

ÉCOLE POLYTECHNIQUE FÉDÉRALE DE
LAUSANNE

MASTER THESIS

Towards an intelligent automatic event
interpretation at LHCb

Author:

Téo GIGANDET

Supervisor:

Elena GRAVERINI

Responsible teacher:

Lesya SHCHUTSKA

June 21, 2024

EPFL

Abstract

The main aim of the LHCb experiment at the Large Hadron Collider at CERN is to gather high-precision measurements of heavy hadron decays. The large amount of data produced can not entirely be stored, necessitating the use of a trigger system. This system performs fast analysis to reject parts of the data that are not the focus of LHCb research, storing only the remaining data for further offline analysis. Future upgrades at LHC will drastically increase the amount of data produced, meaning that triggers will need to select fewer data. In this context, a novel approach is proposed: The Deep Full Event Interpretation (DFEI), which is based on graph neural networks (GNN). The graphs represent the particles in a particular event and their relationships with each other. The DFEI algorithm suppresses the nodes and edges that don't relate to heavy hadron decays, leaving only those decays remaining. This thesis proposes an enhancement of this algorithm. Currently, it has two main shortcomings: charmed hadrons are disregarded and only the charged part of the decays is selected. The primary goal of this work is to extend the capabilities of this algorithm to include neutral particles. The use of Boosted Decision Trees, using the reconstructed decays that the DFEI algorithm has selected, is explored. The encouraging results which are discussed constitute the ground work for future enhancements and for the implementation of this extension.

Contents

1	Introduction	1
2	Context	2
2.1	The Standard Model	2
2.2	The LHC at CERN	3
2.3	The LHCb Experiment	4
2.4	Real-time Data Analysis at LHCb	8
2.5	Graph Neural Network for Deep Full Event Interpretation . .	10
2.6	Bremsstrahlung Recovery at LHCb	13
2.7	LHCb Simulation	14
3	The Classifier Algorithm	15
3.1	Decision Tree Classifier	15
3.2	Boosting	17
3.3	Hyperparameters Optimization	17
3.4	Feature Correlation	18
3.5	Recursive Feature Elimination	19
3.6	Performance Metrics	20
4	Data Processing	23
4.1	Description of the Datasets	23
4.2	Data	27
4.3	Features Analysis	31
5	Results	36
5.1	Application of the Boosted Decision tree	36
5.1.1	BDT with 15 Features	36
5.1.2	BDT with 9 Features	39

5.1.3	Recursive Feature Elimination	41
5.1.4	Hyperparameters Optimization	42
5.2	Detection of Events that contain Neutral Particles coming from a B Hadron Decay	44
5.3	Application to Photons	45
5.4	Translations to Graph Edges	46
6	Conclusions and Outlook	50
A	Additional Figures	57
B	Links to the Datasets	61

Chapter 1

Introduction

The LHCb experiment at CERN's Large Hadron Collider is designed to explore fundamental aspects of particle physics by studying the decays of beauty and charm hadrons. These studies are crucial for investigating phenomena that are not well described by the Standard Model, such as CP asymmetry for instance. The LHCb detector, equipped with a variety of sophisticated subdetectors, collects vast amounts of data from particle collisions, necessitating advanced data analysis techniques to extract meaningful insights.

One of the key challenges in the LHCb experiment is the real-time processing and interpretation of complex event data. New upgrades, such as the migration of the trigger to be fully implemented on the software level, force to adapt the way data reduction is done. The Deep Full Event Interpretation (DFEI) algorithm, based on Graph Neural Networks (GNN), aims at reconstructing the decay trees. Its details are explained in 2.5. It is currently in development with the idea of implementing it in the trigger line for the future high luminosity LHC. However, it does not include neutral particles yet. Neutral particles are produced in many decay processes, and their exclusion from the DFEI algorithm widely limits the depth of analysis.

This thesis aims to address this gap by extending the DFEI algorithm's range to include neutral particles. The proposed solution involves the use of Boosted Decision Trees, a machine learning technique, to classify and interpret neutral particle data accurately. By integrating this BDT with the existing DFEI framework, the aim is to enhance the global range of this algorithm.

Chapter 2

Context

2.1 The Standard Model

The Standard Model (SM) is the theory describing the building blocks of matter. These blocks are divided into categories. There are 6 quarks and 6 leptons, organized in pairs into 3 generations. All stable matter in the universe is composed of particles that belong to the first generation; heavier particles quickly decay into stable ones [1].

The SM focuses on three of the four fundamental forces: the strong force, the electromagnetic force and the weak interaction. This theory disregards the gravitational force. The carriers of these forces are the gauge bosons. Finally, the Higgs boson is a theoretical prediction of the mechanism responsible for the mass of the particles, which was later experimentally detected thanks to the ATLAS and CMS experiments at CERN. A table of these particles is shown in Fig. 2.1.

Standard Model of Elementary Particles

three generations of matter (fermions)			interactions / force carriers (bosons)		
	I	II	III		
mass	$\approx 2.16 \text{ MeV}/c^2$	$\approx 1.2730 \text{ GeV}/c^2$	$\approx 172.57 \text{ GeV}/c^2$	0	$\approx 125.20 \text{ GeV}/c^2$
charge	$\frac{2}{3}$	$\frac{2}{3}$	$\frac{2}{3}$	0	0
spin	$\frac{1}{2}$	$\frac{1}{2}$	$\frac{1}{2}$	1	0
QUARKS	u up	c charm	t top	g gluon	H higgs
	$\approx 4.70 \text{ MeV}/c^2$	$\approx 93.5 \text{ MeV}/c^2$	$\approx 4.183 \text{ GeV}/c^2$	0	
	$-\frac{1}{3}$	$-\frac{1}{3}$	$-\frac{1}{3}$	0	
	$\frac{1}{2}$	$\frac{1}{2}$	$\frac{1}{2}$	1	
	d down	s strange	b bottom	γ photon	
LEPTONS	$\approx 0.5110 \text{ MeV}/c^2$	$\approx 105.66 \text{ MeV}/c^2$	$\approx 1776.93 \text{ MeV}/c^2$	$\approx 91.1880 \text{ GeV}/c^2$	
	-1	-1	-1	0	
	$\frac{1}{2}$	$\frac{1}{2}$	$\frac{1}{2}$	1	
	e electron	μ muon	τ tau	Z Z boson	
	$< 0.8 \text{ eV}/c^2$	$< 0.17 \text{ MeV}/c^2$	$< 18.2 \text{ MeV}/c^2$	$\approx 80.3692 \text{ GeV}/c^2$	
	0	0	0	± 1	
	$\frac{1}{2}$	$\frac{1}{2}$	$\frac{1}{2}$	1	
	ν_e electron neutrino	ν_μ muon neutrino	ν_τ tau neutrino	W W boson	

Figure 2.1: Table of the different components of the SM [2].

The SM is not a complete theory, as some phenomena, such as dark matter and CP asymmetry, are not described. These phenomena are rare and require specific condition to be observed, but potential discoveries in these fields could lead to major physics breakthroughs. The Large Hadron Collider at CERN has been built in the aim of meeting these conditions.

2.2 The LHC at CERN

The Large Hadron Collider (LHC) is a two-ring-superconducting-hadron accelerator and collider installed in the existing 26.7 km tunnel that was constructed between 1984 and 1989 for the CERN LEP machine [3]. The beams are, for most cases, composed of protons (p). In specific cases, heavy ions such as lead (Pb) are used. The LHC is the largest and most powerful particle accelerator in the world, with pp collisions producing a center-of-mass energy of approximately 14 TeV.

Fig. 2.2 shows the main components of the particle accelerator. First, the protons or heavy ions (p and Pb) go through a few linear accelerators, before being accelerated through two other preaccelerators. The first one is the Proton Synchrotron (PS) and the second is the Super Proton Synchrotron (SPS). After that, particles are introduced into the main ring of LHC. The four main experiments conducted at the LHC are highlighted in yellow (ATLAS, CMS, ALICE, LHCb). The context of this work takes place in the LHCb experiment, described in the next part.

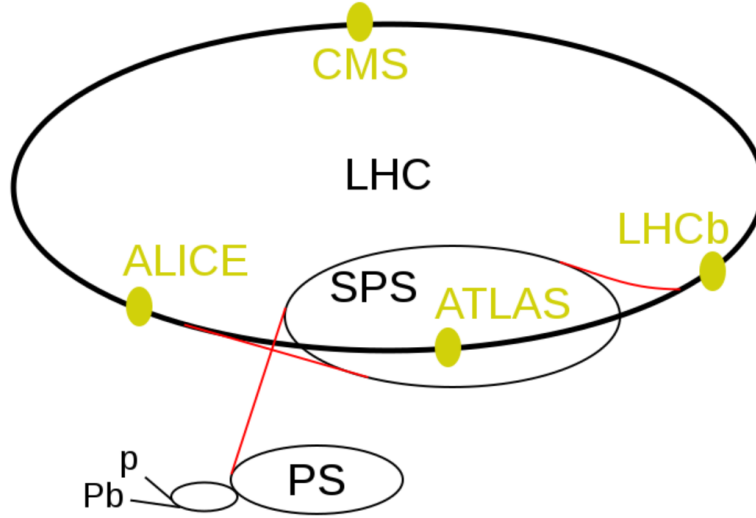


Figure 2.2: Scheme of the LHC layout [4].

2.3 The LHCb Experiment

The main goals of the LHCb experiment are to look for CP-violating processes and for indirect evidence of new physics in the rare decays of beauty and charm hadrons (hadrons that contain a b quark or a c quark) [5]. The high energy provided by the LHC allows for the production of a significant amount of these hadrons.

LHCb is a single-arm spectrometer with an angular coverage in the range of $(10 - 250)$ mrad. This geometry is justified by the fact that most of the B hadrons produced with the energy achieved by the LHC are found in that range [5]. The detector received several upgrades to prepare for the start of

LHC run 3 in July 2022. The upgraded detector layout is shown in Fig. 2.3. Right-handed coordinates are used with the z -axis defined to be along the direction of the beam (left to right in the figure) and the y -axis being vertical.

A collision of two proton bunches is referred to as an event. During an event, hundreds of particles are generated at the points of interaction between two of the protons, called the primary vertices (PV), and some others result from further decays of these particles. The points of creation of these decay products are called secondary vertices (SV). The detector needs to be capable of reconstructing many different particles, which explains the need for multiple subdetectors as described below.

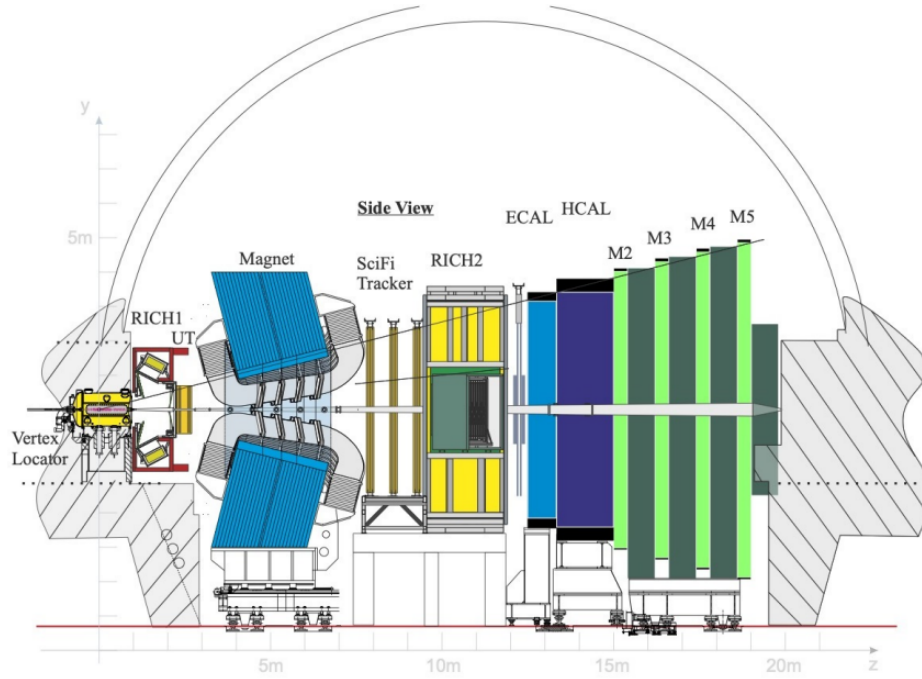


Figure 2.3: Scheme of the LHCb detector layout [6].

Vertex locator (VELO)

The VELO detects tracks of ionizing particles in the region of the initial proton-proton collision. It is composed of 26 modules of pixelated silicon detectors. It separates the primary vertex and the secondary vertices. Those modules open and close to ensure that they don't interfere with the creation

of the beam, when the beam is stabilized, the modules are closed to be at a distance of 3.5 mm of the beam.

Ring-imaging Cherenkov detectors (RICH1 and RICH2)

The Cherenkov detectors are useful for particle identification, mainly to separate pions from kaons in selected B hadron decays. This identification is based on the Cherenkov effect. A charged particle with a high relativistic velocity $\beta = v/c$, can travel faster than the speed of light c in a particular medium; when that happens, light is emitted. The refractive index n of the medium allows to predict the Cherenkov angle θ_c (the angle at which the photons are emitted by this effect), which is given by:

$$\cos \theta_c = \frac{1}{n\beta}$$

By combining the θ_c detected with the momentum p of the particles, the mass can be inferred, allowing for particle identification.

RICH1 is placed after the VELO and allows for detection of particles with momenta ranging from 2.6 - 60 GeV/c, while RICH2 is placed before the calorimeters and is sensitive in a range of higher momentum (15 - 100 GeV/c).

Upstream tracker (UT)

The UT is comprised of four silicon detector layers and is used for tracking charged particles. UT hits (the points that are gathered by the detector indicating a charged particle passed through) and the VELO tracks (trajectories of the particles) are combined to get a first assessment of the momentum p , with a precision of $\approx 15\%$. The proximity of the magnetic field generated by the magnet described below, allows to slightly bend the particles to get this first assessment. This rough estimate is used by the SciFi tracker, described below, to reconstruct the tracks.

Magnet

A warm magnet is placed to generate a magnetic field ($\vec{B}$) and thus, bend charged particles, causing them to have curved trajectories. Their curvature radius (r) depends on the particles' momentum (p) $r = \frac{p}{qB}$, allowing for precise momentum estimation.

Scintillating Fiber tracker (SciFi)

The Scintillating Fiber (SciFi) tracker is located downstream of the magnet. When a charged particle passes through the scintillating fiber, it emits lights, allowing for the reconstruction of the signal. Thanks to the bending that the trajectories of charged particles have between the different subdetectors, the angle of the reconstructed tracks are shifted. These shifts help reconstructing the momenta precisely.

