AGH University of Krakow

Faculty of Physics and Applied Computer Science

Department of Physics

Real-Time Machine Learning-Enhanced Reconstruction of Long-Lived Tracks in LHCb High-Level Trigger and Calibration Analysis of the UT Detector

A thesis submitted for the degree of

Doctor of Philosophy

in Physics

Author:
Sabin Hashmi

Supervisor:
dr hab. inż. Tomasz Szumlak

September, 25 2025

Contents

List of Figures

List of Tables

For Pappa and Imma, who inspired this journey.

Declaration

I declare that this thesis is my own work, carried out under supervision. This thesis is submitted in partial fulfilment of the requirements for the degree of Doctor of Philosophy in Physics.

- Author

Contributions

Over the course of my five-year research journey at Krakow and Geneva, I contributed to the LHCb experiment through two main projects. Despite their differences, these projects focus on improving track reconstruction in the experiment.

The first significant project I embarked on was enhancing downstream track reconstruction in the LHCb experiment. I joined the project during the long shutdown phase. This was a period where I had the opportunity to learn and prepare the most for the vastness of the research. The project's core focused on a machine learning approach to address inefficiencies in the existing reconstruction sequence. The state-of-the-art algorithms designed for track reconstructions have the limitation of sometimes generating combinatorial tracks that do not correspond to actual particle trajectories and cannot distinguish them. As part of the effect, two different classifiers were built, one for improving SciFi track classifiers and the other for downstream tracking more efficiently. In the second case, this would be for selectively identifying and filtering out the ghost tracks from the downstream tracks. Run 2 data laid the groundwork for processing Run 3 data, which posed additional challenges due to the upgraded detector and the heightened precision required for measurements. Throughout this phase, I gained experience in understanding complex algorithms and adapting solutions. My contributions culminated in two major merge requests, integrating and testing the new models into the reconstruction pipeline.

The second part of my doctoral project focused on developing a monitoring method for automatic analysis of the UT calibration with the application of computational intelligence techniques. I used physical data collected by the LHCb experiment as well as the special runs of the UT detector without the circulating beams. This work resulted in two complementary tools, one for real-time analysis and one for forecasting. This thesis documents the outcomes of these projects and explores potential extensions for future research. Addition to these projects, I had the privilege of participating in data-taking shifts at LHCb, assisting final-year projects for master's students.

Publication

- Second Generation Machine Learning Based Algorithm for Long-Lived Particles Reconstruction in Upgraded LHCb Experiment

Summer Schools and Posters

- CERN School of Computing, Tartu, Estonia - 2023
- LHCC Poster Sessions, Geneva, Switzerland - 2023
- INFN - SESAME International School on Efficient Scientific Computing, Amman, Jordan - 2023
- XXVIII Cracow Epiphany Conference, Krakow, Poland - 2022

Acknowledgments

Writing this acknowledgment gives me the opportunity to reflect on the journey that has brought me to this point and to express my gratitude to everyone who helped me make this possible. Growing up far from the place where I am writing this thesis, it was always a deep curiosity in Science and a love of learning new things that laid the path ahead. The dream of pursuing research was not an easy one, and even to dream it seemed ambitious at times.

First and foremost, I would like to express my deepest gratitude to my supervisor, Tomasz Szumlak, for his invaluable guidance, unwavering support, and insightful advice throughout my PhD journey. This thesis would not have been possible without his mentorship. Tomasz helped me become an independent researcher and supported my personal growth. He encouraged me to pursue the paths we believed could succeed. At every stage of this challenging journey, he provided guidance without hesitation. I am sincerely grateful for the freedom and encouragement he gave me to explore and develop my ideas. I cannot thank him enough for making this research possible.

This section would be incomplete without acknowledging my family. To my parents, thank you for believing in me and for instilling in me the values of curiosity, perseverance, and integrity. Your trust, encouragement, and sacrifices shaped every step of my journey. You always treated me as an individual, capable of making my own decisions, and you valued education above all else. I owe everything to you. This work is a testament to your support and unconditional love.

I would also like to thank my friends, who have become like family here in Krakow. You people bring out a different side in me. For all the laughter and constant support, you've added a hue to my life and helped me stay grounded during the most challenging moments.

This thesis represents the culmination of many years of dedication, hard work, and overcoming challenges. It reflects the guidance, support, and inspiration I have received from all those mentioned above. Thank you to everyone who contributed to this journey in any way—through advice, encouragement, or simply being there. Every struggle, every difficult decision, and every choice I made was shaped by the lessons and support I received from all of you.

Finally, Pappa and Imma, I dedicate this chapter, this achievement, and all that I have accomplished to you. Thank you for teaching, guiding, and inspiring me to strive for improvement every day.

Streszczenie

Niniejsza rozprawa przedstawia kompleksowy opis pięciu lat badań prowadzonych w ramach eksperymentu LHCb (ang. Large Hadron Collider beauty experiment) w CERN. Praca koncentruje się na opracowaniu i zastosowaniu technik uczenia maszynowego w celu wzmocnieniu algorytmu do rekonstrukcji śladów córek, rozpadów cząstek długożyciowych (tak zwane ślady downstream) i poprawy wydajności systemu wyzwalania eksperymentu LHCb oraz analizę danych kalibracyjnych detektora UT (ang. Upstream Tracker). Pierwsza część pracy podzielona jest na dwa etapy - zwiększenie czystości próbek śladów SciFi (ang. Scintillating Fibre detector) oraz polepszenia jakości śladów typu downstream.

W rozprawie analizowane są strategie wykorzystania tych nowych linii trygerowych w celu maksymalizacji wydajności rekonstrukcji bez narażania danych fizycznych. Druga część pracy koncentruje się na detektorze UT, umieszczonym przed magnesem dipolowym, który ma kluczowe znaczenie dla precyzyjnych pomiarów pędu oraz wydajności trygera jako całości. Prace dotyczące wzmocnienia algorytmu do rekonstrukcji śladów typu downstream opierały się na opracowaniu dwóch oddzielnych modeli uczenia maszynowego. Pierwszy model ma na celu poprawę czystości próbki zawierającej segmenty śladów SciFi, które stanowią podstawę do bardziej złożonej rekonstrukcji śladów typu downstream. Drugi selektor ma na celu identyfikację śladów downstream i eliminację śladów kombinatorycznych (ang. ghost tracks), powszechnie nazywanych „śladami duchami", które nie odpowiadają żadnym rzeczywistym trajektoriom. Ta faza stanowi jeden z najbardziej wymagających technicznie aspektów badań, obejmujący rygorystyczną ocenę metryczną i ocenę wydajności modeli, a następnie ich integrację i wdrożenie w ramach oprogramowania Gaudi oraz algorytmów rekonstrukcji śladów LHCb.

Drugi projekt ma na celu opracowanie metodologii monitorowania do analizy i kalibracji detektora UT, co pomoże zoptymalizować stosunek sygnału do szumu w celu lepszej rekonstrukcji hitów.

Słowa kluczowe: Ślady downstream, Uczenie maszynowe, Wyzwalacz wysokiego poziomu, Sieci neuronowe, Rekonstrukcja śladów

Abstract

This thesis presents a comprehensive account of five years of research conducted within the LHCb experiment at CERN. The work focuses on developing and applying machine learning techniques to enhance track reconstruction and detector performance. The work is organised into two phases, each addressing distinct but related challenges in particle tracking and data analysis.

The thesis explores strategies to leverage these software triggers to maximise reconstruction efficiency without compromising physics data. The second phase focuses on the Upstream Tracker (UT), a detector positioned upstream of the dipole magnet critical for precise momentum measurements.

The first phase focuses on improving downstream track reconstruction by developing two machine learning models. The first selector aims to enhance the quality of SciFi track segments, which serve as the seeds for more complex downstream track reconstruction. The second selector is designed to identify the downstream tracks and eliminate combinatorial tracks, commonly called "ghost tracks," which do not correspond to real particle trajectories. This phase represents one of the most technically demanding aspects of the research, involving rigorous metric evaluation and performance assessment of the models, followed by their integration and deployment within the Gaudi software framework and the LHCb track reconstruction algorithms. The second project aims to build a monitoring methodology for analysing and recalibrating the UT detector, helping to optimise the signal-to-noise ratio for improved track reconstruction.

Keywords: Downstream Tracks, Machine Learning, High-Level Trigger, Neural Networks, Track Reconstruction

Phase I: Foundation

LHCb Experiment and Downstream Tracking Reconstruction

Chapter 1

Introduction

1 Physics Motivation

High energy physics (HEP), also known as particle physics, is the field of science dedicated to the exploration and study of the fundamental building blocks of matter and the forces that govern them. The current theoretical framework strongly depends on energy scales. On the macroscopic level, the dynamics of spacetime, gravity, and everything associated with it explain the underlying mechanics of the Universe; the General Theory of Relativity with remarkable precision. It is one of the most widely accepted theories for explaining large-scale systems. However, this classical framework cannot explain the quantum realm, and we need Quantum Mechanics to explain that.

"It all starts with one simple question: How did it all begin?

One of the open-ended questions in science and the most puzzling mystery for humankind is how it all started: the cosmic origins of celestial objects and the Universe and everything within.

Big Bang Theory, one of the most widely accepted theories, describes that all began from a quantum fluctuation, which was likely amplified during the phase called cosmic inflation. This initial seed evolved over approximately 13.8 billion years to grow [1] to the current state of the Universe that we observe (it is worth noticing that the recent discoveries of the James Webb Space Telescope may strain the classical cosmology). What existed before the Big Bang remains an open-ended question yet to be understood. In popular opinion, it may have been one of several significant events in the early Universe. It may not be the only event, and the concept of time itself may have lacked meaning before it.

One of the intriguing puzzles in the aspects of the current Universe is its composition. The visible Universe consists mainly of matter components, yet the corresponding antimatter is almost absent. According to the current theories, matter and antimatter should have been produced in equal amounts in the early Universe. However, the antimatter part is conspicuously missing. If a perfect symmetry had existed, they should have annihilated each other, leaving only the energy; no atoms, no universe, no life. However, a slight excess of matter remained over antimatter, about 1 in a billion matter-antimatter pairs, for reasons not yet fully understood. This tiny imbalance is responsible for the matter-dominated Universe in which we live. One proposed explanation for the matter-antimatter asymmetry is based on violating Charge-Parity (CP) symmetry. Although the theoretical frameworks for violating the CP are well established, experimental validation is still ongoing [2]. It is worth noticing that the LHCb (Large Hadron Collider beauty experiment) was especially designed to explore this puzzle in the heavy quark sector.

Particle accelerators are one of the key frontiers in the experimental validation of these theories. Colliding particles at extremely high speeds allow us to recreate energy scales similar to those of the early Universe. Experiments such as those conducted at the Large Hadron Collider (LHC) help to make precision measurements and uncover and study phenomena beyond the Standard Model of Particle Physics.

2 The Standard Model and the Limitations

The *Standard Model*(*SM*) *of Particle Physics* is a well-established theoretical framework that describes the fundamental building blocks of matter and the forces that govern their interactions, excluding gravity. Although it can explain all presently known elementary particles, it is not a complete theory of nature. It fails to explain several key phenomena, including the existence of dark matter and dark energy, the accelerated expansion of the Universe, and it cannot incorporate gravity. SM is built on the foundation of quantum field theory (QFT), and they have been rigorously tested using high-precision experimental instruments like the LHC [3]. Elementary particles, the fundamental building blocks of all known matter, are the constituents of the Standard Model. As shown in Figure [1], they are broadly classified into two categories:

- **Fermions:** Fermions are the matter particles - they build up all the matter we observe. There are 12 fundamental fermions, further split into two subcategories: quarks and leptons.

Each fermion carries $\frac{1}{2}$ spin and is subject to the Pauli Exclusion Principle, which prohibits identical fermions from simultaneously occupying the same quantum state.

Quarks: Quarks are the strongly interacting particles, and they have six flavours that we know so far. Up, Down, Charm, Strange, Top, and Bottom Quarks. They possess a quantum entity color charge that leads them to have a strong nuclear force mediated by gluon force carriers.

Leptons: Leptons come in three generations, the electron, muon, and tau, and their neutral counterpart, the neutrino. They do not experience strong interactions and participate in electromagnetic and weak interactions.

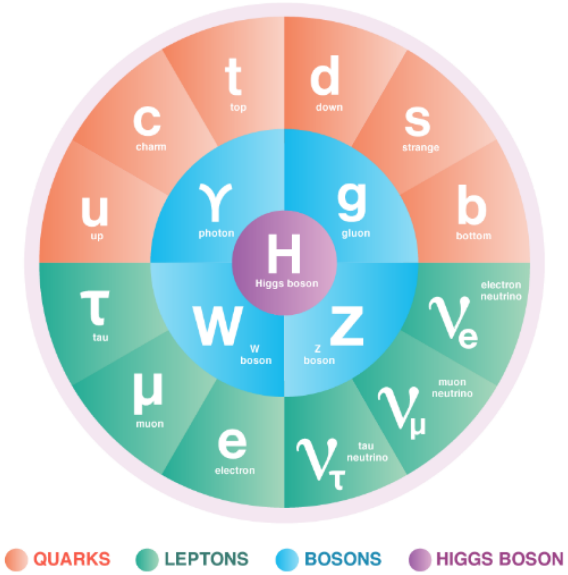


Figure 1: Standard Model of Particles [4]

- **Bosons:** Bosons are force carriers in the standard model — they mediate interactions between fermions. They carry integer spin values and do not obey the Pauli Exclusion Principle, allowing multiple bosons to occupy the same state.

  Gauge Bosons: Gauge bosons mediate the fundamental forces described by the Standard Model. The photon is the electromagnetic force carrier, the gluon mediates the strong force between quarks, and the $W^{\pm}$ and Z bosons are responsible for the weak nuclear force.

  Higgs boson: The Higgs boson, commonly known as God's particle, is a scalar particle with spin 0. It is associated with the Higgs field. The Higgs field is a quantum field that gives mass to the W and Z bosons and fermions through spontaneous symmetry breaking. The Higgs boson was discovered at the LHC in 2012, confirming a key prediction of the Standard Model [5].

3 CERN and the LHC Experiment.

CERN, the European Organisation for Nuclear Research (originally *Conseil Européen pour la Recherche Nucléaire*), is one of the world's largest and leading scientific institutes in particle research. The scientific facility is located near Geneva, Switzerland. CERN was established in 1954 by twelve Western EU countries to conduct scientific experiments with global collaboration, hosting a community of researchers worldwide. Its mission includes exploring fundamental questions in physics, probing beyond the Standard Model, and advancing our understanding of particle physics. CERN houses the Large Hadron Collider (LHC), the world's largest and most powerful particle accelerator. The LHC is a circular proton-proton collider with a circumference of 26.7 kilometres, situated approximately 100 meters underground, and spanning the border between France and Switzerland.
As the name suggests, the Large Hadron Collider (LHC) accelerates and collides hadrons, specifically protons, at near-light speeds. At LHC, hadrons are used for several reasons, including that protons are stable and easy to accelerate using electromagnetic fields. Colliding protons allow us to access the internal structure, quarks, and gluons (partons). Most importantly, the high energies involved in these collisions can recreate conditions similar to those just after the Big Bang, allowing the creation and study of rare or heavy particles or potential new physics beyond the Standard Model.

The LHC hosts four major experiments: ATLAS (a toroidal LHC apparatus), CMS (compact muon solenoid), ALICE (a large ion collider experiment), and LHCb (LHC-beauty). The LHC is designed to accelerate protons and heavy ions in two counterrotating beams that travel in opposite directions within separate beam pipes. Each proton is accelerated to 6.5 TeV in proton-proton collisions, resulting in a total centre-of-mass energy of 13TeV [7].

The final energy is achieved through multiple stages of a complex accelerator chain. The protons travel through several stages before being injected into the LHC. The beams collide at four designated interaction points, supplying data to all four experiments and other experiments. The LHC operates with a bunch-crossing rate of 40MHz, meaning that proton bunches collide up to 40 million times per second, although only a small fraction of these events are recorded after the trigger systems [8].

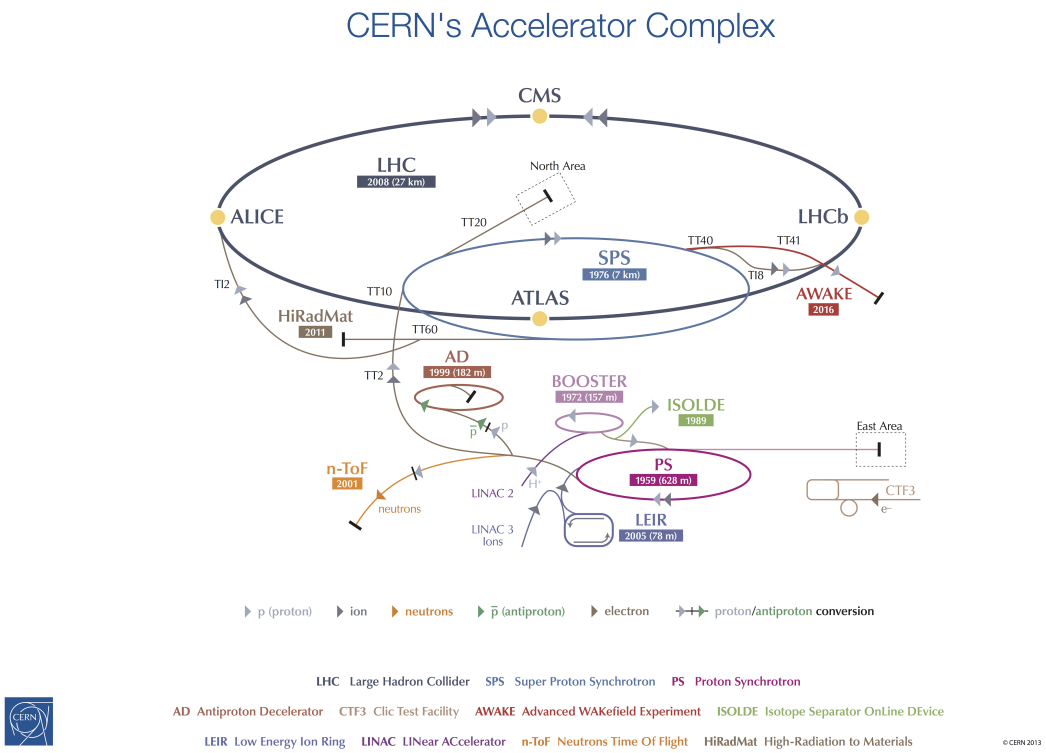


Figure 2: LHC-Complex [6]

Figure [2] shows the LHC complex. The Step-by-step acceleration of protons in the LHC begins with a linear accelerator. Historically, Linac2 accelerated protons to 50MeV, but now it has been replaced by Linac4, which accelerates negative hydrogen ions (H^-) to 160MeV. These ions are then stripped of their electrons to produce protons before injection into the Proton Synchrotron Booster (PSB). The protons are subsequently accelerated through the Proton Synchrotron (PS) and the Super Proton Synchrotron (SPS), reaching an energy of 450GeV before the final injection into the LHC. At LHC, they are accelerated to a maximum energy of 6.5TeV per beam. Filling both beams takes approximately 4 minutes and 20 seconds, followed by an additional 20 minutes to ramp the beams to their maximum energy levels.

4 WLCG - Computing Grid and Dirac.

Due to the high collision rates and precision requirements, the LHC produces a large volume of data. The experiments rely on a multi-level data acquisition and processing system to handle this. CERN has a robust and well-structured framework for managing computational resources, from data acquisition and storage to a global distributed processing system of experimental data.

An important key component in this computational infrastructure is the *World-wide LHC Computing Grid* (*WLCG*). WLCG is a distributed computing system designed to handle and store massive volumes of data generated by the LHC experiments as shown in Figure [3].

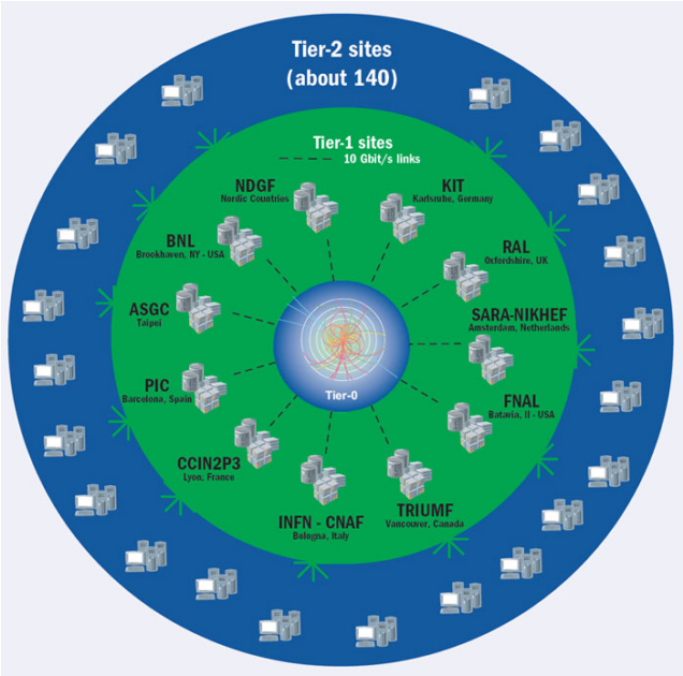


Figure 3: WLCG Network [9]

WLCG comprises thousands of computational nodes distributed across the globe, providing distinct services within the Grid. Globally, it is built and structured in a four-level tier system.

Tier 0 Located at CERN and performs initial data processing, handling about 25% of the total computational load.

Tier 1 Includes 14 large data centres, and is the primary backup support for Tier 0 [10]

Tier 2 Consists of 150 universities and research facilities, focusing mainly on simulation tasks and user-level data analysis.

Tier 3 Consists of many small, locally maintained computing resources.

A powerful middleware framework called *DIRAC*(*Distributed Infrastructure with Remote Agent Control*) orchestrates the efficient coordination of these distributed infrastructures [11]. Dirac plays a central role in job scheduling, resource allocation, and management across the Grid. Scalable, high-throughput distributed systems are critical for an experiment like LHC. WLCG and Dirac together make it feasible.

5 Upgrade I

Following the Long Shutdown 2(LS2), the LHCb experiment entered its Run 3 phase. It is a milestone, marking the beginning of Upgrade 1, in enhancing data acquisition capabilities and precision measurements. The experiment was resumed with the proton-proton collision at a centre-of-mass energy of $\sqrt{s} = 14$ TeV. One of the significant upgrades happened in the detector read-out systems, transitioning to a more robust software trigger system that replaced the old hardware trigger systems. In Run 3, the LHCb experiment will operate at an increased instantaneous luminosity of 2×10^{33} cm^{-2}s^{-1}. This is a significant improvement over Run 1 and Run 2. This change can result in a collection of approximately integrated luminosity. 50 fb^{-1} during Run 3. Compared to the combined integrated luminosity of Run 1 and Run 2, it is 8 fb^{-1}.

5.1 Instantaneous Luminosity

Instantaneous luminosity (L) is one of the improvements from Upgrade 1. It is the measure that quantifies interactions per unit of area per unit of time. It quantifies how often particles collide in the detector.
The instantaneous luminosity L is defined as:

$$L = \frac{N \cdot f}{A} \tag{1}$$

And for practical analysis, *Integrated Luminosity* (L_{int}) is preferred as a relevant parameter. It is the total number of expected signal events for a given process. The event rate is given by:

$$\frac{dN}{dt} = L \cdot \sigma \tag{2}$$

Where:

- N is the number of particles per bunch crossing,
- f is the bunch crossing frequency,
- A is the effective transverse area of the beam overlap.
- σ is the cross-section for the process of interest.

This increased L leads to a higher average of visible interactions per bunch crossing, called pileup. This directly increases the average event size by a factor of 3x. This scale-up requires a more robust trigger system. LHC is designed to handle a maximum instantaneous luminosity of 10^{34} cm^{-2}s^{-1}, using 2808 bunches per

beam. Each bunch contains on the order of 1.1×10^{11} protons [12]. In the case of LHCb, it operates at a significantly reduced event luminosity. This design choice ensures optimal vertex separation, reduced event multiplicity, and a better signal-to-background ratio. This, in turn, improves the precision and the reconstruction fidelity of time-dependent and rare decay measurements.

6 LHC-beauty Experiment

The LHCb experiment is one of the flagship experiments at the LHC. The LHCb detector is a forward spectrometer with distinct geometric coverage as showed in the Figure [4]. The LHCb experiment is designed to make precise measurements and study particles produced within a slight angle relative to the beam line. As the name implies, the experiment's primary focus was on the detailed exploration of beauty quarks, one of the six in the Standard Model.

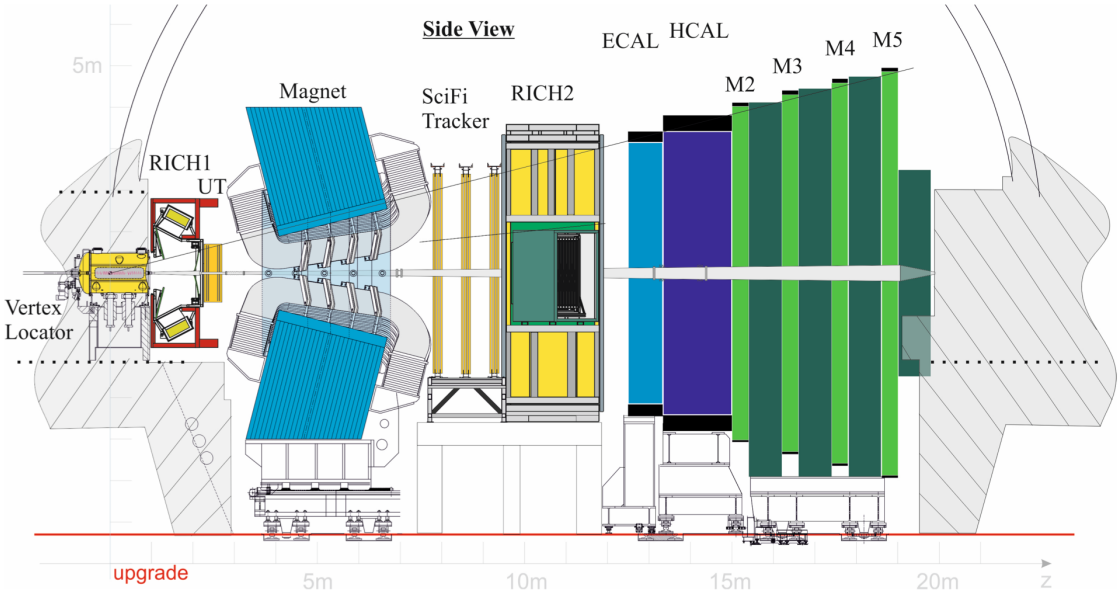


Figure 4: LHCb Experiment [Side View] [13]

The central idea of LHCb is to investigate the asymmetry between matter and antimatter in the Universe generated within the heavy quark sector. In the case of CP violation, a combined symmetry of charge conjugation (C) transforms particles into their corresponding antiparticle component and parity (P), which inverts the spatial coordinates. The violation of this context is a vital element in this experiment in deepening our understanding of the fundamental forces of nature.

LHCb is designed with a unique structure with:

- **Pseudo-rapidity (η) and Coverage:**

$$\eta = -\ln\left(\tan\frac{\theta}{2}\right), \qquad 2 < \eta < 5 \tag{3}$$

  The forward region is optimal for detecting heavy-flavor hadrons.

- **Angular Acceptance:**

$$10 \text{ mrad} < \theta < 300 \text{ mrad} \quad \text{(bending plane)}$$
$$10 \text{ mrad} < \theta < 250 \text{ mrad} \quad \text{(non-bending plane)}$$

  Corresponds to the detector's acceptance at a polar angle.

- **Geometrical Coverage:**

$$\text{Solid angle coverage} \approx 4\% \text{ of full solid angle}$$

  Optimized for heavy-flavor physics.

- **Heavy Quark Detection:**

$$\text{Detection of } \approx 40\% \text{ of all } b\bar{b} \text{ and } c\bar{c} \text{ pairs}$$

- **Hadron Detection Efficiency:**

$$\text{Detects } > 25\% \text{ of all hadrons produced in } pp \text{ collisions}$$

- **Interaction Point:**

$$\text{Single-arm forward spectrometer design}$$

7 LHCb Tracking System

Charged particles' trajectories are reconstructed using very precise momentum measurements while traversing through the detector. This is achieved using a combination of tracking detectors and magnetic deflections.

The LHCb tracking system consists of three main subdetectors: the Vertex Locator (VELO), the Upstream Tracker (UT), and the Scintillating Fibre Tracker (SciFi). A dipole magnet, located between the VELO and UT, introduces curvature into the trajectories of charged particles, helping to estimate their momentum using the Lorentz force [14].

7.1 Vertex Locator (VELO)

The Vertex Locator (VELO) is a hybrid pixel silicon detector. The upgraded VELO detector is redesigned to operate for Run 3 at a much larger instantaneous luminosity, about 5 times larger than compared to Run 1 and Run 2 conditions, and with a bunch crossing of 30 MHz. Pixel sensors help mitigate and significantly reduce channel occupancy and improve spatial resolution.

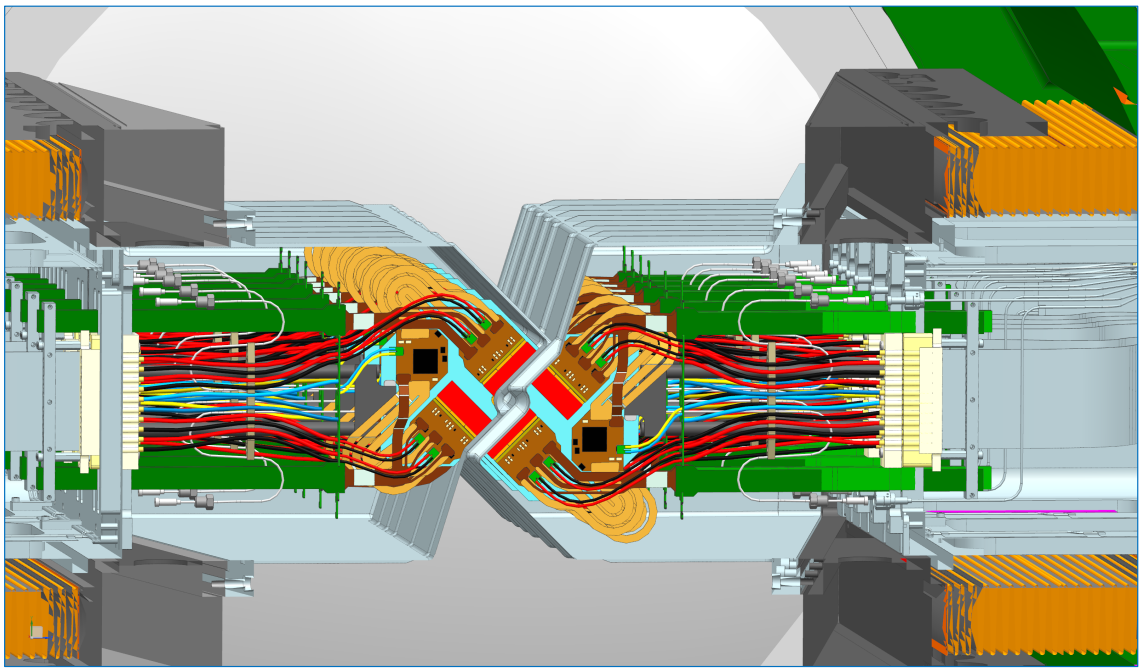


Figure 5: Vertex Locator

As displayed in Figure [5], VELO has a retractable geometry and lightweight construction and is positioned close to the interaction point surrounding the region. It comprises 4.1×10^7 pixels, each with dimensions of 55 μm $\times$ 55 μm [15]. To ensure thermal stability and minimum noise, the detectors are cooled using evaporating CO_2 circulations through microchannel cooling systems embedded within the sensor modules.

The primary function of the VELO is the reconstruction of the vertices. This requires precise identification of primary interactions as well as displaced secondary vertices originating from the decay of particles. Thus, one of the key aspects of VELO is the ability to measure *impact parameter* (IP)—the transverse distance between a reconstructed track and the associated vertex with high precision. Retractable halves can reach as close as 5.1 mm from the beam line, which is critical for accurate IP measurements. Considering the limited geometrical acceptance, VELO is a crucial component of LHCb for reconstructing tracks that would otherwise fall outside the detector's acceptance region. VELO is housed in its vacuum enclosure, separate from the primary beam vacuum to minimise scatter, multiple scattering, and interactions with the residual gases. An aluminium RF foil further protects it. These protections can further mitigate the interactions with the air molecules and preserve the quality of the reconstructed tracks with improved vertex resolution.

7.2 Upstream Tracker (UT)

The Upstream Tracker (**UT**) is a four-plane silicon microstrip detector. It is strategically positioned immediately downstream of the VELO and upstream of the LHCb dipole magnets. This is an important configuration for enhanced track reconstruction precision before the charged particles experience magnetic deflection. UT plays a central role in improving the accuracy of momentum estimation. One of the key aspects of UT is the distinguishing of true particle trajectories from combinatorial artefacts that do not originate from any corresponding real particles - called *Ghost Tracks*. UT uses silicon microstrip technology that helps achieve high spatial resolution, enables reliable identification of charged particle tracks, and minimises false positives in the experiment. This is particularly important for maintaining the integrity of the reconstructed tracks in high-occupancy environments.
The UT replaces its predecessor, Tracker Turicensis (TT), as part of Upgrade I. These transitions have several motivations.

- **Radiation Tolerance:** High radiation levels are expected during Run 3. Because they are positioned within the beam pipe region and are subject to high radiation doses. However, the TT was not designed to operate reliably under high radiation

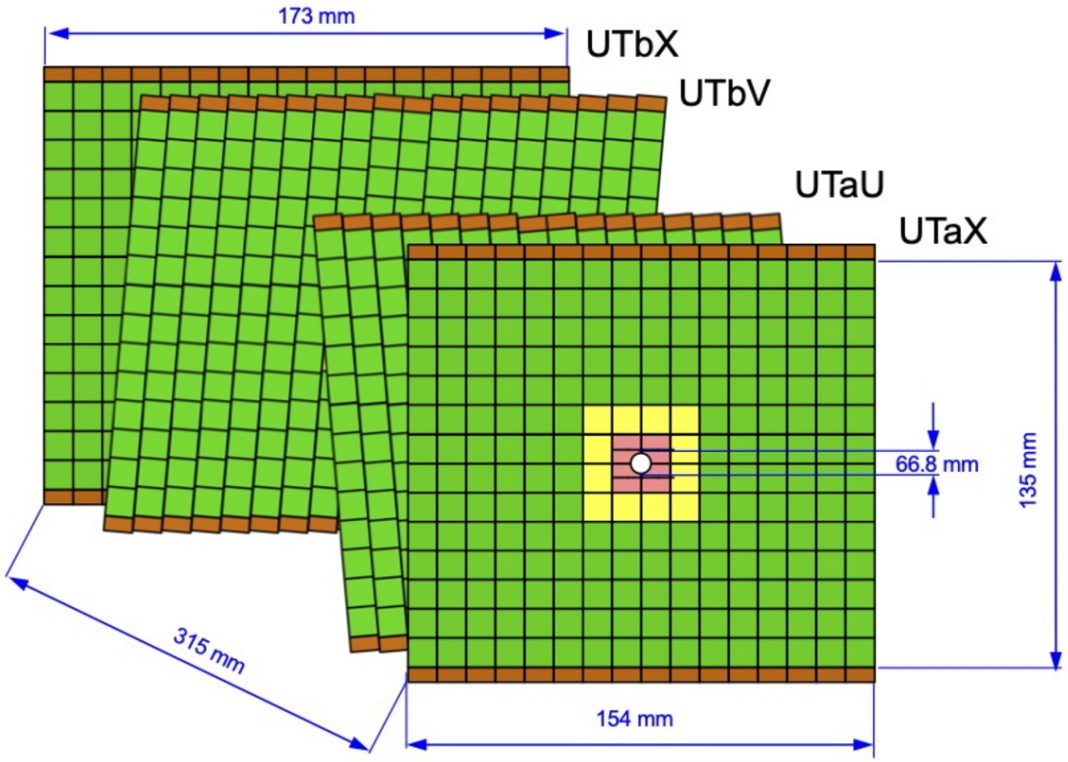


Figure 6: Upstream Tracker [16]

conditions. UT uses improved radiation-hardened sensors.

