University of Warsaw
Faculty of Physics

Illya Yurchanka
Record book number: 450832

Identification of Λ hyperons from simulated Au+Au collisions at 10 GeV/nucleon in CBM spectrometer using TMVA Machine Learning package

Bachelor's thesis
in the field of Physics

The thesis was written under the supervision of
Dr hab. Krzysztof Piasecki
Institute of Experimental Physics
Nuclear Physics Division

Warsaw, July 2025

Summary

This thesis presents a study on the identification of neutral Λ hyperons from simulated Au+Au collisions at the beam energy of 10 GeV/nucleon using the CBM spectrometer at the FAIR facility. The UrQMD transport model and Geant-based detector simulation package were used. The candidates for Λ decays were reconstructed from $p\pi^-$ pairs, and their classification as signal or background was performed using the supervised machine learning methods available in the TMVA toolkit, including k-Nearest Neighbours (k-NN), Multilayer Perceptron (MLP), and Boosted Decision Trees (BDT). For the model comparison, signal significance and S/B (signal to background ratio) were used as main quality measures, while training and evaluation CPU time and mass reconstruction were also considered. The k-NN method demonstrated the highest performance in terms of signal detection and statistical confidence, while BDT showed a high signal-to-background ratio with fast inference time. MLP had the worst performance. However, it has considerable improvement potential for hyperparameter calibration to achieve better results.

Keywords

Hyperons, Heavy-Ion Collisions, Simulation, CBM, TMVA, Machine Learning, Classification, MLP, BDT, kNN

Title of the thesis in Polish language

Identyfikacja hiperonów Λ z symulowanych zderzeń Au+Au przy energii 10 GeV/nukleon w spektrometrze CBM za pomocą pakietu uczenia maszynowego TMVA

Contents

Goal of the Thesis

The goal of the bachelor's thesis is to consider different statistical learning methods for the identification of Λ hyperons from Au+Au collision at 10 GeV/nucleon in the CBM spectrometer. While the Compressed Baryonic Matter Experiment is still under construction, it is already possible to compare different methods with simulated data, thereby improving reconstruction accuracy and statistical significance in high-multiplicity environments.
The choice of Λ hyperons for this task was made due to their lack of electric charge and its . This baryon is detected only through reconstruction from its charged decay products. Their narrow natural width makes them suitable for benchmarking the performance of calibration and reconstruction methods, such as the machine learning [1].

Chapter 1

Theoretical Introduction

1.1. Physical Basics

1.1.1. Elementary Particles

The Standard Model

According to our knowledge, four fundamental interactions exist in our Universe: strong, weak, electromagnetism, and gravitation. While gravitation can only be observed in the macro-world, strong/weak interactions and electromagnetism are observed in the quantum world and described by **The Standard Model**. The interaction of particles occurs through these forces. All the particles can be divided into two groups: bosons and fermions. Bosons are particles whose spin quantum number (intrinsic angular momentum) is an integer value, and fermions are those with spin number equal to a half-integer value: 1/2, 3/2. Each of the three fundamental forces (*strong*, *weak*, *electromagnetism*) is described by the corresponding **gauge boson**. A gauge boson is a spin-1 particle. There is also a 0-spin (*scalar*) boson: **The Higgs boson**, which is responsible for the non-zero mass of the quarks in the Standard Model [2]. You can see all the gauge bosons in the table 1.1.

Table 1.1: Gauge-bosons and corresponding interactions.

Interactions	Boson	Spin
Strong	Gluon (g)	1
Electromagnetism	Photon (γ)	1
Weak	W boson ($W^{\pm}$)	1
	Z boson (Z^0)	1

This table should be complemented with fermions (see Table 1.2). Each quark has its unique quantum number (flavour), for example strange quark has strangeness $S = -1$. For each quark (or lepton), there exists an antiparticle with the same mass and spin, but an opposite charge and flavour (or lepton number) [2]. In this work, we will be focused on strong interactions, which are linked to the existence of quarks and gluons. Quarks constitute the basic bricks of matter. Objects such as protons and neutrons consist of combinations of up-quarks (u) and down-quarks (d) [3].

Table 1.2: List of fundamental particles.

Bosons			
	Strong Interactions	Electromagnetic Interactions	Weak Interactions
	Gluon (g)	Photon (γ)	W ($W^{\pm}$), Z (Z^0)
Fermions			
	I Generation	II Generation	III Generation
Leptons	electron (e^-) electrons neutrino (ν_e)	muon (μ^-) muons neutrino (ν_μ)	taon (τ^-) taons neutrino (ν_τ)
Quarks	up-quark (u) down-quark (d)	charm-quark (c) strange-quark (s)	top-quark (t) bottom-quark (b)

Hadrons

Only quarks can "feel" the strong force, and by the nature of QCD (Quantum Chromodynamics, the fundamental theory of strong interactions), quarks are never observed as free particles. In Nature, we can observe them only in combinations, which we call **hadrons**. The most common hadrons are protons and neutrons [2]. Two basic types of hadrons are mesons and baryons. A meson consists of quark and antiquark ($q\bar{q}$), whereas a baryon consists of three quarks (qqq) (or $\bar{q}\bar{q}\bar{q}$ for antibaryons). There are many possible hadronic states, which are defined by a combination of their quark flavours and different internal angular momentum states. The total angular momentum (spin) of a hadron depends on the orbital angular momenta between quarks and the combination of their spins. Hadronic states can be distinguished by their flavours, their total angular momentum (J) and their parity (P). Each state has its particular mass, which is not just a combination of the masses of its quarks but also results from interactions with the QCD vacuum. The only stable free hadron is the proton [2].

Λ, π^- and p hadrons

For this paper, we will only consider three hadron particles: neutral Λ baryon, π^- meson and proton. You can see the properties of those particles in tables 1.3 and 1.3

Table 1.3: Properties of hadrons considered in this work, values are approximate. See [3] for reference.

Hadron Name	Composition	Charge (e)	Mass (MeV/c^2)
Λ^0 baryon	uds	0	1115.68
π^- meson	$d\bar{u}$	-1	139.57
p	uud	$+1$	938.27

Hadron Name	Lifetime (s)	Dominant decay channel(BR)
Λ^0 baryon	$2.62 \cdot 10^{-10}$	$p + \pi^-$ (64.1%)
π^- meson	$2.60 \cdot 10^{-8}$	$\mu^- + \bar{\nu}_\mu$ (99.99%)

The lifetime of the Λ^0 baryon, about 10^{-10} s, is many orders of magnitude longer compared to the typical lifetime of hadrons decaying via strong interaction ($\sim 10^{-23}$ s). The reason is, the latter case, which requires strangeness conservation, is forbidden due to the additional requirement of baryon number conservation and the mass balance between particles.

1.1.2. Basics of Special Relativity

In 1905, in his fundamental work, "On the Electrodynamics of Moving Bodies", Albert Einstein, inspired by theoretical hints in electrodynamics and empirical evidence, such as the Michelson-Morley experiment, presented his two postulates [4]:

1. **The principle of relativity.** The laws of physics apply in all inertial reference systems.
2. **The universal speed of light.** The speed of light in a vacuum is the same for all inertial observers, regardless of the motion of the source of light.

Based on those two postulates, the Special Theory of Relativity was derived. This theory solved the incompatible Maxwell's equations of electromagnetism with Newtonian mechanics [4]. One of the most important conclusions of this theory is time dilation. Let's consider the problem from Introduction to Electrodynamics [4]: the bulb inside the moving car. A light ray leaves the bulb and hits the floor of the car. The time between the two of those events, from the point of view of the car, will take only:

$$\Delta\bar{t} = \frac{h}{c} \tag{1.1}$$

where h - height of car and c - speed of light. But for an observer standing on the street without moving, the time should take:

$$\Delta t = \frac{h^2 + (\Delta t v)^2}{c} \tag{1.2}$$

where v is - velocity of the car. Solving this equation yields:

$$\Delta t = \frac{h}{c}\frac{1}{\sqrt{1 - \frac{v^2}{c^2}}} \tag{1.3}$$

The time between the two events - (a) the light leaving the bulb and (b) the light hitting the centre of the floor - is measured differently by the two observers. The time interval observed on the cars' clock is shorter by a factor of:

$$\gamma \equiv \frac{1}{\sqrt{1 - \frac{v^2}{c^2}}} \tag{1.4}$$

Time dilation is a natural property that tells us that for the non-moving observer, the clock in the car ticks slower than inside the car [4]. This property is important in the field of elementary particles because, while unstable particles have a short life span at rest, when they are moving at a speed close to the speed of light, we can observe them for much longer than their nominal lifetime. Now, there are several questions in relativistic mechanics that must be addressed in this paper to ensure a proper understanding of future concepts.
In relative mechanic, coordinates (t, x, y, z) can be translated into coordinates $(\bar{t}, \bar{x}, \bar{y}, \bar{z})$ by using the Lorentz transformation:

$$\begin{pmatrix} \bar{t} \\ \bar{x} \\ \bar{y} \\ \bar{z} \end{pmatrix} = \begin{pmatrix} \gamma & -\gamma\beta & 0 & 0 \\ -\gamma\beta & \gamma & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix} \begin{pmatrix} t \\ x \\ y \\ z \end{pmatrix} \tag{1.5}$$

where $\beta = \frac{v}{c}$.
Within Special Relativity one can define the space-time interval, as follows:

$$s^2 = -(c\Delta t)^2 + x^2 + y^2 + z^2 \tag{1.6}$$

It turns out that the value of this interval is independent from the frame of reference, in which it is calculated. How do energy and momentum look in relativistic mechanics? While in classical physics the momentum is defined as $\vec{p} = m\vec{v}$, in relativistic mechanics we define it as:

$$\vec{p} = \gamma m\vec{v} = \gamma m\vec{\beta}c \tag{1.7}$$

Energy in the system's rest frame is:

$$E = mc^2 \tag{1.8}$$

And energy in a moving frame is given by:

$$E = \sqrt{(mc^2)^2 + (pc)^2} \tag{1.9}$$

Using these terms, one can define the "available energy" as follows:

$$s = \left(\sum_i E_i\right)^2 - \left(\sum_i p_i c\right)^2 \tag{1.10}$$

or, for a system consisting of a single body:

$$\sqrt{s} = \sqrt{E^2 - (pc)^2} = mc^2 \tag{1.11}$$

This term also turns out to have the same value regardless from frame of reference in which it is calculated.

1.1.3. Selected physics principles of particle detection in CBM

The general aim of particle detection consists of their measurement of their momentum, velocity, emission angles and energy loss when traversing through detectors.

Momentum measurement

A typical way of measuring the momentum of a charged particle consists of putting it in a deflective magnetic field and mounting several positional sensors along its path. A particle in the magnetic field $\vec{B}$ will curve its track in the plane perpendicular to $\vec{B}$, due to the Lorentz force:

$$\vec{F_L} \;=\; q\vec{v} \times \vec{B} \tag{1.12}$$

where q - is particle charge, v - velocity. Assuming for simplicity the constant B field, the magnetic force provides the centripetal force that keeps the particle in circular motion (in that perpendicular plane):

$$\gamma\frac{mv^2}{R} = qvB \tag{1.13}$$

Where R is the radius of curvature of the track. This leads to:

$$p = \gamma mv = qBR \tag{1.14}$$

A notable feature of equation 1.14 is the fact that it works regardless of whether the particle is relativistic or not. The reasoning above is only conceptual, as the CBM magnetic fields vary in different parts of the detector.

Velocity measurement

To detect particle velocity, we need to measure time. To do this, we use the Start and the Time of Flight detectors. We can place the Start detector in front of the target. When the beam hits Start, we measure the time of this hit - t_{Start}. Knowing the kinetic energy of the beam from the collider setup (from which we can extract its velocity - v_{beam}), we can calculate the moment when the beam hits the target - t_{Target}:

$$t_{\text{Target}} = t_{\text{Start}} + \frac{d_{\text{Start-Target}}}{v_{\text{beam}}} \tag{1.15}$$

where $d_{\text{Start-Target}}$ is the distance between the Start detector and the target.
For example, in the CBM setup, the Time of Flight detector was placed after the tracking detectors - MVD-STS-TRD station, which allows for measuring the positions of the particle and reconstructing the track of a particle. The ToF detector measures t_{ToF} with high precision. Having both t_{Target} and t_{ToF}, it is possible to find the time difference for which particle traverses the target-ToF distance. Having reconstructed the track from particle hits in the MVD-STS-TRD station allows us to integrate the particle trajectory to obtain the path (L) between the target and the ToF detector. Both the path and the time of the particle give us the velocity of the particle:

$$v = \frac{L}{t_{\text{ToF}} - t_{\text{Target}}} \tag{1.16}$$

Mass determination

Having both velocity and momentum over charge, it is possible to find the mass of the particle. After putting equations 1.16 and 1.14 together, we can extract the mass. Usually, the relevant formula is kept in the squared form:

$$\left(\frac{m}{q}\right)^2 = \left(\frac{BR}{c}\right)^2 \left(\left(\frac{tc}{L}\right)^2 - 1\right) \tag{1.17}$$

Such a form originates from the finding that due to experimental uncertainties, some rare fraction of particles can appear to have $\beta > 1$. For completeness, their tracks are also stored, but with masses effectively negative.
A $\frac{m}{q}$ variable derived from equation 1.17 can be used to classify particles. The drawback of this method is that it is not possible to separate particles with the same mass/charge ratio. However, for the needs of the paper, it is not significant, due to its focus on hadrons with no similar mass/charge ratio. Both proton and pion have $|q| = 1$.