Electromagnetic and hadronic calorimeters (ECAL and HCAL)

The calorimeters are placed after the tracking part of the detector because they apply a destructive measurement, meaning that the interaction between the particle and the detector material are now stronger, resulting in a significant loss of energy.

In the ECAL, photons and electrons interact through electromagnetic force. The incoming particle interacts with the atoms composing the detector. This process leads to the creation of new particles with less energy, which in turn, interact with the detector material. This process goes on until the energy of the electrons and photons are low enough to be absorbed. This chain process is called an electromagnetic shower.

The HCAL operates with the same principle, but the hadrons interact also through other interactions such as the strong force. This leads to longer and less predictable showers.

For both calorimeters, the principle is to alternate between iron absorbers and plastic scintillators to sample the light emitted by the showers. The reconstructed showers allow for reconstruction of the energy of particles.

Muon system

Since muons interact weakly with matter, they escape the calorimeters. The muon system is composed of multi-wire proportional chambers, which help track the muons' trajectories, and thick absorbers.

Neutral Particles

Most parts of the detector only detect charged particles. Neutral particles are more difficult to reconstruct and can be found only in specific regions: photons are detected by the ECAL and other neutral particles are found in the HCAL. In both cases, the track is not reconstructed. It is also possible to

indirectly detect neutral particles by detecting the charged products created by the decay of a neutral particle.

2.4 Real-time Data Analysis at LHCb

Before the upgrade, the LHCb experiment collected data at a rate of 4 TB/s. This results in the production of 100 EB (10^{18} bytes) of data by the detector annually [7]. The future increase in luminosity of the LHC accelerator demands fundamentally new approaches to triggering and online event selection [8].

Economically transferring this volume of data is unfeasible. Additionally, storing exabytes of data for analysis is extremely costly.

To address these issues, a “trigger” system is employed at LHCb. This system is designed to permanently discard the majority of “uninteresting” events, retaining only 0.01 to 0.001% of the data. The “real-time” analysis means that the data discarding process must prevent the saturation of computing resources [7].

This concept of real-time analysis is extended by splitting analysis and data reduction into two phases. A first fast analysis is conducted to reject a portion of events that do not require further inspection. Then, the remaining events undergo a second, slower analysis phase. This process is illustrated in Fig. 2.4. One stopwatch indicates the fast preliminary analysis, while two stopwatches indicate the slower, more in-depth analysis. The goal of this flow is to optimize the time required for efficient reconstruction, given that the time budget is insufficient for performing the full analysis upfront. It is important to note that this flow is also applicable to rejecting particles within a single event to extract the desired part.

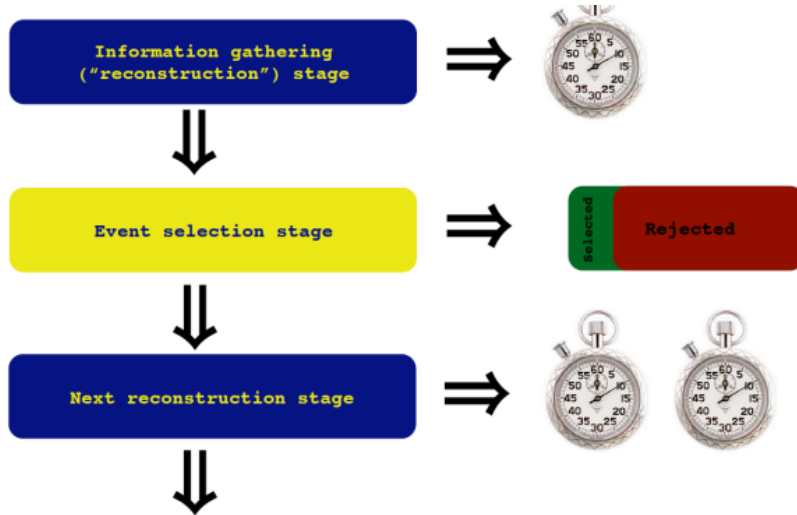


Figure 2.4: Scheme of the real-time analysis flow [7].

To clarify the real-time data analysis performed at LHCb, a brief overview of the current trigger system is helpful. The schematic description of this system is shown in Fig. 2.5. The high-level trigger 1 (HLT1) aims to reduce the event rate at which data can be buffered, it is the fast analysis described with one stopwatch in Fig. 2.4. This buffer allows for real-time alignment, calibration, and, importantly in this context, further processing in HLT2. HLT2 represents the slower analysis represented by two stopwatches in Fig. 2.4. The data selected by HLT2 is then stored on disk for offline analysis.

Future upgrades at LHCb will increase the rate of the full detector read-out, demanding a significant advancement in real-time data analysis. This requirement motivates the exploration of the Deep Full Event Interpretation (DFEI) using Graph Neural Networks (GNN), which is described below.

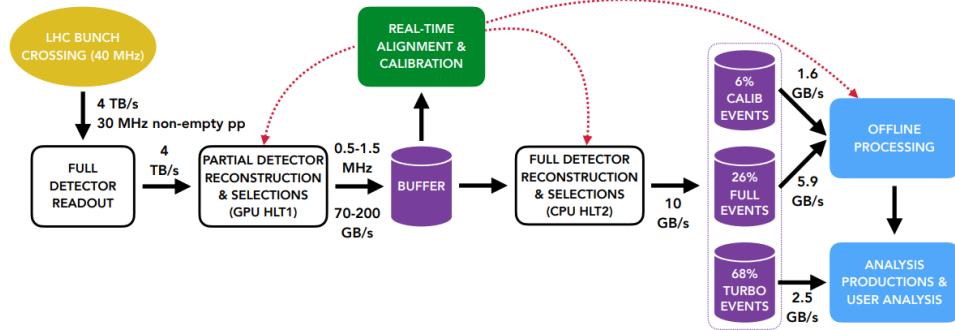


Figure 2.5: Scheme of the data management at LHCb [6].

2.5 Graph Neural Network for Deep Full Event Interpretation

In previous LHC runs, the disk space available to store the events of interest was sufficiently large to allow most of the particles in a particular event to be collected. The study of particles, other than the ones that triggered the event gathering and the background rejection, was conducted offline. With the increasing output rate of the upgraded detector, continuing with that method proves to be difficult as it would saturate the data storage. Furthermore, the size of the events also increases due to higher particle multiplicities and detector granularity. These problems motivate a shift in the trigger paradigm from “which events are interesting?” to “which parts of the event are interesting?” [9].

The principle of the Deep Full Event Interpretation (DFEI) algorithm is quite simple: for a given event, each event is put in the form of a graph. In the general sense, a graph is a structure representing several objects, represented by nodes. Between these nodes, edges can be drawn to represent the relations between two nodes. In the case of DFEI, a node contains data about a particle, such as transverse momentum (p_t). On the other hand, an edge contains information like θ , which is the opening angle between two particles. In this particular example, the goal is to select events that originate from a B hadron.

Fig. 2.6 illustrates the three steps performed by the algorithm to reduce a collision event to the particular event of interest. These steps are as follows:

- Node pruning (NP) is the first step of training that the algorithm undergoes. Most of the particles that do not originate from a B decay are discarded.
- Edge pruning (EP) is subsequently performed with the goal of cutting all edges between particles that do not share a common B hadron ancestor. This can be done using various relations, such as the spatial proximity of these particles.
- Lowest common ancestor inference (LCAI) is the final step where the decay scheme's final ordering is determined. This step allows for the identification of mother and daughter particles.

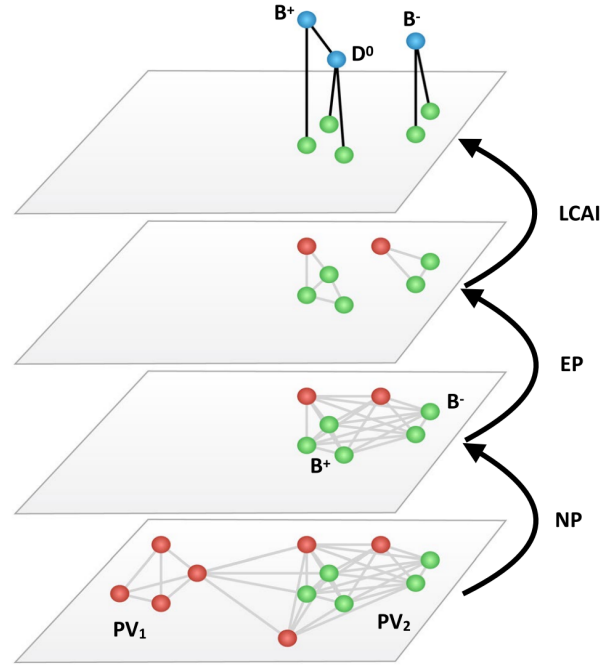


Figure 2.6: Scheme of the graph reduction process used by the algorithm aiming at selecting specific B decays [9].

Fig. 2.7 illustrates the idea of DFEI with a simplified event. All of the particles detected in the event are reconstructed; then, they are selected to only keep the products of B decays.

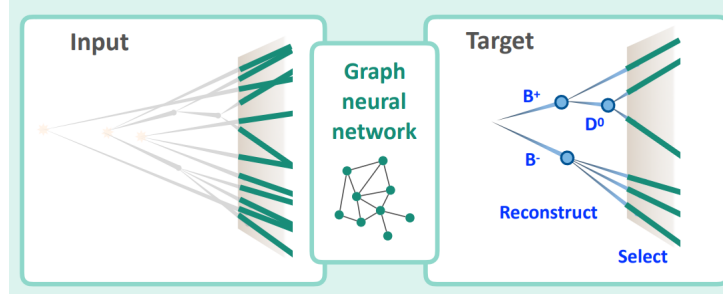


Figure 2.7: Scheme of the DFEI algorithm. [10].

Results for selection efficiency of the current version of the DFEI algorithm are shown in Fig. 2.8. The average selection efficiency for particles that come from a B decay is 94%, while the background rejection is 96%. The average number of selected particles is approximately 10, compared to an initial average number of particles of around 140 [9]. These results demonstrate great promise and justify further investigation.

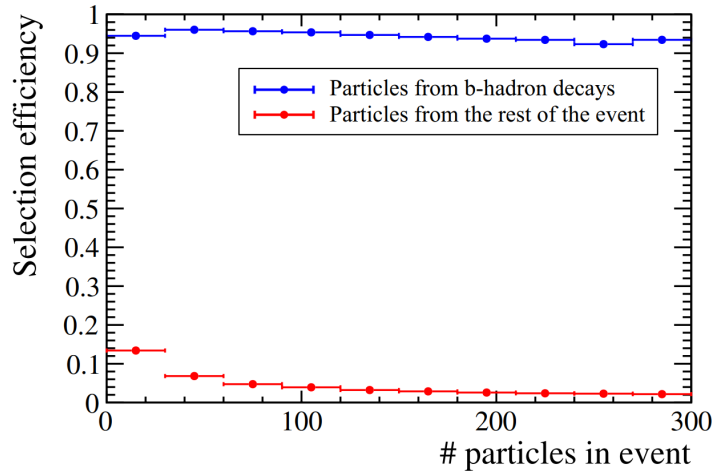


Figure 2.8: Results for the selection efficiency of B decays and background rejection as a function of the number of particles in a particular event [9].

It is important to note that the work of the team developing DFEI currently focuses only on B hadrons, as they are more typically studied at LHCb and their lifetime is longer than that of C hadrons, making the secondary vertices from B decays easier to identify.

The algorithm also disregards neutral particles, as less experimental information is available for those particles. The main goal of this work is to extend the work of DFEI, to include neutral particles. Therefore, a smart method must be devised to also select neutrals originating from B hadrons.

2.6 Bremsstrahlung Recovery at LHCb

Bremsstrahlung is a German word meaning braking radiation. It refers to the process in which decelerating charged particles emit electromagnetic radiation [11]. This process appears through the interaction of the charged particles with the Coulomb field of the atoms composing the detector material. The probability of this interaction is proportional to E^2/m , meaning this effect mainly affects electrons, due to their low mass [12].

Electrons that emit Bremsstrahlung lose energy, making momentum estimation and track reconstruction more difficult. The Bremsstrahlung recovery algorithm currently in place at LHCb is described in Fig. 2.9.

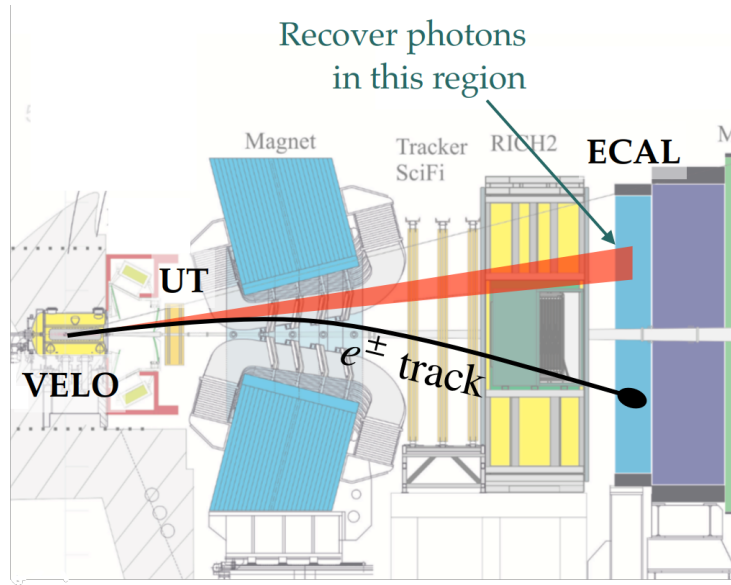


Figure 2.9: Scheme of the LHCb Bremsstrahlung recovery algorithm [12].

The upstream tracks of the electrons are extrapolated to the ECAL. The upstream track is the part of the particle track that comes before the magnet

(and thus the bending). After recovery, the energy of the Bremsstrahlung photon is added back to the electron, allowing for better momentum estimation.

At LHCb, electrons are expected to produce on average ≈ 1 photon. The 50% efficiency of the reconstructed signal is calculated assuming every electron has emitted one Bremsstrahlung photon [12].

2.7 LHCb Simulation

Monte Carlo simulated data is essential for the detector design, commissioning, and physics analysis. Simulated data is processed through a data flow identical to that of pp collision data, to define, tune, and validate reconstruction and selection algorithms. Simulated data also undergo online high-level trigger (HLT) and offline data processing, just like the real process of data gathering.

The software for generating simulated events in LHCb is encapsulated in two separate applications. The **Gauss** package is responsible for event generation and simulates particle interactions in the detector, resulting in energy deposits or hits in the subdetectors. The **Boole** package models the detector and readout electronics response by converting the hits into specific subdetector signals.

