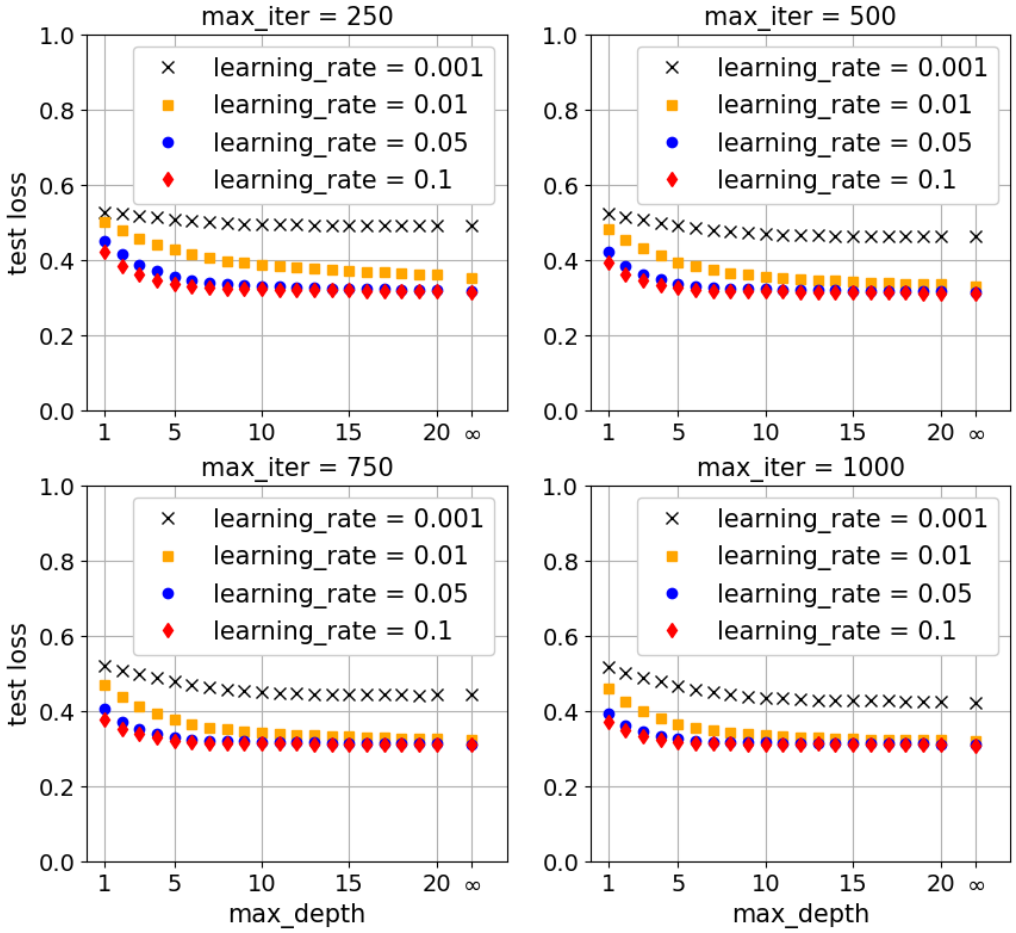

Figure 23: Calculated loss for the test set against *MD* for the hexagonal binning with $n_{\text{bins,xy}} = 61$. Each subplot shows a set of four curves and corresponds to a certain maximum boosting iteration *MI*. Each curve within a subplot corresponds to a certain value of the learning rate *LR*.

The overall behaviour of the test loss with respect to the chosen hyperparameters is consistent with the observations made for the regression model. This was expected since both the regression model and the classification model are based on similar algorithms in *scikit-learn* [8],[9]. Therefore, also for the classifier the parameter setting $MI = 1000$, $LR = 0.1$, and $MD = 20$ seemed reasonable. Additionally, it was observed that setting $MD = \infty$, results in slightly lower values of the test loss for almost every parameter configuration, except for $LR = 0.001$ combined with $MI \in \{500, 750\}$. The lowest test loss was observed for the configuration $MI = 1000$, $LR = 0.1$, and $MD = \infty$. Therefore, this setting was fixed at the beginning of the optimization process. While $MI = 1000$, $LR = 0.1$ were changed again at some point (see later in this section), *MD* was not changed anymore during the subsequent BDT optimization process.

Optimization of the Hexagonal Binning along the xy-Plane:

The number of bins along the xy-plane was set to $n_{\text{bins,xy}} = 61$ before. To check if increasing or decreasing this number has a positive effect on the value of R_{neg}, the classifier was trained at four different hexagonal binning grid configurations with $n_{\text{bins,xy}} \in \{19, 37, 127, 271\}$. The hexagon radius of each bin was set to the respective value of $r_{\text{hex}} \in \{195\,\text{m}, 140\,\text{m}, 75\,\text{m}, 52\,\text{m}\}$ instead of $r_{\text{hex}} = 108\,\text{m}$, to ensure that the resulting grid is approximately equal in size to the one shown in Figure 14. A dataset of equal size to the one used in the previous hyperparameter scan was used for training. Each resulting model was evaluated based on its predictions on the corresponding test set data. Figure 24 shows the corresponding FNR-FPR curves. One can see that a lower number of bins leads to a recognizable increase in the AUC, while a higher number of bins leads to a hardly visible decrease in the AUC. Each row of Figure 25 shows the resulting distributions of the predicted probabilities in the left subplot and the corresponding probability threshold scan in the right subplot. There are visible changes in the distributions and curves for different settings of $n_{\text{bins,xy}}$. Although these effects are relatively small, a positive impact on R_{neg} can be observed in Table 5, when $n_{\text{bins,xy}}$ is increased to 271 bins. The corresponding rates of the prediction categories, as well as the values of R_{neg}, accuracy, error rate, and probability threshold, are listed there for fixed values of FN and different settings of $n_{\text{bins,xy}}$. The positive impact of more bins is further visualized in Figure 26, where zoomed-in plots of FN and R_{neg} against the probability threshold are compared. The 1 % FN limit and the thresholds at which each FN curve intersects the limit are visualized by dashed lines. While the intersection thresholds are almost identical for each setting of $n_{\text{bins,xy}}$, the corresponding value of R_{neg} gets slightly higher at these thresholds with increasing bin number.

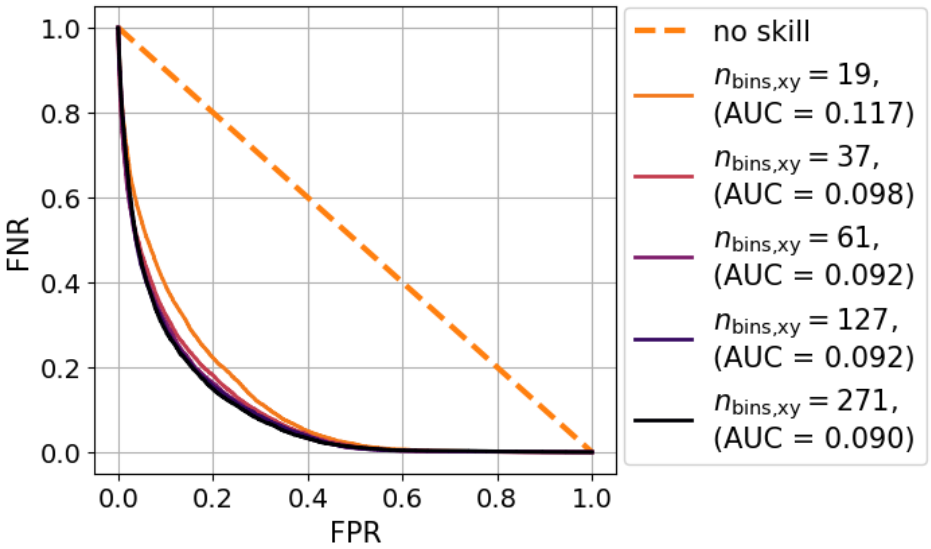

Figure 24: False negative rate (FNR) against false positive rate (FPR) with respect to the test set, for different settings of $n_{\text{bins,xy}}$. The respective area under the curve (AUC) is given in the legend.

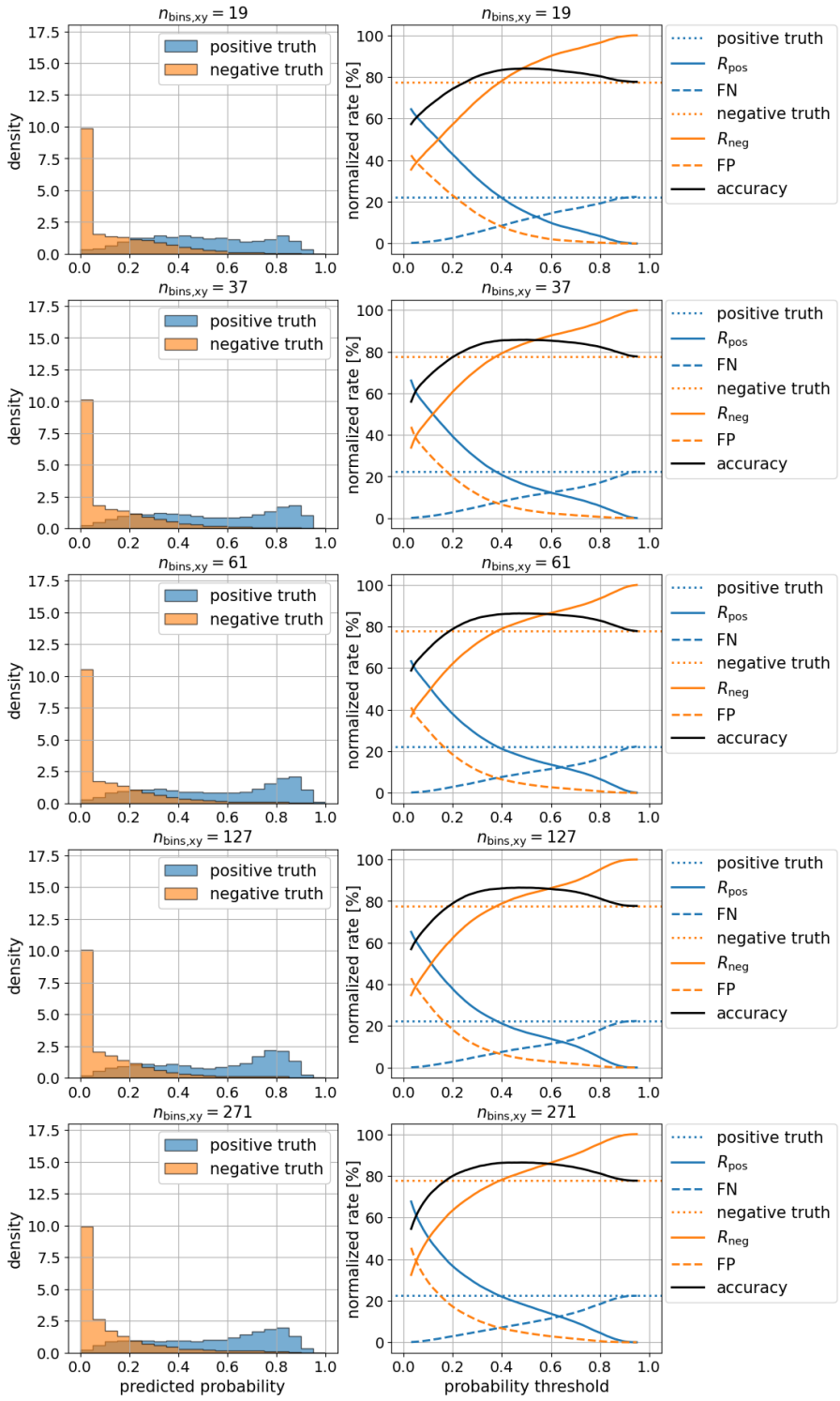

Figure 25: Each row corresponds to a different setting of n_{bins}. **Left:** Distribution of the predicted probabilities by the BDT classifier for actually positive (blue) and actually negative (orange) events. **Right:** Probability threshold scans as explained for Figure 22.

n_{bins} = 19	R_{neg} [%]	TN [%]	FP [%]	TP [%]	accuracy [%]	error rate [%]	threshold
FN = 0.50 %	40.82	40.32	37.32	21.85	62.18	37.82	0.065
FN = 1.00 %	46.86	45.86	31.79	21.35	67.21	32.79	0.115
FN = 5.00 %	67.23	62.23	15.41	17.35	79.59	20.41	0.284
n_{bins} = 37							
FN = 0.50 %	42.75	42.26	35.39	21.85	64.11	35.89	0.068
FN = 1.00 %	48.10	47.11	30.54	21.35	68.46	31.54	0.105
FN = 5.00 %	70.15	65.15	12.50	17.36	82.51	17.49	0.286
n_{bins} = 61							
FN = 0.50 %	44.11	43.61	34.04	21.85	65.46	34.54	0.070
FN = 1.00 %	49.81	48.81	28.84	21.35	70.16	29.84	0.109
FN = 5.00 %	71.26	66.26	11.39	17.35	83.61	16.39	0.291
n_{bins} = 127							
FN = 0.50 %	43.93	43.43	34.22	21.85	65.29	34.71	0.076
FN = 1.00 %	48.99	47.99	29.65	21.35	69.35	30.65	0.107
FN = 5.00 %	71.67	66.67	10.97	17.35	84.03	15.97	0.290
n_{bins} = 271							
FN = 0.50 %	44.65	44.15	33.49	21.85	66.02	33.98	0.074
FN = 1.00 %	50.44	49.44	28.21	21.35	70.80	29.20	0.105
FN = 5.00 %	71.92	66.93	10.72	17.36	84.28	15.72	0.298

Table 5: Rates of the prediction categories as well as R_{neg}, accuracy, error rate, and probability threshold for fixed FN limits and different settings of $n_{\text{bins,xy}}$. The rates were normalized with the total number of samples $N_{\text{test}} = 40312$ in the test set.

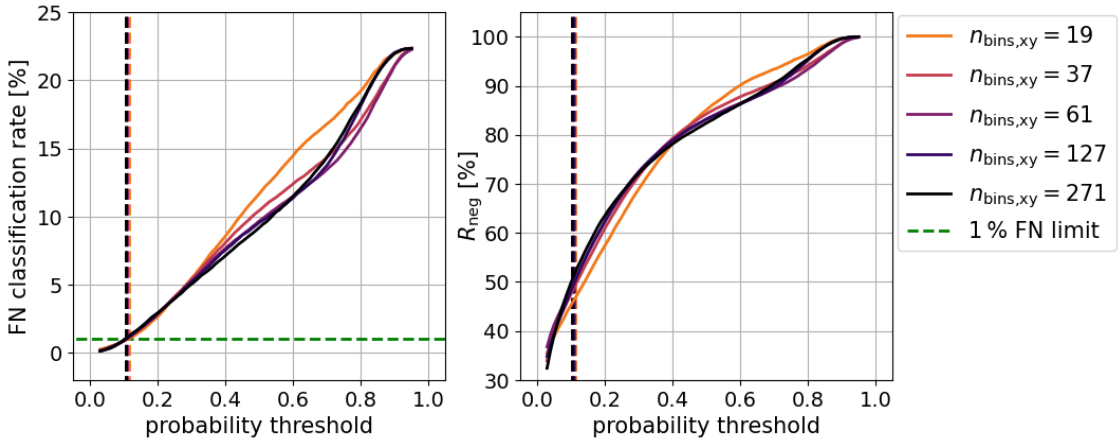

Figure 26: Probability threshold scans of FN (left) and R_{neg} (right). Each curve corresponds to a certain setting of n_{bins}. The green dashed horizontal line on the left marks the 1 % FN limit. The vertical dashed lines show the probability thresholds at which the respective FN curve reaches the FN limit.

The number of bins is fixed at $n_{bins,xy} = 271$ for subsequent optimization results, since the best results of R_{neg} across all FN limits were observed for this setting. Increasing the number of bins even further seemed unreasonable, due to the increase in memory requirements during computation and data storage, and the comparatively small expected improvement of R_{neg}.

Adjustment of Class Weights:

The plots of the left column of Figure 25 show that events of the negative class (orange distributions) have more accurate predictions than events of the positive class (blue distributions). This is probably because there is an imbalance in the number of events belonging to these classes. For the muongun 22552 dataset, about 78 % of the events are rejected by the MESE filter, while the remaining 22 % pass the filter. This underrepresentation of the positive event class seems to have an impact on the training. Therefore, several class weight settings were applied during training in order to counteract this imbalance. For the *scikit-learn* default setting, a class weight of one is applied to each class. Besides that, the *class_weight* parameter can be set to "balanced", which applies weights indirectly proportional to the number of events in the respective class. Another option is to specify the weight setting directly. Figure 27 shows the predicted probability distributions and threshold scans for default weights, balanced weights, and an example of a positively skewed weight setting. Positively skewed means that a weight of one was applied to the negative class and a weight of ten to the positive class.

One can see that the class weight has a strong effect on the resulting distributions, which are shifted to the right for the balanced and the positively skewed weight settings. Consequently, this shift also has an impact on the probability threshold scans. Since the distributions are shifted to the right for both event classes, this does not result in a significant improvement of R_{neg}.

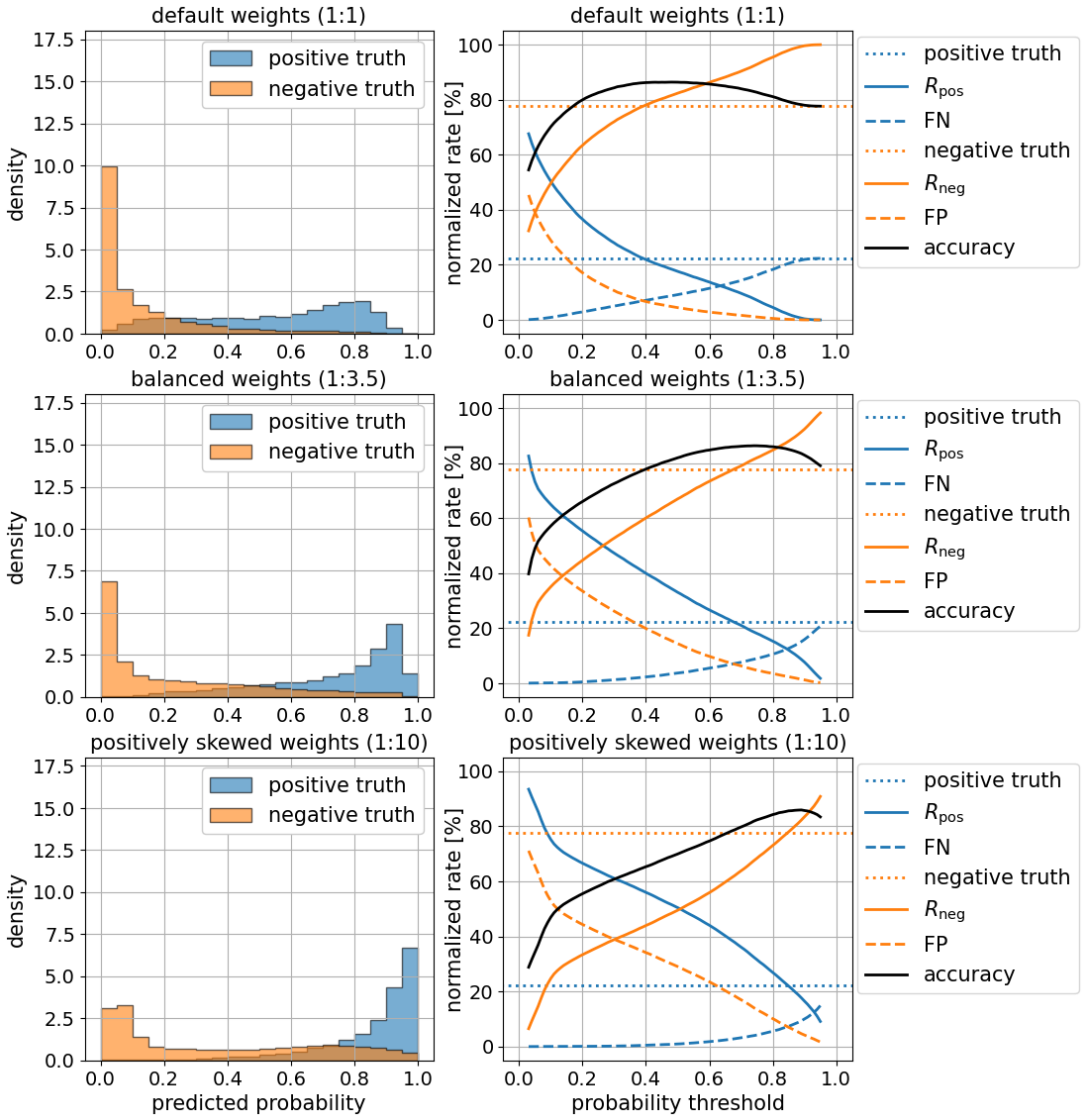

Figure 27: Each row corresponds to a different setting of the class weight. **Top to bottom:** default weights, balanced weights, positively skewed weights. **Left:** Distribution of the predicted probabilities by the BDT classifier for actually positive (blue) and actually negative (orange) events. **Right:** Probability threshold scans as explained for Figure 22.

Figure 28 illustrates this by showing vertical dashed lines at the thresholds where the respective FN curves reach the FN = 1 % limit. A shift of the distribution of the positive class to the right leads to a shift of these thresholds to the right. Since the distribution of the negative class is also shifted, the change of the thresholds has only a minor impact on R_{neg}. This behaviour was observed independent of how the FN limit was set. Table 6 shows the previously introduced rates for different FN limits and class weight settings. For balanced weights, R_{neg} has slightly lower values, but the difference to default weights is marginal. For the positively skewed weight setting, the results of R_{neg} get comparatively worse. Since it was preferred to have an unbiased learning process with respect to the sample classes, balanced weights were applied for ongoing optimization steps.

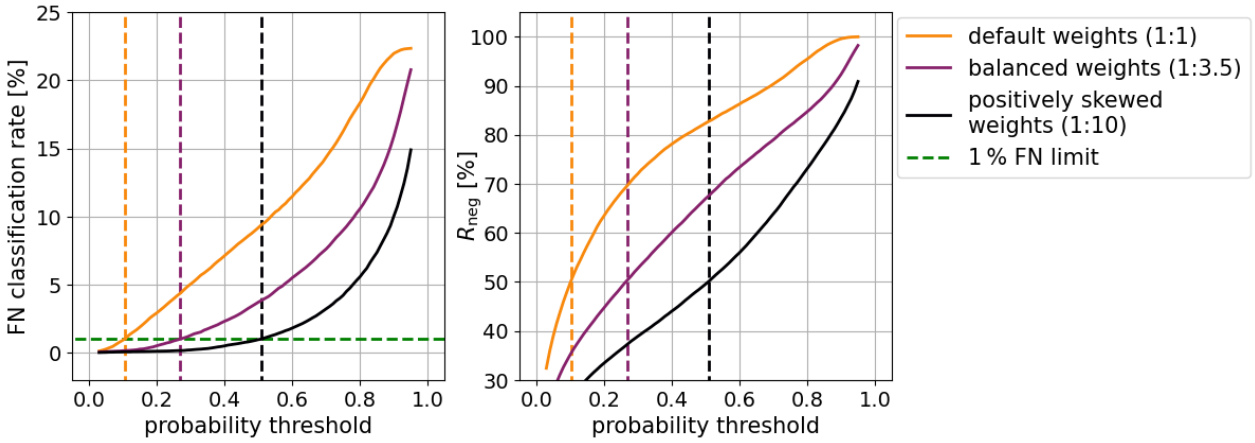

Figure 28: Probability threshold scans of FN (left) and R_{neg} (right). Each curve corresponds to a certain setting of the class weights. The green dashed horizontal line on the left marks the 1 % FN limit. The vertical dashed lines show the probability thresholds at which the respective FN curve reaches the FN limit.

default weights	R_{neg} [%]	TN [%]	FP [%]	TP [%]	accuracy [%]	error rate [%]	threshold
FN = 0.50 %	44.65	44.15	33.49	21.85	66.02	33.98	0.074
FN = 1.00 %	50.44	49.44	28.21	21.35	70.80	29.20	0.105
FN = 5.00 %	71.92	66.93	10.72	17.36	84.28	15.72	0.298
balanced weights							
FN = 0.50 %	44.62	44.12	33.53	21.85	65.98	34.02	0.200
FN = 1.00 %	50.37	49.37	28.28	21.35	70.72	29.28	0.270
FN = 5.00 %	71.82	66.82	10.82	17.35	84.18	15.82	0.575
pos. skewed weights							
FN = 0.50 %	44.04	43.54	34.10	21.85	65.40	34.60	0.401
FN = 1.00 %	50.05	49.05	28.60	21.35	70.40	29.60	0.509
FN = 5.00 %	71.09	66.09	11.56	17.35	83.44	16.56	0.779

Table 6: Rates of the prediction categories as well as R_{neg}, accuracy, error rate, and probability threshold for fixed FN limits and different settings of the class weight. The rates were normalized with the total number of samples $N_{\text{test}} = 40312$ in the test set.