- **Improved Granularity and Precision:** The UT has higher granularity at the chip level than the TT, leading to improved spatial resolution and more precise track measurements.
- **Read-out Capability:** The Beetle chip, which formed the read-out backbone of the TT, cannot support the required 40 MHz full detector read-out that was required by Upgrade I, (the new DAQ architecture also supports triggerless VELO readout) [17]. The UT uses SALT ASIC, which is specifically designed to handle high data rates and perform real-time signal processing on the detector side.

As the Figure [6] illustrates the UT structure, it maintains a detector geometry similar to that of TT. It consists of four layers: X-U-V-X. The inner layers (U and V) are tilted at a stereo angle of $\pm 5^{\circ}$ with respect to the vertical axis (Y) to improve the resolution of the impact in the transverse plane. Sensors close to the beamline are radiation hardened to have a high radiation tolerance to withstand the harsh environment.

7.3 Scintillating Fibre Tracker (SciFi)

The Scintillating Fiber Tracker (**SciFi**) is the main downstream tracking detector. It is located after the dipole magnet. SciFi tracker plays an integral role in track reconstruction strategies. It is designed to help reconstruct the trajectory of the charged particles by detecting the light produced when they traverse the scintillating materials.

As the Figure [7]illustrates, The SciFi Tracker consists of three tracking stations, each consisting of four detection planes arranged in X-U-V-X geometry. Similarly to the case of UT, the inner stereo planes are tilted with respect to the vertical axis for improved resolution and high-precision 3D track reconstruction. SciFi Tracker replaces its predecessor. Like UT, SciFi is radiation hardened to enhance its tolerance and sustain itself in high occupancy and harsh radiation environments. SciFi can provide almost complete angular acceptance of the detector region and achieve a spatial resolution of 80 μm [18].

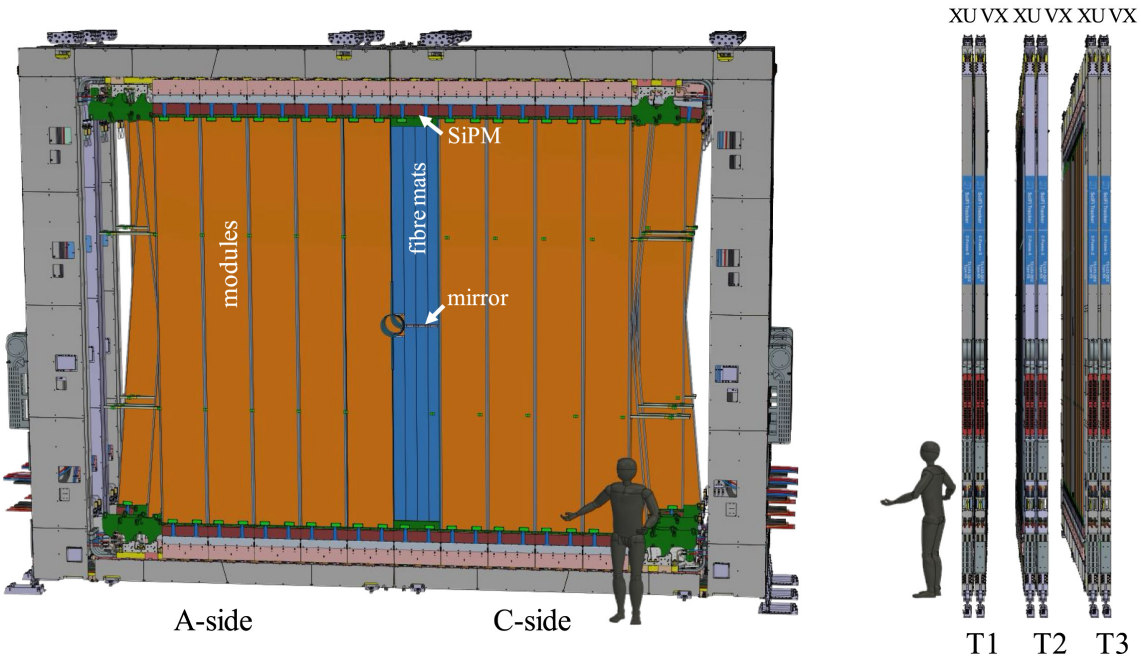


Figure 7: SciFiTracker

Each detector plane is built from scintillating fibres with a diameter of 250 μm and a length of 2.5 m. Scintillating fibres are thin, flexible strands emitting light when charged particles traverse them. At the end of the fibre ribbons, the light is then guided to the *Silicon Photomultipliers* (*SiPMs*) placed on top or bottom of the detector, providing precise spatial and energy deposition information of the charged particle.

Each SiPM contains 128 individual pixels capable of detecting single photons. They collectively help to capture the track trajectory, inferring the patterns and number of photons detected. As a fun fact, the name SciFi originated from its fictional-like concept and difficulty in building, and the feasibility of construction was doubted at the time.

8 LHCb Data Infrastructure

The LHCb data processing infrastructure is built on the x86 microprocessor hardware architecture with support of the GP-GPU chips. The Figure [8] shows the dataflow in LHCb experiment. The digitised data from the detectors' data acquisition components is passed to High-Level Trigger I (HLT I), which is processed in real time (synchronous to the beam). HLT I is operated within the Event Filter Farm (EFF). The partially reconstructed data are then transferred to a buffer. This buffer is used for the Real-Time Calibration and alignment of the detector hardware. Once the calibration and alignments are completed, the events are processed by High Level Trigger II for full event reconstruction.

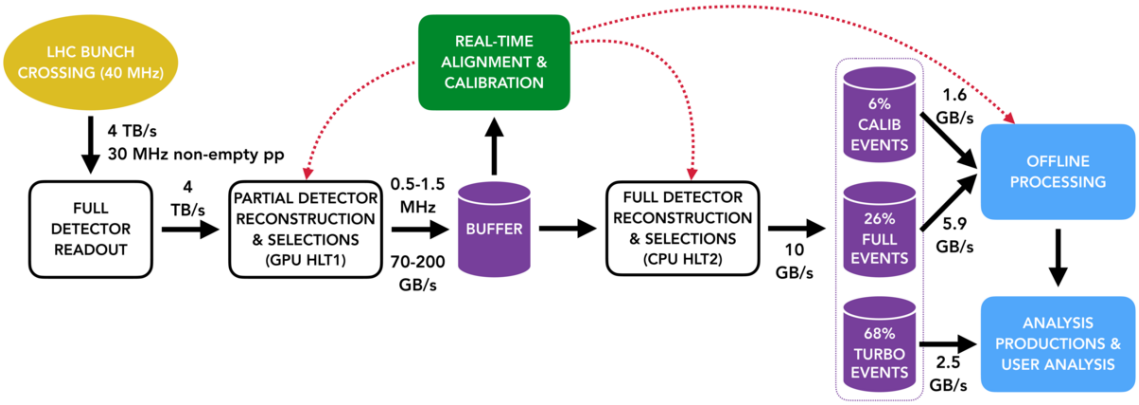


Figure 8: LHCb Data-flow [19]

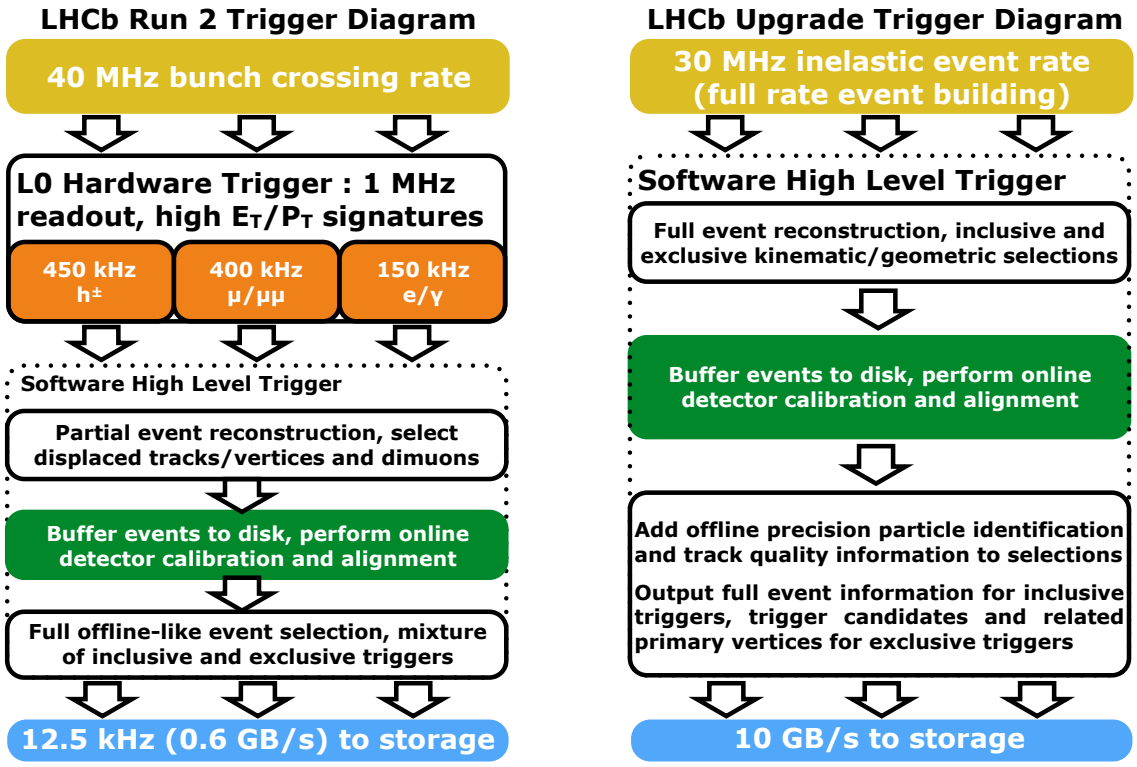
This real-time processing approach essentially eliminates the need for traditional offline reconstruction and RAW data storage (some percentage of full RAW data is stored for reference and calibration purposes). This is a great challenge and a significant shift in computational loads, making it possible to move the offline processing power toward simulation tasks.

9 High-Level Trigger (HLT)

In Run 1 and Run 2, LHCb operated with a hybrid trigger system comprising a hardware-based trigger (L0) and a software-based HLT [20]. The data flow was considered in two main streams.

Persistent Stream: The full event information that is processed offline.

Turbo Stream: The real-time processed data derived from trigger levels with reduced size (only essential objects that triggered the event are retained).



(a) Run 2 : Trigger

(b) Run 3 : Trigger

Figure 9: Trigger Configurations in Run2 and Run3 [21]

In 2017, around 50% of the data was already being processed via **Turbo Stream**, and from Run 3, the experiment has fully transitioned to the online Turbo Stream model [22]. Figure [9 shows the comparison between the Run 2 trigger and Run 3 trigger systems. This real-time processing model output contains all the necessary physics information for later analysis, and only a minor fraction of the RAW data is stored for specialised studies.

It was not an easy, but a necessary transition, which enabled a significant overhaul of the trigger systems. The L0 trigger in the earlier version operated at 1.1 MHz with a limited latency of 4 μs. This was a significant bottleneck, considering the LHC operated at a 40-MHz bunch crossing rate. From being operational in Run 3, the L0 trigger has been entirely replaced by a *fully software-based trigger architecture*. This new, modern, robust trigger system can handle an effective input rate of 30 MHz [23]. The upgraded trigger systems consist of two components:

Low-Level Trigger (LLT) This is an optional throttling mechanism that existed for Run 2. It is a simplified FPGA-based filter designed to pre-select events based on selective conditions, like high-transverse-energy clusters in calorimeters or high-transverse-momentum tracks in muon systems. In fact, this system can reduce the load on the main software trigger by a factor of 2, with minimal loss of physics data. This acts as a backup mechanism that can be used to preserve efficiency in hadronic channels.

High Level Trigger (HLT) This is the major shift in the key components of the trigger systems. HLT follows a hierarchical architecture.

- HLT 1: Performs real-time partial event reconstruction, including track finding and initial physics selections, with a strict time budget and synchronous with data taking.
- HLT 2: After real-time calibration and alignments, execute a full event reconstruction. It helps with the final output used for physics analysis and works asynchronously.

High-Level Trigger 1 (HLT1)	**High-Level Trigger 2 (HLT2)**
Vertex identification using impact parameter (IP)	Uses exclusive algorithms for specific decay reconstruction
Track association and transverse momentum reconstruction using forward tracking	Full reconstruction of decays within detector acceptance
No particle identification	Inclusive trigger selections with PID
Performs online calibration and alignment	Implements Turbo stream returning fully reconstructed data

Table 1: Duties of HLT1 and HLT2 in LHCb

HLT 1 has strict timing constraints that allows for only partial track reconstruction [24]. This process is inspired by Run 2, making HLT a strong focus on real-time calibration and alignments. Alignments are performed using the data collected during each fill, and the calibration is continuously evaluated on a per-run basis, making it possible to perform a complete reconstruction of the event at HLT 2.

The new trigger system is supported by a modular read-out architecture comprised of the following components: Event Builder(EB), Timing and Fast Control System (TFC), Experiment Control System (ECS), and Event Filter Farm (EFF). These structures deliver seamless, flexible, and scalable solutions for real-time event processing. After the software trigger, the event rate reduced from 30-40 MHz to about 10 GB/s of data written to the storage. Compared to Run 2, the final throughput was just 0.6 GB/s due to hardware limitations. This new system provides significantly higher efficiency with greater flexibility in data handling.

10 Thesis Strategy and Contributions

In this research, a two-stage machine learning pipeline is deployed within the HLT2 trigger system. The High-Level Trigger (HLT) provides a stable and robust track selection and reconstruction framework. With particular emphasis on downstream tracks originating from the decays of long-lived particles. The improved selection of long-lived tracks has an entirely new meaning for the upgraded detector, since it may potentially help in identifying processes beyond the Standard Model. The primary objective of this work is to improve Downstream Track reconstruction in the LHCb Experiment. It is addressed in two parts: selecting true seed tracks for reconstruction and correctly identifying true downstream tracks. These form the core contributions of the work presented here and will be discussed in the following chapters. As an extension of detector calibration efforts, the project also explores and implements recalibration techniques and monitoring methods, which constitute the final phase of this thesis.

Chapter 2

Tracking of Charged Particles in LHCb

Track Reconstruction of a charged particles is a very complex experimental task. It involves interpreting the local (low-level observables) signals generated within the active material of the sub-detectors as the particle traverses them. These signals help to derive the particle trajectory it leaves behind. These tracks are then used to reconstruct the high-level observables, such as the impact parameter, the momentum, and other key variables, which are essential for physics selections performed on the trigger level.

1 Track Topology in LHCb

As charged particles cross the detector, they produce a variety of track segments. These segments are broadly classified based on the sub-detectors where the particles leave signals. Figure [10] shows classifications in track types. The classification reflects the origin, energy, and history of the interaction between the detectors, etc.

VELO Tracks: These tracks leave signals exclusively in the VELO detector. VELO tracks do not provide momentum information since they are located upstream of the magnetic field region.

T-Tracks: Also known as SciFi Tracks, these are formed solely from hits in the SciFi tracker. Considering the geometry of the SciFi tracker, these tracks serve as seed candidates for reconstructing more complex track types.

Upstream Tracks: These tracks produce signals in both VELO and UT, but not in SciFi Tracker. Typically made from low-momentum particles that do not reach the downstream tracking stations.

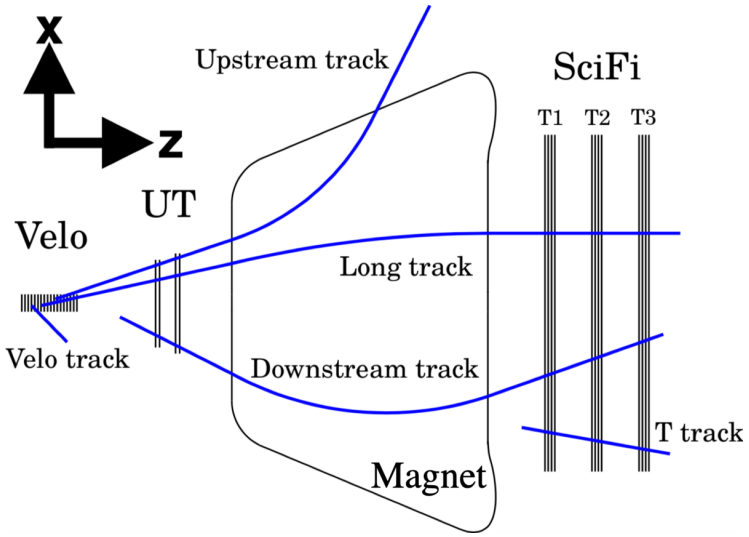


Figure 10: Track Types : LHCb [25]

Long Tracks: These tracks leave hits in all three tracking stations, providing the best momentum resolutions.

Downstream Tracks: These tracks originate from the decay of long-lived particles and leave hits in UT and SciFi Tracker, but not in VELO.

2 Downstream Tracking and its Challenges

Long-lived Particles(*LLPs*) apart from the standard ones (K_s^0 and Λ^0) can also represent new particles beyond the SM. Unlike most known particles, LLPs have relatively long lifetimes. They can travel a measurable distance before decaying, often producing displaced vertices. They do not leave any signals at VELO, but at UT and SciFi Tracker. Making the long-lived displaced vertices an experimental signature that can hint at new physics phenomena.

To study LLPs beyond the SM, we can refer to known particles within the SM. Neutral kaons like K_S^0 and Lambda baryons like Λ^0 are particularly useful due to their well-understood behaviour for benchmark reconstruction validations. Having no signals at VELO poses additional challenges in reconstructing the origin vertex with high precision, on top of the other existing challenges in track reconstructions. It makes the study of LLP decay more complex. In the context of systematic errors and uncertainties, the track reconstruction precision measurements are mitigated by multiple techniques, which include alternating polarities to help reduce biases.

Track reconstruction of the charged particles consists of three main steps:

Pattern Recognition This is the initial step of track reconstruction, where the goal is to associate individual detector hits (local, low-level observables) into a segment originating from a single particle. Because of the large number of particles produced in each collision and, consequently, the huge number of hits that must be associated with each reconstructed particle, the pattern recognition is a highly non-trivial task. One must deal with hits that are created by the detector noise and real ones that may overlap with hits originating from a different particle. This leads to exponential growth of complexity in accurately associating hits. Pattern recognition involves identifying and rejecting unwanted combinatorial tracks that are noise and do not correspond to an accurate trajectory by retaining the hits likely to originate from a particle.

Track Fitting Once we have the sets of hits associated with each reconstructable particle, the next step is to refine this by fitting the track parameters. For this, we widely use algorithms like the Kalman Filter, one of the well-suited algorithms for handling the challenges posed by multiple scattering and energy loss caused when particles interact with the detector material. This involves detailed calculations of the particles propagating through the detector's magnetic field and material layers, making it one of the most computationally intense parts of the process.

Clone Killing Once the track fitting is completed, this step helps remove ambiguous signals and overlapping tracks that can be constructed. The clone killing step helps identify and remove redundant tracks by retaining good-quality track trajectories for further studies.

Having no signals for daughter tracks of the long-lived particles at VELO poses additional challenges in reconstructing the decay vertex with high precision, on top of the other existing challenges in track reconstructions. This makes the study of LLP decay more complex. In the context of systematic errors and uncertainties, the track reconstruction precision measurements are mitigated by multiple techniques, including alternating polarities to help reduce various detection asymmetries.

3 Software Infrastructure

At the LHC, the hardware infrastructure is deeply integrated with a layered software stack to ensure the efficient and seamless operation of the experiments. This integration is particularly vital in the case of the LHCb experiment due to the real-time data processing requirements and high data throughput. The LHCb software stack enables an end-to-end workflow from data acquisition using full detector information in real-time to offline physics analysis. The Figure [11] shows the main projects within the LHCb software stacks:

- **Gaudi:** The core software framework that orchestrates the execution of algorithms in both the online (real-time trigger) and offline (analysis and reconstruction) environments.
- **Ganga:** A front-end for job submission and monitoring, facilitating task management across the computing Grid [26].
- **Simulation Stack (Pythia8/Geant4):** Used for generating Monte Carlo data, including event generation and detector simulation.
- **Moore:** The framework used to implement and test trigger algorithms, specifically for HLT2. Activate the real-time event selection during data collection.
- **Rec:** A reconstruction support framework tightly coupled with Moore, used for detector-level track, particle reconstruction, and real collision data.
- **DaVinci:** A post-trigger analysis framework, used primarily for physics analysis. It allows for the reconstruction and selection of decay channels of interest.

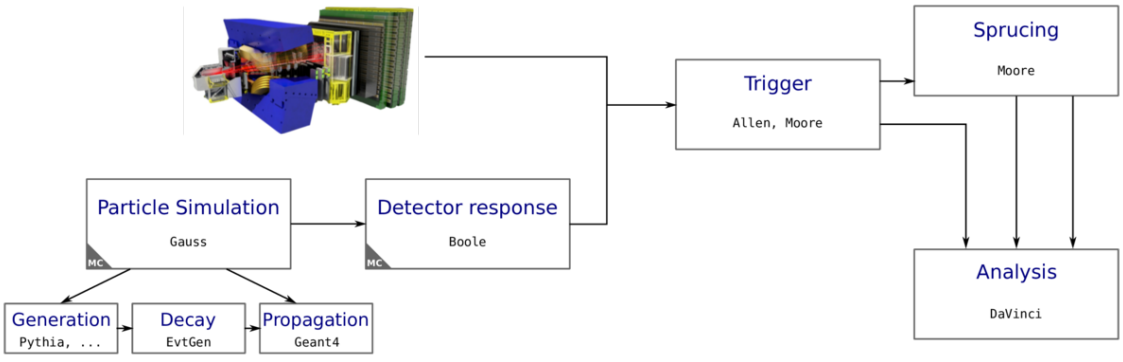


Figure 11: Software Stack at LHCb [27]

3.1 Gaudi Framework

Gaudi is a modular and extendable framework designed for event data processing and algorithm execution [28]. Initially implemented in C++98, the framework has been substantially upgraded to C++11 and subsequent standards. The LHCb Event Filter Farm (EFF) exploits both x86-64 processors as well as GP-GPU chips. For the CPU computations, it leverages *Non-Uniform Memory Access* (*NUMA*) architectures to optimise parallel performance. Using the task-based scheduling model, Gaudi employs a multithreading model that supports concurrent event processing. This allows algorithms to be executed in parallel whenever dependencies permit.

The central component of the Gaudi framework is the *transient event store* (*TES*). It serves as an in-memory data exchange layer between algorithms. TES ensures type-agnostic data storage using standard STL containers, such as `std::vector`, and is designed to be immutable and thread-safe, enabling safe concurrent access. During Run 2, several performance bottlenecks were identified, including blocking I/O operations, suboptimal offloading to coprocessors, limited RAM scalability, etc. These constraints were mitigated in Run 3. This was achieved with the help of:

- Adoption of a concurrent data processing paradigm through simultaneous multithreading.
- Major refactoring of the core algorithms to ensure thread safety.
- Implementing improved dynamic load balancing strategies, eliminating the need for complete process restarts.

Together, these established a foundation for real-time data processing, substantially reducing the computational cost per event.

3.2 Ganga: Grid Interface

Ganga is a job management tool by LHCb and ATLAS. Provides a unified environment for submitting, monitoring, and managing computational tasks across heterogeneous distributed resources across the grid. Enabling Ganga helps seamlessly prototype analyses locally and scale them to a grid with minimal modifications in the configuration files. The necessary job splitting, resubmission, and resource allocations are made behind the scenes for the user, making it an essential tool for high-throughput production data and simulations.

3.3 Simulation Tools

The experiments rely on Monte Carlo simulated data for various reasons, including validating reconstruction algorithms, designing trigger algorithms, and optimising algorithms.
Three popular tools that are designed for the complete chain of particle interaction and detector responses are:

Pythia Generates the initial hard processes and subsequent parton shower using QCD-based models [29]

EventGen Helps simulate the decay of unstable particles and takes into account sophisticated models to generate CP-violation effects in the final states

Geant4 The tool propagates particles through the detailed detector geometry, considering material interactions, energy losses, secondary particle productions, etc. [30]

3.4 Moore

Moore Project serves as the framework for the LHCb trigger application and is responsible for filtering the events that arrive at an input rate of 40 MHz collisions down to an output rate of around 100 kHz. LHC operates with the machine collision rate of 40 MHz while LHCb operates at 30 MHz visible interaction rate. The high-level trigger in LHCb consists of two stages:

HLT1 : Partial reconstruction of tracks using Allen and executed on GPUs.

HLT2 Full reconstruction of tracks using Moore and executed on CPUs.

The Figure [12] shows the HLT 2 architecture design. Moore is built on top of the Gaudi framework and operates in a hybrid implementation strategy: the core functionalities are implemented using C++ for performance, and the trigger logics and the workflow controls are done using configuration packages provided as Python option files. This modular design helps in flexible reconfiguration and rapid development of trigger strategies. The Python modules help test configurations, tools, and dataflows without modifying the core algorithms.
Moore produces two analysis-ready datasets:

Turbo stream : This consists of 60-70%

Full Stream : This is the complete event data stored for calibration and rare analysis.

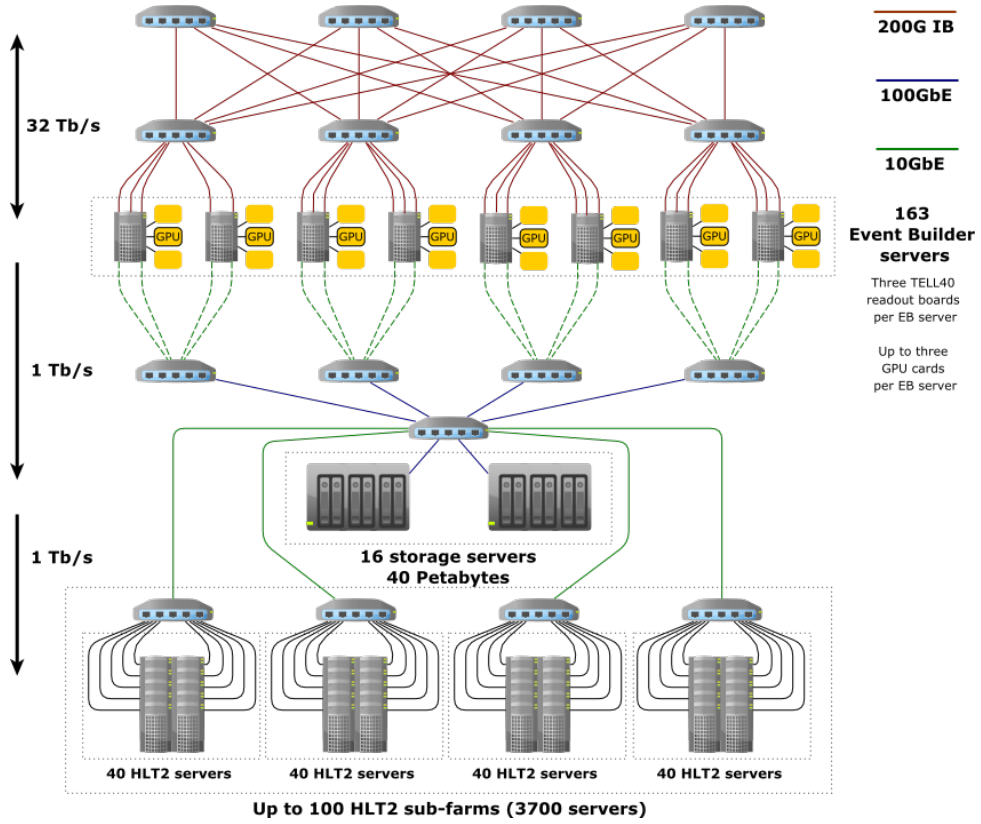


Figure 12: HLT2 Architecture Design [31]

This is the first time we have a complete real-time trigger system that solely depends on the decisions made by the software triggers and does not store most of the RAW data for later processing. Moore is designed to work by executing the trigger lines of interest and writing events into different streams based on the lines it fires.

3.5 Rec Modules

The LHCb stack, the software component of the experiment, consists of a series of interdependent algorithms designed to perform specific, designated tasks. This highly modularised framework is designed to execute high-performance computational tasks most optimally. The *Rec* frameworks provide the complete reconstruction logic required by Moore to perform real-time trigger decisions. Its core functionalities include track reconstructions, particle identifications, and integration with Monte Carlo truth tools for validation and performance studies. In the context of this thesis, two main algorithms, along with several associated ones within the *Rec* framework, were considered for development.

3.5.1 Hybrid Seeding Algorithm

Hybrid Seeding Algorithm is a stand-alone reconstruction procedure developed to identify and label hits to build SciFi tracks. These tracks function as seeds for further track reconstruction. In contrast to other algorithms, it does not require external input from additional detectors and can reconstruct tracks solely with the hits from the SciFi tracker (thus, in this context, it is a stand-alone process). This track reconstruction strategy builds the ground for the reconstruction of other track types mentioned earlier, by extending them upstream through the detector while accounting for the effects of the magnetic field. In the experiment system, z-axis is measured along the beam direction, x-axis points outward from LHC ring centre, and y-axis shows the direction pointing upwards.
To form the tracks, the algorithm follows the following steps, and the functions are mentioned for reference:

Initialisation: Preparation of required prerequisites, including the data structure to hold the hit data from the SciFi tracker.
`[ModPrHits and PrFTHandler]`

Core Parametrisation: This is the algorithm's core. Using the hits from the *T1* and *T3* stations, we can form a doublet of hits in the SciFi tracker. The algorithm considers three different momentum ranges: low, medium, and high transverse momentum, and further splits the detector geometry into two parts: top (y>0) and bottom(y<0), to help solve the problems in different parameter sets and reduce the complexity.

- **Case 1: Tight Tolerance – High Momentum Tracks:** X–Z projection is almost a straight line.
- **Case 2: Medium Tolerance.**
- **Case 3: Loose Tolerance – Low Momentum Tracks:** X–Z projection shows significant parabolic curvature.

Based on the momentum condition, the doublets can be formed into a straight line for high-mass particles and a parabolic equation for low-mass particles. This approach starts with high momentum tracks with the least parabolic curvature and the highest quality of tracks, then eliminates the used hits and narrows the search space. Once this is done, based on the tolerance window, a search for the third hit is initialised, and the closest point is picked.

X-Z Projection Finding: This is one of the most computationally intensive steps in the algorithm. This step tries to form a 2D track in the non-bending (X-Z)

plane in every parametrised evaluation cycle. Once the hits are found and fitted to a cubic corrected parabola, the χ^2 value is evaluated for the track quality. If the value is high, the track is rejected, and for a lower value χ^2, the track is considered.
`[Hybrid::SeedTrackX]`

X-Z Clone Killing: Removing any copy of tracks before it goes for the next steps is essential. This step ensures that no duplicate 2-dimensional track candidates are removed.

Stereo Hits Addition : Using the formed 2D projections, these steps add hits from the UV planes to form 3D tracks.

$$S = x \cos\alpha + y \sin\alpha, \tag{4}$$

where α is the stereo angle of the detector.

Instead of this form, we use a more robust pattern recognition method called the Hough Transform. In simple terms, this method considers all possible candidates and shapes possible and picks the best possible track candidates from the noisy environment. In this context, it forms the clusters of hits formed in the detectors that belong to the same track, creating a peak in the Hough space. Using these hit clusters and the 2D tracks, we can form the 3D SciFi tracks.
`[Hybrid::SeedTrack]`

Hits Flagging : This is one of the essential steps to reduce the complexity of track reconstruction exponentially. After each step, every used hit is tagged and removed to be used in further reconstructions. This prevents it from being used in subsequent cases, helping to reduce the search space drastically.

Track Recovery : This step can be considered as a safety check to recover any missing tracks in the reconstruction. This is achieved if any strong track candidate at any point of reconstruction was removed from the final candidate selection for any reason.

Final Clone Removal : This step involves removing any final duplicated 3D tracks in the reconstruction sequence that can be removed before storing the SciFi tracks.

Output Track Conversion : This step involves converting the final tracks to reusable, stand-alone SciFi Tracks

3.5.2 Long-lived Tracking Algorithm

The Downstream Tracking algorithm undergoes an intensive computational process to address the complexities introduced in Run 3 from the increased data volume and precision demands. The increased precision requirements require meticulous reconstruction of the seed tracks, leveraging signals from the SciFi tracker to accurately identify and initiate (track seeding) the track reconstruction. Subsequently, these reconstructed track segments are extrapolated to the Upstream Tracker (UT) detector, allowing the algorithm to discern and isolate the distinct characteristics of Downstream Tracks.

Run 3 conditions increase the complexities of Downstream Tracking and introduce increased computational tasks. The algorithm now handles an increased volume of data, participating in the computation of numerous additional vertices, etc. Moreover, the increased precision mandates significantly removing combinatorial tracks, reflecting the algorithm's commitment to delivering accurate and reliable results. The *Long Lived Tracking Algorithm* is initiated at the SciFi Tracker from the SciFi tracks (seeds). Three stations on the SciFi Tracker provide the first track segment, which will then be extended to the UT Detector. In the case of MC samples, the algorithm also checks for the initial conditions to be met:

- The track should not originate from an electron
- The track should have at least 70% matching hits
- The hits should be associated with a charged particle.
- The track should not have VELO hits associated.

The Pattern Recognition Algorithms search for clusters in two X-Layers of the UT Tracker, and Clusters from the U-V Layers, showed in Figure [6] are added. The signals from both trackers are then fitted using track χ^2.
Later, a high-threshold bit flag will be associated to reduce the spillover impact on the reconstruction process. This flag will be associated with each cluster if it is generated in collisions that differ from the current one (the spill-over detection algorithm is run at the previous stage and helps reject hits originating in previous or next bunch crossing). Clusters associated with High Threshold Bits will be rejected. As a final step, a Multivariate Classifier is deployed to select the best final Downstream Tracks.

There are a few conditions associated with selecting the right Downstream Tracks, which are as follows:

Initialisation and Loading of Evaluation Models : The algorithm's initialisation involves initialising necessary data structures similar to Hybrid Seeding Algorithms. On top of that, in this case, we have a more sophisticated statistical model implemented in the algorithm for the track selection. At this stage, the algorithm initialises the necessary weight parameters for the models.

Iterating through seed tracks As mentioned, SciFi tracks act as seed tracks for reconstructing Downstream Tracks.

Extrapolation : For each seed track, a simple object is created that takes the state of the seed at the last station and extrapolates to UT upstream, and stores it as a lightweight internal track model.
`[PrDownTrack]`

Pre-selection: In this stage, the hits are collected within the momentum-dependent window defined within the UT detector. Stage of a track composed of the position and the direction of the track and obtained from `[State::Location::EndT]` `[getPreSelection]`

Combinatorial Search : Based on the last step, a combinatorial search is performed in search of track patterns. This is more of a brute force approach, since the pre-selected tracks are fewer compared to the overall noisy environment.
`[createTrackCandidates]`

Track Fit : The algorithm uses a simplified fit method to form the track at this level.
`[simplyFit]`

Final Selection with MLP : A Multi-Layer Perceptron (MLP) model is trained and deployed to evaluate track quality rather than having intricate cuts on the statistical values [32]. The track is accepted based on the probability that the track originates from real decay.

Output Conversion : This is the final stage where the best tracks are selected and saved for further analysis. `[Downstream Track]`

3.5.3 Reconstructed Tracks Monitoring Algorithms

One of the other important components of the offline analysis is the **RecoDumpers**. The algorithm `PrTrackRecoDumper` is a consumer module that produces detailed

information about the reconstructed tracks and stores them in ROOT ntuples using Moore for offline analysis. This monitoring algorithm is particularly significant in the framework of this thesis, as it provides comprehensive event-level information, including the ground truth. This can be used to develop predictive models based on computational intelligence (Machine Learning) to identify the true particle tracks.

The algorithms operate on the reconstructed track container and the MC particles key table, which contain information about true Monte-Carlo particles associated with the reconstructed tracks. They also have access to the complete collection of detector hits and the event number for bookkeeping. The algorithm uses these inputs to produce a single ntuple file, where each entry corresponds to an individual reconstructed track. The algorithm loops over every track in the input container and proceeds as follows:

MC matching : Use the link table to find the MC particles associated with the reconstructed track; If a good match is found, the *isMatched* variable is flagged as true.

Variables of Interest : All predefined variables are set and called. e.g., Kinematical Variable, Fit Quality, Origin, MCTruth, etc.

Associating with Hits Information : Iterate through the LHCb-IDs on the track and find the original hits from the detector hit container.