The information generated by the **Gauss** and **Boole** packages is then transparently propagated through all data processing steps pertaining to collision data.

For the initial exploratory studies of this thesis work, a simplified simulation is used, similar to the bare output of **Gauss** but using a simplified detector model.

Chapter 3

The Classifier Algorithm

3.1 Decision Tree Classifier

A decision tree is a Machine Learning (ML) algorithm that falls into the category of classifiers. A classifier is an algorithm with the objective of assigning data to different categories. In high-energy physics, it is common to treat problems with two categories, which are called background and signal. One could also use 0 and 1, or False and True. The category in which a particular entry falls is called its label. In the context of this work, an entry of a dataset refers to a particular row which, in the case of an ML algorithm has its variables (features) and its label (true category).

It is recommended to split the dataset entries into two subsets: the training set and the testing set. The training set is used to train the algorithm to make predictions on the label of each entry. The algorithm has access to the label of the training entries. At no point does the algorithm access the labels of testing entries; it is simply used to compare the prediction in order to assess the performance of the algorithm.

The principle of a decision tree is quite simple. Consider a sample of signal (s_i) and background (b_j), described by a set of variables $\vec{x}_i$. This sample constitutes the root node of a new decision tree. Starting from the root, the algorithm proceeds as follows [13]:

1. If the node satisfies any stopping criterion, declare it as terminal (that is, a leaf) and exit the algorithm.
2. Sort all events according to each variable in $\vec{x}$.

3. For each variable, find the splitting value that gives the best separation between two children, this process is explained below. If the separation can not be improved by any splitting, turn the node into a leaf and exit the algorithm.
4. Select the variable and splitting value leading to the best separation and split the node in two new nodes (branches), one containing events that fail the criterion and one with events that satisfy it.
5. Apply recursively from step 1 on each node.

Once this algorithm has been trained, the resulting tree (see in Fig. 3.1) is applied to the testing set.

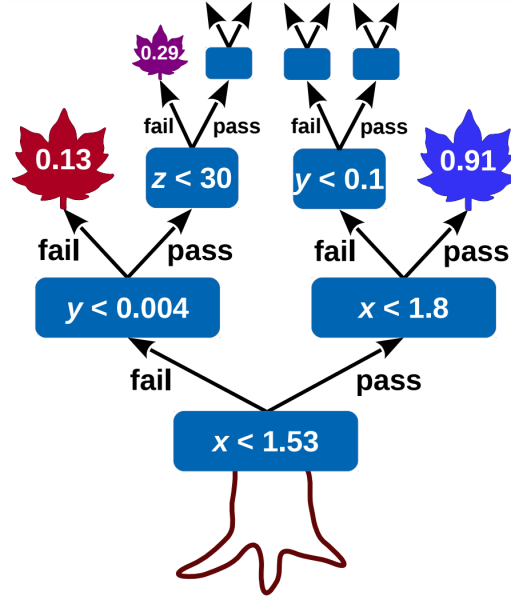


Figure 3.1: Scheme of the decision tree algorithm [13].

For any entry of the testing set i , it is easy to follow the branches on the tree to reach the final leaf. When a particular entry reaches a leaf, the purity of the leaf: $p = \frac{s}{s+b}$ is the number that will be assigned to that entry, it is called the prediction. By selecting a threshold, for instance 0.5, it is possible to turn this prediction into the predicted label (if $pred < 0.5$; background, if $pred \geq 0.5$; signal).

The question of node splitting is central to the architecture of the decision tree. An impurity measure is introduced: $i(t)$ where t is a specific node. Naively this function is wanted to be maximal for an equal mix of signal and background, and minimal when there is either signal or background only. Many different functions satisfy these conditions, but traditionally, for classifiers [14], the cross-entropy has proven to be the most efficient:

$$i(t) = - \sum_{i=s,b} p_i \log p_i$$

The goal is to find a split s^* of the node t into children t_P and t_F such that:

$$\Delta i(s^*, t) = \max_{s \in \{\text{splits}\}} \Delta i(s, t)$$

with

$$\Delta i(s, t) = i(t) - p_P \cdot i(t_P) - p_F \cdot i(t_F)$$

Minimizing overall tree impurity is equivalent to maximizing $\Delta i(s, t)$ [15]

3.2 Boosting

The concept of boosting applies to any classifier and goes as follows [13]:

- Train a classifier T_1 on a sample of N events.
- Train T_2 on a new sample with N events, half of which were misclassified by T_1 .
- Build T_3 on events where T_1 and T_2 disagree.

The boosted classifier is defined with a majority vote on the outputs of T_1 , T_2 and T_3 .

If this concept is applied, the algorithm is called a Boosted Decision Tree (BDT).

3.3 Hyperparameters Optimization

A hyperparameter of an ML algorithm is a parameter that is not learned through training [16]. They are crucial because they determine the structure

of the algorithm and play a role in the overtraining prevention. An example of a hyperparameter for a BDT would be the maximum depth a tree can reach.

Since a BDT has several dependent hyperparameters, they need to be studied together. To achieve this, selected hyperparameters are varied to optimize a chosen objective function. The choice of this function is crucial for creating a good model.

An example of overtraining is illustrated in Fig. 3.2.

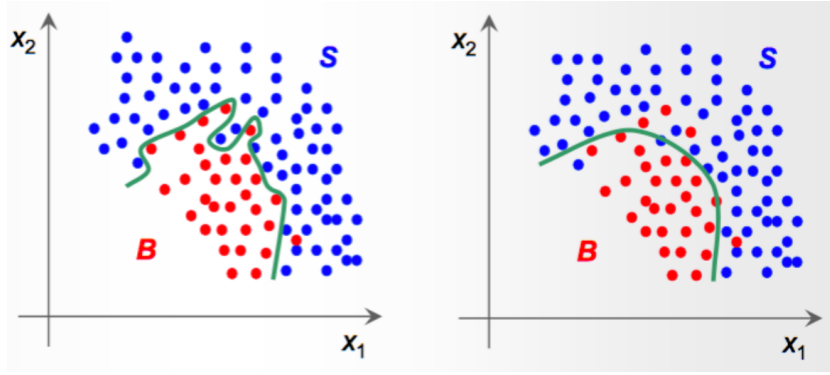


Figure 3.2: Comparison of two classifiers. Left: Overtrained classifier. Right: Optimal classifier [17].

Considering this set of points having 2 features (x_1 and x_2), classified into 2 categories (blue and red), two different classifiers are shown. On the right, the classifier delimits a boundary that wrongly classifies some points, but provides a good description for the distribution of the points.

On the other hand, the classifier on the left aims to perfectly classify each and every point, which leads to problems since it also contains information on the noise and on unimportant features of the category distributions. This is called overtraining.

3.4 Feature Correlation

Empirical evidence from the feature selection literature shows that, along with irrelevant features, redundant information should be eliminated. A feature is said to be redundant if one or more of the other features are highly correlated with it [18].

In the context of this work, only linear correlation between features is verified. First, it is necessary to clarify some statistical variables. Let $\{x_1, \dots, x_n\}$ and $\{y_1, \dots, y_n\}$ be two samples of n entries for the two features x and y . The mean and standard deviation of x (μ_x or $\bar{x}$ and σ_x) are given by Eq. 3.1.

$$\begin{aligned}\mu_x &= \frac{1}{n} \sum_{i=1}^n x_i = \bar{x} \\ \sigma_x &= \sqrt{\frac{1}{n} \sum_{i=1}^n (x_i - \bar{x})^2}\end{aligned}\tag{3.1}$$

Then, the covariance between x and y is given by [19]:

$$\text{cov}(x, y) = \frac{1}{n-1} \sum_{i=1}^n (x_i - \bar{x})(y_i - \bar{y})$$

The linear correlation coefficient also known as Pearson correlation coefficient $\rho_{x,y}$ is given by Eq. 3.2.

$$\rho_{x,y} = \frac{\text{cov}(x, y)}{\sigma_x \sigma_y}\tag{3.2}$$

Two features that are linearly correlated mean the information they convey is very similar, which is unwanted in a ML algorithm in general.

3.5 Recursive Feature Elimination

To determine the quality of the features of a BDT, it is common to rank them. The feature “importances” are calculated based on the average gain of the feature. The gain is a value that assesses the score improvement of a particular leaf [20], the mean gain in every occurrence of the feature in a leaf is taken to determine the importance.

The process of recursive feature elimination (RFE) [21] is based on the feature rankings. The principle is simple. First the BDT is applied using all of the features, then the lowest performing feature is discarded. The BDT is then reapplied with one feature less, and the process can be repeated at will. The evolution of performance metrics discussed below through the application of RFE helps assessing the optimal number of features required for the model.

3.6 Performance Metrics

The application of a BDT algorithm allows for predictions associated with each entry. The prediction ranges between 0 and 1, which allows for the creation of the output plot (example in Fig. 3.3).

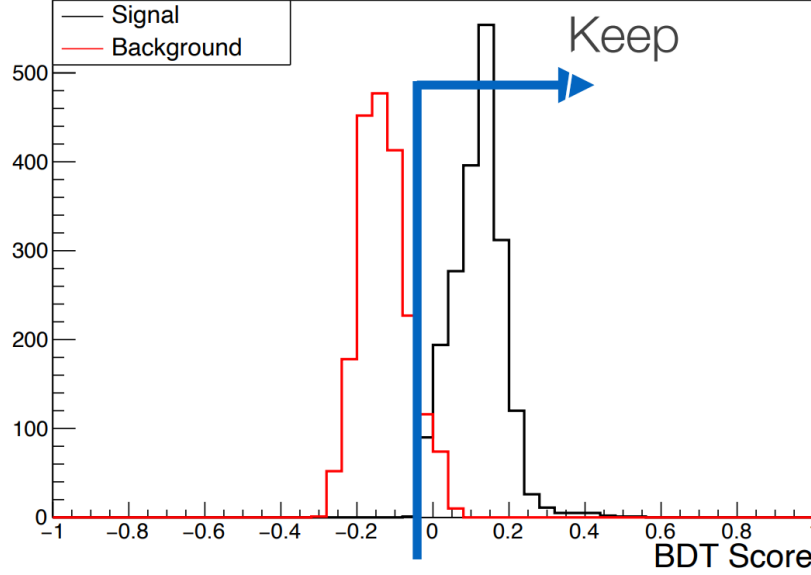


Figure 3.3: Example of the output plot generated by a BDT classifier with a selected threshold [17].

It shows the distribution of the predictions. It is important to understand that this example is for a classifier which classifies into categories -1 and 1 , which is equivalent, after rescaling, to having 0 and 1 as categories. By selecting a threshold, every entry in the dataset is assigned a category, which leads to new metrics associated with classification.

Fig. 3.4 summarizes the common terminology for classification both in general and in the field of high-energy physics. A true positive (TP) or selected signal (S_{sel}) is a signal (1) entry which has been correctly classified, a false positive (FP) or selected background (B_{sel}) is a background (0) entry which has been wrongly classified. Furthermore, a true negative (TN) or rejected background (B_{rej}) is a background (0) entry that is correctly classified, while a false negative (FN) or rejected signal (S_{rej}) is a signal (1) entry that is wrongly classified.

<table><tr><td>TP (S_{sel})</td><td>FP (B_{sel})</td></tr><tr><td>FN (S_{rej})</td><td>TN (B_{rej})</td></tr></table>	TP (S_{sel})	FP (B_{sel})	FN (S_{rej})	TN (B_{rej})	<table><tr><td>TP (S_{sel})</td><td>FP (B_{sel})</td></tr><tr><td>FN (S_{rej})</td><td>TN (B_{rej})</td></tr></table>	TP (S_{sel})	FP (B_{sel})	FN (S_{rej})	TN (B_{rej})	<table><tr><td>TP (S_{sel})</td><td>FP (B_{sel})</td></tr><tr><td>FN (S_{rej})</td><td>TN (B_{rej})</td></tr></table>	TP (S_{sel})	FP (B_{sel})	FN (S_{rej})	TN (B_{rej})
TP (S_{sel})	FP (B_{sel})													
FN (S_{rej})	TN (B_{rej})													
TP (S_{sel})	FP (B_{sel})													
FN (S_{rej})	TN (B_{rej})													
TP (S_{sel})	FP (B_{sel})													
FN (S_{rej})	TN (B_{rej})													
$\text{TPR} = \frac{\text{TP}}{\text{TP} + \text{FN}}$	$\text{PPV} = \frac{\text{TP}}{\text{TP} + \text{FP}}$	$\text{TNR} = \frac{\text{TN}}{\text{TN} + \text{FP}} = 1 - \text{FPR}$												
HEP: “efficiency” $\epsilon_s = \frac{S_{\text{sel}}}{S_{\text{tot}}}$	HEP: “purity” $\rho = \frac{S_{\text{sel}}}{S_{\text{sel}} + B_{\text{sel}}}$	HEP: “background rejection” $1 - \epsilon_b = 1 - \frac{B_{\text{sel}}}{B_{\text{tot}}}$												
IR: “recall”	IR: “precision”	—												

Figure 3.4: Scheme of the classification report [22]. TPR stands for True Positive Rate, PPV for Positive Predictive Value, TNR for True Negative Rate and FPR for False Positive Rate. The high-energy physics (HEP) terminology is explained; signal efficiency ϵ_f and background rejection $1 - \epsilon_b$ are core concepts. IR refers to information retrieval, the terminology is the most common in ML classifiers.

The number of entries in each category heavily depends on the threshold chosen for selection. The accuracy is defined as the fraction of entries correctly classified, using the default threshold (in the case of the two categories being 0 and 1, it is 0.5).

The receiver operating characteristic (ROC) curve attempts to illustrate the performance of classifiers. By changing the threshold, a classifier goes through different true positive rates (TPR or signal efficiency) and false positive rates (FPR or background efficiency). The ROC curve is the signal efficiency as a function of the background efficiency.

Fig. 3.5 illustrates the concept of ROC curve. The perfect classifier assigns each entry correctly with prediction 0 or 1, which means that, regardless of the threshold, every entry in the dataset is correctly classified, hence $\epsilon_s = \epsilon_b = 1$.