1.1.4. Reconstruction Methods

The method provided above works for charged particles. But it's useless if applied directly to neutral particles such as Λ^0. The solution to this problem can be found in the case of a particle that has a decay channel where all the products can be detected (for Λ^0 it is $p + \pi^-$). The mere detection of such particles does not provide sufficient evidence to classify them as decay products, as their origin may also stem from other background processes, not just neutral particle decay. Rigorous background suppression is often required to isolate the true signal and ensure unambiguous identification. One of the most basic techniques is to construct the invariant mass distribution M_{inv} for all the possible combinations of decay products and check whether there is a peak around the mass of the conjectured mother-particle (in our example:

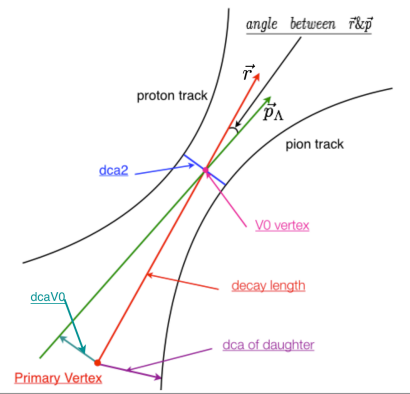


Figure 1.1: Reconstruction of $\Lambda \to p\pi^-$ decay. [5]

M_Λ), which would indicate the presence of the mother-particle. Additionally, a rejection can be more precise by reconstructing **the primary vertex** and **the particle decay vertex**. One can see an example of reconstruction in Fig. 1.1. The primary vertex is the spatial point where the collision of the beam and the target nuclei occurs. It can be assumed that a neutral particle was produced in the primary vertex, and the spatial point of its decay is the particle decay vertex.
The primary vertex can be reconstructed by using tracks of other, more stable, collision products. For reconstructing the particle decay vertex, the trajectories of decay products are extrapolated backwards toward the primary vertex, and the decay point is identified as the position where the distance between these tracks reaches its minimum (distance of closest approach - `dca`). In the Fig. 1.1, the particle decay vertex is denoted as V_0. We can analyse all of this data to classify our candidates as signal or background (all parameters that will be used in filters can be seen in Fig. 1.1):

- The vertex of the V_0 decay should be sufficiently distant from the primary vertex → filter applied: `decay length > min`
- In the fixed-target experiment, the velocity of the centre of mass reconstructed particle is directed forward → filter applied: $\vec{V_0} - \overrightarrow{PV}\Big|_z > 0$, where PV - Primary Vertex.
- To minimise that decay product candidates are coming from different vertices → filter: `dca2 < max`
- The tracks of the decay products should not converge at the primary vertex, because that would indicate that those particles are products of the original collision → filter: `dca of daughter > min`
- The reconstructed momentum of the V_0 candidate should point back to the primary vertex → filter applied, which should indicate that the reconstructed particle originates from the initial reaction: `dcaV0 < max`

While for the true products of the particle decay `dca2` and `dcaV0` are usually greater than zero, due to the finite resolution of the detectors, it is important to find a proper minimum value to cut off background. If the presence of the Λ particle was determined, all the properties of this particle can be found and analysed.

1.2. Machine Learning Methods

1.2.1. Introduction

Statistical learning is a set of mathematical and computational tools for understanding data. It is one of the most important tools in many areas of science. We can use statistical learning to predict future events based on data and the outcome of previous events, to classify objects based on their parameters or to find a correlation between inputs. One of the most common use cases for statistical learning is [6]:

- predicting weather
- identifying objects on a digitised picture
- detecting spam and flagging it in our email box

Physics is no exception. Statistical learning is used in almost all fields of physics, but one of the most basic cases of using statistical learning is experimental data analysis.

1.2.2. Supervised learning and Classification

All the statistical learning problems can be roughly categorised as either **supervised** or **unsupervised**. One can use unsupervised learning to find the associations and patterns among a set of input parameters. For this category, no outcome measure needs to be provided for the learning process. Supervised learning is used to predict an outcome measure based on input measures. There is a need for the presence of already known outcome measures for a set of training input data to guide a learning process. For this work, we will focus on supervised learning [6]. In that category, there are two types of output:

- **regression** - prediction of quantitative output (*example:* predicting the price of a house based on its features)
- **classification** - prediction of qualitative output (*example:* iris flower classification {Versicolor, Setosa or Virginica} based on its features)

An input can be a qualitative or quantitative variable, or a mix of them [6]. That is why there is no universal prediction method for every problem. For each problem, there is a need to find and tune the optimal prediction method. There are many such methods in modern statistical learning, but for this paper, only three of them will be used: **k-Nearest Neighbours Algorithm**, **Multilayer Perceptron** and **Boosted Decision Tree**. The descriptions of these methods are based on The Elements of Statistical Learning [6] and TMVA - Toolkit for Multivariate Data Analysis [7].

1.2.3. k-Nearest Neighbours Algorithm

k-Nearest Neighbours Algorithm (The k-NN) is one of the most popular classification algorithms. The k-NN principle of work is quite simple, but very powerful. It is used in the classification of handwritten digits, satellite image scenes and many other cases.

Principle of work

This method employs supervised learning, so to analyse a real-data dataset, there is a necessity for a training dataset. Both the training sample and the one with real data consist of **data points**. Each data point is a vector of features (feature vector), where each vector component corresponds to a specific particle parameter. The difference between those samples is that each element in the training dataset is pre-classified. The class is quantified by some value y_i. For example, if we want to distinguish the signal from the background:

$$\begin{cases} y_i = 1, & \text{signal} \\ y_i = 0, & \text{background} \end{cases} \tag{1.18}$$

To classify a real data point x_0 we shall find the k closest training points $x_{(i)}$ $(i = 1, 2, \ldots, k)$. For every x_0, we find its distance to any other one by using Euclidean distance in feature space:

$$d_{(i)} = ||x_{(i)} - x_0|| \tag{1.19}$$

The point x_0 shall be assigned the class that represents the majority among its k nearest neighbours. The vote Y for the point x_0 is defined as follows:

$$Y = \frac{1}{k} \sum_{x_i \in N_k(x)} y_i \tag{1.20}$$

where $N_k(x)$ is the neighbourhood of x_0 defined by the k closest points x_i, y_i is class of x_i neighbour. All the data points with $Y > Y_b$ are assigned to the target class. By setting the boundary value $Y_b = 0.5$, the majority class among the neighbours (in the training dataset) determines the predicted class of the real data point. However, in some cases, the boundary value Y_b may need to be set higher than 0.5.

The more sophisticated variant of the k-NN equation 1.20 involves assigning weights (w_i) to each neighbour. An updated formula has the following form:

$$Y = \frac{1}{k} \sum_{x_i \in N_k(x)} \frac{1}{w_i} y_i \tag{1.21}$$

Advantages and limitations

The best use case for the k-NN method is when the boundary that separates classes has an irregular shape that can not be approximated by other learning methods. But this method is not ideal: a considerable problem of the k-NN method is the *curse of dimensionality* [8]. For example, for a 10-dimensional hypercube, with uniformly distributed inputs, to capture 1% or 10% of the data volume to form a local average, we must cover 63% or 80% of the range of each input variable (expected edge length, for a fraction of unit volume r: $e_p(r) = r^{1/p}$ [6], which is no longer local. There are other manifestations of the *dimensional curse* for the k-NN method, but covering all of them would exceed the scope of the paper.

1.2.4. Neural Networks: Multilayer Perceptron

The term **neural network** describes a large class of models and learning methods [6]. A neural network consists of neurons (nodes) that can send signals to one another through their connections (edges). Depending on the architecture of a network, it can be applied to regression or classification.

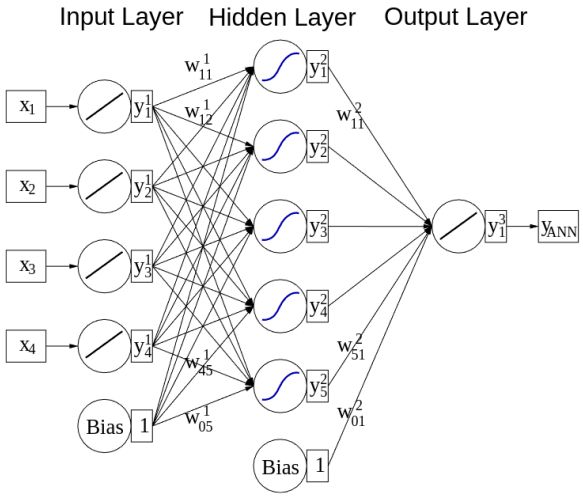


Figure 1.2: chematics of a single hidden layer perceptron (see text for details) [7].

In this paper, we will focus on **perceptron neural networks**. This class of neural networks combine nodes into a set of layers, and nodes can only interact with ones from another layer. The first layer of a perceptron neural network is always the input, while the last one is - output. Layers between them are called **the hidden ones**. A perceptron is a feed-forward neural network, which means that data flows unidirectionally (input $\rightarrow$ hidden $\rightarrow$ output), with no cycles or feedback connections. Multilayer Perceptron neural networks (the MLP) are a subclass with more than one hidden layer.

Principle of work

For clarity, the Perceptron will be described as a network with one hidden layer. The scheme of work is shown in Fig. 1.2. It starts with the vector X, which contains all the input data. The hidden layer, consists of M nodes, where the m-th node ($m = 1, \ldots, M$) works the following way:

- The linear combination L_m is computed from the input values, weighted by their respective coefficients: a bias term α_{0m} and a weight vector α_m.

$$L_m = \alpha_{0m} + \alpha_m^T X \tag{1.22}$$

  These weights are found in the course of a training. It will be explained in the next subsection.

- The activation function σ is applied to L_m, which flattens $\mathbb{R}$ to [-1, 1].. A typical solution here is the hyperbolic tangent [7].

$$Z_m = \sigma(L_m) = \tanh(L_m) \tag{1.23}$$

The output is modelled as the linear combination T of the Z_m responses from the nodes of the hidden layer in the MLP neural network:

$$T = \beta_0 + \beta^T Z \tag{1.24}$$

where β_0 is a bias term and β is a vector of weights, also found during training.

Training of the neural network

As it was previously shown, each node has its weights, and these parameters of the neural network are unknown in the beginning. Adjusting the set of weights (θ) is called the task of training the neural network. For classification, either the sum-of-squared error Eq. 1.25 or cross-entropy Eq. 1.26 is used as a measure of fit quality $R(\theta)$.

$$R(\theta) = \sum_{i=1}^{N} (y_i - f(x_i))^2 \tag{1.25}$$

$$R(\theta) = -\sum_{i=1}^{N} \log(f(x_i)) \tag{1.26}$$

where N is the number of training examples in dataset, y is an output and x is an input. Back-propagation is one of the most common ways of minimising $R(\theta)$ and is based on gradient descent.

Advantages and limitations

Neural networks are one of the best methods to model complex, non-linear relationships in data [7]. They can analyse different types of data and learn automatically. But they require big sets of training data and can easily suffer from biases of the data, over-fitting, etc. In addition, neural networks suffer from a black-box nature. We can only control the number of hidden layers and nodes, but we can't control the decision processes inside them.

1.2.5. Boosted Decision Tree

Decision Tree is quite simple concept. By comparing input parameters against previously calculated threshold values at each node, we can traverse the tree until reaching a terminal node that provides the predicted output. This method is visually represented in Figure 1.3. However, a drawback of this method is the fact that it is considered a "weak" method. Yet, if we combine multiple "weak" methods, it is possible to achieve a reasonably good prediction. Based on this idea, **boosting methods** were created. The simple boosting method combines "weak" methods with their adjusted weights and yields a prediction. A Boosted Decision Tree is a boosting classification method that uses a combination of decision trees to classify an event.

