Western Norway University of Applied Sciences

BACHELOR'S ASSIGNMENT

Applying Machine Learning Detection and Parameter Estimation of the Z' Particle at the LHC
- Analysis Based on the ATLAS' Simulated Collision DATA (HVL)

Bruk av Maskinlæring for Deteksjon og Parameter-estimering av Z'-partikkelen ved LHC
- Analyse basert på ATLAS' Simulerte Kollisjonsdata (HVL)

Hauk Hauge Mo
Lars Erik Risholm
Marius Sellevold Hauger

Bachelor, Dataingeniør
Faculty of Technology, Environmental and Social Sciences
Department of Computer science, Electrical engineering and Mathematical sciences
Supervisor (Dag Toppe Larsen)
13.05.2024

I confirm that the work is self-prepared and that references/source references to all sources used in the work are provided, cf. Regulation relating to academic studies and examinations at the Western Norway University of Applied Sciences (HVL), § 10.

Report Title:	Applying Machine Learning Detection and Parameter Estimation of the Z′ Particle at the LHC
Authors:	Hauk Hauge Mo
	Lars Erik Risholm
	Marius Sellevold Hauger
Field of Study:	Computer Engineering
Contact Person for Field of Study:	Dag Toppe Larsen
Remarks:	None
Client:	HVL ATLAS Group
Client's Contact Person:	Trygve Buanes, Steffen Mæland
Phone:	55 58 70 87, 55 58 77 94

Abstract: This study applies machine learning to detect and analyze the Z′ particle using data from the LHC's ATLAS detector. Phase one employs random forest classifiers for data exploration and feature selection, achieving high accuracy but facing recall limitations. Phase two introduces neural networks to enhance classification of imbalanced data, significantly improving precision and recall. Phase three evaluates classifier results across various mass hypotheses, indicating potential signals but not achieving the 5σ discovery threshold. A regression model for mass estimation is then developed. While effective in controlled environments, the regression models showed diminished performance on new datasets, highlighting the need for improvements in model adaptability and predictive robustness. This research successfully integrates machine learning with particle physics, enhancing our capacity to analyze and interpret high-energy collision data and advancing our understanding of the universe's fundamental components.

Keywords: Fundamental Particle Physics, ATLAS, CERN, Machine Learning and Neural Networks

Date: 13.05.2024
Number of pages without appendices: 64
Number of pages of appendices: 11
Number of disks/CDs: 0
Client's Reference: None

Western Norway University of Applied Sciences, Faculty of Engineering and Science
Mailing Address: P.O. Box 7030, 5020 BERGEN
Visiting Address: Inndalsveien 28, Bergen
Phone: 55 58 75 00 Fax: 55 58 77 90 Email: post@hvl.no Homepage: www.hvl.no

Acknowledgements

We extend our deepest gratitude to the ATLAS group at the Western Norway University of Applied Sciences (HVL) for their invaluable guidance and insights. Their expertise in particle physics and machine learning has significantly supported our research. A special thanks to our supervisor, Dag Toppe Larsen, for his advice and encouragement, which have been crucial to our project's development.

We are also thankful to our peers and the educational resources that have played a significant role in our learning journey, especially the FastAi course for its comprehensive material on machine learning and deep learning techniques.

To our friends and family, we express our heartfelt appreciation for your unwavering support and encouragement throughout this journey. Your belief in our abilities has been a constant source of motivation.

A special thanks to Maya Silva, for her expert advice on thesis writing and Overleaf. As a scientist, her guidance has been key to our project's success, providing clarity and direction when we needed it most.

Additionally, we acknowledge the use of AI-powered tools such as Quillbot AI and OpenAI's ChatGPT for assistance in proofreading and refining the text of this thesis. These tools have helped improve the clarity and readability of our work. The final review and the content, however, remain our responsibility, ensuring the integrity of our academic output.

Finally, we acknowledge the excellent teamwork of our group members in Group D 26. The collaborative spirit, shared goals, and mutual respect within our team have been instrumental in achieving our research objectives.

Thank you,

Group D 26
13.05.2024

Abstract

Background: This research explores the application of machine learning to high-energy physics, focusing on the Z' particle using data from the ATLAS detector at the LHC. The study addresses gaps in the Standard Model of particle physics, particularly in understanding phenomena like dark matter and other elusive components of the universe.

Materials and Methods: Employing a structured, phased approach, the methodology integrates both conventional machine learning models and sophisticated neural networks to address the challenges of high-energy particle physics data. In phase one, data exploration and feature selection are conducted using random forest classifiers to establish a baseline for model performance and data characteristics. Phase two transitions to neural networks, which are designed to capture more complex patterns within the data, enhancing the sensitivity and specificity of the models. The final phase, phase three, analyzes classifier results across a spectrum of mass hypotheses scenarios and develops a regression model for mass estimation of the Z' particle, aiming to leverage the predictive power of machine learning to derive meaningful physical insights from the collision data.

Results: Initial results show that random forest classifiers achieved high accuracy (nearly 99.5%), though with some limitations in recall. Phase two's shift to neural networks, aided by undersampling, significantly enhanced precision and recall. In phase three, while the classifier highlighted potential signals with a peak significance of 1.7751σ at 500 GeV, it fell short of the 5σ discovery threshold. The regression model estimated the Z' particle mass effectively within controlled environments, showing low mean squared error and high R-squared values, but struggled with generalizing to new datasets. Notably, the inclusion of invariant mass calculations improved feature impact. These results, however, underscore the need for further refinement of the models to enhance their adaptability and reliability in diverse experimental conditions.

Conclusion: The integration of machine learning into particle physics research, particularly through the ATLAS experiment at the LHC, has significantly advanced the analytical capabilities available to physicists. The models developed during this study hold the promise of deepening our understanding of the universe's most fundamental structures. Future work should focus on refining such models to improve their adaptability and accuracy, aiming to deepen our understanding of fundamental particles and the universe's structure.

Glossary

Particle Physics

Bosons: Particles that follow Bose-Einstein statistics, including force carriers like photons and the Higgs boson, enabling multiple bosons to occupy the same quantum state.

Fermions: Particles constituting matter, following Fermi-Dirac statistics, which prohibits them from sharing the same quantum state, including quarks and leptons.

Higgs Boson: A fundamental particle responsible for imparting mass to other particles through the Higgs mechanism, discovered at the LHC in 2012.

Quarks: Elementary particles, such as quarks, combine to form hadrons, which include protons and neutrons. These particles interact strongly through the exchange of gluons.

Leptons: A class of elementary particles that do not undergo strong interactions, including electrons and neutrinos.

Dark Matter: A hypothetical form of matter that does not interact through the strong electromagnetic force. It is inferred from its gravitational effects on visible matter and the structure of the universe.

Standard Model: The prevailing theory describing the fundamental particles and their interactions, excluding gravity.

Z' Particle: A hypothetical massive boson predicted by extensions of the Standard Model, potentially mediating new fundamental forces.

Cross-Section: A measure of the probability that a particular interaction process will occur, often used in the context of particle collisions.

Pseudorapidity (η): A coordinate describing angles relative to the beam axis in particle physics, important in describing the geometry of particle collisions.

GeV: Giga Electron Volt, a unit of energy in the field of high-energy physics, 1×10^9 electron volts.

Azimuthal direction: The angular coordinate in the plane perpendicular to the beam axis, often used to describe the direction of particle emissions in collider experiments.

Jet: A stream of particles produced in high-energy processes such as proton-proton collisions at the LHC.

Hadron: A composite particle made of quarks held together by the strong force, as in protons and neutrons.

Gluons: Elementary particles that act as the exchange particles for the strong force between quarks, analogous to the photon's role in electromagnetic interactions.

Transverse momentum: The component of momentum perpendicular to the direction of motion of an object.

Phi (ϕ): An angle measured in radians that represents the azimuthal direction around the beam axis in particle physics. It is often used to describe the orientation of particles or the direction of their motion in collider experiments.

Sigma Levels (σ): A statistical measure used in particle physics to denote the confidence level of a result, commonly used to assess the significance of an observed effect relative to the background.

Invariant Mass: A physical quantity in particle physics that describes the mass of the system of particles, important

in identifying particle decays and interactions.

Decay Products: The particles resulting from the decay of another particle, crucial for detecting and studying particles that cannot be observed directly.

Machine Learning

Convolutional Neural Network (CNN) Architecture: A deep learning architecture optimized for processing structured arrays of data, such as images, by applying convolutional layers.

Gradient Descent: An optimization algorithm used to minimize the cost function in machine learning models by iteratively moving towards the minimum value.

Gradient Boosting: A machine learning technique used for regression and classification problems, which produces a prediction model in the form of an ensemble of weak prediction models, typically decision trees.

Activation Function: Functions applied at each node in a neural network to determine the output, such as ReLU or sigmoid, introducing non-linearity.

Epoch: A complete pass through the entire training dataset, used in the context of training an algorithm.

Overfitting: When a model learns the training data too well, including its noise and outliers, leading to poor performance on new, unseen data.

Regularization: Techniques such as L1 and L2 regularization are employed to mitigate overfitting by introducing a penalty term based on the magnitude of model parameters.

Learning Rate: A hyperparameter that controls how much we are adjusting the weights of our network with respect to the loss gradient.

Loss Function: A function that maps values of one or more variables onto a real number, representing some "cost" associated with the event.

RandomSplitter: A method used in data splitting, particularly in machine learning, to randomly partition a dataset into training and validation sets. This helps in assessing the generalization performance of a model.

Feature Engineering: The process of using domain knowledge to select, modify, or create new features from raw data to increase the predictive power of machine learning algorithms.

Cross-Validation: A technique used to evaluate the generalizability of a model, involving multiple rounds of partitioning the data into complementary subsets, performing training and validation analyses.

Transfer Learning: A research problem in machine learning that focuses on storing knowledge gained while solving one problem and applying it to a different but related problem.

SMOTE (Synthetic Minority Over-sampling Technique): A statistical technique for increasing the number of cases in your dataset in a balanced way. SMOTE works by creating synthetic samples rather than by oversampling with replacement. It is particularly useful in scenarios where the minority class in a dataset is underrepresented.

Miscellaneous

LHC (Large Hadron Collider): The world's largest and most powerful particle accelerator, located at CERN, used to investigate fundamental particles by colliding them at high energy.

ATLAS (A Toroidal LHC ApparatuS): A key experiment at the LHC designed to observe phenomena that occur when proton beams collide at high energy. It investigates a wide range of physics, from the search for the Higgs boson to extra dimensions and particles that could make up dark matter.

Kaggle: An online community and platform for data scientists and machine learning practitioners. Kaggle allows users to find and publish data sets, explore and build models in a web-based data-science environment, and work with other data enthusiasts to solve data science challenges.

RAM (Random Access Memory): A form of computer data storage that stores data and machine code currently being used. RAM allows data items to be read or written in almost the same amount of time irrespective of the physical location of data inside the memory.

CERN (European Organization for Nuclear Research): The world's largest particle physics laboratory, located in Geneva, Switzerland. It is renowned for its research into the fundamental particles of the universe using the Large Hadron Collider (LHC), the world's largest and most powerful particle accelerator.

HVL (Western Norway University of Applied Sciences): A Norwegian public university of applied sciences, established in its current form in 2017 through the merger of several former independent colleges. HVL conducts research and offers higher education in a wide range of disciplines, including engineering, health and social sciences, maritime studies, and teacher education.

Contents

List of Figures

List of Tables

1 Introduction

The chapter discusses high-energy physics, the importance of the LHC, and the hypothesised Z′ particle, with an emphasis on teamwork and organisation. It then discusses theoretical foundations and data analysis methodologies.

1.1 Context

Particle physics is a key part of modern science. It aims to understand the universe by studying the underlying building blocks of matter and how they interact. The Large Hadron Collider (LHC) at CERN is the world's most powerful particle accelerator [1]. The ATLAS experiment at the LHC seeks to investigate the most fundamental aspects of the universe by studying high-energy collisions, including search for new particles such as the theoretical Z′ particle.

Figure 1.1.1: Workers at the tunnel during long shutdown [2]

The Z′ particle, also known as the Z prime boson, is a hypothetical extension of the Standard Model [3], posited to mediate interactions beyond those known within the current framework, hinting at new fundamental forces or components of matter. Its study, propelled by advanced theoretical models and experimental searches at facilities like the LHC, stands at the forefront of efforts to unravel mysteries surrounding dark matter, the matter-antimatter asymmetry, and other phenomena not adequately explained by existing theories. The pursuit of the Z′ boson encapsulates the dynamic interplay between theory and experiment in particle physics, aiming to broaden the understanding of the universe's fundamental structure [4].

Parallel to this, the use of Artificial Intelligence (AI), most notably machine learning (ML) in particle physics, particularly through experiments like ATLAS at the Large Hadron Collider (LHC), is revolutionizing the field [5]. These technologies are essential for analyzing the complex data produced by high-energy collisions, aiding in the detection and analysis of phenom-

ena beyond the Standard Model. By applying advanced ML techniques, such as deep neural networks, researchers can sift through large datasets to identify rare particle interactions, significantly impacting the approach to understanding particle physics.

1.2 Motivation

The exploration of new physics phenomena, such as the Z' particle, marks a frontier in understanding the universe. This project is propelled by the ambition to utilize ML and DL methodologies to detect signals of the Z' particle, potentially resolving some unanswered questions in physics. These encompass the nature of dark matter and the reasons behind the imbalance between matter and antimatter. Discoveries in this area could significantly influence theoretical frameworks and experimental techniques.

The research aims to showcase the efficiency of combining computational sciences and physics to improve the identification and analysis of occurrences beyond the Standard Model. This seeks to broaden the scope of particle physics research while emphasising the importance of multidisciplinary approaches in scientific discovery.

1.3 Project owner

The ATLAS group at the Western Norway University of Applied Sciences (HVL) is at the forefront of research into dark matter, utilizing the Large Hadron Collider (LHC) at CERN to explore this unsolved mystery of physics [6]. This dedicated team is composed of faculty members and researchers, each contributing unique expertise and insights to the collective endeavor.

Therese Berge Sjursen, an associate professor in physics at HVL, leads the project with a focus on data analysis and the search for dark matter as part of the ATLAS experiment at CERN. Trygve Buanes, another associate professor at HVL, contributes his extensive background in physics, emphasizing the analysis of data from the ATLAS experiment. Steffen Mæland, also an associate professor, applies his expertise in physics and computer science to the group's research, particularly in analyzing ATLAS data and exploring CP violation in the Higgs sector.

Igor Slazyk, currently a postdoctoral research fellow at HVL and stationed at CERN, specializes in the implementation of machine learning algorithms for physics analyses within the ATLAS experiment. Tarje Solberg Hillersøy, a PhD candidate, is also at CERN, working on applying supervised and unsupervised machine learning techniques to aid physics analyses at the ATLAS detector.

Aurora Grefsrud, another PhD candidate, focuses her research on employing computer vision inspired by machine learning to analyze ATLAS detector data. Dag Toppe Larsen, an associate professor, brings a cross-disciplinary approach to the team, with a keen interest in artificial intelligence and its application in physics experiments, particularly ATLAS.

The group maintains a rigorous schedule of weekly meetings to coordinate their research activities, discuss findings, and plan future experiments. This routine ensures that all members, including those currently based at CERN, are closely aligned with the group's objectives and progress.

Through their collaborative efforts, the ATLAS group at HVL aims to contribute significantly to the global understanding of dark matter [6], combining advanced computational techniques with experimental physics to uncover new insights into the fundamental structure of the universe.

1.4 Problem description and goals

The primary objective of this study is to employ sophisticated machine learning techniques to develop classifiers that can accurately differentiate Z′ particle signals from Standard Model (SM) background processes using data from the ATLAS detector at the LHC. A secondary aim is to estimate the Z′ particle's mass from its decay products, enhancing our analytical capabilities in particle physics and expanding our understanding of the universe's fundamental structure.

The research will progress through systematically designed phases, each building on the data and insights gained from the previous, to deepen the integration of particle physics and machine learning techniques.

1.4.1 Phase one - Classifier development with random forest

The first phase involves extensive data exploration to identify patterns, distributions, and anomalies within the datasets. This analysis involves both feature selection and the training of random forest models, selected for their effectiveness with tabular data and their inherent resistance to overfitting.

1.4.2 Phase two - Advancing with neural networks

This phase builds upon the insights gained from the random forest models by transitioning to a neural network approach. Tabular neural networks with multiple layers will be employed to detect more subtle patterns and enhance sensitivity to Z′ particle signals. Thorough data preparation will be conducted to optimize the models' performance in identifying these signals.

1.4.3 Phase three - Classifier evaluation and regression model development

In the final phase, the focus shifts from the development and optimization of classification models to their application in performance evaluation across hypothesized mass points for the Z′ particle. This involves utilizing the most effective classifier to test datasets containing signal events at mass values ranging from 500 to 2500 GeV. This step is designed to assess whether the data at these specific mass points supports the hypothesis of the Z′ particle's existence and to

evaluate the classifier's performance across a spectrum of mass scenarios.

Subsequently, a regression model will be developed to estimate the mass of the Z' particle based on its decay signatures. Leveraging detailed features such as energy, momentum, and angular distributions from decay products, this model aims to refine mass estimation techniques to ensure the accurate and reliable identification of the Z' particle's mass, thereby enhancing the understanding of its properties and the underlying physical processes.

Finally, to validate their robustness and predictability, both the classifier and regression model will undergo further testing on new, unseen datasets. The objective of this process is to simulate real-world conditions in order to gain a deeper understanding of their performance outside of controlled experimental settings.

1.4.4 Scientific questions

Based on the previously discussed objectives, the problem descriptions can be defined as follows:

- *"How can machine learning classifiers be developed and optimized to accurately classify Z' particle signals from Standard Model background processes using data from the ATLAS detector at the LHC, and how does classifier performance evaluate across different mass hypotheses of the Z' particle?"*
- *"What machine learning techniques can be applied to estimate the mass of the Z' particle from its decay products, and how does the accuracy of these estimations vary with the hypothetical mass ranges of the Z' particle?"*

In summary, the project will utilize sophisticated machine learning techniques to tackle the challenge of identifying and analyzing the Z' particle within particle physics. The method will advance systematically from data exploration and classifier development to parameter estimation, aiming to merge theoretical physics with computational methods. This strategy is anticipated to improve the search for the Z' particle, particularly by accurately classifying signal files and distinguishing them from background files, and estimating its mass based on the characteristics of decay products from high-energy collisions. Additionally, it aims to highlight the importance of machine learning in particle physics research, contributing to advancements in both fields.

1.5 Previous work

The project's approach benefits from a series of interconnected studies combining machine learning with high-energy particle physics, notably on classifying particle collision data from the ATLAS experiment at the Large Hadron Collider.