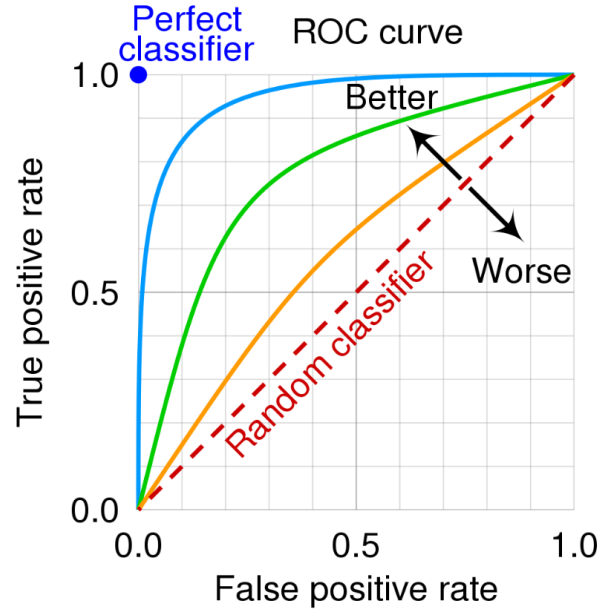


Figure 3.5: Scheme of the ROC curve [23].

The area under the curve (AUC) allows to get a quick assessment of the classifier's performance, knowing that the AUC of a random classifier is 0.5, and 1 for an ideal classifier. The higher the AUC is, the better the classifier performs.

The ROC curve also allows for specific signal efficiency ϵ_f , or background rejection $1 - \epsilon_b$ targeting, to choose the corresponding threshold. This choice depends on the aim of the algorithm.

Chapter 4

Data Processing

4.1 Description of the Datasets

The essential tools for proceeding with the analysis are two datasets described in this section. The first dataset, which will be called the **Particles** dataset, contains all the particles simulated over 49399 events and several pieces of information about those particles, described below. From the graph perspective, this dataset contains node information.

The second dataset, called the **Relations** dataset, contains all the pairs of particles present in the **Particles** dataset and their relationships. This means each entry (row) consists of two particles. This dataset provides information about the edges of the graph. The full paths of these two datasets are found in Appendix B.

The simulated datasets are parametric and simplified versions from “complete” LHCb simulation data, because these datasets were readily available. These simplified versions do not take into account detector inefficiencies and disregard the effects of material interaction or fake tracks. Fake tracks are defined as those reconstructed tracks which do not correspond to the trajectory of a true particle but instead are due to the mismatch of hits from separate particles or from detector noise [24].

The truth-level information of the **Particles** dataset is as follows:

- **EvtNumber**: The event number on which the particles appear, useful for processing the particles of a particular event.
- **ParticleIndex**: An index assigned to the particles, which is useful to identify wanted particles.

- **DecayIndex**: The index of the B decay that the particle is associated with. If the particle does not originate from a B decay, the value of this index is -1 .
- **inGeomAcc**: This variable returns 1 if the particle is within the geometrical acceptance of the detector; otherwise, it returns 0.
- **isCharged**: This variable returns 1 if the particle is charged; otherwise, 0.
- **isFinal**: This variable returns 1 if the particle is final; otherwise, 0. A non-final particle is a particle that is not reconstructed by the detector, but is present in the simulation. For instance, a neutral particle that originated from a secondary vertex and has decayed.

The following features are available in the dataset as truth-level information and also as reconstructed variables:

- **px, py, pz and pt**: The three components of the momentum of the particle (p_x, p_y, p_z) and the transverse momentum $p_t = \sqrt{p_x^2 + p_y^2}$.
- **xProd, yProd and zProd**: The coordinates of the production vertex of the particle $(x_{prod}, y_{prod}, z_{prod})$; these coordinates are not reconstructed for neutral particles.
- **xPV, yPV and zPV**: The coordinates of the primary vertex associated with the particle (x_{PV}, y_{PV}, z_{PV}) .

It is important to note that, since the production vertex is not reconstructed for neutral particles due to the lack of track reconstruction, their coordinates are set by default at $(0, 0, 0)$, the origin point of the coordinate system, which is located in the center of the pp interaction region.

The main features of the **Relations** dataset are as follows:

- **FirstParticleIndex** and **SecondParticleIndex**: The indices of the two particles in an entry. These indices correspond to the **ParticleIndex** in the **Particles** dataset.
- **p1isCharged** and **p2isCharged**: The same values as **isCharged** for both particles.

- **DOCA**: Distance of closest approach. It simply is the smallest distance between two tracks.
- **trdist**: The transverse distance between the two particles.
- **theta**: The opening angle θ between the two particles.
- **FromSameHeavyHadron**: This value returns 1 if the two particles originate from the same B decay; otherwise, it is 0.

Notice that **DOCA**, **trdist** and **theta** are available both at truth level and reconstruction level.

Fig. 4.1 illustrates the meaning of the transverse distance and the opening angle θ . The transverse distance is calculated with the combined 3-momentum (p_x, p_y, p_z). It is the distance between the projection of the two particles on a perpendicular line to the combined 3-momentum. The opening angle is simply the angle between the 3-momenta of both particles.

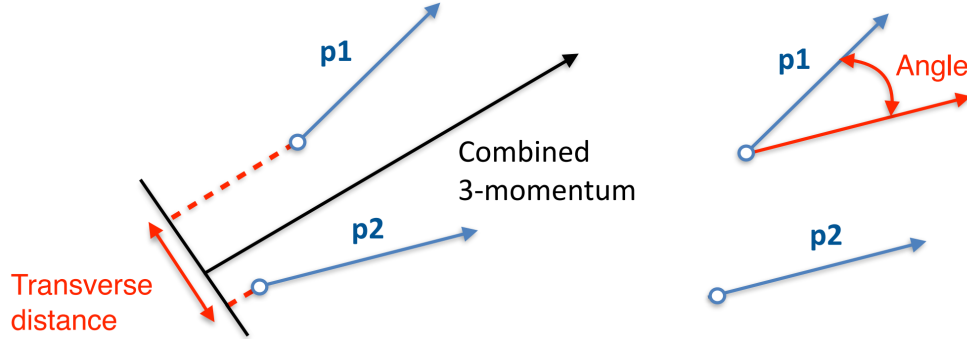


Figure 4.1: Description of transverse distance and opening angle θ [10].

Since the production vertex is not reconstructed for neutral particles, **DOCA** and **trdist** are features that are not expected to be well reconstructed for neutral particles, as the production vertex from which these values are calculated is set by default to $(0, 0, 0)$.

A quick description of the dataset is useful. The left part of Fig. 4.2 shows the fraction of neutral particles per event. This Gaussian distribution indicates that events contain $(35.7 \pm 3.71)\%$ neutral particles. The right part shows the total amount of neutral particles in the event. This clearly

shows that extending the work of the DFEI algorithm to neutral particles is necessary.

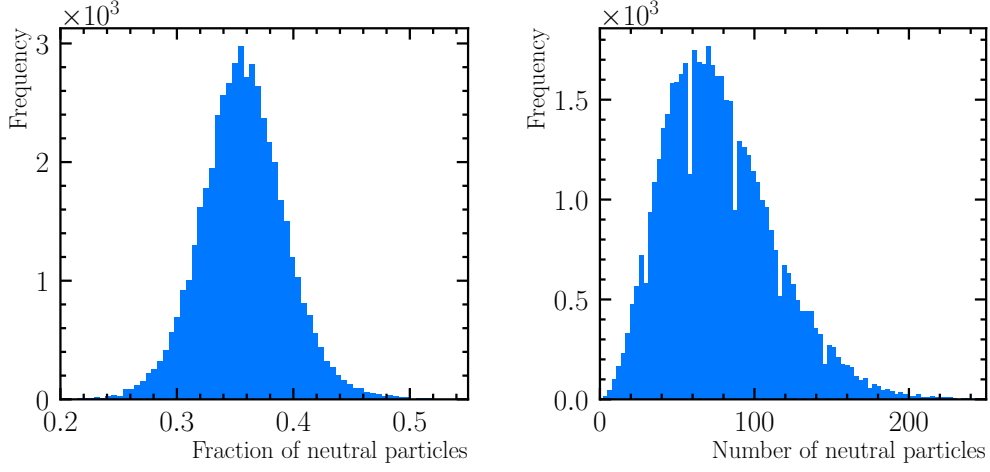


Figure 4.2: Left: Histogram of the fraction of neutral particles per event. Right: Histogram of the total number of neutral particles per event.

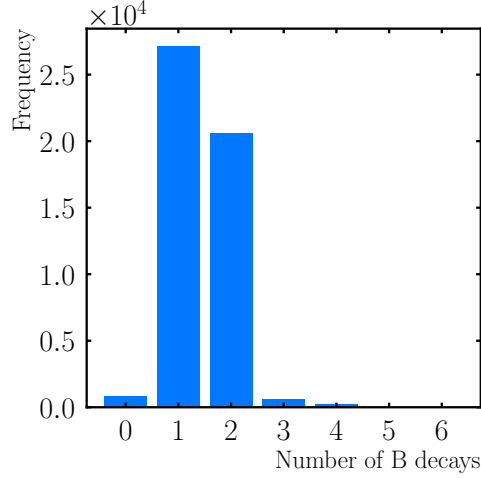


Figure 4.3: Histogram of the number of B decays and their frequency of apparition in the events.

Fig 4.3 shows the number of B decays per event. Most of the events contain 1 or 2 B decays, which produce on average 2.72 ± 1.93 neutral particles and 4.02 ± 2.15 charged particles per decay.

4.2 Data

In the context of DFEI, the computation time scales quadratically with the number of nodes [9]. The straightforward approach to treat neutral particles at the same time as the charged particles in the same way as the original DFEI algorithm seems unrealistic, since it would involve a drastic increase in the number of nodes.

The approach adopted for this thesis work relies on using the output of the DFEI algorithm, more specifically using the selection of charged particles that DFEI makes. Thanks to this output, information on the charged part of the B decays is gathered, which will be compared to the neutral particles, with the objective of assigning each neutral particle to a specific decay. The similarity in momentum and the spatial proximity between the neutral particle and the charged part of the B decays will constitute the most important features.

With this approach, the algorithm works after DFEI, in an additional layer, since it needs its output.

To set up an algorithm with this goal, a single dataset that contains both node and edge informations is needed; therefore, the **Particles** and **Relations** datasets need to be merged. A simple way of arranging the entries, features and labels needs to be found.

For this work, truth-information for the charged particles selection is used, rather than the output from the DFEI algorithm.

The datasets described in 4.1 are shortened to only retain relevant entries. The **Particles** dataset is reduced to only contain the charged particles already assigned to a B decay and all the neutrals within the geometrical acceptance of the detector, that are final. This reduction creates a new, smaller dataset called **Particles_ND** (ND stands for neutral and decay, indicating that only neutral particles and the charged part of B decays are remaining in the dataset).

The **Relations_ND** set is also processed to contain all the neutral-charged relations between particles in **Particles_ND** (neutral-neutral and charged-charged relations are discarded).

An analysis of the quality of reconstructed information is provided. Since both truth-level and reconstructed information are available in the datasets, it is easy to evaluate the difference between the two. The histograms in Fig. 4.4 and 4.5 show the distribution of the normalized residuals (Eq. 4.1) for a few different features.

$$R_{\text{norm}}^x = \frac{x_{\text{reco}} - x_{\text{truth}}}{x_{\text{truth}}} \quad (4.1)$$

where x is a feature.

The distribution for normalized residuals of the distance of closest approach (DOCA) has a similar form to that of transverse distance, as shown in Fig. A.1 in Appendix A. This confirms that DOCA and transverse distance are not well reconstructed for relations involving neutrals. On the other hand, θ , p_t and p_z appear to have Gaussian distributions with means μ and standard deviations σ (Eq. 3.1) shown in Tab. 4.1.

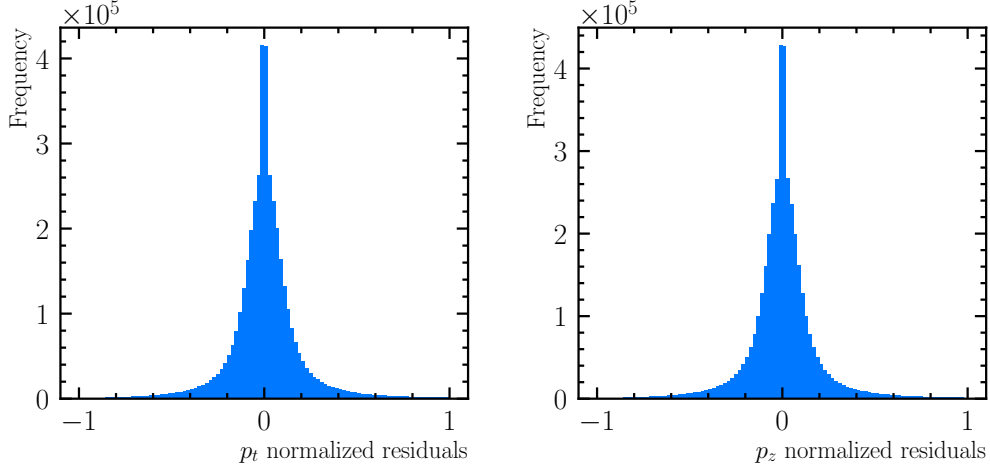


Figure 4.4: Left: Histogram of the normalised residuals of the transverse momentum p_t . Right: Histogram of the normalised residuals of the longitudinal momentum, p_z .

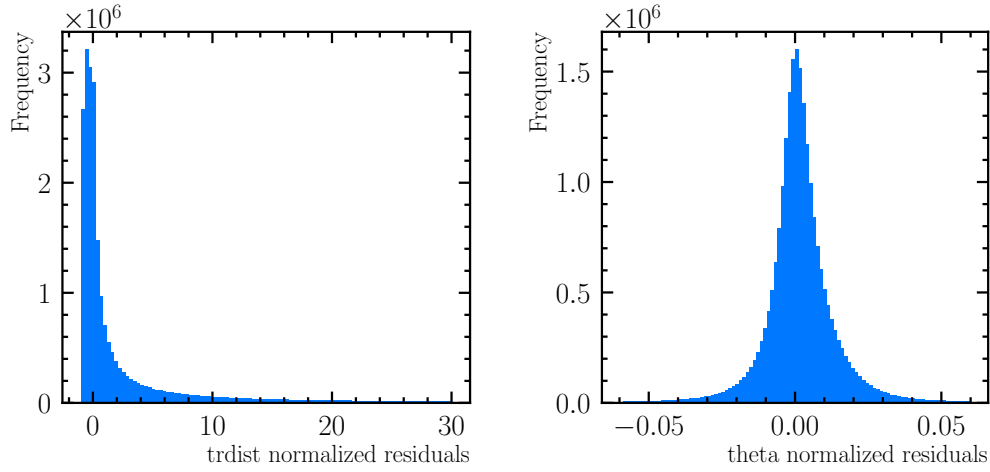


Figure 4.5: Left: Histogram of the normalised residuals of the transverse distance. Right: Histogram of the normalised residuals of the opening angle θ .