Fill Trees : This is the final step, where the collected information is saved to the output ntuple file.

3.6 DaVinci Framework

DaVinci is the central component of the offline physics analysis framework, which is optimised for post-trigger data processing. DaVinci performs the full reconstruction of signal channels, refines offline selection criteria, and produces analysis-ready n-tuples. It acts as a bridge between trigger selections and offline analysis.

DaVinci is built upon Gaudi and follows a similar architecture to the other components, like Moore. It consists of core algorithms written in C++ and configurable option files in Python for modularity and efficiency. DaVinci uses persisted candidates from HLT2 or Sprucing with additional minimal context as input, avoiding the requirement of rerunning full offline reconstruction.

Chapter 3

Data-Driven Approaches in HEP

Artificial Intelligence (AI) and Machine Learning (ML) have become an integral part of research and computational analysis in recent years. As in many other domains, AI and machine learning demonstrate significant potential in high-energy physics. They offer solutions to complex problems, ranging from simulations to real-time triggers and physics selections. The complexity of simulations, detector designs, and large-scale numerical calculations has increased, especially with the added challenges of Upgrade 1. This has created a need for more robust problem-solving approaches. Today, improved hardware technology and larger volumes of data make such approaches feasible and exciting research. Traditional statistical methods struggle with non-linear relationships and high-dimensional data [33]. In contrast, modern ML methods excel at pattern recognition. They can address the challenges facing traditional approaches.

1 How Do Machines Learn?

Machine learning is a computational paradigm that elevates traditional problem-solving to more advanced data-driven methods. At its core, machine learning allows machines to learn intricate patterns and build creative predictive models. This happens without the need for explicit rule-based coding. Machine learning can extract previously hidden insights from large datasets by combining statistics and probabilities with advanced computational machines. Recent years have witnessed a significant shift, driven by machine learning and artificial intelligence. Two primary factors have driven this change: the exponential growth in data production and the advancement of computational hardware [34]. The abundance of data and the increase in computational power have propelled advances.

1.1 Supervised Machine Learning

Supervised Machine Learning uses the labelled data for training. Each input variable is coupled with a corresponding output or value. The objective is to learn a mapping from the variables to the label in a multi-dimensional space.
The central idea is to learn a mapping function

$$f : X \longrightarrow Y \tag{5}$$

that represents the data in the feature space. This function is later used for accurate predictions on new data points. Supervised ML is broadly divided into two categories.

Regression Problems are a core task in supervised machine learning in modelling a relationship from input feature space to predict the output as a continuous value. *Classification Problems* are probability-based problems; the model predicts the class to which a data point belongs. Based on a variable threshold parameter, the class of a data point is decided.

1.1.1 Linear Regression

Linear Regression Models are among the most widely used approaches in supervised machine learning problems, mainly due to their simplicity and interpretability. They provide the foundation for regression and classification problems, making them a key component in predictive modelling [35]. A model response plot for a simple linear model is showed here at Figure [13] The central assumption of any linear modelling is that the relation between the input and output is linear and can be expressed as a weighted sum of the inputs.

For linear regression to yield valid and interpretable results, the following assumptions must be made:

- **Linearity:** The relationship between independent and dependent variables is linear.
- **Independence:** The observations are independent.
- **Homoscedasticity:** The variance of the residuals is constant across all levels of the independent variables.
- **Normality:** The residuals follow a normal distribution with a zero mean.
- **No Multicollinearity:** The independent variables are not highly correlated.

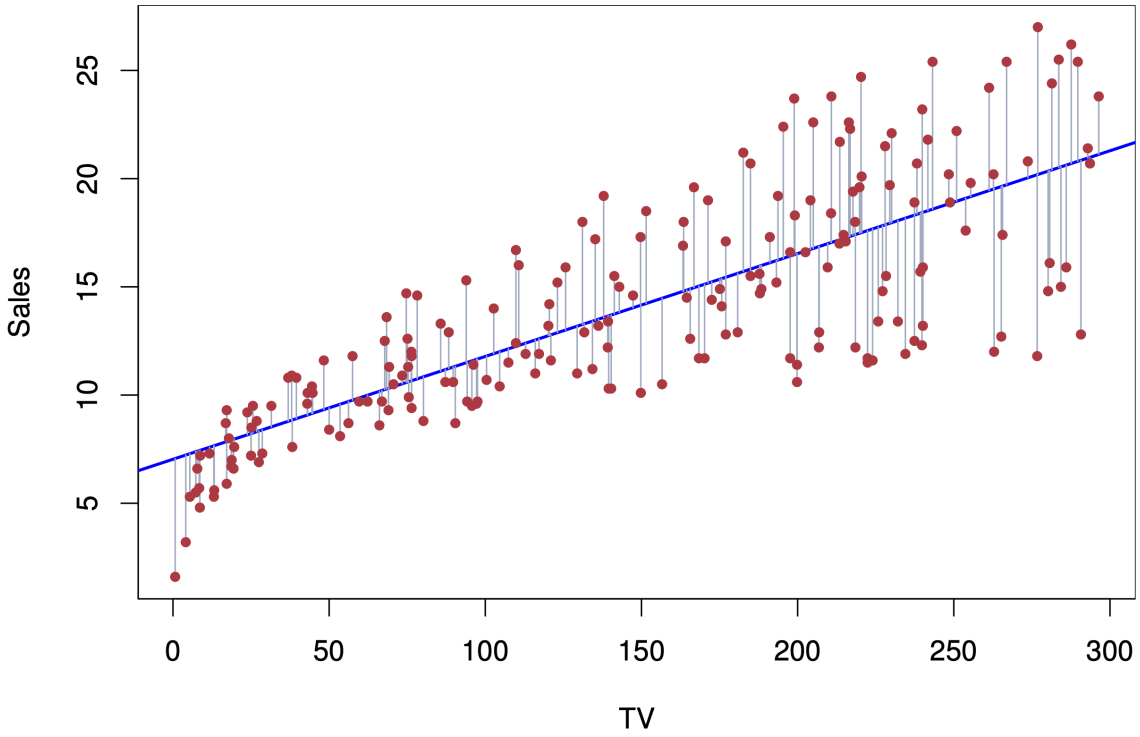


Figure 13: Linear Regression Models [36]

The Figure [13] illustrates the model response for linear regression and can be represented as:

$$Y = \beta_0 + \beta_1 X + \epsilon, \tag{6}$$

where Y is the dependent variable, X is the independent variable, β_0 is the intercept, β_1 is the slope coefficient, and ϵ is the error term. The goal of training the model is to find the optimal values of β_0 and β_1 that minimise the prediction error.

1.1.2 Logistic Regression Models

Logistic Regression is a statistical method used for classification problems, where the target variable can take only two or more distinct classes. They use a *sigmoid* or *soft-max* function to map the linear combination of input variables to probabilities.

The linear composition of this problem is expressed as:

$$z = \beta_0 + \beta_1 x_1 + \beta_2 x_2 + \beta_3 x_3 + \cdots + \beta_n x_n \tag{7}$$

To model the probability of a particular class, a logistic function, also known as the sigmoid function, can be used, defined as:

$$\phi(z) = \frac{1}{1 + e^{-z}} \tag{8}$$

Where:

- z is the input to the function (can be a linear combination of features),
- $\phi(z)$ is the output of the sigmoid function, bounded between 0 and 1.

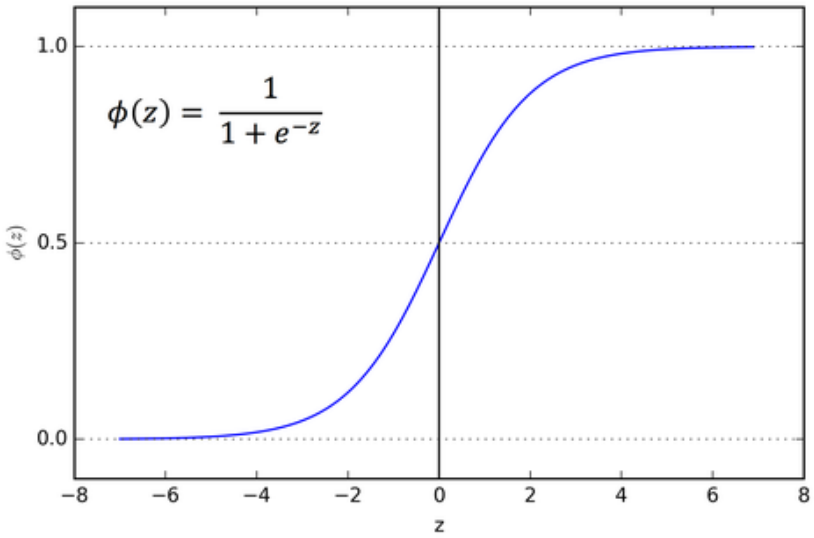


Figure 14: Logit Function [37]

The sigmoid function in, Figure [14] maps any input with real values to a value in the interval (0,1), making it suitable for estimating probabilities in binary classification problems. Logistic regression assumes a linear relationship between the dependent variable, the independent variables, and the log-odds (also called the *logit*). Here, odds represent the ratio between the probability that the event will occur and the likelihood that it will not. The logarithm of the odds or logit is simply the natural logarithm of these odds.

$$\log\left(\frac{p}{1-p}\right) = \beta_0 + \beta_1 X_1 + \beta_2 X_2 + \cdots + \beta_n X_n \tag{9}$$

where:

- p is the probability of the event of interest,
- X_i are the predictor variables ($i = 1, 2, \ldots, n$),
- β_0 is the intercept term,
- β_i are the coefficients associated with each predictor variable.

The objective is to find coefficients $\beta_0, \beta_1, \ldots, \beta_n$ that maximise the likelihood of the data and minimise the prediction errors. Like other linear models, logistic regression is valued for interpretability, simplicity, and computational efficiency. The learned coefficients reveal each feature's relative importance and direction of influence on the outcome. Nevertheless, despite these advantages, logistic regression is limited by its linear decision boundary, making it not preferred for modelling complex or non-linear relationships between features. It is also designed for binary classification; multinomial or *softmax* regression extensions are needed for multi-class tasks.

1.1.3 Nearest-Neighbor-Algorithms

Nearest-Neighbor-Algorithms are a step above linear models in their complexity. The predictions are based on the distance metrics from the data point of interest to the closest labelled data points. The most popular algorithm in this case is the K-Nearest-Neighbours Algorithm or KNN Algorithm. It uses various distance metrics, including Manhattan, Euclidean, and cosine similarity. Here, the assumption is that similar input data points produce similar output data points.

KNN is a non-parametric method that can be used for both regression and classification. They do not have assumptions about the underlying probability distribution in the data, but similar data points are expected to produce similar outputs. Examines and identifies similar data points around the instance and predicts the value or label based on the point. The essential key parameter to use is "k", which represents the number of data points to consider when evaluating the new value of the prediction. In classification, the new label is typically determined by a majority vote system, while in regression, it is often the average of the neighbours' target values. Due to the inherent complexity of distance measurement calculations for individual data points, they are rarely used in large datasets.

1.1.4 Naive Bayes

Naive Bayes is a generative model that connects statistics and probability with advanced computational models. It is based on Bayes' theorem, considering that all input feature spaces are conditionally independent. The theorem calculates the conditional probabilities and describes the probability of any event based on prior knowledge of conditions that could be relevant to the event.

It can be mathematically represented as

$$P(A|B) = \frac{P(B|A) \cdot P(A)}{P(B)} \tag{10}$$

where:

- $P(A|B)$ is the probability that event A occurs given that B has happened,
- $P(B|A)$ is the probability that event B occurs given that A has happened,
- $P(A)$ and $P(B)$ are the independent probabilities of events A and B, respectively.

The assumptions based on the theorem are less likely to be accurate in practice. Naive Bayes has been in practice for a long time across various sectors, and its application in the context of Machine Learning makes it even more helpful.

1.1.5 Decision Trees

A Decision Tree is a parametric model and one of the most popular Machine Learning Algorithms. They represent the decision-making process as a hierarchical structure of nodes and branches, which resembles an inverted tree as shown in the Figure [15]. It works by recursively partitioning the input feature space into smaller regions based on the similarities between categories. The learning process of these models involves selecting the right features and split points that best divide the prior category into more segregated subgroups. It uses splitting criteria such as Gini impurity (GI) and entropy to separate the categories for classification problems and MSE reduction for regression problems.

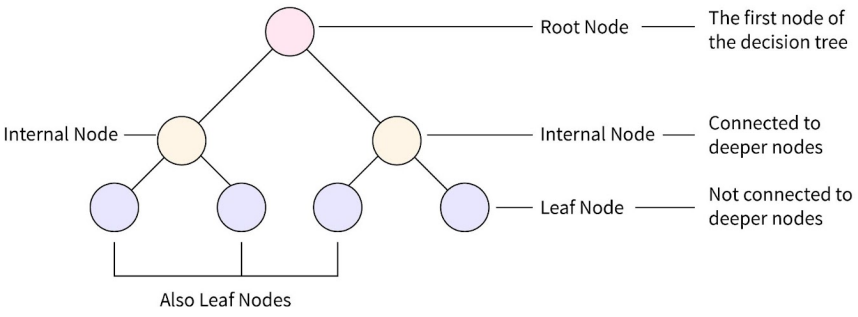


Figure 15: Decision Tree Basic Structure [38]

The algorithm recursively divides the data until pre-defined conditions are met. They inherently redefine the workflow of predictive model building. They are interpretable models that require no feature scaling and can handle continuous and categorical values. One of the main challenges is their tendency to overfit when the tree grows too deep. Pre-defining the depth, or "pruning" techniques, pre-defining multiple trees helps minimise overfitting and avoid memorising the patterns. Combining multiple randomly generated decision trees and aggregating the results is a common practice, known as a *Random Forest* Model. This ensemble learning technique offers several advantages over Decision Trees. The aggregation of the trees improves precision and resilience compared to using a single Decision Tree.

1.1.6 Boosting Algorithms

Boosting is another ensemble method that combines many weak learners to create a more stable and accurate machine learning model. This iterative learning process involves multiple models in different subsets of data, focusing on the mistakes made by the last model. The final model is a weighted sum of all the models, making it more effective in predictive analysis. AdaBoost (Adaptive Boosting), Gradient Boosting (GBM), XGBoost (Extreme Gradient Boosting), Light GBM, and Catboost are some of the most popular boosting algorithms currently widely used [39]. Boosting models show a trend in producing scalable solutions for massive systems and provide GPU-based training, making it easier to develop for large datasets.

1.2 Unsupervised Machine Learning

Unsupervised Machine Learning is a type of ML problem involving no actual or labelled training data, meaning no explicit target variable is provided. In this case, the algorithms used are to discover underlying patterns or structures within the data with minimal instructions. This can be achieved using various methods, such as exploring the inherent structure and relationships between data points in a multi-dimensional space. The most popular methods are *clustering* and *Dimensionality Reduction*. Unsupervised learning is used in synthetic data generation, Generative Adversarial Networks (GANs), Principal Component Analysis, and other applications.

1.2.1 Clustering

The most popular methods in unsupervised machine learning aim to partition datasets into smaller subsets (clusters) as shown in the Figure [16 that share similarities. They are centroid-based methods, where a randomised centroid is generated, and clusters are formed based on the distance between the centroid and the data instances. The objective is to find that the data point should be close to the centroid of a similar cluster and far from the centroids of other clusters.

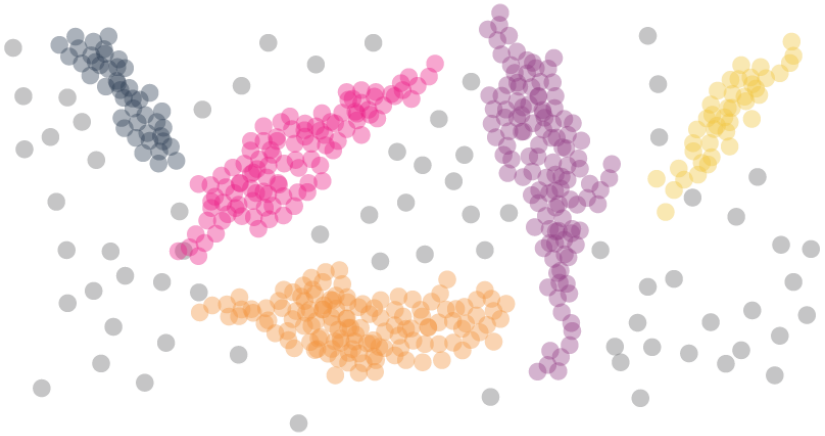


Figure 16: Simple Clustering

K-Means Clustering : It is a centroid-based approach to identifying the underlying patterns within the data by grouping similar data instances.

Hierarchical clustering : This method uses agglomeration or division methods to build a hierarchy of clusters.

DBSCAN : Density-Based Spatial Clustering of Applications with Noise, this method groups points based on density, which is robust to noise and non-spherical clusters.

1.2.2 Dimensionality Reduction

Dimensionality reduction is used in Machine Learning to deal with redundant or noisy datasets. It helps reduce the number of input variables' feature dimensions by minimising information loss. High-dimensional data is often complex to analyse; dimension reduction helps overcome this challenge.

Principal Component Analysis (PCA) : One of the most popular feature extraction methods is reducing the dimensionality while maintaining a high variance. PCA is a statistical protocol that reduces dimensions by creating an uncorrelated feature space.

Auto-Encoders : Neural networks designed to compress and reconstruct data.

1.3 Reinforcement Learning

Reinforcement learning is a slightly different paradigm compared to the first two types of learning. Here, learning focuses on training agents to make decisions, interact with environmental parameters, and improve after each iteration through feedback loops. Feedback can be rewards or penalties for positive and negative decisions, referred to as cost functions.

Reduction learning algorithms are designed to optimise the agent's strategies to maximise cumulative rewards over time, resulting in the most favourable outcomes. RL is driven by a trial-and-error strategy when making future decisions.

2 Neural Networks

At the foundation level, artificial neural networks are designed to mimic the structure and functioning of the human brain—the idea of artificial neural networks for computation dates back to the mid-20th century. The first node, *Perceptron*, was introduced by Frank Rosenblatt in 1958. Implementing back-propagation in the 1980s enabled the development of multilayer perceptrons, also known as multilayer neural networks or MLPs, that are trained more efficiently [40]. However, one of the breakthroughs of NN in mainstream science and industry happened in 2012, when working on a neural network architecture called *AlexNet*, a deep convolutional neural network developed by Alex Krizhevsky and his team, achieving a dramatic breakthrough in the image recognition domain, beating other model performance at the time with significant margins [41]. Deep research in this field took a long time to gain popularity, primarily due to hardware limitations, the large-scale requirements, and the scarcity of data.

2.1 Perceptrons to Deep Neural Networks Architecture

The design of neural networks is inspired by the human brain, which has a structure of interconnected layers of nodes or neurons as shown in Figure [17]. Neurons process the data through weighted connection layers. Each neuron processes a weighted sum of its inputs and passes it to the connected neurons. They are triggered based on non-linear activation functions like step or sigmoid functions. By stacking many such neurons, they act as a complex system that can compute and approximate more complicated functions.

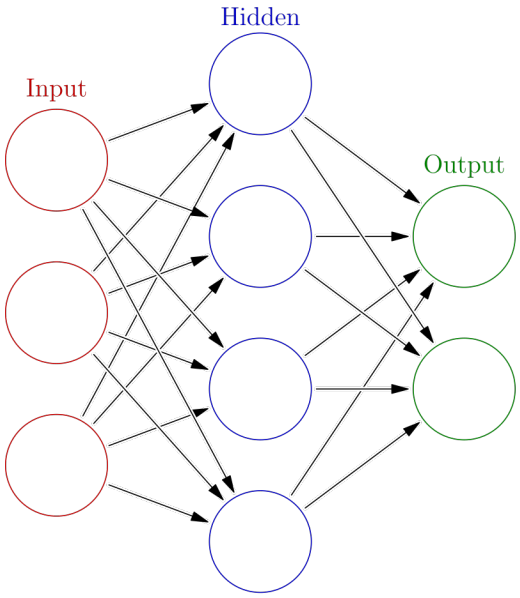


Figure 17: Basic Neural Network Structure [42]

A basic network of connected nodes (also called a feed-forward neural network) consists of three such layers.

Input layer : Taking in raw data features.

Hidden layer Intermediate layer of nodes that transform the input data through activation functions and respective weights.

Output layer : The final nodes responsible for the final predictions, such as probabilities.

During training, the model learns and adjusts the weights and biases of the connected layers. Loss functions help minimise errors during the training process. The model utilises back propagation in the feedback loop for computing loss gradients concerning the weights and improves the model's training. Each type of problem requires a specific architecture to achieve better model performance.

Non-linear activation functions like ReLU are crucial in neural networks, as they enable the learning of complex relationships within the data [43]. ReLU activates the connection when the input is positive and remains inactive when the input is zero or negative. This feature improves computational efficiency and mitigates vanishing gradient problems.
ReLu is represented as:

$$f(x) = \max(0, x) = \begin{cases} x & \text{if } x > 0, \\ 0 & \text{if } x \leq 0. \end{cases} \tag{11}$$

Other popular activation functions include the *Sigmoid function*, the *hyperbolic tangent function* (*Tanh*), the *softmax*, and the *leaky ReLU*.

3 Model Building and Evaluations

Most machine learning algorithms are designed to identify patterns in data and improve their performance over time [35]. At a basic level, they create a mathematical function that maps the input features to the output. Training occurs when the algorithm examines all training data and determines the optimal settings to map the features. The original data are usually split into two sets: one for training the model and one for evaluating how well the model performs.

These are key parameters while building or evaluating a machine learning model.

Loss Function Helps measure how well the model's predictions align with the target values. Model training aims to minimise the loss function and improve the predictions.

e.g., mean squared error (MSE), mean absolute error (MAE), cross-entropy (CE) loss, etc.

Optimisation Algorithms : They help in minimising loss functions. For example, the gradient descent algorithm attempts to find the minimum of a function by taking small steps in the direction of the gradient vector. The algorithm aims to find the global minima and avoid the local minima. In a multi-dimensional feature space, finding the global minima presents challenges, and the complexity of the model and the optimisation algorithm is chosen based on the hyperspace complexity.

Regularisation Techniques : Penalty functions that prevent the model from memorising the patterns and build a more general mapping of feature space. This is called *Over-Fitting* of the model. Regularisation adds a penalty to the loss function, preventing it from growing in complexity, and improving the model's generalisation for better performance when evaluating unseen data. E.g.: Lasso (L1) and Ridge (L2) techniques, dropout, etc.

Evaluation metrics The metrics used after training to evaluate model performance. They are chosen based on the type of problem. In cases where continuous values are the target, metrics such as R^2 or RMSE are widely used. If the model tries to predict classes, accuracy, precision, recall, and the F1 score are generally considered.

3.1 Confusion Matrix

Confusion metrics are essential for assessing the model's performance, as they categorise the predicted outcomes against the actual outcomes. The model aims to have a maximum number of correctly classified data points.
Outputs from the confusion matrix can help fine-tune the model.

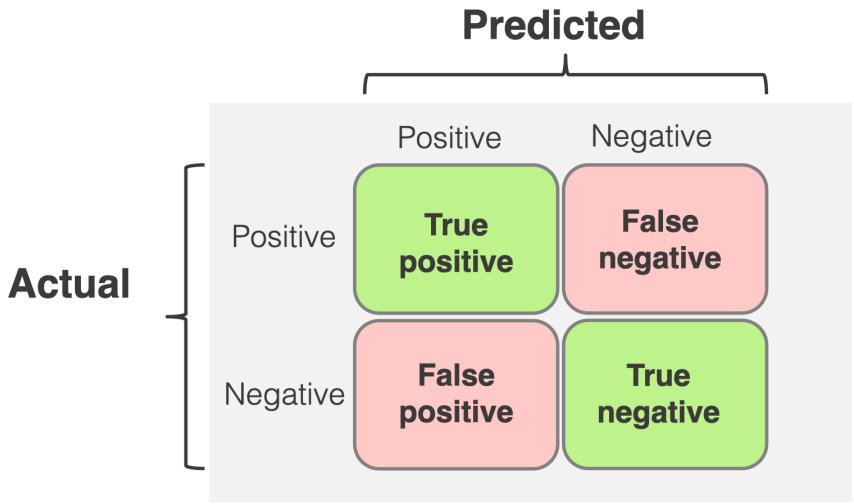


Figure 18: Confusion Matrix for Binary Classification

A confusion matrix for binary classification contains True Positives (TP), where the model correctly predicts a positive. It also contains True Negatives (TN), where the model correctly predicts a negative. False Negatives - FN (Type II Error) occur when the model incorrectly predicts a negative outcome instead of a positive one. False Positives - FP (Type I Errors) occur when the model incorrectly predicts a positive result instead of a negative one. We can use the confusion matrix to evaluate a model using the following metrics:

Accuracy: Measures the overall performance of the model.

$$\text{Accuracy} = \frac{TP + TN}{TP + TN + FP + FN} \tag{12}$$

Recall (Sensitivity / True Positive Rate): Indicates the proportion of actual positives correctly identified.

$$\text{Recall} = \frac{TP}{TP + FN} \tag{13}$$

Precision: Measures how many of the predicted positives are actually correct.

$$\text{Precision} = \frac{TP}{TP + FP} \tag{14}$$

Specificity (True Negative Rate): Shows the proportion of actual negatives correctly identified.

$$\text{Specificity} = \frac{TN}{TN + FP} \tag{15}$$

False Positive Rate (FPR): Indicates how often a negative instance is incorrectly classified as positive.

$$\text{FPR} = \frac{FP}{TN + FP} \tag{16}$$

F1 Score: Harmonic mean of precision and recall, balancing performance for uneven class distributions.

$$\text{F1 Score} = 2 \times \frac{\text{Precision} \times \text{Recall}}{\text{Precision} + \text{Recall}} \tag{17}$$

3.2 ROC Curve

The Receiver Operating Characteristic (ROC) allows for assessing binary classification models [44]. As it shows in Figure [19], it shows the balance between True Positive Rates (TPRs) on the Y-axis and False Positive Rates (FPRs) on the X-axis. ROC gives vital clues:

Point (0,0) : A model with no positive outcomes

Point (1,1) : A perfect model

Line (Y=X) : A random guessing model with 50:50 probabilities.

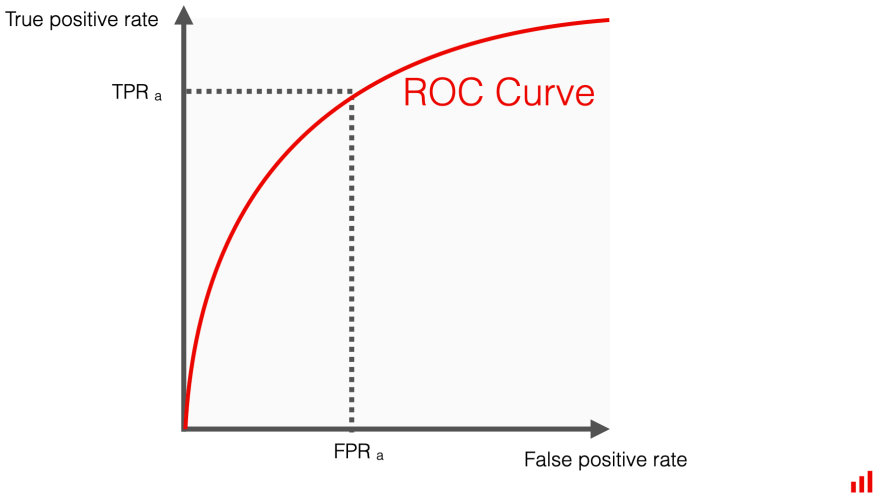


Figure 19: ROC Curve [45]

The area representing the upper side of the y = x line indicates a good model, whereas the area below the line suggests a model worse than random guessing, often due to labelling problems. ROC serves as a valuable metric for pinpointing a classifier's operating point and holds information not affected by the imbalance of the target variable.

4 Interpretability of Models

Most complex ML models act as a black box, making understanding the reasoning behind their decisions difficult [46]. Linear models are easier to interpret but lack the possibility of non-linear activations. In many cases, it is equally important to have an interpretable model. Methods like SHAP provide the interpretability of complex algorithms and their decisions.

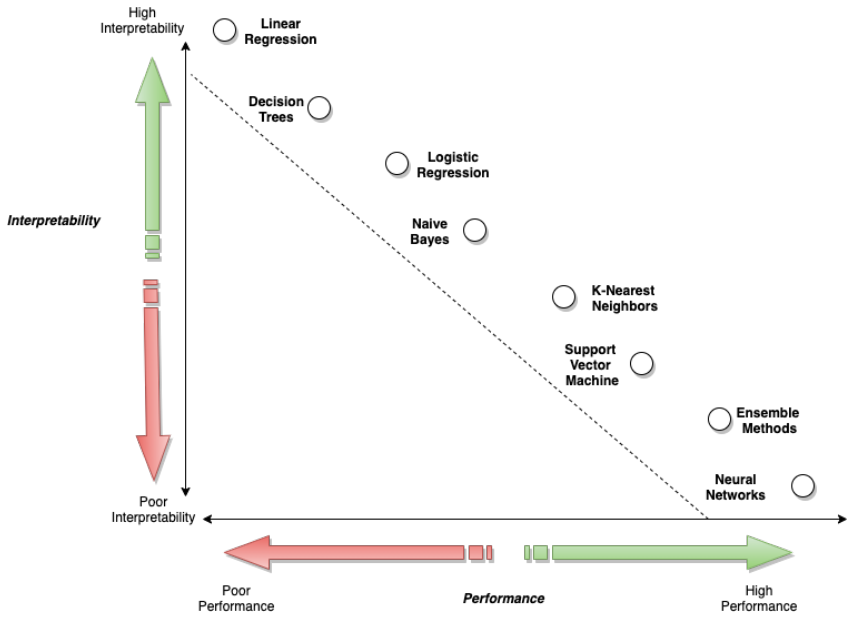


Figure 20: Interpretability and Performance of Machine Learning Algorithms

4.1 SHapley Additive exPlanations(SHAP)

SHapley Additive exPlanations, or SHAP, stands out as a potent and widely utilised method for comprehending machine learning models and their outputs [47]. It operates on the principles of cooperative game theory, leveraging Shapley values to function effectively. These Shapley values facilitate understanding the model's response to each distinct feature.

5 ML Strategy and Real-Time Constraints

This project deployed a cascade machine learning model approach to improve Downstream Track reconstruction. The following chapters present a detailed analysis of these models, focusing specifically on their application to the research use case. The objective was to implement and test two separate ML models, each trained for this particular scenario, as part of a combined two-stage approach. Deployment of the models required strict operational constraints, which are also considered in the study. Due to the scope and volume of the work, some additional studies conducted during the project are not discussed here.

In summary, the research followed a systematic workflow, covering model selection, training data preprocessing, speed and performance constraints, and the choice of libraries and model architectures. Subsequent chapters present only the key results that contributed directly to the final solutions.

Phase II: Strategy

Two-stage Model Building and Integration

Chapter 4

Stage One Selection: Enhancing SciFi Track Purity

1 Objectives

SciFi Tracks serve as the foundation track segment in reconstructing the Downstream Track. The first selection stage in the research focuses on enhancing the quality of these SciFi tracks, as their improvements directly influence the overall performance of subsequent track reconstruction. SciFi tracks are reconstructed using a standalone algorithm that relies solely on hits recorded in the SciFi Tracker.

SciFi tracks are crucial elements used as Seed track segments for all other track reconstruction procedures, including Downstream Tracks. Ensuring all the true tracks are selected and the combinatorial track segments are rejected is a significant step in improving SciFi segments' purity and, subsequently, Downstream Tracks' quality. Any improvements in their selection and quality translate to better efficiency and track quality across all the track reconstruction algorithms using them as Seed Tracks. A primary challenge is distinguishing accurate tracks from combinatorial tracks called Ghost Tracks, which are false, non-physical tracks that do not match any real track trajectories. Selecting true tracks and removing Ghost Tracks is crucial to achieving high reconstruction quality, but this process is computationally expensive. Without effective separation, track quality drops significantly.

The *SciFi Track Classifier* is a core machine learning model developed in this thesis to improve seed track purity in track reconstruction, and it is a crucial step in improving Downstream Track reconstruction. This model is built by learning the detector responses and reconstruction features. Using these learned characteristics, the method can strengthen existing algorithms' robustness and improve performance and efficiency in reconstructing Downstream Tracks.

2 Methodology

The training data set comprises over one million Monte Carlo (MC) simulated tracks. It focuses on the decays of K_s^0 (short-lived neutral kaons), which are particularly relevant for the model's development. Each SciFi track is represented by eight features that capture key selection criteria.

Using the data set described above, the training process utilises *Moore* to produce tracks and configure the trigger conditions for selection. Various classification methods are evaluated to maintain an incremental development of the tracking pipeline. Following this approach, the implementation of the improved Downstream Tracking may be started with a simple benchmark baseline model, such as logistic regression. These initial linear models are valued for their interpretability and help set a comparison standard for more advanced methods. The process then explores more sophisticated classifiers to improve performance further. The goal is to optimise the model to balance computational efficiency and accuracy in track reconstruction. Ultimately, a classifier is selected and fine-tuned to outperform the baseline in all metrics and be suitable for deployment with the Gaudi framework. Throughout, model performance is measured through training and validation cycles using Key Performance Indicators (KPIs) relevant to track reconstruction: efficiency, algorithm throughput, ghost ratio, etc.

3 Introduction to the SciFi Track Data

The classifier is trained on datasets with over one million SciFi track segments. These approximately correspond to about 50k simulated events of decay samples K_s^0 using the *Moore* framework. The events are produced in the MC simulation environment, providing accurate information about the track segments and producing ground-truth information regarding the track type: a True or Ghost Track. This labelling is crucial for supervised learning approaches. Within the data set used for training, the ratio of Ghost Tracks to true tracks is significantly high, while the Ghost Track ratio is relatively small compared to the true track as shown in

Figure [21]. The true tracks produced in the detector are consistent and more as compared to the combinatorial track signals that are produced due to random signals in the detector. This combinatorial tracks poses significant challenges in track reconstruction.

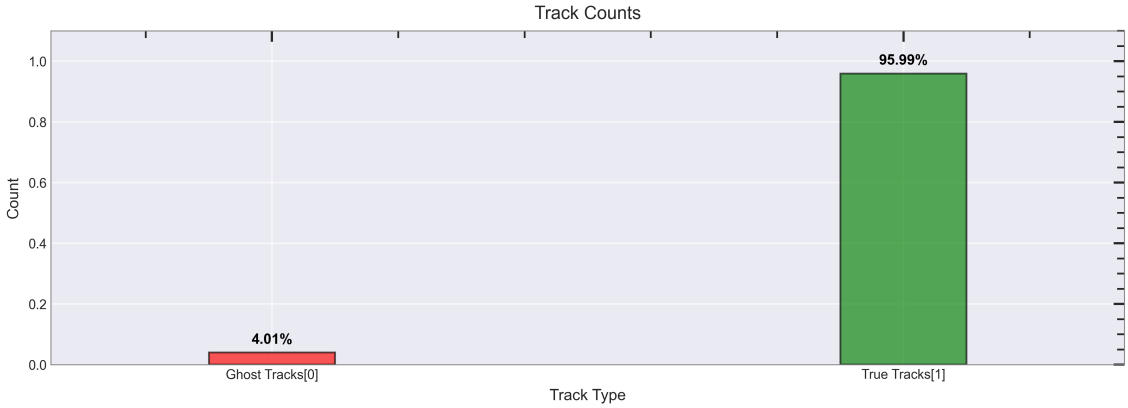


Figure 21: SciFi Track Types Ratio

Model training and evaluation is done with the event in Table [2] is considered:

Description	Notes
Decay	$K_S^0 \to \pi^+\pi^-$
ProductionID	00230966
File Type	SIM
Event Type	30000000 [MinBias]
BKCondition	Beam6800GeV-2024.Q1.2-MagDown-Nu5.7-25ns-Pythia8
BKPath	['MC/2024/Beam6800GeV-2024.Q1.2-MagDown-Nu5.7-25ns-Pythia8///30000000/']
DDDb	dddb-20240427
CondDb	sim10-2024.Q1.2-v1.1-md100
ConfigVersion	2024

Table 2: SciFi Track Event Metadata

4 Feature Space

The model is constructed and trained using eight key variables, which can be broadly categorised into kinematical, geometric, and statistical parameters. Together, these variables capture and provide complementary aspects of the track characteristics.

Figure [22] represents the training data distribution based on their quantiles. It helps in getting a clear idea regarding the distribution and possible outliers.