The foundational understanding is significantly enhanced by previous bachelor theses from the university, notably:

- *Civilgins, V. S., & Lothe, S. S.* study on leveraging symmetry to augment training data for classifying microscopic black holes and sphalerons using ATLAS simulated collision data [7].

- *Ganjei, S., & Gunleiksrud, D. K.* investigation into binary classification of microscopic black holes and sphalerons, optimized through custom loss functions [8].

  These works provide critical insights into the application of machine learning techniques within the particle physics domain and have helped shape the initial research direction and methodology.

- Advancement through a collaborative paper titled "Machine Learning Classification of Sphalerons and Black Holes at the LHC" by Grefsrud et al. [9], which delves into the application of machine learning models like XGBoost and Residual Convolutional neural networks for distinguishing between black hole and sphaleron events at the LHC. This study, incorporating results from earlier bachelor projects, extends the understanding of machine learning's potential to identify complex particle interactions, guiding the exploration of the Z' particle detection.

Lastly, this project also builds upon cooperation with an individual project, "Search for new physics at the Large Hadron Collider (LHC)," by Lars Erik Risholm and Marius Sellevold Hauger [10].

1.6 Resources

The study uses a wide range of materials to support its research objectives and methodology. These resources include:

- **Simulated datasets**
  A comprehensive collection of simulated Monte Carlo (MC) datasets from CERN Open Data [11] is utilized. These datasets encapsulate a wide spectrum of particle interactions within the ATLAS detector at the Large Hadron Collider (LHC) [1], enabling effective training and validation of machine learning models.

- **Computational tools and frameworks**
  The analysis relies on various computational tools and frameworks, including Python libraries such as scikit-learn [12], pandas [13], FastAi [14], and matplotlib [15]. These tools aid in data manipulation, model development, training, and evaluation, facilitating efficient implementation and iteration of machine learning algorithms.

- **Kaggle platform**
  The research is primarily conducted on the Kaggle platform [16], which provides robust computational resources, a user-friendly environment, and data safety features. Kaggle

serves as an ideal environment for prototyping, testing, and refining machine learning models within a collaborative and supportive community.

- **Research collaboration**
  Collaboration with the ATLAS research group at the Western Norway University of Applied Sciences offers access to domain expertise and insights into ongoing research efforts in particle physics and machine learning. This collaboration enriches the depth and relevance of the study.

1.7 Literature

The research is grounded on a robust body of scholarly works, with a particular emphasis on the intersection of particle physics and machine learning. Initial directions and methodological choices were informed by bachelor theses that have also been discussed in Section 1.5, which explore machine learning applications in the context of ATLAS collision data.

These theses have provided valuable introductions to the areas of particle physics and machine learning, vital in advancing the study, offering inspiration and essential references that kick-started understanding, allowing for effective definition of problem descriptions, goals, and decisions on subsequent steps.

Furthermore, the theoretical framework and methodological approach of the project are significantly influenced by key texts in the fields of particle physics and machine learning:

- *"Jet Physics at the LHC: The Strong Force beyond the TeV Scale"* by Klaus Rabbertz [17], aids in understanding jet physics and the strong force, which are fundamental to the research focus.
- *"Particle Physics: A Very Short Introduction"* by Frank Close [18], provides a concise and accessible overview of core principles and challenges in particle physics.
- *"Modern Machine Learning and Particle Physics"* by Schwartz M. [5], bridges the gap between computational techniques and practical applications in particle physics.

Additionally, resources from the CERN Open Data portal [11] and associated explanations have been invaluable in understanding the data context and structure used in the study. This combination of literature from computational and physical sciences supports an interdisciplinary approach, aiming to enhance the synergy between machine learning and particle physics for new frontiers such as the Z' particle exploration.

By integrating insights from these foundational studies, the project contributes to the ongoing exploration of particle physics through machine learning, advancing the understanding of phenomena beyond the Standard Model.

1.8 Limitations

Despite a comprehensive approach and the use of diverse resources, the project acknowledges certain limitations and challenges.

- **Limited knowledge in high-energy particle physics**
  Despite access to extensive literature and guidance from the ATLAS group, understanding of high-energy particle physics continues to evolve. Intricacies of particle interactions and theoretical frameworks present significant challenges, necessitating ongoing learning and collaboration for effective comprehension and application.

- **Algorithm selection bias**
  Selection of machine learning algorithms and techniques brings inherent biases that could affect study outcomes. While widely used algorithms like random forest and neural networks are chosen, exploration of alternative approaches or ensemble methods might provide differing results and insights.

- **Imbalanced datasets**
  Imbalanced nature of datasets, especially in terms of Z' particle signals versus background events, presents challenges for model training and evaluation. Despite the use of techniques such as oversampling and undersampling, class imbalance remains an intricate and persistent area of research.

- **Computational limitations**:
  Large sizes of datasets present challenges in RAM utilization and training time. Data subsets are used during training phases to mitigate these challenges, allowing for more manageable processing and analysis.

- **Theoretical assumptions**
  The study is based on theoretical assumptions and models that are used to interpret experimental results and validate machine learning predictions. Variations in theoretical frameworks or uncertainties in predictions could add ambiguity or introduce limitations to the analysis.

These limitations are acknowledged to clearly share the findings and support open discussion, encouraging further research in particle physics and machine learning.

1.9 Particle physics

In order to comprehend the subsequent analyses utilizing machine learning techniques in this thesis, it is important to first grasp the fundamentals within the context of particle physics.

1.9.1 High-energy proton collisions

Proton-proton collisions at the LHC serve as a microscopic crucible for recreating the conditions a mere moment after the Big Bang, allowing physicists to probe the fundamental forces and particles of nature. In these high-energy collisions, protons are accelerated to velocities near the speed of light and smashed together, yielding a shower of particles. This process, pivotal to experiments like ATLAS, enables the observation of rare phenomena and the potential discovery of new particles, such as the elusive Higgs boson or theorized long-lived particles (LLPs). The data harvested from these collisions are immense and complex, necessitating advanced machine learning techniques to sift through and identify signals of new physics amidst the vast background noise [19].

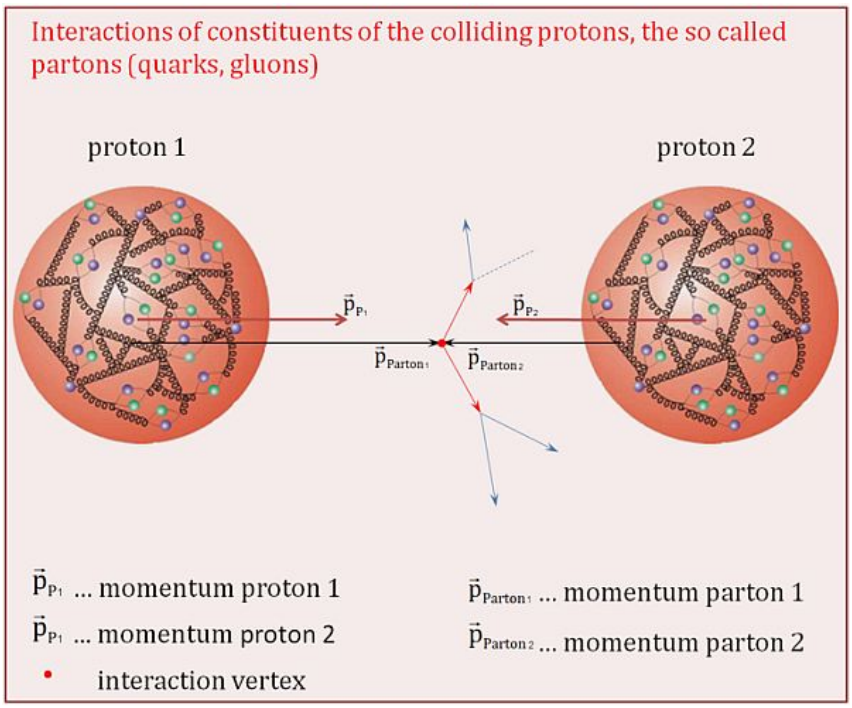


Figure 1.9.2: Two protons about to collide [19].

1.9.2 The Standard Model

The Standard Model posits that matter is composed of fundamental particles called quarks and leptons, which interact through exchange particles known as gauge bosons. There are six types of quarks (up, down, charm, strange, top, and bottom), and six types of leptons (electron, muon, tau, and their corresponding neutrinos), all of which are fermions. The gauge bosons, which are bosons, include the photon, W and Z bosons, and the gluons. The Higgs boson, discovered at CERN in 2012, is also a major component of the Standard Model as it is associated with the Higgs field, which gives mass to other particles [3].

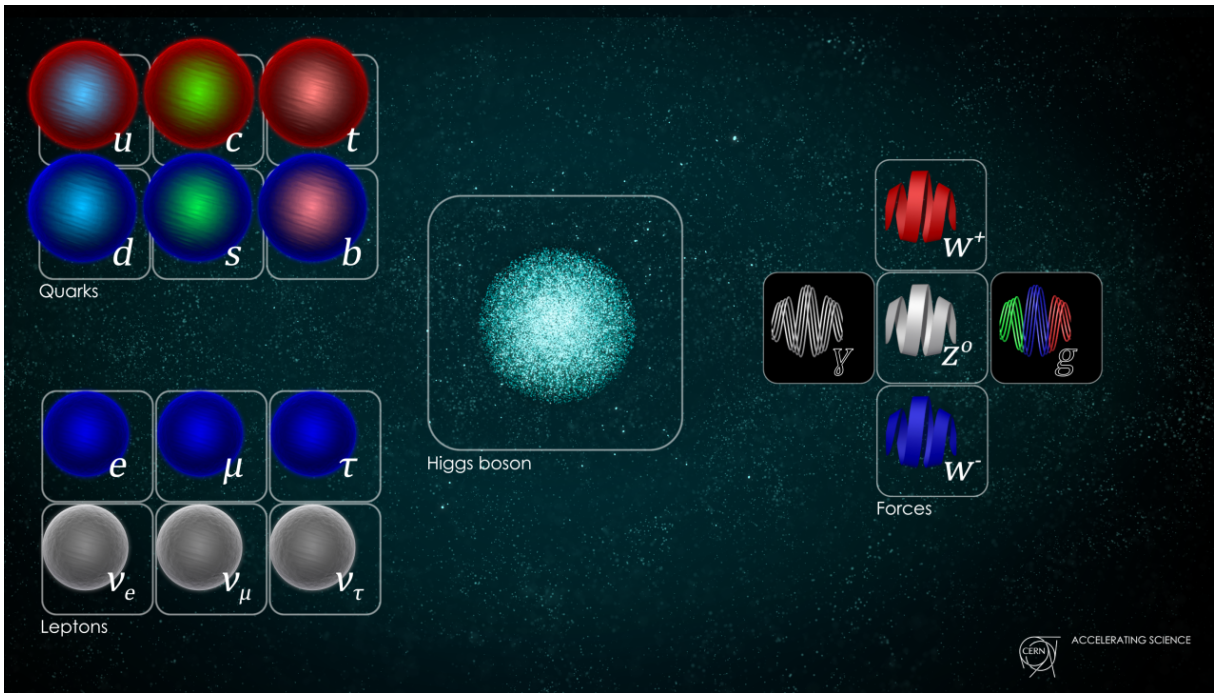


Figure 1.9.3: The Standard Model of particle physics, showing the fermions (quarks and leptons), gauge bosons (force carriers), and the Higgs boson [3].

1.9.3 Quarks and Leptons

Expanding on the Standard Model's basic concepts, quarks and leptons have special characteristics that are crucial for studying particle physics. Quarks are always found within hadrons due to the 'color force' and never exist alone; this fact causes jets to form during high-energy collisions, which are vital for experimental analysis. Leptons, on the other hand, don't interact with the strong force, making them give clearer signals in detectors and helping to distinguish between different interactions, particularly during high-energy collisions.

Additionally, understanding how quarks and leptons differ in mass and charge is key for exploring the universe's symmetry principles and the unusual behaviors leading to theories about things like the Z' boson. Their roles in proton-proton collisions, which are modeled and studied, are vital for spotting signs of new physics beyond the Standard Model. This thesis uses the distinct properties of these particles, combined with advanced machine learning methods, to improve the detection of the hard-to-find Z' boson, shedding light on its potential mass and how it interacts. This thorough study highlights how quarks and leptons work together in the ongoing effort to understand the universe's structure [18].

1.9.4 Jets in particle physics

Jets are streams of particles produced in high-energy processes such as proton-proton collisions at the LHC. They result from the hadronization of quarks and gluons ejected in these events, providing direct insights into the strong force and quark-gluon dynamics as illustrated in 1.9.4.

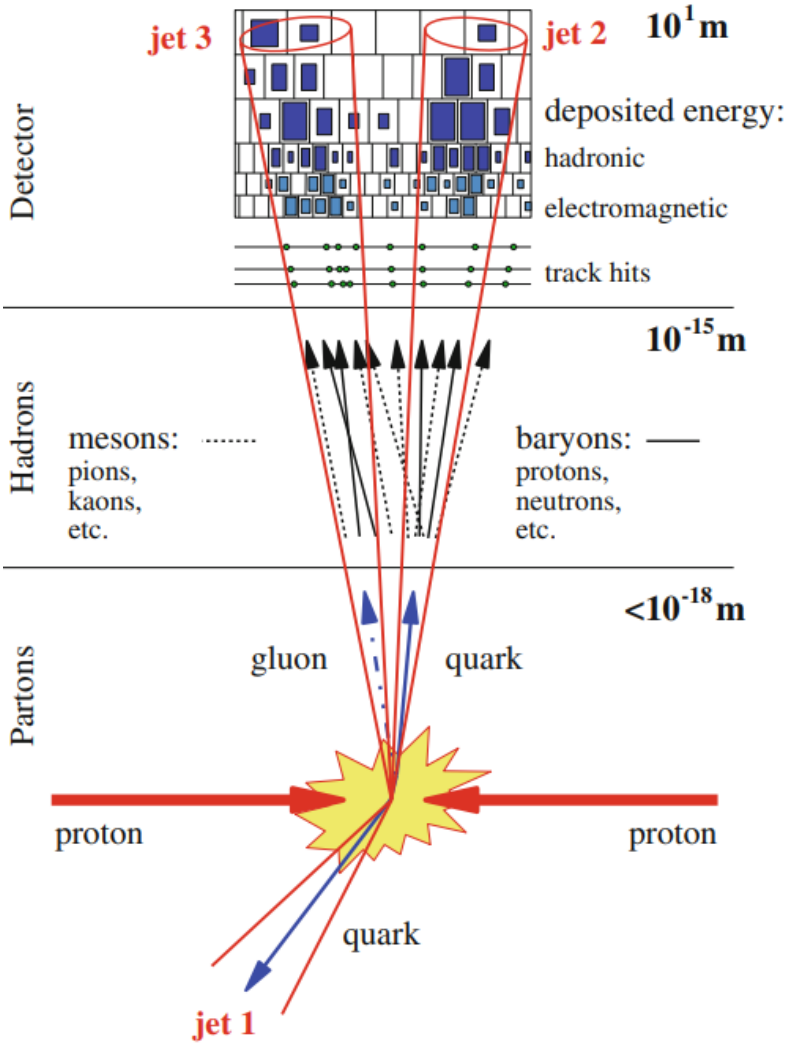


Figure 1.9.4: An illustration of a jet formed by the combination of partons, hadrons, or detector signals [20].

1.9.5 The Z' Boson: Beyond the Standard Model

The Z′ Boson emerges from theoretical extensions to the Standard Model, suggesting a realm of particle physics filled with untapped interactions and particles. This conjectured entity diverges from the established Z boson by possessing a significantly greater mass and potentially mediating unknown forces. The pursuit of the Z′ boson involves high-energy collision experiments designed to detect its unique signatures, a task for which advanced machine learning techniques are increasingly vital [21].

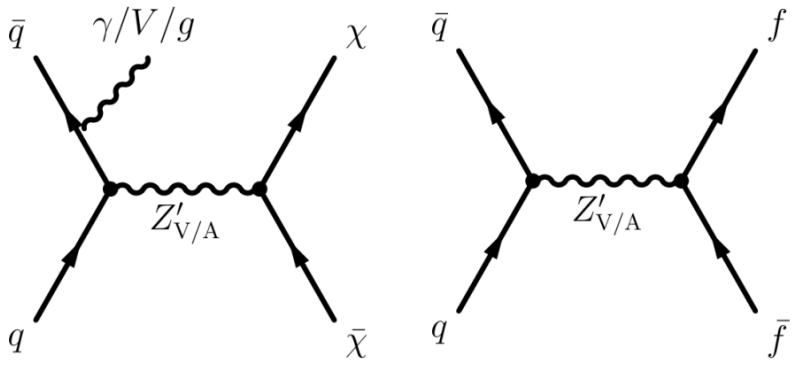


Figure 1.9.5: Dominant production and decay pathways of the Z' mediator in a simplified model, illustrating s-channel vector or axial-vector interactions [22].

The Feynman diagrams in Figure 1.9.5 detail specific instances of Z′ boson interactions. They depict the annihilation of a quark and an antiquark leading to the formation of a Z′ boson, a key process that underscores the potential high-energy interactions capable of producing new bosonic fields. Additionally, they differentiate between the decay of the Z′ into standard model particles and its potential to interact with dark matter particles (χ), thus highlighting the Z′ boson's significant role in expanding current understanding of particle interactions beyond the existing models [23].

1.10 Machine learning in particle physics

This section covers the fundamental concepts of machine learning, an interdisciplinary topic that combines computer science, statistics, and domain knowledge. Machine learning's adaptability and predictive capability make it essential for analyzing complex and large amounts of data produced by experiments such as ATLAS at CERN. Furthermore, machine learning approaches are critical in developing classifiers capable of detecting new physics events, such as the hypothetical Z′ particle, within these datasets.

1.10.1 Artificial Intelligence (AI)

Artificial Intelligence (AI) is the science and engineering of creating intelligent machines that can simulate human thinking and decision-making processes. This includes capabilities such as reasoning, learning, problem-solving, and understanding language [24]. A schematic model is illustrated in figure 1.10.6.