Table 4.1: Mean and standard deviation of the normalized residuals of features of the datasets.

Feature	p_t	p_z	θ
μ	2.07×10^{-2}	1.85×10^{-2}	1.85×10^{-3}
σ	53.7×10^{-2}	53.4×10^{-2}	23.1×10^{-3}

These results show that θ is the feature best reconstructed, after that comes the momentum. Note that p_x and p_y have been replaced by p_t . As expected, the reconstruction of the `trdist` and `DOCA` (see Fig. A.1, in Appendix A) is poor when it comes to neutral-charged relations, as the neutral tracks are not detected.

After the selection of particles of interest in the `Particles_ND` and `Relations_ND` datasets, a merging into a single dataset is required in order to train a BDT. This dataset is called: BDT dataset, described in Fig. 4.6, with a small example subset.

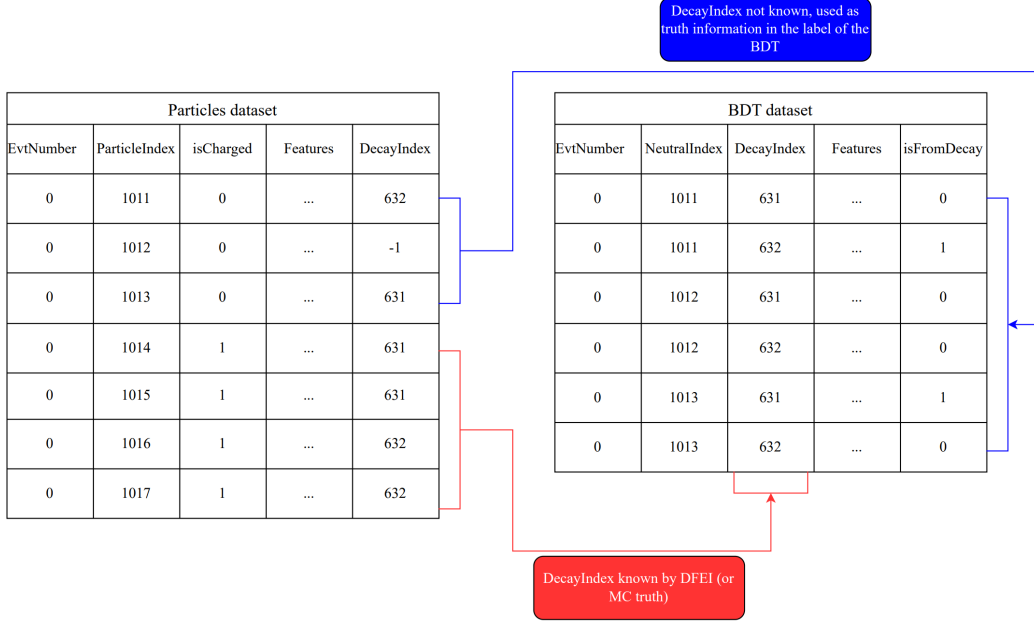


Figure 4.6: Scheme of the creation of the BDT dataset.

Each entry (row) in the BDT dataset refers to one neutral particle and one B decay, which means that for each event, every neutral particle is processed with every single B decay. This implies that for a particular event, there are $N \cdot n$ entries, where N is the number of neutral particles in the event, while n is the number of B decays in the event.

As it is the starting point of this work's approach, the **DecayIndex** of charged particles is known a priori by using truth information. The **DecayIndex** of neutral particles, on the other hand, is used for the label **isFromDecay**. Indeed the **DecayIndex** allows for each entry to be assigned a value for **isFromDecay**; 1 if the **DecayIndex** of the neutral particle in the **Particles** dataset matches with the **DecayIndex** of its entry in the BDT dataset, 0 otherwise. The three columns **NeutralIndex**, **DecayIndex** and **EvtNumber** are used as identifiers, they are neither features nor labels. The column **isFromDecay** is the label that the BDT should learn to predict.

The features are either event-wise (global features of the underlying pp collision event), or represent the relation between the candidate neutral particle and the charged particles belonging to the decay. This way, the BDT classifier predicts the likelihood of a neutral particle belonging to a certain

B decay.

This method of processing the dataset discards B decays that only produce neutral particles, as it is not possible to compare the neutrals to charged particles in such decays. In total, 3.12% of the events contain a B decay that produces no charged particles, representing 2.19% of the total amount of B decays in the dataset.

4.3 Features Analysis

A list of 15 features are chosen for the algorithm a priori. Any superfluous feature is easily detected by processing the dataset and analyzing the algorithm's results. This is the reason why a high number of features are initially proposed.

The event-wise features are as follows:

- **NNeutrals**: The total number of neutral particles in the event.
- **NHeavyHadrons**: The total number of B decays in the event.
- **pt_sum_all_neutral**: $\sum_i p_{t,i}$, where i is the index of neutral particles in a particular event.
- **pz_sum_all_neutral**: $\sum_i p_{z,i}$

There are also features that describe only the neutral particle or the charged particles in a decay:

- **pt**: The transverse momentum p_t of the neutral particle.
- **pz**: The longitudinal momentum p_z of the neutral particle.
- **pt_sum_charged**: $\sum_j p_{t,j}$, where j is the index of a charged particle in a particular B decay.
- **pz_sum_charged**: $\sum_j p_{z,j}$

The remaining features describe the relationship between the neutral and the charged particles in the decay:

- **pt_sum**: $\sum_j p_{t,j} + p_{t,i}$, where i is the index of neutral particles in a particular event and j is the index of a charged particle in a particular B decay.

- **pz_sum**: $\sum_j p_{z,j} + p_{z,i}$
- **pt_diff**: $\frac{p_{t,i} - \frac{1}{n} \sum_j p_{t,j}}{\frac{1}{n} \sum_j p_{t,j}}$
- **pz_diff**: $\frac{p_{z,i} - \frac{1}{n} \sum_j p_{z,j}}{\frac{1}{n} \sum_j p_{z,j}}$, where n is the total number of charged particles in the particular B decay. This means $\frac{1}{n} \sum_j p_{t,j}$ is the mean of the transverse momenta of all the charged particles in an event.
- **theta_mean**: $\frac{1}{n} \sum_j \theta_j^i$, where θ_j^i is the opening angle between the charged particle j and the neutral particle i .
- **DOCA_mean**: $\frac{1}{n} \sum_j \text{DOCA}_j^i$.
- **trdist_mean**: $\frac{1}{n} \sum_j \text{trdist}_j^i$.

It is important to note that the momenta, DOCA, trdist and θ are taken as reconstructed values, because the goal is to assess the possibility of implementing the resulting algorithms in the real case. The goal of these features is to assign the neutral particles to B decays based on the spatial proximity and momentum similarity of the neutral particles with the charged part of the B decays.

Fig. 4.7 shows the correlation matrix of the different features. The Pearson correlation coefficient (Eq. 3.2), discussed in 3.4, is calculated for every pair of feature. It is clear that some features are heavily correlated, which means that some features can be discarded.

for all the features are created and compared, to determine which features provide the most separating power. The two parts corresponding to the label categories are normalized, making it easier to interpret the results, as there are more entries where the neutral particle is not assigned to a B decay (`isFromDecay = 0`).

The distributions of the best visibly separated features, `theta_mean` and `NNeutrals`, are shown in Fig. 4.8. It is evident that the distributions are partially separated and a clear tendency is observed. On the left, it is shown that the entries where the neutral particle is assigned to the decay tend to have a smaller opening angle, which confirms that spatial proximity is a great feature to exploit.

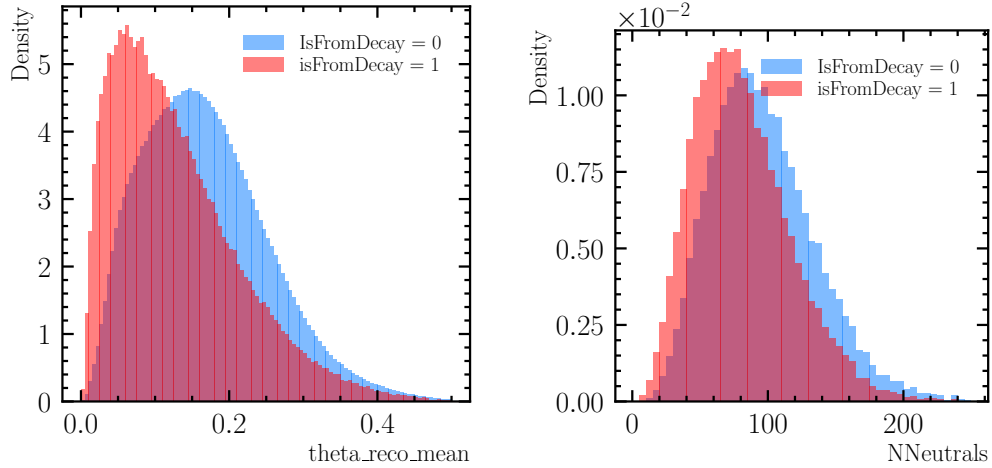


Figure 4.8: Left: Histogram of the separated and normalized distributions; `isFromDecay = 0` in blue, `isFromDecay = 1` in red, of the feature `theta_mean`. Right: Histogram of the separated distributions of the feature `NNeutrals`.

On the right, the separation power does not seem as intuitive as the one on the left. The red distribution appears to have a softer peak compared to the blue one, for the total number of neutrals in an event. The reason why the red bins are higher than the blue ones when `NNeutrals` < 100 is explained in the following.

In the BDT dataset, only events with at least one B decay are considered, which means that the fraction of neutral particles that come from a B decay

over the total amount is higher for low `NNeutrals`, and vice-versa. These fractions are shown in Fig. 4.9.

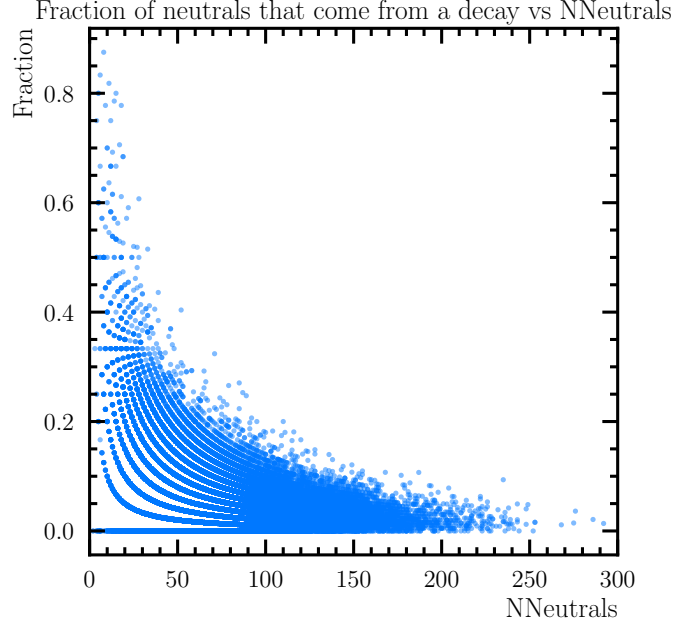


Figure 4.9: Fraction of neutral particles that come from a B hadron over the total amount as a function of `NNeutrals`.

These fractions explain the difference in height observed in the histograms on the right side of Fig. 4.8. It is interesting to see that `NNeutrals`, while being an event-wise feature that is not based on spatial proximity, can provide significant separation.

In appendix A, the rest of the distributions for all remaining features are shown (Figs. A.2, A.3, A.4, A.5, A.6 and A.7).

Chapter 5

Results

5.1 Application of the Boosted Decision tree

5.1.1 BDT with 15 Features

After preprocessing, a BDT (see in 3.1 and 3.2) is trained to assign the neutral particles to a specific B decay, using the features described in the previous chapter. The library `xgboost` is used, it uses “extreme gradient boosting”, which is an efficient way to boost decision trees [25].

The dataset is split into a training set and a testing set, with the latter representing 20% of the total dataset size. When training a classifier, it is important to have an equivalent number of entries in each category. That’s why some entries of the training set are randomly discarded to ensure an equal distribution between the categories of labels (`isFromDecay` = 0, 1). An optimization of the hyperparameters (see in 3.3) is also performed and described in 5.1.4.

The output plot (see in 3.6) of the BDT using all 15 features is shown in Fig. 5.1. This plot shows the predictions of the BDT, comparing the training and testing sets, as well as the two categories of labels. The error on the y-axis is Poissonian: $\sigma = \sqrt{N}$ where N is the number of entries in the bin. Apart from the first and last bins, the training and testing distributions are very similar, indicating no clear overtraining.

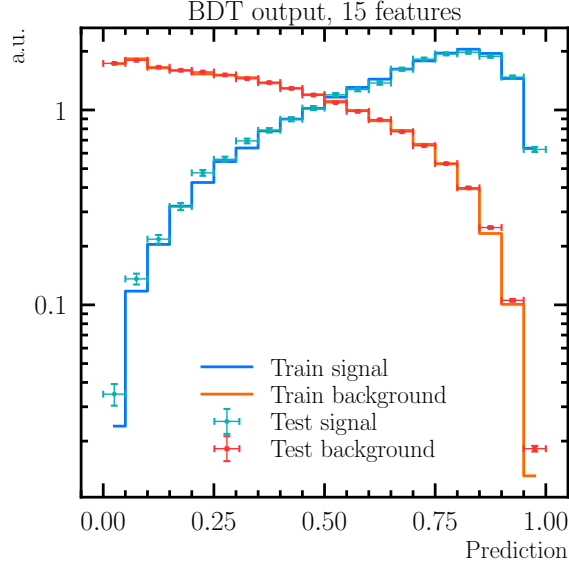


Figure 5.1: Output plot of the BDT using 15 features.

The ROC curve (see in 3.6) performed on the test set is shown in Fig. 5.2. The area under the curve (AUC) is visible and a red horizontal line is drawn for a signal efficiency (or TPR) of 90% and its associated threshold, as well as the corresponding background efficiency (or FPR). This target signal efficiency is used as an example, indicating that with a target of 90.0% signal efficiency, there is a 50.4% background rejection (BKG rej. = $1 - \text{BKG eff.}$).