Number of Training Samples and Optimization of the internal BDT Structure:

As mentioned earlier in this section, a dataset of $\sim 4 \cdot 10^5$ muon samples was used for training. The actual training set involved $N_{\text{train}} \approx 3.3 \cdot 10^5$ samples, because the validation and test sets were split from the total dataset. In order to find out if increasing the number of training samples leads to higher values of R_{neg}, the classifier was trained on a training set of $N_{\text{train}} \approx 6.5 \cdot 10^5$ samples. It was observed that due to the larger training set, more boosting iterations were applied during training. Therefore, the maximum number of boosting iterations was increased from $MI = 1000$ to $MI = 2000$ for the larger training set, to ensure that the training proceeds until early stopping is triggered. The upper two rows of Figure 29 show the corresponding predicted probability distributions and threshold scans for a training with $N_{\text{train}} \approx 3.3 \cdot 10^5$ and $N_{\text{train}} \approx 6.5 \cdot 10^5$. One can see that the distribution of the positive class (blue) is slightly shifted towards one, and the distribution of the negative class (orange) is slightly shifted towards zero for $N_{\text{train}} \approx 6.5 \cdot 10^5$. This leads to a small improvement of R_{neg} across all three FN limits, as the values listed in Table 7 show.

Additionally, the internal BDT structure was optimized based on the training set with $N_{\text{train}} \approx 6.5 \cdot 10^5$ samples, by adjusting two further hyperparameters. The first one is called *max_leaf_nodes* (*MLN*). For each boosting iteration, this parameter sets the maximum number of leaves of the added decision tree. The second parameter is called *min_samples_leaf* (*MSL*) and sets the minimum number of samples that can be assigned per leaf [9]. Together with the maximum tree depth (MD), which was introduced in section 3.1.3, these are the main parameters to control the internal structure of the BDT (see also section 2.3.1). The maximum depth was fixed to $MD = \infty$ (unlimited depth), as mentioned earlier in this section, while the other two parameters were configured using their default values of $MLN = 31$ and $MSL = 20$. Scanning all possible combinations of *MD*, *MLN*, and *MSL* to optimize the internal BDT structure did not seem feasible. Instead, the parameter space was restricted by setting $MD = MLN = \infty$. In that way, the depth and number of leaf nodes of the decision tree of each boosting iteration are only constrained by the *MSL* parameter. As a consequence, the complexity of the internal BDT structure can be varied using only *MSL*, which simplifies the hyperparameter scan. To ensure reproducibility of the BDT model, it is necessary to calculate the ratio $R_{MSL} = N_{\text{train}}/MSL$. If a reproduced model has to be trained on a dataset of size $N_{\text{train}}^{\text{new}}$ in the future, MSL_{new} has to be adjusted so that $R_{MSL}^{\text{new}} = R_{MSL}$. Otherwise, the internal BDT structure will differ from the original model.

Figure 30 shows the results of a parameter scan of *MSL* based on the datatset with $N_{\text{train}} \approx 6.5 \cdot 10^5$. The highest R_{neg} score for FN = 5.00 % as well as the second highest scores for FN = 0.50 % and FN = 1.00 % were achieved at $MSL = 1000$, which corresponds to $R_{MSL} \approx 653$. At higher values of *MSL*, R_{neg} decreases. This is probably due to the decision trees becoming too shallow. Consequently, the BDT is unable to process the information provided by the training input precisely enough. At lower values of *MSL*, a decrease of R_{neg} was observed as well. This is most probably related to overfitting to the training data, due to overly complex decision trees.

The corresponding evaluation plots after the optimization of *MSL* are shown in the bottom row of Figure 29. The peaks of the predicted probability distributions are further shifted towards zero for the negative class and one for the positive class, compared to

before the optimization. The FN and FP curves of the probability threshold scan tend to reach smaller values overall. The resulting R_{neg} scores were increased by the optimized *MSL* setting across all FN limits (see Table 7). The decrease of the AUC in the FNR-FPR curves of Figure 31 further proves that using $N_{\text{train}} \approx 6.5 \cdot 10^5$ combined with the parameter settings of $MD = MLN = \infty$ and $MSL = 1000$ is a valid choice for improving the performance of the classifier. Therefore, these settings were applied during further model optimizations.

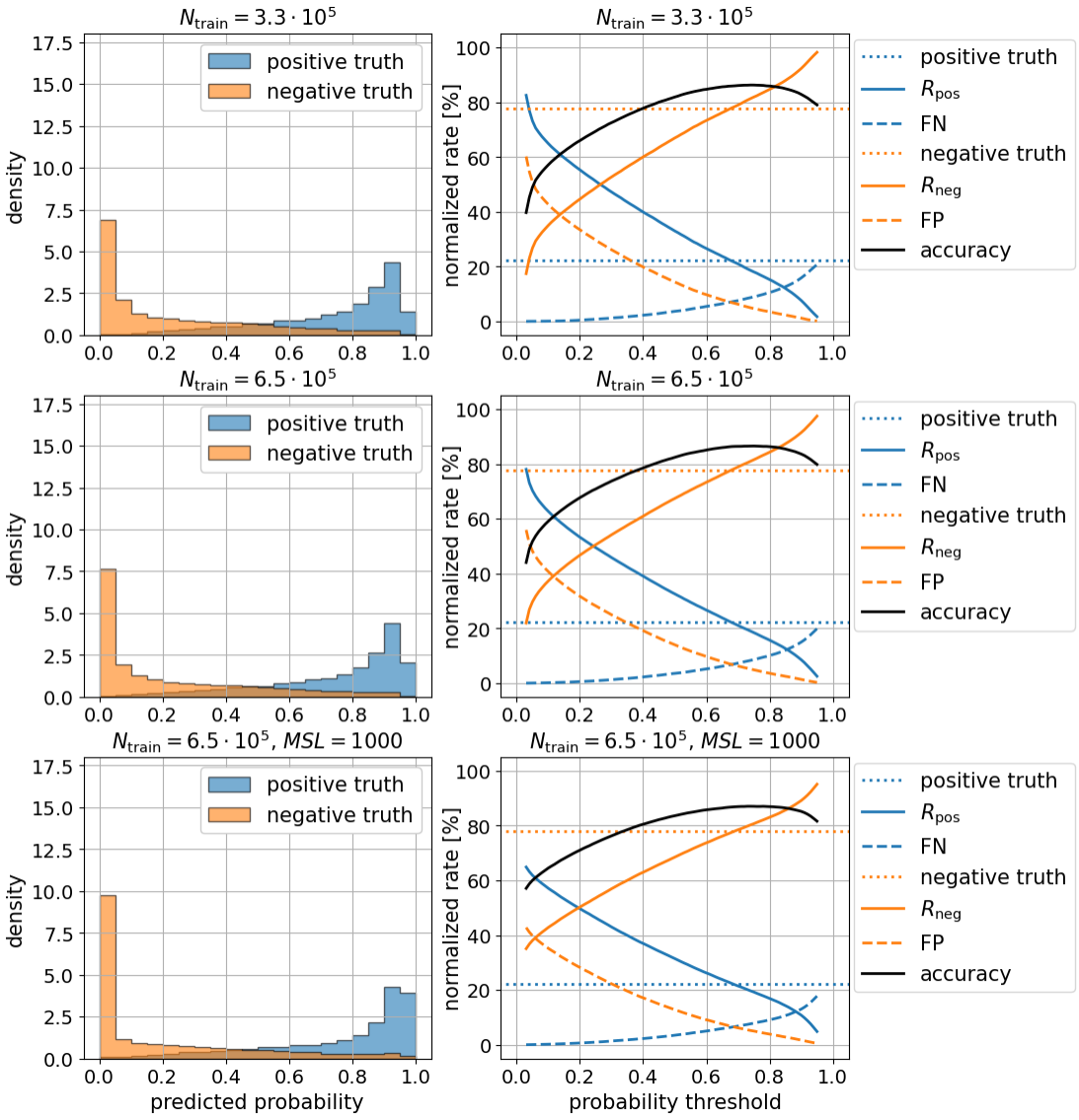

Figure 29: **Left:** Distribution of the predicted probabilities by the BDT classifier for actually positive (blue) and actually negative (orange) events. **Right:** Probability threshold scans as explained for Figure 22. **Top to bottom:** $N_{\text{train}} \approx 3.3 \cdot 10^5$, $N_{\text{train}} \approx 6.5 \cdot 10^5$, $N_{\text{train}} \approx 6.5 \cdot 10^5$ with optimized BDT structure at $MSL = 1000$.

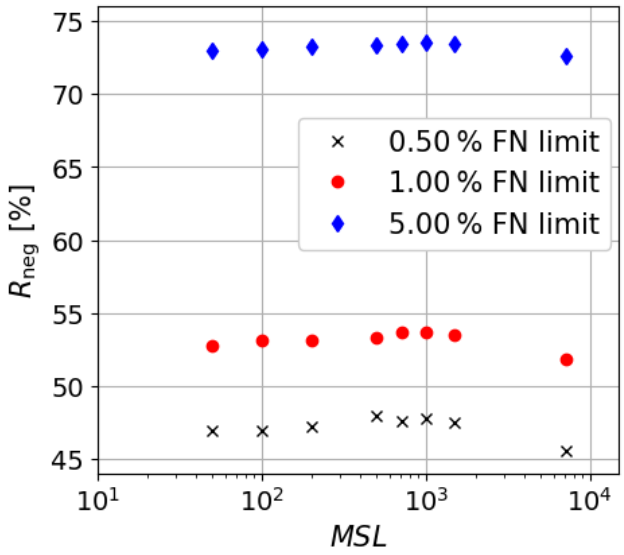

Figure 30: R_{neg} score against the *MSL* hyperparameter for different FN limits.

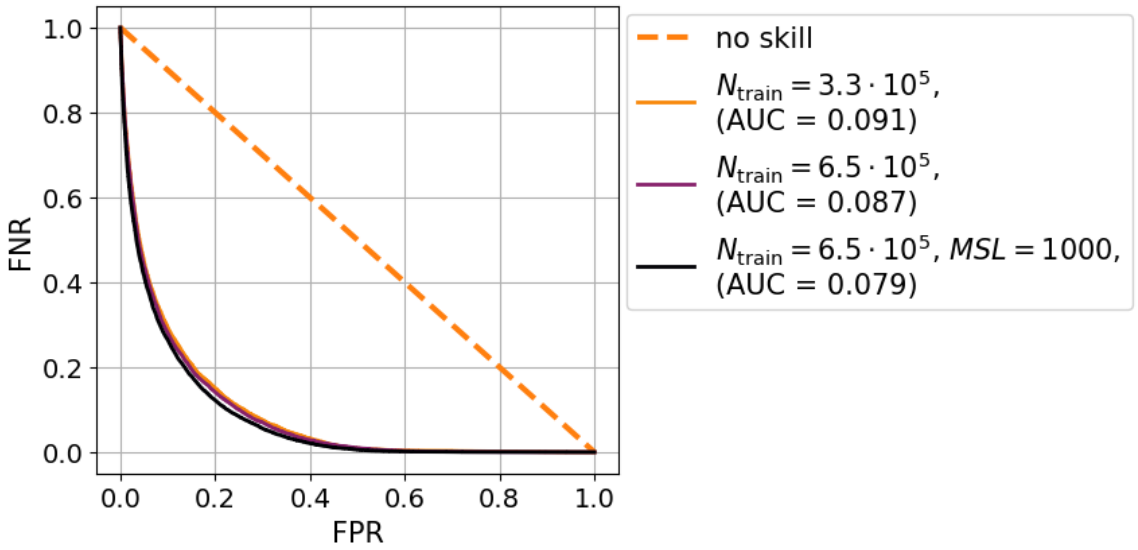

Figure 31: False negative rate (FNR) against false positive rate (FPR) with respect to the test set, for different settings of N_{train} and the optimized *MSL* setting. The respective area under the curve (AUC) is given in the legend.

$N_{\text{train}} \approx 3.3 \cdot 10^5$	R_{neg} [%]	TN [%]	FP [%]	TP [%]	accuracy [%]	error rate [%]	threshold
FN = 0.50 %	44.62	44.12	33.53	21.85	65.98	34.02	0.200
FN = 1.00 %	50.37	49.37	28.28	21.35	70.72	29.28	0.270
FN = 5.00 %	71.82	66.82	10.82	17.35	84.18	15.82	0.575
$N_{\text{train}} \approx 6.5 \cdot 10^5$							
FN = 0.50 %	45.27	44.77	33.08	21.65	66.43	33.57	0.183
FN = 1.00 %	51.68	50.68	27.17	21.15	71.83	28.17	0.265
FN = 5.00 %	72.63	67.63	10.12	17.16	84.79	15.21	0.588
$N_{\text{train}} \approx 6.5 \cdot 10^5$, $MSL = 1000$							
FN = 0.50 %	47.76	47.26	30.59	21.65	68.92	31.08	0.164
FN = 1.00 %	53.64	52.64	25.20	21.15	73.80	26.20	0.250
FN = 5.00 %	73.57	68.57	9.28	17.15	85.72	14.28	0.596

Table 7: Rates of the prediction categories as well as R_{neg}, accuracy, error rate, and probability threshold for fixed FN limits and different settings of the number of training samples N_{train} and the internal BDT structure. The rates were normalized with the total number of samples N_{test} in the test set ($N_{\text{test}} = 40312$ for $N_{\text{train}} \approx 3.3 \cdot 10^5$, $N_{\text{test}} = 80584$ for $N_{\text{train}} \approx 6.5 \cdot 10^5$).

Optimization of the Training Input:

The evaluation of the regression model in section 3.1.3 showed that the way information is encoded in the input vector has a major impact on the performance of the BDT model. Therefore, several adjustments were made to the input vector to provide more information on the muon events during training. After optimizing the number of bins $n_{\text{bins,xy}}$ along the xy-plane earlier in this section, the input vector of each muon was fixed to have 271 entries. As explained in section 3.1.2, each entry represents the summed muon energy losses within a certain hexagonal bin of the binning grid. Each of these bins reaches from −600 m to 600 m along the z-axis. To provide more information on the location of the energy losses along this axis, a subdivision of the bins along z was introduced. The number of bins along the z-axis is called $n_{\text{bins,z}}$. Furthermore, six additional muon quantities were extracted from the IceCube simulation data and appended to the input vector (see Table 8). This resulted in a new input vector with $(271 \cdot n_{\text{bins,z}} + 6)$ entries. The new binning grid with equidistant bins along the z-axis is illustrated in Figure 33 for $n_{\text{bins,z}} = 2$ as an example. The BDT model was trained based on the new input vector for the settings of $n_{\text{bins,z}} \in \{2, 4, 8\}$. In Figure 32 the results are compared to the model trained with the old input vector (without binning in z ($n_{\text{bins,z}} = 1$) and without added properties). Using the new input vector leads to sharper peaks of the predicted probability distributions at zero and one for the respective event class. The probability threshold scans also show an overall flattening of the FN and FP curves, resulting in a higher accuracy of the model across a broad range of thresholds.

added property	explanation
initial energy [$\log_{10}$(GeV)]	Energy of the muon at the beginning of its simulated track within the ice. Since this value potentially covers multiple orders of magnitude, the logarithm is applied.
initial zenith [rad]	Initial zenith angle $\in [0, \pi]$ of the muon. Together with the initial azimuth angle, this angle defines the propagation direction of the muon within the ice.
initial azimuth [rad]	Initial azimuth angle $\in [0, 2\pi[$ of the muon.
initial θ [rad]	The initial position of the muon within the IceCube coordinate system can be described in spherical coordinates, where $\theta \in [0, \pi]$ is the angle between the position vector and the z-axis. In the MuonGun generator of the IceCube simulation, muons are injected on a two-dimensional cylinder surface (see section 2.2). Therefore, initial θ and initial ϕ are sufficient to encode all information on the muon initial position.
initial ϕ [rad]	The initial position of the muon within the IceCube coordinate system can be described in spherical coordinates, where $\phi \in [0, 2\pi[$ is the angle between the projection of the position vector on the xy-plane and the x-axis.
total deposited energy [$\log_{10}$(GeV)]	Total deposited energy of the muon via energy losses within the binning grid. Since this value potentially covers multiple orders of magnitude, the logarithm is applied.

Table 8: Explanation of the muon quantities that were appended as additional entries to the training input vector.

These effects were observed independent of whether two, four, or 8 bins were used along the z-axis. Table 9 shows that the R_{neg} score is strongly improved by applying the new input vector with $n_{\text{bins,z}} = 2$. Increasing $n_{\text{bins,z}}$ to four or eight did not lead to a further improvement. Instead, a tendency to slightly lower values of R_{neg} was found compared to $n_{\text{bins,z}} = 2$. A consistent behaviour was observed for the AUC values in Figure 34, which show the lowest values for $n_{\text{bins,z}} = 2$ and $n_{\text{bins,z}} = 4$. Overall, using the new input vector with a setting of $n_{\text{bins,z}} = 2$ was considered the best choice for maximizing the R_{neg} score at given FN limits. Therefore, this setting was applied for subsequent optimization steps.

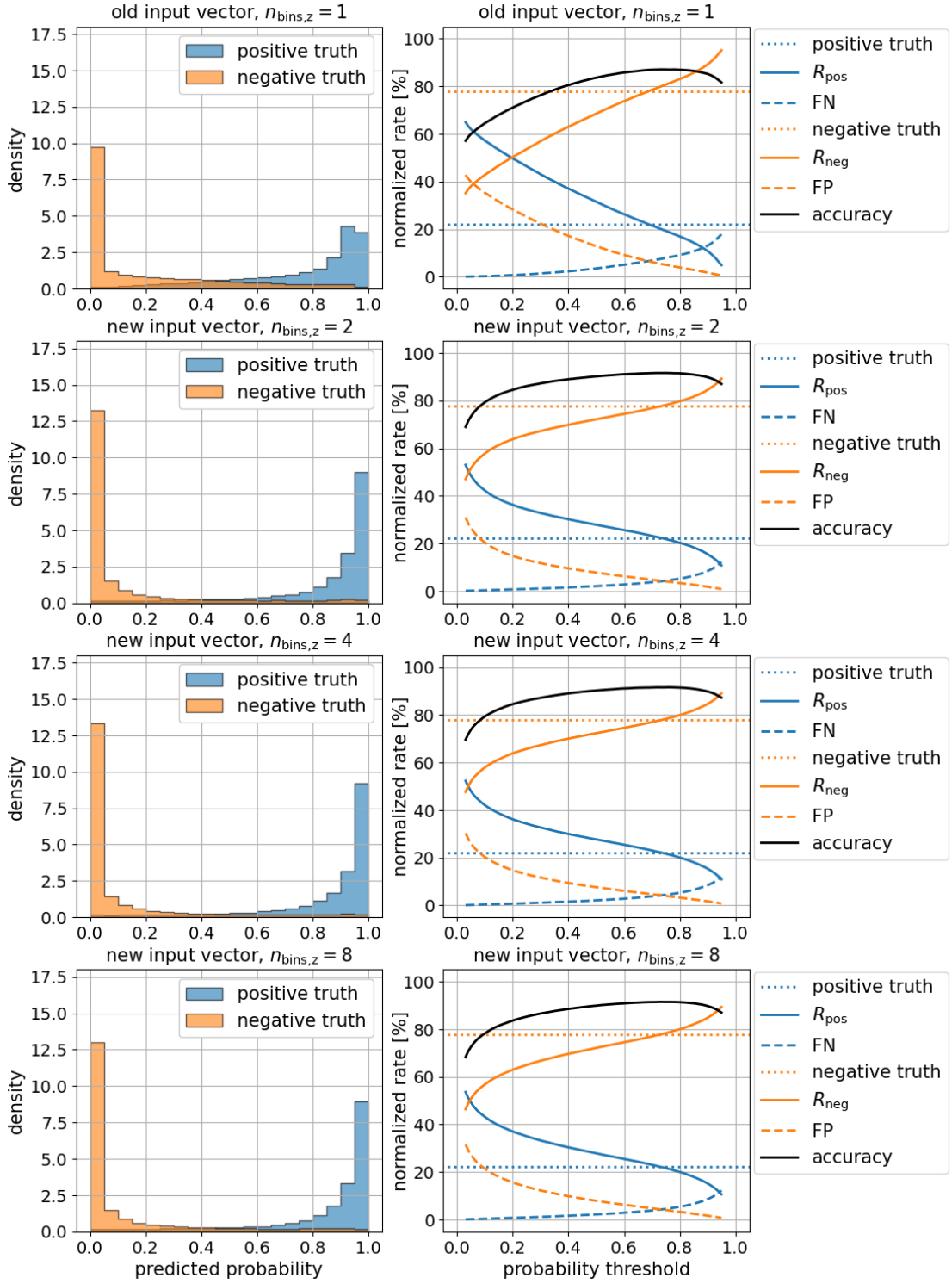

Figure 32: **Left:** Distribution of the predicted probabilities by the BDT classifier for actually positive (blue) and actually negative (orange) events. **Right:** Probability threshold scans as explained for Figure 22. Each row corresponds to a different version of the BDT input vector.

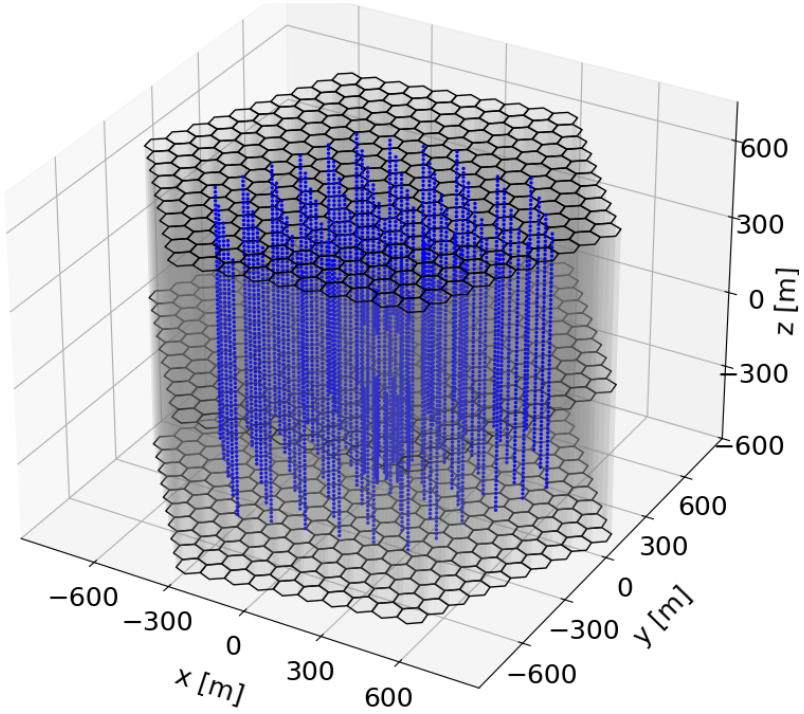

Figure 33: New version of the hexagonal binning grid with $n_{\text{bins,xy}} = 271$ and $n_{\text{bins,z}} = 2$. This leads to 542 bins in total. The blue dots visualize the DOMs of the IceCube detector. Rotated variations of this plot can be found in section A.1.3 of the appendix.