SciFi Hits: The number of hits that construct the seed track.

Tx: The trajectory slope in the X–Z plane.

Ty: The trajectory slope in the Y–Z plane.

X-Position: The X-coordinate of the seed position.

Y-Position: The Y-coordinate of the seed position.

Pseudo-rapidity (η): An angular variable that describes the angle of the track relative to the beam axis.

Phi (ϕ): The azimuthal angle of the particle around the beam line.

Chi2PerDoF: A statistical measure that assesses how well a model fits the observed data. It is normalized by degrees of freedom.

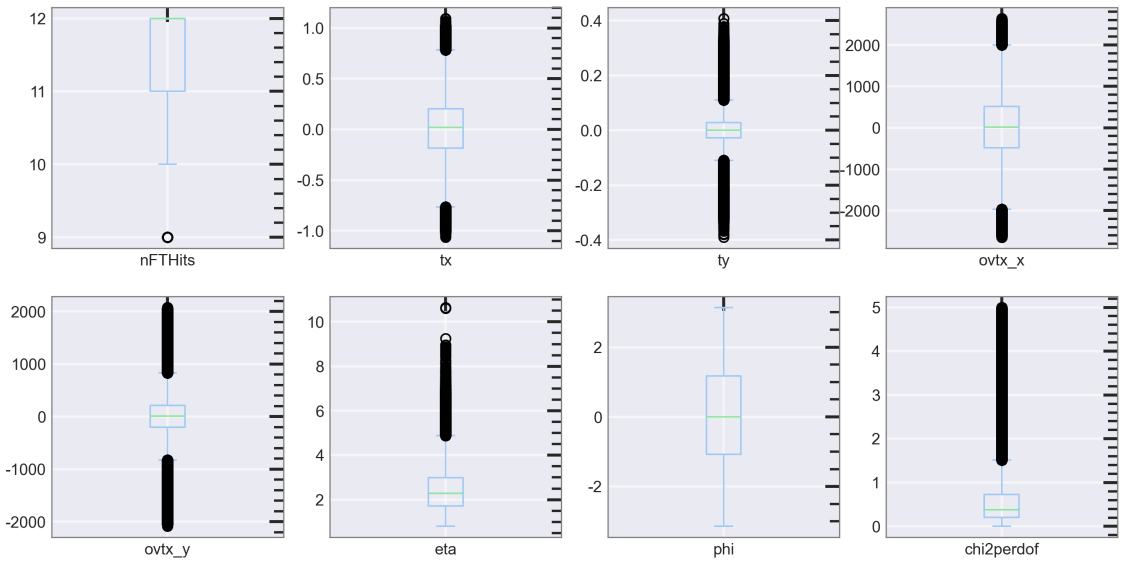


Figure 22: Boxplot: Variables used for SciFi Track Classifier

4.1 Spatial Distribution of Track Hits

Figure [23] represents the spatial distribution of SciFi track hits in the detector plane. It helps visualise the SciFi tracks' detector coverage and hit density. It can give hints about regions that are prone to combinatorial Ghost Tracks. The *isMatched* represents the true hits and Ghost Tracks, marking the region more susceptible to combinatorial Ghost Tracks.

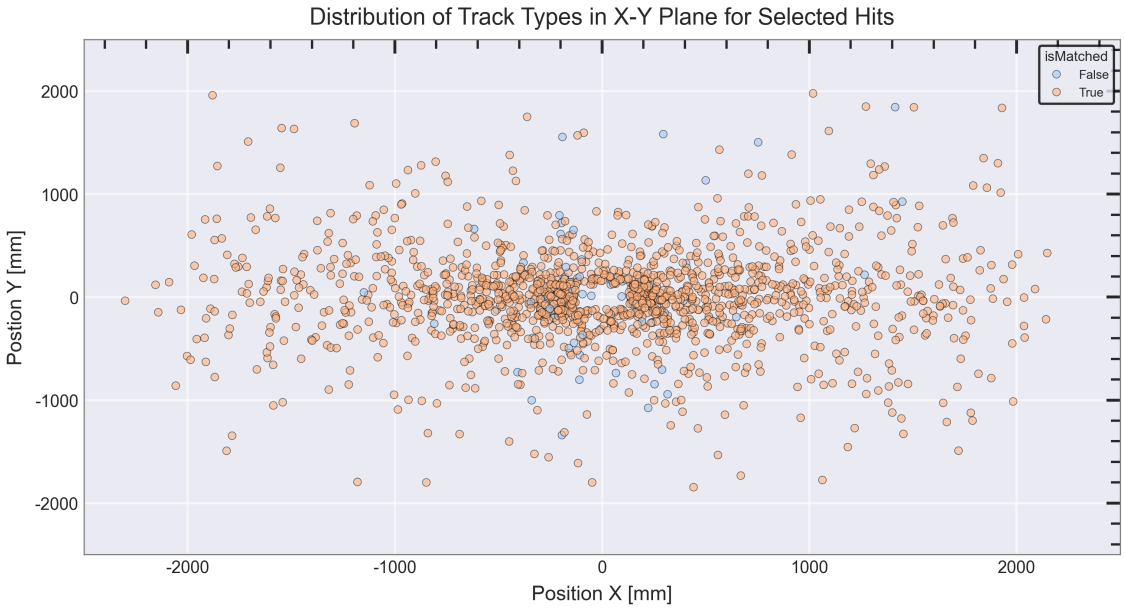


Figure 23: SciFi Hits Spatial Distribution

4.2 Momentum Distribution of SciFi Tracks

The Figure [24] displays the momentum component of the SciFi tracks in the transverse plane. Even though SciFi Tracker does not measure absolute momentum directly, it is possible to compute Px and Py proxies from the angle of tracks. It helps assess the coverage of track momenta in the dataset and identify the regions of high or low density.

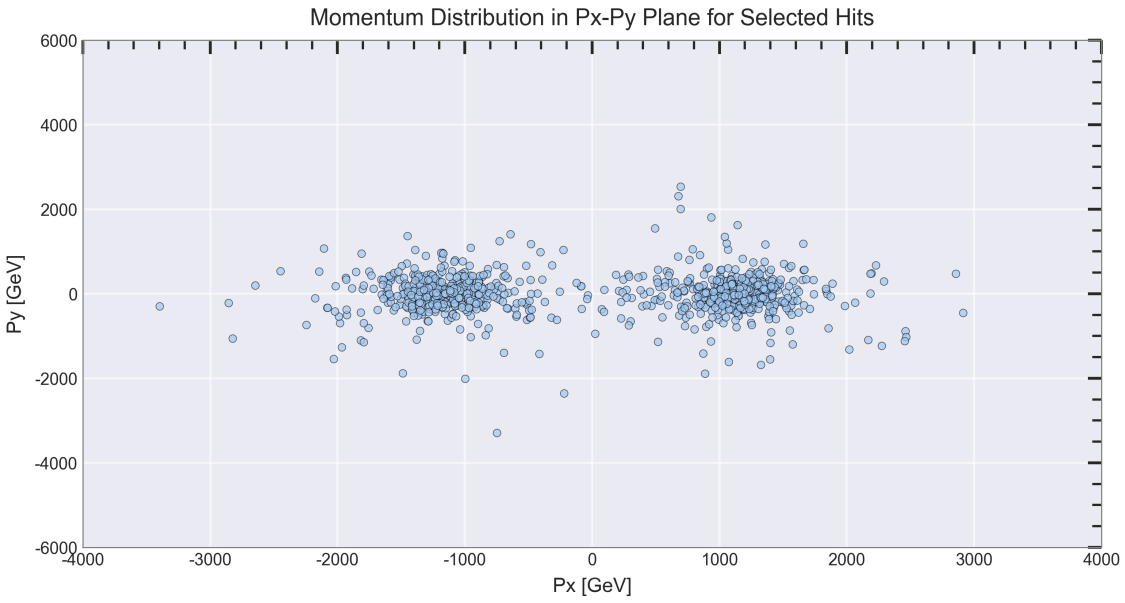


Figure 24: SciFi Track Momentum Distribution

5 Linear Model - Logistic Regression

Linear models are frequently used as a starting point in developing classification algorithms due to their simplicity and interpretability. The coefficients in a linear model directly quantify the contribution of each input to the final prediction. However, linear models have significant limitations: they cannot capture non-linear relationships or complex intrinsic patterns within the data.

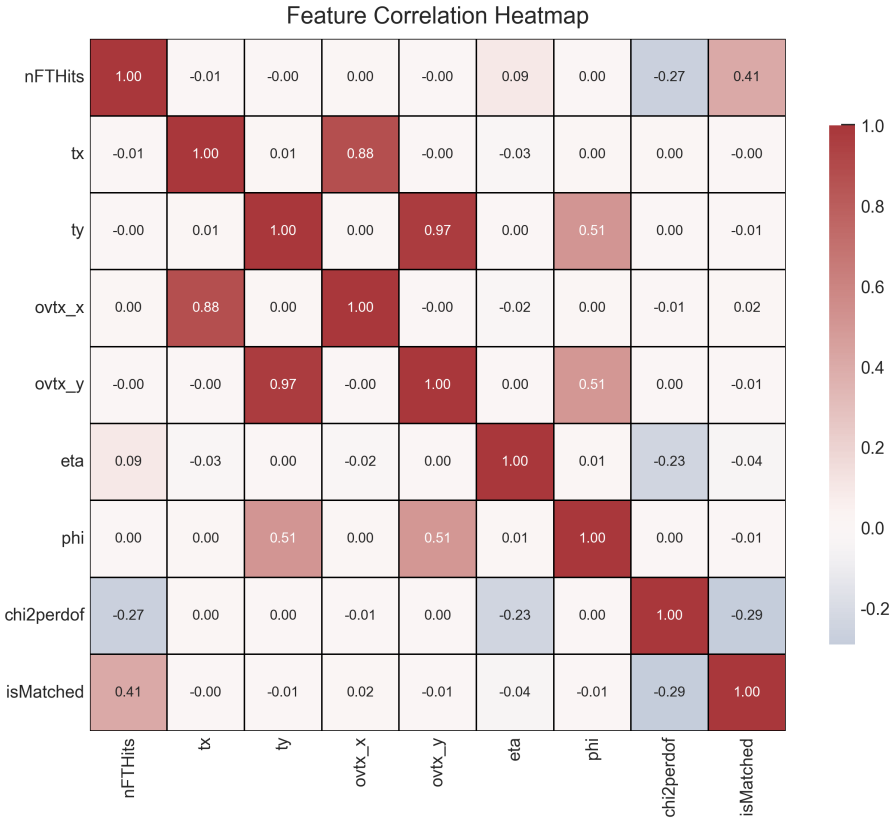


Figure 25: Pearson Correlation Matrix of SciFi Track Data

Such patterns are often crucial for distinguishing true tracks from Ghost Tracks in this context. Given these setbacks, the Logistic Regression Model, a linear model, is employed here as a baseline to benchmark more advanced non-linear models. The Pearson correlation coefficient is used as a metric to understand the relationships among variables. Figure [25] displays the Pearson Correlation Coefficient of the feature space, providing information on their linear relationships. Some variables show significant dependencies among the track variables from the heat map.

5.1 Model Evaluation

Model evaluation is a critical step in the workflow, providing a quantitative assessment and insight into the classifier's performance. Understanding how the trained model performs on unseen track data is essential. In this section, we discuss several key evaluation matrices used to assess the performance of the logistic regression model. In the case of differentiating between true tracks and Ghost Tracks, the prediction of the model will be calculated using a sigmoid function to evaluate the probability. By default, the model predictions will be based on the 50% threshold; any probability value falling below the threshold will be considered a Ghost Track, and the values above the threshold will be treated as true tracks.

5.1.1 ROC Curve

The ROC curve is the graphical tool discussed earlier that helps to evaluate the discriminating power of a classifier across different decision thresholds. The following ROC curve [26] illustrates the model performance for varying thresholds. The AUC score for model performance shows 95%, making this a good model for classifying the track segments in SciFi.

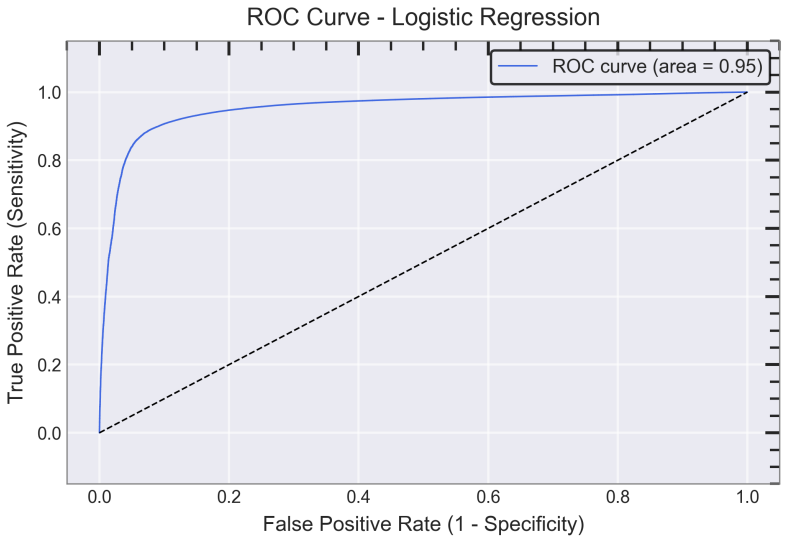


Figure 26: Logit SciFi Selector: ROC and AUC

5.1.2 Confusion Matrix

The confusion matrix illustrates the classification results of the model. Each track is classified according to the predicted probability predicted by the model based on the threshold probability. Figure [27] represents the model classification with a threshold of 50%, showing a clear distinction between tracks.

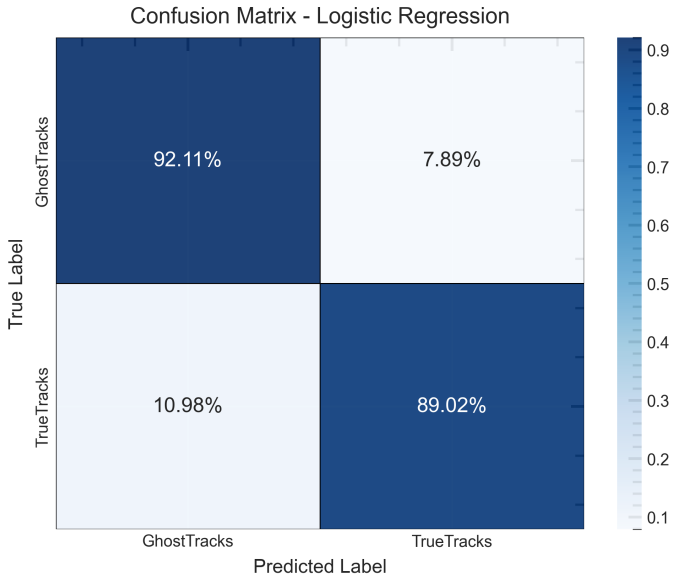


Figure 27: Logit SciFi Selector: Confusion Matrix

5.1.3 Weight Co-efficients

Feature importance measures the contribution of each input variable to the model's output. Not all features have equal impact, so evaluating their importance helps identify the most significant variable. In linear models, the magnitude of a feature's coefficient reflects its influence on model decisions. Figure [28] shows the variables used for model training and their coefficients. In this case, the number of SciFi hits has the highest influence on the model prediction, followed by χ^2/dof.

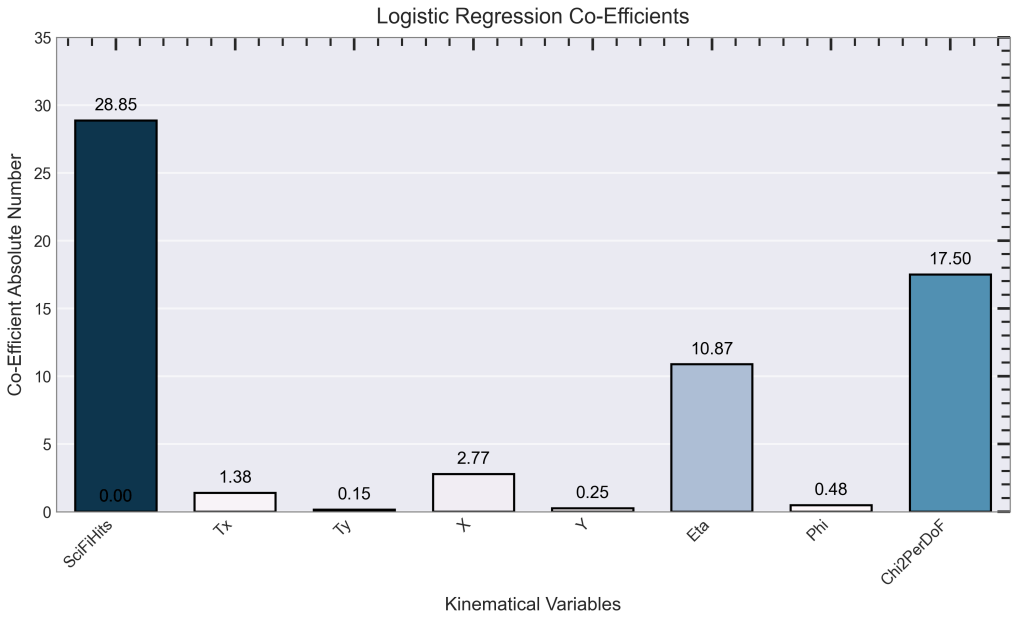


Figure 28: Logit SciFi Selector: Feature Importance

5.1.4 Model Response

The model response histogram, also known as the probability distribution plot, visualises the predicted probability distribution of each class as output by the classifier and the known track type. Two datasets are used to evaluate the impact of data imbalance: one raw and unprocessed dataset, and one processed and upsampled to an equal proportion of each track type. The following figure [29] illustrates how the model's predictions align with the known track types from the raw, unsampled dataset. The figure [30] represents the upsampled data for better visualisation. In either of the cases, the separation is very distinct, making it a good classifier.

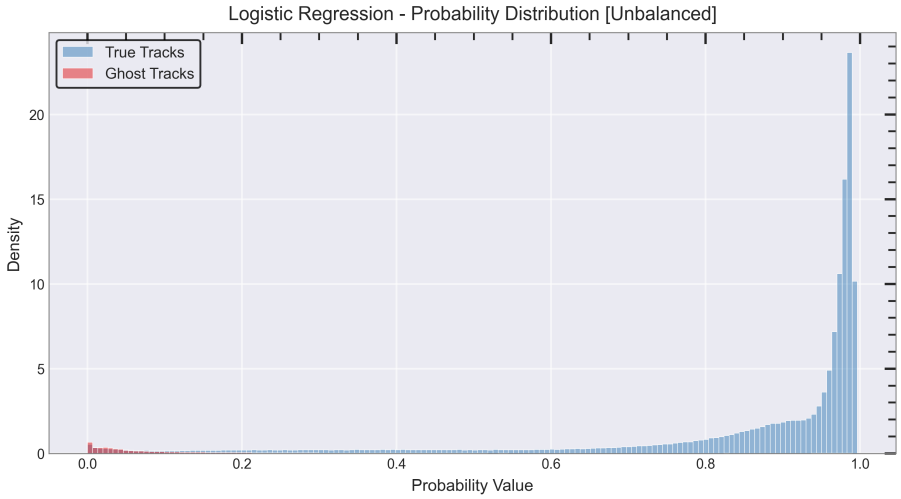


Figure 29: Logit SciFi Selector: Model Response (Unbalanced)

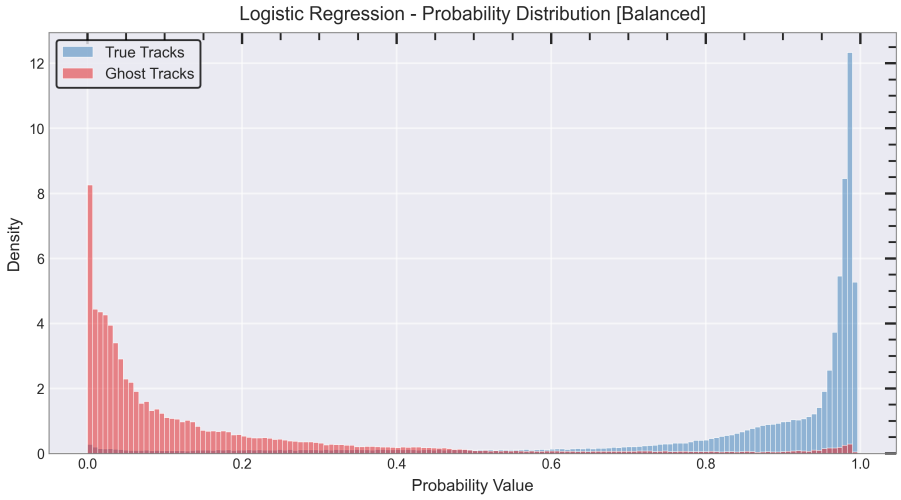


Figure 30: Logit SciFi Selector: Model Response (Balanced)

5.2 Results and Interpretation

As the logistic regression model provides direct mapping from the input variables to the probability of the type of track, the following Table [3] represents the weighted input for the predictions of the true tracks model, and Table [4] represents the known Ghost Tracks.

Table 3: True SciFi Track Weighted Inputs

Features	Coefficient (β)	True Track (Raw)	True Track (Scaled)	Weighted Input
nFTHits	1.687	12	0.707	1.194
tx	-0.081	-0.268	-0.884	0.071
ty	-0.009	0.021	0.336	-0.003
ovtx_x	0.162	-1055.846	-1.293	-0.21
ovtx_y	-0.014	173.193	0.342	-0.005
eta	-0.636	2.024	-0.442	0.281
phi	-0.028	3.062	1.496	-0.042
chi2/ndof	-1.023	0.294	-0.487	0.489

Table 4: Ghost Track Weighted Inputs

Features	Coefficient (β)	Ghost Track (Raw)	Ghost Track (Scaled)	Weighted Input
nFTHits	1.687	10	-2.017	-3.402
tx	-0.081	-0.306	-1.004	0.081
ty	-0.009	0.195	3.099	-0.027
ovtx_x	0.162	348.226	0.405	0.066
ovtx_y	-0.014	1640.88	3.285	-0.047
eta	-0.636	1.74	-0.78	0.496
phi	-0.028	2.574	1.257	-0.035
chi2/ndof	-1.023	3.153	4.893	-5.008

$$\text{logit}(p) = \beta_0 + \beta_1 X_1 + \beta_2 X_2 + \cdots + \beta_k X_k \tag{18}$$

Table 5: Decision: SciFi Track Types Decision Summary

Metric	True Track	Ghost Track
Intercept (β_0)	2.349	2.439
Sum (Σ)	1.785	-7.877
Y-value ($\Sigma + \beta_0$)	4.134	-5.528
Probability (p)	0.984	0.004

Table [5] summarises the coefficients obtained and their probability calculation for true and Ghost Tracks. In this case, a higher probability above 50% can indicate a higher probability that the track is a true track, and a probability below the threshold will be classified as a Ghost Track.

5.2.1 SHAP Analysis for Logit Model

While coefficients measure the average effect of each variable across the entire dataset, SHAP (Shapley Additive exPlanations) provides a more detailed examination of each variable's impact on individual predictions. The plots [31] and [32] show each feature's full distribution of SHAP values. This visualises the contribution of each variable to the prediction changes for different data points. It helps identify the importance of variables and the direction of their effects on probability predictions. The bar plot displays the absolute SHAP values for each feature. This lets features be ranked by the overall importance of the feature in model prediction. Here, similar to the linear coefficient, the number of hits in the SciFi tracker and the χ^2/dof show the variables most influenced.

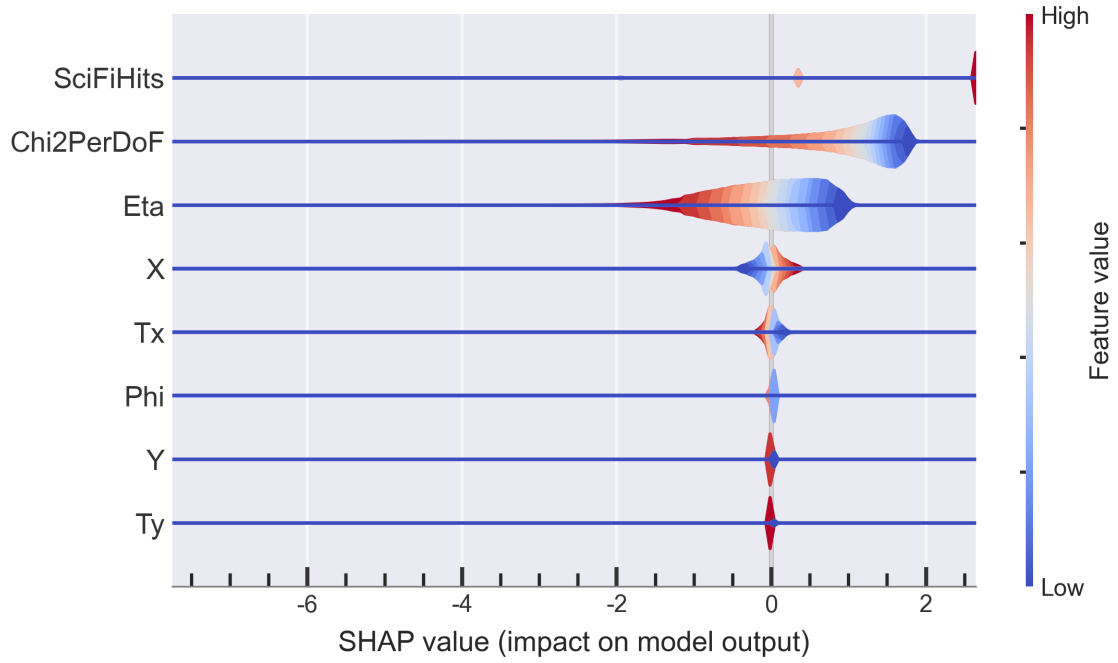


Figure 31: Logit Model SHAP Value Distribution

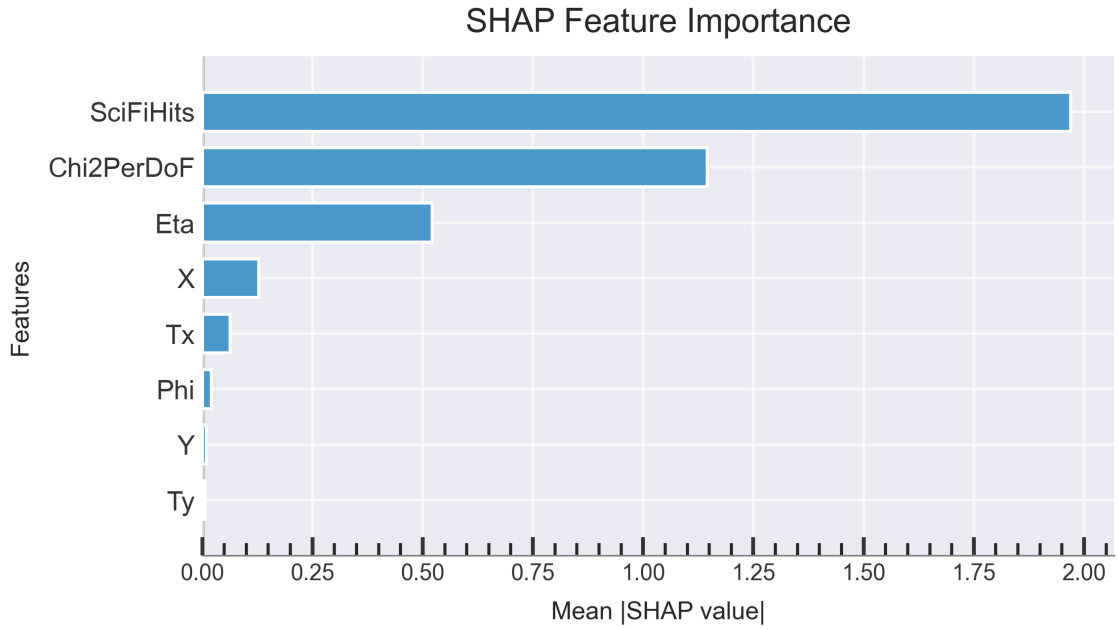


Figure 32: Logit Model Average Shap Values

6 Model Comparison

Once the baseline benchmark is established with the linear model, the next step is to explore other popular models to assess their performance on the same dataset under similar conditions. This phase considers other non-linear models and the baseline benchmark for a fair comparison.

The ROC curve and AUC scores are selected as evaluation metrics to ensure a consistent and fair model comparison. These metrics are robust to the class imbalance in the track types, supporting the comparative analysis. Table [6] shows the models that are considered and the corresponding scores, and the bar chart illustrated in Figure [33].

Table 6: ML Model Benchmark for SciFi Selector

Model Names	ROC [%]	F1 [%]
Logistic	95.29	93.94
DecisionTree	86.65	97.40
RandomForest	97.09	98.36
GradientBoosting	97.35	96.77
AdaBoost	96.58	94.87
KNN	92.92	97.16
MLP	97.29	97.43
Catboost	97.10	97.81

Based on iterative testing and evaluation, Catboost by Yandex achieved superior performance compared to alternative models **ProkhorenkovaCatboost2017**. Its scalability, GPU support, and deployment APIs made Catboost the preferred choice for the SciFi track classifier.

7 Catboost Model

Analysis of various Machine Learning Algorithms shows that Boosting Algorithms, especially Catboost, consistently perform better with complex datasets. These algorithms are called ensemble methods as they combine several weaker models to form a more accurate and robust model. Boosting algorithms work iteratively, with each new model fixing the errors of the previous one and leading to improvement.

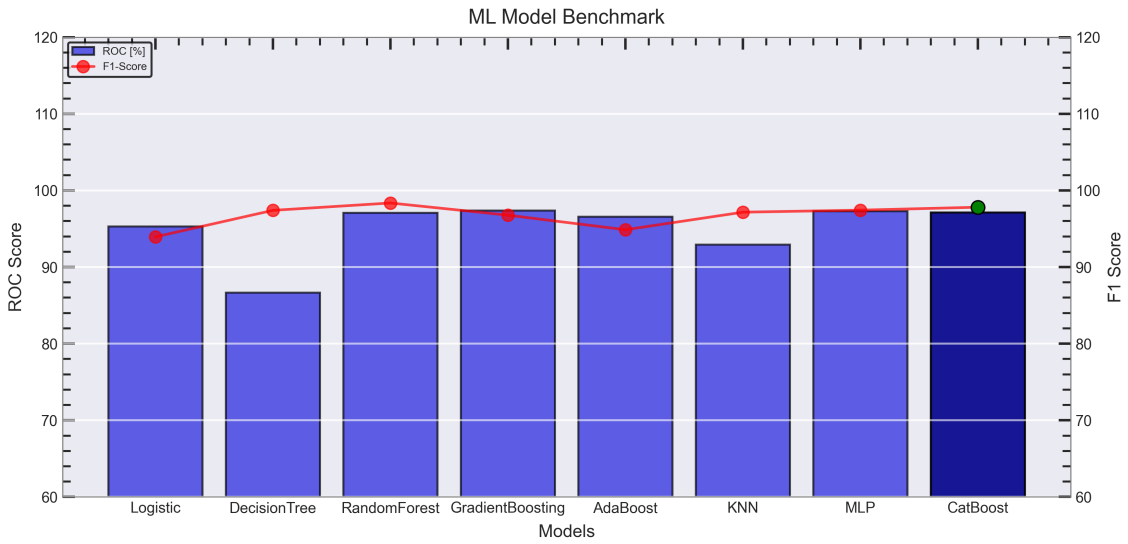


Figure 33: SciFi Selector: Model Comparison Chart

Building on these strengths, Catboost utilises gradient and ordered enhancement to improve prediction accuracy and minimise overfitting. It handles categorical variables well and supports GPU training, ONNX integration, etc. These features make Catboost a top choice for the SciFi track classifier, in addition to improved model evaluation metrics.

7.1 Hyper Parameter Tuning and Optimisation

Every machine learning model depends on its hyperparameters, which define it. Imagine them as the tuning knobs on the radio. These settings cannot be learned from the data used for training. In general, they can be derived from various variables, such as the depth of trees, the number of iterations, and the learning rate. Fine-tuning the model with the optimal hyperparameters is crucial, as poor choices can significantly impact the model's performance.

7.1.1 Optuna

The traditional search for the best parameter values usually involves searching in different regions of the feature space, but it is often inefficient and costly. Optuna is an open-source tool that addresses this inefficiency using Bayesian Optimisation and Tree-structured Parzen Estimators (TPE). It quickly identifies and converges promising hyperparameter regions, delivering faster and better performance. Building on these capabilities, Optuna employs a trial mechanism in which each trial involves training the model with a distinct set of hyperparameters and evaluating its performance on a validation data set. This feedback loop suggests that improved

hyperparameters should be used in each iteration. After a thorough hyperparameter search for the model using Optuna, the parameters chosen for the final model are as in the Table [7].

Parameter	Description	Value
loss_function	Objective function to optimize	Logloss
eval_metric	Evaluation metric used for validation	AUC
task_type	Execution mode (CPU/GPU)	GPU
use_best_model	Whether to use the best iteration on validation	True
learning_rate	Step size shrinkage to prevent overfitting	0.08
iterations	number of boosting iterations (trees)	945
depth	Depth of individual trees	6
l2_leaf_reg	L2 regularization coefficient	7.85
min_data_in_leaf	Minimum samples required in a leaf	69
bagging_temperature	Controls sampling randomness for bagging	0.95
border_count	number of splits for numerical features	107

Table 7: SciFi Track Catboost Model Hyper-parameters

7.2 Model Evaluation

The following section presents a similar analysis as the linear model benchmark, with a fine-tuned Catboost model as the SciFi track classifier and its performance.

7.2.1 ROC Curve

Regarding the ROC curve and the AUC score, the Catboost models outperform linear models by a significant margin. This figure [34]clearly represents the effect of working point optimisation and how the non-linear models learn compared to the linear models discussed earlier in the chapter.

7.2.2 Confusion Matrix

As with previous confusion matrices for linear models, the following Confusion Matrix illustrate in Figure [35] shows the model's performance on unseen data. In this case, two working points are assessed to learn the impact of changing the threshold for track type identification. Reducing the cut-off point from 50% to 40% keeps the maximum number of true tracks while minimising Ghost Tracks.

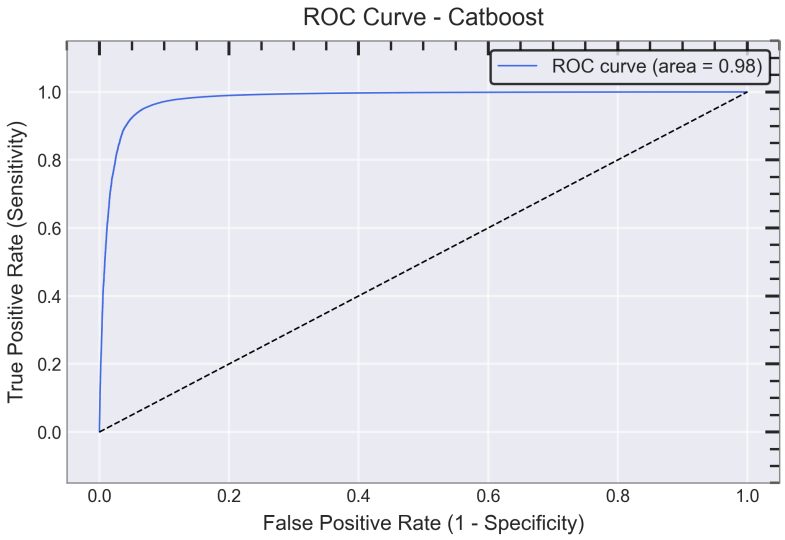
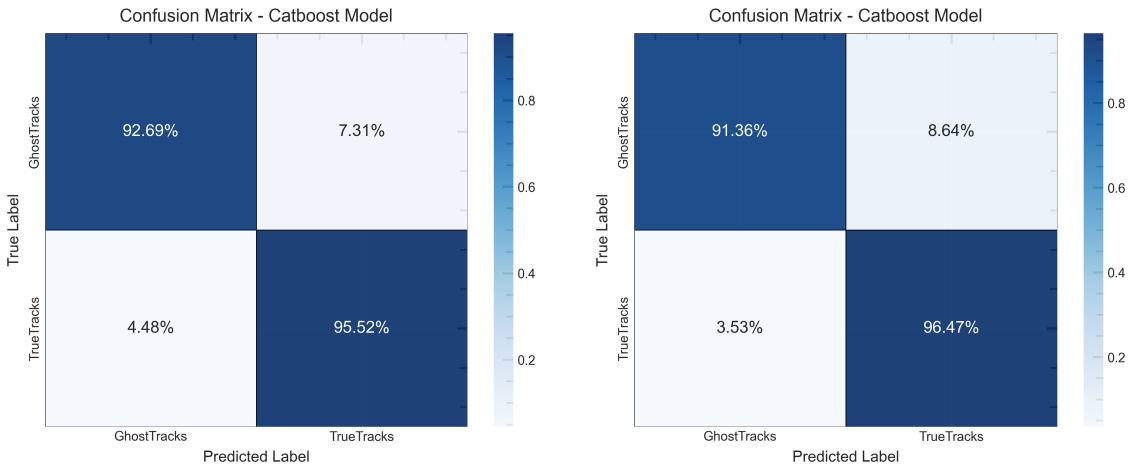


Figure 34: Catboost SciFi Selector: ROC Curve and AUC



(a) With 50% Threshold

(b) With 40% Threshold

Figure 35: Catboost SciFi Selector: Confusion Matrix

7.2.3 Feature Importance

In the case of the influence of variables on model predictions, SciFi hits still show high decision power compared to the other variables. The bar graph showed in Figure [36] shows this expected behaviour, considering that the quality of the track sample is higher with the higher number of hits in the SciFi tracker.