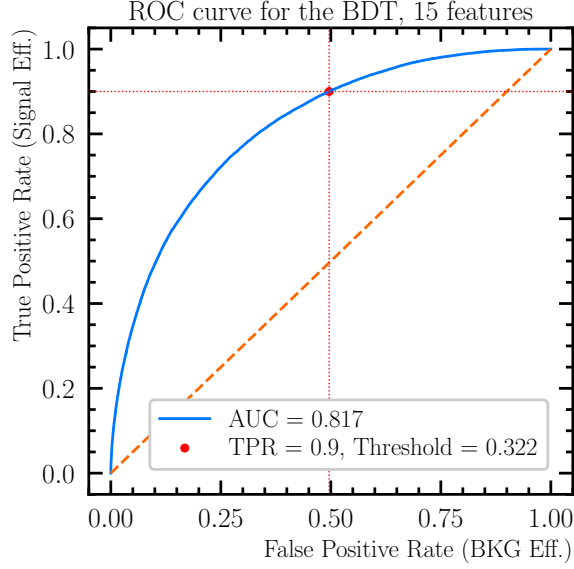


Figure 5.2: ROC curve of the BDT using 15 features.

In Tab. 5.1, the classification report (see in 3.6) of the BDT is shown. The precision and recall do not show extreme variations from this result.

Table 5.1: Classification report for the training set using 15 features.

	Precision	Recall
$label = 0$	0.748	0.727
$label = 1$	0.734	0.755

The classification report is not relevant for the testing set, as this set does not have an equal amount of entries in each category. The performance metrics used are the ROC AUC and the accuracy. These results are shown in Tab. 5.2.

Table 5.2: Results of the ROC AUC and accuracy using 15 features.

Set	ROC AUC	Accuracy
Training	0.822	74.1%
Testing	0.817	72.8%

The features importances allow to rank them. These rankings appears in Tab. 5.3.

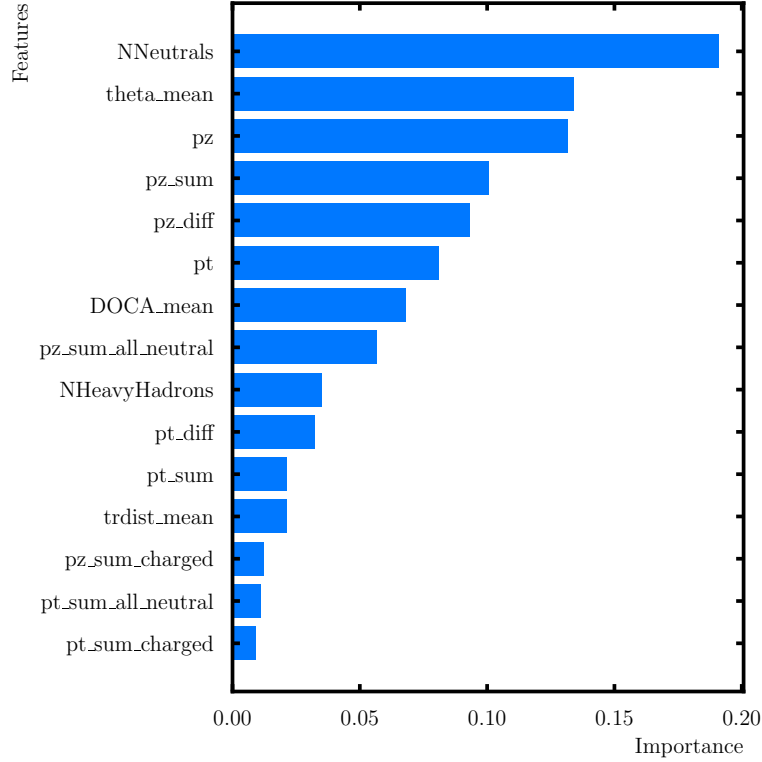


Figure 5.3: The 15 features of the BDT ranked according to their importances.

This importance ordering is useful in order to proceed to feature reduction, see below.

5.1.2 BDT with 9 Features

Using the rankings in Tab. 5.3 and the correlation matrix (Fig. 4.7) discussed in 4.3, the number of features used in the BDT can be reduced. For the correlated features, the lowest-ranked features are suppressed, namely `pt_sum_charged`, `pz_sum_charged`, `pt_sum_all_neutral` and `pz_sum_all_neutral`. Then, as discussed in 4.2, `DOCA_mean` and `trdist_mean` are features

that are poorly reconstructed, hence their suppression. After this selection, a BDT with 9 features is trained, similarly to what is done in 5.1.1.

Fig. 5.4, Tabs. 5.3 and 5.4 show results that are really similar to the BDT using 15 features. This demonstrates that the feature reduction was successful in simplifying the computation of the algorithm without any significant loss in discriminating power.

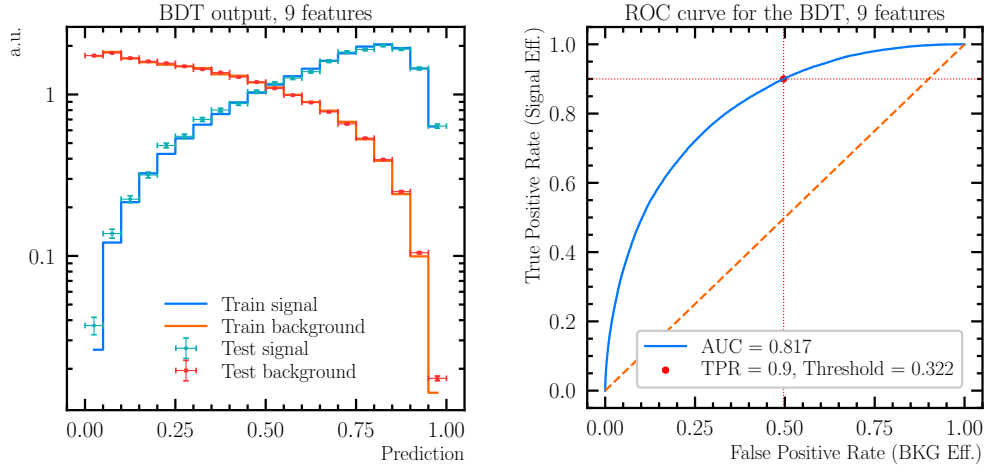


Figure 5.4: Left: Output plot for the BDT with 9 features. Right: ROC curve for the testing set of the BDT using 9 features.

Table 5.3: Classification report for the training set using 9 features.

	Precision	Recall
$label = 0$	0.748	0.725
$label = 1$	0.733	0.755

Table 5.4: Results of the ROC AUC and accuracy using 9 features.

Set	ROC AUC	Accuracy
Training	0.822	74.0%
Testing	0.817	72.7%

5.1.3 Recursive Feature Elimination

Another way to reduce the number of features is by using recursive feature elimination (RFE), explained in 3.5. The BDT is trained as described in 5.1.1, then RFE is applied to reduce the number of features in a range of 4 to 14 features. The results of this reduction are shown in Fig. 5.5.

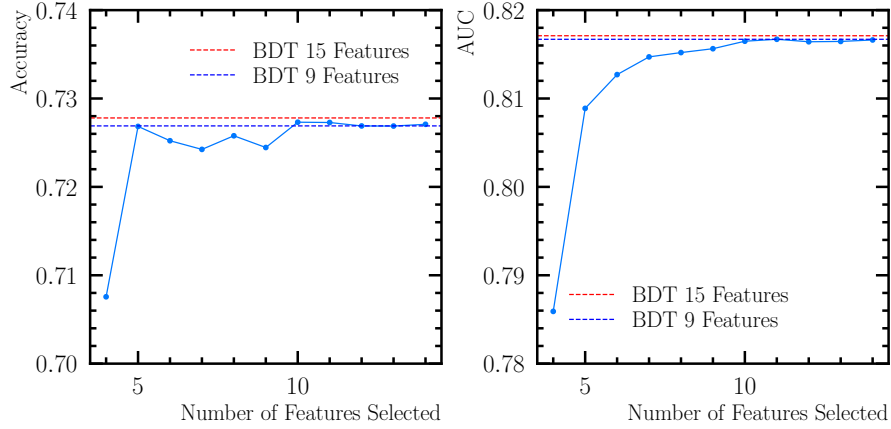


Figure 5.5: Left: Accuracy as a function of the number of features selected. Right: ROC AUC as a function of the number of features selected.

The blue and red dotted lines indicate the performance of the two BDT algorithms described previously. Comparing both algorithms, the only difference is that RFE keeps `DOCA_mean` and discards `pt_sum`. It is observed that RFE with 9 features performs slightly worse than the previous BDT using 9 features selected in 5.1.2. Those results are shown in Tab. 5.5.

Table 5.5: Comparison of the results for the BDT with 9 features using RFE with the BDT manually choosing 9 features, described in 5.1.2.

RFE		Manual	
ROC AUC	Accuracy	ROC AUC	Accuracy
0.816	72.4%	0.816	72.8%

For subsequent studies, the BDT with 9 features selected manually will be used. These features are, as a reminder:

- NNeutrals
- NHeavyHadrons
- pt
- pz
- pt_diff
- pz_diff
- pt_sum
- pz_sum
- theta_mean

5.1.4 Hyperparameters Optimization

In both algorithms discussed in 5.1.1 and 5.1.2, hyperparameters optimization (see in 3.3) is performed. Using the `optuna` library [26], three hyperparameters are optimized:

- `max_depth`: The maximum depth a tree is allowed to go.
- `n_estimators`: The number of trees explored by the boosting algorithm.
- `learning_rate`: An important parameter of the boosting process.

For effective optimization, it is necessary to select an appropriate objective function.

$$\text{Obj} = \text{AUC}_{\text{test}} \quad (5.1)$$

$$\text{Obj} = \text{AUC}_{\text{test}} - |\text{AUC}_{\text{train}} - \text{AUC}_{\text{test}}| \quad (5.2)$$

Initially, a simple metric such as Eq. 5.1 could be tried, but this can lead to overtraining. Another possibility is described in Eq. 5.2, with the right part of this equation aiming to minimize overtraining. This particular objective functions is used for the BDT using 9 features shown in 5.1.2. A

comparison of output plots with both objective functions is shown in Fig. 5.6. Tab. 5.6 shows the results of both algorithms.

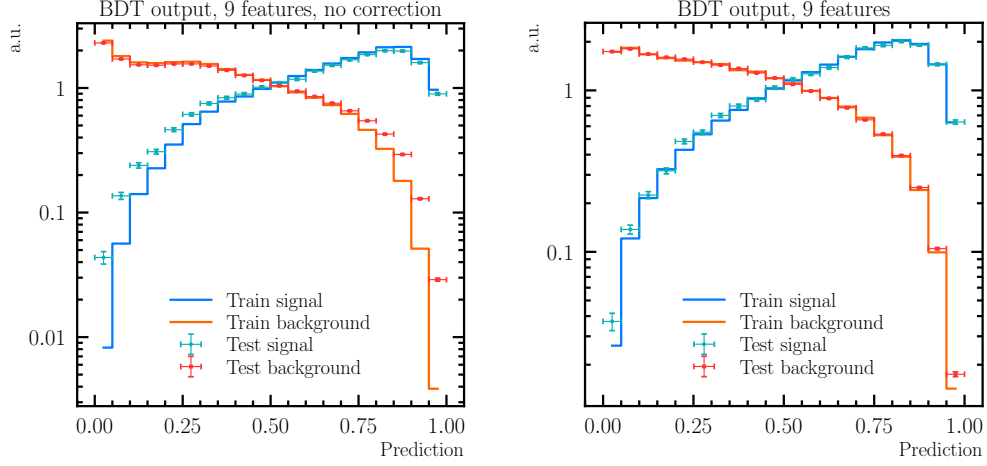


Figure 5.6: Left: Output plot for the BDT with 9 features, using Eq. 5.1 as objective function. Right: ROC curve for the testing set of the BDT with 9 features, using using Eq. 5.2 as objective function.

Table 5.6: Comparison of the BDT results with both objective functions

	Eq. 5.1		Eq. 5.2	
Set	ROC AUC	Accuracy	ROC AUC	Accuracy
Training	0.855	76.7%	0.822	74.0%
Testing	0.818	73.2%	0.817	72.7%

The BDT trained without the correction term has significantly better performance for training than for testing, which is a hint for overtraining. In the output plot in Fig. 5.6, the distributions associated with the two sets significantly differ, the results in Tab. 5.6 also show this behavior. This proves that the objective function aiming to minimize overtraining achieves that, as the difference of performance between training and testing sets is way smaller. Therefore, Eq. 5.2 is selected for the BDT algorithms previously shown.

5.2 Detection of Events that contain Neutral Particles coming from a B Hadron Decay

The output of the BDT algorithm described in 5.1.2 can be used to select events where at least one neutral particle coming from a B hadron decay has been found. This could be useful as a filter before proceeding to the main algorithm to initially discard some events. A new simple dataset with three categories can be created in order to assess the capacity of the BDT algorithm to perform this task:

- **EvtNumber** is the index of the event. Each entry in this dataset represents an individual event.
- **NFromHH** is equal to 1 if there is at least one neutral particles that come from a B decay, else it is 0. It is used as the label. The dataset used for the BDT algorithm contains 7.85% of events labeled 0.
- **pred**: The prediction for the label. It is the mean of all predictions of the BDT algorithm corresponding to the particular **EvtNumber**.

The ROC curve is shown in Fig. 5.7. A clear drop in performance compared to the original BDT is observed. This result is logical since the original BDT aims to assign a candidate neutral to a B decay. Averaging all predictions using the BDT output is certainly not an efficient way to assess the likelihood of an event containing a neutral particle from a B decay. For instance, if an event contains decays that are spatially close, the predictions will be higher than if the decays are not.

A BDT with more event-wise information would be required to enhance the performance of such an algorithm.

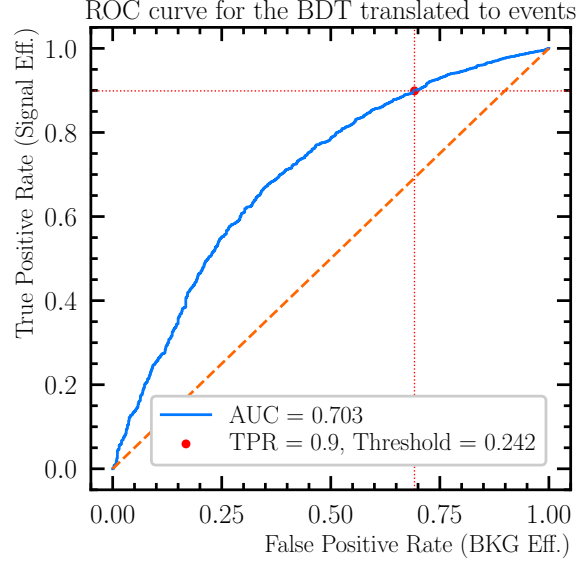


Figure 5.7: ROC curve for the dataset related to events.