Principle of work

Decision Tree

For simplicity, the binary decision tree will be considered. It can be mathematically described as follows: the feature space is partitioned into a set of rectangles (**nodes**), in each of which some simple model is fitted [6]. It begins with splitting space into two regions, then we split each one of them into two more, and this process is continued until the stopping rule is applied. Each time we are looking for the split-point, which helps us achieve the best fit. This process is called **tree growing**.
For a classification problem with K input values, to grow the tree, we recursively partition the feature space by selecting splits that minimise node impurity Q_m. Node impurity is a measure of how mixed the data at a particular node is concerning the target variable. There are different measures of node impurity, but for all of them, there is a need to know

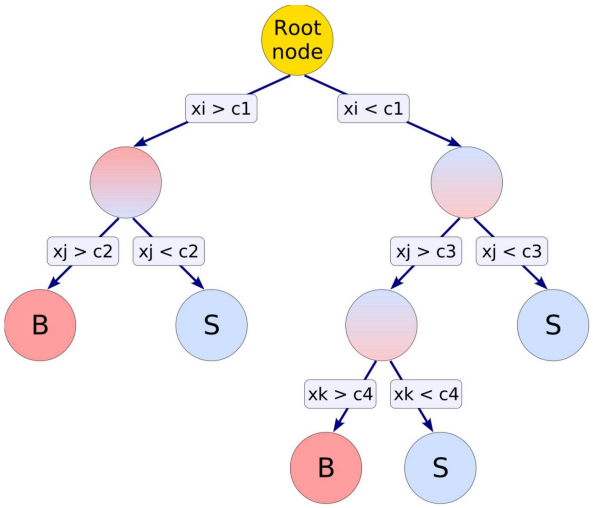


Figure 1.3: Example of decision tree [7].

the proportion of the class for each node. If R_m is m-th region of feature space and N_m observations have features falling into this region, then class k ($k = 0$ for background and $k = 1$ for signal) has the proportion $\bar{p}_{mk}$ in node m:

$$\bar{p}_{mk} = \frac{1}{N_m} \sum_{x_i \in R_m} \delta(y_i - k) \tag{1.27}$$

where y_i is the class of the i-th data point and x_i is feature vector of i-th data point. Then node m gets class $k(m) = \arg\max_k \bar{p}_{mk}$. Having the proportion $\bar{p}_{mk}$ in node m for each class and class of node m, we can find the node impurity. Common ways of finding node impurity are [6]:

- **Misclassification error:** $\frac{1}{N_m} \sum_{x_i \in R_m} (1 - \delta(y_i - k)) = 1 - \bar{p}_{mk(m)}$
- **Gini index:** $\sum_{k \neq k'} \bar{p}_{mk} \bar{p}_{mk'} = \sum_{k=1}^{K} \bar{p}_{mk}(1 - \bar{p}_{mk})$

The Gini index is more suited to numerical optimisation (because it is differentiable) and is more sensitive to changes in the node probabilities in comparison to the misclassification rate. For this reason, the Gini index is usually used for tree growing [6].

Boosting

One of the most used boosting algorithms is Adaptive Boost (AdaBoost) [7]. In the training process, data points that were misclassified in the previous tree are given a higher weight in the following tree. The weights of misclassified data points are getting bigger by multiplication by a common boost weight α:

$$\alpha = \ln\left(\frac{1 - \text{err}}{\text{err}}\right) \tag{1.28}$$

Where $\text{err} = \frac{1}{N} \sum_{i=1}^{N} (1 - \delta(y_i - G(x_i)))$ is the weighted error rate of the previous tree classifier ($G(x_i)$). After updating weights, they are renormalised so that the sum of weights remains constant. This forces subsequent trees to focus on previously misclassified observations. The

boosted classification is then given by the sum of classifiers with their weights:

$$G(x) = \frac{1}{T}\left[\sum_{t=1}^{T} \alpha_t G_t(x)\right] \tag{1.29}$$

where T is the number of trees, in the end, we are getting input between -0.5 and 0.5. Values closer to 0.5 indicate high confidence for *signal*, while values closer to -0.5 indicate high confidence for *background*.

Advantages and limitations

The main advantage of Boosted Decision Tree is working "out of the box"; it doesn't require a lot of tuning to get a good result, but its theoretically best performance will be inferior to more complex methods such as Neural Networks. That's why the best use case for Boosted Decision Tree is to analyse new real-time data, for which we do not have enough training data and time to fine-tune our method [7].

Chapter 2

CBM - description of setup and data generation

2.1. Overview of the CBM Experiment

The currently built Facility of Antiproton and Ion Research is going to be one of the key research facilities worldwide. The main component of FAIR is the SIS100 ring accelerator. It has a circumference of 1,100 meters and can accelerate the ions of all the natural elements in the periodic table to velocities as high as 99% of the speed of light. The Compressed Baryonic Matter (CBM) will be part of the FAIR. It is a fixed-target experiment designed to explore the QCD phase diagram in the region of high net-baryon densities [9]. It is designed for interaction rates up to 10^7 Hz to enable measurements of rare observables.

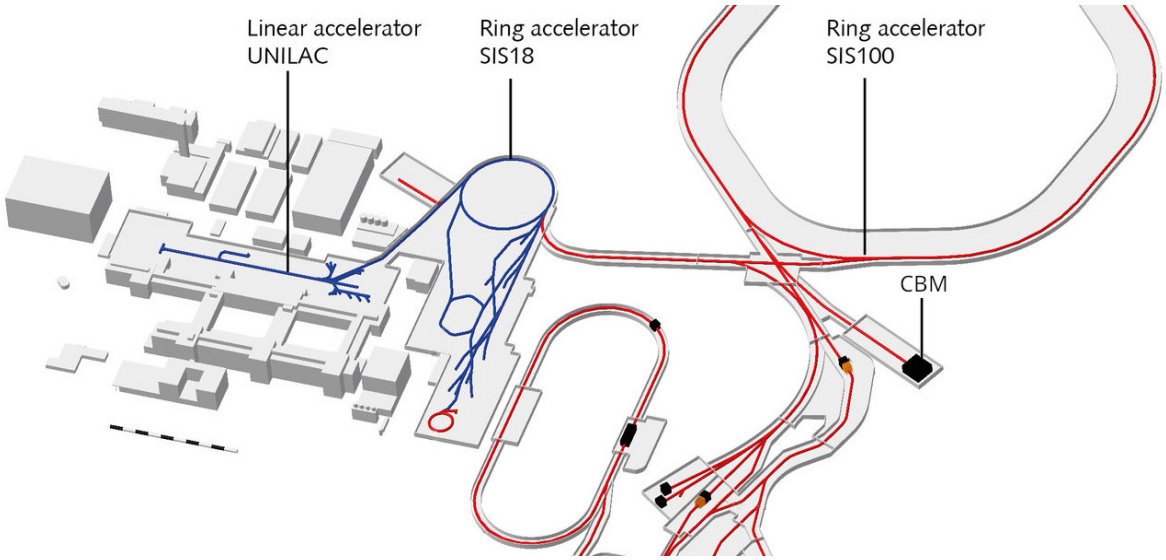


Figure 2.1: FAIR scheme with CBM

The CBM experiment is a combination of fast and precise detectors, a powerful data read-out and analysis software, and high-performance front-end electronics (FEE).

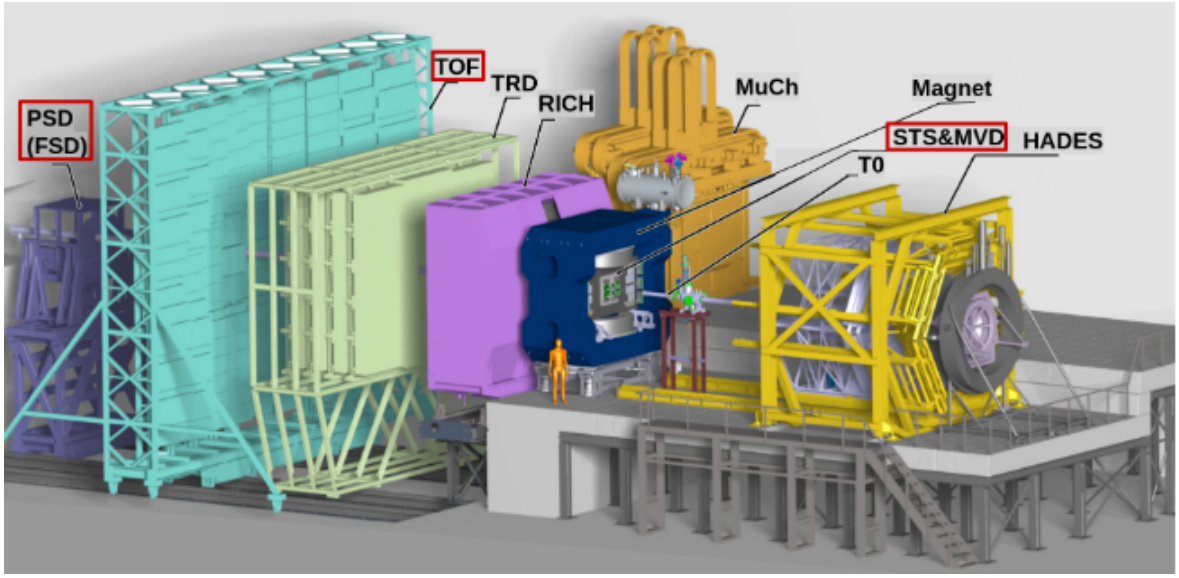


Figure 2.2: CBM experiment design.

2.2. Experimental setup

As can be seen on Fig. 2.2, CBM consists of the following subsystems:

- Micro Vertex Detector (MVD);
- Silicon Tracking System (STS);
- superconducting dipole magnet
- Ring Imaging Cherenkov (RICH);
- Muon Chamber system (MuCH);
- Transition radiation detector (TRD);
- Time-of-Flight wall (ToF);
- Forward Spectator Detector (FSD).

Micro Vertex Detector

The Micro Vertex Detector (MVD) is the first detector that a particle traverses from the target. It is used for detecting secondary vertices with high precision (mostly utilised for D-meson reconstruction). As can be seen in Fig. 2.3, MVD consists of four layers of ultra-thin and highly granular Monolithic Active Silicon Pixel Sensors. They are located between 8 cm and 20 cm from the target in the traditional tracker configuration [10]. MVD works in conjunction with STS, and they both have an angular acceptance within $\theta \in [2.5°, 25°]$ [10][11]. For MVD, the required spatial resolution of extremely granular pixels, for observers who are reaching the targeted secondary, is around 5 μm [10].

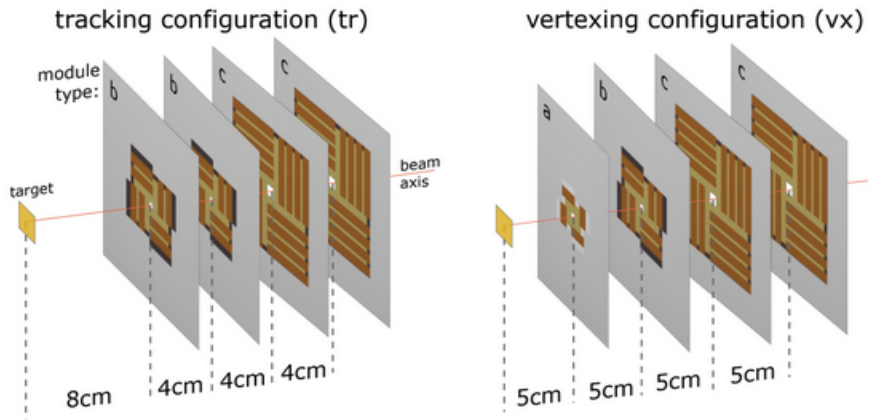


Figure 2.3: Scheme of the positional configuration of the 4 layers of the Micro-Vertex Detector (MVD) [10].

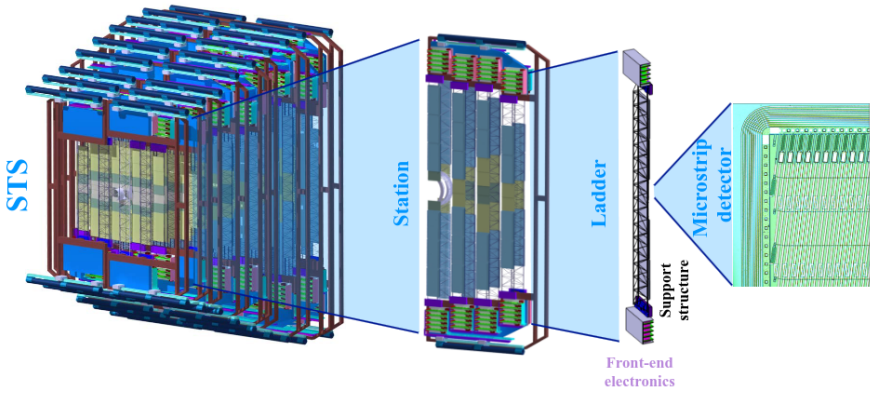


Figure 2.4: Sketch of the STS detector of the CBM experiment [12][11].

Silicon Tracking System

The Silicon Tracking System (STS) is the most important part of the experimental setup. STS consists of 8 low-mass silicon micro-strip detectors, which can be seen in Fig. 2.4. They are located between 30 cm and 100 cm from the target. The charged product of the collision produces a lot of electron-hole pairs when it passes through the active volume of the detector. The application of reverse bias voltage creates pair separation, with electrons and holes drifting toward the n- and p-type regions, respectively. The resulting charge migration induces a current pulse, detected at the read-out electrodes. This pulse gives a precise position of the charged particle (in STS, the typical hit resolution is of the order of 25 μm). Having these positions at different moments of time, it is possible to reconstruct and extrapolate the charged particle track. STS is a powerful detector that allows track reconstruction in a wide momentum range from about 100 MeV up to more than 10 GeV with a momentum resolution of about 1.5% [11].