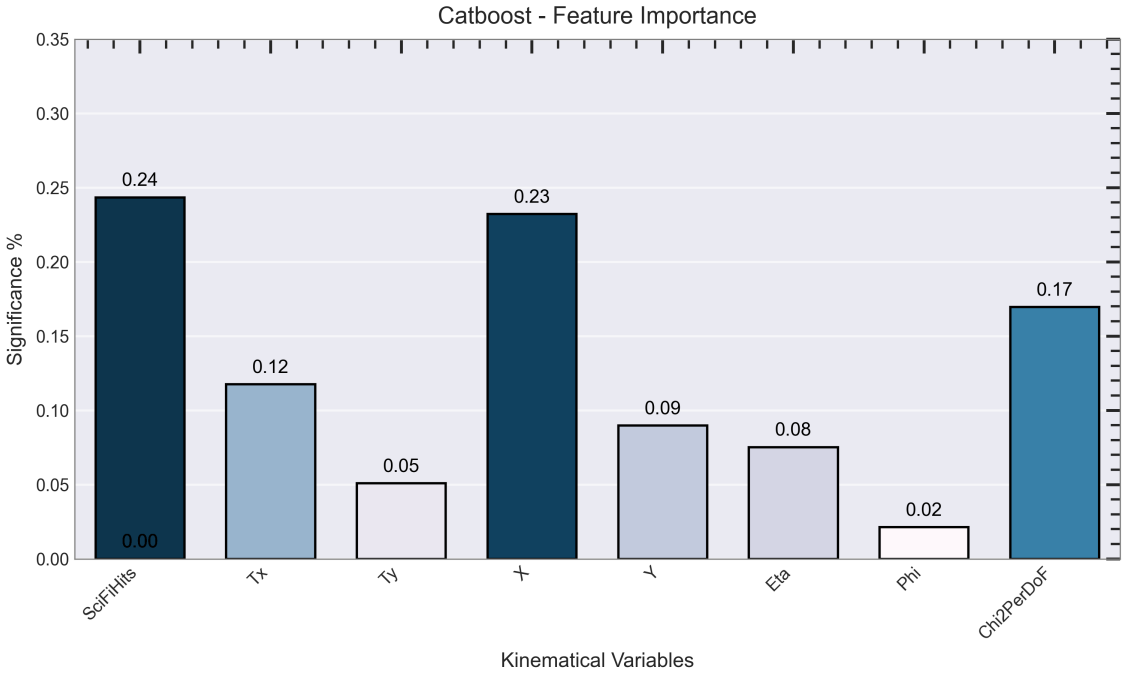


Figure 36: Catboost SciFi Selector: Feature Importance

7.2.4 Model Response

The separation between the track types was good in the case of the linear model for the SciFi classifier. In the case of Catboost, a non-linear model, it can learn more intrinsic patterns within the tracks and their patterns. Figure [37] represents the upsampled track data with inherent imbalance, and figure [38] displays the upsampled balanced data. It is essential to note that the imbalanced data will be evaluated at runtime; the balanced dataset reveals that the model prioritises the minority class, in this case, Ghost Tracks.

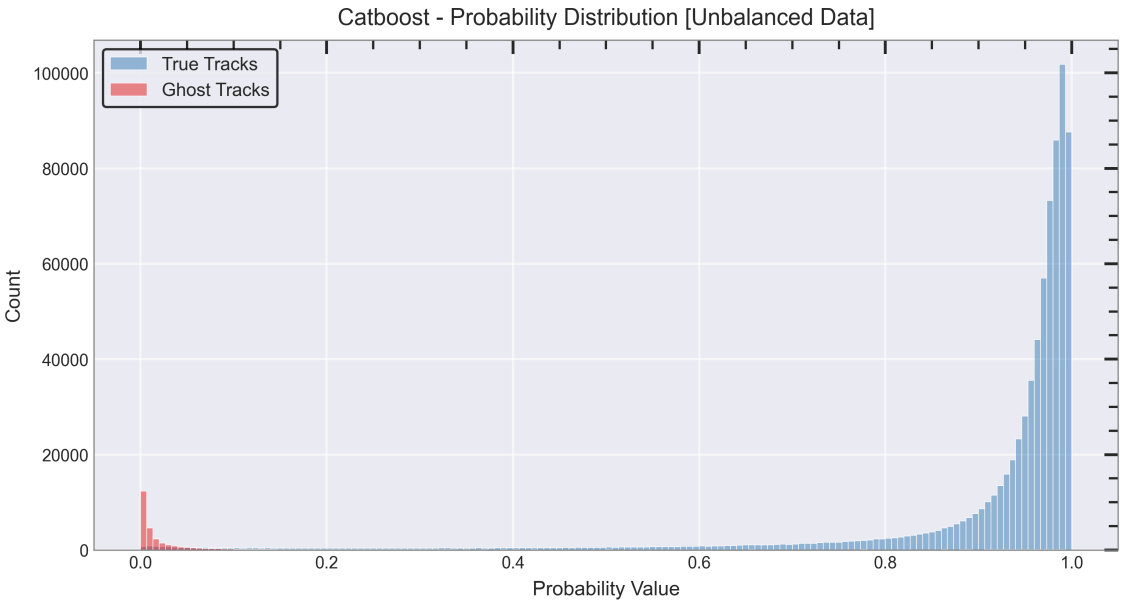


Figure 37: Catboost SciFi Selector: Model Response (Unbalanced)

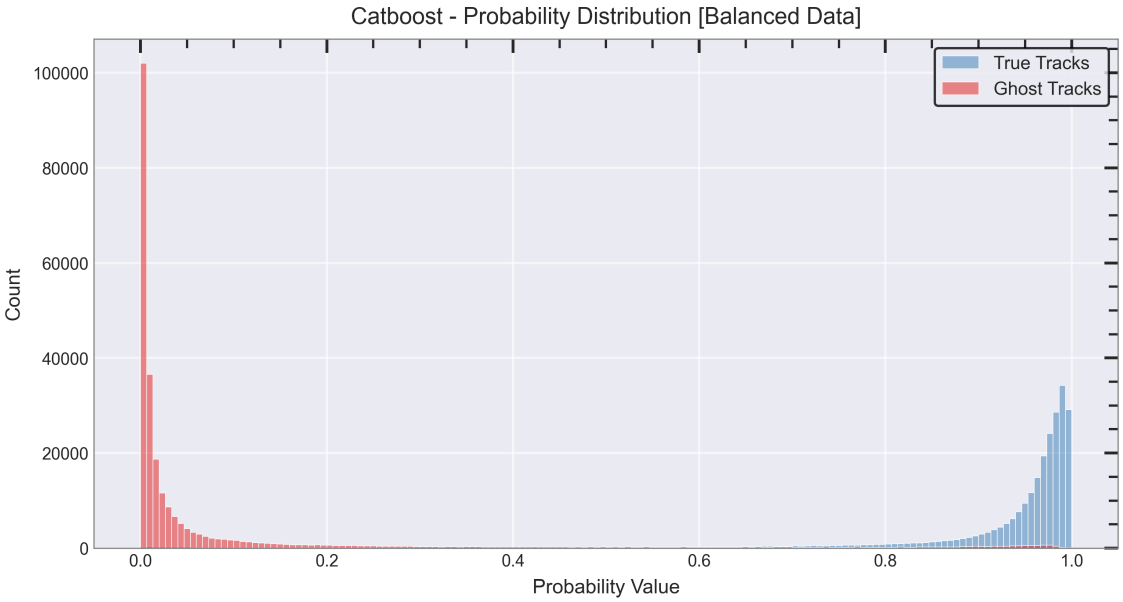


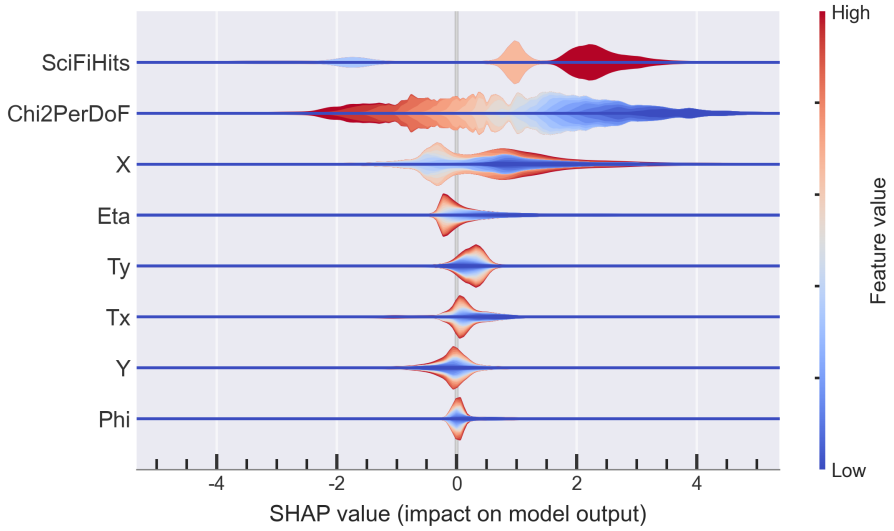
Figure 38: Catboost SciFi Selector: Model Response (Unbalanced)

7.3 Results and Interpretation

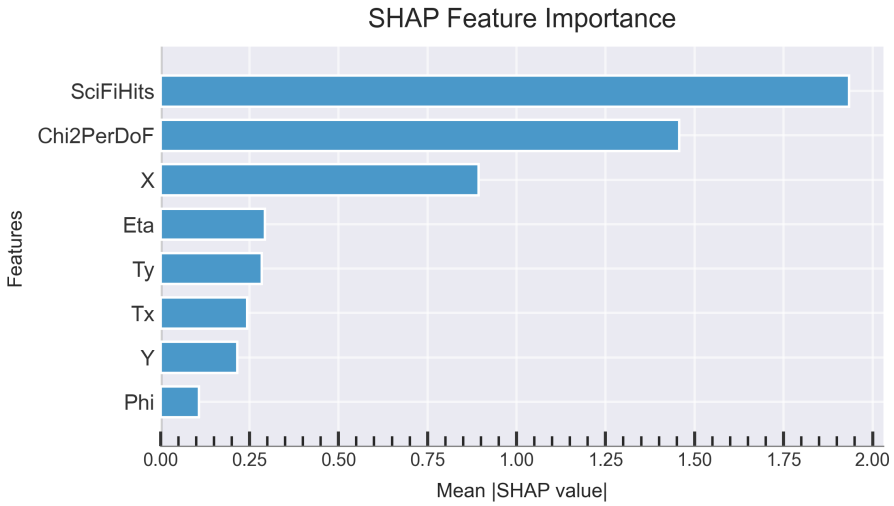
Interpreting advanced non-linear models is not easy. In this case, we rely on SHAP analysis to understand the variables' dependence on each other when calculating the model predictions.

7.3.1 SHAP Analysis for Catboost Model

Similar to the previous SHAP Analysis for Logit Model, the Figure [39] show case the Shapley Summary and the model variable dependencies in the case of Catboost Model. Here, similar to the case of feature importance plot, the number of SciFi hits shows the highest valuable variable in model training.



(a) SHAP Value Distribution



(b) Average Shap Values

Figure 39: Catboost Model SciFi Selector: Shap Analysis

7.4 Conclusion: SciFi Track Classifier

This concludes the model development for the first stage of the track selector. The SciFi (seed) tracks presented a significant class imbalance in the training dataset due to the nature of the problem, which posed a significant challenge for effective model learning. For the initial benchmark, logistic regression was employed and achieved an AUC score of approximately 95%, demonstrating baseline model performance and interpretability. However, a fine-tuned Catboost classifier was implemented to push the model performance further. This approach improved the AUC score beyond the baseline logistic regression and provided a more robust framework for handling the complex nonlinear relationships in the dataset.

The feature importance analysis revealed that the number of hits in the SciFi tracker strongly influences the model's decisions, highlighting its critical role in discriminating between track types. In conclusion, the SciFi track classification task for the first stage of the track selector is successfully finalised with the Catboost model providing a solid foundation for the subsequent stages of the track selection pipeline.

Chapter 5

Stage Two Selection: Selecting Downstream Tracks

1 Objectives

The previous chapter focused on purity optimisation of SciFi segments as Seed Tracks for Downstream reconstruction. The selection process has two stages: refining Seed Tracks and selecting the best Downstream Track by precisely rejecting ghost tracks, which is a computationally intensive task.

As discussed earlier, Downstream Tracks result from the decay of Long-lived Particles. These particles travel longer distances and do not leave signals on the VELO tracker. To address this, SciFi tracks are propagated upstream through the detector to find matching hits in the UT, allowing the reconstruction of the Downstream Track. This process presents challenges, particularly the risk of forming combinatorial track segments a.k.a. ghost tracks that are difficult to distinguish from true Downstream Tracks. The *Downstream Track Classifier* is the second and more complex stage. It uses a machine learning model to tag and select Downstream Tracks for improved reconstruction. The classifier aims to evaluate the likelihood of being a true track, accepting it, and rejecting as many ghost tracks as possible. This step is crucial in the real-time trigger system to minimise signal loss.

In summary, the second-stage selection acts as a strong, decisive filter for improving the quality of Downstream Track reconstruction. It is built upon the selected seed tracks from the Hybrid Seeding algorithm. Together with this two-stage strategy, the thesis explores approaches to progressively reducing the combinatorial background and improving the overall track quality of Downstream Tracks.

2 Methodology

The development of the second-stage selection follows a plan closely aligned with the first-stage SciFi track selection. In this case, similar methods are implemented to maintain consistency while improving performance across all metrics. The choice is motivated by the similarity in the underlying problems. Using a uniform approach is easier to explain and more straightforward to evaluate within the chapter.

As in the first stage, a linear model is initially built as a reference. Linear models are easier to interpret and help understand track selection factors and their influences. This model is a baseline for comparison with other models used for Downstream Track selection. Linear models are not expected to perform competitively due to the complexity of the problem. However, they help establish a point of reference and interpretability. Next, test other popular machine learning algorithms and frameworks. They are scored on two goals: to maximise the number of correctly identified true Downstream Tracks and to minimise the number of combinatorial ghost tracks. The final model selection will use a fine-tuned non-linear approach. It was chosen after comparing it with other models across all evaluation metrics and is based on seamless integration with the Gaudi framework.

3 Introduction to the Downstream Track Data

The models are trained on nearly one million Downstream Tracks produced with MC Truth using simulation data to maintain uniformity across model building and evaluation. The tracks for the model training were produced using the *Moore* Framework with 50k of simulated events. The ground truth of the track type is important for the supervised classification problem.

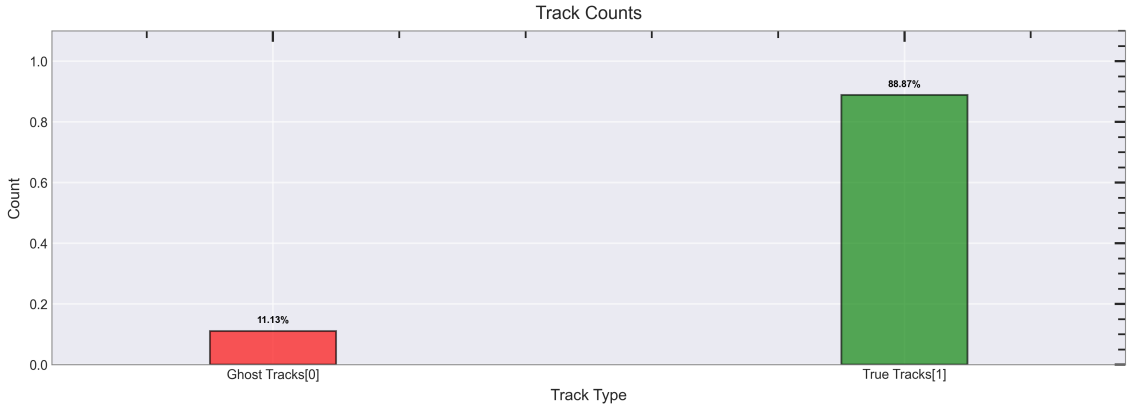


Figure 40: Downstream Track Types Ratio

Figure [40] illustrates the imbalance of track types in the training data, which complicates the identification of true tracks among ghost tracks. This arises from the complexity of Downstream Track reconstruction and the relative rarity of such events. For the training and evaluation of the ML models, the following event is considered:

Description	Notes
Decay	$K_S^0 \to \pi^+\pi^-$
ProductionID	00230966
File Type	SIM
Event Type	30000000 [MinBias]
BKCondition	Beam6800GeV-2024.Q1.2-MagDown-Nu5.7-25ns-Pythia8
BKPath	['MC/2024/Beam6800GeV-2024.Q1.2-MagDown-Nu5.7-25ns-Pythia8///30000000/']
DDDb	dddb-20240427
CondDb	sim10-2024.Q1.2-v1.1-md100
ConfigVersion	2024

Table 8: Downstream Track Event Metadata

4 Feature Space

The Downstream Track classifier is designed to work with 11 variables gathered from the SciFi and UT detectors. The variables sourced from SciFi and UT include kinematic and geometrical measurements and statistical values such as χ^2. Together, these features capture intrinsic patterns and statistical parameters from track trajectories. Figure [41] represents the distribution of the downstream training data, including its median, quantile, and potential outliers. This information helps to understand the nature of the data and improve the pre-processing of track information during model training.

P: Track Momentum.

Pt: Track Transverse Momentum

Tx: The trajectory slope in the X–Z plane.

Ty: The trajectory slope in the Y–Z plane.

X-Position: The X-coordinate of the track position.

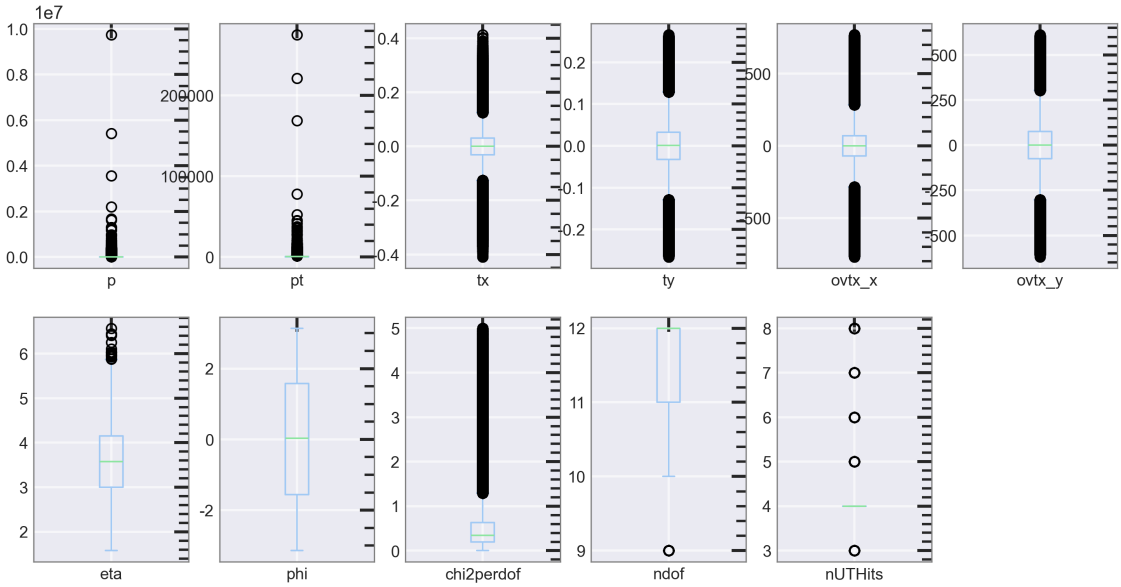


Figure 41: Boxplot : Downstream Training Data

Y-Position: The Y-coordinate of the track position.

UT Hits: The number of hits in Upstream Tracker.

SciFi Hits: The number of hits from the Seed Tracks.

Chi2PerDoF: A statistical track quality measure.

Pseudo-rapidity (η): Angular variable representing the angle of the track relative to the beam axis.

Phi (ϕ): The azimuthal angle of the track with respect to the beam line.

4.1 Spatial Distribution of Track Hits

Figure [42] represents the spatial distribution of the Downstream Tracks. It helps identify the regions that may be more prone to combinatorial tracks. Similar to the spatial distribution of SciFi tracks, *isMatched* represents the MC association of tracks with a particle. The figure shows a clear overlap of the tracks, making it difficult to differentiate between them.

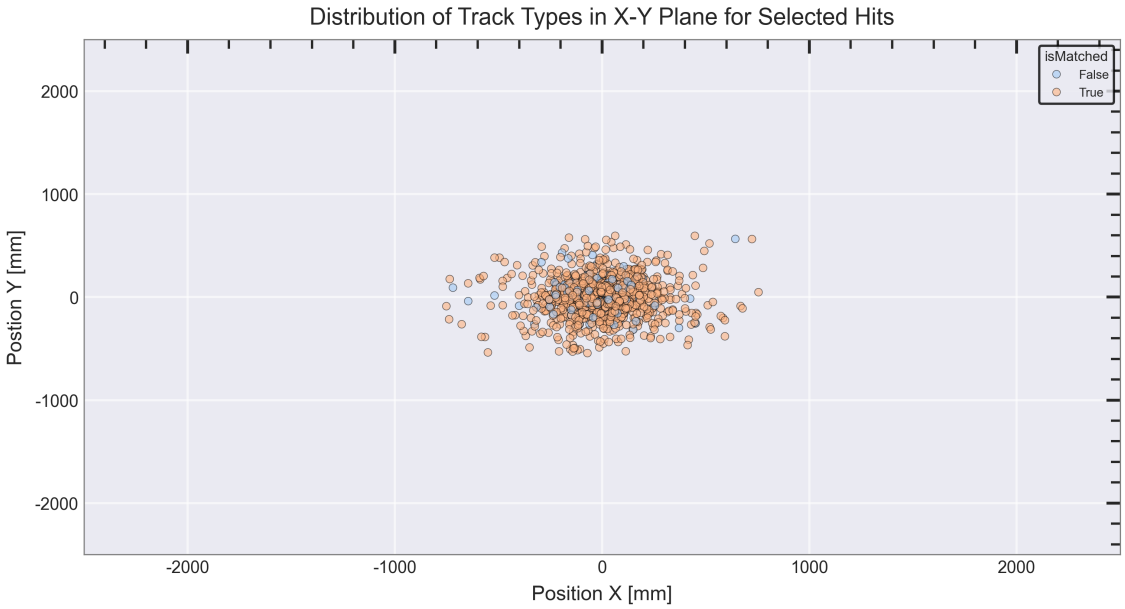


Figure 42: Downstream Hits Spatial Distribution

4.2 Momentum Distribution of SciFi Tracks

The Figure [43] illustrates the distribution of the momentum component of the Downstream Tracks in the transverse plane.

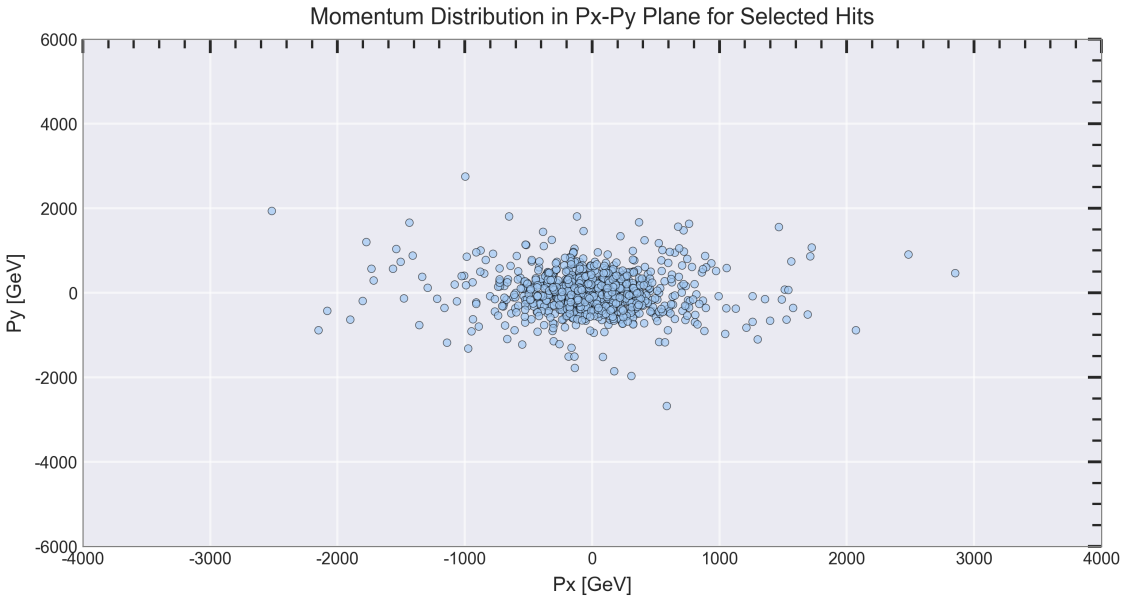


Figure 43: Downstream Track Momentum Distribution

This plot helps to evaluate the coverage of track momentum and its distribution, and highlights how the tracks are distributed across different momentum ranges.

5 Linear Model - Logistic Regression

In this section, *Logistic Regression* is used as the baseline benchmark, reflecting the strengths and limitations of linear models. To examine feature dependencies, Pearson's correlation was applied (Figure [44]), illustrating the linear relationships between variables.

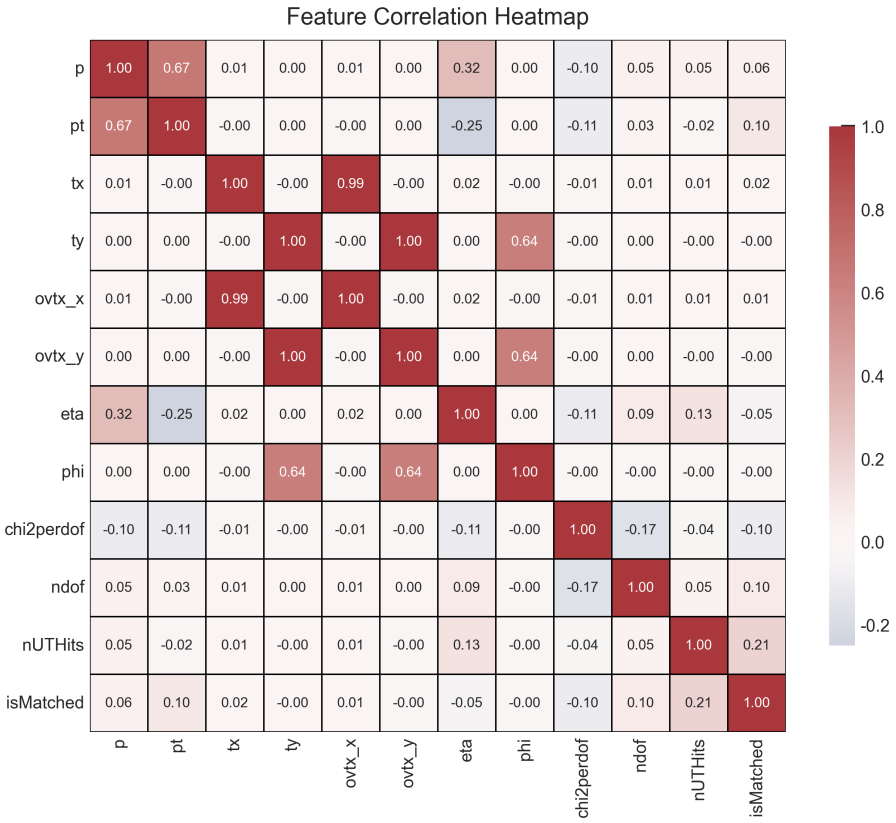


Figure 44: Pearson Correlation Matrix of Downstream Track Data

5.1 Model Evaluation

Model evaluation and understanding model outputs are easier for linear models than non-linear ones. Since similar evaluation principles were outlined in the previous chapter, the model evaluation follows a similar approach. The model uses the training data and produces outputs that can be translated to probabilities using the sigmoid function. The output probability allowed the likelihood that a track is true downstream to be quantified.

5.1.1 ROC Curve

The ROC curve [45] displays the model's performance for varying thresholds. According to the expectation that linear models cannot capture complex patterns, the ROC curve and the AUC score for the model's performance indicate relatively low performance. This requires implementing more sophisticated ML models.

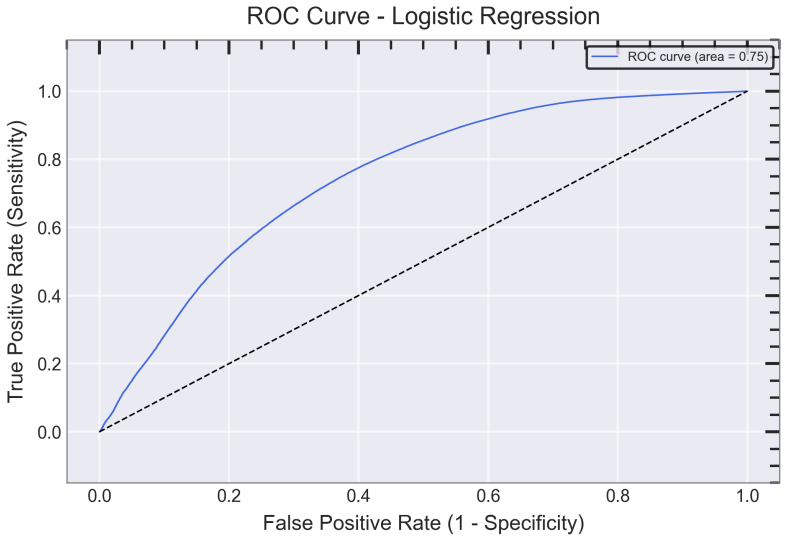


Figure 45: Logit Downstream Classifier: ROC and AUC

5.1.2 Confusion Matrix

The confusion matrix [46] shows the model's classification results. From the ROC, it is expected to have a poor-performing linear model to identify true Downstream Tracks. Confusion Matrics shows how they classify with the fixed threshold of 50%.

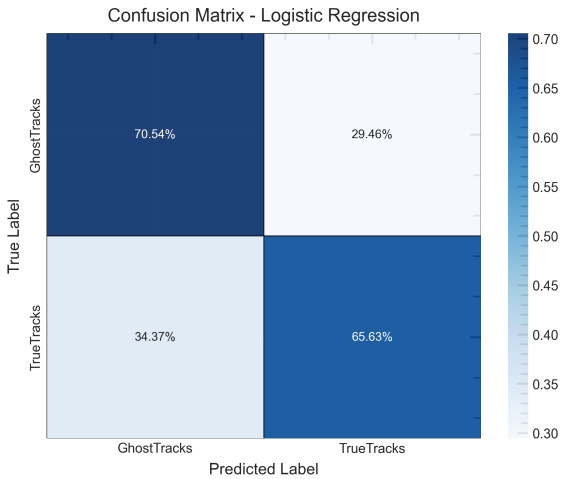


Figure 46: Logit Downstream Classifier: Confusion Matrix

5.1.3 Weight Co-efficients

Representing the linear coefficients for the linear model and illustrating their impact on the model predictions is relatively straightforward.

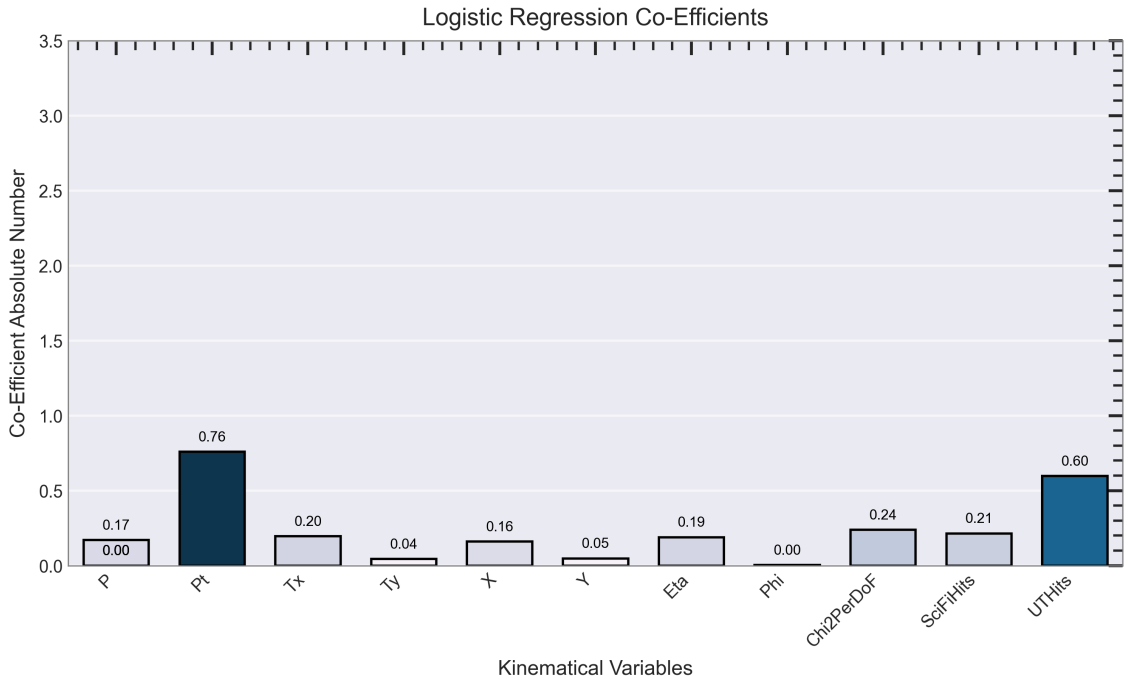


Figure 47: Logit Downstream Classifier: Feature Importance

Figure [47] shows the linear coefficients for each variable, with the transverse momentum and the number of UT hits having the most significant influence on the model predictions.

5.1.4 Model Response

The model response histogram illustrates how well predictions map track type from the feature space. As with earlier examples, it is crucial to examine how model predictions correspond with known track types and address the imbalance in track type distribution. To illustrate these points, figure [48] shows the raw unprocessed data, while [49] represents the upsampled data that has an equal number of true and ghost tracks. Examining these visualisations, we observe that the track types and their predictions overlap in both cases, causing the classifier to underperform and accurately distinguish between true and ghost tracks.