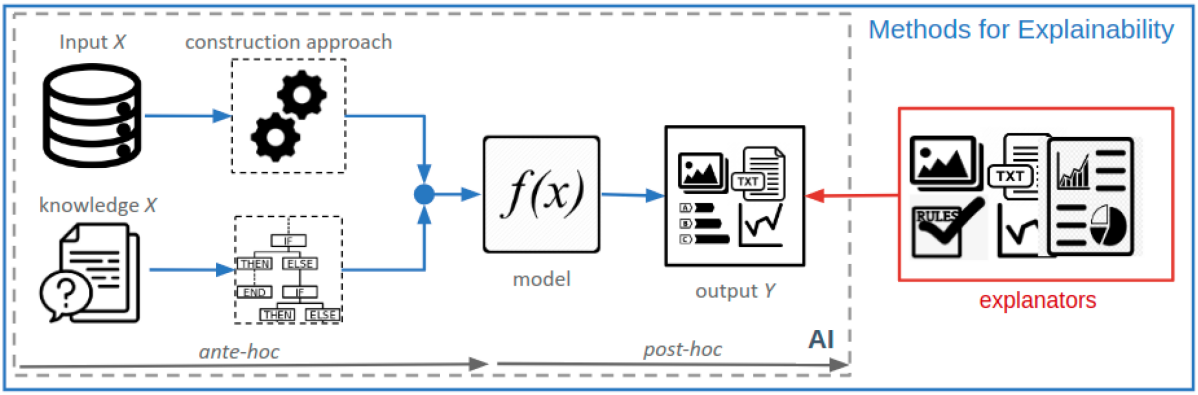


Figure 1.10.6: Workflow and Explainability Methods in AI Modeling. Illustration taken from [25].

Machine learning techniques

Machine learning represents a major subclass of artificial intelligence, characterized by its ability to enable systems to autonomously learn from data and identify patterns, with minimal external input. Integrating disciplines like computer science, statistics, and domain-specific expertise, this field is instrumental in processing the extensive datasets encountered in scientific investigations. As illustrated in Figure 1.10.7, machine learning can be divided into supervised learning, involving methodologies such as Binary Classification and Regression, and unsupervised learning, known for techniques like Clustering. Each plays a distinct role depending on the data analysis objectives.

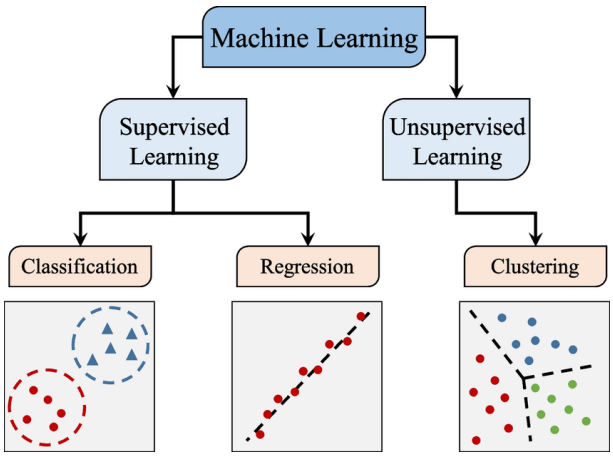


Figure 1.10.7: Overview of Supervised and Unsupervised Learning techniques in Machine Learning. Image collected from [26].

1.10.2 Supervised learning

Supervised learning is a central methodology in machine learning, involving learning a mapping between a set of input variables (X) and an output variable (Y), typically used for classifying data or predicting outcomes [27]. This method is particularly relevant to this study as it involves inferring a function from labeled training data to predict discrete outcomes, such as

distinguishing between signal and background events in particle collision data.

1.10.3 Binary classification

Binary classification, a specific approach within supervised learning, is particularly relevant to the project's research as it involves categorizing outcomes into two distinct groups, such as differentiating between signal and background events in particle collision data. In binary classification, outcomes are labeled as true (positive) or false (negative), which directly aligns with the objective of distinguishing potential Z′ particle signals from standard model background noise in the datasets. This method enables the machine learning models to make precise, binary decisions, enhancing the accuracy of particle identification and event classification within the complex datasets generated by high-energy physics experiments [28].

1.10.4 Regression

Regression analysis is a fundamental statistical technique used in particle physics to model and analyze the relationships between variables. Unlike classification, which divides data into discrete labels, regression aims to predict a continuous quantity. This makes it an indispensable tool for estimating key physical parameters from experimental data, such as the energies, momenta, and, most importantly, the masses of particles such as the Z′ boson.

In the exploration of the subatomic world, accurately determining the mass of particles is vital for verifying existing theories and discovering new physics. For instance, to estimate the mass of hypothetical particles like the Z′ boson, regression models are employed to analyze the decay products or jets emanating from particle collisions. These jets carry essential information about the mass of the originating particle. By modeling the relationship between the observed characteristics of jet decay -such as energy, momentum distribution, and decay angles -and the mass, regression allows us to infer the mass of particles that cannot be directly observed.

1.10.5 Random forests

Random forests are an ensemble learning method that operates by constructing multiple decision trees during training [27]. Their robustness and ease of interpretation have made them a popular choice for classification problems in high-energy physics, where they are used to identify patterns and classify events in complex datasets.

1.10.6 Neural networks

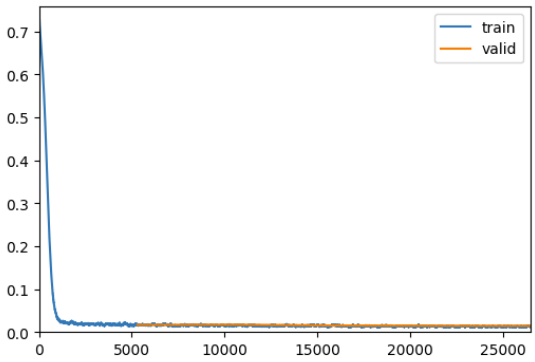
Neural networks, particularly deep learning models, are integral to processing the high-dimensional and complex data in high-energy physics [29]. These models, inspired by the structure and function of the human brain, are capable of learning intricate patterns from large datasets, making them highly effective for identifying potential new particle events, such as those from the hypothetical Z′ particle.

1.10.7 Epochs

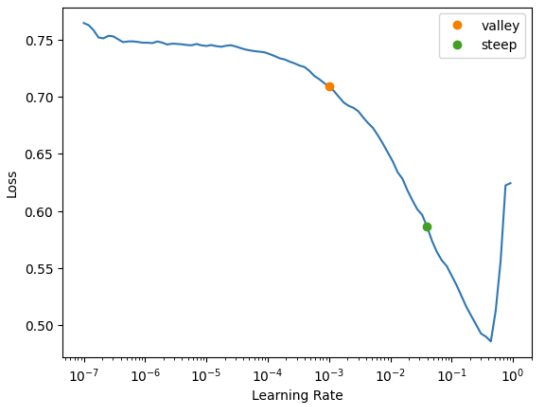
In the neural network training process, an epoch represents a complete pass through the entire dataset [30]. Multiple epochs are used to ensure the model sufficiently learns from the data, balancing the risk of underfitting against overfitting. The number of epochs is chosen based on validation performance, ensuring the model generalizes well to unseen data while capturing the underlying distributions of particle collision events.

1.10.8 Optimization of learning rate using FastAI's lr_find

Optimizing the learning rate is necessary for efficient training of neural networks. The `lr_find` tool from FastAI offers a robust method for this purpose, utilizing a technique that progressively tests a range of learning rates to observe corresponding changes in the loss [31]. As depicted in Figure 1.10.8i, the first plot displays training and validation losses across epochs. The blue line represents the training loss, which generally decreases over time as the model learns from the training data. In contrast, the orange line represents the validation loss, which should ideally decrease alongside the training loss but stabilize or slightly increase when the model begins to overfit the training data. The second plot, Figure 1.10.8ii, maps the loss against learning rates on a logarithmic scale. In this plot, the 'valley' (orange dot) represents the point where the loss is minimized, suggesting an optimal learning rate for stable training. The 'steepest' decline (green dot) indicates where the loss decreases most rapidly, which can be used to identify a more aggressive learning rate that potentially accelerates convergence but may risk instability.



(i) Training and validation losses, with the training loss decreasing and validation loss plateauing, indicating potential overfitting.



(ii) Loss vs. learning rate, showing optimal (valley) and aggressive (steepest) learning rate.

Figure 1.10.8: Visualization of training dynamics and learning rate optimization using FastAI's Learning Rate Finder tool.

These visuals aid in selecting a learning rate that allows the model to learn efficiently without diverging. However, they should be viewed as initial guides rather than definitive answers. The results can vary and often require further empirical testing to confirm and refine the optimal learning rate settings, ensuring the training process is truly optimized.

1.10.9 Overfitting and underfitting

In the context of particle physics and the study on the Z′ particle, overfitting and underfitting are important considerations. Overfitting occurs when the model, for example a deep neural network, learns the training data too precisely, incorporating its noise and outliers. This can lead to decreased effectiveness in generalizing from simulated datasets to real-world collider data, potentially resulting in inaccurate detection of new physics events. Underfitting, conversely, arises when the model is too simplistic and fails to capture the underlying data patterns, resulting in poor performance on both training and unseen data. Balancing between these extremes is crucial, particularly given the imbalanced nature of the datasets, where Z′ particle signals are scarce compared to background events. Proper feature selection and engineering, coupled with techniques like cross-validation, can alleviate these issues, ensuring that machine learning models are accurate and generalizable [32].

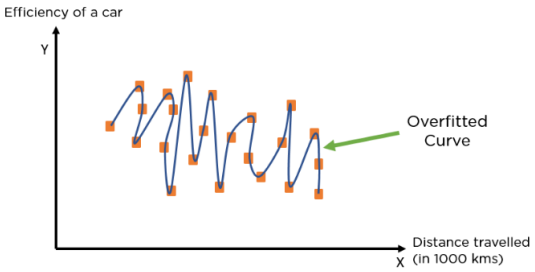


Figure 1.10.9: Example of overfitting in machine learning. This figure illustrates how the model captures noise and outliers in the training data, leading to poor generalization [33].

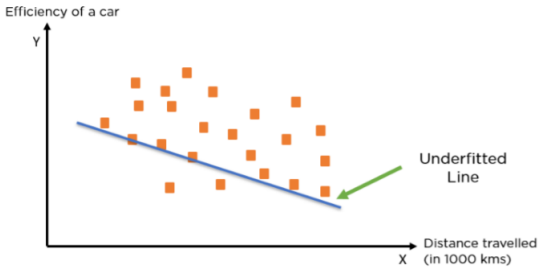


Figure 1.10.10: Example of underfitting in machine learning. This figure shows a model that is too simplistic to capture the complex patterns in the data, resulting in poor predictive performance [33].

1.10.10 Undersampling and oversampling

To combat data imbalance, undersampling and oversampling techniques can be employed. Undersampling involves reducing the dominant class (background events) to match the minority class (signal events), ensuring a balanced dataset for model training. Conversely, oversampling increases the minority class by duplicating signal events to match the count of background events. Both approaches aim to improve model training and generalization by addressing the imbalance in datasets.

1.10.11 Feature and event selection in high energy physics

In high-energy physics, the selection of features and events plays a vital role in improving model performance and accuracy [34]. This process entails identifying the most pertinent detector measurements and event attributes that significantly influence the detection of novel particles like the Z′ boson. Efficient selection aids in diminishing noise, enhancing model interpretability, and augmenting the likelihood of uncovering new physical phenomena. Within this framework, event selection assumes the guise of feature engineering, where both individual event charac-

teristics and overarching data patterns are refined to better serve our research objectives, such as discerning between signal and background events in collider data.

1.11 Testing and validation techniques

To ensure the reliability and accuracy of our machine learning models, a comprehensive evaluation and testing strategy will be implemented. The cornerstone of our evaluation process will involve the use of performance metrics such as accuracy, precision, recall, F1-Score, ROC Curve and significance (σ). These metrics will provide a multifaceted view of model performance, taking into account different aspects of classification quality and statistical significance.

1.11.1 Confusion matrix

The confusion matrix is a critical tool for evaluating the performance of binary classification models. It provides a detailed breakdown of the model's predictions, categorizing them into four fundamental groups: true positives (TP), true negatives (TN), false positives (FP), and false negatives (FN), as depicted in figure 1.11.11.

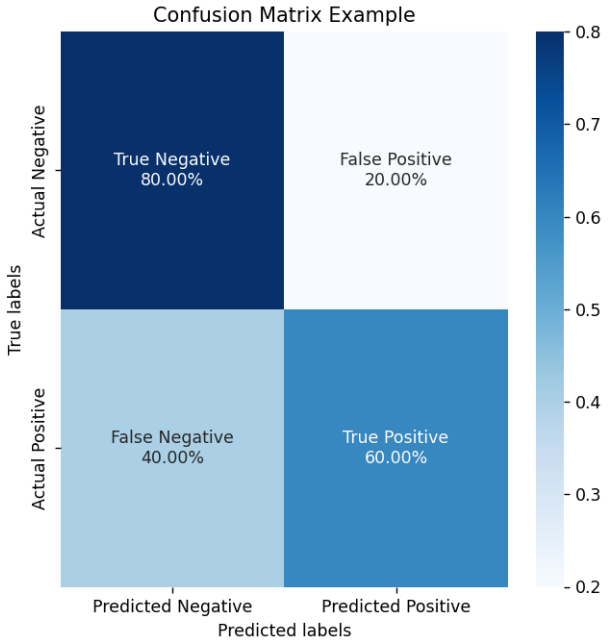


Figure 1.11.11: Example of a confusion matrix created in python, using matplotlib and seaborn.

True Positives (TP): These are instances where the model correctly predicts the positive class, meaning it correctly identifies the Z′ particle signals. **True Negatives (TN):** These are instances where the model correctly predicts the negative class, meaning it correctly identifies background events as non-signals. **False Positives (FP):** These occur when the model incorrectly predicts the positive class, meaning it incorrectly identifies background events as Z′ particle signals. This is also known as a Type I error. **False Negatives (FN):** These occur when the model incorrectly predicts the negative class, meaning it fails to identify actual Z′ particle signals. This is also known as a Type II error.

The confusion matrix allows researchers and analysts to visualize the model's performance and assess its predictive capabilities in distinguishing between the signal (Z′ particles) and the background noise (Standard Model processes). By examining the balance between TP, TN, FP, and FN, one can derive insights into the model's sensitivity (recall), specificity, and overall accuracy. This matrix serves as the foundation for calculating other performance metrics such as precision, recall, and the F1 score, thereby providing a comprehensive understanding of the model's effectiveness in identifying Z′ particle events amidst complex background noise [35].

By utilizing the confusion matrix alongside other performance metrics, a robust evaluation framework is ensured that emphasizes both the accuracy and reliability of the machine learning models in the detection of Z′ particles.

1.11.2 Precision and recall

Precision measures the proportion of true positive results in all positive predictions made by the model, reflecting its accuracy in identifying Z′ particle signals (Equation 2). Recall, or sensitivity, assesses the model's ability to detect all actual Z′ particle events within the dataset (Equation 3). Balancing these metrics is integral for minimizing false positives and negatives, ensuring that the model is both accurate and comprehensive in its classifications [36].

$$P = \frac{T_p}{T_p + F_p}$$

Equation 1: Equation for precision.

$$R = \frac{T_p}{T_p + F_n}$$

Equation 2: Equation for recall.

1.11.3 F1 score

The F1 score harmonizes the precision and recall metrics through their harmonic mean, offering a single measure to evaluate the model's accuracy. This is particularly useful when dealing with imbalanced datasets, where positive examples (Z′ particle signals) are rare compared to the background noise [37].

1.11.4 ROC Curve and AUC

The Receiver Operating Characteristic (ROC) curve, and the Area Under the Curve (AUC) provide insights into the model's performance across various threshold settings. By plotting the true positive rate against the false positive rate, the ROC curve illustrates the trade-offs between sensitivity and specificity, as illustrated in 1.11.12. A model with an AUC close to 1 indicates excellent discriminatory ability, whereas an AUC closer to 0.5 suggests no better than random guessing [38].

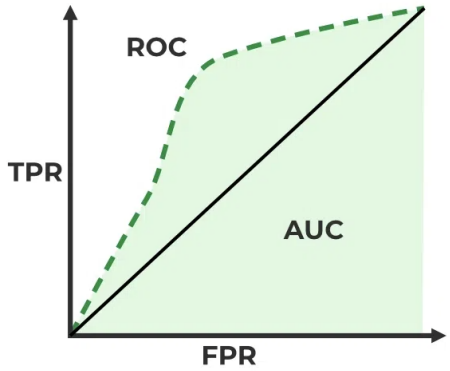


Figure 1.11.12: Illustration of a ROC-AUC Classification Evaluation Metric [39].

1.11.5 Significance in particle physics

In particle physics, the concept of significance is crucial, particularly in the search for new particles such as the hypothetical Z' boson. Significance is quantified in terms of the standard deviation, σ, from the null hypothesis, which represents the background-only model. This statistical measure helps scientists determine whether an observed signal is a real effect or merely a fluctuation of the background noise, as illustrated in Figures 1.11.13 and 1.11.14.
In particle physics, significance (σ), often denoted as 'Sigma,' is an essential metric used to evaluate the statistical significance of a signal compared to the background. A common initial estimate for significance is calculated as the ratio of the signal (s) to the square root of the background $(\sqrt{b})$. However, this is a simplified approximation, and thorough statistical treatments are required for precise calculations. Higher values of (σ) suggest stronger statistical evidence for the presence of a true signal.

$$\sigma = \frac{s}{\sqrt{b}}$$

Equation 3: Calculation of significance (σ) as the ratio of signal (s) to the square root of background $(\sqrt{b})$, where s represents the signal and b represents the background.

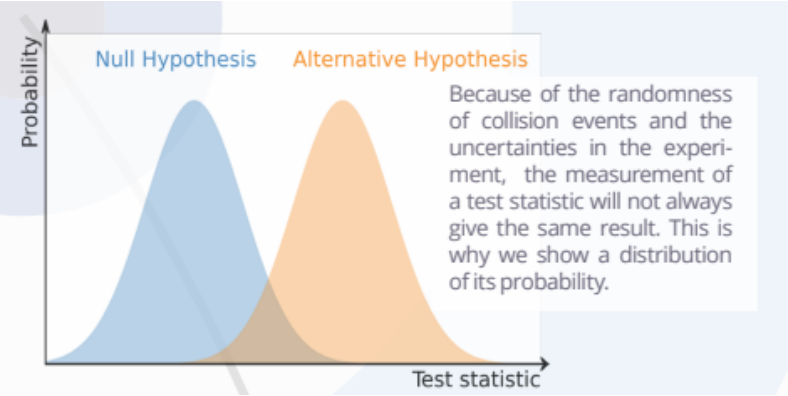


Figure 1.11.13: Null and alternative hypothesis distribution. Illustration taken from CERN [40].