The superconducting dipole magnet

For momentum detection, both MVD and STS were positioned within the superconducting dipole magnet, which measures 144 cm in height and 300 cm in width. A schematic of this

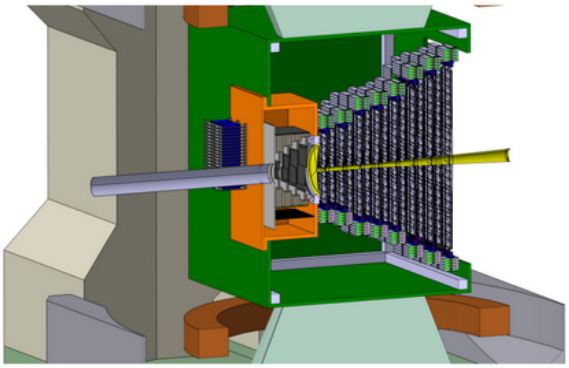


Figure 2.5: Detailed view of the STS isolation box (green), MVD vacuum vessel (orange), beam pipe (yellow) and target inside the dipole magnet [11].

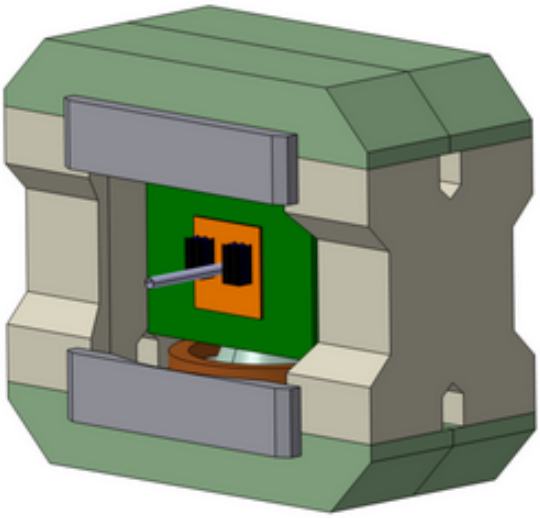


Figure 2.6: View of STS and MVD systems inside magnet [11].

construction can be seen in Fig. 2.5 and Fig. 2.6. The generated magnetic field is not homogeneous, primarily aligned along the vertical axis, with a peak magnitude of approximately 1 T [11].

The two following systems, RICH and MuCH, are used interchangeably, being mounted on a rail.

Ring Imaging Cherenkov detector

The main task of the Ring Imaging Cherenkov detector (the RICH detector) is the identification of electrons and positrons with momenta up to 10 GeV, to separate them from pions. The physical working principle is based on the Cherenkov radiation. Particles that move in a medium at velocities higher than the speed of light in a medium emit photons. This emission forms a cone-shaped front which is spread at an angle θ. This angle depends on the velocity

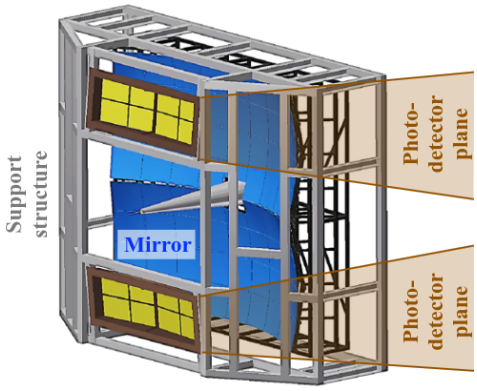


Figure 2.7: The RICH detector geometry [13].

of a particle in a medium (v) and the speed of light in a medium (c/n):

$$\cos(\theta) = \frac{c}{nv} \tag{2.1}$$

These emitted photons fall on mirrors, where they form rings, with radius dependent on angle θ. Having position-sensitive photon detectors, it is possible to reconstruct those rings and therefore the angle θ, from which we can get the particle velocity. Having velocity and momentum, we can find the particle's mass and identify this particle.
The RICH detector uses CO_2 as a medium and is located 1.6 m from the target [13].

The Muon Chamber system

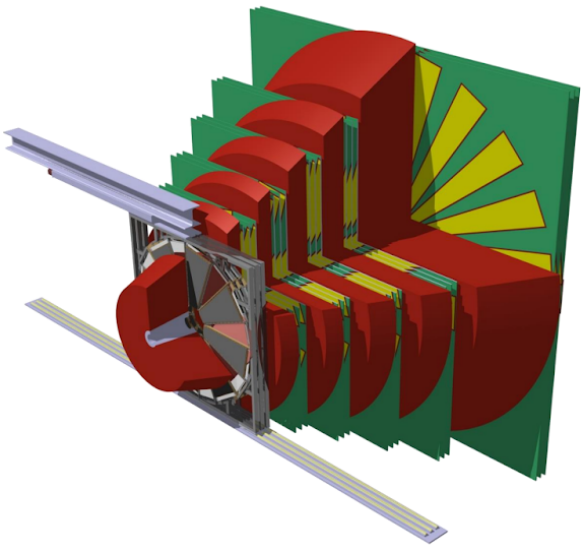


Figure 2.8: Two views of MuCH mechanics in SIS100 configuration [14].

The Muon Chamber system (MuCH) is used for the identification of low-momentum muons in an environment of high particle densities. It is mounted as an alternative to the RICH detector. The MuCH detectors track particles through a hadron-absorber system and perform a momentum-dependent muon identification. The MuCH system uses hadron absorbers separated into several layers, whereas tracking detectors are placed between absorbers. The first

hadron absorber consists of carbon, while the rest of them consist of iron. Tracking planes are based on Gas Electron Multiplier (GEM) and Multigap Resistive Plate Chambers (MRPC) technologies [14]. You can see MuCH schematic in Fig. 2.8.

Transition Radiation Detector

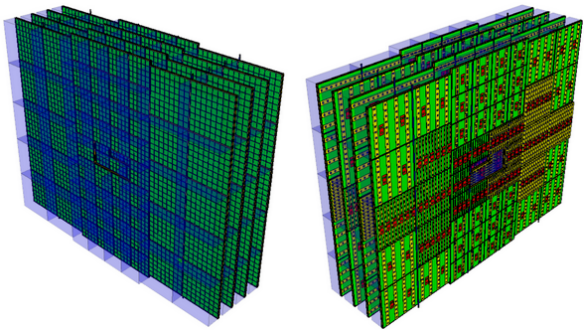


Figure 2.9: The TRD station layout [15].

The Transition Radiation Detector (TRD) is used for the identification of electrons and positrons with momenta larger than 1.5 GeV/c. The physics working principle is based on the fact that charged particles, when crossing the boundary between two media with different refractive indices, emit transition radiation. The total energy loss depends on the Lorentz factor: $\gamma = \frac{E}{mc^2}$. The TRD detector is located between the STS and TOF detectors and can be used as an additional tracking detector [15], its layout can be seen in Fig. 2.9.

Time of Flight detector

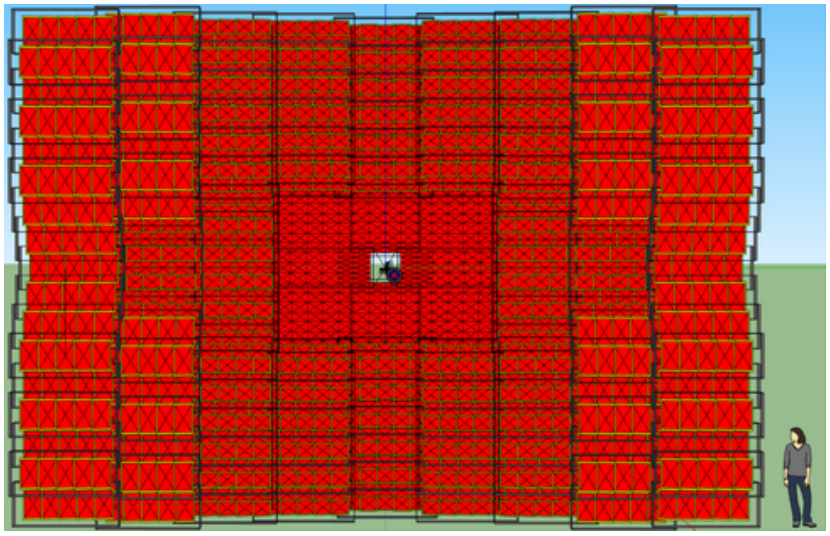


Figure 2.10: The ToF wall front view. Modules are marked by dark boxes, and the red crossed boxes denote the non-overlapping active areas of the single MRPC detectors inside. The yellow frames represent the overlap of the MRPCs [16].

The Time of Flight detector (ToF) is used for the identification of charged particles, mostly hadrons. The physics principle of identification was generally described in Section 1.1.3. The

ToF detector consists of Multigap Resistive Plate Chambers (MRPC). MRPC is a modern gas detector which replaced the metallic electrodes with resistive electrodes and contains in the gas mixture the avalanche quenching factor. This allows the detector to operate continuously and restricts discharges to a local area. MRPC is a relatively new technology for the TOF system. MRPC has very high detection efficiency ($> 95\%$) and very high time resolution < 100 ps. The TOF wall has an area of 12×9 m^2 and is placed at a distance of 7 m from the target [16].

Forward Spectator Detector

The Forward Spectator Detector (FSD) replaced the PSD detector in the CBM setup in 2022. Its main goal is to measure the projectile-like nucleus and light charged fragments emitted from it. Positioned at forward rapidity, the FSD detects a projectile-like nucleus. This detector helps with the reconstruction of the event plane and the centrality of the collision by measuring collective energy loss. Simulation studies have demonstrated that the FSD achieves an event plane resolution of approximately 70% for the x-component and about 40% for the y-component in Au+Au collisions at 11 AGeV [17][18].

2.3. Simulation Steps

A full chain of simulation of Au+Au collisions at 10 GeV/nucleon was performed by the members of the CBM collaboration. As the first step, the dynamics of collision were simulated with the help of the UrQMD transport model [19]. As the output from this model, an event-based file containing all the emitted particles was generated. In the next step, the Geant package [20] was used to transport the events with particles through the virtual representation of the whole volume of the CBM spectrometer. At the digitisation level, interactions of the simulated particles with the active layers of the detectors were generating the electronic responses, stored in the dedicated files. Events in these files were analysed to perform a track reconstruction. Next, the events with track were stored in the AnalysisTree format. Finally, protons and π^- mesons were combined into candidate pairs using the PFSimple [21] package prepared by Oleksii Lubynets.

2.4. Structure of trees of pairs

The trees of pairs of tracks obtained by PFSimple contain 36 branches, each of which represents one of the candidate parameters. For this paper, only 8 branches will be used:

1. `Candidates_chi2_geo`
2. `Candidates_chi2_prim_first`
3. `Candidates_chi2_prim_second`
4. `Candidates_chi2_topo`
5. `Candidates_l`
6. `Candidates_l_over_dl`
7. `Candidates_mass`

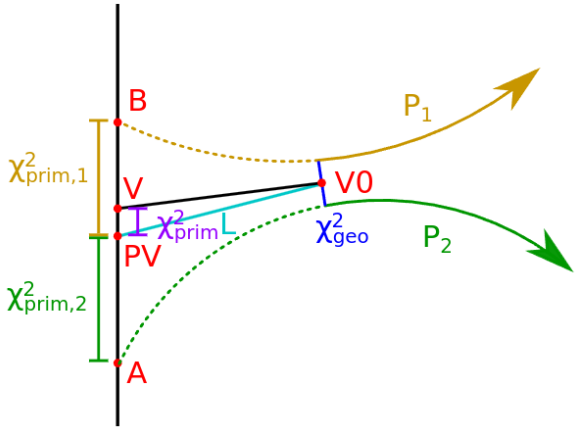


Figure 2.11: A scheme of topology of V0 particle decay and relevant variables, see text for details [22].

8. `Candidates_generation`

These variables were discussed in Section 1.1.4, and their choice is based on the physics argument of distinguishing V0 particles from pairs directly from the collision. However, the naming of relevant variables is currently different. Fig 2.11 shows the naming in correspondence with the used data structure.
The generation parameter is an additional one and is available only for the simulation. It marks the true origin of the particle as follows:

- 0 - candidate is not the particle, it is just a combination of two unrelated particles;
- 1 - candidate is the particle that came directly from a collision (first-generation particle);
- 2 - candidate is the particle that came from the decay of first generation particle;
- 3 - candidate is the particle that came from the decay of a second-generation particle;

For example, a value of 1 represents the Λ emitted directly from the collision, whereas a value of 2 can be a product of Σ^0 decay into Λ. To distinguish between signal and background for the training stage, we used the following criteria: `Candidates_generation` > 0 and `Candidates_generation` $= 0$, respectively. However, at the application level, no such cutting is made to mimic the true experimental situation, where signal and background can not be distinguished. In the future, we will train models using simulated data and apply them to real-world data, making this approach the most effective way to replicate true experimental conditions.