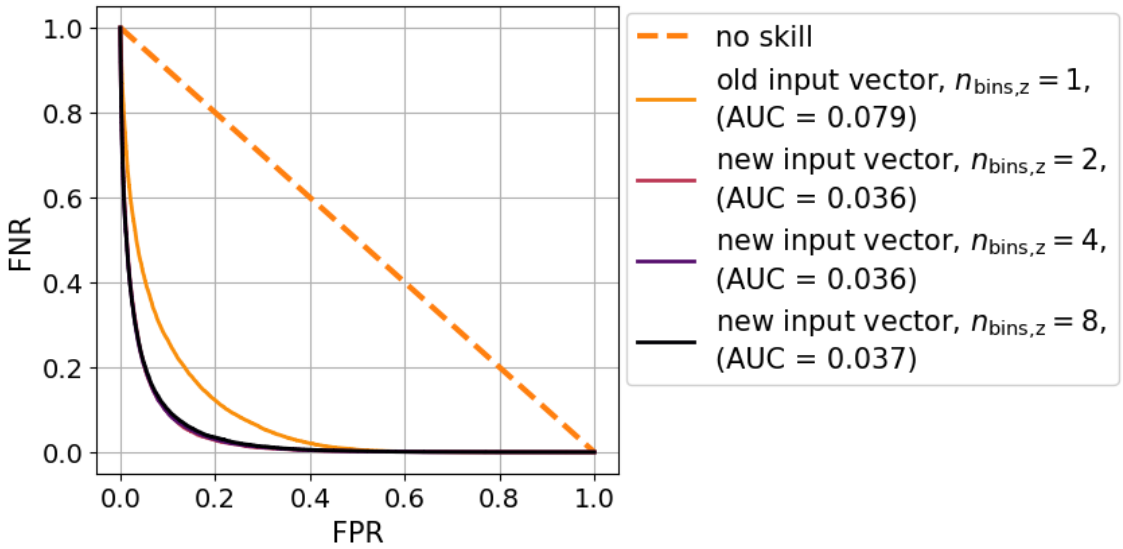

Figure 34: False negative rate (FNR) against false positive rate (FPR) with respect to the test set, for different versions of the BDT input vector. The respective area under the curve (AUC) is given in the legend.

old input vector, $n_{\text{bins,z}} = 1$	R_{neg} [%]	TN [%]	FP [%]	TP [%]	accuracy [%]	error rate [%]	threshold
FN = 0.50 %	47.76	47.26	30.59	21.65	68.92	31.08	0.164
FN = 1.00 %	53.64	52.64	25.20	21.15	73.80	26.20	0.250
FN = 5.00 %	73.57	68.57	9.28	17.15	85.72	14.28	0.596
new input vector, $n_{\text{bins,z}} = 2$							
FN = 0.50 %	61.02	60.52	17.33	21.65	82.18	17.82	0.144
FN = 1.00 %	66.65	65.65	12.20	21.15	86.80	13.20	0.281
FN = 5.00 %	79.32	74.32	3.52	17.15	91.47	8.53	0.788
new input vector, $n_{\text{bins,z}} = 4$							
FN = 0.50 %	60.60	60.10	17.75	21.65	81.76	18.24	0.137
FN = 1.00 %	66.25	65.25	12.60	21.15	86.41	13.59	0.265
FN = 5.00 %	79.34	74.34	3.50	17.15	91.50	8.50	0.784
new input vector, $n_{\text{bins,z}} = 8$							
FN = 0.50 %	59.41	58.91	18.94	21.65	80.57	19.43	0.133
FN = 1.00 %	65.51	64.52	13.33	21.15	85.67	14.33	0.264
FN = 5.00 %	79.27	74.27	3.58	17.15	91.42	8.58	0.779

Table 9: Rates of the prediction categories as well as R_{neg}, accuracy, error rate, and probability threshold for fixed FN limits and different versions of the BDT input vector. The rates were normalized with the total number of samples $N_{\text{test}} = 80584$ in the test set.

Final Parameter Scan of *LR* and *MSL*:

At the end of the optimization process, two further scans of the previously introduced hyperparameters *LR* and *MSL* were performed. This was done to check if the changes in the input vector had any impact on the optimal setting for those parameters. The values involved in the scans are listed in Table 10. Since both parameters have an influence on how many boosting iterations are applied during the training, the maximum boosting iteration was increased to higher values than $MI = 2000$ for some of the combinations (see Table 19 in section A.1.3 of the appendix). This was done to ensure that the training proceeds until the early stopping algorithm is triggered, and not until the maximum boosting iteration is reached. Otherwise, a comparison between the scanned parameter combinations might become unfair.

hyperparameter	applied values
learning_rate (*LR*), (first scan)	$[1 \cdot 10^{-2}, 2.5 \cdot 10^{-2}, 5 \cdot 10^{-2}, 1 \cdot 10^{-1}, 2 \cdot 10^{-1}]$
min_samples_leaf (*MSL*), (first scan)	$[5, 50, 100, 250, 500, 750, 1000, 1250, 1500, 1750, 2000, 5000, 10000]$
learning_rate (*LR*), (second scan)	$[1 \cdot 10^{-3}, 2.5 \cdot 10^{-3}, 5 \cdot 10^{-3}, 7.5 \cdot 10^{-3}]$
min_samples_leaf (*MSL*), (second scan)	[750]

Table 10: Values of *LR* and *MSL* involved in the final hyperparameter scans.

Figure 35 shows the results of the first parameter scan, where R_{neg} is plotted against *MSL* for different combinations of the learning rate and FN limit. As for Figure 30, a decrease of R_{neg} was observed at higher and at relatively low values of *MSL*. As explained previously, this is most probably due to the decision trees becoming too shallow or overly complex. At FN = 5.00 % the learning rate does not seem to have any visible effect on the R_{neg} score, while the highest value was found at ($MSL = 750$, $LR = 0.01$) with $R_{\text{neg}} \approx 79.40\,\%$. At FN = 0.50 % and FN = 1.00 %, a learning rate of $LR = 0.2$ tends to reduce the R_{neg} score for most of the *MSL* settings. At the other learning rates, the resulting score shows small variations, but overall, changing the learning rate has a minor effect. The highest scores were observed at ($MSL = 750$, $LR = 0.1$) with $R_{\text{neg}} \approx 61.43\,\%$ for FN = 0.50 % and ($MSL = 500$, $LR = 0.01$) with $R_{\text{neg}} \approx 67.09\,\%$ for FN = 1.00 %. Overall, $MSL = 750$ was considered as the optimal setting, which corresponds to a ratio of $R_{MSL} = N_{\text{train}}/MSL \approx 870$.

To reevaluate if there is an optimal learning rate at $MSL = 750$, a second scan was conducted, which involved lower learning rates (see Table 10) than the first scan. The combined results of the first and second scan at $MSL = 750$ are shown in Figure 36, where R_{neg} is plotted against the learning rate for different FN limits. The second scan confirms that at lower learning rates R_{neg} seems to fluctuate around a certain value but shows no clear maximum. The highest scores of the second scan were observed at $LR = 1 \cdot 10^{-3}$ with $R_{\text{neg}} \approx 61.43\,\%$ (FN = 0.50 %), $LR = 7.5 \cdot 10^{-3}$ with $R_{\text{neg}} \approx 67.14\,\%$ (FN = 1.00 %), and $LR = 7.5 \cdot 10^{-3}$ with $R_{\text{neg}} \approx 79.41\,\%$ (FN = 5.00 %). Therefore, the final optimized BDT

model was configured with $MSL = 750$ and $LR = 7.5 \cdot 10^{-3}$. Differences in the evaluation plots of the predicted probability distributions, probability threshold scans and FNR-FPR curves of the model before and after the final hyperparameter optimization were barely visible. The corresponding plots can be found in section A.1.3 of the appendix. However, a small numerical improvement of R_{neg} and other rates was observed for all three FN limits (see Table 11).

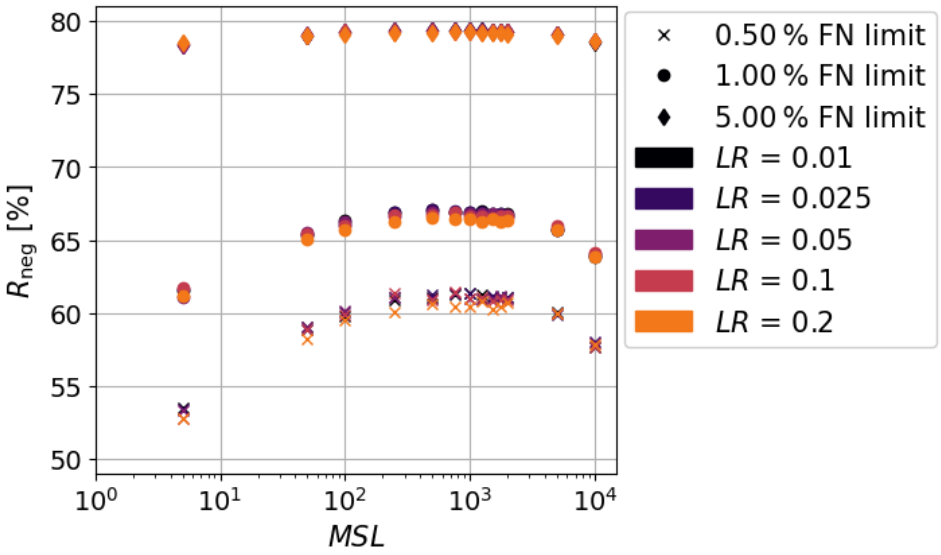

Figure 35: R_{neg} score against the MSL hyperparameter for different learning rates and FN limits.

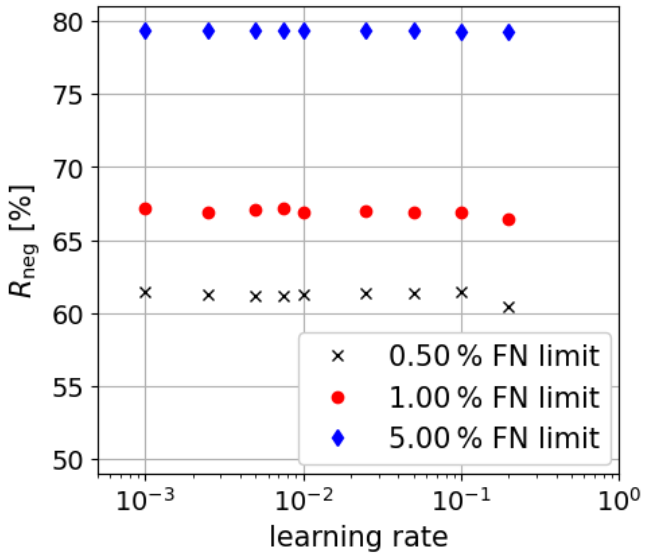

Figure 36: R_{neg} score against the LR hyperparameter at $MSL = 750$ for different FN limits.

before final parameter optimization	R_{neg} [%]	TN [%]	FP [%]	TP [%]	accuracy [%]	error rate [%]	threshold
FN = 0.50 %	61.02	60.52	17.33	21.65	82.18	17.82	0.144
FN = 1.00 %	66.65	65.65	12.20	21.15	86.80	13.20	0.281
FN = 5.00 %	79.32	74.32	3.52	17.15	91.47	8.53	0.788
after final parameter optimization							
FN = 0.50 %	61.16	60.66	17.19	21.65	82.31	17.69	0.138
FN = 1.00 %	67.14	65.15	11.70	21.15	87.30	12.70	0.282
FN = 5.00 %	79.41	74.41	3.44	17.15	91.56	8.44	0.788

Table 11: Rates of the prediction categories as well as R_{neg}, accuracy, error rate, and probability threshold for fixed FN limits, before and after the final optimization of *LR* and *MSL*. The rates were normalized with the total number of samples $N_{\text{test}} = 80584$ in the test set.

Parameters of the final optimized Classifier and Cross-Validation:

The final parameters of the optimized BDT classification model are given in Table 13. A cross-validation, as explained in section 2.3.2, was performed to check how robust the final classifier is against changes in the test set data. The corresponding mean and standard deviation of the R_{neg} score and probability thresholds at different FN limits were calculated based on 10 different test/training set folds and are listed in Table 12. The observed standard deviations are relatively small, proving the stability of the final model. The cross-validation is also a measure for the uncertainty caused by the internal *random_state* of the classifier. This parameter was configured with *None*, which means that the resulting BDT models are not deterministic across multiple training runs with identical hyperparameters [9]. However, the cross-validation indicated that the uncertainty induced by that for the R_{neg} score is acceptably small. In the following section, the final BDT model is compared with the RNN that was trained by B. Mayer [4].

FN limit [%]	R_{neg} [%]	threshold
FN = 0.50 %	61.11 ± 0.29	0.143 ± 0.005
FN = 1.00 %	66.71 ± 0.15	0.273 ± 0.006
FN = 5.00 %	79.33 ± 0.08	0.790 ± 0.003

Table 12: Results of the R_{neg} score and probability thresholds for the final BDT classification model at different FN limits. The mean and standard deviation values are based on 10 test/training set folds of a cross-validation (see section 2.3.2).

binning grid parameter	setting
type	hexagonal
height	$[-600\,\mathrm{m}, 600\,\mathrm{m}[$
tilt along z-axis	8°
r_{hex}	52 m
$n_{\mathrm{bins,xy}}$	271
$n_{\mathrm{bins,z}}$	2
hyperparameter	**setting**
loss	'*log_loss*'
learning_rate (*LR*)	0.0075
max_iter (*MI*)	5000
max_leaf_nodes (*MLN*)	*None*
max_depth (*MD*)	*None*
min_samples_leaf (*MSL*)	750, (R_{MSL} = 870)
l2_regularization	0.0
max_features	1.0
max_bins	255
categorical_features	'*warn*'
monotonic_cst	*None*
interaction_cst	*None*
warm_start	*False*
early_stopping	'*auto*'
scoring	'*loss*'
validation_fraction	0.1
n_iter_no_change	10
tol	*1e-07*
verbose	0
random_state	*None*
class_weight	'*balanced*'
dataset	**number of samples**
training set	652728
validation set	72525
test set	80584

Table 13: Optimized binning grid and hyperparameter settings for the BDT classification model, corresponding to the *scikit-learn* model given in [9]. The respective number of muon samples used for training, validation of early stopping, and evaluation of the final BDT model is listed at the bottom of the table.

3.2.2. Comparison between BDT and RNN

Besides the BDT classifier, a recurrent neural network (RNN) was trained on the muongun 22552 dataset by B. Mayer [4]. Both models were trained for the same purpose. Thus, their predictions can be directly compared to each other. Figure 37 shows the results of the best performing RNN model that were taken from figure 27 of [4]. The plots visualize the rates of the prediction categories at two different probability thresholds, the predicted probability distributions, and the FNR-FPR curve. Equivalent plots of the final optimized BDT model are shown in Figure 38. The illustrated probability thresholds were chosen to match the amount of FN events given by the RNN results. One can see that the predicted probability distributions of the RNN have sharper peaks at zero and one. This leads to a smaller area under the FNR-FPR curve of AUC = 0.02 compared to AUC = 0.035 for the BDT. At FN = 2.70 % the BDT shows a 3.53 % increase in FP events. At FN = 1.40 % it shows a 5.03 % increase in FP events compared to the RNN. Overall, the recurrent neural network achieves better results than the BDT, which is expected since it is a more complex machine learning model. However, this also leads to differences in computational resource requirements between the models.

Training the RNN model took up to four hours using two "NVIDIA A100 Tensor Core-GPUs", depending on the number of training samples [4]. In comparison, training the BDT model with the parameters given in Table 13 took about one hour using ten "Intel Xeon Gold 6326 @2.9 GHz"-CPUs. This shows that, in general, less powerful devices are needed to train the BDT in less time. The prediction time per muon sample for the RNN was determined by first calculating the mean prediction time of a batch of 1024 muons, based on a total dataset of 403928 muons. Afterwards, the resulting mean value and standard deviation of the prediction time per batch were divided by 1024 [4]. This resulted in a prediction time per muon of $(9 \pm 1) \cdot 10^{-3}$ ms on a "NVIDIA A100 Tensor Core-GPU" and 1.3 ± 0.1 ms on a "AMD EPYC 7662 @2.0 GHz"-CPU [4]. In the context of this thesis, a framework was developed to make model predictions on the provided muongun 22552 dataset using the BDT or RNN model. The corresponding code can be found in an ECAP internal git repository [12]. There, the prediction time measurement as described above was implemented in a comparable way for both models, based on scripts and data provided by B.Mayer [13]. By using this framework with one "Intel Xeon E3-1240 v6 @3.7 GHz"-CPU, the prediction time per muon was determined to be 1.3 ± 0.1 ms for the BDT and 21.5 ± 0.3 ms for the RNN. In summary, the BDT model makes less accurate predictions but also requires less computational resources for training and predictions.

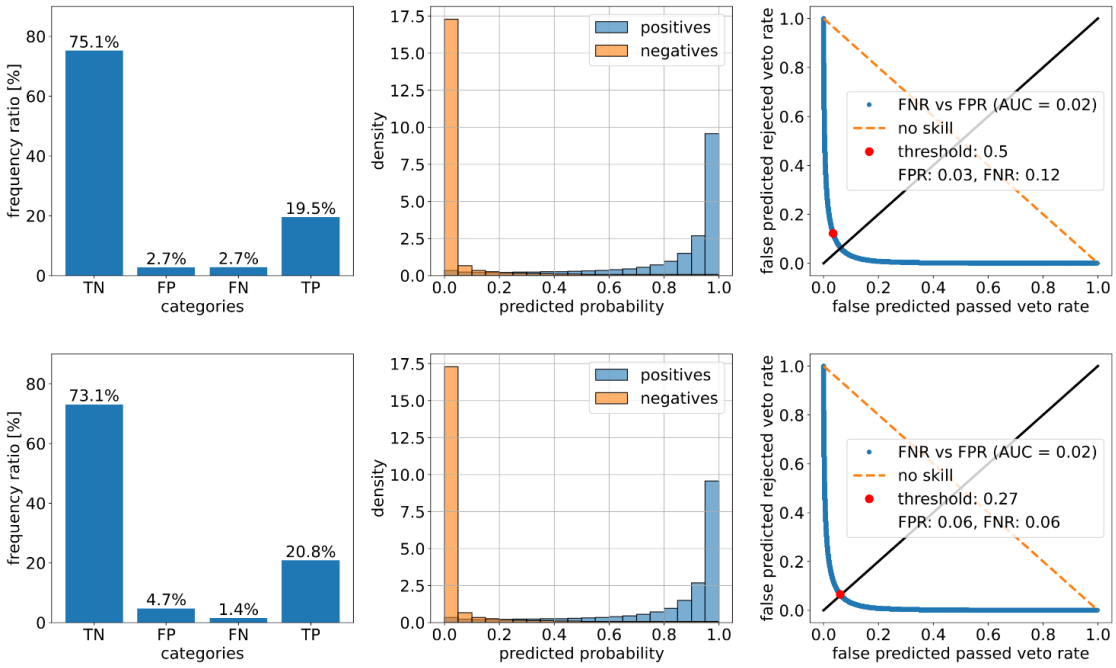

Figure 37: Results of the final optimized RNN classification model. The plots were taken from figure 27 of [4].

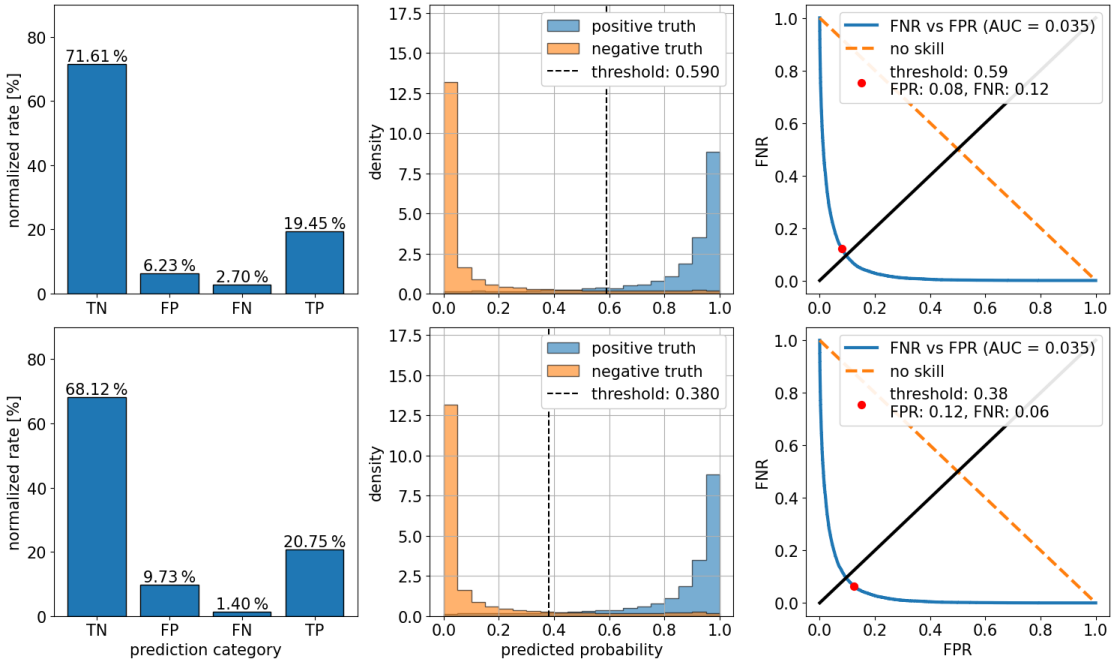

Figure 38: Results of the final optimized BDT classification model. The probability threshold of each row was chosen to match the FN rate of the plots shown in Figure 37.

3.2.3. Investigation of the Classification Bias with Respect to Muon Properties

Using a classification model can be problematic if it introduces a bias towards certain samples. In the context of this work, such a bias could appear as incorrect model predictions of muon events with specific physical properties. For example, the BDT model might only produce FN events for muons with a certain initial energy or propagation direction. In that way, a specific category of muons would be missing for the calculation of the background reduction efficiency. This kind of misinterpretation of the underlying physics of the IceCube detector would have critical implications for data analysis. A bias for muon events of the FP prediction category would be less crucial. However, identifying the corresponding FP events would be important for improving the classification accuracy in future model developments.

Investigations into the existence of such a bias in the final BDT model of section 3.2.1 were done by analysing the distributions of muon events with respect to different physical quantities. These include the initial muon energy, initial muon direction given by (cos(zenith), azimuth), initial muon position in cylindrical coordinates (r, ϕ, z), and the total deposited energy within the binning grid. All these properties are directly or indirectly encoded in the training input (see Table 8). Figure 39 shows the corresponding distributions of events belonging to the positive class (blue histograms), the negative class (orange histograms), and the FP prediction category (dashed black histograms). Figure 40 shows the same distributions for the FN prediction category. The probability thresholds used to define the prediction categories were chosen to be ~ 0.79 for the FP event distributions and ~ 0.29 for the FN event distributions. This was done to focus on the prediction outliers of the negative and positive event classes of the used test set.

It was observed that the events of the muongun dataset 22552, which pass the MESE event filter, tend to be generated more frequently on the lateral surface of the muongun injection cylinder and propagate along a direction that is closer to the horizon (larger zenith angle) compared to the events that are rejected by the MESE filter. Furthermore, they show a larger average amount of deposited energy within the binning grid. In Figure 39, one can see that the distributions of FP events are relatively similar to the distributions of the positive event class. This indicates that the FP outliers tend to have similar physical properties as events that typically are considered to belong to the positive class, based on the quantities provided in the training input. Accordingly, the model is capable of correctly interpreting the underlying physics, but some information appears to be missing that is necessary for the model to classify FP outliers correctly. The distributions of FP events are not constrained within a small parameter range, but spread out broadly, similar to the distributions of the positive class. Therefore, no clear indications of a bias were observed.

The FN distributions in Figure 40, show a comparable behaviour for most of the analyzed quantities, meaning that they tend to be similar to the distributions of the negative class. An exception was observed with respect to the total deposited energy within the binning grid. There, the distribution of the FN outliers is similar to the distribution of the positive class. Thus, FN outliers tend to have physical properties that partially correspond to the positive or the negative event class. The BDT model appears to have problems classifying these events correctly. Therefore, the FN outliers are likely biased towards events showing this behaviour. Apart from that, the azimuth and ϕ distributions

of the FN events exhibit relatively strong over- and underfluctuations for some bins. This is probably a binning artefact, due to a sparsely covered parameter space. At the corresponding probability threshold of ~ 0.29, there are only about 800 events involved in the FN distributions. Analyzing the distributions provides only superficial insight into why the model fails to predict certain muon events correctly. For future investigation, it might be useful to look at single muons in an event display to obtain more information in this regard.