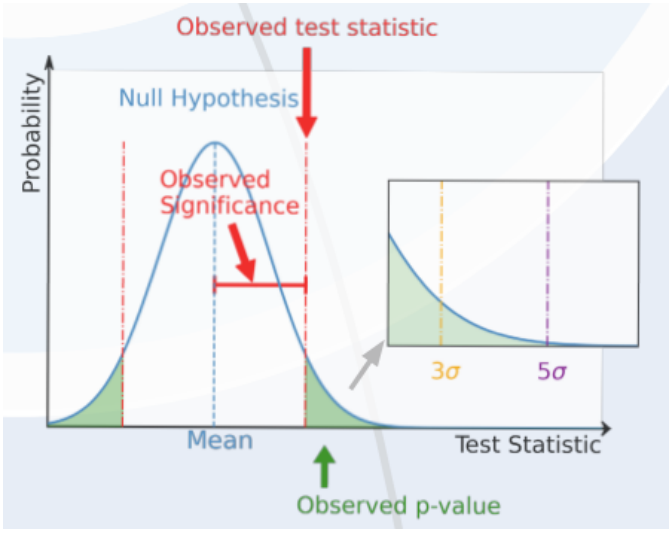


Figure 1.11.14: Observed test statistics and their significance. Illustration taken from CERN [40].

The significance levels are interpreted as follows:

- **1σ (One Sigma):** Corresponds to a 68.27% confidence level. At this level, there's about a 32% chance that the result is merely a statistical fluctuation. In particle physics, this is not considered significant enough to denote a discovery.
- **2σ (Two Sigma):** Represents a 95.45% confidence level. There is still around a 5% chance (1 in 20) that the observed effect is due to random chance. This level is more compelling but still not regarded as definitive evidence.
- **3σ (Three Sigma):** Corresponds to a 99.73% confidence level. Results at this level are often considered evidence of a new phenomenon but still fall short of conclusive proof.
- **4σ (Four Sigma):** This level represents a 99.9937% confidence level, indicating an extremely low probability that the observed effect is due to background fluctuations alone.
- **5σ (Five Sigma):** This is considered the gold standard in particle physics [41], corresponding to a 99.99994% confidence level. At this level, the chance that the result is a statistical fluke is about 1 in 3.5 million, providing compelling evidence of discovery.

The sigma levels are calculated based on the standard deviation of the background distribution and the observed excess of events. A 5σ significance is required to claim a discovery in particle physics to ensure that the probability of a false positive is exceedingly low. This rigorous standard helps maintain the integrity of findings in the field, where the datasets are large, and the stakes of false discoveries are high. The process of calculating significance involves comparing the observed data against the expected background and determining how likely it is to observe such data if there were no new particle. This involves statistical models and hypothesis testing, often employing techniques like maximum likelihood estimation and p-value calculations. The higher the significance, the less likely the result is due to chance, and the more confidence scientists have in the presence of new phenomena.

1.11.6 Evaluation metrics for regression models

For regression models aimed at estimating an unknown particle's mass, distinct performance metrics are essential. These metrics, pivotal for verifying the accuracy and reliability of predictions, include:

- **Mean Squared Error (MSE):** Calculates the average of the squares of the errors between actual and predicted values, with lower values indicating better fit.
- **Root Mean Squared Error (RMSE):** The square root of MSE, providing error magnitude in the same units as the predicted value.
- **Mean Absolute Error (MAE):** The average of the absolute differences between predicted and actual values, offering a straightforward measure of prediction accuracy without penalizing large errors heavily.
- **R-squared (R^2):** Reflects the proportion of the variance in the dependent variable that is predictable from the independent variables, with values closer to 1 indicating a better model fit.

1.12 Project plan

The project plan is illustrated as a Gantt chart, which outlines the project's schedule and tasks. The lengths of the bars represent the duration of each task, offering a clear visual on the timeline and allowing for the monitoring of milestones and potential delays. This approach ensures systematic progress tracking against predefined deadlines. See Appendix 7.A.1 for the detailed Gantt chart.

Risk assessment

The risk assessment provides a systematic evaluation of potential risks, prioritizing them based on the likelihood of occurrence and potential impact. Each identified risk is assigned a score, reflecting its probability and potential consequences. This process aids in formulating strategies for risk mitigation and management. The comprehensive risk assessment can be found in Appendix 7.B.2.

Evaluation Plan

The evaluation plan outlines a systematic approach for assessing the performance of machine learning models, aimed at achieving the project goals highlighted in Section 1.4. Acknowledging the challenges posed by the imbalanced dataset, characterized by a significantly lower frequency of Z′ particle signals compared to background events, techniques such as oversampling and undersampling will be utilized to balance the class distribution. This aims to mitigate bias towards the majority class and enhance the models' capability in detecting rare Z′ particle signals.

For the evaluation of classifier models, metrics such as accuracy, precision, recall, and the F1 score are crucial to provide a comprehensive assessment of performance. Additionally, the significance of findings, represented by sigma levels, will play an integral role in the evaluation process, involving calculations of significance to gauge the statistical strength of the results and assess the likelihood of discovering new physics phenomena within the simulated datasets.

Following the assessment of classifier model performance and significance, the focus will shift to mass estimation using regression techniques. The effectiveness of the regression models will be evaluated using metrics such as Mean Squared Error (MSE), Root Mean Squared Error (RMSE), Mean Absolute Error (MAE), and R-squared. These metrics will help validate the machine learning models' effectiveness in real-world particle physics scenarios, taking into account factors such as energy, momentum, and angles derived from detected particles.

2 Material and methods

This chapter details a combined approach of particle physics and machine learning for analyzing the Z′ particle, following objectives outlined in Section 1.4. It includes data exploration and preparation, framework utilization, algorithm selection, and evaluation metrics critical for accurate model assessment in high-energy physics. The project structures into three interconnected phases, designed to systematically enhance understanding and application of machine learning in detecting and analyzing the Z′ particle.

2.1 Information and data

The study leverages a comprehensive collection of simulated Monte Carlo (MC) datasets, meticulously prepared to represent a spectrum of particle interactions that could occur within the ATLAS detector at the Large Hadron Collider (LHC). These datasets are pivotal for analysis, enabling the training and validation of machine learning models to distinguish between the Standard Model (SM) processes and hypothetical scenarios involving the Z′ particle. Importantly, the use of simulated data allows for the establishment of a "ground truth," particularly critical for hypothesized particles like the Z′ particle, which has not yet been empirically observed. This approach provides a clear basis for evaluating the performance of our machine learning models against known outcomes. The Z′ particle datasets, tagged as "signal," are contrasted against various "background" datasets, encompassing standard particle interactions such as diboson productions and single top quark processes. This rich dataset repository, originally in ROOT format and converted to HDF5 for use, is hosted on Kaggle [42], ensuring accessible and reproducible analysis.

Dataset overview

These datasets fall into two main categories:

- **"Standard Model" processes datasets (Background)**: These include simulations of events such as diboson production (e.g., ZZ and WZ bosons) and single top quark production through different channels. Representing the "background" processes, these datasets are crucial in training models to accurately identify non-signal events amidst the noise of typical collider data.
- **Z′ particle datasets (Signal)**: These datasets range from ZPrime500 to ZPrime2500, simulating the production of the hypothetical Z′ boson with varying masses from 500 GeV/c^2 to 2500 GeV/c^2. Serving as the "signal" in analysis, these datasets enable the training and testing of models' ability to detect potential Z′ particle events against the standard model background.

Data imbalance

The main datasets include 7 signal files having 163308 samples and 20 background files containing 22442595 samples, at a ratio of 1:137.

Unseen data

In the later stages of phase 3, the last two signal files with mass parameters (1250GeV/c^2 and 2250GeV/c^2) having 46157 samples will be added for testing on unseen data.

Source attribution

All datasets derive from the CERN Open Data Portal [11], ensuring the use of high-quality, realistic simulation data for research.

This initial view of the data, as depicted in Figure 2.1.1, underscores the complexity and diversity of the features processed by machine learning models to differentiate between types of particle collision events.

```
data.head()
```

	alljet_n	channelNumber	eventNumber	jet_1_E	jet_1_MV1	jet_1_SV0	jet_1_eta	jet_1_jvf	jet_1_m
0	0	105986	25001	54897.203125	0.059601	0.000000	0.143288	0.882837	6843.874023
1	1	105986	25003	171373.312500	0.056311	0.000000	-2.006945	0.582158	8385.192383
2	0	105986	25006	171373.312500	0.056311	0.000000	-2.006945	0.582158	8385.192383
3	2	105986	25010	735410.562500	0.053790	0.000000	-2.270990	1.000000	21263.775391
4	2	105986	25011	67439.773438	0.994451	22.765493	-0.158554	0.967006	9051.469727

Figure 2.1.1: Example of the first few rows from one of the particle physics datasets.

The dataset contains columns with 169 features, with objects ordered based on their transverse momentum ('pt') from high to low. Attributes such as energy, mass, and other related metrics for a range of jets ('jet_1' to 'jet_9') and leptons ('lep_1' to 'lep_5') are crucial for the identification and classification of events within collider data [43]. **alljet_n** refers to the total number of jets detected in a single event. **channelNumber** is a unique identifier for the type of simulated process. **eventNumber** provides a unique identifier for each event within the dataset.

Attributes for each jet and lepton are detailed in tables 2.1.1 and 2.1.2

Feature	Description
jet_1_E to jet_9_E	Energy of jets 1 through 9
jet_1_MV1 to jet_9_MV1	Machine learning score MV1 for jets 1 through 9
jet_1_SV0 to jet_9_SV0	Machine learning score SV0 for jets 1 through 9
jet_1_eta to jet_9_eta	Pseudorapidity of jets 1 through 9
jet_1_jvf to jet_9_jvf	Jet vertex fraction for jets 1 through 9
jet_1_m to jet_9_m	Mass of jets 1 through 9
jet_1_phi to jet_9_phi	Azimuthal angle of jets 1 through 9
jet_1_pt to jet_9_pt	Transverse momentum of jets 1 through 9
jet_1_trueflav to jet_9_trueflav	Flavor of jets as determined by the simulation
jet_1_truthMatched to jet_9_truthMatched	Indicator if jets are matched to truth-level particles

Table 2.1.1: Attributes of Jets in the Dataset

Feature	Description
lep_1_E to lep_5_E	Energy of leptons 1 through 5
lep_1_charge to lep_5_charge	Electric charge of leptons 1 through 5
lep_1_eta to lep_5_eta	Pseudorapidity of leptons 1 through 5
lep_1_etcone20 to lep_5_etcone20	Isolation energy within a cone around the leptons
lep_1_flag to lep_5_flag	Quality flag for the leptons
lep_1_phi to lep_5_phi	Azimuthal angle of leptons 1 through 5
lep_1_pt to lep_5_pt	Transverse momentum of leptons 1 through 5
lep_1_ptcone30 to lep_5_ptcone30	Sum of the pt of tracks within a cone around the leptons
lep_1_trackd0pvunbiased to lep_5_trackd0pvunbiased	Transverse impact parameter relative to the primary vertex
lep_1_tracksigd0pvunbiased to lep_5_tracksigd0pvunbiased	Significance of the transverse impact parameter
lep_1_type to lep_5_type	Type identifier of leptons 1 through 5
lep_1_z0 to lep_5_z0	Distance from the primary vertex in the beam direction

Table 2.1.2: Attributes of Leptons in the Dataset

MC weight and scale factors in particle physics simulations:
In Monte Carlo (MC) simulations for particle physics, the application of MC weights (McWeight) and scale factors accurately replicates experimental conditions. McWeight adjusts the simulation's normalization to match real-world data, accounting for theoretical cross-sections, dataset luminosity, and observed data rates. Scale factors correct discrepancies between simulated and real detector data, essential for accurate emulation of actual collider experiments. They cover aspects such as trigger efficiency, lepton identification, b-tagging efficiency, and pile-up interactions [44, 45].

The implementation of scale factors is detailed in Figure 2.1.2, depicting the adjustment process for simulated datasets, ensuring that Monte Carlo simulations provide a realistic representation of experimental data.

scaleFactor_BTAG	scaleFactor_ELE	scaleFactor_JVFSF	scaleFactor_MUON	scaleFactor_PILEUP	scaleFactor_TRIGGER	scaleFactor_ZVERTEX
1.000000	1.000000	1.0	1.001046	2.186532	0.958796	0.095748
0.998036	0.967795	1.0	1.000000	2.186532	0.995144	0.948323
1.000000	1.000000	1.0	1.004650	2.186532	1.009498	1.142215
0.994364	0.975427	1.0	1.000000	2.186532	0.982606	0.971224
0.961822	1.000000	1.0	0.999771	2.186532	0.914313	1.206364

Figure 2.1.2: Illustration highlighting Scale Factors adjustments.

2.2 Work environment

Following the project descriptions and goals outlined in Section 1.4, Kaggle was selected as the central working environment for this project. Three Kaggle notebooks were created, one for each phase. (Phase 1: [46], Phase 2: [47], Phase 3: [48]).

Libraries utilized:
Key python libraries such as **scikit-learn**, **FastAi**, **matplotlib**, and **seaborn** played an important part in this project:

- **Python**:
  The primary programming language for data manipulation, model development, and analysis.
- **Scikit-learn**:
  Employed for implementing machine learning models and conducting data preprocessing tasks.
- **Fastai**:
  Used for neural network development, particularly for classification and regression tasks.
- **Matplotlib** and **Seaborn**:
  Utilized for data visualization, aiding in the interpretation and presentation of results.

These core libraries, along with other supporting tools, provided a robust and efficient environment for conducting experiments, processing data, and analyzing results throughout the project.

2.3 Phase one - Classifier development with random forest

The initial phase of the methodological approach concentrated on comprehending the dataset's structure, content, and underlying patterns. Furthermore, development and testing were carried out on three random forest models: a baseline model, an oversampled model, and an undersampled model. The random forest (Scikit-learn's "RandomForestClassifier") approach was first selected due to its effectiveness in handling tabular data. Performance metrics for all of them were consistent, including precision, recall, F1-score and ROC-AUC.

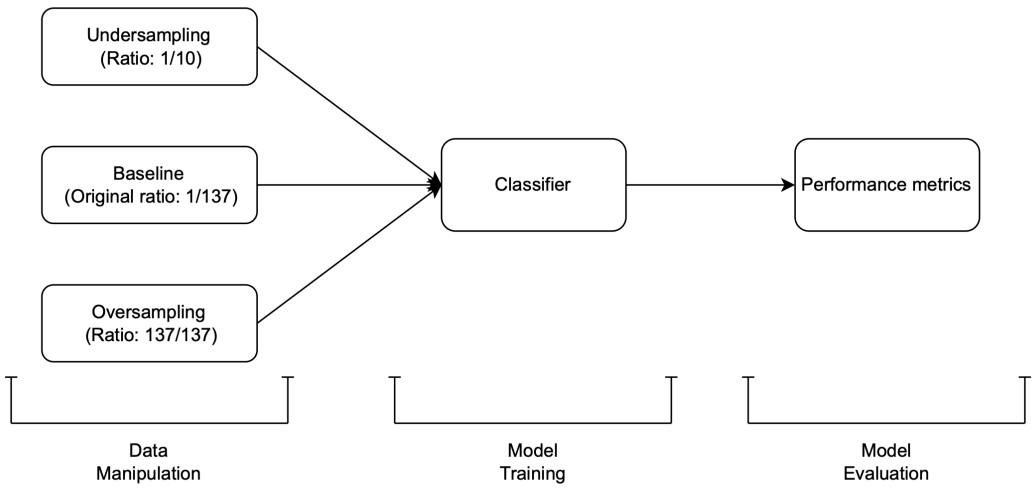


Figure 2.3.3: Schematic of the stages for training the classifier.

a) **Data exploration**

The initial analysis focused on understanding the dataset of simulated Monte Carlo events, examining physical properties from particle collisions. Using python's pandas library, descriptive statistics were computed to assess each feature's distribution, median, mean, and standard deviation. Histograms plotted via matplotlib using the code displayed in figure 2.3.4. further revealed distributions, pinpointing skewed data, outliers, and less variable features.

```
# Extract all features excluding the ones specified -> note: these are manully chosen initially beca
use they are irrelevant for training our model.
# However, we still have to do a more refined feature validation as you will see further down
plot_features = [col for col in data.columns if col not in [
    'eventWeight', 'mcWeight', 'channelNumber', 'runNumber', 'data_type', 'label', 'eventNumber']]

# Create separate figures for each group of features
for i in range(0, len(plot_features), 3):
    # Determine the number of subplots needed
    n_subplots = min(3, len(plot_features) - i)
    fig, axs = plt.subplots(1, n_subplots, figsize=(18, 4))
    # If there's only one subplot, axs will not be an array, so we wrap it in one
    if n_subplots == 1:
        axs = [axs]
    for j in range(n_subplots):
        feature_name = plot_features[i+j]
        axs[j].hist(data[feature_name], bins=50, color='blue', alpha=0.7)
        axs[j].set_xlabel('Value')
        axs[j].set_ylabel('Frequency')
        axs[j].set_title(feature_name)
    plt.show()
```

Figure 2.3.4: Code snippet used to plot histograms for each feature using matplotlib (plt).

Subsequently, heatmaps were employed to assess feature correlation. By generating correlation matrices and visualizing them through seaborn heatmaps as shown in 2.3.5, highly correlated features were identified, which helped in understanding the relationships between different variables. This step was crucial for feature selection, as redundant or highly correlated features could lead to multicollinearity, adversely affecting the machine learning model's performance.

```
# For jet features, select a random percentage
jet_features = [col for col in data.columns if 'jet' in col]

np.random.seed(42)  # For reproducibility
half_jet_features = np.random.choice(jet_features, size=len(jet_features) // 4, replace=False) # S
elect the percentage here //
jet_data_half = data[half_jet_features]
jet_correlation_half = jet_data_half.corr()

plt.figure(figsize=(18, 16))
sns.heatmap(jet_correlation_half, annot=True, fmt=".2f", cmap='coolwarm')
plt.title('Correlation Heatmap for Selected Jet Features')
plt.show()
```

Figure 2.3.5: Code snippet used to plot a heatmap for the jet features using seaborn and matplotlib.

Following the examination of jet features, a similar analysis was conducted for leptonic features to understand their interrelationships better. The process utilized for generating the lepton feature heatmap is illustrated through the code snippet shown in Figure 2.3.6. This code snippet underscores the approach taken to visualize the correlations among leptonic variables, which aids in distinguishing relevant features for the models.

```
# For lep features, select a random percentage (due to fitting issues)
lep_features = [col for col in data.columns if 'lep' in col]

np.random.seed(fixed_seed)  # For reproducibility
half_lep_features = np.random.choice(lep_features, size=len(lep_features) // 2,
replace=False) # Select the percentage here //
lep_data_half = data[half_lep_features]
lep_correlation_half = lep_data_half.corr()

plt.figure(figsize=(18, 16))
sns.heatmap(lep_correlation_half, annot=True, fmt=".2f", cmap='coolwarm')
plt.title('Correlation Heatmap for Selected Lep Features')
plt.show()
```

Figure 2.3.6: Code snippet used to plot a heatmap for the lepton features using seaborn and matplotlib.

i) Detailed feature analysis:

During initial data exploration, attention was directed towards certain jet features for detailed analysis due to their unique distributions. Figure 2.3.7 presents a code snippet illustrating the methodological approach to evaluating the relationships and characteristics of these specific features.

```
# List of specific features to plot
specific_features = ['jet_6_SV0', 'jet_7_SV0', 'jet_8_SV0', 'jet_9_SV0', 'jet_8_truefla
v', 'jet_9_trueflav']

# Selecting the data for these specific features
specific_data = data[specific_features]

# Computing the correlation matrix
specific_correlation = specific_data.corr()

# Plotting the heatmap
plt.figure(figsize=(10, 8))
sns.heatmap(specific_correlation, annot=True, fmt=".2f", cmap='coolwarm', square=True)
plt.title('Correlation Heatmap for Specific Jet Features')
plt.show()
```

Figure 2.3.7: Analysis of specific jet features to assess their characteristics and interrelations.