Chapter 3

ROOT/TMVA

3.1. Description of the Tools

ROOT [23] is an open-source, object-oriented data analysis framework developed at CERN. It is designed to analyse high-energy physics data and provides the relevant tools for this. The main feature of ROOT is the database-like TTree class and the ability to store ROOT objects in files in .root format. They are optimised to work with large datasets. That is why all the data used in this paper, both the training and test sets, as well as the analysis results, are stored in TTrees inside .root files.
ROOT can work with different machine learning libraries, but it also offers native support for supervised statistical learning techniques, both classification (multi-class and binary) and regression, the already built-in library: TMVA - Toolkit for Multivariate Data Analysis with ROOT [7].
TMVA offers a wide variety of machine learning techniques. However, for this paper, only three of them will be used: k-Nearest Neighbours, Multilayer Perceptron and Boosted Decision Tree. They were described in Section 1.2.

3.2. Configuration of chosen TMVA methods

A communication with TMVA proceeds through macros, written in C or C++ with the ROOT and TMVA libraries. Macros that will be used in this work are based on training and application examples for binary classification, provided by ROOT/TMVA tutorial [24]: TMVAClassification.C [25] and TMVAClassificationApplication.C [26], respectively. These macros were modified to train and use 3 selected methods: k-NN, BDT and MLP, whereas MLP was used with 3 different settings:

- k-NN method with 20 nearest neighbours considered for classification. 80% of the training data is used to determine variable scaling (normalisation). We used weights for neighbours similar to the equation in 1.21. Fitting those weights constituted the entire training process for this model. Further in this paper, we will reference this model as k-NN.

- Boosted Decision Tree method, which uses 850 shallow trees with a depth of each tree being no more than 3 levels. Each tree was trained on randomly sampled 50% of the data to avoid over-fitting. For splitting, the Gini Index was used, with the optimal number of splits set to 20. To prevent nodes from becoming too small, the stopping rule

requires each node to have at least 2.5% of the training data. Further in this paper, we will reference this model as BDT.

- The MLP method had one hidden layer. For the activation function, we used the tanh function. The number of training epochs (full passes through the dataset) was set to 700, with validation checks every 5 steps. We used three variants of this method, with hidden layers containing 2, 5, and 12 nodes, respectively. This was done to analyse the systematic error due to the number of nodes in the hidden layer. Further in this paper, we will reference those models as MLP2, MLP7, and MLP12.

Chapter 4

Data Analysis and Comparison of ML Methods

4.1. Data Preparation and Training of Models

To generate a dataset of candidates for $p\pi^-$ pairs, the simulation of Au+Au collision at the beam kinetic energy of 10 GeV/nucleon was performed. To achieve this, the UrQMD simulation model was utilised. A step-by-step description of the simulation procedure is provided in Section 2.3, and the structure of the final dataset is described in Section 2.4. This dataset consists of about $1.9 \cdot 10^9$ events: about 30600 signal and the rest - background of chance coincidences, and has the name `urqmd-5000-events.tree.with.signal.and.background.root`.
For the model training process, we do not need the entire dataset. To speed up the data processing, the macro `create_train.C` trimmed data to a subset consisting of all the signal and $1 \cdot 10^6$ background events.
As it was said in Section 3.1, the macro for training was based on `TMVAClassification.C`, an exemplary macro. Methods that were used, and their parameters, were described in Section 3.2. For each iteration of the training process, our macro randomly selected 14000 signal events and 50000 background events. In Table 4.1, you can see the CPU time for training and evaluation for each model. It appears that the k-NN model is fastest in the training stage. However, in the evaluation stage, the MLP method shows the best result, while k-NN is the slowest one.

Table 4.1: CPU time for training and evaluation with 64000 events for each method.

Method	CPU time for training with 64000 events [s]	CPU time for evaluation of 64000 events [s]
k-NN	0.886	100
BDT	41	4.92
MLP (12 nodes)	1210	0.552
MLP (7 nodes)	1030	0.533
MLP (2 nodes)	783	0.441

Following training, a `dataset` directory was created that contains all the fitted parameters per method in files in `.xml` format.

4.2. The ROC Curve

A key advantage of the `TMVAClassifiction.C` macro is that it gives tools to compare methods that were used. One such tool is the ROC curve.
The Receiver operating characteristic (ROC) curve is a widely used method for visually comparing the performance of different binary classification algorithms [27].
It illustrates the model's efficiency on a graphical plot, where the x-axis represents signal efficiency (chance of survival), ranging from 0 (0%) to 1 (100%). In turn, the y-axis represents the probability for background rejection, which also ranges from 0 (0%) to 1 (100%).

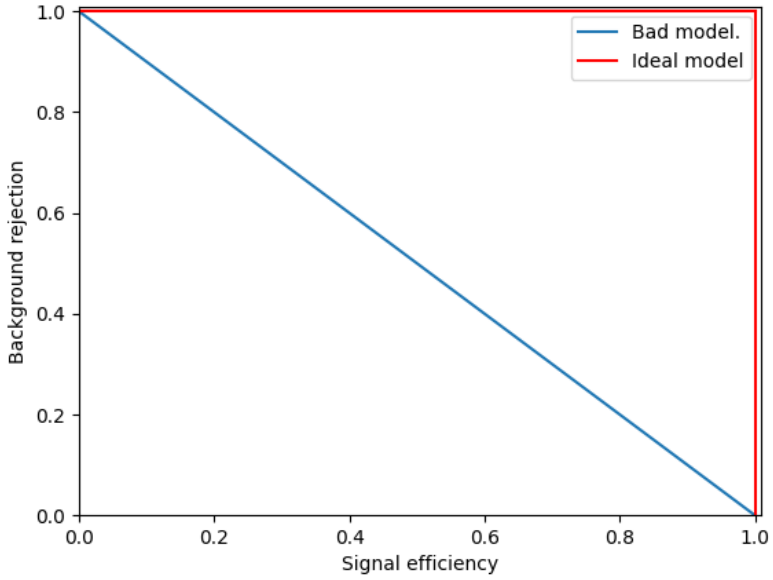


Figure 4.1: Example of the ROC curve. With the red curve, the ideal model is shown, while with the blue line, a bad (random) model is shown. See text for details.

An ideal model that rejects 100% of the background while saving 100% of the signal will create the ROC curve that forms a square. A bad model, which for 50% of rejected background, will identify only 50% of the signal (that is basically random selection), will form an anti-diagonal line on the plot. Both of those models can be seen in Fig. 4.1.
Consequently, the efficiency of different classification methods can be compared: the closer a method's ROC curve is to the upper-left corner (resembling a square), the better its performance. The primary disadvantage of the ROC curve method for analysing the performance of the models is the requirement of prior knowledge of the true class of each event, which limits us to analysing the training data only.
In Fig. 4.2, the ROC curve plot can be seen, generated by the TMVAClassification.C model. This plot indicates that the k-NN method has the best performance, whereas all 3 MLP methods are difficult to distinguish due to their similar performance. The BDT method falls between the k-NN method and the MLP method in this plot, so as the method's efficiency.

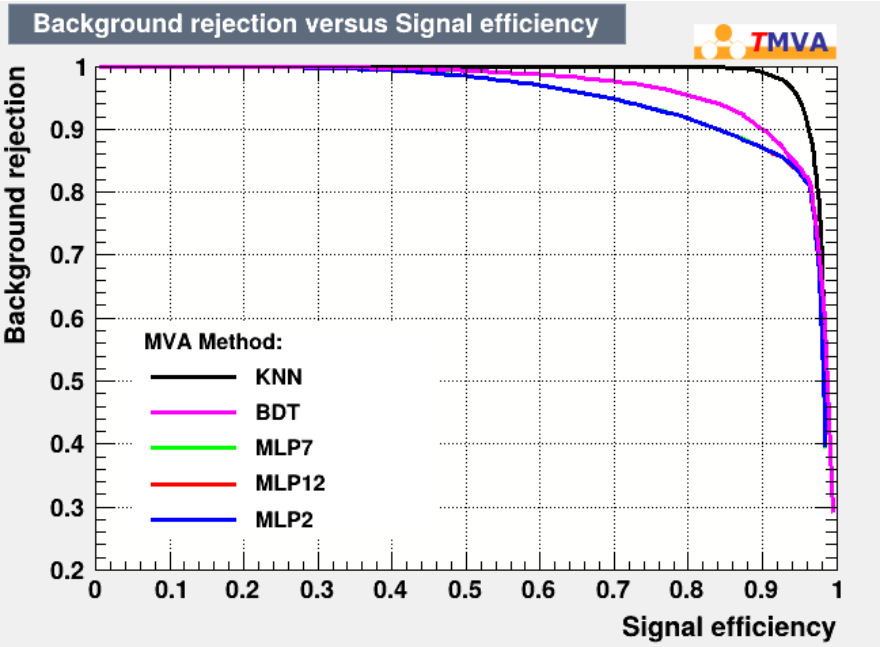


Figure 4.2: ROC curves for the training dataset, comparing the performance of each classification method implemented in `TMVAClassification.C`. MLP7 and MLP12 are under the MLP2 curve.

4.3. Application and optimisation of classifiers

To apply fitted parameters per method from the `dataset` directory to the whole test dataset, the modified `TMVAClassificationApplication.C` macro was used. It created `TMVApp.root` file, where for each event the corresponding value of the classifier was assigned (one per method): k-NN, BDT, MLP2, MLP7 and MLP12 values.
These values do not directly classify the event. Instead, they produce numerical scores within a predefined range. A key task is to identify a signal-rich subrange within this classifier distribution. Quantitatively, we aim to find the position of this threshold that would maximise the signal quality measure (the latter called the *objective function*). To do this, for each method, we will compare them for different threshold values and search for their maximum. The discussion on the justification for choosing those quality measures will be presented further in this chapter.

Within a given ML method, for a given minimal value of the classifier, a histogram of invariant mass for accepted $p\pi^-$ pairs is created. An example of such a histogram at the minimal value of the BDT classifier of 0.39 can be seen in Fig. 4.3. Based on the observed histogram, a hypothesis can be put forward that the signal events distribution can be modelled through a normal distribution and the background events distribution - with a linear function. The basis of this hypothesis is that the natural width Γ of Λ^0 hyperon is of the order of 10^{-12} MeV ($\Gamma \propto \frac{\hbar}{\tau}$), so it can be negligible. That's why the main source of mass dispersion is the measurement inaccuracy, and based on the Central Limit Theorem [28], the mass distribution will approach a normal distribution. The selection of linear distribution for the remaining background is based on the fact that the number of background counts per bin is small and varies slowly compared to the signal peak.

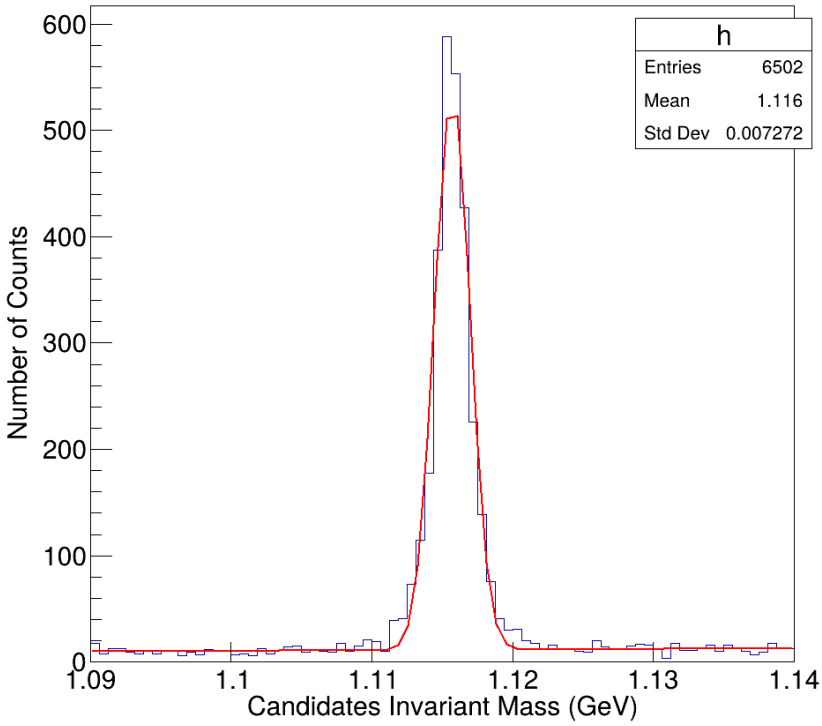


Figure 4.3: The selected event sample (BDT response > 0.39) is shown as a grey mass histogram, with a Gaussian fit (red curve) describing the signal peak.