5.3 Application to Photons

In the context of Bremsstrahlung recovery (see in 2.6), it is of interest to perform the same BDT as the one described in 5.1.2 for photons only. The only difference being that only photons are considered for both training and testing. The results are shown in Fig. 5.8, and Tab. 5.7

Table 5.7: Results of the ROC AUC and accuracy using the BDT algorithm for photons.

Set	ROC AUC	Accuracy
Training	0.851	77.4%
Testing	0.845	76.0%

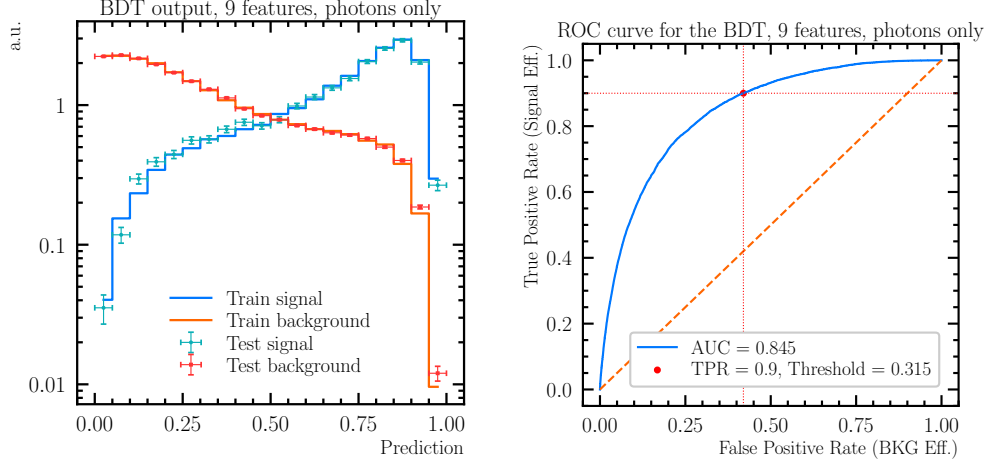


Figure 5.8: Left: Output plot for the BDT with photons. Right: ROC curve for the testing set of the BDT with photons.

These results indicate that the BDT algorithm performs better for photons than for neutral particles in general. Notably, when aiming at a signal efficiency of 90.0%, the corresponding background rejection is of 58.0%. To assess this algorithm's performance, it would need to be compared with the Bremsstrahlung recovery algorithm currently in place at LHCb. Which is complicated since both algorithms work differently (see in 2.6) and the test would need to be performed on the same set of data.

5.4 Translations to Graph Edges

An important aspect is to translate the efficiency of the BDT output to the edges of the graph, as the DFEI algorithm operates based on graphs. This translation is done by assigning the predictions learnt from the BDT to the `Relations_ND` dataset described in 4.2. This requires some processing of the new `BDT_Output` dataset, which is simply the testing set of the BDT dataset with a new `pred` column added, corresponding to the predictions of the classifier.

It is necessary to assign a prediction to each relationship. For neutral-charged (N-C) relations, the assignment of a prediction is straightforward:

$$\text{pred}(\text{Rel}_{n_i, c_j}) = \text{pred}(\text{BDT}_{n_i, d(c_j)}) \quad (5.3)$$

where $\text{pred}(\text{Rel}_{n_i, c_j})$ is the prediction in the `Relations_ND` dataset for a neutral i (n_i) and a charged j (c_j), $1 < i < N$ where N is the total number of neutral particles in the event, $1 < j < n$ where n is the total number of charged particles in a particular decay. Similarly, $\text{pred}(\text{BDT}_{n_i, d(c_j)})$ is the prediction in the `BDT_Output` dataset for a neutral i and the decay that the charged j belongs, this decay is noted as $d(c_j)$. Note that these predictions only include the edges between neutral particles and the charged particles coming from a B decay.

For neutral-neutral (N-N) relations, there is no straightforward way to assign a prediction, since the BDT algorithm assigns neutral particles to the charged part of a particular decay. A smart method of predicting these relations is required. The method starts by selecting the most likely decay d_l for the two neutral particles to be a part of, ($1 < l < M$) with M the total number of decays in a particular event:

$$l = \arg \max_l \left(\frac{1}{2} (\text{pred}(\text{BDT}_{n_i, d_l}) + \text{pred}(\text{BDT}_{n_k, d_l})) \right) \quad (5.4)$$

After selecting the most likely decay d_l , the minimum of the two predictions is chosen:

$$\text{pred}(\text{Rel}_{n_i, n_k}) = \min_{i, k} (\text{pred}(\text{BDT}_{n_i, d_l}), \text{pred}(\text{BDT}_{n_k, d_l})) \quad (5.5)$$

This choice is the most natural. It is illustrated by an example. Assume $\text{pred}(\text{BDT}_{n_i, d_l}) = 0.1$ and $\text{pred}(\text{BDT}_{n_k, d_l}) = 0.9$ which would mean that n_k is very likely to belong to the decay d_l but n_i is very unlikely to. It is logical to take the minimum of those two values because the two neutral particles are most likely not coming from the same decay.

After applying Eq. 5.3 to the N-C relations together with Eq. 5.4 and 5.5 to the N-N relations, the ROC curve for both subsets is computed, as shown in Fig. 5.9.

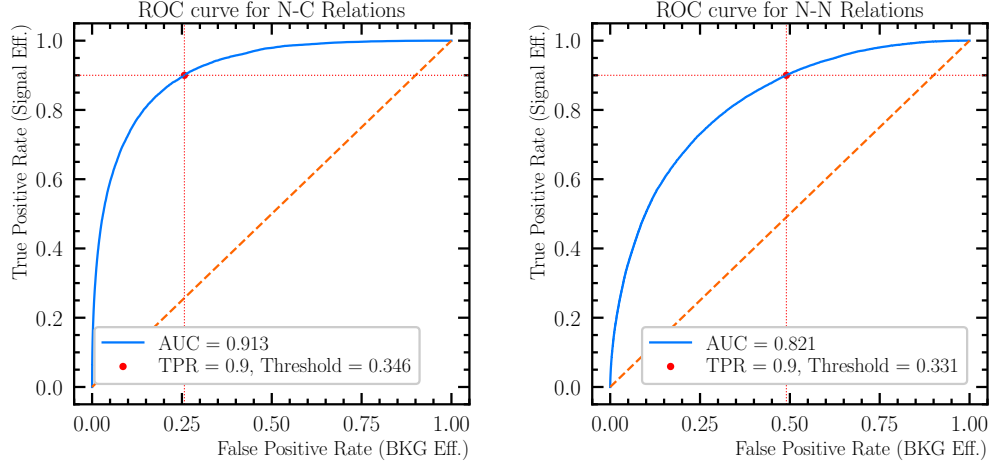


Figure 5.9: Left: ROC curve for N-C relations. Right: ROC curve for N-N relations.

The ROC curve for N-N relations is very similar to the ROC of the BDT output (Fig. 5.4, 5.1.2). The curve for N-C relations shows a better performance for this type of edge. The main possible reason for this is that the BDT algorithm is more effective at assigning neutral particles to the charged part of a B decay if there are many charged particles in it. If that is the fact, it is natural that the translation to edges yields better performance, since an assignment in the BDT relations will translate to as many edges as there are charged particles in the decay. The ROC curve for all relations is shown in Fig. 5.10.

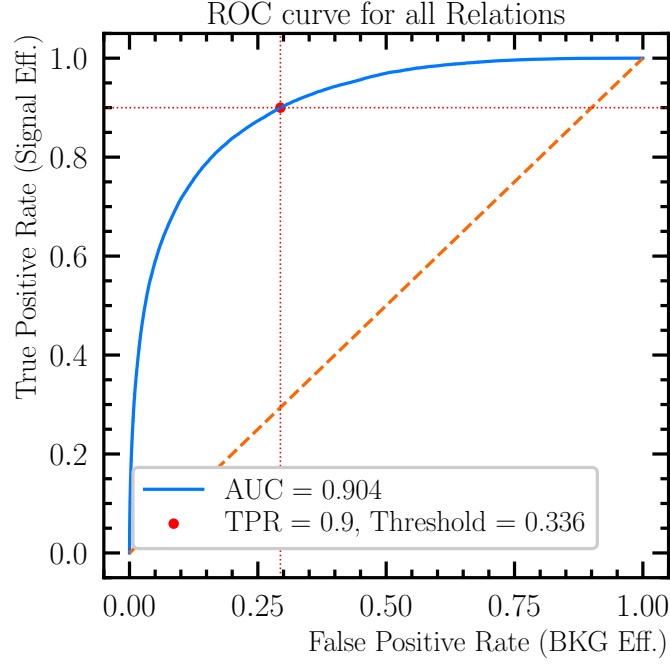


Figure 5.10: ROC curve for all relations.

The overall results are better than the raw output of the BDT algorithm, with 70.6% background rejection at 90.0% signal efficiency. These results are promising, considering it is the first time such an algorithm is being developed. Furthermore, it is done on a simplified simulation with the use of a simple BDT algorithm, meaning that it is likely that further enhancements can further improve the performance.

Chapter 6

Conclusions and Outlook

This thesis advances the exploration of the LHCb experiment’s real-time data interpretation capabilities, by extending the Deep Full Event Interpretation (DFEI) algorithm, currently under development, to include neutral particles. The DFEI algorithm aims at switching from an event-based data filtering model to deciding which parts of the event to save to disk, based on real time interpretation of what particles were produced and how they decayed in the detector. It will do so with automatic learning, trained to select particles in a specific event based on their association with a B decay. This innovative approach addresses the forthcoming challenge of data selection due to the future increase in LHC luminosity. Utilizing graph reduction through neural networks, the DFEI algorithm aims at identifying interesting B decays. The proposed extension incorporates Boosted Decision Trees (BDT) within the existing DFEI framework as a second layer, enhancing its range to effectively include neutral particles, which had been so far completely disregarded, despite their presence in a wide range of decays of interest for physics research. This integration is therefore crucial for improving the precision and comprehensiveness of particle decay analysis, ultimately contributing to a deeper understanding of fundamental particle physics.

The initial part of this thesis presents an overview of the context of the work and the theoretical knowledge about BDT algorithms. Subsequently, the structure of the datasets is analyzed. This part is of crucial importance, as the algorithm proposed is a BDT, which is not compatible with graphs. The transformation of the initial graph-structured datasets into a single, well-suited dataset for a BDT is discussed. The underlying concept is that each entry represents a neutral particle, and the charged particles assigned to a

specific B decay. Consequently, the algorithm is trained to assign the neutral particles to a particular B decay.

Then, the BDT is trained. Initially, 15 features are employed, in the aim to evaluate the devised features. Two distinct approaches to feature reduction are employed to assess the relative merits of each. The first one discards features based on correlation and reconstruction quality, the second one is using recursive feature elimination, a method based primarily on the separation power provided by the feature which is calculated through training. The former approach was selected, providing a BDT with 9 features, to be further commissioned on simulated data. The question of hyperparameters optimization is addressed. Two objective functions are devised, comparing them to chose the one not leading to overtraining. The final algorithm chosen has a 50.4% background rejection for a target 90.0% signal efficiency.

A modification of the BDT output is done, aimed at being able to discriminate whether or not a particular event contains a neutral particle originating from a B decay, which could be useful as a prefilter. The resulting algorithm is found to perform worse than the main algorithm, which is expected since the features used in the main algorithm mainly describe the neutral particle and its relations to the charged part of B decays.

An interesting result about the reconstruction of photons, which could enhance the Bremsstrahlung recovery algorithm currently used at LHCb, is discovered. The performance of the algorithm only trained and tested on photons performs better, with a 58.0% background rejection for a target 90.0% signal efficiency. A more precise comparison between the proposed approach and that currently being used in LHCb is required, to confirm that this method has potential, and to lead to its prospective inclusion in the LHCb data analysis routines.

Finally, the output of the main BDT algorithm is translated back to the edges of the graph to assess its quality the same way as the DFEI algorithm, based on graphs. This translation results in better performance than the raw BDT output, with 74.6% background rejection for a target 90.0%. These figures represent the best assessment of the overall performance of the BDT classifier developed in this thesis.

These algorithms were supposed to be applied to a dataset coming from the official full LHCb simulation software, as well. Due to technical difficulties on the simulation side, that require additional locally-developed features to be merged into the official software, such a dataset could not be produced in time for this thesis. The link to the corresponding merge request can be

found in Appendix B. This dataset could give improvements to the algorithm for several reasons. These simulations would include more sources of neutral particles, such as particles resulting from material interactions, but would also include more variables, that could increase the separation power. For instance, the quality of reconstruction of the neutral particles.

Knowing that the DFEI algorithm achieves on average a signal efficiency of 94% and a background rejection of 96%, it is clear that the algorithms proposed here are performing slightly worse. However this is to be expected, due to the lack of detector information available for neutral particles, and the simplicity of the exploratory algorithm we implemented. Despite that, the very encouraging results obtained in this thesis work constitute an important ground work from where to build in order to achieve the goal of extending the DFEI algorithm to neutral particles.

Improvements could be made by using more advanced machine learning models. A model using the DFEI output for training must be implemented, to assess the performance of both algorithms working together. This work is currently ongoing. Finally, integration in the real LHCb experiment should be done in order to fully fathom this algorithm's potential.

Acknowledgements

I would like to express my gratitude to Prof. Lesya Shchutska and Prof. Olivier Schneider for allowing me to conduct my thesis research in their laboratory. It has been a genuine privilege to work in such an environment, surrounded by knowledgeable and dedicated professionals. I would also like to extend my appreciation to Elena Graverini, who proposed this project and provided invaluable guidance throughout, in collaboration with the insights and suggestions of Julian Garcia Pardinás. Finally, I would like to express my gratitude to my friends and family for their unwavering support, both in terms of practical assistance and moral encouragement.

Bibliography

- [1] Description of the standard model by CERN. <https://home.cern/science/physics/standard-model>.
- [2] Cush. https://commons.wikimedia.org/wiki/File:Standard_Model_of_Elementary_Particles.svg, 2019.
- [3] Lyndon Evans and Philip Bryant. LHC machine. *Journal of Instrumentation*, 3(08):S08001, 2008.
- [4] Arpad Horvath. <https://commons.wikimedia.org/wiki/File:LHC.svg>, 2006.
- [5] LHCb Collaboration. The LHCb Detector at the LHC. *JINST*, 3:S08005, 2008. Also published by CERN Geneva in 2010.
- [6] LHCb collaboration. The LHCb upgrade I. Technical report, 2023.
- [7] Vladimir Gligorov. Real-time data analysis at the LHC: present and future. 2014.
- [8] C. Fitzpatrick and V. Gligorov. Anatomy of an upgrade event in the upgrade era, and implications for the LHCb trigger. Technical report, CERN, Geneva, 2014.
- [9] García Pardiñas J. et al. GNN for Deep Full Event Interpretation and Hierarchical Reconstruction of Heavy-Hadron Decays in Proton-Proton Collisions. Technical report, 2023.
- [10] García Pardiñas J. et al. A deep-learning based Full-Event Interpretation (DFEI) algorithm. Slides presented in Bologna, 2022.