Another vital component of this phase was ensuring data quality. Inspection of the dataset for missing values, data type inconsistencies, and anomalous entries was conducted, as addressing these issues was essential for maintaining the integrity of the analysis and ensuring that machine learning models would be trained on accurate and complete data. The code snippet for printing all the rows, checking for missing values is depicted in figure 2.3.8

```
# Check for missing values
pd.set_option('display.max_rows', None) # Print all rows
data.isnull().sum()
```

Figure 2.3.8: Code snippet for checking missing values for all features.

Based on the insights gained from data exploration, the process proceeded with feature selection and engineering. This involved selecting relevant features that would likely contribute to distinguishing between signal and background events and engineering new features that could enhance the machine learning model's predictive power. These features were chosen through collaborative analysis with the ATLAS Research Group.

ii) Inspecting data imbalance:
Finally, before training the baseline model, it was necessary to evaluate the data imbalance between the signal and background files, as this would be the baseline model for comparison. This step served as the foundation for future research. The code snipped depicted in figure 2.3.9

```
# Function to get dataset length without loading data
def get_dataset_length(file_path):
    with h5py.File(file_path, 'r') as file:
        length = len(file['mini'])
    return length

# Sum up the total lengths
total_background = sum(get_dataset_length(path + f) for f in background_files)
total_signal = sum(get_dataset_length(path + f) for f in signal_files)
print(f"Total Background Data Length: {total_background}")
print(f"Total Signal Data Length: {total_signal}")
print(f"Signal to Background Ratio: 1:{total_background // total_signal}")
```

Figure 2.3.9: Python code that gets the length of the dataset without the need to load it into a dataframe.

b) Baseline model

A specific subset was chosen for initial analysis to efficiently manage the extensive volume of data. The approach is described in the following steps, implemented using Python's pandas library, to ensure efficient and reproducible model development.

i) **Data subset loading:** A function was devised to load a 10% subset of the total dataset, facilitating quicker initial model evaluations and adjustments. This subset approach aids in reducing computational demands while retaining a representative sample of the overall data.

ii) **Data segmentation:** The loaded data was segmented into two groups, 'df_background' and 'df_signal', based on their labels ('0' for background and '1' for signal samples), enabling binary classification to distinguish between signal and background files.

iii) **Data shuffling:** To ensure the model's generalization capability, the data was shuffled using a fixed seed (in this case, 42), guaranteeing that results are consistent and reproducible across different runs and comparisons.

```
# Load a subset of background and signal data - set the percentage here
subset_percentage = 0.1
df_background = load_subset_data(background_files, 0, subset_percentage)
df_signal = load_subset_data(signal_files, 1, subset_percentage)

# Combine background and signal data
df_combined = pd.concat([df_background, df_signal])

# Shuffle the combined DataFrame
df_combined = shuffle(df_combined, random_state=fixed_seed)
```

Figure 2.3.10: Procedure for loading and preparing data for the random forest baseline model.

Figure 2.3.10 highlights the steps taken to handle data preparation, ensuring the robustness of the modeling workflow. The process began with a the organization and randomization of the dataset to prepare for the random forest model's training and testing. Features not relevant to the model were marked for removal and tracked in a designated list to facilitate updates if more features became redundant. The dataset was then partitioned into training and testing subsets, following a 60/40 split via the train_test_split function from scikit-learn, to balance model training with validation on new data. The scikit-learn toolkit provided the necessary functions for fitting the model, conducting feature selection, and evaluating the model's efficacy against the test set.

An analysis of feature importance was conducted post-training to discern the physical parameters most critical for signal detection, informing further model optimization. Scikit-learn was used to construct key measures, including precision, recall, and the F1-score, for the initial assessment of the random forest models. These measures helped to assess the model's effectiveness in distinguishing between signal and background events in the unbalanced dataset.

Finally, ROC curves were generated to visually represent the trade-offs between the rates of true positives and false positives at different thresholds. The Area Under the ROC Curve (AUC) was employed to measure the overall capacity of the model in differentiating between signal and background events. A greater Area Under the Curve (AUC) value indicates superior performance of the model.

c) Undersampled model

The undersampled model was created by utilizing background data from preliminary analyses, with a focus on reducing the majority class in order to address the imbalance in the dataset. In this methodology, the signal-to-background ratio was precisely established at a ratio of 1:10 (as described in Figure 2.3.11). To maintain this ratio, the right number of background entries was chosen to match the signal data during the loading phase. This ensured a balanced distribution for training and improved the model's capacity to learn. The ratio was chosen to achieve a practical balance: it maintains the advantages of undersampling, such as enhanced model sensitivity and reduced training time, while presenting a more tolerable level of imbalance compared to the high ratios observed in natural datasets. This method seeks to increase the model's exposure to background events, hence enhancing its performance on highly imbalanced data and lowering the likelihood of overfitting to a training set that has been artificially balanced.

```python
# Load all signal data and add label
df_signal_list = []
for file in signal_files:
    with h5py.File(path + file, 'r') as f:
        data = pd.DataFrame(f['mini'][:])
        data['label'] = 1  # Signal label
        df_signal_list.append(data)

# Concatenate all signal data into a single DataFrame
df_signal = pd.concat(df_signal_list, ignore_index=True)

# Determine the number of background entries to match the signal data size
total_signal_entries = len(df_signal)
background_entries_per_file = total_signal_entries // len(background_files)

# Load a balanced subset of background data and add label
df_background_list = []
for file in background_files:
    with h5py.File(path + file, 'r') as f:
        data = pd.DataFrame(f['mini'][:]).sample(n=background_entries_per_file)
        data['label'] = 0  # Background label
        df_background_list.append(data)

# Concatenate all background data into a single DataFrame
df_background = pd.concat(df_background_list, ignore_index=True)

# Combine signal and background data
df_balanced = pd.concat([df_signal, df_background], ignore_index=True)

# Shuffle the combined DataFrame
df_balanced = shuffle(df_balanced, random_state=fixed_seed).reset_index(drop=True)
```

Figure 2.3.11: Process of loading signal data, determining necessary background entries, and creating a balanced dataset for the undersampled model.

To verify the balance between classes, implementation of a routine that counts and prints the number of background and signal entries loaded into the dataframe was conducted. This process is observable in the provided code snippets within Figure 2.3.12.

```
# Count the number of signal and background entries
signal_count = df_balanced[df_balanced['label'] == 1].shape[0]
background_count = df_balanced[df_balanced['label'] == 0].shape[0]

print(f"Number of signal entries: {signal_count}")
print(f"Number of background entries: {background_count}")
```

Figure 2.3.12: Code snippet used to verify the number of background entries vs. signal entries, ensuring a balanced dataset for the undersampled model.

Upon completing the training, application of the established performance metrics – precision, recall, and the F1-score, alongside the Receiver Operating Characteristic (ROC) curve and Area Under the ROC Curve (AUC) – identical to those used in evaluating the baseline model, was conducted.

d) Oversampled model

Oversampling is a technique that increases the number of instances in the minority class in order to achieve a balance with the majority class, unlike undersampling which reduces the majority class. The objective was to improve the model's ability to distinguish between signal and background events, while still preserving its sensitivity. A subset of data was created to mirror the proportions of the baseline model, ensuring a consistent evaluation across different models. This procedure entailed merging signal and background data into a unified dataset and randomizing it to avoid any possible bias in training caused by the sequence. The utils package in scikit-learn offers the resample function, which augments the signal samples to match the number of background occurrences, effectively rebalancing the dataset for training purposes. The augmented signal data were combined with the background data to create a new dataset that includes an equal number of samples from each classes, as illustrated in figure 2.3.13.

```
from sklearn.utils import resample

# Oversample signal data to match the number of background samples
df_signal_oversampled = resample(df_signal,
                                 replace=True,
                                 n_samples=len(df_background),
                                 random_state=fixed_seed)

# Combine the oversampled signal data with the background data
df_oversampled = pd.concat([df_background, df_signal_oversampled])

# Shuffle the combined DataFrame to ensure a good mix
df_oversampled = shuffle(df_oversampled, random_state=fixed_seed).reset_index(drop=True)
```

Figure 2.3.13: Application of oversampling to balance the dataset using scikit-learn's resample.

Once the oversampled dataset was created, it was shuffled to ensure randomized distribution, crucial for unbiased model training and evaluation. The dataset was then divided, using a 60-40 training-testing split, aligning with previous evaluation frameworks.

Training the oversampled model was conducted similarly to the baseline model using Scikit-learn's "RandomForestClassifier", with the distinction of utilizing a balanced dataset, anticipated to alter the model's learning dynamics. Post-training, the model's efficacy was assessed using precision, recall, F1-score, and ROC-AUC—identical metrics to the baseline model—to provide an insight into its signal detection capabilities enhanced by balanced data.

2.4 Phase two - Advancing with neural networks

Phase Two further developed machine learning techniques, specifically neural networks, to analyze more intricate data patterns, building upon previous work. As illustrated in figure 2.4.14, data preprocessing was performed using the FastAI library. This included the categorization of categorical variables, imputation of missing values, and normalization of all features.

```
# Assuming df_combined is your combined DataFrame with features and labels
X = df_combined[features]
y = df_combined['label']

# Convert the feature matrix (X) and the target vector (y) into a FastAI TabularPandas object
procs = [Categorify, FillMissing, Normalize]  # Preprocessing steps: categorify categorical variables, fill missing values, normalize

metrics = [accuracy, Precision(), Recall(), F1Score()]

# Split data into training and validation sets
splits = RandomSplitter(valid_pct=0.4)(range_of(df_combined))  # Split data into training and validation sets

# FastAI needs the data in a single DataFrame, so let's combine X and y again for convenience
df_combined['label'] = y

to = TabularPandas(df_combined, procs=procs, cat_names=None, cont_names=features, y_names="label", splits=splits, y_block=CategoryBl
ock)

# Create a dataloader
dls = to.dataloaders(bs=256)  # bs is the batch size
```

Figure 2.4.14: Data preparation code for neural network

a) Baseline model

As in phase one, the process began by developing a baseline classifier using the same dataset, but now with neural networks instead of random forest models. Data preparation consisted of selecting and retaining crucial attributes that were identified in phase one. A 10% subset of the data was used for the initial testing. Subsequently, the model setup was refined using the FastAI library, which offers tools like `lr_find()` to refine learning rates and batch sizes. The architecture of the model, with layers consisting of 200 and 100 nodes, was designed to enhance the model's ability to process and learn from the dataset's complexities effectively (Figure 2.4.15).

```
# Define the learner
learn = tabular_learner(dls, metrics=metrics, layers=[200,100], cbs=ShowGraphCallback())

# Find and plot the optimal learning rate
lr_min, lr_steep = learn.lr_find(suggest_funcs=(valley, steep))
print(f"Minimum/10: {lr_min:.1e}, Steepest point: {lr_steep:.1e}")

# Train the model with the new batch size and an adjusted learning rate
learn.fit_one_cycle(6, lr_max=lr_min)
```

Figure 2.4.15: Code snippet showcasing the neural network's configuration and training procedure. The model uses established evaluation metrics like accuracy, precision, recall, and F1-Score, adjusting epochs based on training and validation losses to mitigate overfitting.

b) Undersampled model

In this phase, the group further developed undersampling techniques with the goal of achieving a more accurate signal-to-background ratio of 1:10, in order to enhance the robustness of the model (Figure 2.4.16). The training set was organized and monitored following the same procedure previously described for phase one.

```
# Define the background to signal ratio
background_to_signal_ratio = 10  # 10 will give 10 times more background than signal data

# Load all signal data and add label
df_signal_list = []
for file in signal_files:
    with h5py.File(f"{path}/{file}", 'r') as f:
        data = pd.DataFrame(f['mini'][:])
        data['label'] = 1  # Assigning '1' to signal label
        df_signal_list.append(data)

# Concatenate all signal data into a single DataFrame
df_signal = pd.concat(df_signal_list, ignore_index=True)

# Determine the number of background entries to match the signal data size
total_signal_entries = len(df_signal)
background_entries_needed_per_file = (total_signal_entries // len(background_files) * background_to_signal_ratio)

# Load a balanced subset of background data and add label
df_background_list = []
for file in background_files:
    with h5py.File(f"{path}/{file}", 'r') as f:
        data = pd.DataFrame(f['mini'][:])
        # Ensure not to exceed available entries and only take a sample if necessary
        entries_to_sample = min(len(data), background_entries_needed_per_file)
        sampled_data = data.sample(n=entries_to_sample, random_state=fixed_seed)
        sampled_data['label'] = 0  # Assigning '0' to background label
        df_background_list.append(sampled_data)

# Concatenate all background data into a single DataFrame
df_background = pd.concat(df_background_list, ignore_index=True)

# Combine signal and background data
df_combined = pd.concat([df_signal, df_background], ignore_index=True)

# Shuffle the combined DataFrame
df_combined = shuffle(df_combined, random_state=fixed_seed).reset_index(drop=True)
```

Figure 2.4.16: Code demonstrating the data preparation process for the undersampled neural network model.

c) **Oversampled model**

Finally, a last neural network model was created using oversampling (1:1 ration) to address the imbalance in class distribution. A 10% subset of the data was utilized, following the same technique as prior models, to ensure consistent and comparable results. The subset was then merged into a single DataFrame and randomly rearranged to guarantee reliable and repeatable training results.

2.5 Phase three - Classifier evaluation and regression model development

In the final stage, the focus shifted from optimizing classifiers to utilizing them for performance evaluations across hypothesized mass points for the Z' particle, and to building a regression model for estimating the mass of the Z' particle.

a) **Classifying signal events for different Z' mass hypotheses**

The neural network - undersampled model, which had yielded the best results in the previous phase, was employed to evaluate each signal dataset with its specific mass hypothesis, ranging from 500 to 2500 GeV, against a uniform background dataset. This evaluation preserved the original data imbalance ratio of 1:137 (Figure 2.5.17). This approach ensured that the classifier's performance could be accurately assessed, even in the face of significant data imbalance, while testing individual signal datasets against all background datasets. The quantity of samples added was carefully monitored to maintain the integrity of the new imbalance.

```python
def prepare_data(signal_files, background_files, path, subset_size):
    # Initialize lists for holding the dataframes
    df_signal_list = []
    df_background_list = []

    # Load and sample signal data for each Z' particle mass scenario
    for s_file in signal_files:
        df_signal = load_data(s_file, path, 1, subset_size)  # Load a subset (global variable)
        df_signal_list.append(df_signal)  # Collect all signal dataframes

    # Since we aim for the background to be evenly distributed across tests,
    # each portion of the background data should be a signal subset (global variable) divided by 7
    background_subset_frac = subset_size / 7

    # Sample this fraction from each background dataset
    for b_file in background_files:
        df_background = load_data(b_file, path, 0, background_subset_frac)  # Use calculated fraction directly
        df_background_list.append(df_background)  # Collect adjusted background dataframes

    # Combine individual dataframes into two large ones for signal and background
    df_signal_combined = pd.concat(df_signal_list, ignore_index=True)
    df_background_combined = pd.concat(df_background_list, ignore_index=True)

    return df_signal_combined, df_background_combined
```

Figure 2.5.17: The Python function "prepare_data" loads and samples signal data for each Z' particle mass scenario, combining them with a proportionate subset of background data to prepare for the classification task.

After preparing the data, an additional performance metric, significance (σ), was introduced for mass hypothesis evaluation. Significance was calculated by determining the optimal cutoff for maximizing significance using event weights, as illustrated in Figure 2.5.18.

```
# Function to find the optimal cutoff for maximizing significance with weights
def find_optimal_cutoff(df_signal, df_background, feature='ML_output', weight='total_weight'):
    best_significance, best_cutoff = 0, 0
    for cutoff in np.linspace(df_signal[feature].min(), df_signal[feature].max(), num=100):
        S = df_signal[df_signal[feature] > cutoff][weight].sum()
        B = df_background[df_background[feature] > cutoff][weight].sum()
        significance = S / np.sqrt(B) if B > 0 else 0
        if significance > best_significance:
            best_significance = significance
            best_cutoff = cutoff
    return best_cutoff
```

Figure 2.5.18: Illustration of determining the optimal cutoff for maximizing significance.

This function is called within the "evaluate_model_significance" function, which evaluates the model's significance along with other metrics. The method for evaluating the model significance is shown in Figure 2.5.19.

```
# Evaluate model significance along with performance metrics
def evaluate_model_significance(df_signal_combined, df_background_combined, learn, path, features,
subset_size):
    significance_results = {}  # Storing significance results for each Z' mass hypothesis
    performance_results = {}  # Storing performance metrics results for each Z' mass hypothesis
    sigma_results = {}  # Dictionary to store sigma results for each Z' mass hypothesis

    # Loop through each file (each represents a different Z' mass hypothesis)
    for s_file in signal_files:
        mass = extract_mass_from_filename(s_file)

        # Prepare the dataset for this specific Z' mass hypothesis
        df_signal = load_data(s_file, path, 1, subset_size)
        df_combined = pd.concat([df_signal, df_background_combined], ignore_index=True).sample(fra
c=1, random_state=fixed_seed)

        # Get model predictions for the combined dataset
        dl = learn.dls.test_dl(df_combined[features], bs=1024)
        preds, _ = learn.get_preds(dl=dl)
        df_combined['ML_output'] = preds[:, 1].numpy()

        # Calculate optimal cutoff for maximizing significance
        optimal_cutoff = find_optimal_cutoff(df_combined[df_combined['label'] == 1], df_combined[d
f_combined['label'] == 0], 'ML_output', 'total_weight')

        # Apply the optimal cutoff to calculate significance using event weights
        weighted_S = df_combined[(df_combined['ML_output'] > optimal_cutoff) & (df_combined['labe
l'] == 1)]['total_weight'].sum()
        weighted_B = df_combined[(df_combined['ML_output'] > optimal_cutoff) & (df_combined['labe
l'] == 0)]['total_weight'].sum()
        weighted_significance = weighted_S / np.sqrt(weighted_B) if weighted_B > 0 else 0

        # Convert weighted significance to sigma level
        sigma = significance_to_sigma(weighted_significance)
```

Figure 2.5.19: Code snippet for evaluating model significance.