Based on this hypothesis, the fit function looks like this:

$$f(m) = \left[A + B \cdot m + \frac{N_{sig}}{\sqrt{2\pi}\sigma} \exp\left(-0.5 \left(\frac{(m - m_0)}{\sigma} \right)^2 \right) \right] \cdot dm \tag{4.1}$$

After the fit is performed, following parameters, that will be used further, and their errors ($\Delta_{\text{parameter}}$) are obtained: the average mass (m_0), the standard deviation of the distribution (σ), the number of signal counts (S). In Fig. 4.3, you can see the fit result as a red curve.
To find the number of background counts (B), the histogram is integrated within the area $m_0 \pm 3\sigma$. Because of this choice of area, approximately 99.7% of the signal is accepted [29]. This procedure gives us the total number of counts (T), whereas the number of background counts is: B = T - S.
An example of the implementation of this process in the C++ code using ROOT/TMVA libraries for the BDT method can be seen in Appendix A.
To determine the optimal thresholds for each method, two signal quality metrics (*objective functions*) were employed: S/B ratio and S/Δ_S (the signal significance), where Δ_S is the error of S, which was obtained during fitting. The choice of these metrics was motivated by the goal of our analysis - to maximise the number of counts of reconstructed Λ^0 particles. That is why, for this task, signal significance will be the best measure of the performance of our models. In addition, we will also investigate S/B behaviour. For each method, two plots were created, showing S/Δ_S and S/B ratio as a function of the threshold (minimal) value of a relevant classifier. A code creating these plots can be seen in Appendix B, while the results

can be seen in Fig. 4.4 for the k-NN method, Fig. 4.5 for the BDT method and Fig. 4.6, Fig. 4.7, Fig. 4.8 for the MLP method with 2, 7 and 12 nodes in the hidden layer, respectively. The threshold values of classifiers which maximise the S/Δ_S per method can be seen in the Tab. 4.2, whereas the ones which maximise the S/B ratio can be seen in the Tab. 4.3.

Table 4.2: The threshold values of classifiers which maximise the S/Δ_S per method, along with the corresponding S/B ratio and S/Δ_S ratio at this value.

Method	Boundary value of classifiers	S/B ratio	S/Δ_S
MLP2	0.995	1.21	61.8
MLP7	0.996	1.6	61.4
MLP12	0.999	1.4	61.5
k-NN	0.99	1.96	123.9
BDT	0.22	1.41	78.42

Based on Tab. 4.2, the k-NN method achieves significantly higher signal significance compared to all the other method. Figure 4.4 shows that the k-NN method's signal significance growth is discrete due to the model's vote-based design. This model has 20 neighbours, so the performance improvements occur only within specific value ranges, such as [0.95, 1), [0.9, 0.95), [0.85, 0.9) and so on, as described by Eq. 1.20 (if we didn't consider the weights of neighbours, which was applied in the real model). This design prevents potential overfitting at high thresholds, which explains the absence of a signal significance drop. For BDT and MLP, the signal significance drops at a certain point, as can be seen in Fig. 4.5, Fig. 4.6, Fig. 4.7, and Fig. 4.8. One of the possible causes of this decline can be overfitting.

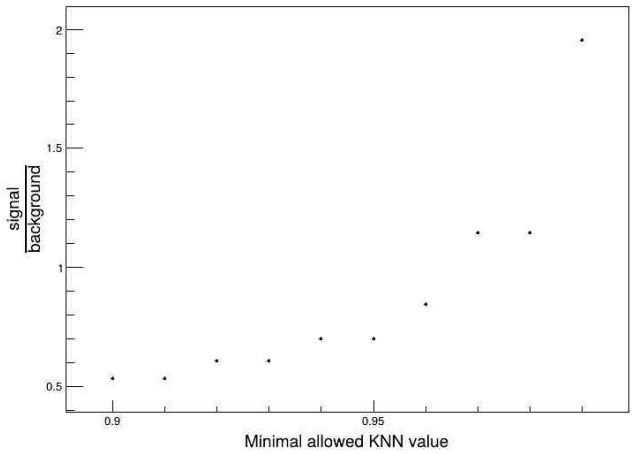
Table 4.3: The threshold values of classifiers which maximise the S/B ratio per method, along with the corresponding S/B ratio and S/Δ_S ratio at this value.

Method	Boundary value of classifiers	S/B ratio	S/Δ_S
MLP2	0.9968	2.2	56
MLP7	0.9981	6.2	16.8
MLP12	0.9998	2.3	41.9
k-NN	0.99	1.96	124
BDT	0.37	12.8	61.6

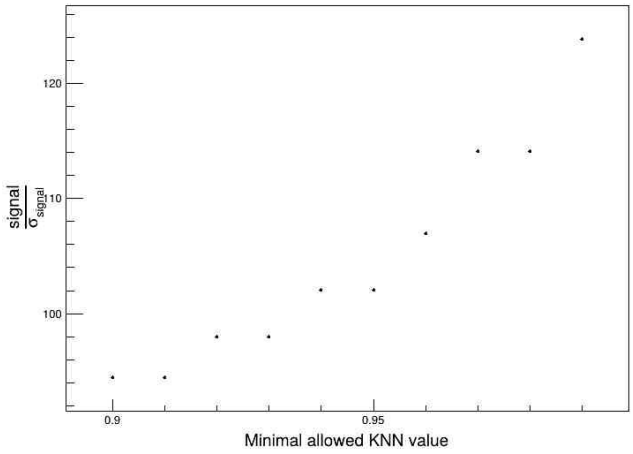
Figures 4.6 and 4.8 demonstrate a steady rise in the S/B ratio for both MLP2 and MLP12. However, for MLP7, as can be seen in Fig. 4.7, the S/B ratio reaches a maximum at the threshold point 0.9981 before declining. In order to understand the reason, the values of S and B were inspected. As can be seen in Table C.1 in Appendix C, in the range of higher threshold values of the classifier, the signal counts drop faster than the background. A similar behaviour of the S/B ratio for the BDT method can be seen in Fig. 4.5. A corresponding table C.2 in Appendix C reveals the same reason for this method.
This drop in the S/B ratio can also occur for other MLPs. However, the drop in signal significance occurs at considerably smaller values of threshold classifiers, meaning that by the time the S/B ratio starts declining, the significance is already too low to justify testing those threshold values. In terms of S/B ratio, the BDT method emerges as the top performer, while the k-NN method and the MLP methods exhibit comparable performance to each other.
In addition, the absolute signal yields for each method can be examined - Tab. 4.4. As one can see, also in this respect, the k-NN model demonstrates the best performance again, with MLP2 in second place.

Table 4.4: Number of accepted signal events at the optimal decision boundary threshold (with respect to S/Δ_S) for each method.

Method	Boundary value of classifiers	Number of Signal Counts
MLP2	0.995	7577
MLP7	0.996	6526
MLP12	0.999	6917
BDT	0.37	4044
k-NN	0.99	22510

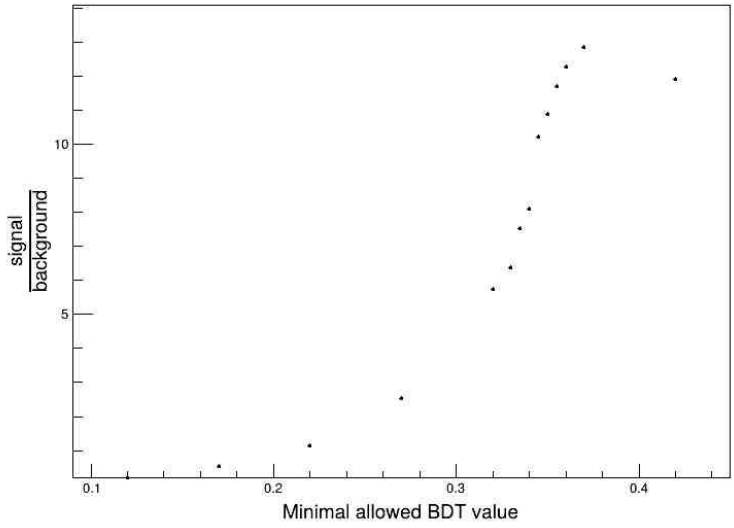


(a) S/B ratio for k-NN across threshold interval [0.9, 0.99].

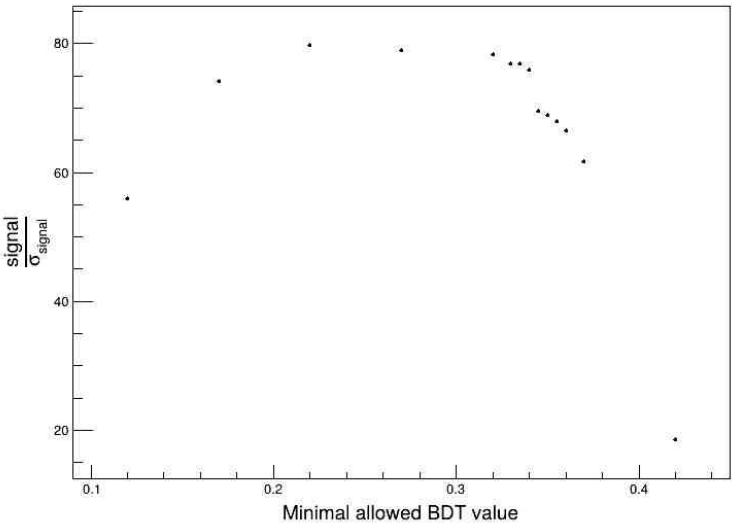


(b) Signal significance for k-NN across threshold interval [0.9, 0.99].

Figure 4.4: Quality measures for k-Nearest Neighbours (k-NN) classifier. Left panels show signal-to-background ratio (S/B), right panels show signal significance (S/Δ_S). Threshold intervals were optimised for this method.

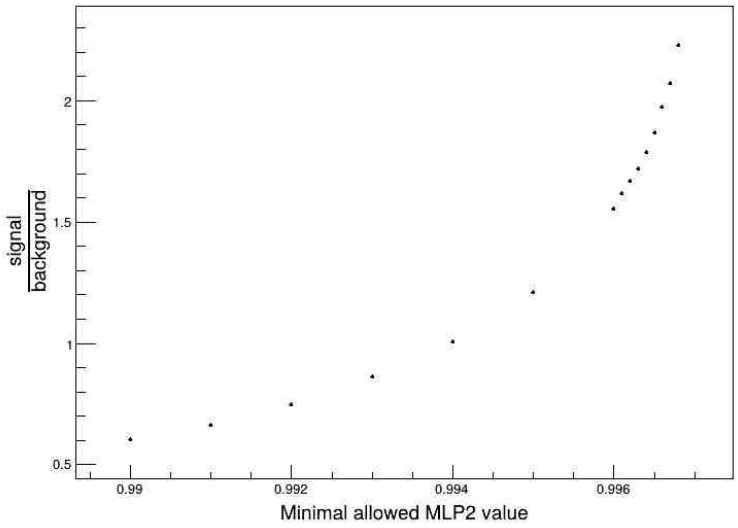


(a) S/B ratio for BDT across threshold interval [0.12, 0.42].

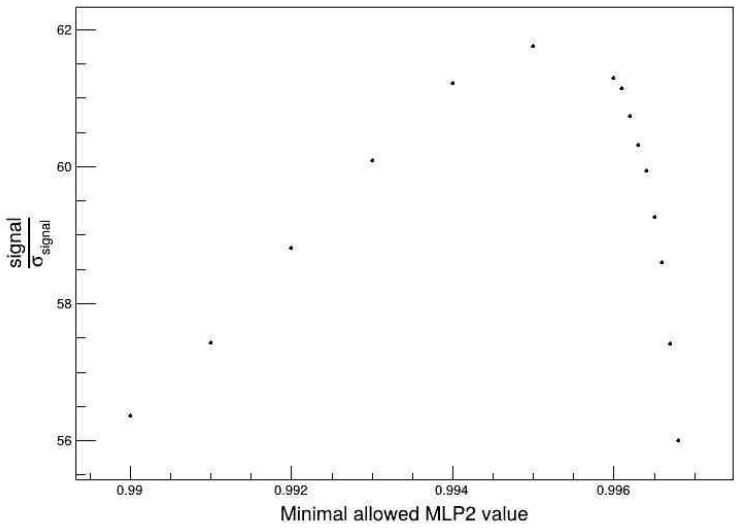


(b) Signal significance for BDT across threshold interval [0.12, 0.42].

Figure 4.5: Quality measures for Boosted Decision Trees (BDT) classifier. Left panels show signal-to-background ratio (S/B), right panels show signal significance (S/Δ_S). Threshold intervals were optimised for this method.

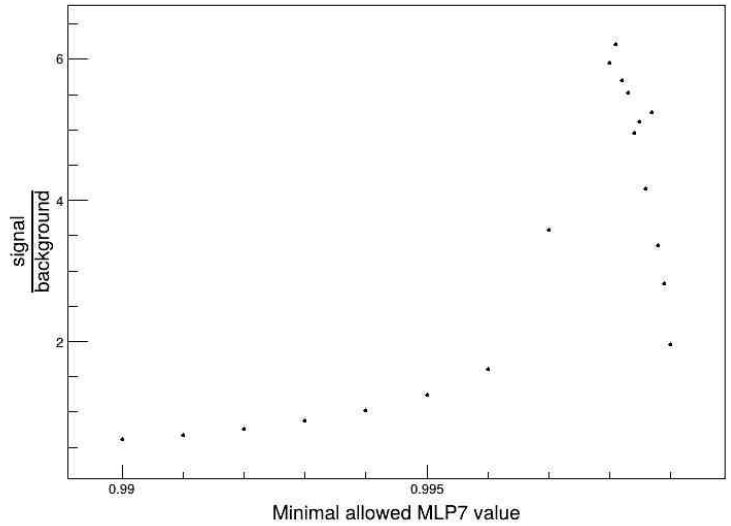


(a) S/B ratio for MLP (2 nodes) across threshold interval [0.99, 0.997].