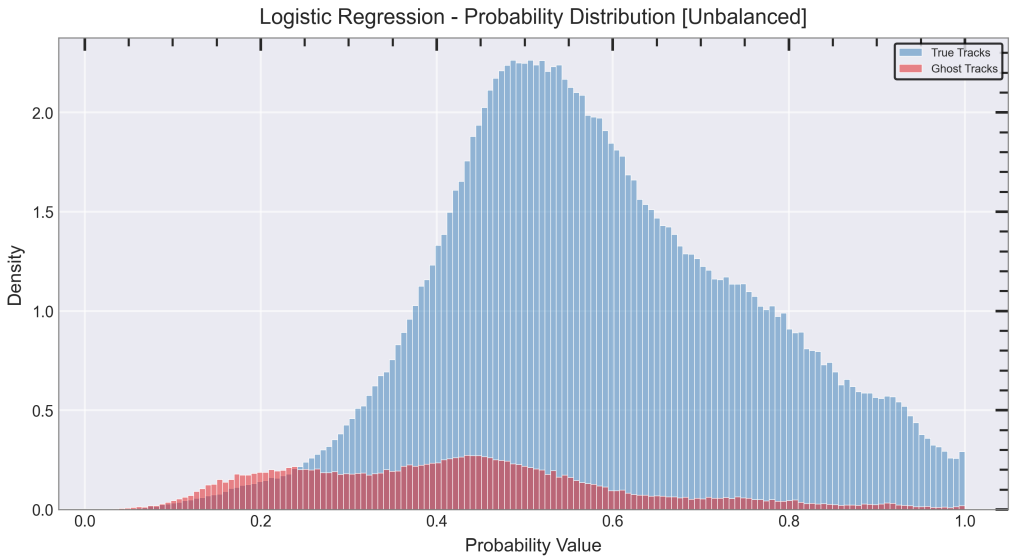


Figure 48: Logit Downstream Classifier: Model Response (Unbalanced)

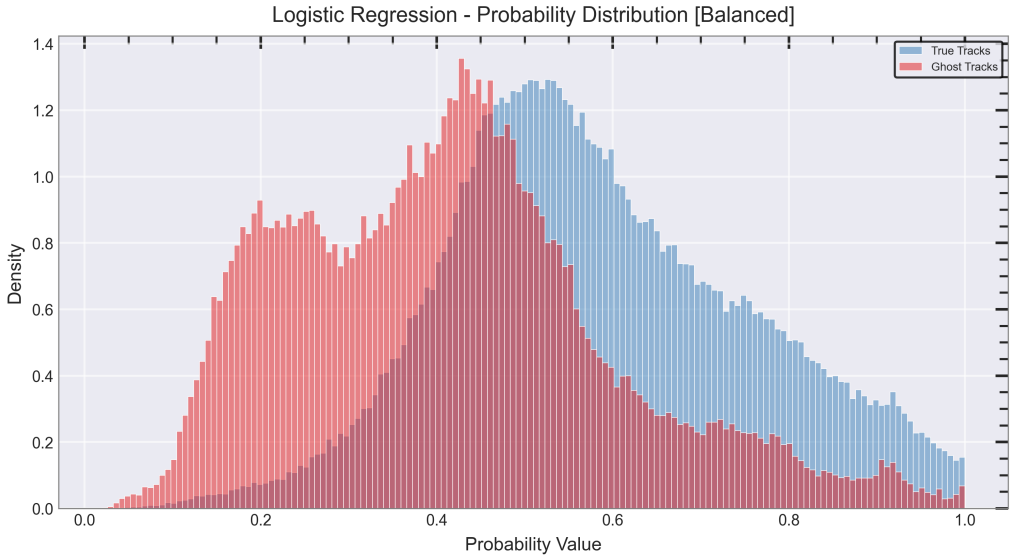


Figure 49: Logit Downstream Classifier: Model Response (Balanced)

5.2 Results and Interpretation

This section includes the weighted input evaluation and probability calculations to complete the model's interpretation. Both a true track and a ghost track are considered. Table [9] shows the weighted input for true Downstream Tracks. Table [10] presents the known ghost tracks of MC Truth. Table [11] summarises the probability calculation for each track type and the threshold dependency.

Table 9: True Downstream Track Weighted Inputs

Features	Coefficient (β)	True Track (Raw)	True Track (Scaled)	Weighted Input
p	-0.167	5909.818	-0.300	0.050
pt	0.742	428.278	-0.204	-0.151
tx	0.190	-0.069	-1.005	-0.191
ty	0.043	0.022	0.333	0.014
ovtx_x	-0.156	-154.245	-0.965	0.151
ovtx_y	-0.047	51.722	0.332	-0.016
eta	-0.185	3.316	-0.333	0.061
phi	-0.004	2.836	1.568	-0.006
chi2perdof	-0.233	0.294	-0.429	0.100
ndof	0.210	12.000	0.644	0.135
nUTHits	0.584	4.000	-0.172	-0.101

Table 10: Ghost Downstream Track Weighted Inputs

Features	Coefficient (β)	Ghost Track (Raw)	Ghost Track (Scaled)	Weighted Input
p	-0.167	26774.692	1.047	-0.175
pt	0.742	362.705	-0.313	-0.232
tx	0.190	-0.010	-0.138	-0.026
ty	0.043	0.009	0.135	0.006
ovtx_x	-0.156	-26.337	-0.163	0.025
ovtx_y	-0.047	24.775	0.158	-0.007
eta	-0.185	4.995	1.933	-0.357
phi	-0.004	2.418	1.336	-0.005
chi2perdof	-0.233	0.236	-0.561	0.131
ndof	0.210	12.000	0.644	0.135
nUTHits	0.584	3.000	-2.342	-1.367

$$\text{logit}(p) = \beta_0 + \beta_1 X_1 + \beta_2 X_2 + \cdots + \beta_k X_k \tag{19}$$

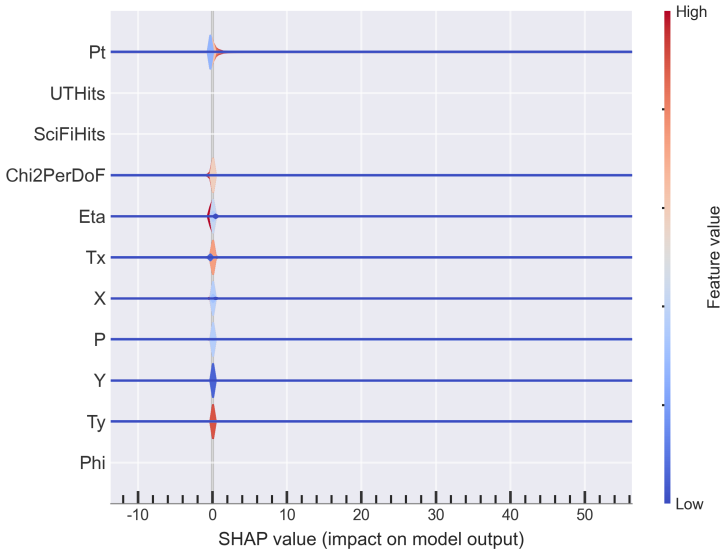
Table 11: Decision: Downstream Track Types Decision Summary

Metric	True Track	Ghost Track
Intercept (β_0)	0.338	0.338
Sum (Σ)	0.047	-1.874
Y-value ($\Sigma + \beta_0$)	0.385	-1.535
Probability (p)	0.595	0.177

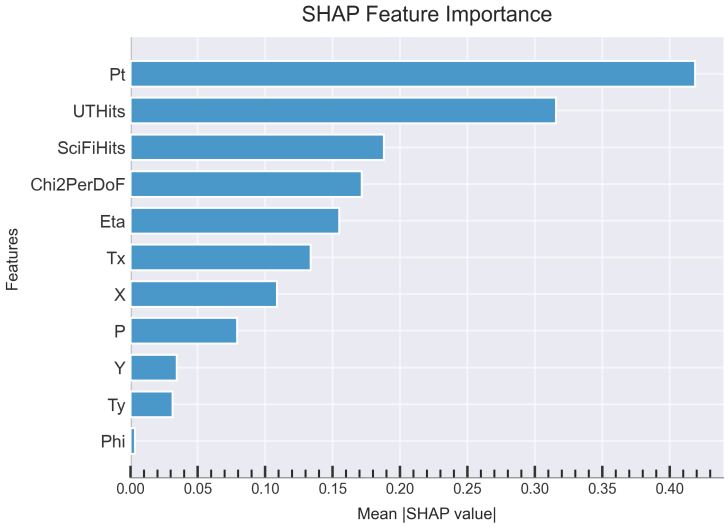
The probability output in this example shows a probability of 0.595 for a true track and a probability of 0.177 for a ghost track.

5.2.1 SHAP Analysis for Logit Model

The calculated probability values are still around the default threshold, which can influence the track classification. The following SHAP analysis can help to understand the importance of features and their directions for the linear model. The Figure [50] show each feature's full distribution of SHAP values, illustrating Transverse Momentum and Number of UT Hits as the most important features in the model output.



(a) SHAP Value Distribution



(b) Average Shap Values

Figure 50: Logit Model Downstream Selector: Shap Analysis

6 Model Comparison

The baseline model could not capture complex patterns within the data; the next logical step is to explore other ML algorithms to identify a better-performing model. To facilitate this comparison, the Table [**??**] presents several popular models alongside their ROC and F1 scores, which serve as primary evaluation metrics. This is followed by a bar graph [51], which visualises the performance of these models under similar conditions.

Table 12: ML Model Benchmark - Downstream Classifier

Model Names	ROC [%]	F1 [%]
Logistic	75.04	77.08
DecisionTree	66.48	86.88
RandomForest	81.20	92.08
GradientBoosting	80.11	84.08
AdaBoost	78.30	84.01
KNN	70.51	85.45
MLP	85.00	89.00
Catboost	85.72	90.48

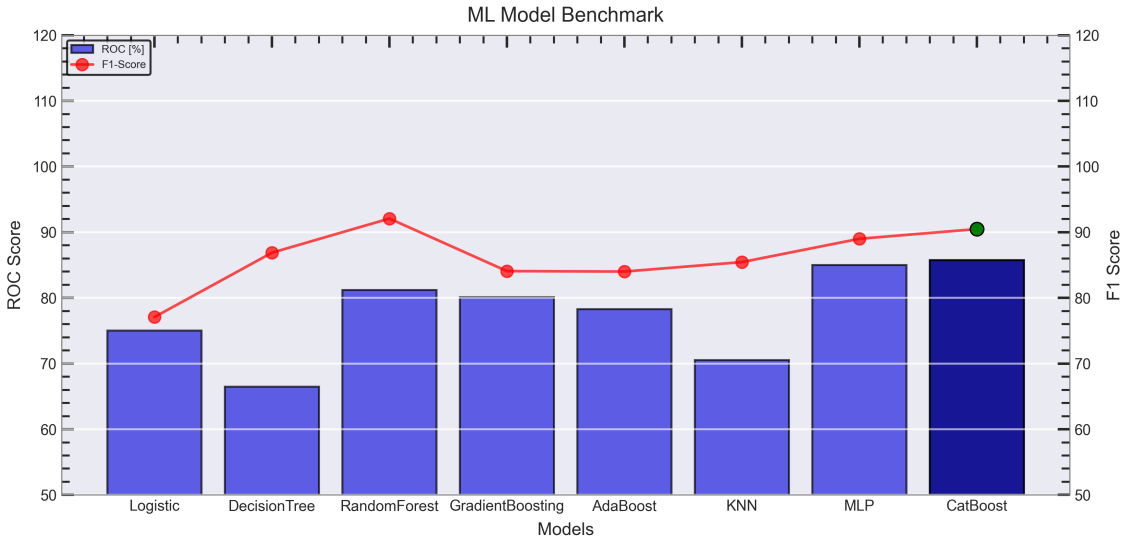


Figure 51: Downstream Classifier: Model Comparison Chart

These comparative results highlight that Catboost demonstrated superior performance and consistency across multiple runs, similar to the SciFi track classifier. Moreover, Catboost showed potential for seamless integration with the Gaudi framework, further supporting its selection as a scalable solution.

7 Catboost Model

Catboost is the primary classifier for enhancing the SciFi seed and UT hits association algorithm due to its ability to handle complex Downstream Track data. It outperformed other models by a large margin. Its learning process, which combines weak learners to capture complex patterns, makes it well-suited for further tuning as the final classifier.

7.1 Hyperparameter Tuning and Optimisation

Catboost performs optimally with the correct hyperparameters, including tree depth, learning rate, number of iterations, and regularisation terms. These settings shape the learning and capacity of the model. Tune them carefully to balance bias and variance, thus improving model performance and stability.

Parameter	Description	Value
loss_function	Objective function to optimize	Logloss
eval_metric	Evaluation metric used for validation	AUC
task_type	Execution mode (CPU/GPU)	GPU
devices	Device index for GPU execution	1
auto_class_weights	Automatic class weight balancing	Balanced
use_best_model	Whether to use the best iteration on validation	True
learning_rate	Step size shrinkage to prevent overfitting	0.24
iterations	number of boosting iterations (trees)	922
depth	Depth of individual trees	8
l2_leaf_reg	L2 regularization coefficient	10
min_data_in_leaf	Minimum samples required in a leaf	55
bagging_temperature	Controls sampling randomness for bagging	1.65
border_count	number of splits for numerical features	243

Table 13: SciFi Track Classifier Hyper-parameters

As with the first Catboost model, a thorough hyperparameter search is performed using Optuna. Table [13] shows the parameters selected for the final model.

7.2 Model Evaluation

This section evaluates the fine-tuned Catboost model as a Downstream Track classifier. Every model has a limit on the improvements it can achieve. Considering the complexity of the track reconstruction using only the SciFi seeds and UT hits (no VELO information), the following metrics demonstrate the model's performance in handling such objects.

7.2.1 ROC Curve

Figure [52] shows the ROC curve for the fine-tuned Catboost model. The AUC score of 85% represents a significant improvement over the linear model. It is not a perfect model from the ROC curve perspective, indicating that the complexity lies within the data structure. Applying a more complicated model or using a larger data set is not a way to improve the performance in our case. First of all, the algorithm must conform within the limits of a real-time LHCb trigger system with a memory footprint that is as small as possible and a response time as fast as possible. Then, increasing the training sample may have an adverse effect on the generalisation since the Monte-Carlo events do not reproduce the data features accurately. Thus, it was concluded that the achieved result is satisfactory and much better than the current trigger baseline algorithm.

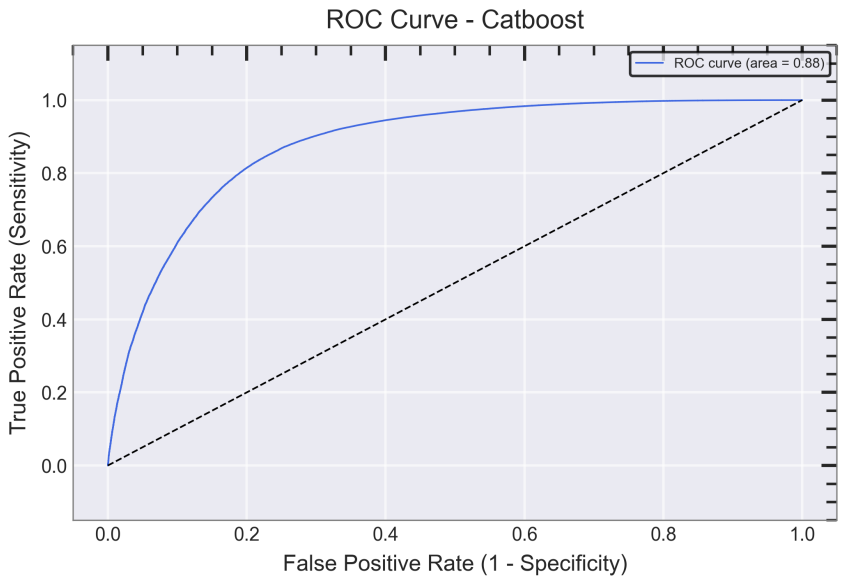
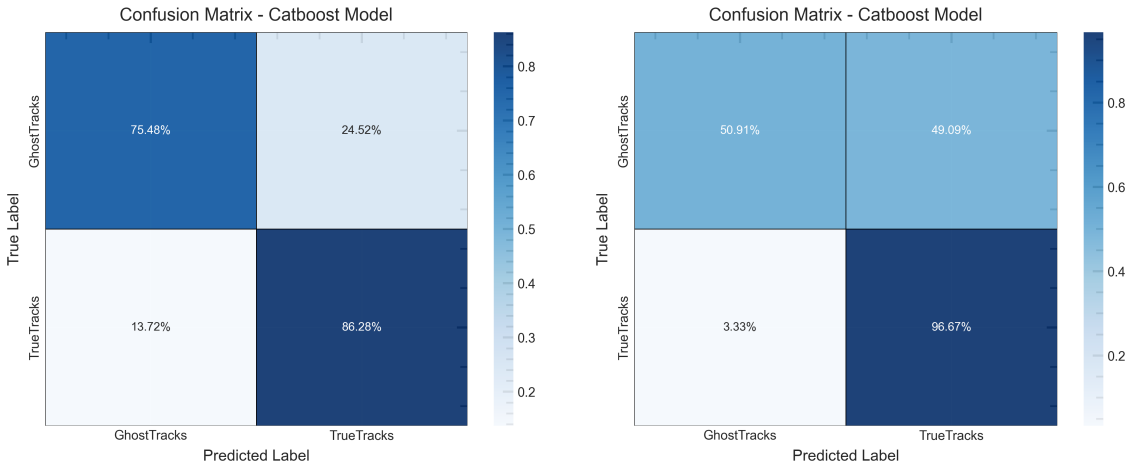


Figure 52: Catboost Downstream Classifier: ROC Curve and AUC

7.2.2 Confusion Matrix

The Confusion Matrix in Figure [53] shows two different threshold points, 0.5 and 0.4, as a reference. An improved confusion matrix indicates a boost in model performance in identifying patterns and learning from them. For the first threshold, one can see that the trained and optimised model is able to recognise both the true and ghost tracks. Lowering the threshold seems to increase the track reconstruction efficiency significantly, but it comes with the cost of a significant misclassification rate of the ghost tracks as true ones. The variation of model classification with varying working points is essential to fine-tune the model performance later in the thesis.



(a) With 50% Threshold **(b)** With 40% Threshold

Figure 53: Catboost Downstream Classifier: Confusion Matrix

7.2.3 Feature Importance

The evaluation metrics show significant improvements in the performance of the Catboost model. The bar plot shown in figure [54] displays the feature importance for the model.

As expected, the number of UT hits influences the predictions the most. Also, there is no significant disproportion in the importance of features among themselves, which indicates that the model uses all the input features to make decisions.

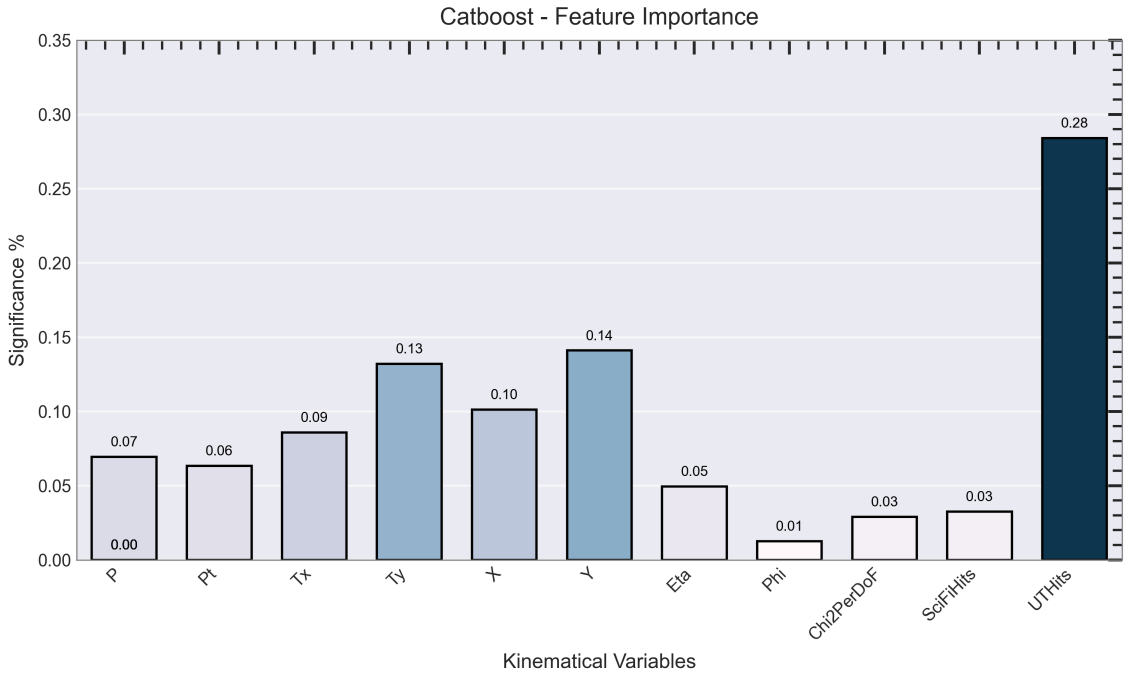


Figure 54: Catboost Downstream Classifier: Feature Importance

7.2.4 Model Response

The fully trained model can be used to obtain response plots for balanced and unbalanced data. The results indicate significant improvements compared to the benchmark model. While the predicted true and ghost tracks overlap, there is still a distinct separation between them that cannot be achieved by the baseline model. Thus, it is evident that the non-linear model, like Catboost, can learn complex patterns within the data. The plot [55] indicates the raw unprocessed data, while figure [56] shows the response of the model if the track type ratio is balanced. Note how the model prioritises the minority class and improves overall model performance.

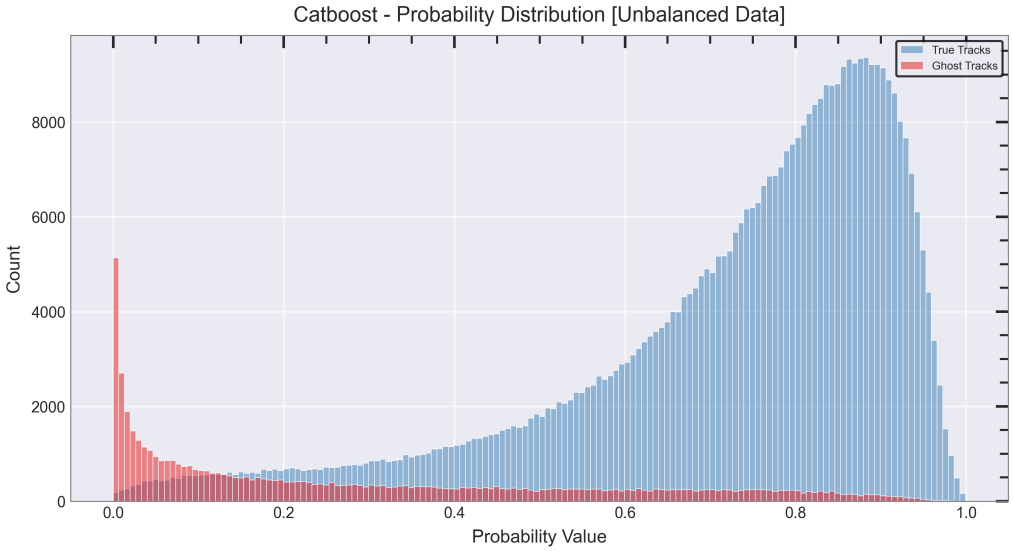


Figure 55: Catboost Downstream Classifier: Model Response (Unbalanced)

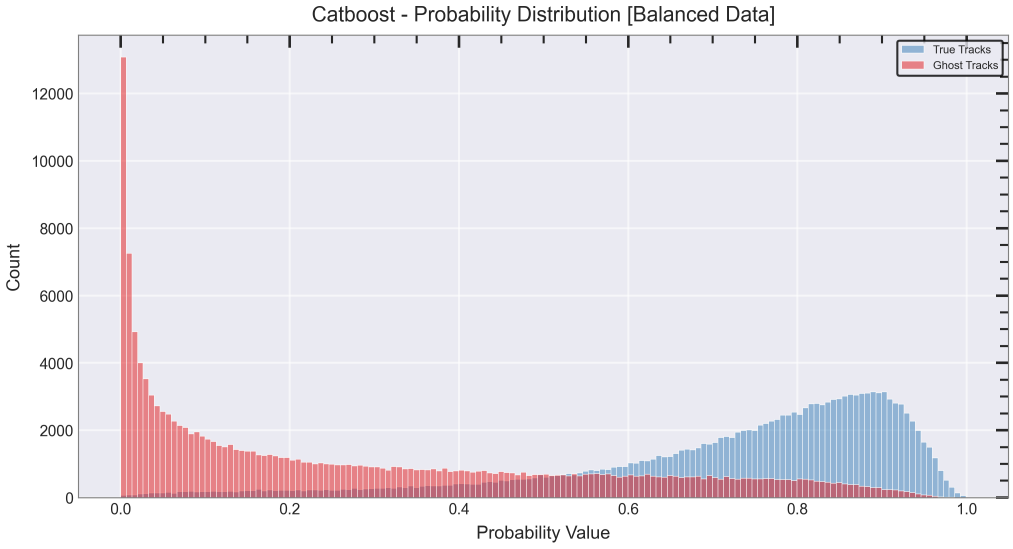


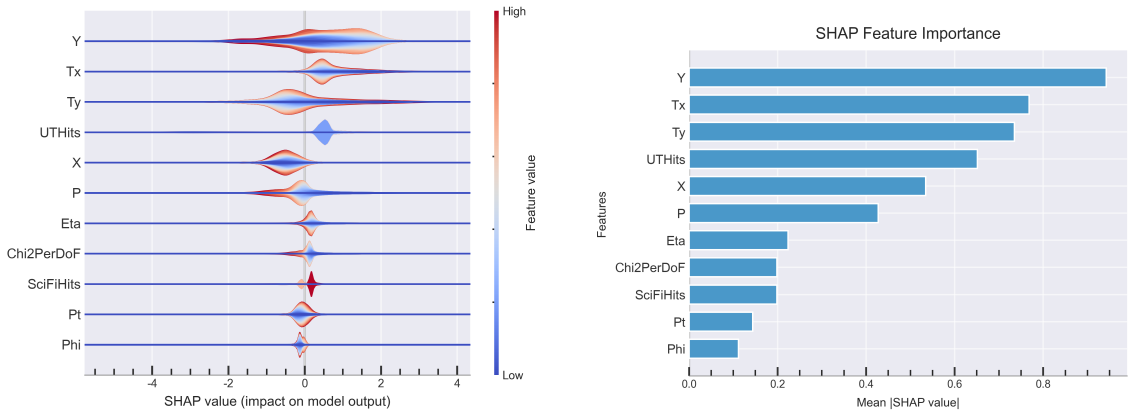
Figure 56: Catboost Downstream Classifier: Model Response (Balanced)

7.3 Results and Interpretation

Understanding the model's output and its impact on data selection is crucial. Complex models are challenging to interpret, making explanations of results difficult. A SHAP analysis that identifies variable dependencies in the model output can be used again to address this.

7.3.1 SHAP Analysis for Catboost Model

However, when these interpretability methods are applied, discrepancies may be observed. For example, the importance of the feature and the SHAP values do not align in their rankings. According to the importance of the Catboost feature, the most influential variable is the number of UT hits. In the case of SHAP analysis, it is ranked third, with Y and Tx ranked as the most influential variables at the track level. The SHAP metric and feature importance may differ because they measure slightly different things, often on various scales. SHAP explains predictions by attributing each feature's marginal contribution to the model output, while Catboost feature importance can mean split-based impact or permutation impact on our loss metric. These target different quantities, so rankings need not match. It is more important to compare the consistency of both explainability procedures, which in this case is clear. Figure [57] show the summary plots.



(a) SHAP Value Distribution **(b)** Average Shap Values

Figure 57: Catboost Downstream Classifier: Shap Analysis

7.4 Conclusion: Downstream Track Classifier

This concludes the chapter on model development and building the second stage of the cascade track selector. As a baseline, the initial benchmarks from logistic regression were suboptimal. However, with a fine-tuned Catboost model, we were still able to achieve a very good model that shows clear improvement compared to the baseline one. At this stage, two frozen snapshots of these models were saved for integration with the Gaudi framework.

Chapter 6

Final Models Integration and Performance Evaluation

1 Scope

This chapter outlines the integration and commissioning pipeline for the two-stage machine learning selection models developed in the previous chapters with the existing LHCb track reconstruction framework. It describes integration methods and evaluation procedures for incorporating these models into tracking algorithms, guided by strict constraints and iterative evaluation of latency, throughput, efficiency loss, and ghost-ratio improvement to quantify the impact of physics.

As mentioned before, Moore is the High-Level Trigger application responsible for trigger analysis and event handling in offline reconstructions. Reconstruction algorithms include *PrHybridSeeding* and *PrLonglivedTracking* algorithms, as detailed in Chapter 2. The objective is to integrate these cascade classifiers into these algorithms and assess their performance. This chapter provides a technical overview, test procedures, and results, with comparisons to baseline benchmarks.

2 Software and Hardware Stack

Model training and evaluation are performed on two separate computing nodes to simplify the process and take advantage of the computing nodes' specific capabilities. The local node (that is similar to those used in the trigger farm) is used primarily for training, while an Lxplus node at CERN is used for model integration studies (real operation environment).

The local machine is preferred for training due to its dedicated NVIDIA Quadro RTX 6000 support and unrestricted access. Table [14] compares the hardware specifications of the computing nodes.

Parameter	LHCb D2 (bare metal)	Lxplus (VM)
OS	AlmaLinux 9.4	Red Hat Enterprise Linux 9.6 (Plow)
CPU	2× AMD EPYC 7F72	16 vCPUs, AMD EPYC-Milan
RAM	250 GB	57 GB
Disks	∼87 TB RAID6 + 1.5 TB NVMe	160 GB root + 150 GB /var/tmp
GPU	2× NVIDIA Quadro RTX 6000	Virtio virtual GPU
CUDA	12.5	No CUDA support
Compiler	GCC 11.5.0	GCC 11.5.0

Table 14: Hardware and system specification comparison between D2 and Lxplus.

The other important information for the reproducibility of the results is the LHCb software stack used for the analysis. Table [15] lists the versions and releases of the various frameworks and components used. At the time of this study, the most recent stable versions of the software stack were used, ensuring compatibility with the experiment's detector configuration and reconstruction workflows.

Framework	Version / Tag
Moore	v58r3
Rec	v39r3
DaVinci	v66r4
Allen	v7r3
LbCom	v38r3
LHCb	v58r3
Gaudi	v40r0
Detector	v3r5
LCG / ROOT	106c / 6.32.10
Compiler	GCC 13
Platform	x86_64_v3-el9-gcc13+detdesc-opt+g

Table 15: LHCb software components and versions used in this work.

3 Design and Dataflow

This work has implemented two methods to integrate the machine learning model into the LHCb tracking framework. Each has distinct advantages and use cases, and both were carefully designed with the LHCb tracking group.

The first method uses a standalone C++ model, directly integrated with the tracking algorithms *PrHybridSeeding* and *PrLongLivedTracking*. This enables iterative development, prototyping, and easy retraining and reevaluation of parameters. The C++ codebase fits the existing LHCb software and is mainly used for comparative studies and rapid experimentation. The second method, in contrast, utilises ONNX Runtime, which is provided as an external dependency by the LHCb software stack. In this approach, the runtime removes the need for a dedicated inference engine, allowing ONNX Runtime to handle inference directly. It should be stressed that developing and implementing such a pipeline is not trivial since it requires direct binding of a Python-based machine learning model, developed in a completely stand-alone environment, with the C++ runtime.

To summarise these approaches, in both methods, track data is passed to the model for inference, where probabilities are evaluated, and decisions are made. The ONNX Runtime is the preferred integration method discussed in this thesis due to its ability to integrate seamlessly within the stack's existing algorithms. This makes the ONNX models easy to swap and requires no extra dependencies. Meanwhile, the C++ integration offers a flexible and transparent environment for fast development and testing. Together, these complementary solutions facilitate rapid prototyping and robust deployment of machine learning models within the LHCb reconstruction workflow.

4 Model Selection and Integration

Catboost was chosen for both selection tasks due to its strong performance and easy deployment. The built-in model saving allows a conversion from Python to C++ and ONNX formats. In C++, the model produces raw scores that can be converted to probabilities with a sigmoid function. This flexibility allows for updates or retraining without significant changes in the reconstruction algorithm. This is especially important for the maintenance and further refinement of the model. For ONNX integration, the model provides predictions and probabilities as separate branches. However, the ONNX Runtime in the LHCb stack was designed for single-entry MVA outputs. Extra preprocessing was needed to remove the second branch and produce a single probability output for downstream inference. This

step ensures compatibility with the production evaluation pipeline and matches the C++ workflow results.
These two separate processes will not lead to significant differences in the model response since they are independent of the inference engine used. In the following sections, ONNX Runtime was chosen as the primary method for model selection. This choice simplifies the chapter's presentation and the reproducibility of the results.

5 Threshold Optimisation

Track selection is directly influenced by the choice of working point, determined by applying a threshold cut to the classifier output. Model responses were computed for various threshold values using the Moore framework to identify optimal thresholds, with evaluations of more than 1,000 events. This approach enables systematic balancing of true track retention and Ghost Track rejection.

Figures [58] and [59] illustrate how adjusting threshold values affects track selection in the case of both algorithms. The primary objective is to find a working point that keeps as many true tracks as possible while reducing the number of Ghost Tracks. This optimisation is carried out separately for each classifier in the analysis.

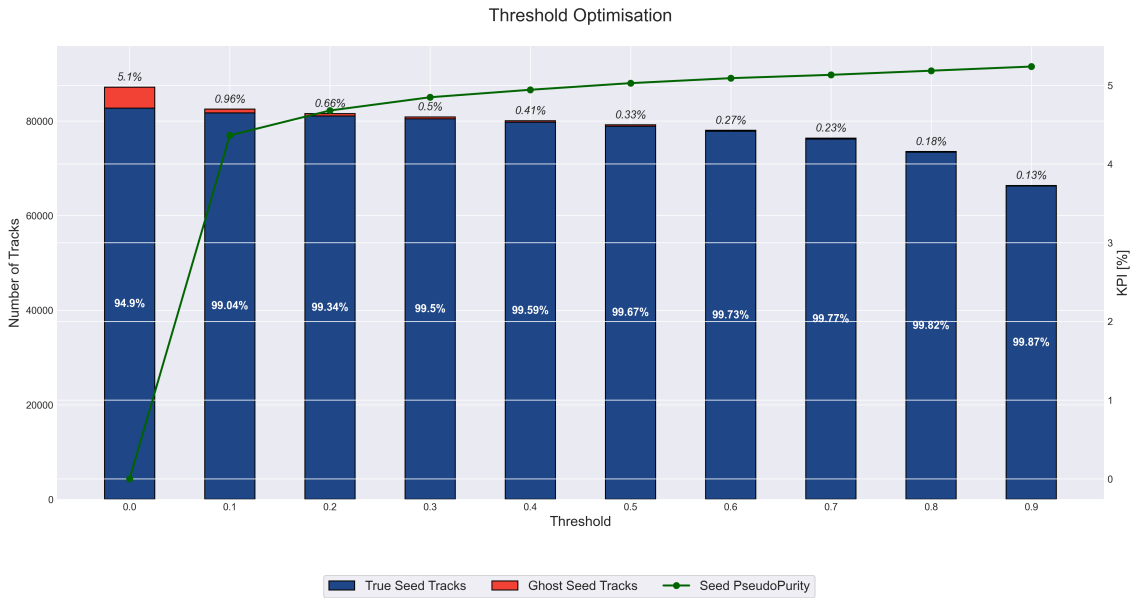


Figure 58: Threshold Optimization to Determine the Working Point of the SciFi Track Classifier in the Tracking Algorithm

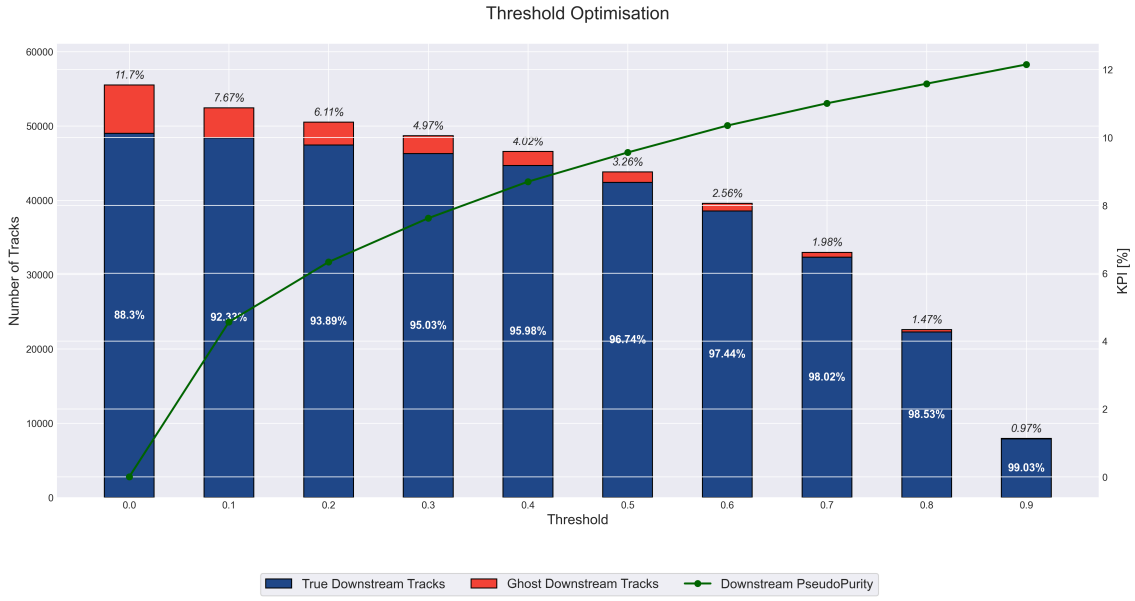


Figure 59: Threshold Optimization to Determine the Working Point of the Downstream Track Classifier in the Tracking Algorithm

In these Figure [58]and [59], the dark blue bars represent the total number of true tracks. The red stacked bars indicate the total number of Ghost Tracks. The ratios of true and Ghost Tracks to the total number of reconstructed tracks are noted in the plots. The green line displays pseudopurity, an additional metric that shows improvements over the baseline, which is set at a threshold of 0.0, where no model-based selection occurs.