In summary, detailed information on the optimization of the BDT classification model was provided in section 3.2. By performing a cross-validation, it was verified that the

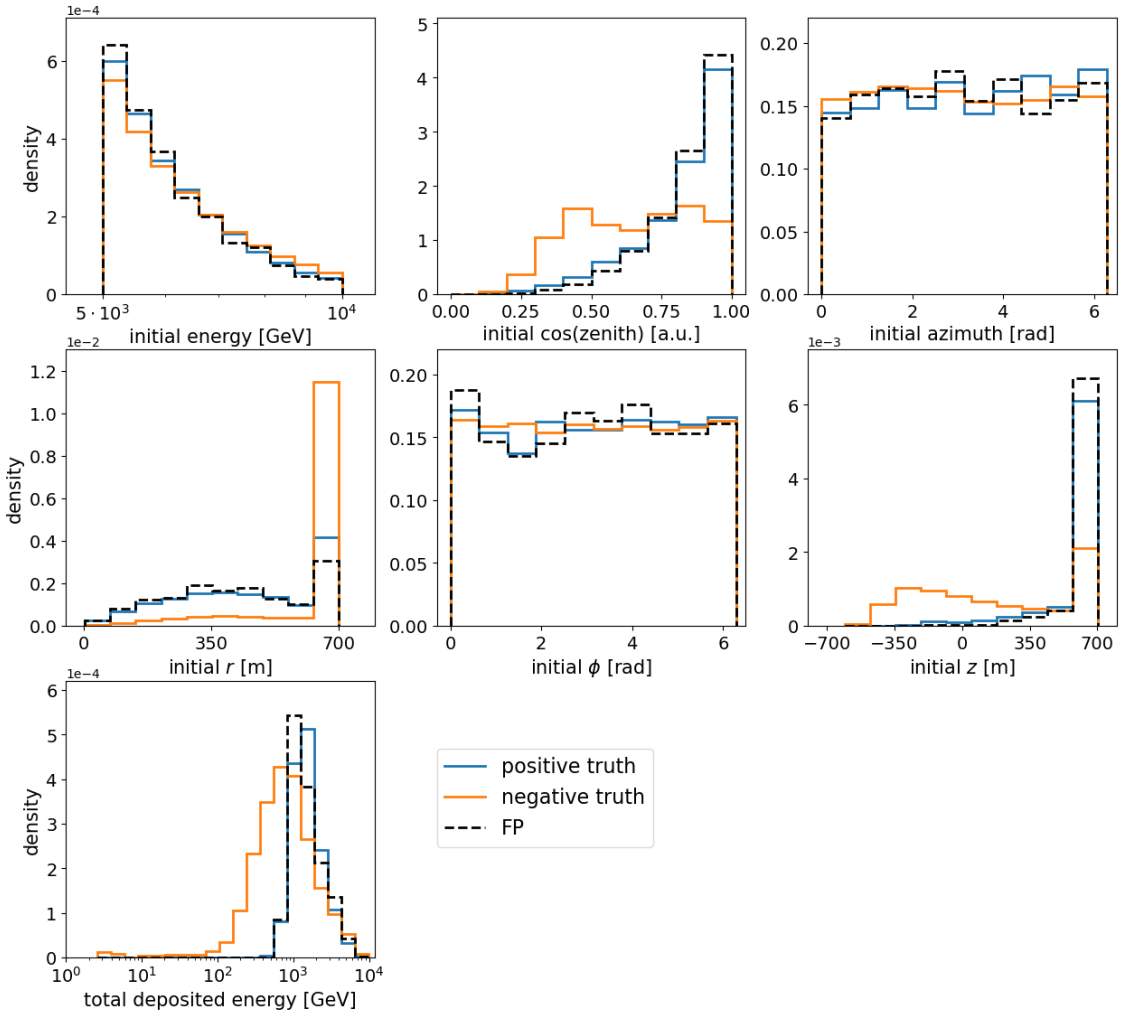

Figure 39: Distributions of muon events with respect to seven different muon quantities. These include the initial muon energy, initial muon direction given by (cos(zenith), azimuth), initial muon position in cylindrical coordinates (r, ϕ, z), and the total deposited energy within the binning grid. For each of the quantities, three distributions are shown, which correspond to events belonging to the positive class (blue), the negative class (orange), and the FP prediction category (dashed black), with respect to the test set. The FP histograms were determined using a probability threshold of ~ 0.79.

predictions of the BDT classifier are robust against changes in the test set data. The BDT was trained on the same muongun dataset from the IceCube simulation as the RNN developed in [4]. A comparison between the two models showed that the BDT outperforms the RNN with respect to computational resource requirements, while the RNN model is more accurate in making predictions. The following section provides information on a testing framework for optimized muon simulations that was developed in the context of this work. The computational gain obtained by integrating the BDT classifier into the IceCube simulation chain within this framework is presented and discussed.

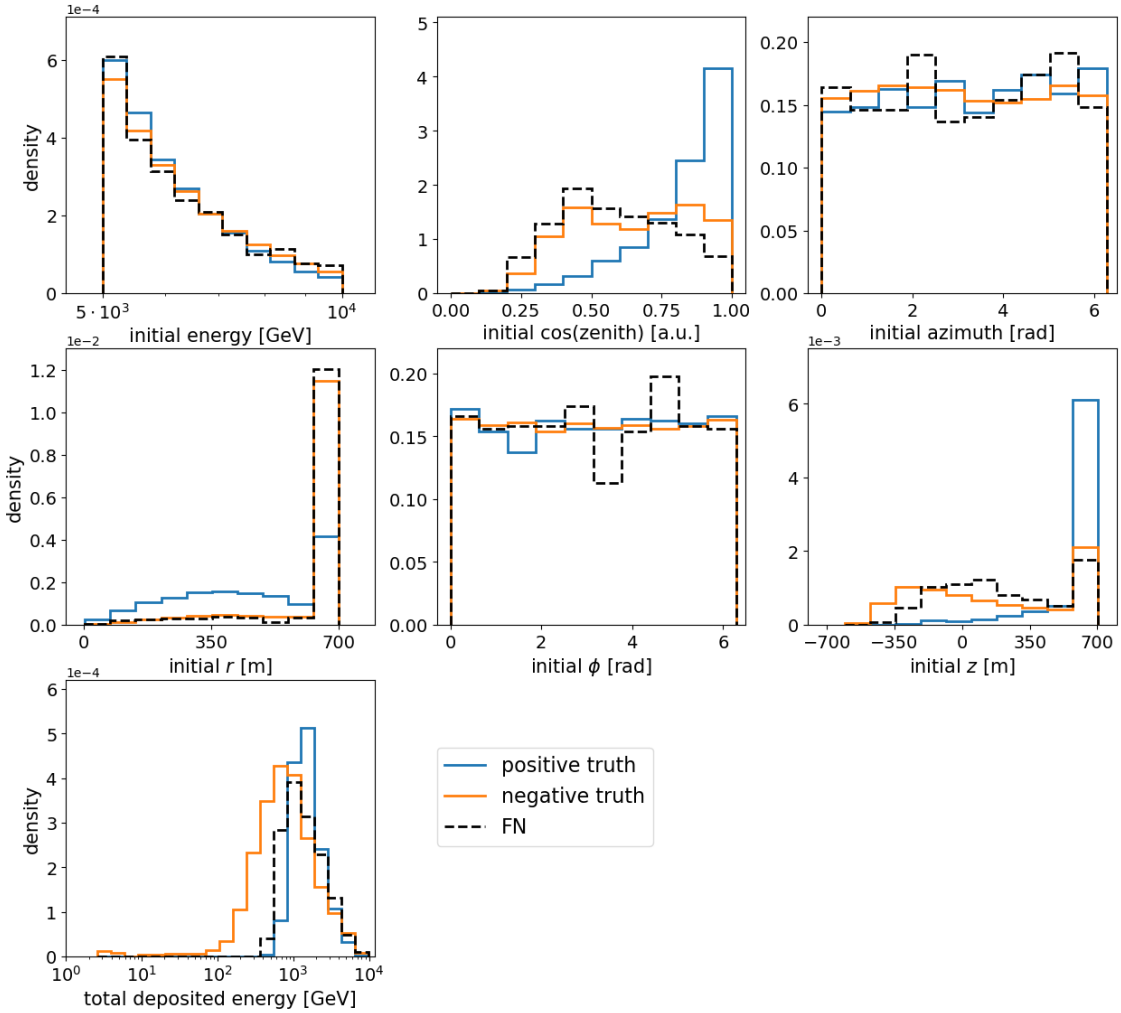

Figure 40: Distributions of muon events with respect to seven different muon quantities. These include the initial muon energy, initial muon direction given by (cos(zenith), azimuth), initial muon position in cylindrical coordinates (r, ϕ, z), and the total deposited energy within the binning grid. For each of the quantities, three distributions are shown, which correspond to events belonging to the positive class (blue), the negative class (orange), and the FN prediction category (dashed black), with respect to the test set. The FN histograms were determined using a probability threshold of ~ 0.29.

4. Optimized Muon Simulations

Official IceCube simulations are managed via the IceProd job submission framework [14]. However, the development of a hybrid simulation approach, as presented in this thesis, is still at an early stage. Therefore, it was considered appropriate to develop a dedicated testing framework for optimized muon simulations, which ensures full access to the parameters of IceCube muon simulations, but also allows for quick and flexible testing and debugging. This was achieved by implementing a data-processing workflow using *Snakemake* (v9.3.0) [15],[16], based on the official IceCube simulation software provided by the IceTray software framework [17]. In section 4.1, the setup of the Snakemake simulation workflow is explained in more detail. The final results of this thesis regarding the computational gain obtained by fully integrating the BDT classifier into the muon simulation chain are presented in section 4.2.

4.1. Testing Framework for Optimized Muon Simulations based on Snakemake

Snakemake is a Python-based tool for creating automated data processing workflows that enable reproducible and scalable data analyses. The execution of tasks on high-performance computing (HPC) clusters is natively supported. A Snakemake workflow is defined as a sequence of *rules*, which are implemented in a *snakefile*. These *rules* can be understood as modular building blocks of the workflow, which automatically track dependencies between intermediate input and output data. The main configuration parameters of a Snakemake workflow, such as file paths, resource settings for HPC-cluster job submissions, or input parameters of involved scripts, can be defined in a *config* file (see Figure 41). The *rules* implemented in the *snakefile* are parametrized based on the corresponding *config* file (see Figure 42). Arbitrarily complex workflows can be realized by concatenating different *rules* with each other and executing several runs of the same workflow in parallel. More information on how to use Snakemake can be found in the corresponding documentation [16].

As described in section 2.2, the IceCube simulation is based on a modular structure. Therefore, Snakemake was considered to be a suitable tool for developing a testing framework for optimized muon simulations in IceCube. The corresponding workflow was constructed by defining individual *rules* for the simulation steps that were explained in section 2.2. The muongun dataset 23260 from 2024, was used as a template for that [18]. Full access to the original simulation parameters was ensured by integrating the scripts used in that template into the Snakemake workflow. All input parameters available for these scripts were transferred to the *config* file of the workflow. The input parameter settings were chosen as consistent as possible with the original settings of the muongun dataset 23260. Details are available in the GitHub repository of the testing framework, where the entire code, including the *config* file, can be found [19]. Information on where to find the used simulation scripts within the IceTray GitHub repository is provided in section A.2.1 of the appendix.

```yaml
general_parameters:
  conda_env: "myenv"

rules:
  myrule:
    params:
      script_path: "/path/to/script"
      input_path1: "/path/to/input1"
      input_path2: "/path/to/input2"
      output_path: "/path/to/output"
      par1: 42
      par2: "second_parameter"
      par3: 1.234
    resources:
      time: "01:00:00"
      cluster: "mycluster"
      threads: 1

  new_rule:
  # etc. ...
```

Figure 41: Example of a *config*.yaml file defining the parameters of the *snakefile* shown in Figure 42.

Figure 43 shows a scheme of the developed simulation workflow, where the *rules* corresponding to the simulation steps taken from the dataset 23260 are illustrated as blue boxes. The BDT classifier was integrated into the workflow by splitting it into two sub-workflows. The idea is to generate training data based on the conventional muon simulation and train the BDT on it in sub-workflow A. The trained model is then used in sub-workflow B to generate muon data based on the optimized muon simulation. Each sub-workflow includes certain additional *rules*. In sub-workflow A, two *rules* were added after filter level 2, to extract the training data and train the BDT model (red boxes in Figure 43). The training parameters of the model were fixed according to Table 13. In sub-workflow B, a *rule* was added between the step of muon propagation and photon propagation (green box in Figure 43). This is where the BDT predicts the probability of passing the MESE filter for each muon event. Muons, which are unlikely to pass the filter, are excluded from further simulation by applying a rejection sampling method (see section 2.3.2). For reasons of efficiency, the initial simulation steps up to and including muon propagation are executed in parallel for both sub-workflows. Furthermore, the overall workflow supports parallel executions of several simulation runs with the same parameter settings, meaning that multiple instances of the whole workflow, as shown in Figure 43, can be submitted as parallel jobs to the HPC-cluster. Additional *rules*, besides the ones illustrated in Figure 43, were implemented for evaluation purposes. Since they do not contribute to the actual muon simulation chain, they are not further discussed, but can be found in [19]. In the following section, the results of optimized muon simulations based on the testing framework are presented.

```
1
2   configfile: "config.yaml"
3
4   conda_env = config['general_parameters']['conda_env']
5
6   myrule_pars = config['rules']['myrule']['params']
7   myrule_res  = config['rules']['myrule']['resources']
8
9   rule myrule:
10      params:
11          script_path = myrule_pars['script_path'],
12          par1 = myrule_pars['par1'],
13          par2 = myrule_pars['par2'],
14          par3 = myrule_pars['par3']
15
16      input:
17          input_file1 = myrule_pars['input_path1'],
18          input_file2 = myrule_pars['input_path2']
19
20      output:
21          output_file = myrule_pars['output_path']
22
23      resources:
24          time = myrule_res['time'],
25          clusters = myrule_pars['cluster']
26
27      threads: myrule_res['threads']
28
29      conda:
30          conda_env
31
32      shell:
33          """
34          python {params.script_path} \
35          --par1 {params.par1} --par2 {params.par2} --par3 {params.par3} \
36          --in1 {input.input_file1} --in2 {input.input_file2} --out {output.output_file}
37          """
38
39  rule new_rule:
40  # etc. ...
```

Figure 42: Example of a *snakefile* showing one *rule* based on the parameters of the *config*.yaml file of Figure 41.

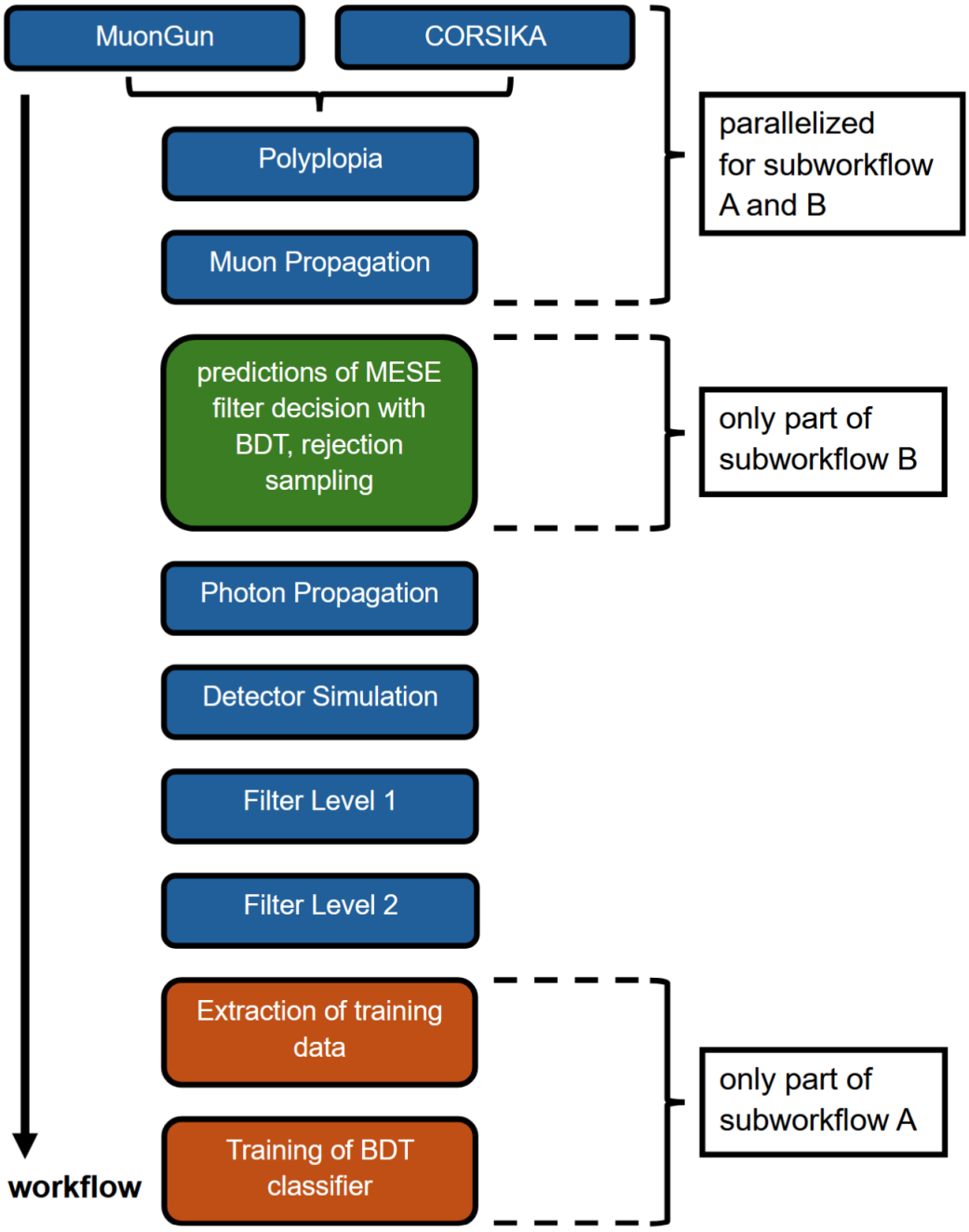

Figure 43: Scheme of the simulation workflow, implemented with Snakemake [15],[16]. Each box corresponds to a certain *rule*. Different colors denote if the respective *rule* is present for sub-workflow A (red), sub-workflow B (green), or both (blue). The blue boxes correspond to simulation steps taken from the muongun dataset 23260 [18]. The underlying code of the workflow can be found in [19].

4.2. Efficiency of Optimized Muon Simulations

The simulation framework introduced in the previous section can be used to evaluate the efficiency of optimized muon simulations with an integrated the BDT classifier. Maximum efficiency is achieved when the classifier makes no false predictions, implying optimal computational resource savings. The classification of muon events within this framework is not based on a fixed probability threshold, as it was done in section 3.2, but on a rejection sampling method (see section 2.3.2). Therefore, the computational gain g_{comp} was defined to quantify the efficiency of the optimized simulation compared to the standard simulation without using the BDT classifier:

$$g_{\text{comp}} \coloneqq f_1 \cdot f_2 = \frac{N_\gamma}{N_\gamma^{\text{opt}}} \cdot \frac{N_{\text{eff}}\big|_{\text{pos. events}} + N_{\text{pos,subev}}^{\text{opt}}}{N_{\text{pos}} + N_{\text{pos,subev}}} \tag{4.1}$$

This formula is based on two factors f_1 and f_2. N_γ is the number of photons simulated during photon propagation of the standard simulation. The equivalent number obtained for the optimized simulation is called N_γ^{opt}. The factor f_1 is the ratio between these numbers. Depending on the amount of TP and FP predictions, this factor takes values of $f_1 \geq 1$. A high computational gain is reflected by high values of f_1, since the simulation time spent on photon propagation scales with the number of simulated photons. Considering the case where the amount of FN predictions is relatively large, shows that f_1 is not sufficient to describe the computational gain. In such a case, the number of events remaining after applying the MESE filter might be much smaller than in the standard simulation. Thus, more muons would need to be simulated overall to achieve the same number of events surviving the MESE filter as in the simulation without BDT. This effect partially compensates for the computational gain achieved by simulating fewer photons during photon propagation. The factor f_2 was introduced to account for this in the calculation of g_{comp}. $N_{\text{eff}}\big|_{\text{pos. events}}$ is the effective sample size obtained by evaluating Equation 2.20 for all events remaining after rejection sampling and applying the MESE filter. N_{pos} is the number of events remaining after applying the MESE filter in the standard simulation. In rare cases, a muon event can be split into multiple sub-events during the simulation of filter level one and filter level two. The MESE filter is also applied to these sub-events. The number of additional sub-events passing the MESE filter in case of the standard simulation is given by $N_{\text{pos,subev}}$. The equivalent number corresponding to the optimized simulation is called $N_{\text{pos,subev}}^{\text{opt}}$. Overall, this results in a value of $f_2 \leq 1$. As a consequence, f_1 is scaled down by f_2 depending on the accuracy of the BDT classifier. With that, g_{comp} provides a reasonable measure of the computational gain. In relation to Figure 43, the results of N_γ, N_{pos}, and $N_{\text{pos,subev}}$ were obtained by simulating all events that were rejected after applying the BDT in sub-workflow B, in parallel to the events that remained in the optimized event sample. This is redundant for an optimized simulation and was done only for evaluation purposes.

4.2.1. Optimized Muon Simulations at different Muon Energies

In this section, the results of optimized muon simulations at different muon energies are presented, based on the simulation framework introduced in section 4.1. In the context of section 3.2.1, the BDT classifier was trained and optimized based on the muongun dataset 22552 [10]. The muons of this dataset were simulated within an energy range of $5 \cdot 10^3\,\mathrm{GeV} - 1 \cdot 10^4\,\mathrm{GeV}$ with a spectral index of $\gamma = 4.5$. For reasons of consistency, optimized simulations were performed using these settings for γ and the initial muon energy, among other energy ranges. Based on the power law implemented for MuonGun (see section 2.2), a setting of $\gamma = 4.5$ results in a steeply falling energy spectrum with many events at the lower energy bound. However, the energy ranges used are relatively small, which counteracts this effect. An overview of all energy ranges for which optimized simulations were performed, and the corresponding identifiers used to refer to them, is given in Table 14.

identifier	energy range [GeV]	N_{events}	N_{train}
i)	$1 \cdot 10^2 - 1 \cdot 10^3$	$2.0 \cdot 10^6$	655163
ii)	$1 \cdot 10^3 - 1 \cdot 10^4$	$1.2 \cdot 10^6$	627859
iii)	$5 \cdot 10^3 - 1 \cdot 10^4$	$8.8 \cdot 10^5$	536114
iv)	$1 \cdot 10^4 - 1 \cdot 10^5$	$8.8 \cdot 10^5$	571847
v)	$1 \cdot 10^5 - 1 \cdot 10^6$	$8.0 \cdot 10^5$	604225

Table 14: Different energy ranges, for which optimized simulations were performed, and their identifiers. A spectral index of $\gamma = 4.5$ was used for all of them. In the respective execution of subworkflow A, the BDT classifier was trained on a dataset including N_{train} events. The initial number of events simulated by muongun in subworkflow A was set to the respective value of N_{events}.

The total number of muon events N_{events} simulated in run iii) was set to $8.8 \cdot 10^5$. The resulting number of events involved in the BDT training was $N_{\mathrm{train}} \approx 5.4 \cdot 10^5$. The difference in event number occurs due to several reasons, besides the split of the training set and the early stopping validation set. During photon propagation, a filter is applied that removes all muon events from the simulation that produce a number of Monte Carlo photo electrons (MCPE) below a certain threshold. During the detector simulation, another filter is applied, which removes all events that do not trigger the detector. After the simulation of filter level 2, some of the muon events end up being split into multiple sub-events. All of the sub-events are considered as individual training samples. Therefore, $N_{\mathrm{train}} \neq N_{\mathrm{events}}$ is observed. The extent of this discrepancy depends on the energy range of the simulated muons. In section 3.2.1, using a training set with $N_{\mathrm{train}} \approx 6.5 \cdot 10^5$ samples showed reasonable results. Therefore, N_{events} was set accordingly, to obtain a value of N_{train} of the same order of magnitude whenever the BDT was trained during an execution of the simulation framework (see Table 14).