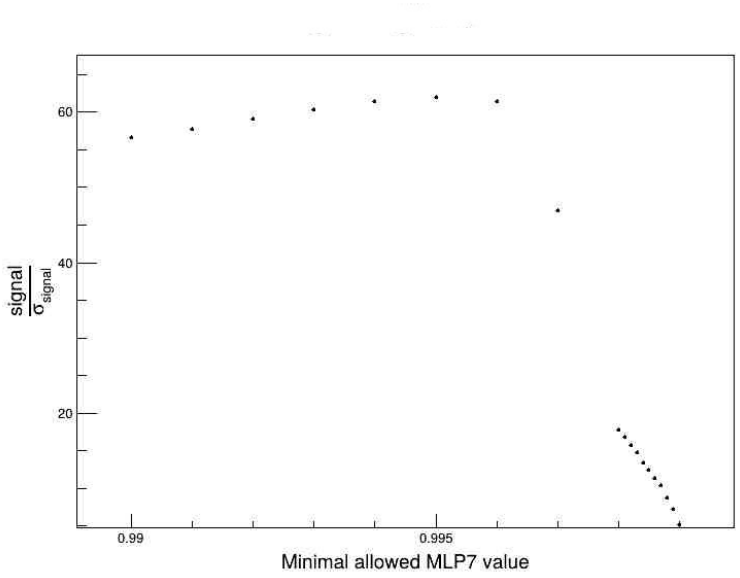


(b) Signal significance for MLP (2 nodes) across threshold interval [0.99, 0.997].

Figure 4.6: Quality measures for the MLP classifier with 2 hidden-layer nodes. Left panels show signal-to-background ratio (S/B), right panels show signal significance (S/Δ_S). Threshold intervals were optimised for this method.

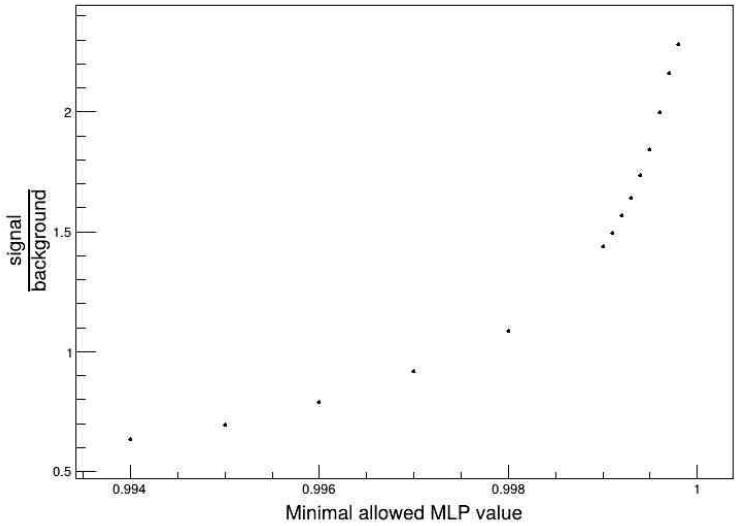


(a) S/B ratio for MLP (7 nodes) across threshold interval [0.99, 0.9989].

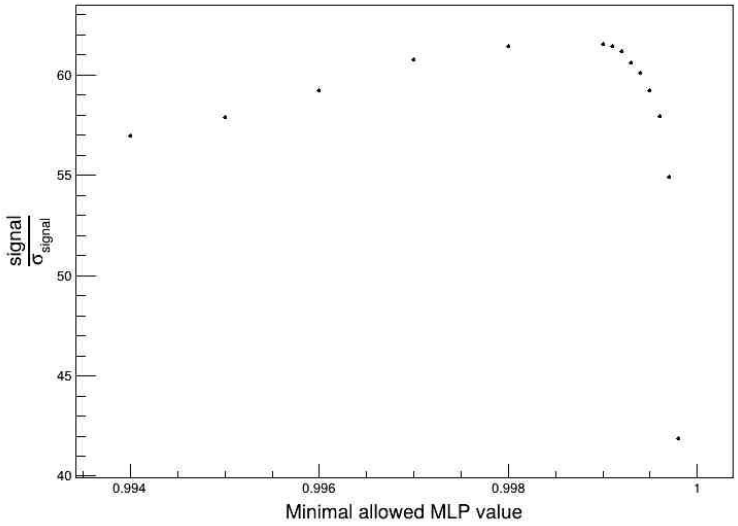


(b) Signal significance for MLP (7 nodes) across threshold interval [0.99, 0.9989].

Figure 4.7: Quality measures for the MLP classifier with 7 hidden-layer nodes. Left panels show signal-to-background ratio (S/B), right panels show signal significance (S/Δ_S). Threshold intervals were optimised for this method.



(a) S/B ratio for MLP (12 nodes) across threshold interval [0.994, 0.9998].



(b) Signal significance for MLP (12 nodes) across threshold interval [0.994, 0.9998].

Figure 4.8: Quality measures for the MLP classifier with 12 hidden-layer nodes. Left panels show signal-to-background ratio (S/B), right panels show signal significance (S/Δ_S). Threshold intervals were optimised for this method.

4.4. Λ mass reconstruction cross check

The performance of those methods can be cross-checked in terms of correctness, Λ hyperon mass reconstruction. For reference, Λ^0 hyperon mass is given by the PDG: 1115.683 ± 0.006 MeV [3]. Fig. 4.9 shows the comparison of the extracted mass centroids for examined ML methods at optimal threshold classifier values, to Eq. 4.1 model fitting. All of them are found to be in agreement with the PDG value within a 2σ range.

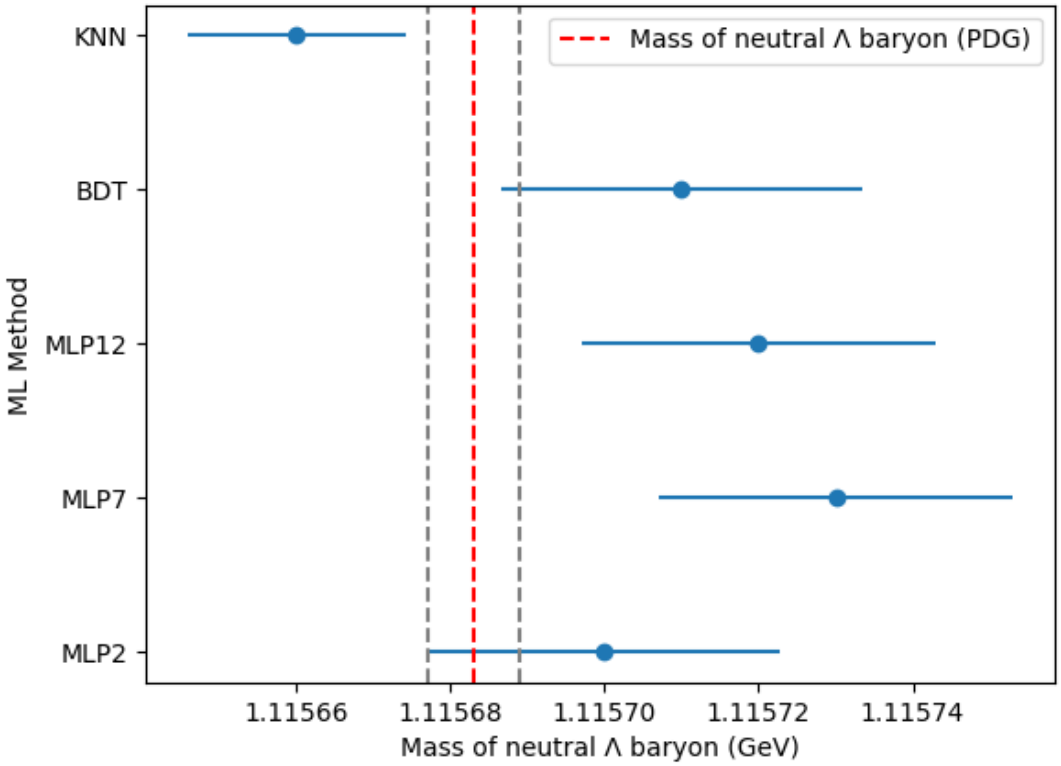


Figure 4.9: Comparison of reconstructed Λ^0 hyperon masses across classification methods at optimal threshold values of classifiers. The dashed vertical line indicates the PDG reference value, with its uncertainty represented by the grey band. Points show the fitted mass values (with statistical uncertainties) for different methods.

Chapter 5

Conclusion and Outlook

In this bachelor's thesis, the identification of Λ^0 hyperons from simulated Au+Au collisions at 10 GeV/nucleon using the CBM experimental setup and the TMVA machine learning package was examined. The work begins with a theoretical introduction on modern particle physics, particle detection and reconstruction, and supervised learning techniques. An overview of the CBM station and the simulation of particle collisions is provided. The application of machine learning methods for classification and a comparative analysis of their performance in distinguishing signal from background events was the main focus of this thesis.
The results of the comparative analysis are as follows:

- Considering the S/Δ_S quality measure, and neglecting the evaluation time, the k-NN method shows the best performance. This method detects the largest number of signal counts while being the most significant. The drawback is the necessity of having a big training dataset and strong computational resources, due to the dimensionality curse, we have the longest evaluation time among all the methods. The only room for model improvement by is by changing the number of neighbours, however in comparison to other models presented in this paper, it is a restrictive hyperparameter optimisation strategy

- The Boosted Decision Tree method showed itself as the fastest method, while in terms of performance, qualified by the signal significance, it was the second one. The BDT method also showed good results in terms of S/B ratio, but in number of signal counts it had the worst performance.

- The MLP methods, while being tested with different configurations of hidden layer nodes, showed similar results across all of them, which means, that there is a necessity of more thorough testing of hyperparameters (different number of nodes or hidden layers) in the MLP methods, to see progression or regression in terms of performance. Right now, this model shows the worst results in terms of the signal significance among all the models. The key advantage is the ability to fine-tune, and theoretically, it can provide the best performance across all models [7]. But fine-tuning of the MLP model is quite CPU-consuming and exceeds the volume of this Bachelor's thesis.

Based on these findings, we conclude that for the reconstruction of Λ from Au+Au at 10A GeV, the k-NN model was found to be the best choice, prioritising statistical significance without time constraints. For faster processing, the BDT method is the preferred alternative. While the MLP method holds the greatest theoretical potential, further examination is required to fully realise its capabilities.

Bibliography

[1] R. Dvořák and L. Chlad. "Performance studies for the mCBM experiment campaigns in 2022". In: *PoS* FAIRness2022 (2023), p. 013. DOI: 10.22323/1.419.0013

. [2] M. Thomson. *Modern particle physics*. New York: Cambridge University Press, 2013. DOI: 10.1017/CBO9781139525367

. [3] S. Navas et al. "Review of particle physics". In: *Phys. Rev. D* 110 (2024). DOI: 10.1103/PhysRevD.110.030001

. [4] D. J. Griffiths. *Introduction to Electrodynamics*. 4th ed. Cambridge University Press, 2017. DOI: 10.1017/9781009397735

. [5] *Lambda topology scheme on STAR collaboration web-page*. URL: https://drupal.star.bnl.gov/STAR/system/files/userfiles/3639/tcut.png

. [6] T. Hastie, R. Tibshirani, and J. Friedman. *The Elements of Statistical Learning*. Springer Series in Statistics. New York, USA: Springer New York Inc., 2001. DOI: 10.1007/978-0-387-84858-7

. [7] A. Hoecker *et al. TMVA - Toolkit for Multivariate Data Analysis*. 2009. DOI: 10.48550/arXiv.physics/0703039. URL: https://root.cern.ch/download/doc/tmva/TMVAUsersGuide.pdf

. [8] R E Bellman. *Adaptive Control Processes: A Guided Tour*. First introduction of curse of dimensionality. Princeton University Press, 1961. ISBN: 9780691652214

. [9] T. Ablyazimov *et al.* (CBM Collaboration). "Challenges in QCD matter physics. The scientific programme of the Compressed Baryonic Matter experiment at FAIR". In: *Eur. Phys. Jour. A* 53, 60 (2017). DOI: 10.1140/epja/i2017-12248-y

. [10] *The Micro-Vertex Detector of the CBM Experiment at FAIR*. Tech. rep. Darmstadt, Germany: CBM Collaboration, GSI Helmholtz Center for Heavy Ion Research, 2021. URL: https://repository.gsi.de/record/246516

. [11] *Technical Design Report for the CBM Silicon Tracking System (STS)*. Tech. rep. Darmstadt, Germany: CBM Collaboration and The STS Workgroup, GSI Helmholtz Center for Heavy Ion Research, 2013. URL: https://repository.gsi.de/record/54798

. [12] M. Zyzak. "Online selection of short-lived particles on many-core computer architectures in the CBM experiment at FAIR". URN: urn:nbn:de:hebis:30:3-414288. PhD thesis. Faculty of Computer Science and Mathematics of the Johann Wolfgang Goethe University in Frankfurt am Main, 2015