The pseudopurity for a given track type is defined mathematically as follows:

$$\text{Pseudopurity}[\gamma] = \frac{R_{\text{tracks}} - R_{\text{baseline}}}{R_{\text{baseline}}} \times 100 \tag{20}$$

Here, R is defined as the fraction of true tracks that equates to the total number of tracks shown in the plots. This metric highlights the relative improvement in track purity from applying selection criteria.

A threshold of **0.2** was selected for both classifiers. This implies that any track with a classifier-assigned probability below 0.2 is considered a Ghost Track and excluded from the reconstruction process. This point of operation provides a balanced trade-off between signal retention and background rejection, resulting in improved pseudopurity while maintaining high track-reconstruction efficiency. All subsequent results and analyses presented in this work are interpreted within the context of the 20% threshold, ensuring consistency in the evaluation.

6 Benchmark Analysis: Baseline and Fine-Tuned Model

In the previous section, the analysis using low event statistics helped to identify a working point in the range of varying probability thresholds. The following sections evaluate and study this fine-tuned model implementation in detail.

Table 16: Comparison of Baseline, C++ Implementation, and ONNX Implementation metrics.

Metric	Baseline	C++ Model	ONNX Model
Overall run time [Event/s]	2.28887	2.42699	2.29775
Total Time (40k Events)	17,475,863	16,481,352	17,408,353
Total Seed Tracks	4,245,176	4,017,458	3,625,892
Ghost Tracks	170,215	21,978	6,329
Ghost Ratio [%]	4.01	0.55	0.17
Hybrid Tracking Algorithm Average Time [μs]	1576.521	2218.108	7276.340
Total Downstream Tracks	2,723,937	2,455,890	2,266,303
Ghost Tracks	303,085	140,097	126,058
Ghost Ratio [%]	11.13	5.70	5.56
DS Tracking Algorithm Average Time [μs]	1966.830	2906.976	7602.41

The results from the Moore logs show that the runs with the integrated models show a significant reduction in ghost ratio compared to the baseline as shown in Table [16]. From the overall runtime and the algorithm-specific average time, it is observed that the C++ implementation method uses no heavy resources; in the case of ONNX, it changes, still producing good overall performance. At the time of writing this thesis, the support of ONNX Runtime is relatively new to the project. This still requires some more optimisations to reduce the computations and time.

7 Track Reconstruction KPI Evaluation

Two primary key performance indicators (KPIs) are used to assess the quantitative impact of the models on track reconstruction: Track Reconstruction Efficiency and Ghost Ratio. Together, these KPIs offer a comprehensive view of each model's performance and reliability in track reconstruction.

7.1 Reconstruction Efficiency

Track reconstruction efficiency is a key metric used to assess the quality of a reconstruction algorithm. It is the relationship between the number of tracks correctly reconstructed by the algorithm and the total number of reconstructible tracks. A reconstructible track is one that, based on the data and detector acceptance, could be reconstructed by an ideal algorithm.

Mathematically, it can be expressed as follows:

$$\text{Reconstructed Track Efficiency} = \frac{N_{Reconstructed\ Tracks}}{N_{Reconstructible\ Tracks}} \tag{21}$$

Based on this metric, figure [60] shows the efficiency of the SciFi track reconstruction relative to the baseline performance of the standard reconstruction algorithm. In parallel, figure [61] highlights the corresponding results for Downstream Tracks, allowing a direct comparison between tracking strategies. The new cascade filter presents a lower overall track finding efficiency; however, this is expected since it is more restrictive than the baseline one. As we see in the next section, this small reduction will bring huge changes in the ghost rate. The impact of the lower efficiency can be carefully studied as a function of the working point of the trained models.

The previous results are based on the standalone C++ implementation to maintain stability during evaluation rather than the ONNX-based approach. In summary, the observed metrics reveal a slight reduction in efficiency compared to baseline, which is expected due to the increased complexity inherent in the problem.

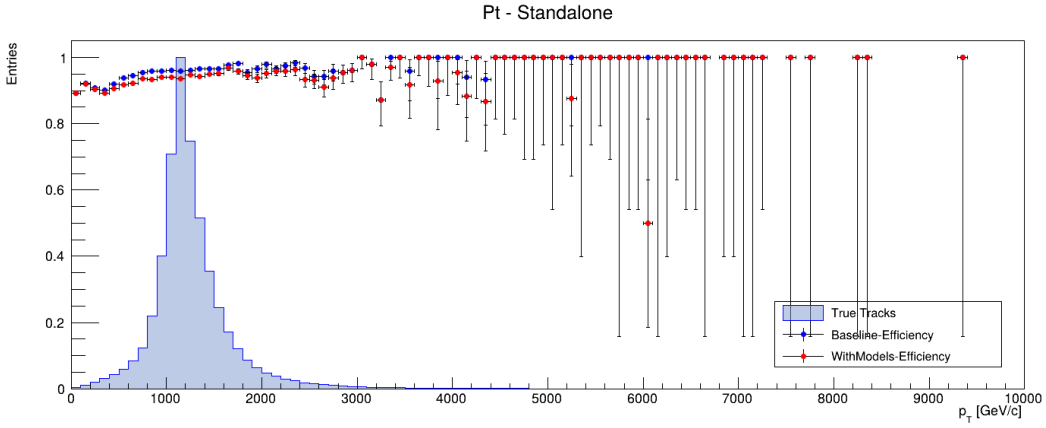


Figure 60: Track Reconstruction Efficiency of SciFi Tracks for Baseline and Model Integrated Configurations as a Function of Transverse Momentum

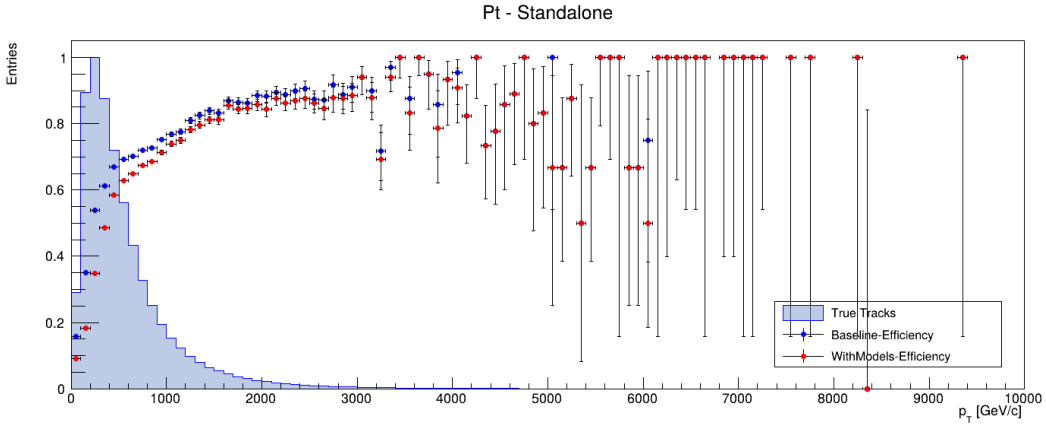


Figure 61: Track Reconstruction Efficiency of the Downstream Tracks for Baseline and Model Integrated Configurations as a Function of Transverse Momentum

7.2 Ghost Ratio

The ghost ratio is another vital KPI used in the analysis to evaluate track reconstruction. The ghost ratio serves as a direct measure of model discrimination capabilities. It is defined as the fraction of combinatorial tracks produced in reconstruction to the total number of reconstructed tracks:

$$\text{Ghost Ratio} = \frac{N_{GhostTracks}}{N_{TotalTracks}} \tag{22}$$

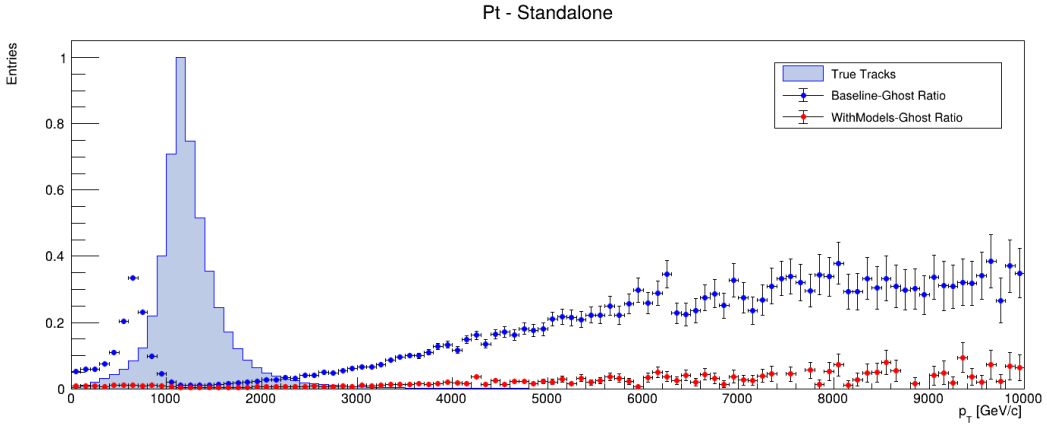


Figure 62: Reduction in Ghost Ratio in the SciFi Tracking for Baseline and Model Integrated Configurations as a Function of Transverse Momentum

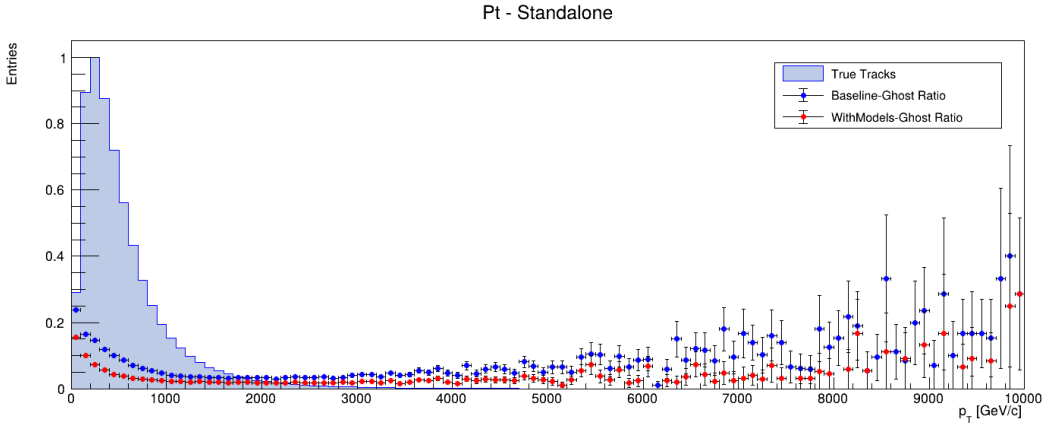


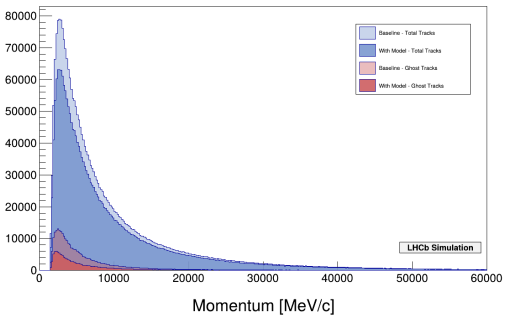
Figure 63: Reduction in Ghost Ratio in the Downstream Tracking for Baseline and Model Integrated Configurations as a Function of Transverse Momentum

Building on this definition, figure [62] presents the reduced ghost ratio for SciFi tracks. Figure [63] follows, highlighting the reduction in the Ghost Tracks downstream. Together, these figures demonstrate that the new cascade models, developed as part of this thesis, reject Ghost Tracks much more effectively than the baseline approach. The fine-tuned cascade model with optimised working point can complete the removal of the ghost SciFi segments virtually. The effect on full Downstream Tracks is less pronounced (figure 63), since the first stage of our model already removed the fake SciFi segments, but nevertheless is still significant.

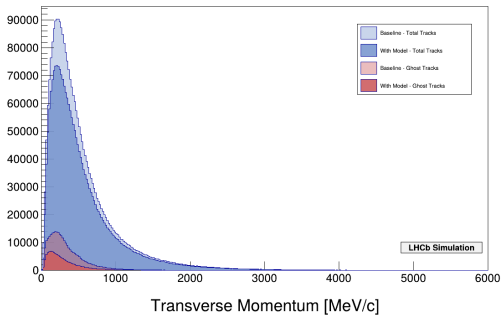
8 Comparison of Kinematical Variables

The previous section evaluated the model's performance under different conditions. Building on this analysis, the next step is to compare the distribution of kinematic variables resulting from model-integrated reconstructions against those from the baseline reconstruction. This approach allows us to observe any significant distortions in the distributions of variables after integrating the model.

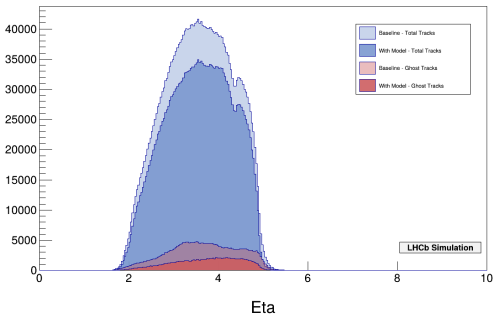
To illustrate any possible effects, each figure presented in the [64] shows four histograms: total tracks for both the baseline and model scenarios, and Ghost Tracks for both cases. This approach offers a direct visual comparison of how model integration affects both tracks. As can be seen, the new enhanced filters do not introduce any changes in the shapes of the key kinematical variables associated with the reconstructed tracks.



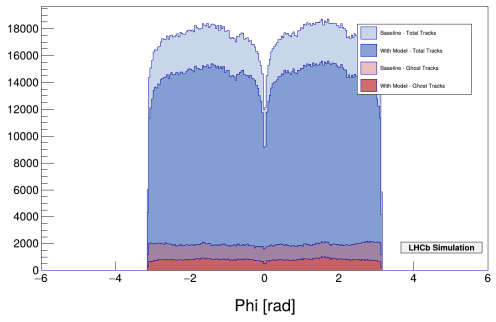
(a) Momentum



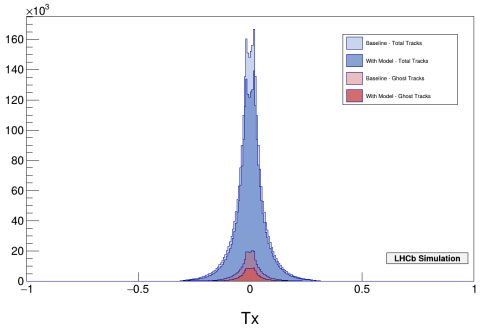
(b) Transverse Momentum



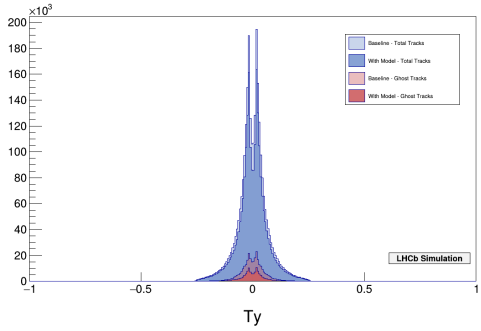
(c) Eta



(d) Phi



(e) SlopeX



(f) SlopeY

Figure 64: Kinematical–Geometric Values Distribution of Reconstructed Downstream Tracks

9 DaVinci

As the final step of the new cascade filter impact analysis, this work extends to the DaVinci framework to validate the reconstruction performance with the classifier within the LHCb reconstruction environment. Figure [65] shows the invariant mass distribution comparison for a sample of selected K_s^0 particles in the baseline and the reconstruction with the model integrated. This additional physics-related performance is a key element of evaluating the developed algorithm.

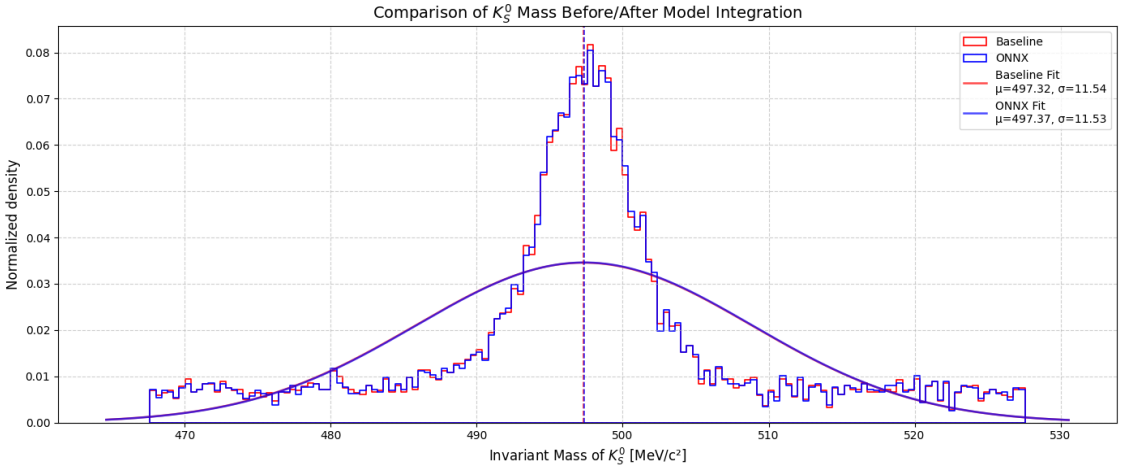


Figure 65: Invariant Mass Distribution Comparison: Produced using DaVinci for baseline particles reconstructed with baseline and with incorporating the integrated model changes

The reconstructed mass peak has no biases, and its width is almost identical to that reconstructed by the baseline algorithm. There is also no change in shape. The main advantage of the new models lies in a significant reduction of the time spent by the trigger code in processing Ghost Tracks. The peak obtained by the baseline algorithm using the polluted Downstream Tracks is not significantly different from the one reconstructed using the purified tracks because of the sophisticated kinematic decay fitting procedure. Also, further studies of the new filtering cascade model may lead to improving physical performance, for instance, by providing better mass resolution but with more substantial cuts on the tracking efficiency. Moreover, the line selecting just K_s^0 particles may not tell the whole story. To study this in more detail, one would require a more complicated decay with K_s^0 particles in the final state. However, such studies are out of the scope of this thesis.

10 Conclusion

This chapter discusses in detail the impact of two models built and integrated with the Downstream Track reconstruction for improved performance. The efficiency and ghost ratio trade-off was the most interesting performance indicator evaluated in this thesis. None of the models built during this work is perfect; they still show flaws, and there is always room to improve them. The current pipeline in this chapter shows a substantial improvement in reducing combinatorial tracks produced in the detector. Selectively picking Downstream Tracks will help in other analyses, as the downstream process now has to deal with fewer and cleaner tracks. The current tuning of the cascade models reproduces almost perfectly the K_s^0 state compared to the baseline algorithm. More complex studies regarding the tuning of the new filter's working point may lead to better physics performance, but they require the fully commissioned algorithm and real data.

Phase III: Consolidation

Detector Calibration and Conclusion

Chapter 7

Upstream Tracker and Calibration

1 Scope of Research

The Upstream Tracker (UT) is a four-plane detector built with silicon microstrip sensors positioned upstream of the LHCb dipole magnet (towards the VELO detector). As a part of the LHCb spectrometer, the UT is crucial for precise momentum estimation in HLT triggers. It provides precise hit information in the bending plane (X axis in the LHCb coordinate system) before the dipole magnet, making it a critical component in helping to minimise ghost tracks.

Silicon sensors in the UT need regular recalibration to maintain an optimal signal-to-background ratio. The calibration process requires taking special data samples on a regular basis, both with the colliding beams (end-of-fill samples) and with no particles present (noise runs). This chapter discusses methods to improve pedestal benchmarks for UT sensors and reduce the noise hits. The pedestal offset in silicon sensors typically refers to the baseline signal present even without a particle hit, caused by electronic noise, leakage current, or various common-mode effects (like a pick-up from the circulating beams). Each readout channel has its own individual offset, which, if uncorrected, can mimic low-energy hits and significantly degrade the sensor's resolution. To mitigate this, experiments perform regular calibration runs to measure baselines, subtract them in digital processing, and apply common-mode corrections, since pedestals can drift with temperature, radiation damage, or electronics and silicon bulk ageing. The pedestal correction is considered the most essential part of the daily calibration for the UT detector. Precise adjustment and monitoring of these values improve the signal-to-noise ratio by reducing background interference.

The following chapter offers an overview of the detector and the system's current state. Topics include signal optimisation, on-chip data processing, approaches to forecasting pedestal variations over time, etc.

2 Detector Layout and DAQ Systems

The UT detector was already described in detail Chapter 1, thus, in this section, we only briefly discuss the detector's most relevant features to aid in explaining the calibration data analysis presented in this Chapter. The UT has four active measuring planes arranged sequentially along the beam axis. They are called UTaX, UTaU, UTbV, and UTbX, where U and V are, so called, stereo coordinates necessary to obtain 3D hit information out of 1D measuring strip channels.

The detector employs a complex approach to data acquisition and processing. When a charged particle traverses the detector, it ionises the silicon bulk and produces an analogue current signal [48]. The microstrips collect the charge from the local silicon region. They serve as the primary channel for data acquisition. The strips connect to the SALT ASIC. Each of which can process 128 such stip-channels simultaneously. Several SALT ASICs can be combined to form a hybrid that can process up to 1024 channels. There are two types of hybrids: VERA, which has 4 ASICs, and SUSI, which has 8 ASICs.

A UT module combines a silicon sensor and a hybrid support. These modules connect to eLinks, which deliver serialised data from SALT to GBTx for packaging and transmission to further processing steps performed off the detector. The UT tracker has four sensor types. Type A is the most commonly used, accounting for approximately 92% of the planes. Types B, C, and D are placed primarily near the beam pipe for greater granularity.

2.1 The SALT ASIC

The **Silicon ASIC for LHCb Tracking** (**SALT**) is a custom-developed readout chip designed to handle high-density and high-volume data quickly and reliably from UT. It operates at a 40 MHz clock rate and can do 128-channel readouts simultaneously. SALT is fabricated using radiation-hard 130nm CMOS technology (TSMC), optimised for the LHCb experiment.

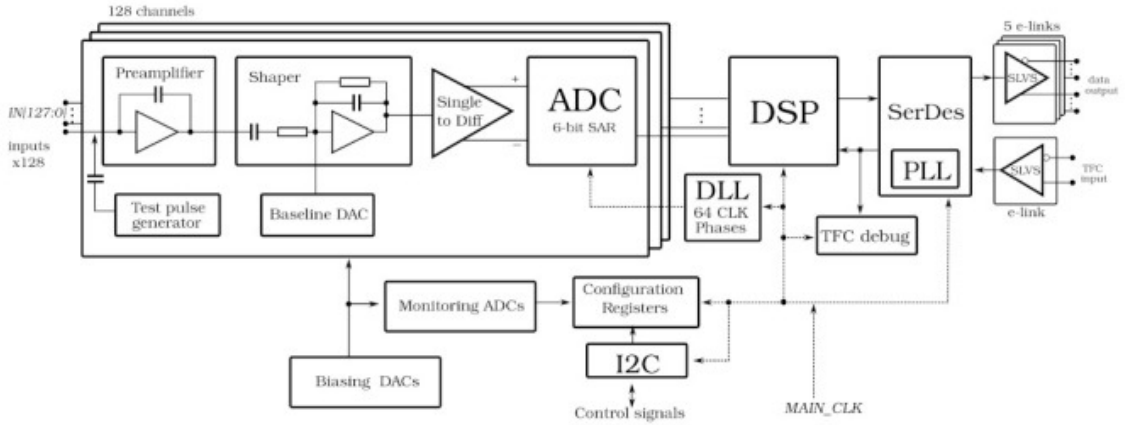


Figure 66: SALT ASICs Diagram

SALT helps significantly to reduce the detector data volume by performing on-chip data preprocessing before transmission. SALT ASIC comprises two functional components:

Analogue Component: This block helps translate analogue signals to digital signals. It consists of

- Charge-Sensitive Amplifier
- Fast Shaper with a peaking time of 25ns
- Signal-to-Differential Signal Converter
- Fully differential 6-bit Successive Approximation Register (SAR) Analogue-to-Digital Converter (ADC)

The fast shaper facilitates bunch-crossing separation, which the differential converter improves signal quality by converting single-ended signals to differential forms.

Digital Component: This block houses a Digital Signal Processing (DSP) unit and is responsible for on-chip signal processing and data reduction. DSP helps with the following

- Channel Masking: Disable noisy or non-functional channels.
- Pedestal subtraction: Optimise the baseline offsets per channel
- Mean Common Mode Subtraction: Eliminates external noise by subtracting the average signal from all channels.
- Zero Suppression: Retain the signal based on a threshold, which helps to reduce data.

This digitised data is transmitted through *e-Links* using *Scalable Low Voltage Signalling* (*SLVS*) interface technology. This interface is connected downstream of the DSP to ensure efficient low-noise readouts at high bandwidth. A key feature of SALT is its on-chip data preprocessing, which mainly involves three types.

2.1.1 Pedestal Subtraction

For a channel, pedestals represent the baseline signal in the strip in the absence of a particle hit. The RAW ADC values will include these pedestal values for each channel, which must be carefully removed to obtain an accurate detector signal.

2.1.2 Common-mode-Subtraction

In addition to pedestal correction, SALT can execute common-mode subtraction (CMS), a digital method for correcting detector noise that may affect a collection of channels (non-local effects) such as temperature variations or power supply interference. By calculating and removing the average offset shared across multiple channels for each event, CMS ensures that these common fluctuations do not influence the measurement of individual signals. At the moment, the SALT chip is capable of executing a simple linear offset correction. Together, Pedestal Subtraction and CMS form a complementary approach to reduce readout noise and improve accuracy.

$$\text{Signal} = ADC_{raw} - Pedestal - CMS_{offset} \tag{23}$$

2.1.3 Zero Suppression

Following noise reduction, data efficiency becomes essential. In the Physics stream, most strips do not register hits in an event. Transmitting data from all 128 channels within SALT when no hits is inefficient. By applying zero suppression, a channel event is recorded only if its value exceeds a set threshold. This approach significantly reduces the data sent downstream, improving efficiency.

In summary, SALT ASICs form the backbone of the UT readout system, providing 128-channel readout, on-chip digital processing, and channel masking for bad channels before passing the data downstream.

3 Pedestal Calibration

SALT ASICs form the backbone of the UT readout system, providing 128-channel readout, on-chip digital processing, and channel masking for bad channels before passing the data downstream. *Pedestal calibration* is an important step in optimizing the signal-to-noise ratio.

Monitoring and calibration tools like Vetra decode non-zero suppressed (NZS) data from the UT detector and help recalibrate the detector. Using non-zero suppressed run data from a RAW ADC for calibration runs is essential. The Physics stream contains empty channels, but the NZS data provides a complete distribution. Calibration is typically performed by injecting a test pulse or at the end of fill runs. The results of these runs are stored and used for the zero-suppression threshold.

This chapter explores two approaches to pedestal calibration studies in UT. The first examines the observed pedestal value for a calibration run and uses KL Divergence Statistics with a baseline benchmark calibration run to find their similarity in distribution. The second method studied involves forecasting pedestal values for the forthcoming calibration run using LSTM. This project intends to build a foundation for potential future extensions.

3.1 Pedestal Calibration Data

In this study, 51 NZS runs were collected using Vetra, a UT analysis tool built on Gaudi. Calibration Run Number `[ 298593 ]` is considered the benchmark calibration run for reference.

Given the data granularity, the UTaX plane is used as a test case for the subsequent analyses. Figure [67] shows the transition from channel-level to module-level analysis, illustrating the mean-averaged pedestal values per module for the benchmark run in the UTaX plane.

To understand the patterns of pedestal distributions, Figure [68] presents the pedestal values per channel for three consecutive runs after the benchmark run, providing a visual basis for comparison, and Figure [69] shows the density distribution of pedestal values as a starting reference for comparison.

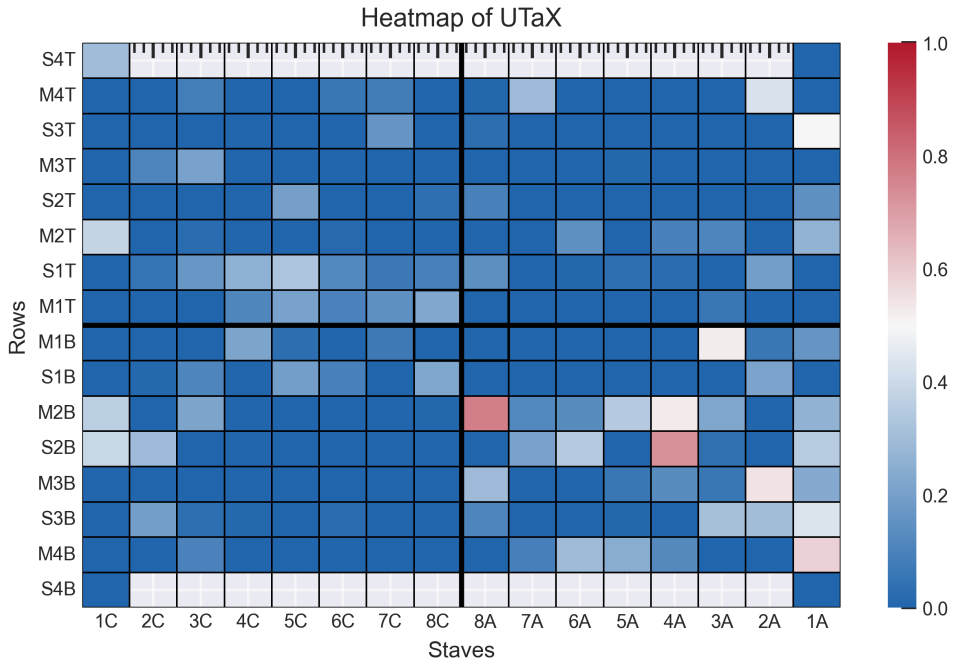


Figure 67: Average Pedestal Values, calculated per module, in UTaX measuring plane for the Benchmark Run. The presented projection follows the physical layout of the modules in the plane UTaX

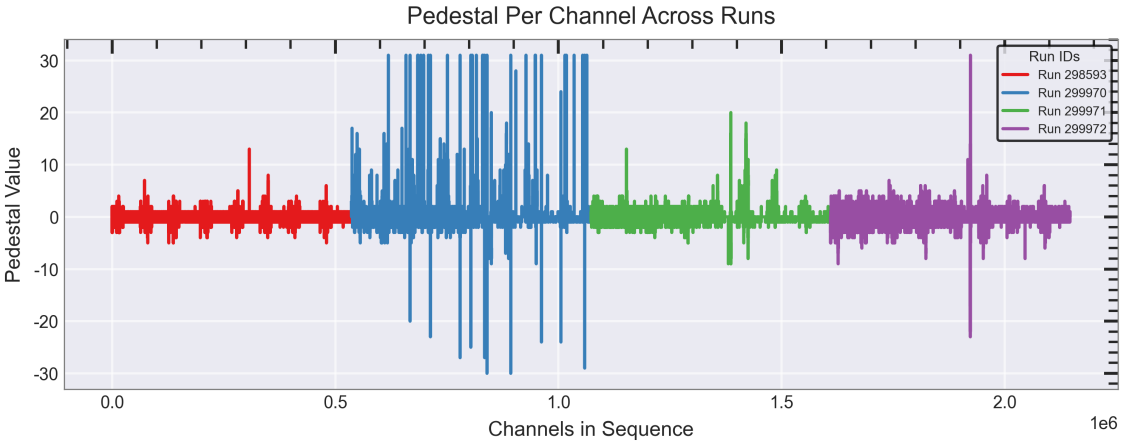


Figure 68: Absolute Pedestal Values per Channel for three consecutive runs after the benchmark run.

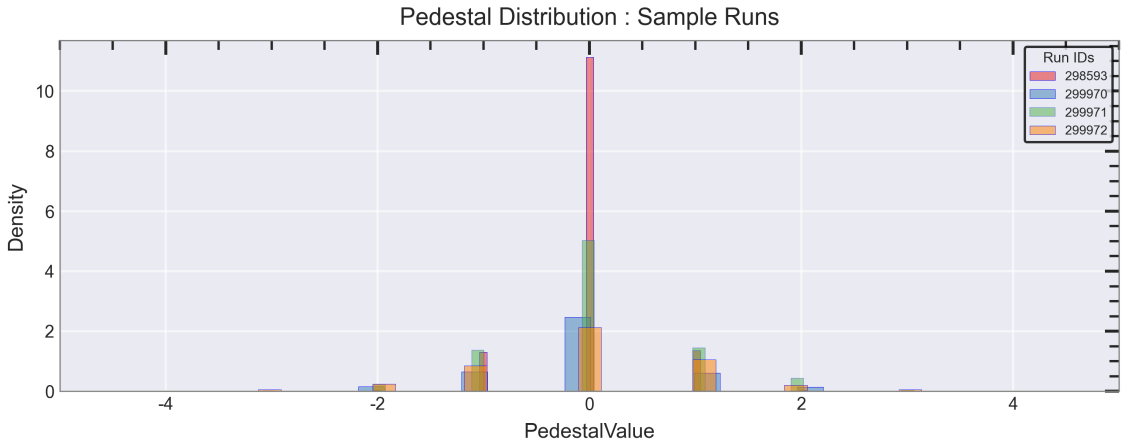


Figure 69: Pedestal Value Distribution Histogram : Benchmark Runs and Three Consecutive Calibration Runs

4 KL Divergence Evaluation

In the scope of this work's calibration studies, the Kullback-Leibler divergence (KL divergence) is considered a method for comparing two calibration runs. The purpose is to find a reliable and easier-to-implement quantitative measure for comparing similarities or dissimilarities in 2D distributions of the average pedestal values between two runs.

The goal is to create a single quantity that indicates when a recalibration is needed based on distribution changes between runs. Such a metric would be vital for creating an automatic and autonomous (not requiring a human operator) software procedure capable of triggering a new calibration and parameter update. The analysis is done at a module level for each UT plane. The divergence is measured relative to a baseline benchmark run, with the run number `298593` serving as the reference. This benchmark is assumed to serve as a reliable reference for identifying any need for recalibration or significant differences in distribution compared to the current run.