Discrepancy in Training Data:

Figure 44 shows a comparison of different loss curves. The parameter settings of the underlying BDT models were set according to Table 13. The leftmost plot shows the training and validation loss curves obtained in the context of section 3.2.1, where the BDT model was trained on muongun dataset 22552 [10]. A validation loss of $L_{\text{val}} \approx 0.24$ was achieved after 2250 boosting iterations. The central plot shows the loss curves corresponding to the BDT that was trained in sub-workflow A during simulation iii). There, a higher validation loss of $L_{\text{val}} \approx 0.44$ was achieved after 839 boosting iterations. Since the BDT parameters applied during training were identical, the inequality of the loss curves implies that there are differences in the muon data of dataset 22552, compared to the data generated in sub-workflow A based on the parameter settings of dataset 23260 [18]. Although the muon energy range and spectral index were set to identical values, a discrepancy was found in the ratio of the number of events in the positive (passing MESE filter) and the negative (rejected by MESE filter) event classes. For dataset 22552 this ratio is about $\sim 22\,\%$, while for the data generated in sub-workflow A, this ratio is about $\sim 5.4\,\%$. Therefore, it was assumed that the increase in the validation loss is related to the comparably strong underrepresentation of the positive event class in the training set. To verify this assumption, another simulation at the energy range of iii) was conducted, where the ratio of events in the positive and negative classes was artificially adjusted to be $\sim 22\,\%$. This was done by simulating more training data and randomly removing some of the events belonging to the negative event class. The resulting loss curve can be seen in the rightmost plot of Figure 44. A small decrease of the final validation loss to $L_{\text{val}} \approx 0.41$ after 2773 boosting iterations was observed compared to the loss of the central plot. Still, the loss shown in the leftmost plot is much smaller. This leads to the conclusion that there must be further differences between the training data of dataset 22552 and the data generated in sub-workflow A. Some further insights in this regard are discussed in section 4.2.2.

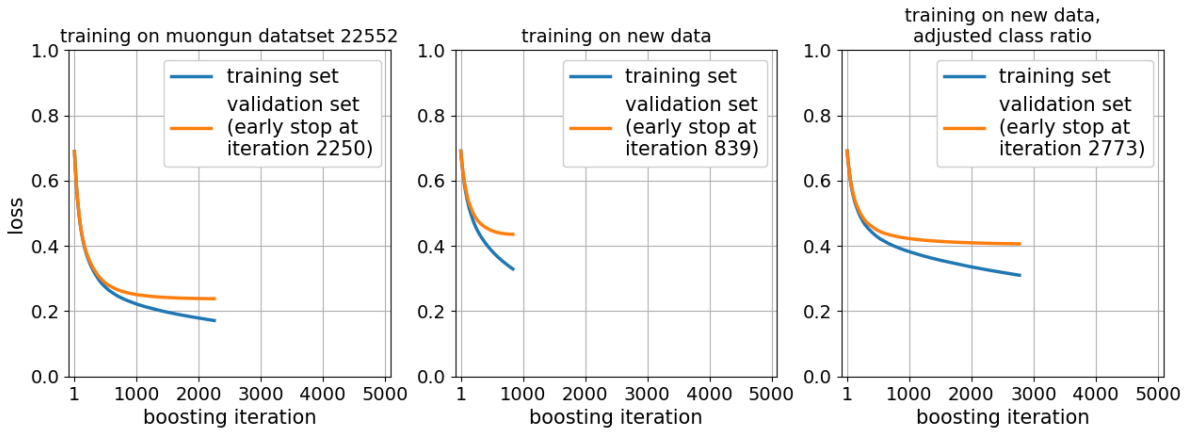

Figure 44: Training loss (blue curves) and validation loss (orange curves) against the boosting iteration obtained by training the BDT classifier on different datasets. **Left:** Muongun dataset 22552 [10]. **Center:** Data generated by the simulation framework introduced in section 4.1. **Right:** Data generated by the simulation framework with an artificially adjusted ratio of event classes in the training set.

Evaluation of Optimized Muon Simulations:

The results of sub-workflow B of simulation iii) are illustrated in Figure 45. The classification rates of the prediction categories (see Table 3) are shown based on two different normalizations. Furthermore, the distributions of the predicted probabilities with respect to the positive (blue) and negative (orange) event classes are presented. Additionally, the distributions of the predicted probabilities according to the classification after rejection sampling are shown. The black distribution involves all events that are excluded from the optimized dataset, while the green distribution corresponds to all events that are fully simulated. The plots are based on all events generated in sub-workflow B except for the sub-events added during the filter simulation. This is because no BDT prediction exists for these sub-events. Using rejection sampling instead of a fixed probability threshold for event classification generally works. On the other hand, it also reduces transparency regarding which events are sampled into the optimized dataset and which are not. Compared to the plots of section 3.2.2, one can see that the distributions of the negative and positive event classes are less sharp around zero and one. This is a consequence of the increase in the final loss that occurred during training. This is also reflected in the error rate of 24.46 %, which exceeds the values given in Table 11. Thus, the BDT predictions of the simulation framework are less accurate compared to the final results of section 3.2.1 and section 3.2.2.

The corresponding prediction results obtained by artificially adjusting the ratio of event classes in the training set are presented in Figure 46. A small decrease in the FN and FP rate was observed, consistent with the small decrease in the loss, leading to an error rate of 23.66 %. The resulting distributions are shifted closer towards zero and one compared to Figure 45. Still, the predictions are not as good as observed in section 3.2.1 and section 3.2.2 for the optimized BDT model that was trained on dataset 22552 [10]. Another investigation was carried out by making predictions on the data generated in sub-workflow B using the BDT model that was trained on dataset 22552 (see Figure 47). It was found that while the predictions of the positive event class are more accurate, the predictions of the negative class are worse compared to Figure 46, leading to an increase of FP and a decrease of FN events. This resulted in a relative increase of the error rate to 31.18 %. The probability distribution of the negative class shows a small second peak at one, besides the expected peak at zero. This misinterpretation of events further indicates that there is a systematic difference between dataset 22552 and the data generated by the simulation framework. The plots showing the loss curves and prediction results of the other energy ranges of Table 14, can be found in section A.2.2 of the appendix.

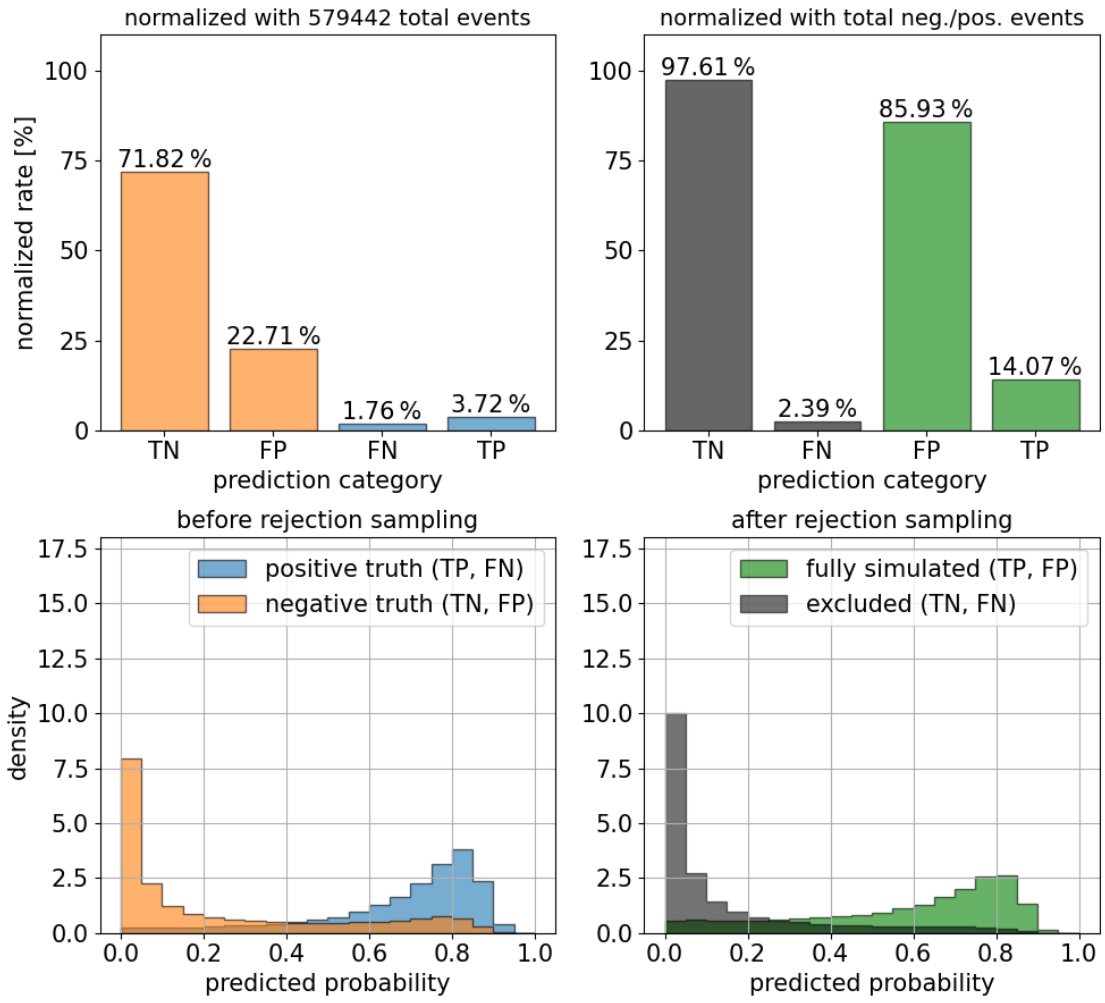

Figure 45: Results of the optimized muon simulations of sub-workflow B for muons within the energy range of $5 \cdot 10^3\,\mathrm{GeV} - 1 \cdot 10^4\,\mathrm{GeV}$ (spectral index $\gamma = 4.5$). Sub-events are excluded from the plots as explained in this section. **Top row:** Classification rates of the prediction categories (see Table 3) normalized with respect to the total number of events generated in sub-workflow B (left) and with respect to the total number of positive or negative predictions (right). **Bottom row:** Distribution of the predicted probabilities by the BDT classifier for actually positive (blue) and actually negative (orange) events before rejection sampling (left). Distribution of predicted probabilities for fully simulated (green) and excluded (black) events according to rejection sampling (right). Within the left or right column of the figure, the colour of each distribution indicates the associated prediction categories.

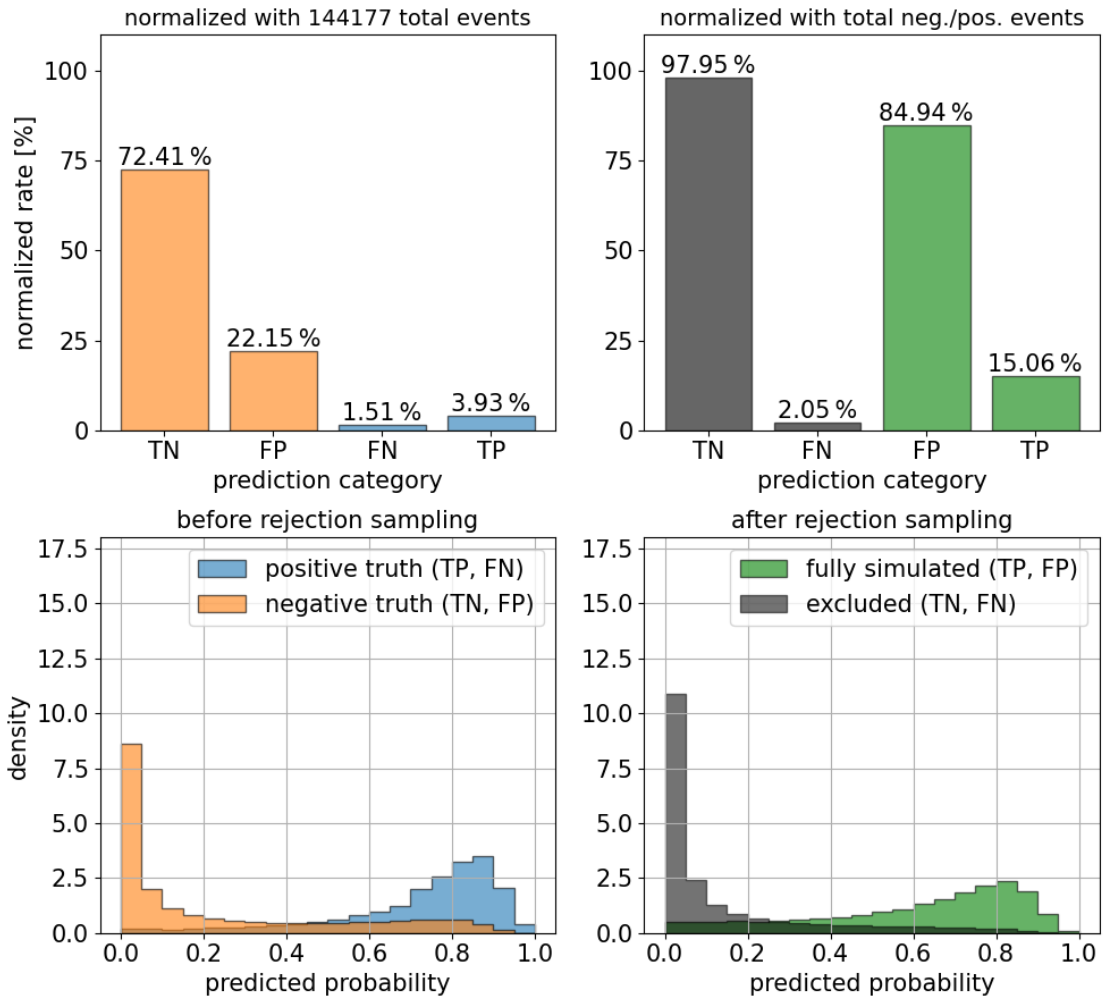

Figure 46: Results of the optimized muon simulations of sub-workflow B for muons within the energy range of $5 \cdot 10^3\,\mathrm{GeV} - 1 \cdot 10^4\,\mathrm{GeV}$ (spectral index $\gamma = 4.5$)). The ratio of the event classes in the training set was artificially adjusted to be ~22 %. Sub-events are excluded from the plots as explained in this section. **Top row:** Classification rates of the prediction categories (see Table 3) normalized with respect to the total number of events generated in sub-workflow B (left) and with respect to the total number of positive or negative predictions (right). **Bottom row:** Distribution of the predicted probabilities by the BDT classifier for actually positive (blue) and actually negative (orange) events before rejection sampling (left). Distribution of predicted probabilities for fully simulated (green) and excluded (black) events according to rejection sampling (right). Within the left or right column of the figure, the colour of each distribution indicates the associated prediction categories.

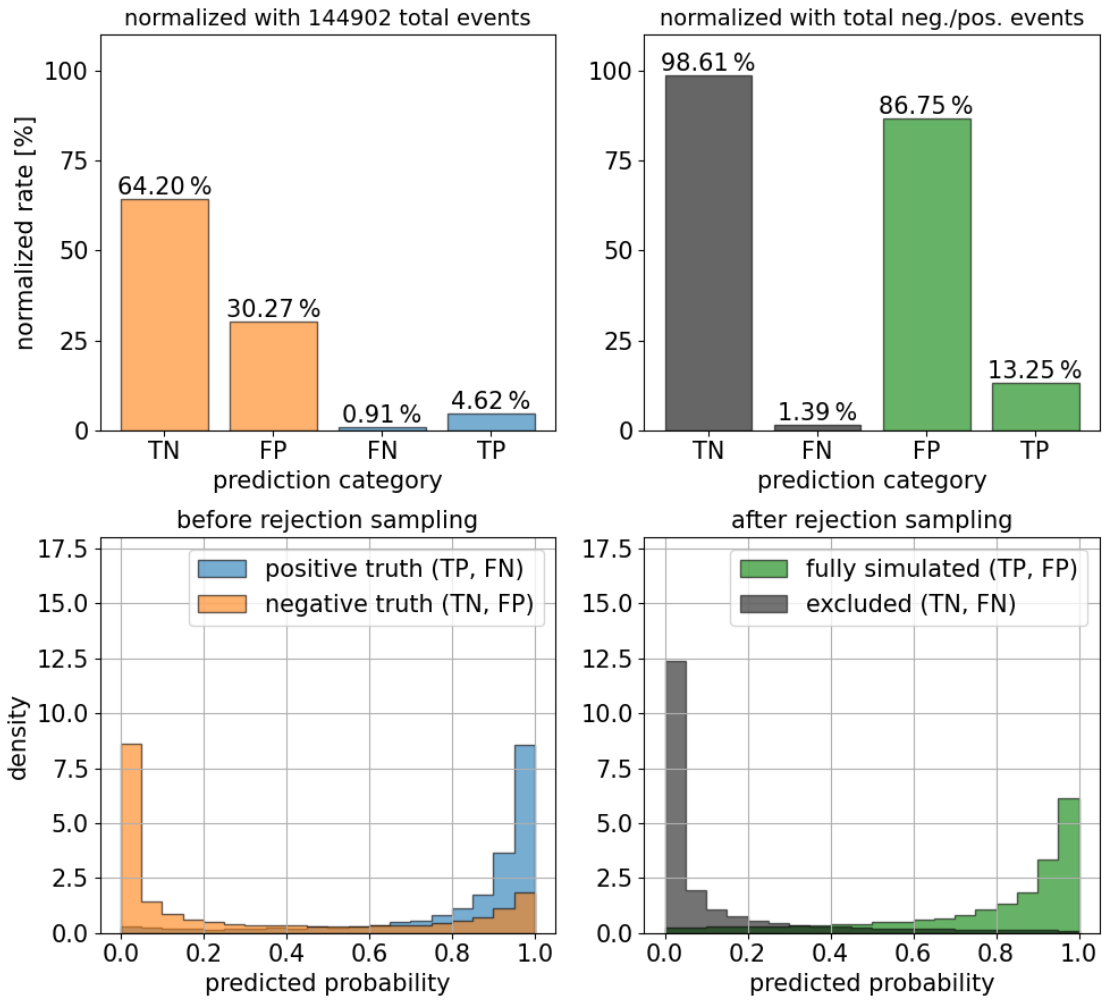

Figure 47: Results of the optimized muon simulations of sub-workflow B for muons within the energy range of $5 \cdot 10^3\,\text{GeV} - 1 \cdot 10^4\,\text{GeV}$ (spectral index $\gamma = 4.5$). The predictions are based on the final BDT model trained in section 3.2.1. Sub-events are excluded from the plots as explained in this section. **Top row:** Classification rates of the prediction categories (see Table 3) normalized with respect to the total number of events generated in sub-workflow B (left) and with respect to the total number of positive or negative predictions (right). **Bottom row:** Distribution of the predicted probabilities by the BDT classifier for actually positive (blue) and actually negative (orange) events before rejection sampling (left). Distribution of predicted probabilities for fully simulated (green) and excluded (black) events according to rejection sampling (right). Within the left or right column of the figure, the colour of each distribution indicates the associated prediction categories.

For simulation ii) and iv), error rates of 21.25 % and 24.49 % were observed. With ~ 1.8 % (ii)) and ~ 3.2 % (iv)) instead of ~ 5.4 %, the ratios of training samples in the different event classes were smaller than for simulation iii). Given the classification results, the trained BDT still showed predictions of similar quality as for simulation iii). Simulation i) and v) showed a further decrease of the ratios of the event classes to ~ 0.13 % and ~ 0.10 %. This strong underrepresentation of the positive event class is probably one of the reasons why the training at the corresponding energy ranges did not work properly. The resulting BDT predictions are close to a balanced 50 %/50 % classification of events into the true and false prediction categories. Another problem at lower energies of $1 \cdot 10^2\,\mathrm{GeV} - 1 \cdot 10^3\,\mathrm{GeV}$ might be that the BDT input vectors tend to have more entries that are zero or close to zero, since less energy deposition within the binning grid is expected. In section 3.1.3, this was identified to be a general problem. Overall, this leads to the conclusion that the BDT hyperparameters and the parameters of the binning grid have to be optimized individually for the respective energy range. In this regard, section 3.2.1 provides a detailed insight into which parameter adjustments have the highest impact. Furthermore, it might be necessary to artificially adjust the event class ratio in the training set for certain energy ranges, as it was done for simulation iii). Depending on the extent of the underrepresentation of the positive event class, this might not be reasonable. Computational resources required to generate suitable training sets could outweigh the resource savings achieved by the optimized simulation approach.

Table 15 and Table 16 show the fractions f_{MCPE} and f_{trig} of muon events that were rejected by the MCPE filter during photon propagation and by the trigger filter during detector simulation of sub-workflow B for different energy ranges. The fractions were calculated with respect to the dataset obtained by the optimized simulation ($f_{\dots}^{\mathrm{opt}}$) and with respect to the dataset obtained by the standard simulation. By comparing these numbers, one can see that at energy ranges where the BDT showed reasonable results, a large part of the events that would be removed by the MCPE filter and the trigger filter are already rejected by the BDT.

Furthermore, Table 15 and Table 16 show the resulting mean computational gain for all simulation runs that were discussed above. It was calculated by averaging over multiple parallel runs of sub-workflow B. The uncertainty was determined via a Gaussian error propagation based on the standard deviations of the quantities of Equation 4.1. The mean values and standard deviations of these quantities are also given in the tables. For the energy ranges of simulation ii), iii) and iv), a computational gain of about $g_{\mathrm{comp}} \approx 2$ was achieved. The highest computational gain of $g_{\mathrm{comp}} = 2.2 \pm 0.4$ was obtained at the energy range of iii) with the additional adjustment of the event class ratio in the training set. The computational gain obtained for simulation i) and v) did not show a significant improvement compared to conventional simulations. This was expected from the poor results observed for the corresponding BDT predictions. As discussed above, the BDT predictions within the simulation framework are not optimal. Further investigations into suitable parameter settings for the BDT at different energy ranges were beyond the scope of this work, but will be necessary in the future. However, at energy ranges already showing acceptable results, the optimized muon simulations require only about half the computational resources of the conventional muon simulations. This demonstrates the potential of using a hybrid simulation approach for muon background simulations in IceCube.

energy range	iii)	iii), adjusted ratio of event classes	iii), BDT trained on dataset 22552
number of parallel runs sub-workflow B	4	4	4
N_{events} per run of sub-workflow B	$2.2 \cdot 10^5$	$5.5 \cdot 10^4$	$5.5 \cdot 10^4$
f^{opt}_{MCPE} [%]	2.3	2.0	0.8
f_{MCPE} [%]	16.7	16.8	16.8
f^{opt}_{trig} [%]	2.2	1.8	0.7
f_{trig} [%]	17.4	17.6	17.4
$\langle N_{\text{eff}}\big\vert_{\text{pos. events}} \rangle$	3605 ± 336	1030 ± 187	1100 ± 340
$\langle N_{\text{pos}} \rangle$	7930 ± 63	1931 ± 32	2004 ± 23
$\langle N^{\text{opt}}_{\text{pos,subev}} \rangle$	88 ± 7	28 ± 4	26 ± 5
$\langle N_{\text{pos,subev}} \rangle$	131 ± 9	40 ± 5	31 ± 5
$\langle N^{\text{opt}}_{\gamma} \rangle$	$(1.68 \pm 0.01) \cdot 10^{12}$	$(4.10 \pm 0.06) \cdot 10^{11}$	$(5.20 \pm 0.05) \cdot 10^{11}$
$\langle N_{\gamma} \rangle$	$(6.810 \pm 0.004) \cdot 10^{12}$	$(1.692 \pm 0.005) \cdot 10^{12}$	$(1.704 \pm 0.002) \cdot 10^{12}$
$\langle g_{\mathbf{comp}} \rangle$	1.9 ± 0.2	2.2 ± 0.4	1.8 ± 0.5

Table 15: Mean values and standard deviations of the quantities of Equation 4.1, based on multiple parallel runs of sub-workflow B (see first row) at different energy ranges. The energy ranges are represented by their indentifier corresponding to Table 14. The respective number of events N_{events} per run is given in the second row. The resulting mean computational gain g_{comp} is given with an uncertainty based on a Gaussian error propagation. The qunatities f_{MCPE} and f_{trig} are the fractions of muon events that were rejected by the MCPE filter during photon propagation and by the trigger filter during detector simulation of sub-workflow B. They are normalized with respect to the optimized dataset ($f^{opt}_{...}$) as well as to the dataset obtained in the standard simulation.

energy range	i)	ii)	iv)	v)
number of parallel runs sub-workflow B	4	4	5	4
N_{events} per run of sub-workflow B	$5.0 \cdot 10^4$	$3.0 \cdot 10^4$	$2.2 \cdot 10^4$	$5.0 \cdot 10^4$
f^{opt}_{MCPE} [%]	27.8	11.5	6.1	10.4
f_{MCPE} [%]	26.0	21.0	14.4	8.0
f^{opt}_{trig} [%]	38.3	12.1	3.9	10.7
f_{trig} [%]	38.7	22.4	15.3	9.7
$\langle N_{\text{eff}}\big\vert_{\text{pos. events}} \rangle$	8 ± 2	134 ± 24	230 ± 70	17 ± 4
$\langle N_{\text{pos}} \rangle$	19 ± 1	296 ± 13	504 ± 8	44 ± 3
$\langle N^{\text{opt}}_{\text{pos,subev}} \rangle$	1 ± 1	4 ± 3	4 ± 2	0 ± 0
$\langle N_{\text{pos,subev}} \rangle$	2 ± 1	6 ± 1	8 ± 2	1 ± 1
$\langle N^{\text{opt}}_{\gamma} \rangle$	$(9.52 \pm 0.03) \cdot 10^{10}$	$(8.59 \pm 0.04) \cdot 10^{10}$	$(3.08 \pm 0.05) \cdot 10^{11}$	$(1.20 \pm 0.01) \cdot 10^{13}$
$\langle N_{\gamma} \rangle$	$(2.60 \pm 0.02) \cdot 10^{11}$	$(3.50 \pm 0.02) \cdot 10^{11}$	$(1.38 \pm 0.01) \cdot 10^{12}$	$(2.97 \pm 0.01) \cdot 10^{13}$
$\langle g_{\textbf{comp}} \rangle$	1.1 ± 0.3	1.9 ± 0.3	2.0 ± 0.6	1.0 ± 0.3

Table 16: Part two of Table 15. Mean values and standard deviations of the quantities of Equation 4.1, based on multiple parallel runs of sub-workflow B (see first row) at different energy ranges. The energy ranges are represented by their indentifier corresponding to Table 14. The respective number of events N_{events} per run is given in the second row. The resulting mean computational gain g_{comp} is given with an uncertainty based on a Gaussian error propagation. The qunatities f_{MCPE} and f_{trig} are the fractions of muon events that were rejected by the MCPE filter during photon propagation and by the trigger filter during detector simulation of sub-workflow B. They are normalized with respect to the optimized dataset ($f^{opt}_{...}$) as well as to the dataset obtained in the standard simulation.