- [11] Syed Naeem Ahmed. *Physics and Engineering of Radiation Detection*. 2015.
- [12] Martino Borsato. Electron reconstruction at LHCb and Belle II. Presentation slides, https://indico.in2p3.fr/event/18845/contributions/73189/attachments/54347/71096/MBorsato_eReco_LHCbBelle2.pdf.
- [13] Yann Coadou. *Boosted Decision Trees*. WORLD SCIENTIFIC, 2022.
- [14] Breiman L. et al. *Classification and Regression Trees*. 1984.
- [15] Yann Coadou. Boosted decision trees. Slides presented in Archamps, 2016.
- [16] Lenka Zdeborová. Machine learning for physicists, lecture notes, 2023.
- [17] Jason Koskinen. Multivariate method - boosted decision tree. Lecture notes, university of Copenhagen, 2021.
- [18] Mark A Hall. Correlation-based feature selection for machine learning. 1999.
- [19] Charles Zaiontz. Basic concepts of correlation. <https://real-statistics.com/correlation/basic-concepts-correlation/>.
- [20] Xgboost developers. Introduction to boosted trees. <https://xgboost.readthedocs.io/en/latest/tutorials/model.html>.
- [21] Sklearn Documentation : https://scikit-learn.org/stable/modules/generated/sklearn.feature_selection.RFE.html.
- [22] Andrea Valassi. Roc curves, auc's and alternatives in hep event selection and in other domains. Presentation slides, https://indico.cern.ch/event/679765/contributions/2814562/attachments/1590383/2516547/20180126-ROC-AV-IML_v008_final.pdf.
- [23] Martin Thoma. https://commons.wikimedia.org/wiki/File:Roc_curve.svg.

- [24] De Cian M. et al. Fast neural-net based fake track rejection in the LHCb reconstruction. Technical report, CERN, Geneva, 2017.
- [25] Tianqi Chen and Carlos Guestrin. Xgboost: A scalable tree boosting system. In *Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, KDD '16. ACM, 2016.
- [26] Takuya Akiba et al. Optuna: A next-generation hyperparameter optimization framework, 2019.
- [27] <https://zenodo.org/records/7799170>.
- [28] Merge request for the complete LHCb simulation dataset. https://gitlab.cern.ch/lhcb/Rec/-/merge_requests/3527.

Appendix A

Additional Figures

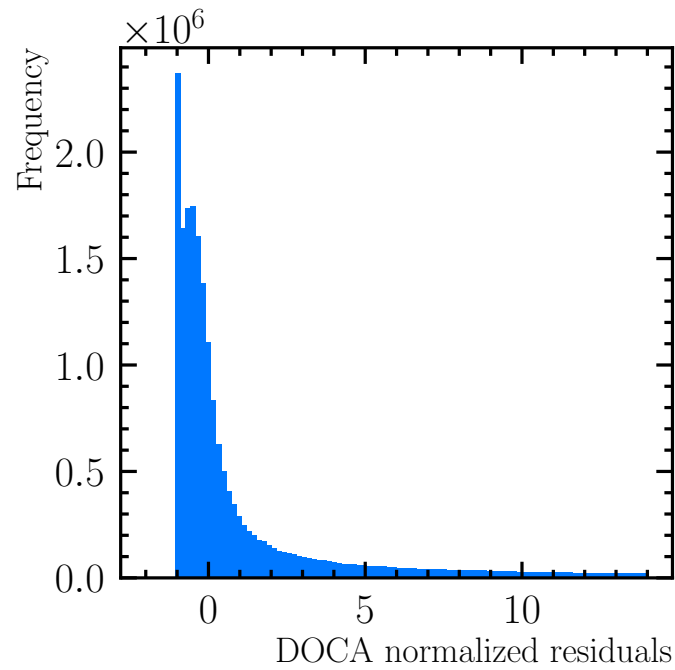


Figure A.1: Histogram of the normalised residuals for distance of closest approach (DOCA).

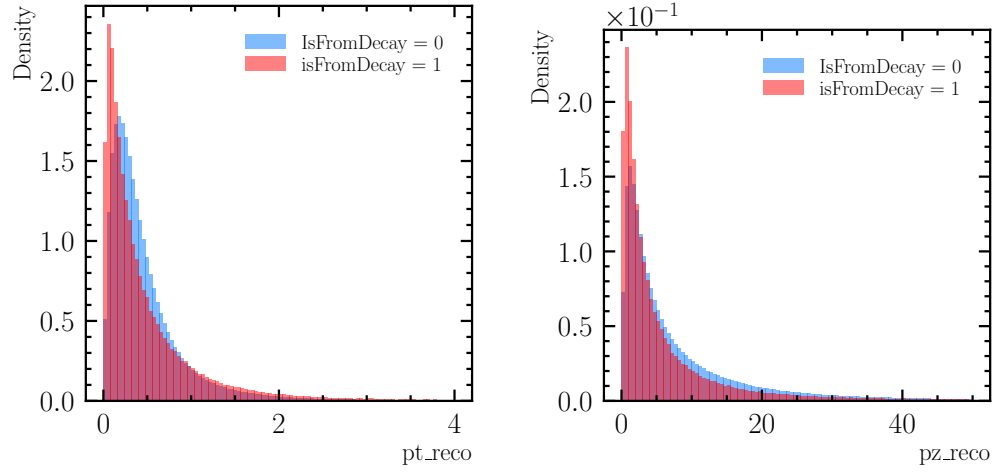


Figure A.2: Left: Histogram of separated distributions for pt . Right: Histogram of separated distributions for pz .

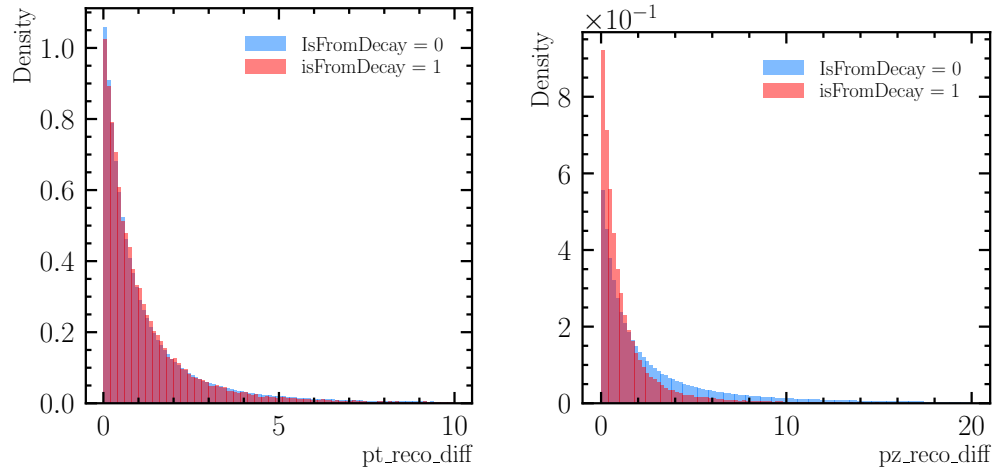


Figure A.3: Left: Histogram of separated distributions for pt_diff . Right: Histogram of separated distributions for pz_diff .

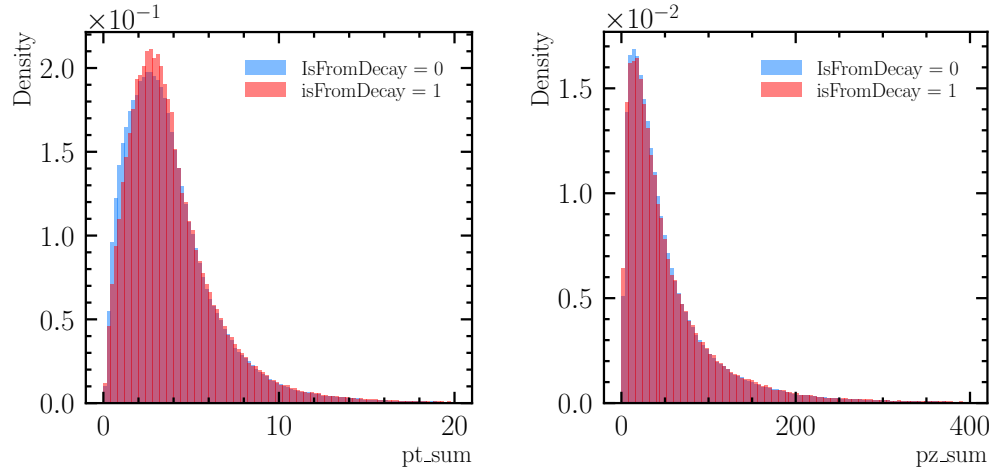


Figure A.4: Left: Histogram of separated distributions for pt_sum . Right: Histogram of separated distributions for pz_sum .

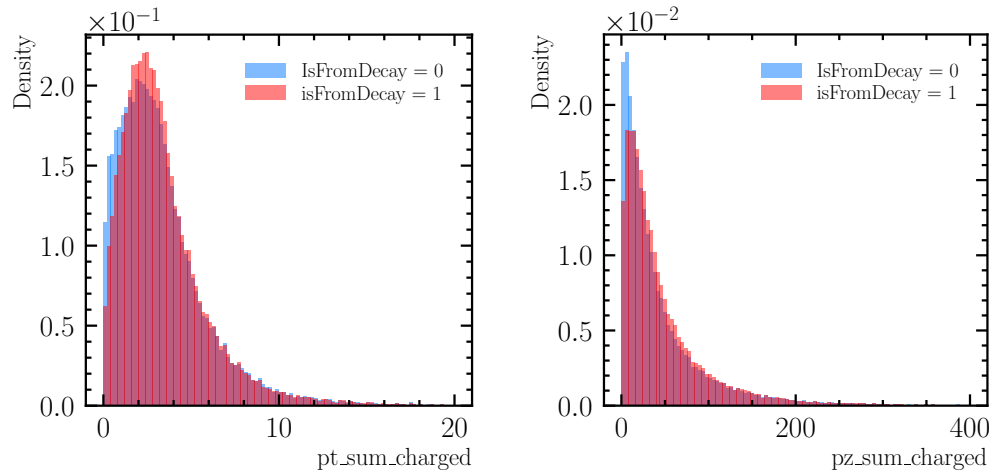


Figure A.5: Left: Histogram of separated distributions for $pt_sum_charged$. Right: Histogram of separated distributions for $pz_sum_charged$.

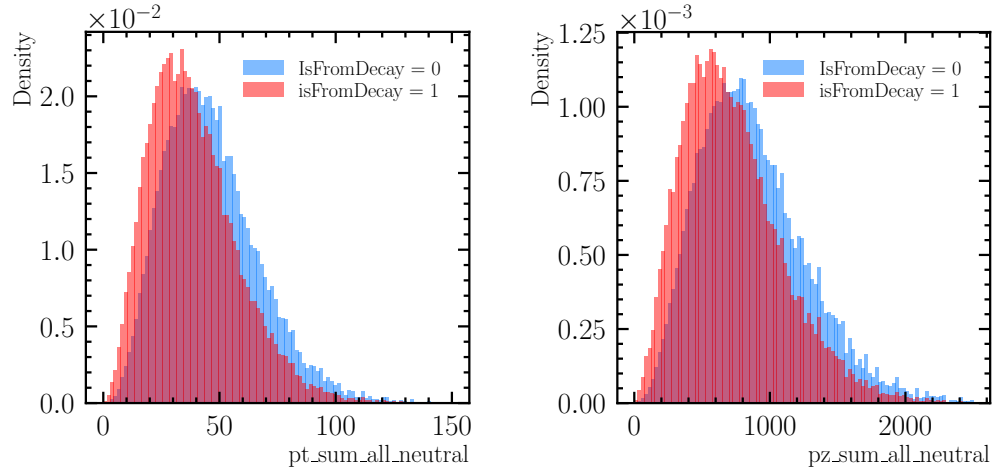


Figure A.6: Left: Histogram of separated distributions for `pt_sum_all_neutral`. Right: Histogram of separated distributions for `pz_sum_all_neutral`.

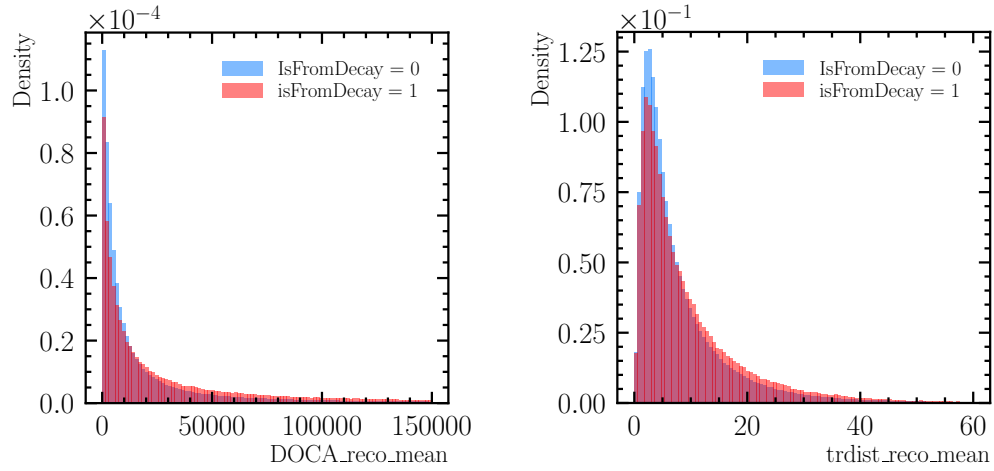


Figure A.7: Left: Histogram of separated distributions for `DOCA_mean`. Right: Histogram of separated distributions for `trdist_mean`.

Appendix B

Links to the Datasets

The datasets described in 4.1, are found on the lxplus server at CERN, they are similar to the dataset `Dataset_InclusiveHb_Evaluation.root` used for the DFEI algorithm [27]:

- `/eos/lhcb/user/j/jugarcia/PYTHIA_tuples/Event_data_particles_-bquark_nu7_6_49399events_allbkgtracks_events_0_to_49398.root`
- `/eos/lhcb/user/j/jugarcia/PYTHIA_tuples/Event_data_relations_-bquark_nu7_6_49399events_allbkgtracks_events_0_to_49398.root`

The major difference is that the dataset used in this work contain neutral particles.

The merge request for the complete LHCb simulation is found here [28].