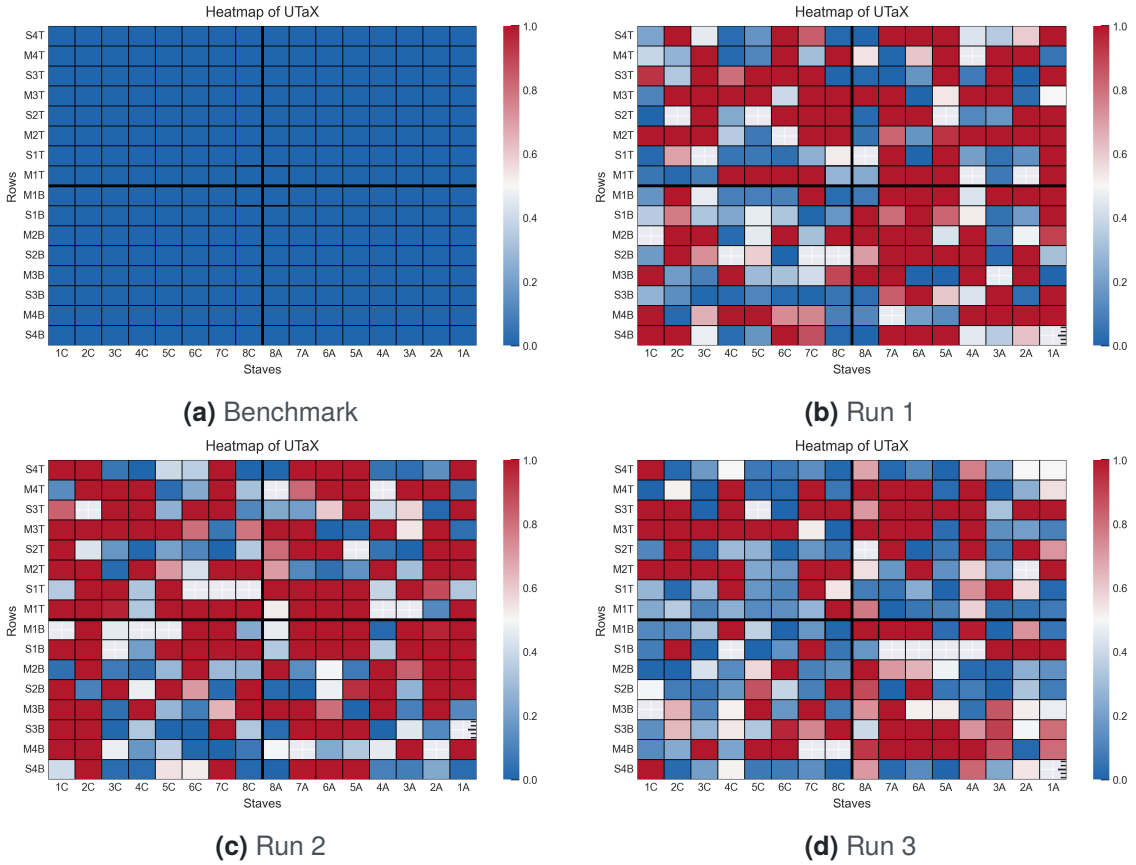
Given two probability distributions between two runs, P(x) and Q(x), the KL divergence can be mathematically represented as

$$D_{KL}P(||Q) = \sum_{n} P(x).log\frac{P(x)}{Q(x)} \tag{24}$$

P(x): Pedestal distribution from benchmark run

Q(x): Pedestal distribution from new run

The figure illustrates the variation in the KL divergence between the benchmark and specific runs. Red indicates the highest divergence from the benchmark, while blue indicates a module with low KL divergence.



(a) Benchmark **(b)** Run 1

(c) Run 2 **(d)** Run 3

Figure 70: KL Divergence Calculation with Benchmark Run Probability Distribution per Stave for UTaX and the next three consecutive calibration runs

The first UT plot in [70] represents a self-comparison of the benchmark, which naturally results in zero values for the metrics. In addition to this, the overall probability can also be considered for further analysis. Examining the KL divergence value makes it possible to assess the calibration requirement and gain insight into the underlying probability distribution, without extra steps.

5 LSTM-Based Pedestal Forecasting

This is a more challenging approach that was initiated during the course of this research. The goal is to build a neural network that captures and remembers the time dependence of the pedestals across runs. Each calibration run serves as a snapshot in time. The objective is to train a model that learns from past run events' history and forecasts future events before the next run. This approach enables thresholds to be set in advance, facilitating the anticipation of calibration needs. A

capable model that captures these intrinsic patterns facilitates anomaly detection when new runs do not align with the forecast calibration data.

5.1 Long Short-Term Memory (LSTM)

LSTM is a Recurrent Neural Network (RNN) type that retains temporal dependency between time steps. It is designed to handle both vanishing and exploding gradient problems and perform well with sequential data. Unlike other neural networks, the LSTM has a memory cell orchestrating memory transfers.

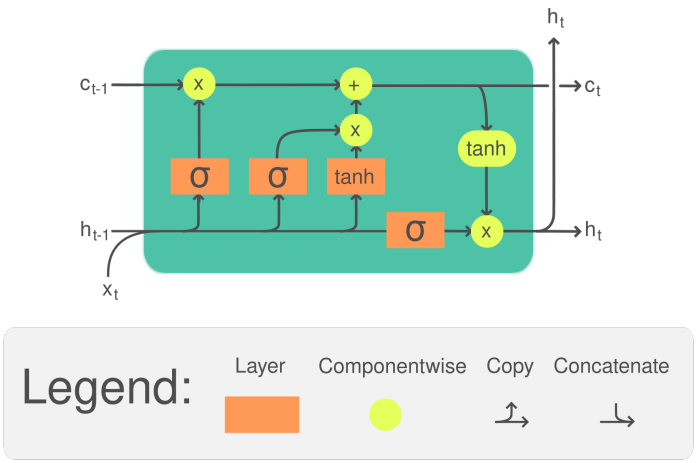


Figure 71: LSTM Memory Unit

As shown in figure [71], a unit memory cell consists of three main gates:

Input Gate: Determines the extent to which new information is stored by processing the current input and the previous hidden state.

Forget Gate: Decides which information should be retained or discarded from the memory cell.

Output Gate: Calculates the amount of information from the memory cell transmitted to the next step for further processing.

5.2 Model Architecture and Training

The LSTM model learns the temporal dependencies between the calibration runs to predict the next run. For consistency, the model is trained to predict 16x16 modules in the UTaX plane, resulting in 256 modules. This study uses 51 calibration runs, divided into two groups. The first 40 runs are for training the model, and the remaining 11 are for testing. Sequences of 10 calibration runs are used to predict

the next in a rolling fashion to forecast pedestal values in the upcoming calibration run.

The architecture for the LSTM model is as follows:

input_size = 256: Number of input features corresponding to the flattened sequence of modules in one plane (all 16x16 modules).

hidden_size = 128: the hidden state vector's size determines the memory capacity within an LSTM layer.

num_layers = 2: Number of stacked LSTM layers, allowing the extraction of deeper patterns from sequential data.

dropout = 0.3: Dropout rate to prevent overfitting.

num_epochs = 1000: Model training cycles in the entire training dataset.

loss_function = MAE: Mean Absolute Error, a statistical measure of the differences between prediction and true values.

optimiser = Adam: Adam (Adaptive Moment Estimation), an algorithm that adjusts learning rates during training.

5.2.1 Loss Function : MAE

The MAE (or L1) loss function targets the magnitude difference between true values and predictions, not directions.

$$\text{MAE} = \frac{1}{n}\sum_{i=1}^{n}|y_i - \hat{y}_i| \tag{25}$$

n**:** Total number of data points.

y_i**:** The actual (observed) value.

$\hat{y}_i$**:** The predicted value from the model.

The mean absolute error (MAE) is evaluated over 1000 iterations during training. Figure [72] shows how the training loss decays over iterations.

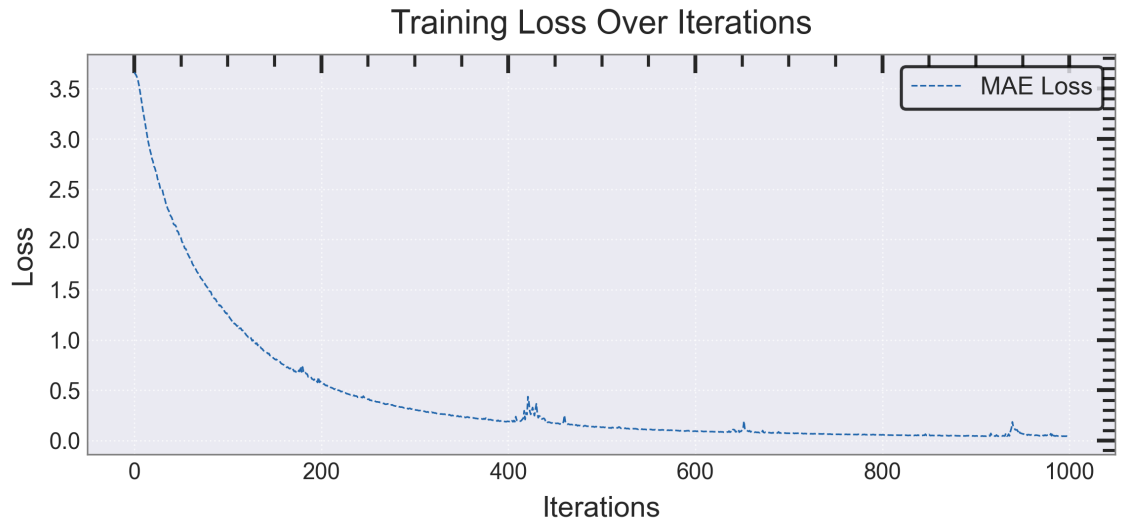


Figure 72: LSTM Forcasting Model Training Loss Over 1k Iterations

5.3 Model Forecast

As described earlier, the test dataset includes 10 calibration runs for model prediction and one additional run for evaluation. The trained model uses the 10 historical runs to forecast the subsequent run, which is then compared against the known evaluation run.

Each run contains 256 mean pedestal values, applied sequentially to predict the following UTaX plane sequence of 256 modules. Figure [73] illustrates the evaluation run along with the 10 calibration runs used in forecasting and the corresponding predictions generated by the LSTM model.

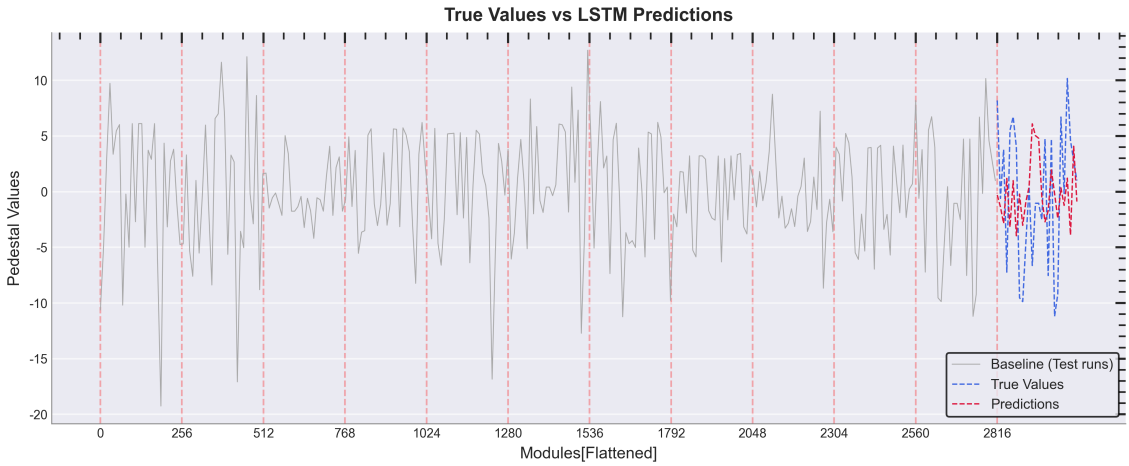


Figure 73: LSTM Forecast for the upcoming calibration run based on 10 historic calibration runs

To better understand figure [73], the gray lines indicate the sequence of past calibration data used for the predictions, and the vertical grid separates the different calibration runs. The blue line represents the true pedestal values observed, while the red lines show the model output.

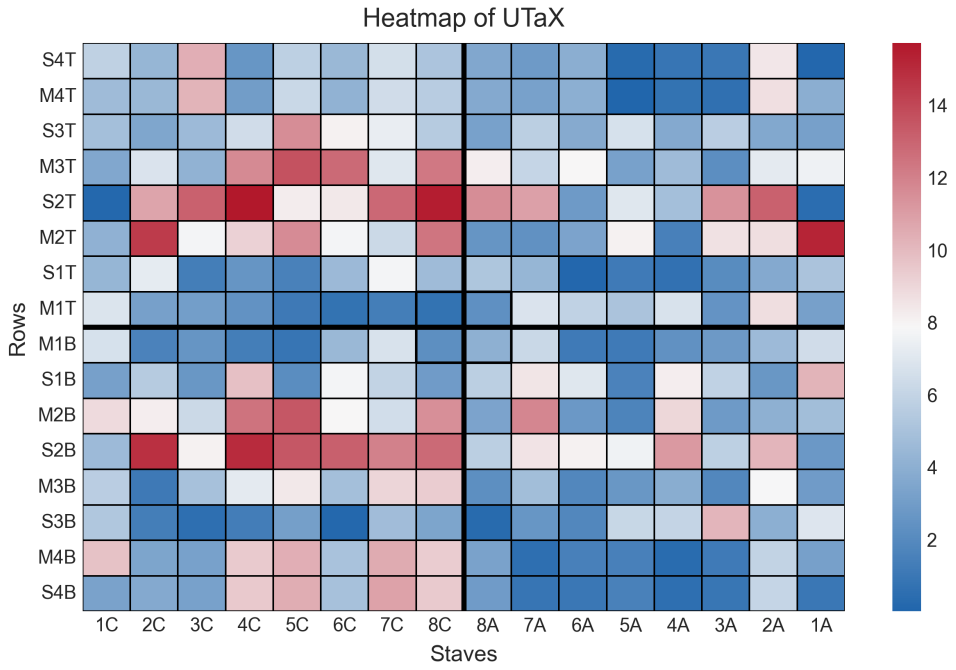


Figure 74: Difference in true value and model prediction [Absolute]

Figure [74] illustrates the absolute difference between the true calibration values and the predicted values of the model, which can help identify any significant deviations from the model forecast. The initial model gives a reasonable approximation of the average pedestal values. In order to improve it further, it requires more data and more capacity (the initial model may be too simple). The prospective work may include more sophisticated recurrent models such as Mamba. The goal is to train a robust model to learn the temporal dependencies and evaluate the results. This serves as an extension to the future scope of research.

Chapter 8

Conclusion

1 Summary of Research

The most significant achievement of this thesis is developing and demonstrating a study that significantly enhances the downstream track reconstruction in High-Level Trigger (HLT) systems for the LHCb experiment. The results and conclusions presented here are particularly important in the context of LHC Upgrade I, specifically in enabling the transition to a robust, fully software-based trigger system. My approach also shows promise for tuning intelligent models beyond Upgrade I.

In the LHCb experiment, the real-time trigger system plays a central role in selecting the most relevant events from an immense flow of collision data. The algorithms implemented at this stage are exceptionally balanced both in terms of memory footprint and speed of execution, as even minor modifications can have far-reaching consequences for efficiency and physics reach. Within this demanding environment, I contributed to the development of the tracking procedure for reconstructing the daughters of long-lived particles decaying beyond the vertex detector by designing and integrating additional machine learning–based filters. These enhancements were implemented with great care to preserve the stability of the system, while at the same time improving its ability to discriminate significantly the ghost tracks. The work demonstrates how advanced data-driven methods can be safely and effectively incorporated into one of the most sensitive components of the LHCb data processing chain. The obtained new algorithm is able to almost completely remove the ghost from the SciFi track sample (seed segments) and significantly reduce the ghost contamination in the downstream track sample. The main impact of the enhanced algorithm, using the current working point tuning,

is related to the reduced time of processing in the HLT trigger. On the physics performance level, we see that the new algorithm is able to reproduce the K peak without any biases or distortions. Future work, after the commissioning of the new algorithm within the HLT stack, may be devoted to retuning the working point of the cascade filters to improve further the physics performance. It can be done at the cost of decreasing the reconstruction efficiency and needs more detailed studies using the full reconstruction of a selected signal decay with K in the final state.

Beyond the software trigger and the study on Downstream Tracking, the thesis also aimed to develop a calibration and monitoring tool to optimise the Upstream Tracker (UT) signal-to-background ratio, which directly influences the overall improvement in tracker DAQ. This part of the thesis used the real physics data collected by the LHCb experiment during the proton-proton collisions and dedicated UT calibration noise runs.

2 Results and Findings

This research explores and tests multiple machine learning models to evaluate different classes of events. The most effective strategy in this context is the adoption of a two-stage track selection method to improve the performance of the downstream tracking algorithm. The results presented in this thesis are obtained under specific conditions and test cases described in detail in earlier chapters. Intermediate steps and results are explained, and the final results are derived after integrating the models with reconstruction algorithms.

The two-stage selection strategy enhances the reconstruction of downstream tracks and the associated SciFi track segments, which serve as their seeds. This method primarily aims to remove the combinatorial tracks, referred to as ghost tracks, that arise during track reconstruction. In the case of both stages, two different integration approaches are explored for model deployment: a C++ standalone implementation and an ONNX Runtime–based deployment. The results are compared with the baseline benchmark run, with no additional models discussed in the thesis that are not integrated. In the first-stage selection, the standalone implementation achieves a ghost track reduction of approximately 87%. At the same time, the ONNX-based approach signals a reduction of 96% ghost tracks produced at the time of SciFi reconstruction compared to the baseline benchmark. The second-stage selection operates in a noisier environment during the final downstream reconstruction. Here, both deployment methods achieve a ghost track reduction of approximately 50–60% relative to total ghost tracks produced in the baseline run.

In the context of UT calibration, KL divergence is set to be a robust evaluation

metric. It helps in effectively comparing pedestal value distributions between a benchmark run and the current run, which helps assess the detector's recalibration requirements. Furthermore, an LSTM-based approach extends this analysis by forecasting future pedestal values based on historical run data. Both approaches are discussed in detail in earlier chapters.

3 Limitations of Current Approach

The results obtained in the work are the culmination of multiple methods implemented in due time with this research. The evaluation metrics showed a consistent trade-off between the efficiency of track reconstruction and the purity of reconstructed tracks. This balance indicates that the proposed two-stage selection method concludes a balanced method for removing ghost tracks generated in the detector without losing many true tracks in the process.

One of the key constraints in deploying machine learning–based models for track reconstruction is their impact on track selection and overall throughputs. In order to tackle this, two different deployment strategies are considered, primarily focusing on a comparative study and the ease of implementation of a new trained model in the reconstruction selection. Even though both integration methods showed a significant reduction in the ghost ratio, they differ in computational time and efficiency. The C++ standalone method is an easy-to-implement prototyping solution with resources similar to the baseline. In contrast, the ONNX runtime currently faces performance bottlenecks. However, ONNX still shows a better integration method due to its modularity and ease of implementation in the LHCb stack. The performance overhead makes ONNX the secondary choice for integration in the present study, despite its otherwise promising generality and flexibility.

4 Future Explorations

The evaluation and metrics presented in this work are not a final solution to the challenges of track reconstruction in the LHCb experiment. Instead, the contributions made here consolidate and lay a solid foundation for future developments in this direction. The two-stage track selection method explored in this thesis demonstrates promising results for downstream tracks and has the potential to be extended to other reconstruction streams within the experiment.

The recent integration of ONNX Runtime support into the Moore framework provides a new layer of flexibility and introduces additional possibilities for ex-

ploring other ML-based track selection methods. In this work, Run 3 data were evaluated through two distinct integration pathways: a C++ standalone and an ONNX Runtime method. From the consolidated results, the deployed models point the way toward the possibilities of further improvements. For instance, retraining the models with a broader and more diverse range of track reconstruction events could strengthen generalisation, and fine-tuning the models even further may make capturing more complex patterns in event data possible. In addition, the strict time constraints, the throughput requirements, and other constraints of the LHCb experiment remain a critical area of focus. Techniques such as model pruning, compression, and the design of lightweight architectures could significantly reduce inference latency, potentially a solution for the high-throughput production environments.

Parallel to the downstream reconstruction studies, this thesis investigated methods for improving UT calibration strategies. The forecasting model built for the pedestal prediction using an LSTM-based model shows good results, although it is not optimised as a highly reliable model for the operations. The framework and the pipelines are being built and can be further improved with the models. With access to larger datasets and the adoption of more advanced neural network architectures, the predictive capabilities of this approach can be substantially improved in future work.

5 Remarks

While the present research provides a reliable and strong candidate for improving downstream track reconstruction in the LHCb experiment, it establishes a solid foundation. The work presented here represents a comprehensive account of the models, evaluation strategies, and integration techniques explored throughout the study. Earlier attempts at model design and enhancement strategies further revealed promising directions for extension and refinement.

In summary, this research provides both a conclusive summary of improvements observed in downstream track reconstruction with the aid of a machine learning-based track selection approach and acknowledges the future possibilities of further improvements.

References

[1] Axios. "Scientists find the universe is 13.77 billion years old." (2021), [Online]. Available: https://www.axios.com/2021/01/07/age-of-the-universe-estimate (visited on 09/28/2025).

[2] L. Pescatore. "Cp violation measurements at the lhcb experiment." arXiv: 1410.2293. (2014), [Online]. Available: https://arxiv.org/abs/1410.2293 (visited on 09/28/2025).

[3] E. Boos, *Quantum Field Theory and the Electroweak Standard Model*. CERN, 2016. DOI: 10.5170/CERN-2015-004.1. [Online]. Available: https://doi.org/10.5170/CERN-2015-004.1.

[4] U.S. Department of Energy. "Doe explains the standard model of particle physics." (), [Online]. Available: https://www.energy.gov/science/doe-explains-the-standard-model-particle-physics (visited on 06/28/2025).

[5] H. Wang, "Discovery of the higgs boson by the atlas and cms experiments at the lhc," *Science China Physics, Mechanics & Astronomy*, 2014. DOI: 10.1007/s11433-014-5558-2. [Online]. Available: https://link.springer.com/article/10.1007/s11433-014-5558-2.

[6] CERN. "The accelerator complex." (), [Online]. Available: https://home.cern/science/accelerators/accelerator-complex (visited on 06/28/2025).

[7] "Measurement of the z-boson mass." arXiv: 1807.06325. (2025), [Online]. Available: https://arxiv.org/abs/1807.06325 (visited on 09/28/2025).

[8] F. Alessio, P. Durante, and G. Vouters. "The readout supervisor firmware for controlling the upgraded lhcb detector and readout system." arXiv: 1806.08626. (2018), [Online]. Available: https://arxiv.org/abs/1806.08626 (visited on 09/28/2025).

[9] CERN Courier, "The lhc's worldwide computer," *CERN Courier*, vol. 53, no. 3, pp. 21–24, Mar. 2013, Feature article. [Online]. Available: https://cerncourier.com/a/the-lhcs-worldwide-computer/ (visited on 06/29/2025).

[10] "Dynamic distribution of high-rate data processing from cern to remote hpc data centers," *SpringerLink*, 2013. DOI: 10.1007/s41781-020-00052-w. [Online]. Available: https://link.springer.com/article/10.1007/s41781-020-00052-w (visited on 09/28/2025).

[11] A. C. Smith and A. Tsaregorodtsev, "Dirac: Reliable data management for lhcb," *Journal of Physics: Conference Series*, vol. 119, p. 062 045, 2008. DOI: 10.1088/1742-6596/119/6/062045. [Online]. Available: https://doi.org/10.1088/1742-6596/119/6/062045 (visited on 09/28/2025).

[12] R. Steerenberg. "Lhc report: Full house for the lhc." CERN CDS record. Describes LHC beam conditions, luminosity records, and BCMS beam scheme. (2017), [Online]. Available: https://cds.cern.ch/record/2272573 (visited on 09/28/2025).

[13] A. A. A. Jr. and others (LHCb Collaboration), "The lhcb detector at the lhc," *Nuclear Instruments and Methods in Physics Research Section A: Accelerators, Spectrometers, Detectors and Associated Equipment*, vol. 727, pp. 57–73, 2013. DOI: 10.1016/j.nima.2013.01.005. [Online]. Available: https://doi.org/10.1016/j.nima.2013.01.005, Image reproduced from this publication.

[14] P. Billoir, M. D. Cian, P. A. Günther, and S. Stemmle, "A parametrized kalman filter for fast track fitting at lhcb," *Computer Physics Communications*, vol. 265, p. 108 026, 2021, LHCb-DP-2021-001. DOI: 10.1016/j.cpc.2021.108026. arXiv: 2101.12040. [Online]. Available: https://arxiv.org/abs/2101.12040 (visited on 09/28/2025).

[15] K. Hennessy, "Lhcb velo upgrade," *arXiv*, vol. 1604.05045, 2016, LHCb-DP-2016-001. DOI: 10.48550/arXiv.1604.05045. [Online]. Available: https://arxiv.org/abs/1604.05045 (visited on 09/29/2025).

[16] Y. Li and LHCb UT Upgrade II team, "Maps for the upstream tracker in lhcb upgrade ii," *Nuclear Instruments and Methods in Physics Research Section A: Accelerators, Spectrometers, Detectors and Associated Equipment*, vol. 1032, p. 166 629, 2022, Open Access; Published online 1 June 2022. DOI: 10.1016/j.nima.2022.166629. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S0168900222002017 (visited on 09/29/2025).

[17] A. Falabella, M. Manzali, and U. Marconi, "The 40 mhz trigger-less daq for the lhcb upgrade," *Nuovo Cimento C*, vol. 40, p. 71, 2017, Published online 9 February 2017; Open Access. DOI: 10.1393/ncc/i2017-17071-0. [Online]. Available: https://www.sif.it/papers/?pid=ncc11272 (visited on 09/29/2025).

[18] A. Massafferri, "Scifi, the new tracker of the lhcb experiment," *Journal of Instrumentation*, vol. 15, p. C08006, 2020, Presented at the International Conference on Instrumentation for Colliding Beam Physics (INSTR2020). DOI: 10.1088/1748-0221/15/08/C08006. [Online]. Available: https://iopscience.iop.org/article/10.1088/1748-0221/15/08/C08006 (visited on 09/29/2025).

[19] LHCb Starterkit Lessons. "Lhcb data flow in run 3." Overview of changes to the LHCb trigger, data acquisition, HLT, and persistent streams in Run 3. (2025), [Online]. Available: https://lhcb.github.io/starterkit-lessons/first-analysis-steps/dataflow-run3.html (visited on 09/29/2025).

[20] A. Morris. "First experiences with the lhcb heterogeneous software trigger." Proceedings of the CTD 2023, PROC-CTD2023-51, on behalf of the LHCb collaboration real-time analysis group. arXiv: 2412.05041v1. (2023), [Online]. Available: https://arxiv.org/html/2412.05041v1 (visited on 09/29/2025).

[21] R. Quagliani, "Novel real-time alignment and calibration of lhcb detector for run ii and tracking for the upgrade," *Journal of Physics: Conference Series*, vol. 762, no. 1, p. 012 046, 2016, CC BY 3.0; published October 2016. DOI: 10.1088/1742-6596/762/1/012046. [Online]. Available: https://cds.cern.ch/record/2235193/files/pdf.pdf (visited on 09/29/2025).

[22] D. Derkach, N. Kazeev, R. Neychev, *et al.*, "Lhcb trigger streams optimization," *Journal of Physics: Conference Series*, vol. 898, p. 062 026, 2017, Track 4: Data Handling, published under licence by IOP Publishing Ltd. DOI: 10.1088/1742-6596/898/6/062026. [Online]. Available: https://iopscience.iop.org/article/10.1088/1742-6596/898/6/062026 (visited on 09/29/2025).

[23] L. Eklund. "The lhcb upgrade." Presented at the 11th Conference on the Intersections of Particle and Nuclear Physics (CIPANP 2012), St Petersburg, Florida, US; AIP Conference Proceedings Vol 1560. arXiv: 1709.04709. (2017), [Online]. Available: https://arxiv.org/abs/1709.04709 (visited on 09/29/2025).

[24] V. Kholoimov, B. K. Jashal, A. Oyanguren, V. Svintozelskyi, and J. Zhuo, "A downstream and vertexing algorithm for long lived particles (llp) selection at the first high-level trigger (hlt1) of lhcb," *Computing and Software for Big Science*, vol. 9, p. 10, 2025, Published in Computing and Software for Big Science, V2.0; uses GPU-based neural network algorithms for LLP selection at HLT1. DOI: 10.1007/s41781-025-00141-8. arXiv: 2503.13092. [Online]. Available: https://arxiv.org/abs/2503.13092 (visited on 09/29/2025).

[25] M. Stahl and LHCb Collaboration, "Machine learning and parallelism in the reconstruction of lhcb and its upgrade," *Journal of Physics: Conference Series*, vol. 898, no. 4, p. 042 042, 2017, Proceedings of CHEP 2016; images such as track types figure reproduced. DOI: 10.1088/1742-6596/898/4/042042. [Online]. Available: https://arxiv.org/abs/1710.08947 (visited on 09/29/2025).

[26] A. Maier, "Ganga— a job management and optimising tool," *Journal of Physics: Conference Series*, vol. 119, p. 072 021, 2008, Distributed Data Analysis and Information Management; published under licence by IOP Publishing Ltd; Open Access. DOI: 10.1088/1742-6596/119/7/072021. [Online]. Available: https://iopscience.iop.org/article/10.1088/1742-6596/119/7/072021 (visited on 09/29/2025).

[27] LHCb Starterkit Lessons. "Lhcb data flow." Overview of LHCb dataflow for Runs 1–2 and introductory analysis pipeline. (2025), [Online]. Available: https://lhcb.github.io/starterkit-lessons/first-analysis-steps/dataflow.html (visited on 09/29/2025).

[28] G. Barrand, I. Belyaev, P. Binko, *et al.*, "Gaudi — a software architecture and framework for building hep data processing applications," *Comput. Phys. Commun.*, vol. 140, pp. 45–55, 2001. DOI: 10.1016/S0010-4655(01)00254-5. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S0010465501002545 (visited on 09/29/2025).

[29] C. Bierlich, S. Chakraborty, N. Desai, *et al.*, "A comprehensive guide to the physics and usage of pythia 8.3," *arXiv*, 2022. DOI: 10.48550/arXiv.2203.11601. arXiv: 2203.11601. [Online]. Available: https://arxiv.org/abs/2203.11601 (visited on 09/29/2025).

[30] S. Agostinelli, J. Allison, K. Amako, *et al.*, "Geant4 — a simulation toolkit," *Nucl. Instrum. Meth. A*, vol. 506, pp. 250–303, 2003. DOI: 10.1016/S0168-9002(03)01368-8. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S0168900203013688?via%3Dihub (visited on 09/29/2025).

[31] LHCb Collaboration, "A comparison of cpu and gpu implementations for the lhcb experiment run 3 trigger," *Computing and Software for Big Science*, vol. 6, no. 1, p. 1, 2022, LHCb-DP-2021-003; 30 pages, 15 figures, 8 tables; CC BY 4.0. DOI: 10.1007/s41781-021-00070-2. arXiv: 2105.04031. [Online]. Available: https://cds.cern.ch/record/2766501 (visited on 09/29/2025).

[32] F. I. Giasemis, V. Lončar, B. Granado, and V. V. Gligorov. "Comparative analysis of fpga and gpu performance for machine learning-based track reconstruction at lhcb." Focuses on ML-based track reconstruction in the LHCb first-level trigger using FPGAs and GPUs. arXiv: 2502.02304. (2025), [Online]. Available: https://arxiv.org/abs/2502.02304 (visited on 09/29/2025).

[33] K. Fraser and M. D. Schwartz, "Jet charge and machine learning," *Journal of High Energy Physics*, vol. 2018, no. 10, p. 093, 2018, Uses convolutional, recurrent, and recursive neural networks for jet charge classification. DOI: 10.1007/JHEP10(2018)093. arXiv: 1803.08066. [Online]. Available: https://arxiv.org/abs/1803.08066 (visited on 09/29/2025).

[34] K. Terashi, M. Kaneda, T. Kishimoto, M. Saito, R. Sawada, and J. Tanaka, "Event classification with quantum machine learning in high-energy physics," *Computing and Software for Big Science*, vol. 5, p. 2, 2021, Explores variational quantum algorithms for event classification and compares performance with classical ML methods. DOI: 10.1007/s41781-020-00047-7. arXiv: 2002.09935. [Online]. Available: https://arxiv.org/abs/2002.09935 (visited on 09/29/2025).

[35] E. Kauffman, A. Held, and O. Shadura. "Machine learning for columnar high energy physics analysis." Submitted as CHEP 2023 conference proceedings to EPJ; integration of ML training and inference into the IRIS-HEP Analysis Grand Challenge pipeline. arXiv: 2401.01802. (2024), [Online]. Available: https://arxiv.org/abs/2401.01802 (visited on 09/29/2025).

[36] S. Bacallado and J. Taylor. "Simple linear regression." Stanford STATS 202 course notes; following James et al., ISLR 2nd edition. (2022), [Online]. Available: https://web.stanford.edu/class/stats202/notes/Linear-regression/Simple-linear-regression.html (visited on 09/30/2025).

[37] Wikipedia contributors. "Sigmoid function." Overview of the sigmoid function, its properties, and applications. (2025), [Online]. Available: https://en.wikipedia.org/wiki/Sigmoid_function (visited on 09/30/2025).

[38] A. Singh. "Decision tree in machine learning." Overview of decision tree algorithm, its components, and construction process. (2024), [Online]. Available: https://www.appliedaicourse.com/blog/decision-tree-in-machine-learning/ (visited on 09/30/2025).

[39] T. Keck. "Fastbdt: A speed-optimized and cache-friendly implementation of stochastic gradient-boosted decision trees for multivariate classification." Optimized implementation of stochastic gradient-boosted decision trees; used in high-energy physics, including Belle II. arXiv: 1609.06119. (2016), [Online]. Available: https://arxiv.org/abs/1609.06119 (visited on 09/29/2025).

[40] D. E. Rumelhart, G. E. Hinton, and R. J. Williams, "Learning representations by back-propagating errors," *Nature*, vol. 323, pp. 533–536, 1986, Letter; introduces the back-propagation algorithm for neural networks. DOI: 10.1038/323533a0. [Online]. Available: https://www.nature.com/articles/323533a0 (visited on 09/29/2025).

[41] A. Krizhevsky, I. Sutskever, and G. E. Hinton, "Imagenet classification with deep convolutional neural networks," *Communications of the ACM*, vol. 60, no. 6, pp. 84–90, 2017, Introduces the AlexNet deep convolutional neural network for ImageNet classification, pioneering GPU-based deep learning. DOI: 10.1145/3065386. [Online]. Available: https://dl.acm.org/doi/10.1145/3065386 (visited on 09/29/2025).

[42] Wikipedia contributors. "Neural network (machine learning)." Overview of artificial neural networks, their structure, training, and applications. (2025), [Online]. Available: https://en.wikipedia.org/wiki/Neural_network_%28machine_learning%29 (visited on 09/30/2025).

[43] C. Liu and L. Hui. "Relu soothes the ntk condition number and accelerates optimization for wide neural networks." Analyzes how ReLU activation improves feature separation and NTK conditioning in wide neural networks, accelerating gradient descent convergence. arXiv: 2305.08813. (2023), [Online]. Available: https://arxiv.org/abs/2305.08813 (visited on 09/29/2025).

[44] T. Gneiting and P. Vogel. "Receiver operating characteristic (roc) curves." Introduces beta-family fitting for ROC curves, including concavity constraints and R software for inference. arXiv: 1809.04808. (2018), [Online]. Available: https://arxiv.org/abs/1809.04808 (visited on 09/29/2025).

[45] E. AI. "How to explain the roc curve and roc auc score?" Overview of the ROC curve, its components, and the ROC AUC score in binary classification. (2025), [Online]. Available: https://www.evidentlyai.com/classification-metrics/explain-roc-curve (visited on 09/30/2025).

[46] D. Ma, J. Bortnik, X. Chu, S. G. Claudepierre, A. Kellerman, and Q. Ma. "Opening the black box of the radiation belt machine learning model." Analyzes interpretability of the ORIENT neural network model for Earth's radiation belt electron flux using DeepSHAP explanations. arXiv: 2208.08905. (2022), [Online]. Available: https://arxiv.org/abs/2208.08905 (visited on 09/29/2025).

[47] R. Pezoa, L. Salinas, and C. Torres, "Explainability of high energy physics events classification using shap," *Journal of Physics: Conference Series*, vol. 2438, p. 012 082, 2023, Presented at ACAT 2021; applies SHAP for interpretability of machine learning models in HEP event classification. DOI: 10.1088/1742-6596/2438/1/012082. [Online]. Available: https://doi.org/10.1088/1742-6596/2438/1/012082 (visited on 09/29/2025).

[48] C. A. Beteta, D. Andreou, M. Artuso, *et al.*, "The salt-readout asic for silicon strip sensors of upstream tracker in the upgraded lhcb experiment," *Sensors (Basel)*, vol. 22, no. 1, p. 107, 2021, Describes the SALT ASIC for the Upstream Tracker of LHCb, including architecture, ADC design, DSP implementation, and performance measurements. DOI: 10.3390/s22010107. [Online]. Available: https://pubmed.ncbi.nlm.nih.gov/35009648/ (visited on 09/29/2025).