. [13] *Technical Design Report for the CBM Ring Imaging Cherenkov Detector (RICH)*. Tech. rep. Darmstadt, Germany: CBM Collaboration and The CBM-TRD Working Group, GSI Helmholtz Center for Heavy Ion Research, 2013. URL: https://repository.gsi.de/record/65526

. [14] *Technical Design Report for the CBM Muon Chambers (MuCh)*. Tech. rep. Darmstadt, Germany: CBM Collaboration and The MUCH TDR Workgroup, GSI Helmholtz Center for Heavy Ion Research, 2015. URL: https://repository.gsi.de/record/161297

. [15] *Technical Design Report for the CBM Transition Radiation Detector (TRD)*. Tech. rep. Darmstadt, Germany: CBM Collaboration and The CBM-TRD Working Group, GSI Helmholtz Center for Heavy Ion Research, 2018. URL: https://repository.gsi.de/record/217478

. [16] *Technical Design Report for the CBM Time-of-Flight System (ToF)*. Tech. rep. Darmstadt, Germany: CBM Collaboration and The TOF Workgroup, GSI Helmholtz Center for Heavy Ion Research, 2014. URL: https://repository.gsi.de/record/109024

. [17] M. Teklishyn. "Detectors and Electronics for the CBM experiment at FAIR". In: (2025). DOI: 10.48550/arXiv.2506.20545

. [18] R. Dvořák. "Forward Spectator Detector for CBM". EPJ Featured Poster, Quark Matter 2025, Goethe University Frankfurt, Germany, poster session 1, Detectors & Future Experiments (Apr. 2025). URL: https://indico.cern.ch/event/1334113/contributions/6289932/

. [19] S.A. Bass *et al.* "Microscopic models for ultrarelativistic heavy ion collisions". In: *Prog. Part. Nucl. Phys* 41 (1998), p. 255. DOI: 10.1016/s0146-6410(98)00058-1

. [20] R. Brun et al. *GEANT: Detector Description and Simulation Tool; 1994*. CERN Program Library. Geneva: CERN, 1993. URL: https://cds.cern.ch/record/1082634

. [21] O. Lubynets. *PFSimple*. URL: https://github.com/HeavyIonAnalysis/PFSimple

. [22] R. Korsak. *Hypertriton reconstruction using machine learning technique at the CBM experiment*. Tech. rep. GSI Helmholtz Center for Heavy Ion Research, 2023

. [23] ROOT Team/CERN. *ROOT*. URL: https://root.cern

. [24] ROOT Team/CERN. *TMVA*. URL: https://root.cern/manual/tmva/

. [25] ROOT Team/CERN/TMVA. *TMVA Classification Example*. URL: https://root.cern/doc/master/TMVAClassification_8C.html

. [26] ROOT Team/CERN/TMVA. *TMVA Classification Applcation Example*. URL: https://root.cern/doc/master/TMVAClassificationApplication_8C.html

. [27] M. Junge and J. Dettori. "ROC Solid: Receiver Operator Characteristic (ROC) Curves as a Foundation for Better Diagnostic Tests." In: *Global Spine Journal, 8, 424* (2018). DOI: 10.1177/2192568218778294

. [28] I. Bárány and V. Vu. "Central limit theorems for Gaussian polytopes". In: *The Annals of Probability* 35.4 (2007), pp. 1593–1621. DOI: 10.1214/009117906000000791

. [29] F. Huber. *A Logical Introduction to Probability and Induction*. Oxford University Press, 2018. ISBN: 9780190845407

Appendix A

Classifying events with the BDT method.

```
#include <cmath>
#include <vector>
#define PI 3.1425
#define plot_start 1
#define plot_end 1.25
#define fit_start 1.10
#define fit_end 1.13

//Function
  Double_t m_fun (Double_t *xarg, Double_t *par) {
    Double_t m = xarg[0];
    Double_t A = par[0], B = par[1],
        Nlam = par[2], m0 = par[3], sig = par[4], dm = par[5];

    Double_t exp_arg = (m - m0)/sig;

    return ( A + B * m + Nsig / (TMath::Sqrt(2*PI*sig*sig)) *
        TMath::Exp( - 0.5 * exp_arg * exp_arg ) ) * dm  ;
  }

vector<float> classify_BDT (Float_t b_value)
{

  //CREATING HISTOGRAM AND CANVAS
  TCanvas* c1 = new TCanvas("c1", "Canvas with Legend", 1000, 1000);
  c1->SetGrid (0, 0);
  c1->SetTopMargin (0.05);
  c1->SetBottomMargin (0.15);
  c1->SetRightMargin (0.04);
  c1->SetLeftMargin (0.16);
  TH1F* h = new TH1F("h", "", 400, plot_start, plot_end);

  //READING
```

```
TFile *f = TFile::Open ("TMVApp.root");
Float_t mass, BDT;
TTree *OutputTree = (TTree*) f->Get ("OutputTree");
OutputTree->SetBranchAddress ("Candidates_mass", &mass);
OutputTree->SetBranchAddress ("BDT", &BDT);

//FILLING HISTOGRAM
for (int evt=0; evt < OutputTree->GetEntries(); evt++)
{
  OutputTree->GetEvent (evt);
  if(BDT >= b_value && mass > plot_start && mass < plot_end){
      h->Fill(mass);
  }
}

//PLOTTING
h->SetFillColor(18);
h->GetXaxis()->SetTitle ("Candidates Invariant Mass (GeV)");
h->GetXaxis()->CenterTitle (true);
h->GetXaxis()->SetTitleOffset (1.25);
h->GetXaxis()->SetTitleSize(0.055);
h->GetXaxis()->SetRangeUser(1.09, 1.14);
h->GetYaxis()->SetTitle ("Number of Counts");
h->GetYaxis()->CenterTitle (true);
h->GetYaxis()->SetTitleOffset (1.25);
h->GetYaxis()->SetTitleSize(0.055);
h->Draw();

//FIT PREPARATION
TF1 fun("function", m_fun, fit_start, fit_end, 6);
fun.SetParameters(1.0, 0.0, 200, 1.115, 0.001, h->GetBinWidth(2));
fun.FixParameter ( 5 , h->GetBinWidth(2));
//fun.SetParLimits(2, 1e3, 1e6);
fun.SetParLimits(3, 1.1, 1.13);
fun.SetParLimits(4, 0, 0.003);

//FITTING
TFitResultPtr fitRes = h->Fit ( &fun , "RSIEM" );

//GETTING RESULTS
float signal = round(fun.GetParameter(2));
float signal_error = fun.GetParError(2) ;
float min_mass = fun.GetParameter(3) - 3 * fun.GetParameter(4);
float max_mass = fun.GetParameter(3) + 3 * fun.GetParameter(4);

//GETTING BACKGROUND FOR 3sigma SEGMENT
float tot = 0;
int bin_min = h->FindBin(min_mass);
int bin_max = h->FindBin(max_mass);
```

```
  tot = h->Integral(bin_min, bin_max);
  float background = tot - signal;


  //SAVING PLOT
  std::stringstream filenameStream;
  filenameStream <<
    "./plots/BDT/plot_iteration_" << b_value << "_value_" << ".jpg";
  std::string filename = filenameStream.str();
  c1->SaveAs(filename.c_str());
  std::cout << "Canvas saved as: " << filename << std::endl;

  //Creating output vector
  vector<float> values;
  values.push_back(signal);
  values.push_back(signal_error);
  values.push_back(background);

  //CLOSING EVERYTHING
  delete c1;
  delete h;
  f->Close();

  return values;
}
```

Appendix B

Comparing quality measures of the BTD method for different thresholds.

```
#include <vector>
#include <array>
#include "compare_BDT.h"

#define min_b 12
#define max_b 42
#define min_b_2 330
#define max_b_2 360
#define div 100
#define div_2 1000

int BDT_treshold_comparing()
 vector<float> b_vec;
 vector<float> sign_error_vec;
 vector<float> sign_back_vec;
 vector<float> classification_quality;
 //BDT FOR SET b_values
 for(Float_t i = min_b; i <= max_b; i = i + 5){
        classification_quality = classify_BDT(i/div);
        b_value = i/div;
        signal = a[0];
        error = a[1];
        background = a[2];
        sign_error = signal/error;
        sign_back = signal/background;
        b_vec.push_back(b_value);
        sig_error_vec.push_back(sign_error);
        sign_back_vec.push_back(sign_back);
        classification_quality.clear();
 }
 for(Float_t i = min_b_2; i <= max_b_2; i = i + 5){
        classification_quality = classify_BDT(i/div);
        b_value = i/div;
```

```
        signal = a[0];
        error = a[1];
        background = a[2];
        sign_error = signal/error;
        sign_back = signal/background;
        b_vec.push_back(b_value);
        sig_error_vec.push_back(sign_error);
        sign_back_vec.push_back(sign_back);
        classification_quality.clear();
}
//CREATING GRAPH
TGraph *graph1 = new TGraph(b_vec.size(), &b_vec[0], &sign_back_vec[0]);
TGraph *graph2 = new TGraph(b_vec.size(), &b_vec[0], &sig_error_vec[0]);

//CREATING CANVAS FOR SIGNAL/BACKGROUND
TCanvas *c2 = new TCanvas("c2", "signal/background", 800, 600);
gStyle->SetOptStat (0);
gStyle->SetLegendBorderSize (0);
gStyle->SetLegendTextSize (0.025);
gStyle->SetLabelSize (0.025, "XY");
gStyle->SetNdivisions ( 505 , "XY");
gStyle->SetTextFont (42);
graph1->SetTitle("signal/background");
//X
graph1->GetXaxis()->SetTitle ("Minimal allowed BDT value");
graph1->GetXaxis()->CenterTitle (true);
graph1->GetXaxis()->SetTitleOffset (1);
graph1->GetXaxis()->SetTitleSize(0.035);
//Y
graph1->GetYaxis()->SetTitle ("#frac{signal}{background}");
graph1->GetYaxis()->CenterTitle (true);
graph1->GetYaxis()->SetTitleOffset (0);
graph1->GetYaxis()->SetTitleSize(0.035);

//DRAWING AND SAVING
graph1->SetMarkerStyle(20);
graph1->SetMarkerSize(0.5);
graph1->Draw("AP");
std::stringstream filenameStream;
filenameStream << "./plots/BDT_compare/signal_background_"<< min_b << "_" << max
std::string filename = filenameStream.str();
c2->SaveAs(filename.c_str());

//CREATING CANVAS FOR SIGNAL/ERROR
TCanvas *c3 = new TCanvas("c3", "signal/signal_error", 800, 600);
graph2->SetTitle("signal/signal error");
//X
graph2->GetXaxis()->SetTitle ("Minimal allowed BDT value");
graph2->GetXaxis()->CenterTitle (true);
```

```
 graph2->GetXaxis()->SetTitleOffset (1);
 graph2->GetXaxis()->SetTitleSize(0.035);
 //Y
 graph2->GetYaxis()->SetTitle ("#frac{signal}{#sigma_{signal}}");
 graph2->GetYaxis()->CenterTitle (true);
 graph2->GetYaxis()->SetTitleOffset (1);
 graph2->GetYaxis()->SetTitleSize(0.035);

 //DRAWING AND SAVING
 graph2->SetMarkerStyle(20);
 graph2->SetMarkerSize(0.5);
 graph2->Draw("AP");
 std::stringstream filenameStream1;
 filenameStream1 << "./plots/BDT_compare/signal_signal_error_"<< min_b << "_" <
 std::string filename1 = filenameStream1.str();
 c3->SaveAs(filename1.c_str());

 outputFile->Write();
 outputFile->Close();

return 0;
}
```

`sign_error_vec` and `sign_back_vec` are vectors, that stored $N_{sig}/err_{N_{sig}}$ and N_{sig}/N_{back} values respectfully, for different boundary values, that are stored in `b_vec`.

Appendix C

Tables

Table C.1: S/B ratio and number of counts for signal and background as a function of threshold for the MLP7 method.

Threshold	S	B	S/B
0.99	9276	15205	0.61
0.991	8996	13380	0.67
0.992	8808	11562	0.76
0.993	8570	9753	0.88
0.994	8151	7955	1.02
0.995	7556	6093	1.24
0.996	6526	4069	1.60
0.997	2868	802	3.58
0.998	369	62	5.95
0.9981	323	52	6.21
0.9982	285	50	5.7
0.9983	254	46	5.52
0.9984	208	42	4.95
0.9985	184	36	5.11
0.9986	154	37	4.16
0.9987	131	25	5.24
0.9988	101	30	3.37
0.9989	76	27	2.81
0.999	47	24	1.96

Table C.2: S/B ratio and number of counts for signal and background as a function of threshold for the BDT method.

Threshold	S	B	S/B
0.12	26253	127107	0.21
0.17	17144	31440	0.55
0.22	11282	10004	1.13
0.27	8469	3338	2.54
0.32	6998	1219	5.74
0.33	6623	1039	6.37
0.335	6497	865	7.51
0.34	6282	775	8.11
0.345	5245	513	10.2
0.35	5125	471	10.9
0.355	4946	423	11.7
0.36	4715	384	12.3
0.37	4044	315	12.8
0.42	357	30	11.9