4.2.2. Investigation of Event Distributions with Respect to Muon Properties

As a final investigation, the event distributions with respect to different muon properties were analysed, analogue to section 3.2.3. The best results of the computational gain were achieved for the energy range of $5 \cdot 10^3\,\text{GeV} - 1 \cdot 10^4\,\text{GeV}$. Therefore, the plots provided in this section are based on prediction results illustrated in Figure 45 (without the adjusted ratio of event classes). Figure 48 shows the corresponding distributions of events belonging to the positive class (blue histograms), the negative class (orange histograms), and the FP prediction category (dashed black histograms). Figure 49 shows the same distributions for the FN prediction category. In section 3.2.3 the probability thresholds were chosen so that only the prediction outliers were included in the FP and FN histograms. Since the optimized simulations are based on rejection sampling, the histograms include not only prediction outliers. However, due to the probabilistic nature of the rejection sampling process, the contribution of events which are no outliers is suppressed.

Comparing the distributions provided this section to the corresponding distributions of section 3.2.3, gives some insights on why the BDT predictions on the data generated with the simulation framework in section 4.2.1 appear to be less accurate compared to predictions on dataset 22552 [10]. One observation is, that the distributions of the positive and negative event classes of this section are more similar to each other compared to the corresponding distributions of section 3.2.3. This effect is most dominant for the initial energy, the initial cos(zenith), the r-component of the initial position and the z-component of the initial position. This implies that the muon events generated by the simulation framework are more similar to each other regarding their physical properties, despite having different MESE filter outcomes. This might be the main reason why the BDT has problems to learn the differences between the muon event classes during training, resulting in an increased loss and worse predictions.

The same script (see Table 20 in the appendix) with similar parameter settings was used for executing muongun in dataset 22552 and in the simulation framework. This indicates that the internal code of the muongun script was possibly changed since dataset 22552 was produced. To make a more confident statement on that, more detailed investigations in this regard are required. This was not part of this work, but might be interesting for future studies.

The distributions of FP and FN events seem to follow the distributions of the positive event class. This is an expected behaviour for the FP events but not for the FN events, which should be more similar to the distributions of the negative event class. This is another indicator for the BDT not being able to properly learn the differences between the event classes. Apart from that, the distributions of FP and FN events are not constrained within a small parameter range, but spread out broadly. Therefore, no clear indications of a bias with respect to certain event properties were observed. As mentioned in section 3.2.3 it might be useful to look at single muons in an event display to obtain more information on why certain events end up in the FP or FN prediction category.

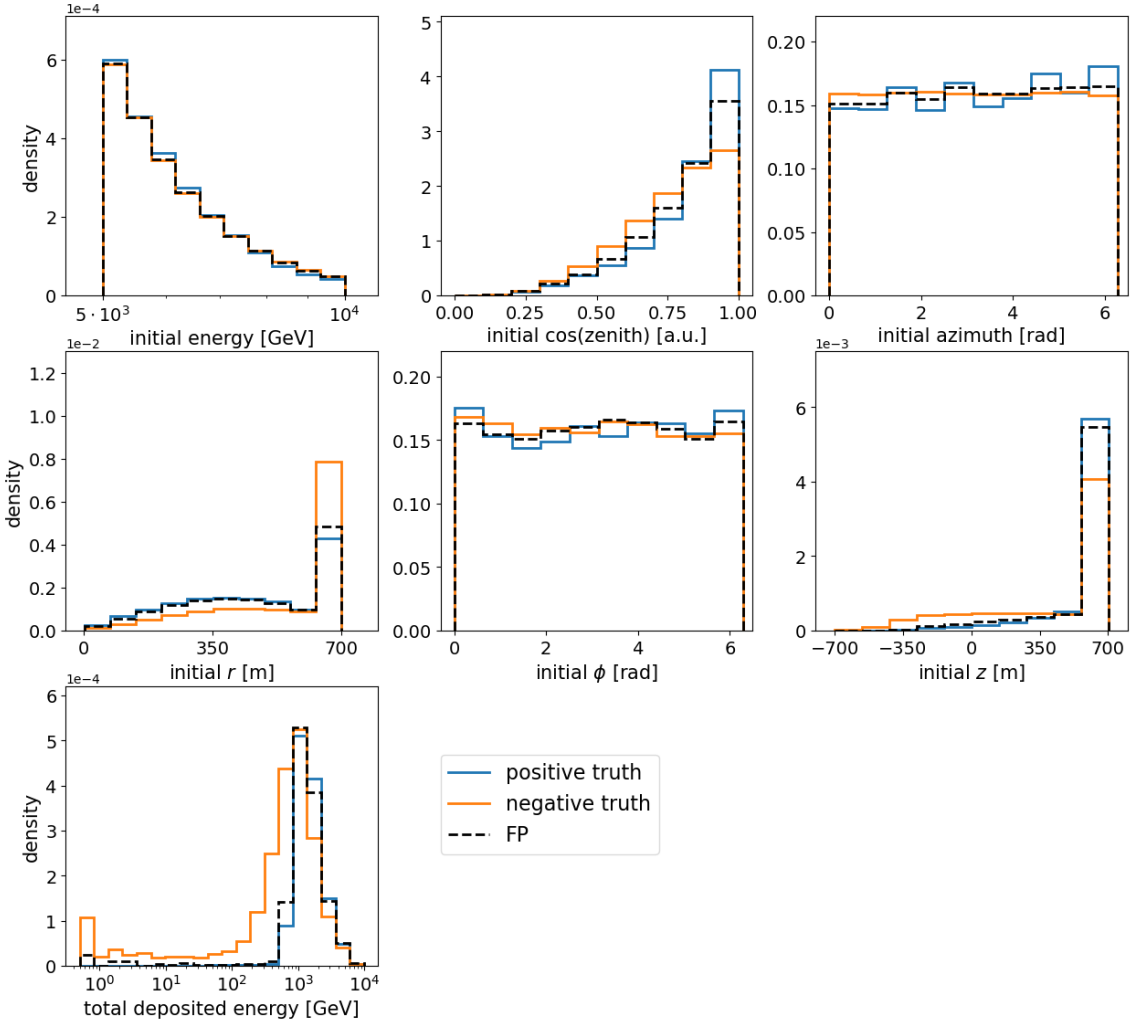

Figure 48: Distributions of muon events with respect to seven different muon quantities. These include the initial muon energy, initial muon direction given by (cos(zenith), azimuth), initial muon position in cylindrical coordinates (r, ϕ, z), and the total deposited energy within the binning grid. For each of the quantities, three distributions are shown, which correspond to events belonging to the positive class (blue), the negative class (orange), and the FN prediction category (dashed black). The FP histograms were determined with respect to the rejection sampling applied in sub-workflow B.

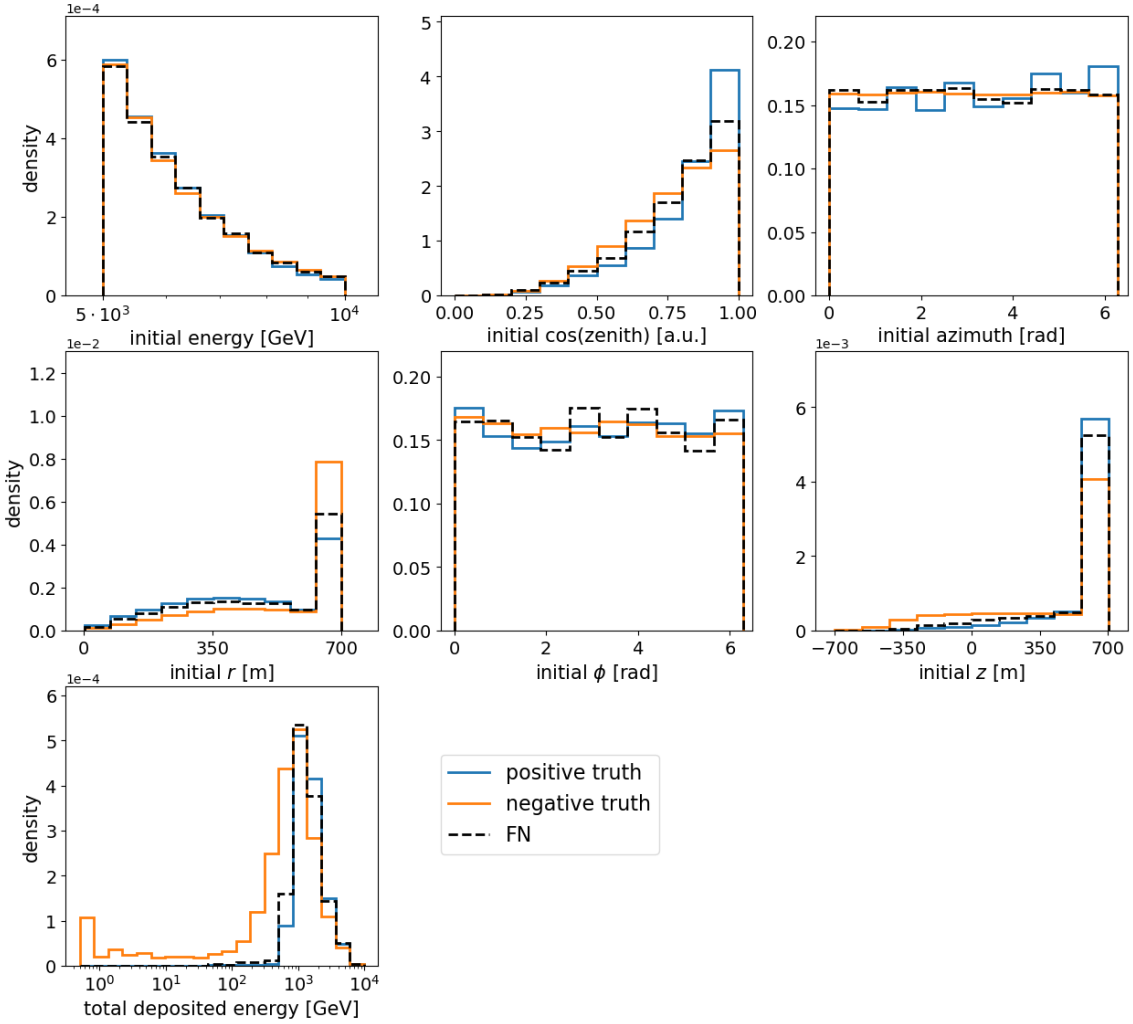

Figure 49: Distributions of muon events with respect to seven different muon quantities. These include the initial muon energy, initial muon direction given by (cos(zenith), azimuth), initial muon position in cylindrical coordinates (r, ϕ, z), and the total deposited energy within the binning grid. For each of the quantities, three distributions are shown, which correspond to events belonging to the positive class (blue), the negative class (orange), and the FN prediction category (dashed black). The FN histograms were determined with respect to the rejection sampling applied in sub-workflow B.

5. Conclusion and Outlook

Detecting astrophysical neutrinos with the IceCube Neutrino Observatory presents several challenges, including a large background of atmospheric muons. During data analysis, multiple event filters have to be applied to suppress this background. The resulting background reduction efficiency and its associated uncertainty are estimated from extensive muon simulations, which involve the computationally expensive propagation of Cherenkov photons in ice. By applying the event filters to the simulated data, a large number of muons is removed at the end of the simulation chain. Consequently, a lot of computational resources are unnecessarily spent on photon propagation. In this thesis, a hybrid simulation approach was presented that combines conventional muon simulations with a boosted decision tree that predicts the MESE event filter outcome based on the muon energy loss information. To save computational resources, the full simulation, including photon propagation, is carried out only for muon events predicted to pass the MESE filter. By training a BDT regression model on muon data of a toy simulation, it was shown that this kind of simulation approach generally works. In a second step, a BDT classifier was trained on muon data of the IceCube simulation [10]. After optimizing various hyperparameters, the BDT classifier achieved a prediction accuracy close to that of a recurrent neural network trained on the same task [4]. Finally, a dedicated testing framework for optimized muon simulations was developed [19], providing full access to the IceCube simulation parameters used for a recently generated muon dataset [18]. The BDT was integrated into this framework, including the training process. A computational gain g_{comp} was defined that acts as a measure of the resource savings achieved by using the hybrid simulation approach. Depending on the initial energy range of the simulated muons, a computational gain of up to 2.2 ± 0.4 was observed. This implies that optimized muon simulations require only about half the computational resources of conventional muon simulations. Therefore, about twice as many background muons can be simulated within the same amount of time. This leads to the conclusion that employing a hybrid simulation approach as proposed by this thesis might reduce the statistical uncertainty of the estimated background reduction efficiency, without increasing the budget spent on simulations of the muon background.

The BDT as trained within the simulation framework showed an error rate of $20\,\% - 25\,\%$ in the best cases. Therefore, further investigations are required to reduce the number of wrong event classifications. This also shows that the maximum achievable computational gain is potentially higher than the factor obtained in this work. Future studies should include an individual optimization of the BDT hyperparameters for different muon energy ranges. Further insights are needed on the physical characteristics of falsely classified events to understand why these predictions are observed. A first step in gathering information on that would be to analyse individual muon events in an event display. Furthermore, it would be interesting to integrate other classifiers, such as the RNN developed by B. Mayer [4], into the simulation framework and analyse the resulting computational gain. Since the simulation framework provides full access to the parameters of the involved scripts from IceTray [17], various simulation configurations can be tested in future analyses of optimized muon simulations.

References

[1] F. Pedregosa, G. Varoquaux, A. Gramfort, V. Michel, B. Thirion, O. Grisel, M. Blondel, P. Prettenhofer, R. Weiss, V. Dubourg, J. Vanderplas, A. Passos, D. Cournapeau, M. Brucher, M. Perrot, and E. Duchesnay. *Scikit-learn: Machine Learning in Python*. Journal of Machine Learning Research, 12:2825-2830, 2011.

[2] (ECAP internal):
B. Mayer, *Git repository for toycube simulation and neural network training*, https://git.ecap.work/ob70ifit/toycube, accessed: September 16, 2025

[3] (ECAP internal):
S. Koch, *Side branch of [2] used for development of a BDT model*, https://git.ecap.work/ob70ifit/toycube/-/tree/GBR_branch?ref_type=heads, accessed: September 27, 2025

[4] B. Mayer, *Development of a Recurrent Neural Network for optimizing the muon simulation efficiency in IceCube*, Bachelor Thesis in Physics at Erlangen Centre for Astroparticle Physics (ECAP, FAU), submission date: March 4, 2025, https://ecap.nat.fau.de/wp-content/uploads/2025/07/2025_Bachelor_Thesis_Benedikt_Mayer.pdf, accessed: November 16, 2025

[5] J.H. Koehne et al., *PROPOSAL: A tool for propagation of charged leptons*, Computer Physics Communications, 184(9):2070-2090, 2013, DOI: 10.1016/j.cpc.2013.04.001

[6] M. G. Aartsen, R. Abbasi, M. Ackermann, J. Adams, et al., *Energy reconstruction methods in the icecube neutrino telescope*, Journal of Instrumentation 9, P03009, 2014, DOI: 10.1088/1748-0221/9/03/P03009

[7] C. Haack, private communication, 2025

[8] Histogram-based BDT regression model of *scikit-learn* (v1.5.2), https://scikit-learn.org/1.5/modules/generated/sklearn.ensemble.HistGradientBoostingRegressor.html, accessed: September 30, 2025

[9] Histogram-based BDT classification model of *scikit-learn* (v1.5.2), https://scikit-learn.org/1.5/modules/generated/sklearn.ensemble.HistGradientBoostingClassifier.html, accessed: October 20, 2025

[10] (IceCube collaboration internal):
IceProd muongun datatset 22552, https://simprod.icecube.wisc.edu/dataset/22552/, accessed: October 21, 2025

[11] Log loss function of *scikit-learn* (v1.5.2),https://scikit-learn.org/1.5/modules/generated/sklearn.metrics.log_loss.html,
accessed: October 21, 2025

[12] (ECAP internal):
S. Koch, *Git repository for performing MESE veto predictions of muons generated by muongun*, https://git.ecap.work/di81jiny/muon_veto_prediction,
accessed: October 26, 2025

[13] B. Mayer, private communication, 2025

[14] M.G. Aartsen, et al., *The IceProd framework: Distributed data processing for the IceCube neutrino observatory*, Journal of parallel and distributed computing, 75:198-211, 2015, DOI: 10.1016/j.jpdc.2014.08.001

[15] Mölder F, Jablonski KP, Letcher B et al., *Sustainable data analysis with Snakemake [version 1; peer review: 1 approved, 1 approved with reservations]*, F1000Research, 10:33, 2021, DOI: 10.12688/f1000research.29032.1

[16] Snakemake (v9.3.0) documentation, https://snakemake.readthedocs.io/en/v9.3.0/, accessed: October 29, 2025

[17] (IceCube collaboration internal): IceCube Collaboration, *Git repository of the IceTray software framework.*, https://github.com/icecube/icetray, accessed: October 29, 2025

[18] (IceCube collaboration internal): IceProd muongun datatset 23260, https://simprod.icecube.wisc.edu/dataset/23260/, accessed: October 29, 2025

[19] (IceCube collaboration internal): S. Koch, *GitHub repository of the testing framework for optimized muon simulations*, https://github.com/icecube/snakemake_muon_simulation, accessed: October 30, 2025

[20] T.K. Gaisser, R. Engel, E. Resconi, *Cosmic Rays and Particle Physics*, 2nd edition Cambridge University Press, 2016

[21] M. Ahlers, F. Halzen, *Opening a new window onto the universe with IceCube*, Progress in Particle and Nuclear Physics, 102:73-88 , 2018, DOI 10.1016/j.ppnp.2018.05.001

[22] J.A. Aguilar and J. Yang, IceCube/WIPAC, https://icecube.wisc.edu/news/research/2016/10/neutrinos-and-gamma-rays-partnership-to-explore-extreme-universe/, accessed November 05, 2025

[23] KATRIN Collaboration et al., *Direct neutrino-mass measurement based on 259 days of KATRIN data*, Science, 388:180-185, 2025, DOI: 10.1126/science.adq9592

[24] The IceCube Collaboration, *IceCube gallery*, https://icecube.wisc.edu/gallery/, accessed November 06, 2025

[25] The IceCube Collaboration, *Evidence for High-Energy Extraterrestrial Neutrinos at the IceCube Detector*, Sience, 342, 1242856, 2013, DOI: 10.1126/science.1242856

[26] The IceCube Collaboration et al., *Multimessenger observations of a flaring blazar coincident with high-energy neutrino IceCube-170922A*, Sience, 361, eaat1378, 2018, DOI: 10.1126/science.aat1378

[27] R. Abbasi et al., *Evidence for neutrino emission from the nearby active galaxy NGC 1068*, Science, 378:538-543, 2022, DOI: 10.1126/science.abg3395

[28] The IceCube Collaboration, *Observation of high-energy neutrinos from the Galactic plane*, Science, 380:1338-1343, 2023, DOI: 10.1126/science.adc9818

[29] V. Basu and A. Balagopal V. on behalf of the IceCube Collaboration, *From PeV to TeV: Astrophysical Neutrinos with Contained Vertices in 10 years of IceCube Data*, Proceedings of Science, ICRC2023:1007, 2023, DOI: 10.22323/1.444.1007

[30] R. Abbasi et al., *IceCube high-energy starting event sample: Description and flux characterization with 7.5 years of data*, Physical Review D, 104, 022002, 2021, DOI: 10.1103/PhysRevD.104.022002

[31] M. Kronmueller, T. Glauch on behalf of the IceCube Collaboration, *Application of Deep Neural Networks to Event Type Classification in IceCube*, Proceedings of Science, ICRC2019:937, 2019, DOI: 10.22323/1.358.0937

[32] (IceCube collaboration internal):
B. Smithers, J.C. Diaz-Velez, A. Olivas, K. Meagher, *Presentation materials of the IceCube Software Session*, IceCube Bootcamp: Summer 2021, https://events.icecube.wisc.edu/event/134/

[33] G. Carminatti et al., *MUPAGE: a fast atmospheric MUon GEnerator for neutrino telescopes based on PArametric formulas*, arXiv:0907.5563v1 [astro-ph.IM], 31 Jul 2009, DOI: 10.48550/arXiv.0907.5563

[34] Karlsruhe Institute of Technology, *CORSIKA documentation website*, https://www.iap.kit.edu/corsika/, accessed November 09, 2025

[35] I. Narsky, F.C. Porter, *Statistical Analysis Techniques in Particle Physics - Fits, Density Estimation and Supervised Learning*, WILEY-VCH, 2014

[36] L.C. Raedel, *Measurement of High-Energy Muon Neutrinos with the IceCube Neutrino Observatory*, Dissertation, RWTH Aachen University, 2017, https://publications.rwth-aachen.de/record/709576/files/709576.pdf, accessed November 10, 2025

[37] IceCube-Gen2 Collaboration, *IceCube-Gen2 Technical Design Report - Part I and II*, July 2023, https://icecube-gen2.wisc.edu/science/publications/tdr/, accessed November 10, 2025

[38] A. Zollanvari, *Machine Learning with Python - Theory and Implementation*, Springer Nature Switzerland AG, 2023, DOI: 10.1007/978-3-031-33342-2

[39] R. Abbasi et al., *The IceCube data acquisition system: Signal capture, digitization, and timestamping*, Nuclear Instruments and Methods in Physics Research Section A: Accelerators, Spectrometers, Detectors and Associated Equipment, 601(3):294-316, 2009, DOI: 10.1016/j.nima.2009.01.001

[40] (IceCube collaboration internal):
B. Riedel, *Presentation materials of the 2025 Computing report*, Fall 2025 IceCube Collaboration meeting in Salt Lake City, 2025, https://events.icecube.wisc.edu/event/330/contributions/11280/, accessed November 11, 2025

[41] User guide of *scikit-learn* (version 1.5.2), https://scikit-learn.org/1.5/user_guide.html, accessed November 11, 2025

[42] J.H. Friedman, *Stochastic gradient boosting*, Computational Statistics & Data Analysis, 38(4):367-378, 2002, DOI: 10.1016/S0167-9473(01)00065-2

[43] J.H. Friedman, *Greedy function approximation: A gradient boosting machine*, The Annals of Statistics, 29(5):1189-1232, 2001, DOI: 10.1214/aos/1013203451

[44] BDT regression model of *scikit-learn* (v1.5.2), `https://scikit-learn.org/1.5/modules/generated/sklearn.ensemble.GradientBoostingRegressor.html`, accessed: November 12, 2025

[45] C.P. Robert, G. Casella, *Introducing Monte Carlo Methods with R*, Springer Science+Business Media, 2010, DOI: 10.1007/978-1-4419-1576-4

[46] A.C. Argüelles et al., *Unified atmospheric neutrino passing fractions for large-scale neutrino telescopes*, Journal of Cosmology and Astroparticle Physics, JCAP07(047), 2018, DOI: 10.1088/1475-7516/2018/07/047

A. Appendix

A.1. Appendix - Development of a Classifier

A.1.1. Appendix - section 3.1.2

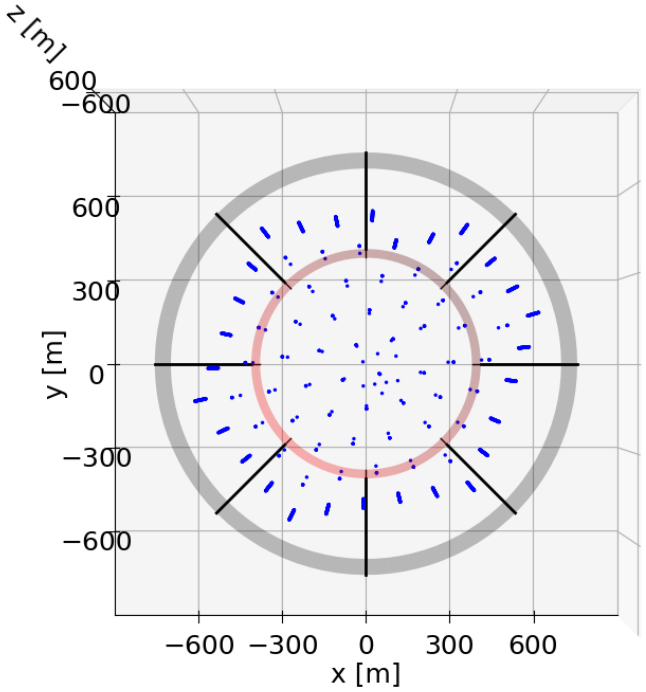

Figure 50: Top view of Figure 13.