All other performance metrics and graphs where then plotted using the same methods as in the previous stages.

b) Regression model development for mass estimation

The primary objective of this stage was to construct a regression model using the FastAI package to accurately infer the mass of the Z' particle. This marked a transition in approaches, from classifying categories to estimating continuous variables, integrating a neural network specifically intended for regression tasks. In this phase, the dataset was subdivided in 40/60 for training and validation sets. The learner was modified by adding a final layer adjustment to calculate the anticipated mass and its performance was monitored using Mean Squared Error (MSE), Root Mean Squared Error (RMSE), Median Absolute Error (MAE) and R2-Score (Figures 2.5.20 and 2.5.21).

```python
from fastai.callback.tracker import EarlyStoppingCallback

# Set up data processing
splits = RandomSplitter(valid_pct=0.4, seed=fixed_seed)(range_of(df_signal_combined))

# Prepare the TabularPandas object
to_reg = TabularPandas(df_signal_combined, procs=[Categorify, FillMissing, Normalize],
                       cont_names=features_for_regression,
                       y_names='mass',
                       splits=splits,
                       y_block=RegressionBlock())

dls_reg = to_reg.dataloaders(bs=1024)

# Set up learner
learn_reg = tabular_learner(dls_reg, layers=[500, 200], n_out=1,
                            loss_func=F.mse_loss, metrics=[rmse, mse, mae, R2Score()])

# Find learning rate
learn_reg.lr_find()

# Train model
learn_reg.fit_one_cycle(50, lr_max=1e-2)
```

Figure 2.5.20: This python code prepares and trains the regression model

```python
# Evaluating the Regression Model
preds, targs = learn_reg.get_preds()

# Calculate Mean Squared Error
mse = mean_squared_error(targs, preds)

# Calculate Root Mean Squared Error
rmse = mean_squared_error(targs, preds, squared=False)

# Calculate Mean Absolute Error
mae = mean_absolute_error(targs, preds)

# Calculate R-squared
r2 = r2_score(targs, preds)

# Print the results
print(f'Mean Squared Error: {mse}')
print(f'Root Mean Squared Error: {rmse}')
print(f'Mean Absolute Error: {mae}')
print(f'R-squared: {r2}')
```

Figure 2.5.21: Code snippet for calculating and printing the performance metrics of the regression model.

The histograms and scatter plots were generated using visualization tools such as matplotlib and seaborn (Figure 2.5.22). These plots were utilized to compare the predicted and real masses, detect any biases, and evaluate the precision of the model. The visualizations play a vital role in comprehending the model's efficacy in estimating mass and will be thoroughly examined in the results section.

```
preds, targets = learn_reg.get_preds()
predicted_masses = preds.numpy().flatten()  # Convert predictions to a flat array
actual_masses = targets.numpy().flatten()   # Convert actual values to a flat array
# Histogram of predicted masses
plt.figure(figsize=(10, 6))
plt.hist(predicted_masses, bins=50, alpha=0.7, color='blue')
plt.title('Histogram of Predicted Masses')
plt.xlabel('Mass (GeV)')
plt.ylabel('Frequency')
plt.grid(True)
plt.show()
# Histogram of actual masses
plt.figure(figsize=(10, 6))
plt.hist(actual_masses, bins=50, alpha=0.7, color='red')  # Red for distinction
plt.title('Histogram of Actual Masses')
plt.xlabel('Mass (GeV)')
plt.ylabel('Frequency')
plt.grid(True)
plt.show()
# Histogram of Actual vs Predicted Masses
plt.figure(figsize=(10, 6))
plt.scatter(actual_masses, predicted_masses, alpha=0.5)
plt.title('Actual vs Predicted Masses')
plt.xlabel('Actual Mass (GeV)')
plt.ylabel('Predicted Mass (GeV)')
plt.plot([actual_masses.min(), actual_masses.max()], [actual_masses.min(), actual_masses.max()], 'k--')
plt.grid(True)
plt.show()
# Residuals vs Predicted values
residuals = targets - preds
plt.figure(figsize=(10, 6))
plt.scatter(preds, residuals, alpha=0.5)
plt.title('Residuals vs Predicted Values')
plt.xlabel('Predicted Values')
plt.ylabel('Residuals')
plt.axhline(y=0, color='r', linestyle='-')
plt.grid(True)
plt.show()
```

Figure 2.5.22: Python code snippets used for generating histograms and scatter plots to visualize the comparison between predicted and actual Z' particle masses.

The training method included re-scaling and standardizing numerical characteristics, along with parameter tuning, to enhance the predictive precision for Z' particle masses using collision event data. The "load_data_for_regression" function, as depicted in Figure 2.5.23, was developed to prepare data for a regression model that predicts particle masses. This function processes data from a specified file and path, incorporating particle mass from the filename if the "include_mass" parameter is set to true. It calculates the momentum components px, py, and pz for jets and leptons based on energy and pseudorapidity values. Additionally, it computes three features derived from the energy and momentum, crucial for the mass estimation of Z' particles. The processed dataset is then returned, ready for model training.

```
def load_data_for_regression(filename, path, include_mass=True):
    file = h5py.File(os.path.join(path, filename), 'r')
    data = pd.DataFrame(file['mini'][:]).apply(pd.to_numeric, errors='coerce', downcast='floa
t')

    if include_mass:
        mass = int(''.join(filter(str.isdigit, filename.split('.')[1])))
        data['mass'] = float(mass)

    # Add cosTheta calculation for jets and leptons
    for i in range(1, 7):
        jet_eta_col = f'jet_{i}_eta'
        if jet_eta_col in data.columns:
            data[f'jet_{i}_Theta'] = Theta_from_eta(data[jet_eta_col])
            data[f'jet_{i}_px'] = data[f'jet_{i}_pt'] * np.cos(data[f'jet_{i}_Theta'])
            data[f'jet_{i}_py'] = data[f'jet_{i}_pt'] * np.sin(data[f'jet_{i}_Theta'])
            data[f'jet_{i}_pz'] = np.where(data[f'jet_{i}_Theta'] == 0, 0, data[f'jet_{i}_pt
'] / np.tan(data[f'jet_{i}_Theta']))


    for i in range(1, 6):
        lep_eta_col = f'lep_{i}_eta'
        if lep_eta_col in data.columns:
            data[f'lep_{i}_Theta'] = Theta_from_eta(data[lep_eta_col])
            data[f'lep_{i}_px'] = data[f'lep_{i}_pt'] * np.cos(data[f'lep_{i}_Theta'])
            data[f'lep_{i}_py'] = data[f'lep_{i}_pt'] * np.sin(data[f'lep_{i}_Theta'])
            data[f'lep_{i}_pz'] = np.where(data[f'lep_{i}_Theta'] == 0, 0, data[f'lep_{i}_pt
'] / np.tan(data[f'lep_{i}_Theta']))
    # Calculate features for invariant mass estimates
    try:
        data['feature_1'] = (data[[f'jet_{i}_E' for i in range(1, 7)]].sum(axis=1))**2 - \
                            (data[[f'jet_{i}_px' for i in range(1, 7)]].sum(axis=1)**2 +
                             data[[f'jet_{i}_py' for i in range(1, 7)]].sum(axis=1)**2 +
                             data[[f'jet_{i}_pz' for i in range(1, 7)]].sum(axis=1)**2)

        data['feature_2'] = (data[[f'jet_{i}_E' for i in range(1, 5)] + ['lep_1_E']].sum(axis
=1))**2 - \
                            (data[[f'jet_{i}_px' for i in range(1, 5)] + ['lep_1_px']].sum(axis=1
)**2 +
                             data[[f'jet_{i}_py' for i in range(1, 5)] + ['lep_1_py']].sum(axis=1
)**2 +
                             data[[f'jet_{i}_pz' for i in range(1, 5)] + ['lep_1_pz']].sum(axis=1
)**2)

        data['feature_3'] = (data[['jet_1_E', 'jet_2_E', 'lep_1_E', 'lep_2_E']].sum(axis=1))*
*2 - \
                            (data[['jet_1_px', 'jet_2_px', 'lep_1_px', 'lep_2_px']].sum(axis=1)**
2 +
                             data[['jet_1_py', 'jet_2_py', 'lep_1_py', 'lep_2_py']].sum(axis=1)**
2 +
                             data[['jet_1_pz', 'jet_2_pz', 'lep_1_pz', 'lep_2_pz']].sum(axis=1)**
2)

    except KeyError as e:
        print(f"Skipping feature calculation: {str(e)}")

    return data
```

Figure 2.5.23: Code snippet illustrating the methodology used to prepare data for the regression model.

The formulas set to calculate the features directly within the "load_data_for_regression" function aimed to enhance the performance of the regression model on both training and unseen data, facilitating a more accurate estimation of the Z' particle's mass. This enhancement was achieved by calculating the invariant mass of the decay products and utilizing the derived variables.

The specific formulas used in these calculations are as follows:

$$m^2 = \left(\sum_{i=1}^{6} E_{\text{jet}_i}\right)^2 - \left(\left(\sum_{i=1}^{6} px_{\text{jet}_i}\right)^2 + \left(\sum_{i=1}^{6} py_{\text{jet}_i}\right)^2 + \left(\sum_{i=1}^{6} pz_{\text{jet}_i}\right)^2\right)$$

Equation 4: Total squared energy minus the vector sum of squared momentum components for six jets, capturing the invariant mass of the system.

$$m^2 = \left(\sum_{i=1}^{4} E_{\text{jet}_i} + E_{\text{lep}_1}\right)^2 - \left(\left(\sum_{i=1}^{4} px_{\text{jet}_i} + px_{\text{lep}_1}\right)^2 + \left(\sum_{i=1}^{4} py_{\text{jet}_i} + py_{\text{lep}_1}\right)^2 + \left(\sum_{i=1}^{4} pz_{\text{jet}_i} + pz_{\text{lep}_1}\right)^2\right)$$

Equation 5: Summation of squared energies and vector components of momentum for four jets plus one lepton, calculating the invariant mass to evaluate stability and interactions within the system.

$$m^2 = \left(\sum_{i=1}^{2} \text{jet}_i\, E + \text{lep}_i\, E\right)^2 - \left(\left(\sum_{i=1}^{2} p_{x,\text{jet}_i} + p_{x,\text{lep}_i}\right)^2 + \left(\sum_{i=1}^{2} p_{y,\text{jet}_i} + p_{y,\text{lep}_i}\right)^2 + \left(\sum_{i=1}^{2} p_{z,\text{jet}_i} + p_{z,\text{lep}_i}\right)^2\right)$$

Equation 6: Energy and momentum squared differences for two jets and two leptons.

$$\mathsf{X}_n\, \cos(\theta) = 2 * \arctan(e^{-\mathsf{X}_n\, eta})$$

Equation 7: Relationship between the cosine of the emission angle and the pseudorapidity for jets, which is crucial for correcting particle trajectory computations.

3 Results

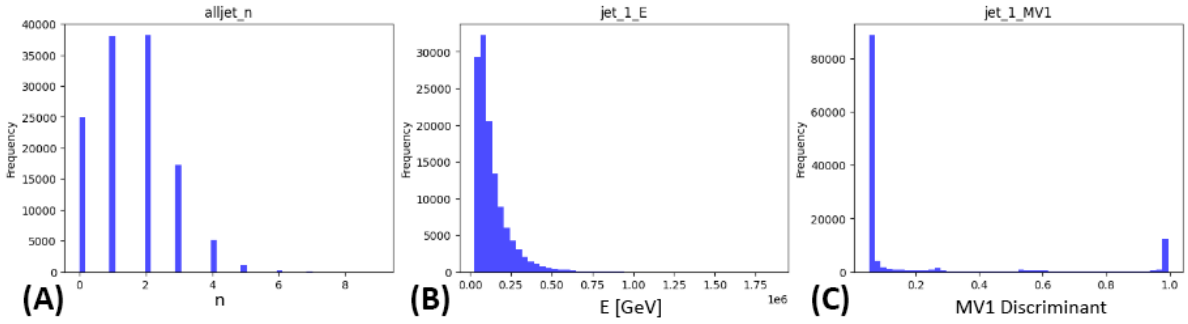
This chapter showcases the results obtained from data exploration as well as the implementation of various machine learning models and techniques as outlined in Section 2.

3.1 Phase One - Classifier development with random forest

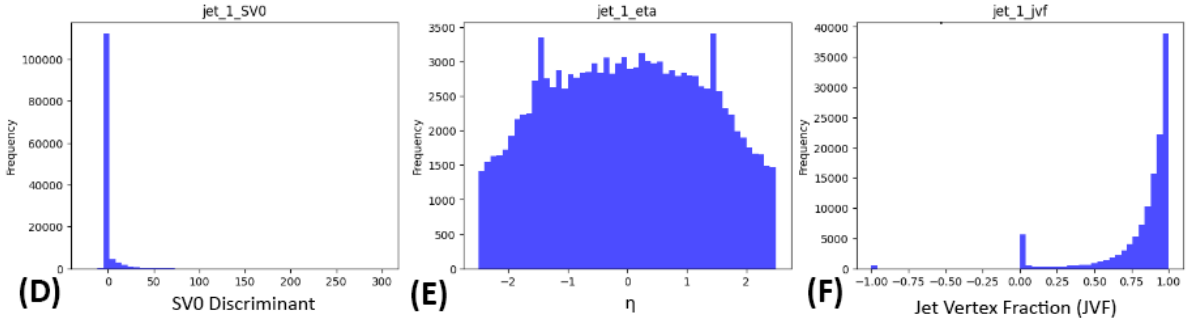
a) Data exploration

As outlined in Section 2, the research began with a full examination of the dataset, including data inspection, feature histogram generation, and correlation heatmap analysis, followed by the construction of three random forest classifiers. A total of 169 features, notably from jets and leptons, were investigated with key features emphasized. Detailed histograms of these features are included in the appendix 7.

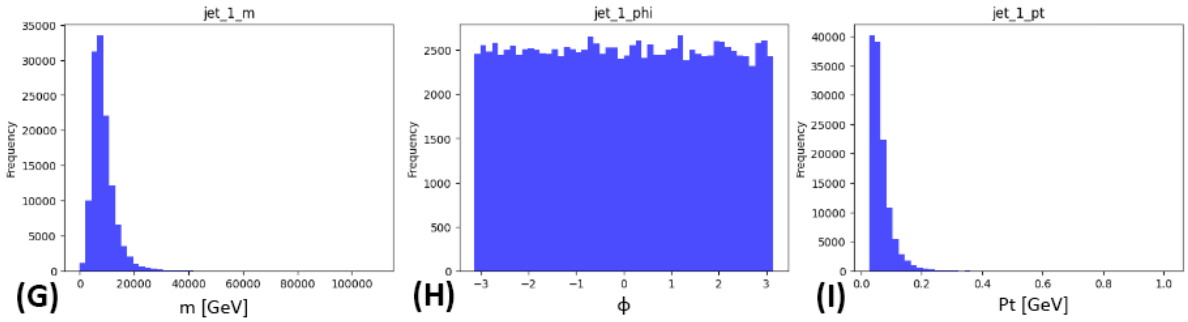
Jets: The histograms in Figure 3.1.1 (i-iii) show the frequency distribution of selected jet feature values in the dataset, with feature correlations detailed in Appendix 7. The x-axis of these graphs is fully extended to encompass all data points, ensuring full data integrity and analysis, even in less populated areas.



(i) Image (A) displays the "alljet_n" histogram, quantifying the jets per event, typically ranging from 0 to 3. Image (B) shows the energy (in gigaelectronvolt (GeV)) of the first jet, predominantly at the lower end of the spectrum. Image (C) illustrates the "jet_1_MV1" histogram's bimodal machine learning score distribution.



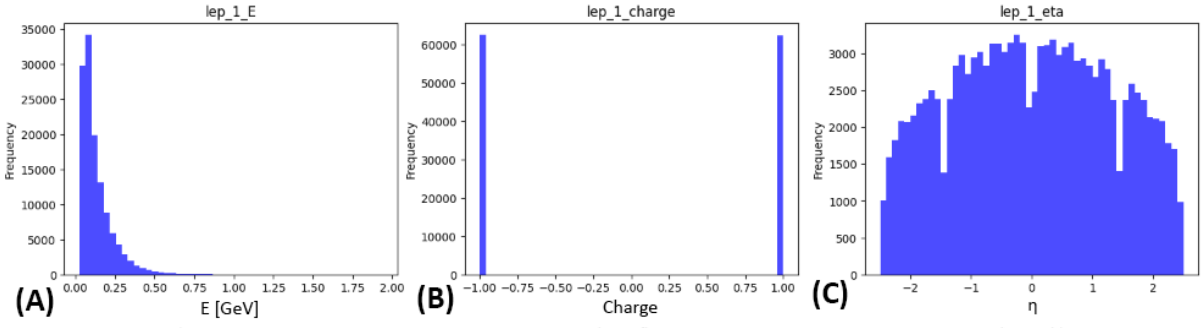
(ii) Image (D) shows the "jet_1_SV0" histogram, indicating b-quark identification scores mostly near zero, highlighting the efficacy of secondary vertices. Image (E) displays the "jet_1_eta" histogram, showing a symmetric pseudorapidity distribution around zero, suggesting uniformity in jet production angles. $\eta = 0$ indicates directions orthogonal to the beam axis. Image (F) illustrates the "jet_1_jvf" histogram, with a notable increase in frequency as the value approaches 1.0, suggesting many jets are closely associated with the primary vertex, indicating a strong correlation with the primary interaction point.



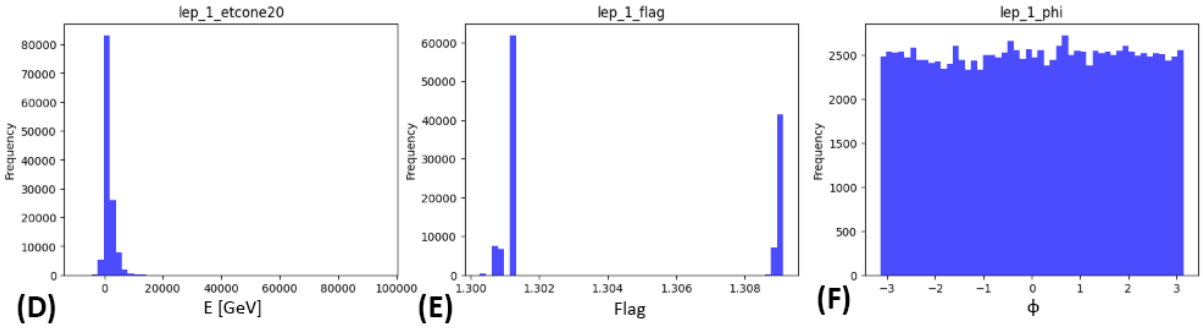
(iii) Image (G) presents the "jet_1_m" histogram, showing the first jet's mass with a right-skewed distribution, indicating predominantly lower masses. Image (H) displays the "jet_1_phi" histogram, illustrating a uniform distribution of the azimuthal angle, showing no preferential direction for jet emissions. Image (I) depicts the "jet_1_pt" histogram, highlighting a right-skewed transverse momentum distribution, with most jets having lower momentum.