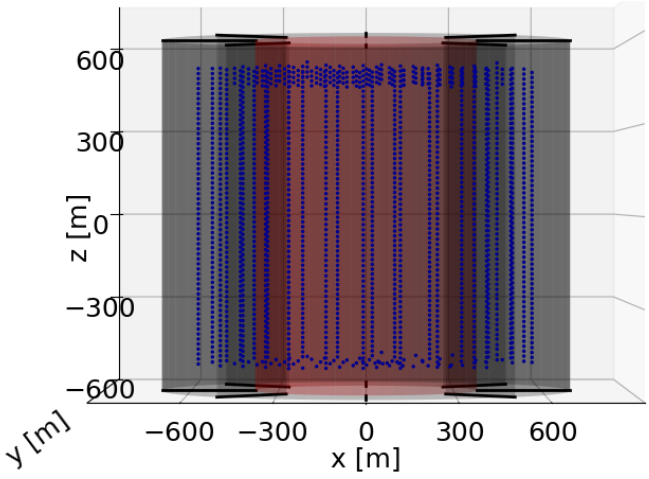

Figure 51: Side view of Figure 13.

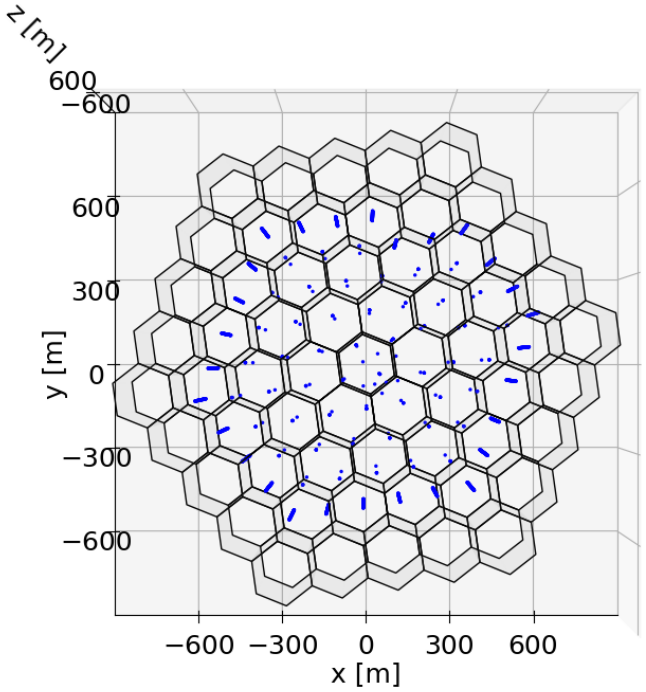

Figure 52: Top view of Figure 14.

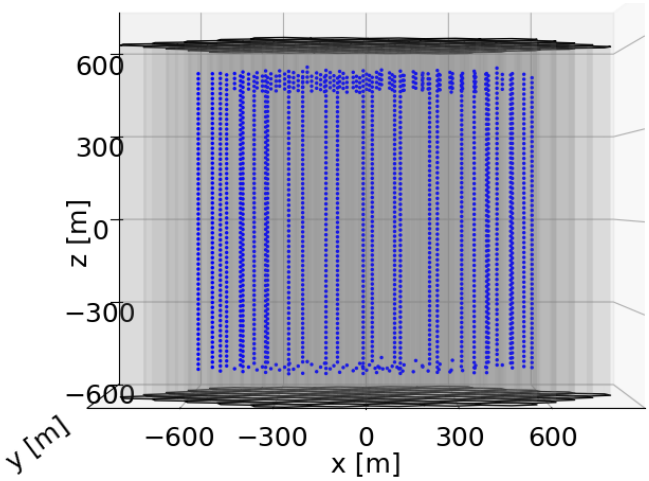

Figure 53: Side view of Figure 14.

A.1.2. Appendix - section 3.1.3

hyperparameter	default setting
loss	'*squared_error*'
quantile	*None*
learning_rate	0.1
max_iter	100
max_leaf_nodes	31
max_depth	*None*
min_samples_leaf	20
l2_regularization	0.0
max_features	1.0
max_bins	255
categorical_features	'*warn*'
monotonic_cst	*None*
interaction_cst	*None*
warm_start	*False*
early_stopping	'*auto*'
scoring	'*loss*'
validation_fraction	0.1
n_iter_no_change	10
tol	*1e-07*
verbose	0
random_state	*None*

Table 17: Default hyperparameter settings of the regression model as given by [8].

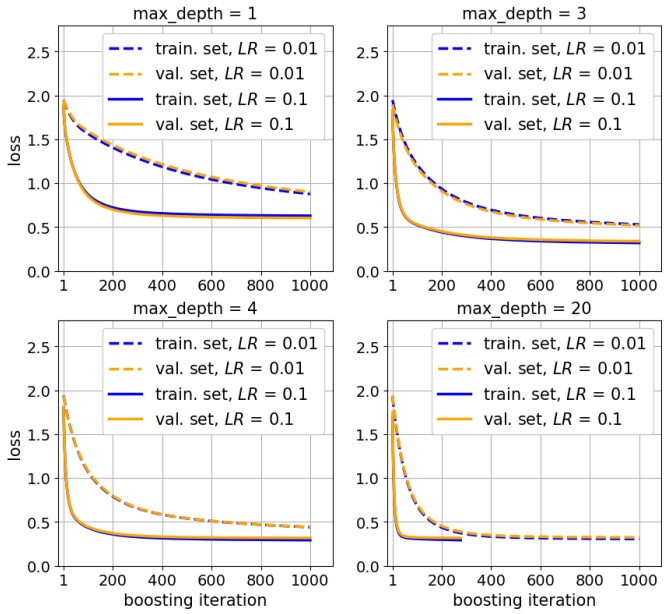

Figure 54: Training loss (blue curves) and validation loss (orange curves) over the boosting iteration ($MI = 1000$) for the cylindrical binning with $n_{\text{bins}} = 17$. Each subplot shows two pairs of curves and corresponds to a certain value of MD. Each pair of curves corresponds to a certain value of LR: $LR = 0.1$ (solid curves), $LR = 0.01$ (dashed curves).

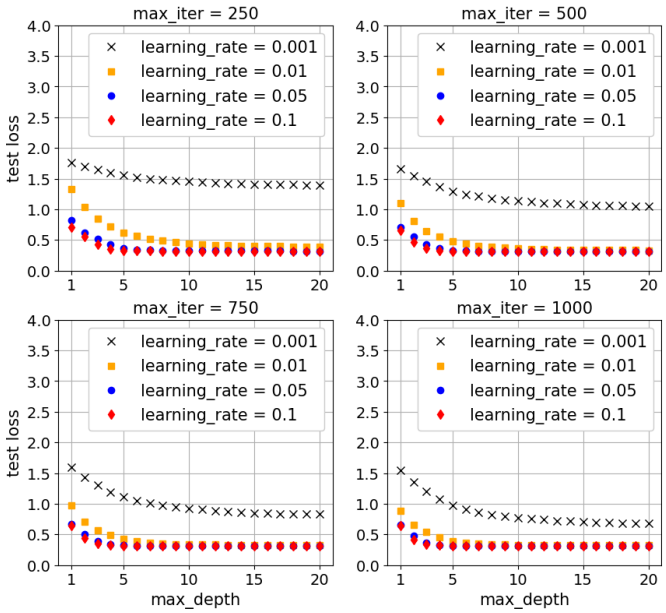

Figure 55: Calculated loss for the test set over MD for the cylindrical binning with $n_{\text{bins}} = 17$. Each subplot shows a set of four curves and corresponds to a certain maximum boosting iteration MI. Each curve within a subplot corresponds to a certain value of the learning rate LR.

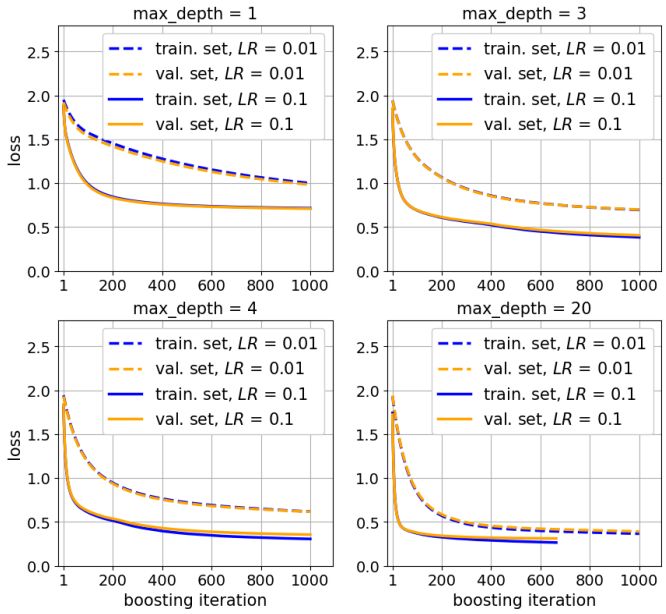

Figure 56: Training loss (blue curves) and validation loss (orange curves) over the boosting iteration ($MI = 1000$) for the cylindrical binning with $n_{\text{bins}} = 73$. Each subplot shows two pairs of curves and corresponds to a certain value of MD. Each pair of curves corresponds to a certain value of LR: $LR = 0.1$ (solid curves), $LR = 0.01$ (dashed curves).

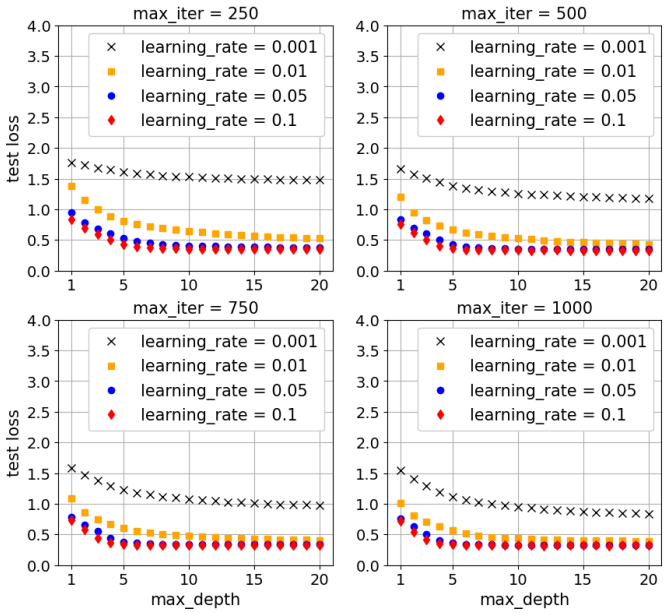

Figure 57: Calculated loss for the test set over MD for the cylindrical binning with $n_{\text{bins}} = 73$. Each subplot shows a set of four curves and corresponds to a certain maximum boosting iteration MI. Each curve within a subplot corresponds to a certain value of the learning rate LR.

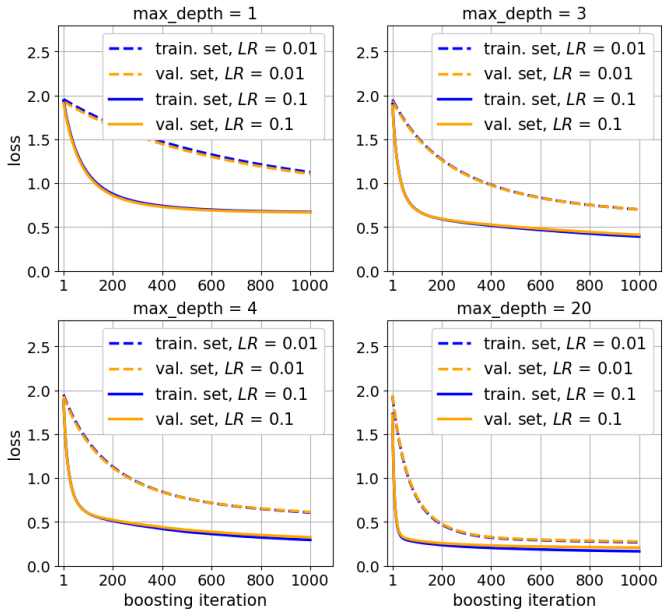

Figure 58: Training loss (blue curves) and validation loss (orange curves) over the boosting iteration ($MI = 1000$) for the hexagonal binning with $n_{\text{bins}} = 61$. Each subplot shows two pairs of curves and corresponds to a certain value of MD. Each pair of curves corresponds to a certain value of LR: $LR = 0.1$ (solid curves), $LR = 0.01$ (dashed curves).

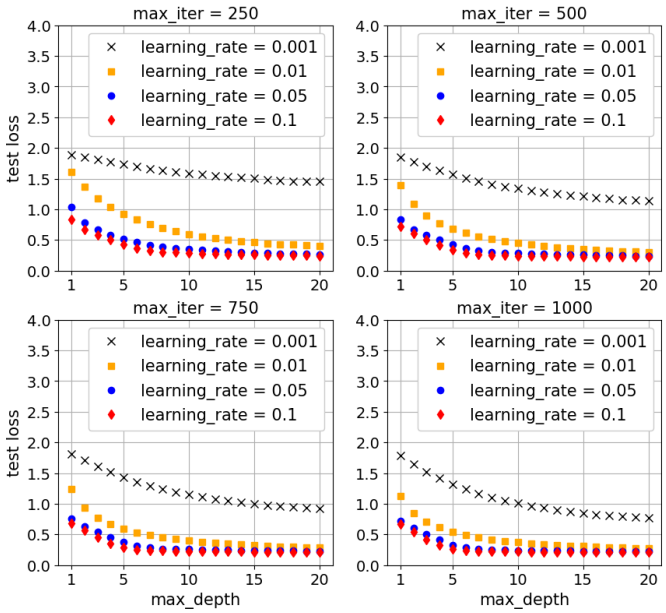

Figure 59: Calculated loss for the test set over MD for the hexagonal binning with $n_{\text{bins}} = 61$. Each subplot shows a set of four curves and corresponds to a certain maximum boosting iteration MI. Each curve within a subplot corresponds to a certain value of the learning rate LR.

A.1.3. Appendix - section 3.2 and section 3.2.1

hyperparameter	default setting
loss	'*log_loss*'
learning_rate	0.1
max_iter	100
max_leaf_nodes	31
max_depth	*None*
min_samples_leaf	20
l2_regularization	0.0
max_features	1.0
max_bins	255
categorical_features	'*warn*'
monotonic_cst	*None*
interaction_cst	*None*
warm_start	*False*
early_stopping	'*auto*'
scoring	'*loss*'
validation_fraction	0.1
n_iter_no_change	10
tol	*1e-07*
verbose	0
random_state	*None*
class_weight	*None*

Table 18: Default hyperparameter settings of the classification model as given by [9].

MI	combinations of *LR* and *MSL*
2000	all combinations of: $LR \in \{2.5 \cdot 10^{-2}, 5 \cdot 10^{-2}, 1 \cdot 10^{-1}, 2 \cdot 10^{-1}\}$, $MSL \in \{250, 500, 750, 1000, 1250, 1500, 1750, 2000\}$
6000	all combinations of: $LR \in \{2.5 \cdot 10^{-2}, 5 \cdot 10^{-2}, 1 \cdot 10^{-1}, 2 \cdot 10^{-1}\}$, $MSL \in \{5, 50, 100, 5000, 10000\}$
8000	all combinations of: $LR \in \{1 \cdot 10^{-2}\}$, $MSL \in \{5, 50, 100, 250, 500, 750, 1000, 1250, 1500, 1750, 2000, 5000\}$
200000	all combinations of: $LR \in \{1 \cdot 10^{-3}, 2.5 \cdot 10^{-3}, 5 \cdot 10^{-3}, 7.5 \cdot 10^{-3}\}$, $MSL \in \{750\}$, and at $(LR = 1 \cdot 10^{-2}, MSL = 10000)$

Table 19: Settings of the maximum boosting iteration *MI* for the final hyperparameter scans of *LR* and *MSL*.

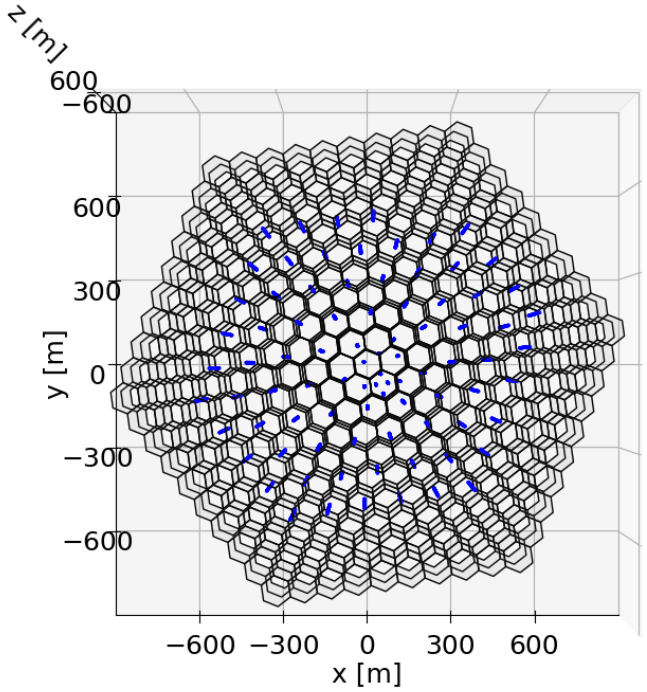

Figure 60: Top view of Figure 33.

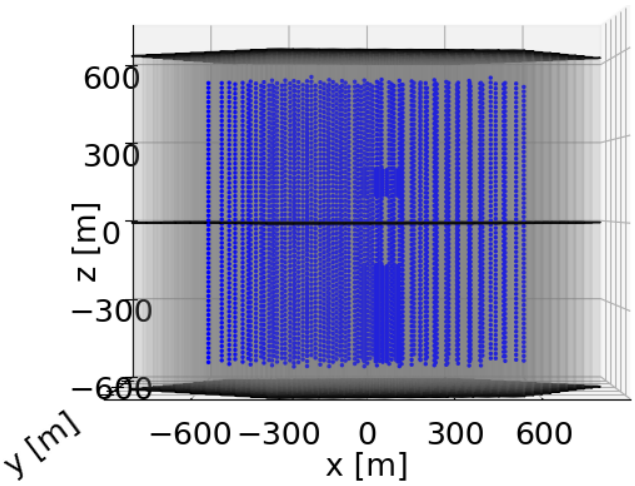

Figure 61: Side view of Figure 33.

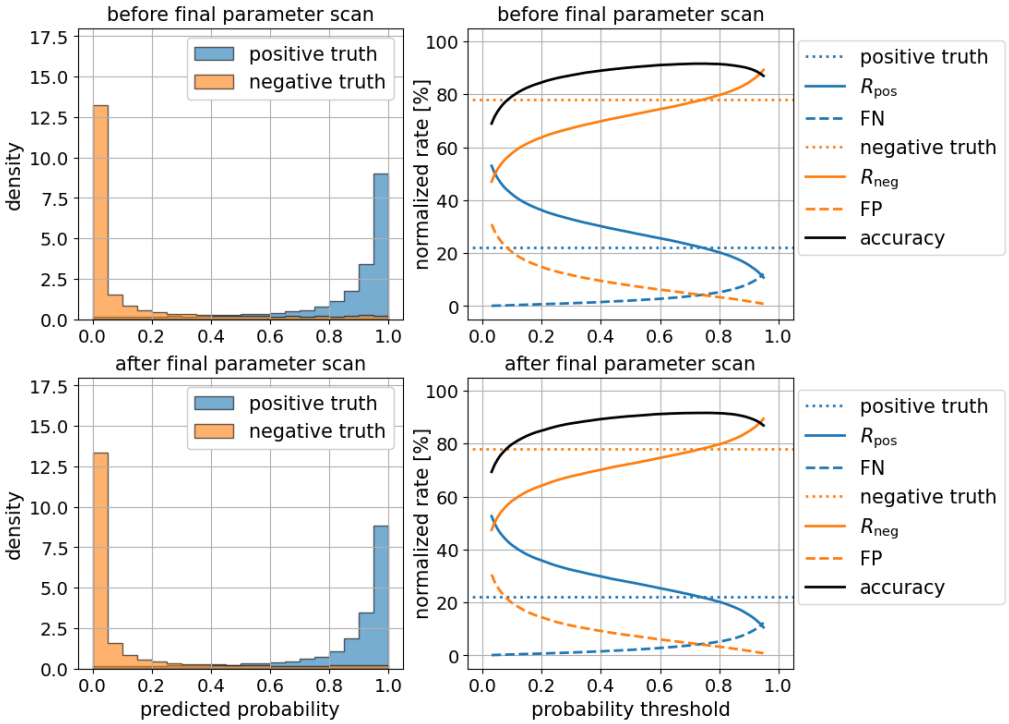

Figure 62: **Left:** Distribution of the predicted probabilities by the BDT classifier for actually positive (blue) and actually negative (orange) events. **Right:** Probability threshold scans as explained for Figure 22. The top and bottom row correspond to the BDT classification model before and after the final hyperparameter scan of *LR* and *MSL*.

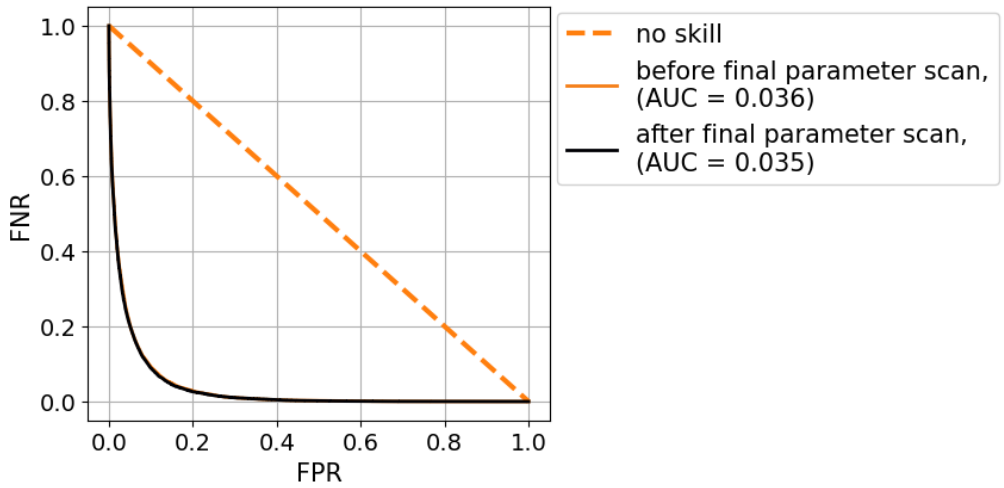

Figure 63: False negative rate (FNR) against false positive rate (FPR) with respect to the test set, for the BDT classification model before and after the final hyperparameter scan of *LR* and *MSL*. The respective area under the curve (AUC) is given in the legend.

A.2. Appendix - Optimized Muon Simulations

A.2.1. Appendix - section 4.1

simulation step	associated script
Muongun	/simprod-scripts/resources/scripts/muongun.py
CORSIKA	/simprod-scripts/resources/scripts/corsika.py
Polyplopia	/simprod-scripts/resources/scripts/SnowSuite/2-Polyplopia.py
Muon Propagation	/simprod-scripts/resources/scripts/SnowSuite/2-Propagate.py
Photon Propagation	/simprod-scripts/resources/scripts/SnowSuite/3-Snowstorm.py
Detector Simulation	/simprod-scripts/resources/scripts/detector.py
Filter Level 1	/filterscripts/resources/scripts/SimulationFiltering.py
Filter Level 2	/filterscripts/resources/scripts/offlineL2/process.py
environment shell	/cvmfs/icecube.opensciencegrid.org/py3-v4.4.1/icetray-env icetray/v1.14.0

Table 20: Location of the scripts within the IceTray GitHub repository [17], that were used for the execution of the respective simulation steps in the testing framework for optimized muon simulations [19]. All scripts were executed within the same IceTray environment shell provided at the bottom of the table. The input parameter configurations of the respective scripts can be found in the *config*.yaml file of [19]. The respective script parameter settings were chosen to be as close as possible to the original muongun dataset 23260 [18].

A.2.2. Appendix - section 4.2.1

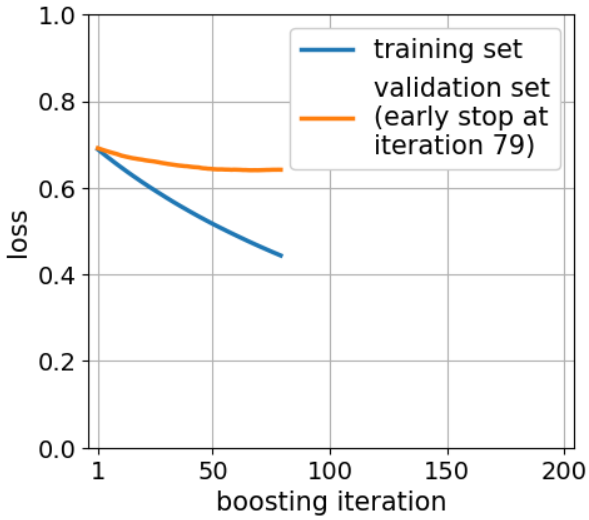

Figure 64: Training loss (blue curves) and validation loss (orange curves) against the boosting iteration obtained by training the BDT classifier during simulation i) (see Table 14).

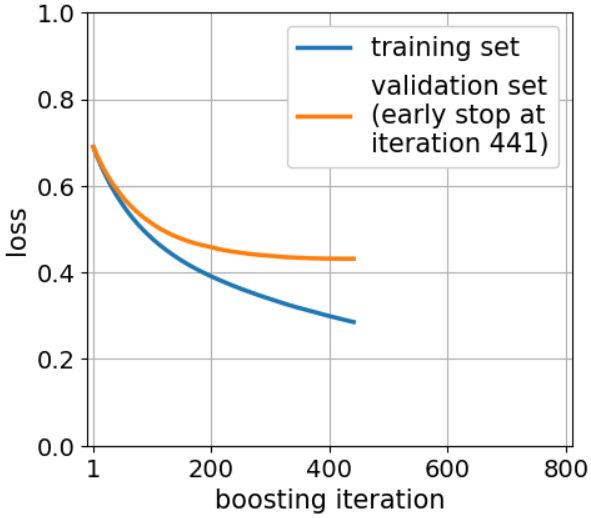

Figure 65: Training loss (blue curves) and validation loss (orange curves) against the boosting iteration obtained by training the BDT classifier during simulation ii) (see Table 14).

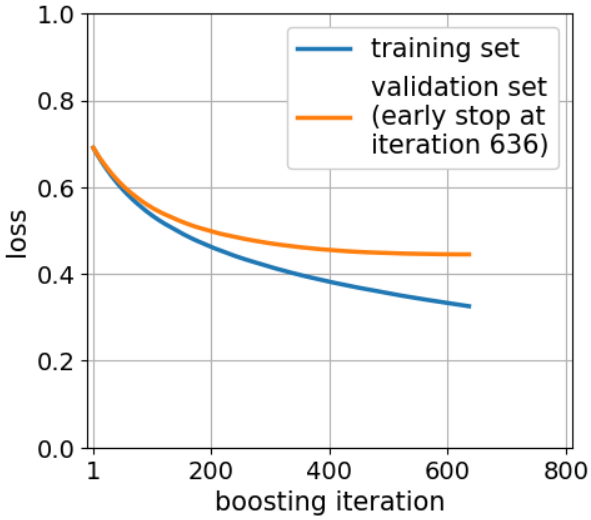

Figure 66: Training loss (blue curves) and validation loss (orange curves) against the boosting iteration obtained by training the BDT classifier during simulation iv) (see Table 14).

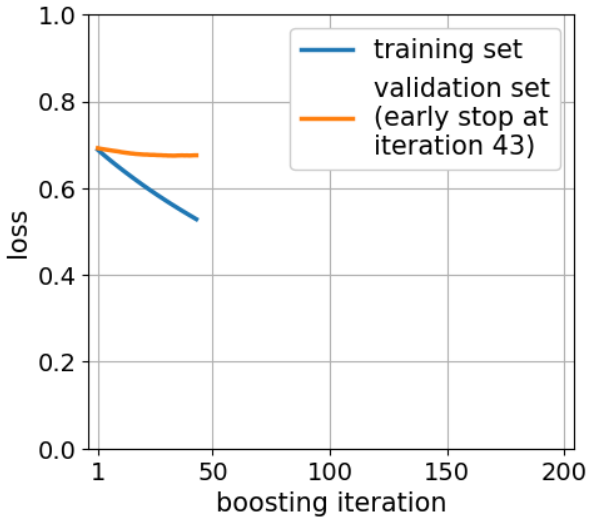

Figure 67: Training loss (blue curves) and validation loss (orange curves) against the boosting iteration obtained by training the BDT classifier during simulation v) (see Table 14).

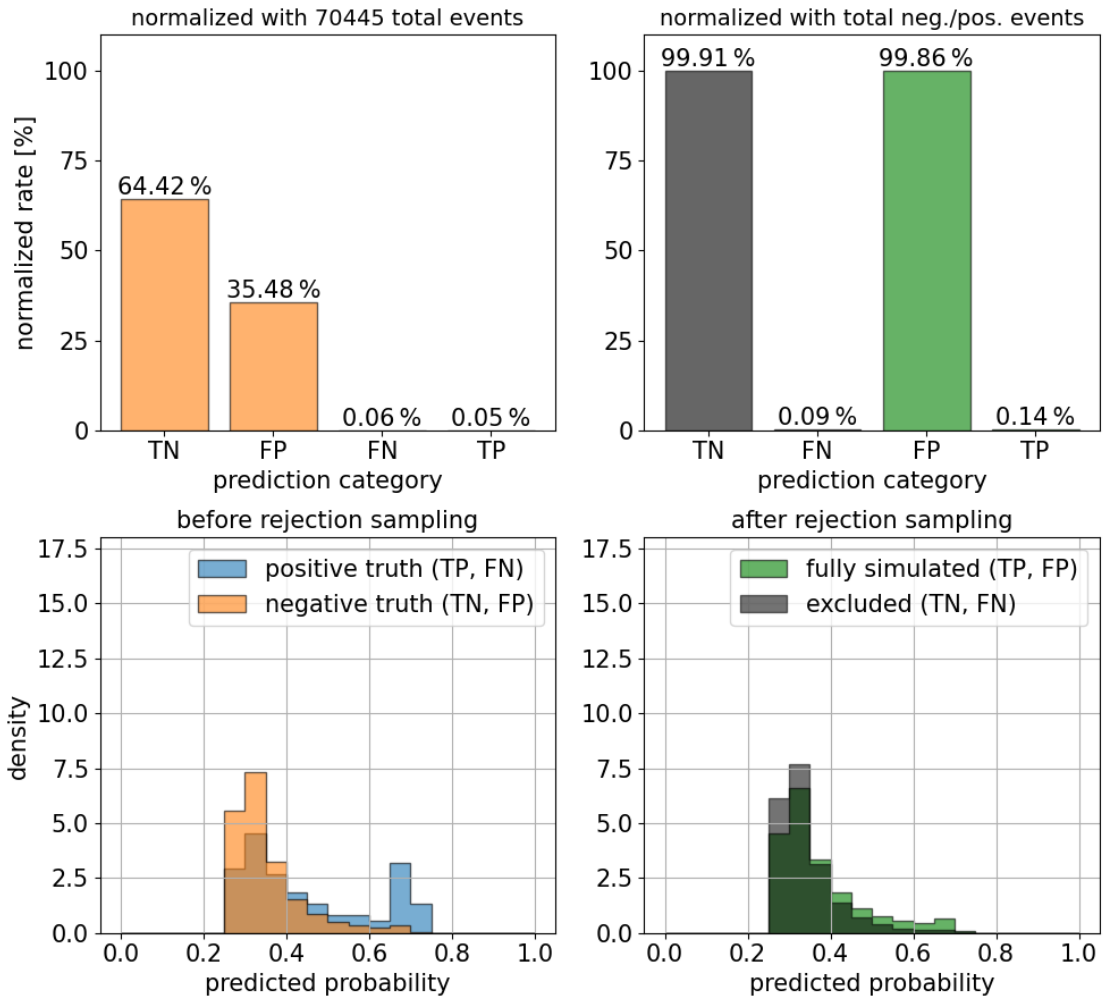

Figure 68: Results of sub-workflow B of simulation i) (see Table 14). Sub-events are excluded from the plots as explained in this section. **Top row:** Classification rates of the prediction categories (see Table 3) normalized with respect to the total number of events generated in sub-workflow B (left) and with respect to the total number of positive or negative predictions (right). **Bottom row:** Distribution of the predicted probabilities by the BDT classifier for actually positive (blue) and actually negative (orange) events before rejection sampling (left). Distribution of predicted probabilities for fully simulated (green) and excluded (black) events according to rejection sampling (right). Within the left or right column of the figure, the colour of each distribution indicates the associated prediction categories.

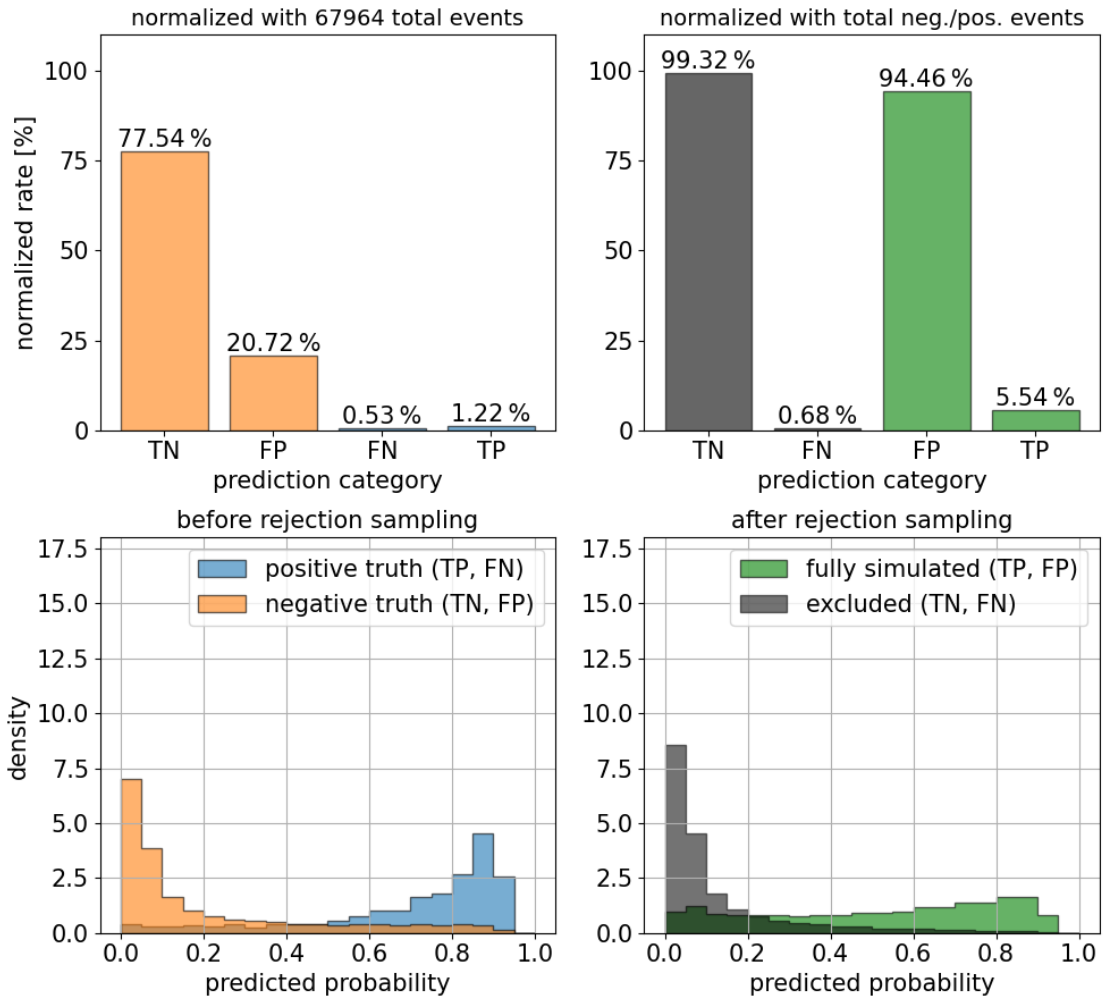

Figure 69: Results of sub-workflow B of simulation ii) (see Table 14). Sub-events are excluded from the plots as explained in this section. **Top row:** Classification rates of the prediction categories (see Table 3) normalized with respect to the total number of events generated in sub-workflow B (left) and with respect to the total number of positive or negative predictions (right). **Bottom row:** Distribution of the predicted probabilities by the BDT classifier for actually positive (blue) and actually negative (orange) events before rejection sampling (left). Distribution of predicted probabilities for fully simulated (green) and excluded (black) events according to rejection sampling (right). Within the left or right column of the figure, the colour of each distribution indicates the associated prediction categories.

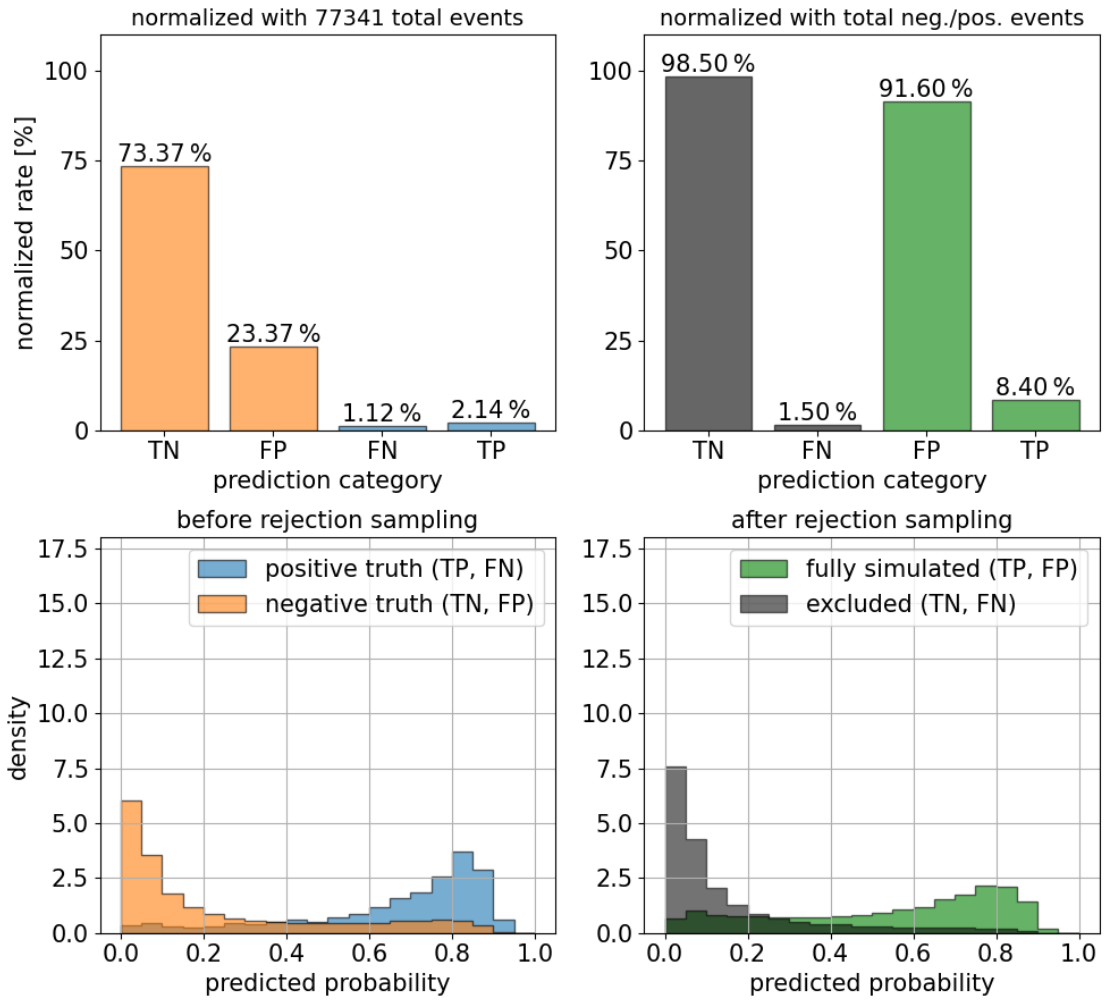

Figure 70: Results of sub-workflow B of simulation iv) (see Table 14). Sub-events are excluded from the plots as explained in this section. **Top row:** Classification rates of the prediction categories (see Table 3) normalized with respect to the total number of events generated in sub-workflow B (left) and with respect to the total number of positive or negative predictions (right). **Bottom row:** Distribution of the predicted probabilities by the BDT classifier for actually positive (blue) and actually negative (orange) events before rejection sampling (left). Distribution of predicted probabilities for fully simulated (green) and excluded (black) events according to rejection sampling (right). Within the left or right column of the figure, the colour of each distribution indicates the associated prediction categories.

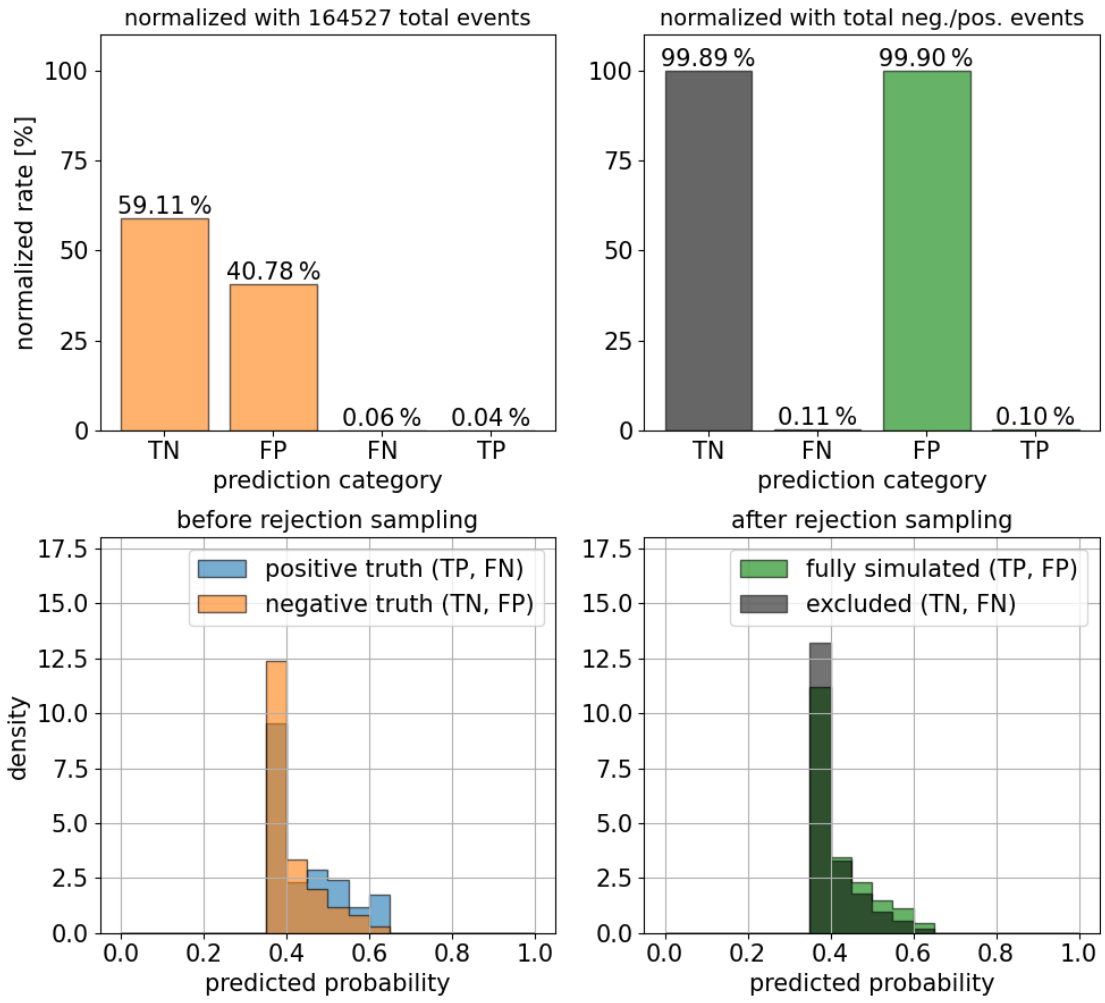

Figure 71: Results of sub-workflow B of simulation v) (see Table 14). Sub-events are excluded from the plots as explained in this section. **Top row:** Classification rates of the prediction categories (see Table 3) normalized with respect to the total number of events generated in sub-workflow B (left) and with respect to the total number of positive or negative predictions (right). **Bottom row:** Distribution of the predicted probabilities by the BDT classifier for actually positive (blue) and actually negative (orange) events before rejection sampling (left). Distribution of predicted probabilities for fully simulated (green) and excluded (black) events according to rejection sampling (right). Within the left or right column of the figure, the colour of each distribution indicates the associated prediction categories.

Acknowledgements:

I would like to thank all the members of ECAP for the welcoming working atmosphere, interesting meetings and seminars, as well as for many enjoyable lunch breaks, coffee breaks, and Daggerheart sessions. Especially, I want to thank all members of the IceCube working group and my office for their support and collaboration over the past year. Finally, I am grateful to my supervisors, Christian and Claudio, for their guidance throughout this thesis.

Declaration of Authorship:

I, Simon Koch, hereby declare that I have written the present thesis by myself and have not used any sources or aids other than those stated in the thesis. The present work does not substantially correspond to any other work, that has already been submitted to another examination authority.

Erlangen, ...

Date Signature