Figure 3.1.1: Histogram for jet features.

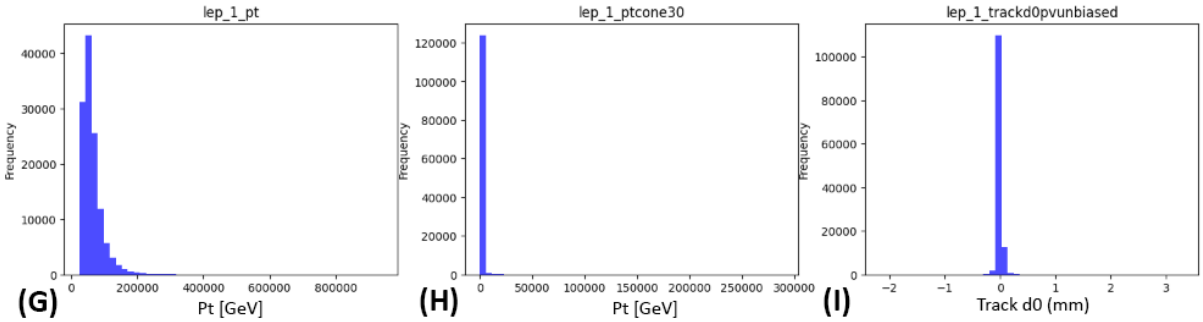
Leptons: Similarly, Figure 3.1.2 (i-iii) shows the frequency distribution of selected lepton features. Correlations within these features can be seen in Appendix 7.



(i) Image (A) illustrates the "lep_1_E" histogram, depicting a right-skewed energy distribution that predominantly features lower energy values (GeV) for the first lepton. Image (B) displays the "lep_1_charge" histogram, showing discrete bars that represent the distinct charge states of leptons. Image (C) shows the "lep_1_eta" histogram, symmetric around $\eta = 0$ with a central peak, indicating a predominance of events with small pseudorapidity. Notable dips near -1.5, 0, and 1.5 may reflect detector effects or specific experimental conditions.



(ii) Image (D) shows the "lep_1_etcone20" histogram, highlighting a sharp peak near zero and a concentration at the lower end, suggesting this measures the isolation energy (in GeV) around the lepton. Image (E) displays the "lep_1_flag" histogram, indicating specific lepton states. Image (F) illustrates the "lep_1_phi" histogram with a uniform distribution, implying that the detection of the first lepton's azimuthal direction is consistent across all angles.



(iii) Image (G) presents the right-skewed "lep_1_pt" histogram, showing that the transverse momentum of the first lepton is typically lower. Image (H) displays the "lep_1_ptcone30" histogram with a sharp peak near zero, suggesting it measures isolation. Image (I) features the "lep_1_trackd0pvunbiased" histogram, with a prominent peak at zero, indicating the lepton's trajectory closely passes the primary vertex.

Figure 3.1.2: Histogram for lepton features.

Missing values

No missing values were found in any of the features. Table 3.1.1 presents the occurrence of missing values in selected features within the dataset.

Feature	Missing Values
jet_1_eta	0
jet_1_pt	0
jet_2_eta	0
jet_2_pt	0
lep_1_eta	0
...	...

Table 3.1.1: Features showing no missing values or datatype inconsistencies.

Inspecting data imbalance

The data imbalance between the signal and background datasets, detailed in Table 3.1.2, significantly challenges model training. Such imbalance typically biases classifiers toward the majority class, here the background, leading to high accuracy but poor signal detection. This skew undermines the model's utility in identifying rare events crucial for experiments like the ATLAS. Addressing this imbalance is essential for ensuring the model's effectiveness and generalization.

Data Type	Total Length
Total Background Data Length	22442556
Total Signal Data Length	163308
Ratio	**1:137**

Table 3.1.2: Summary of Data Imbalance

Feature selection

During the initial phases of model preparation, feature importance was evaluated to identify the most significant variables for model training, with the top 55% of these variables illustrated in Figure 3.1.3 and a complete list available on Kaggle phase one [46]. A total of 169 features were considered, out of which 104 were selected based on the guidance of the ATLAS Research Group. This group advised excluding all weight and scale factor features like 'eventWeight', 'mcWeight', etc., to avoid inflating the model's performance artificially. Features such as 'channelNumber' and additional jets and leptons (jet 3-9, lep 3-5, and their variants) were also removed (table 3.1.3) due to their potential to introduce bias or because they provided limited value, focusing the model training on essential physical properties of the data.

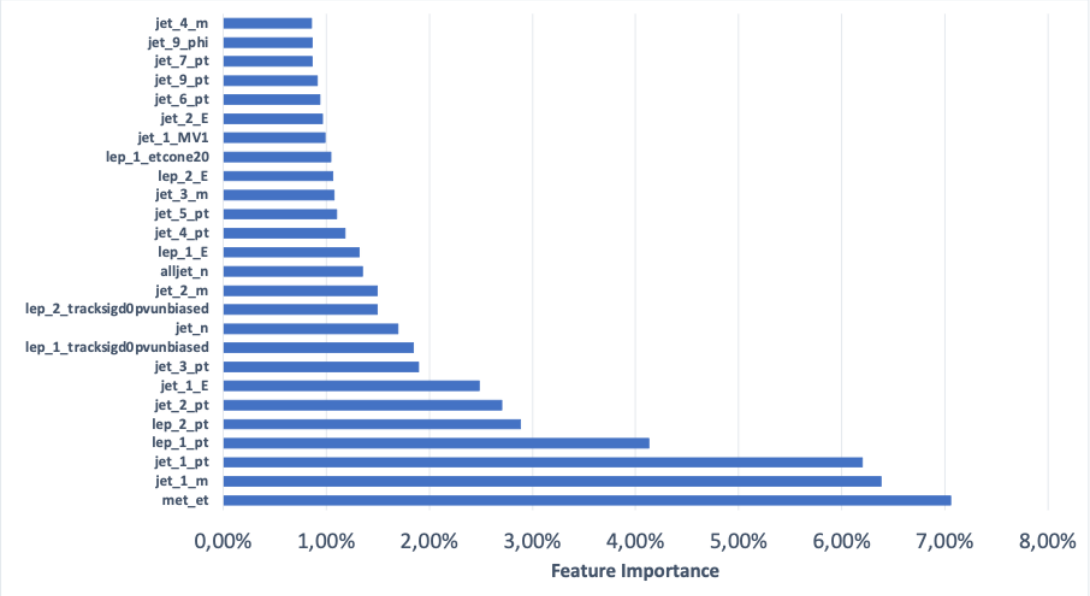


Figure 3.1.3: Bar graph of the top 26 (approx. 55%) most important features for the creation of the classifier model.

Excluded features list

Following iterative refinement during the initial phase, Table 3.1.3 displays the final list of excluded features.

Feature Name		
eventWeight *	mcWeight *	channelNumber *
runNumber *	data_type *	label *
eventNumber *	jet_6_SV0	jet_7_SV0
jet_8_SV0	jet_9_SV0	jet_8_trueflav *
jet_9_trueflav *	lep_4_E	lep_4_charge
lep_4_eta	lep_4_etcone20	lep_4_flag
lep_4_phi	lep_4_pt	lep_4_ptcone30
lep_4_trackd0pvunbiased	lep_4_tracksigd0pvunbiased	lep_4_type
lep_4_z0	lep_5_E	lep_5_charge
lep_5_eta	lep_5_etcone20	lep_5_flag
lep_5_phi	lep_5_pt	lep_5_ptcone30
lep_5_trackd0pvunbiased	lep_5_tracksigd0pvunbiased	lep_5_type
lep_5_z0	lep_trigMatched *	jet_1_trueflav *
jet_1_truthMatched *	jet_2_trueflav *	jet_2_truthMatched *
jet_3_trueflav *	jet_3_truthMatched *	jet_4_trueflav *
jet_4_truthMatched *	jet_5_trueflav *	jet_5_truthMatched *
jet_6_trueflav *	jet_6_truthMatched *	jet_7_trueflav *
jet_7_truthMatched *	jet_8_trueflav *	jet_8_truthMatched *
jet_9_trueflav *	jet_9_truthMatched *	scaleFactor_BTAG *
scaleFactor_ELE *	scaleFactor_JVFSF *	scaleFactor_MUON *
scaleFactor_PILEUP *	scaleFactor_TRIGGER *	scaleFactor_ZVERTEX *

Table 3.1.3: Excluded Features from the Analysis for Model Training. Features marked with an '*' are classified as metadata.

b) Classifier development with random forest

i) Baseline

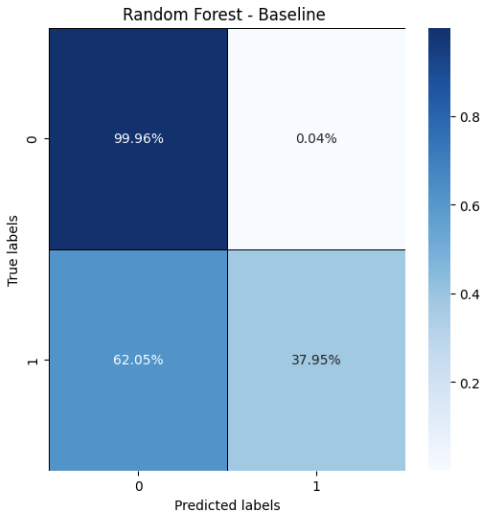


Figure 3.1.4: Confusion matrix created for the baseline random forest classifier showing number of: true positives (37.95%), false positives (0.04%), false negatives (62.05%) and true negatives (99.96%).

Performance metrics	
Metrics	**Performance**
Accuracy	0.995
Precision	0.881
Recall	0.380
F1-Score	0.531
ROC	0.99

Table 3.1.4: Performance metrics for a random forest classifier trained with an imbalance of 1:137.

ii) Undersampling

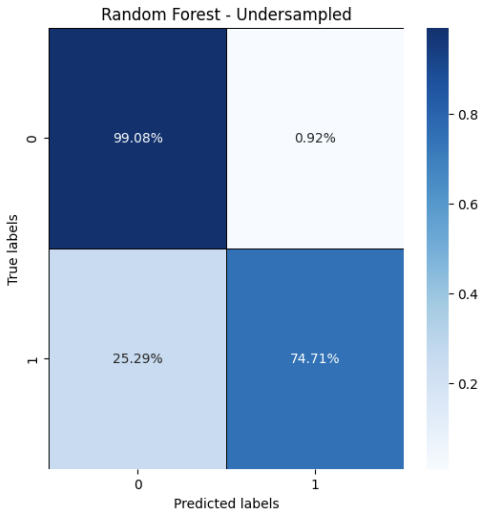


Figure 3.1.5: Confusion matrix created for the undersampled random forest classifier showing number of: true positives (74.71%), false positives (0.92%), false negatives (25.29%) and true negatives (99.08%).

Performance metrics	
Metrics	**Performance**
Accuracy	0.966
Precision	0.902
Recall	0.747
F1-Score	0.817
ROC	0.99

Table 3.1.5: Performance metrics for a random forest classifier trained with an imbalance of 1:10.

iii) Oversampling

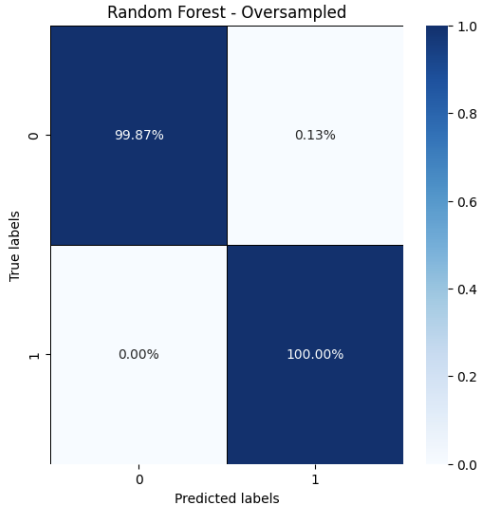


Figure 3.1.6: Confusion matrix created for the oversampled random forest classifier showing number of: true positives (100.00%), false positives (0.13%), false negatives (0.00%) and true negatives (99.87).

Performance metrics	
Metrics	**Performance**
Accuracy	0.999
Precision	0.999
Recall	1.000
F1-Score	0.999
ROC	1

Table 3.1.6: Performance metrics for a random forest classifier trained with an imbalance of 137:137.

c) Random forest models comparison

To facilitate the comparison of the previously presented results, the metric scores for each model are collectively outlined in Table 3.1.7. The ROC curve images for all models can be seen in Appendix 7, Figures 7.F.15i–7.F.15iii

Model	Validation Accuracy	Precision	Recall	F1-Score	ROC AUC
Baseline	0.9952	0.8812	0.3795	0.5306	0.99
Undersampled	0.9662	0.9019	0.7471	0.8173	0.99
Oversampled	0.9994	0.9988	1.0000	0.9994	1

Table 3.1.7: Summary of performance metrics for random forest models

The artificially high accuracy and ROC AUC score observed in the Baseline model (1:137 signal to background ratio) is primarily due the significant data imbalances. The Baseline model's predominance of background events leads it to excel at predicting the majority class but fail to effectively identify the minority class, i.e., the signal, as indicated by the low recall value of 37.95%.

In contrast, the Oversampled model, with its balanced dataset (137:137 ratio), shows perfect recall but is indicative of overfitting—where the model learns to recognize repeated instances of signals rather than generalizing from actual patterns. The Undersampled model, with a 1:10 signal to background ratio, demonstrates a more balanced approach, achieving lower overall accuracy but higher recall and F1-score, indicating a better capability to identify signals without sacrificing precision.

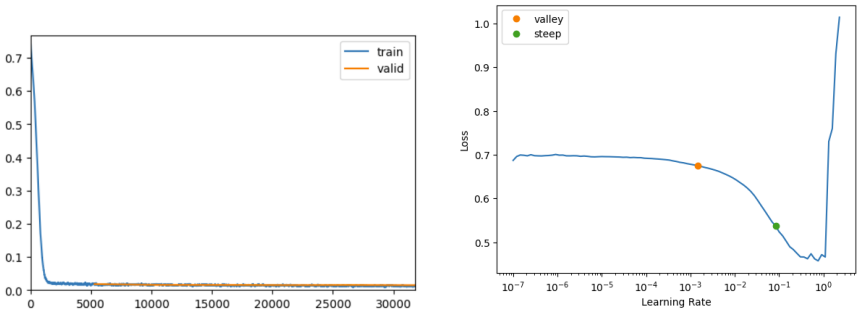
3.2 Phase two - Advancing with neural networks

a) Neural network models development

i) Baseline

Epoch	Train Loss	Valid Loss	Accuracy	Precision	Recall	F1 Score	Time
0	0.018714	0.016228	0.995278	0.758498	0.495112	0.599136	02:00
1	0.013242	0.016609	0.995110	0.902187	0.352056	0.506473	01:48
2	0.013141	0.015764	0.995534	0.859361	0.446548	0.587707	02:00
3	0.014343	0.014362	0.995622	0.847401	0.470442	0.605008	02:06
4	0.013194	0.014599	0.995583	0.834197	0.474631	0.605024	01:52
5	0.013337	0.014607	0.995607	0.825823	0.486268	0.612109	02:01

Table 3.2.8: Epoch training results for the neural network model



(i) Training and validation loss plot for the baseline neural network. Both losses decrease and stabilize.

(ii) The loss minimizes near a learning rate of 10^{-3}, suggesting a starting point for finding the optimal learning.

Figure 3.2.7: Neural network analysis plots.

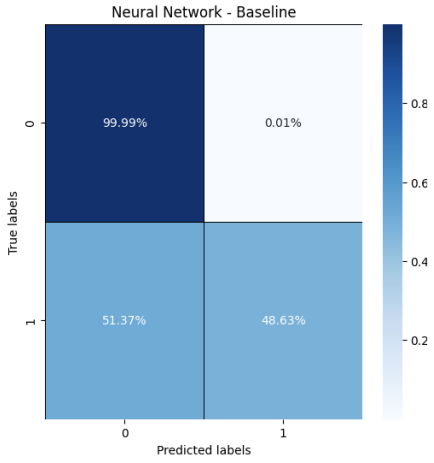


Figure 3.2.8: Confusion matrix created for the baseline neural network classifier showing number of: true positives (48.63%), false positives (0.01%), false negatives (51.37%) and true negatives (99.99%).

Performance metrics	
Metrics	**Performance**
Accuracy	0.996
Precision	0.826
Recall	0.486
F1-Score	0.612
ROC	0.99

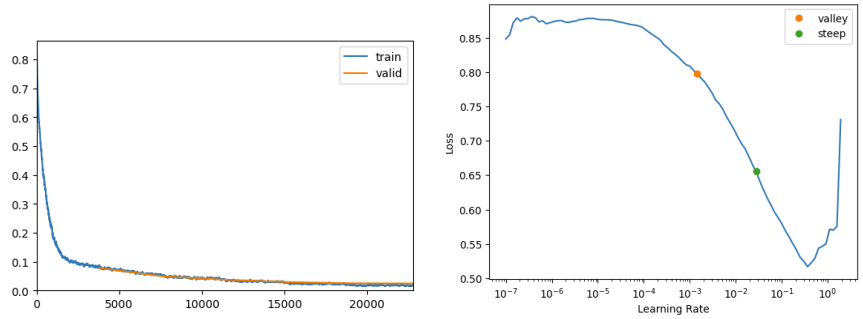
Table 3.2.9: Performance metrics for a neural network classifier trained with an imbalance of 1:137.

ii) Undersampling

The undersampled neural network model training utilized balanced datasets, aligning with methods described in Figure 2.4.16 in Section 2. The dataset was composed of approximately 10x as much background entries vs signal entries (163.308 vs 1.455.606).

Epoch	Train Loss	Valid Loss	Accuracy	Precision	Recall	F1 Score	Time
0	0.084790	0.077733	0.968470	0.882071	0.792006	0.834616	01:20
1	0.053226	0.049308	0.981103	0.914512	0.895603	0.904959	01:31
2	0.038326	0.037293	0.986108	0.930073	0.931760	0.930916	01:35
3	0.026410	0.029325	0.989462	0.945017	0.950392	0.947697	01:18
4	0.020494	0.025159	0.991042	0.953730	0.957264	0.955494	01:23
5	0.017916	0.024423	0.991386	0.958917	0.955173	0.957041	01:32

Table 3.2.10: Epoch Training Results for the undersampled neural network model



(i) Training and validation loss plot for the undersampled neural network. Both losses decrease and stabilize.

(ii) The loss minimizes near a learning rate of 10^{-3}, suggesting a starting point for finding the optimal learning rate for the model

Figure 3.2.9: Undersampled neural network analysis plots.

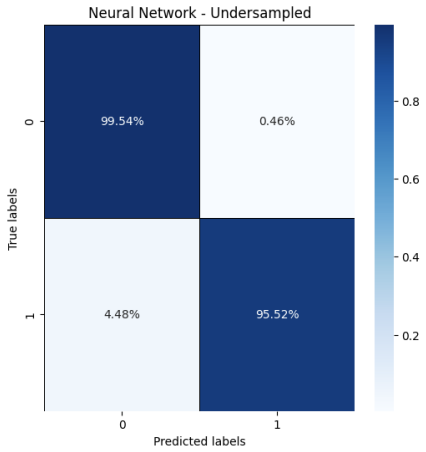


Figure 3.2.10: Confusion matrix created for the undersampled neural network classifier showing number of: true positives (95.52%), false positives (0.46%), false negatives (4.48%) and true negatives (99.54%).

Performance metrics	
Metrics	**Performance**
Accuracy	0.991
Precision	0.959
Recall	0.955
F1-Score	0.957
ROC	1

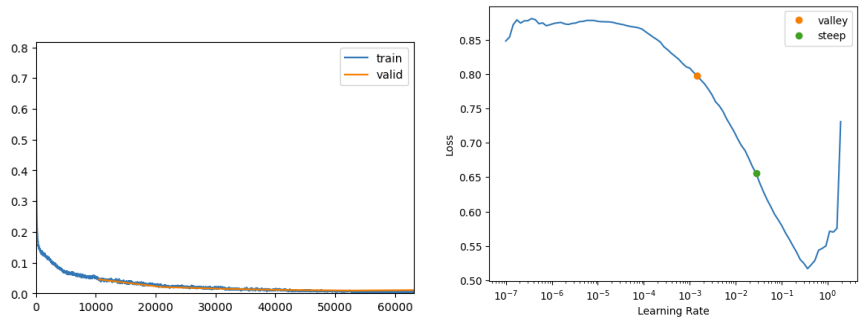
Table 3.2.11: Performance metrics for a neural network classifier trained with an imbalance of 1:10.

iii) Oversampling

The oversampled neural network model training utilized balanced datasets, aligning with methods described in Section 2.

Epoch	Train Loss	Valid Loss	Accuracy	Precision	Recall	F1 Score	Time
0	0.084790	0.077733	0.968470	0.882071	0.792006	0.834616	01:20
1	0.053226	0.049308	0.981103	0.914512	0.895603	0.904959	01:31
2	0.038326	0.037293	0.986108	0.930073	0.931760	0.930916	01:35
3	0.026410	0.029325	0.989462	0.945017	0.950392	0.947697	01:18
4	0.020494	0.025159	0.991042	0.953730	0.957264	0.955494	01:23
5	0.017916	0.024423	0.991386	0.958917	0.955173	0.957041	01:32

Table 3.2.12: Epoch training results for the oversampled neural network model



(i) Training and validation loss plot for the oversampled neural network. Both losses decrease and stabilize.

(ii) The loss minimizes near a learning rate of 10^{-3}, suggesting a starting point for finding the optimal learning rate for the model.

Figure 3.2.11: Oversampled neural network analysis plots.

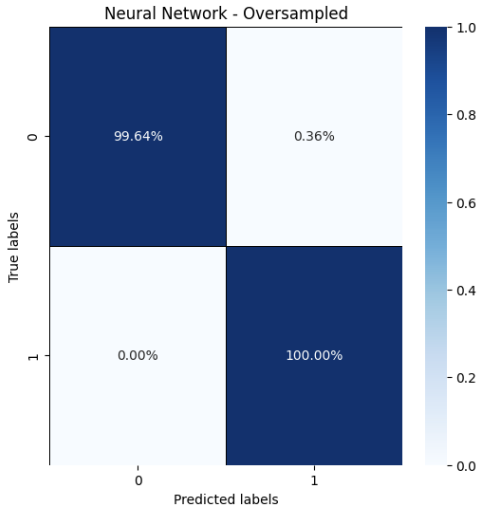


Figure 3.2.12: Confusion matrix created for the oversampled neural network classifier showing number of: true positives (100.00%), false positives (0.36%), false negatives (0.00%) and true negatives (99.64%).

Performance metrics	
Metrics	**Performance**
Accuracy	0.998
Precision	0.996
Recall	1.000
F1-Score	0.998
ROC	1

Table 3.2.13: Performance metrics for a neural network classifier trained with an imbalance of 137:137.

b) Neural network models comparison

To facilitate the comparison of the previously presented results, they are collectively outlined in Table 3.2.14. The ROC curve images for all models can be seen in Appendix 7, Figures 7.F.16i–7.F.16iii

Model	Validation Accuracy	Precision	Recall	F1-Score	ROC AUC
Baseline	0.9956	0.8565	0.4661	0.6037	0.99
Undersampled	0.9922	0.9579	0.9648	0.9613	1
Oversampled	0.9982	0.9964	1.0000	0.9982	1

Table 3.2.14: Summary of performance metrics for neural network models

Similar to the random forest models, the artificially high accuracy observed in the neural network models can be attributed to the imbalance between signal and background data. This effect is less pronounced in the undersampled neural network, which, despite its high accuracy, demonstrates a more balanced performance with superior recall and F1-score. This model, with a signal to background ratio of 1:10, exhibits the best overall performance metrics, reflecting its effectiveness in balancing precision and recall, thus better generalizing across different data scenarios.

3.3 Phase three - Classifier evaluation and regression model development

As outlined in Section 2, the most effective neural network, which incorporated undersampling techniques, was selected to classify signal events across various Z′ mass hypotheses. Subsequently, a regression model was developed to estimate the mass of the Z′ particles.

a) Classifying signal events for different Z′ mass hypotheses

The classifier's performance was assessed using an initial dataset imbalance ratio of 1:137, as described in Section 2.1. The quantities of the dataset utilized for this method are listed in Table 3.3.15. The total number of signal samples reported, 163308, is the combined count from all seven signal files. Each of these files, including around 23330 samples, was assessed against the complete background dataset, comprising 3206085 samples, in order to maintain the original imbalance ratio of the data.

Data Type	Quantity
Total Signal Files	7
Total Background Files	20
Total Signal Samples	163308
Total Background Samples	3206085

Table 3.3.15: Dataset quantities for classifier evaluation across mass ranges

Confusion matrices and performance metrics

Figures 3.3.13 to 3.3.19 show the confusion matrices across the mass range, and Figures 3.3.16 to 3.3.22 show the performance metrics.

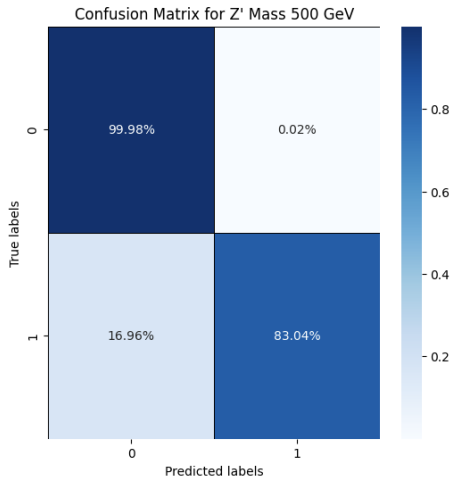


Figure 3.3.13: The confusion matrix for 500 GeV signals classified shows true positives (83.04%), false positives (0.02%), false negatives (16.96%), and true negatives (99.98%).

Performance metrics	
Metrics	**Performance**
Accuracy	0.998
Precision	0.965
Recall	0.830
F1-Score	0.893
Significance	1.775
ROC Score	0.999
Optimal Cutoff*	0.9900

Table 3.3.16: Performance metrics for 500 GeV signals classified by the best neural network classifier. *Optimal Cutoff value used to calculate the significance score.

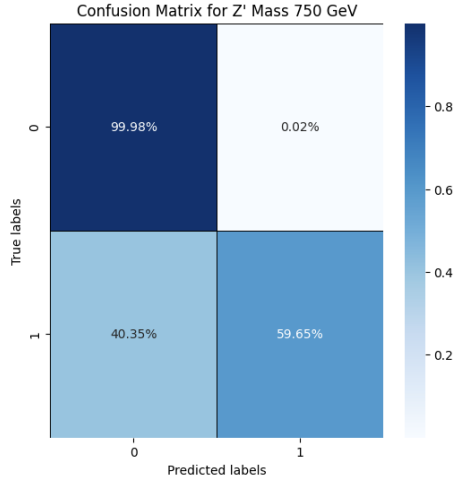


Figure 3.3.14: The confusion matrix for 750 GeV signals classified shows true positives (59.65%), false positives (0.02%), false negatives (40.35%), and true negatives (99.98%).

Performance metrics	
Metrics	**Performance**
Accuracy	0.997
Precision	0.955
Recall	0.596
F1-Score	0.734
Significance	1.360
ROC Score	0.999
Optimal Cutoff*	0.9899

Table 3.3.17: Performance metrics for 750 GeV signals classified by the best neural network classifier. *Optimal Cutoff value used to calculate the significance score.

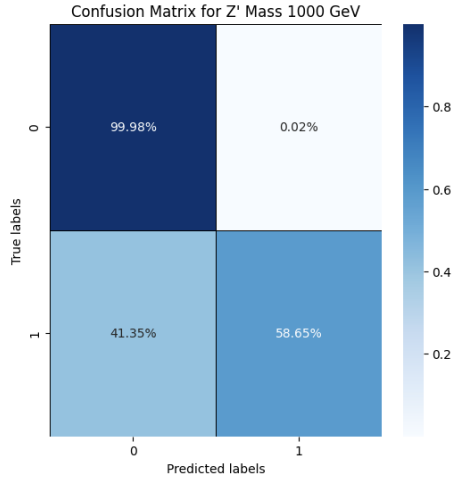


Figure 3.3.15: The confusion matrix for 1000 GeV signals classified shows true positives (58.65%), false positives (0.02%), false negatives (41.35%), and true negatives (99.98%).

Performance metrics	
Metrics	**Performance**
Accuracy	0.996
Precision	0.955
Recall	0.586
F1-Score	0.727
Significance	1.417
ROC Score	0.999
Optimal Cutoff*	0.9899

Table 3.3.18: Performance metrics for 1000 GeV signals classified by the best neural network classifier. *Optimal Cutoff value used to calculate the significance score.

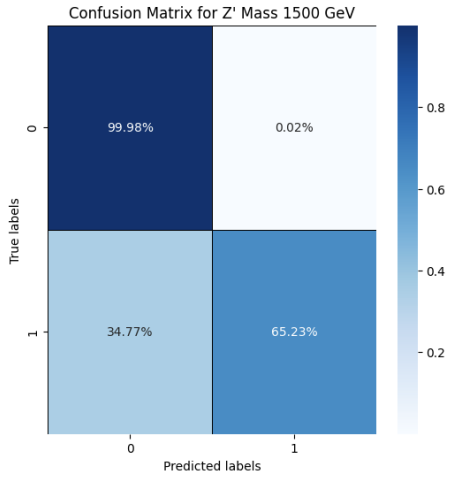


Figure 3.3.16: The confusion matrix for 1500 GeV signals classified shows true positives (65.23%), false positives (0.02%), false negatives (34.77%), and true negatives (99.98%).

Performance metrics	
Metrics	**Performance**
Accuracy	0.997
Precision	0.957
Recall	0.652
F1-Score	0.776
Significance	1.458
ROC Score	0.999
Optimal Cutoff*	0.9899

Table 3.3.19: Performance metrics for 1500 GeV signals classified by the best neural network classifier. *Optimal Cutoff value used to calculate the significance score.

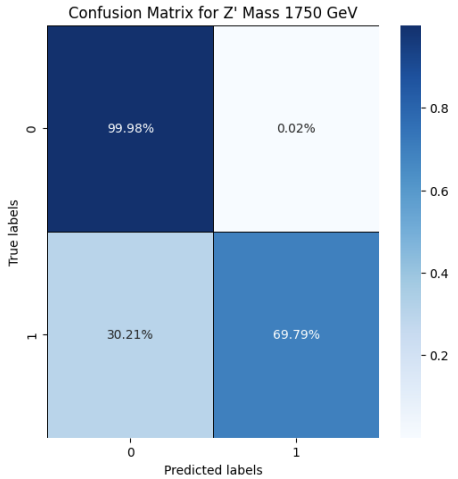


Figure 3.3.17: The confusion matrix for 1750 GeV signals classified shows true positives (69.79%), false positives (0.02%), false negatives (30.21%), and true negatives (99.98%).

Performance metrics	
Metrics	**Performance**
Accuracy	0.998
Precision	0.958
Recall	0.698
F1-Score	0.808
Significance	1.528
ROC Score	0.999
Optimal Cutoff*	0.9899

Table 3.3.20: Performance metrics for 1750 GeV signals classified by the best neural network classifier. *Optimal Cutoff value used to calculate the significance score.

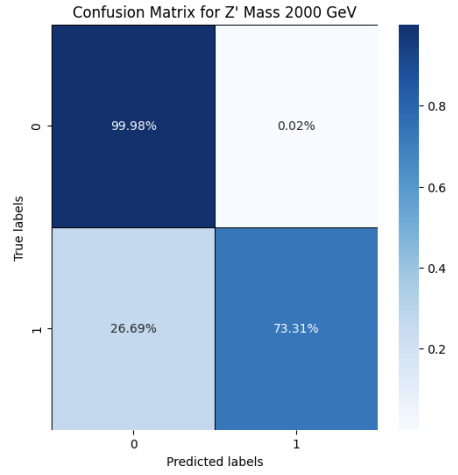


Figure 3.3.18: The confusion matrix for 2000 GeV signals classified shows true positives (73.31%), false positives (0.02%), false negatives (26.69%), and true negatives (99.98%).

Performance metrics	
Metrics	**Performance**
Accuracy	0.998
Precision	0.958
Recall	0.733
F1-Score	0.831
Significance	1.546
ROC Score	0.999
Optimal Cutoff*	0.9899

Table 3.3.21: Performance metrics for 2000 GeV signals classified by the best neural network classifier. *Optimal Cutoff value used to calculate the significance score.

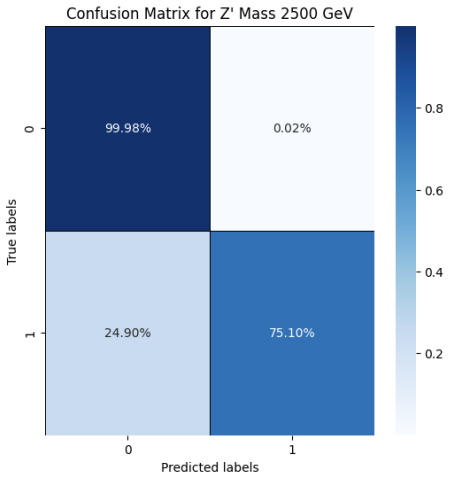


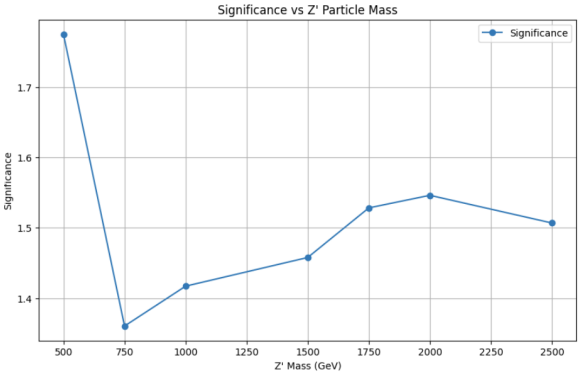
Figure 3.3.19: The confusion matrix for 2500 GeV signals classified shows true positives (75.10%), false positives (0.02%), false negatives (24.90%), and true negatives (99.98).

Performance metrics	
Metrics	**Performance**
Accuracy	0.998
Precision	0.956
Recall	0.751
F1-Score	0.841
Significance	1.507
ROC Score	0.999
Optimal Cutoff*	0.9899

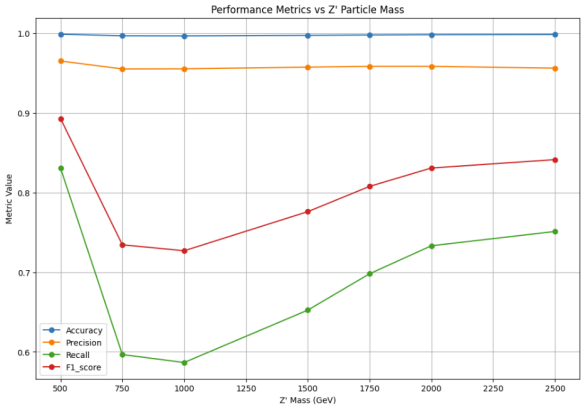
Table 3.3.22: Performance metrics for 2500 GeV signals classified by the best neural network classifier. *Optimal Cutoff value used to calculate the significance score.

Z' Mass (GeV)	Accuracy	Precision	Recall	F1-Score	ROC Score	Significance	Optimal Cutoff
500	0.9986	0.9650	0.8304	0.8926	0.9998	1.7751	0.9900
750	0.9967	0.9550	0.5965	0.7343	0.9994	1.3605	0.9899
1000	0.9965	0.9551	0.5865	0.7268	0.9993	1.4171	0.9899
1500	0.9972	0.9573	0.6523	0.7759	0.9992	1.4579	0.9899
1750	0.9976	0.9582	0.6979	0.8076	0.9992	1.5283	0.9899
2000	0.9980	0.9582	0.7331	0.8306	0.9992	1.5462	0.9899
2500	0.9982	0.9560	0.7510	0.8412	0.9990	1.5068	0.9899

Table 3.3.23: This table summarizes the performance metrics: accuracy, precision, recall, F1-score, ROC score, significance, and optimal cutoff values for Z' masses ranging from 500 GeV to 2500 GeV.



(i) Significance versus Z' Particle Mass.



(ii) Performance Metrics versus Z' Particle Mass.

Figure 3.3.20: Summary for significance and performance metrics for the classification of the Z' particle with mass ranging from 500-2500 GeV.

b) Regression model - mass estimation

The regression model's training over fifty epochs is summarized in Table 3.3.24. The learning rate was set at 1×10^{-2}, and epochs 0, 9, 19, 29, 39, and 49 showcase the model's development. The RMSE's standard deviation across these epochs is 441.88, reflecting early learning and subsequent stabilization.

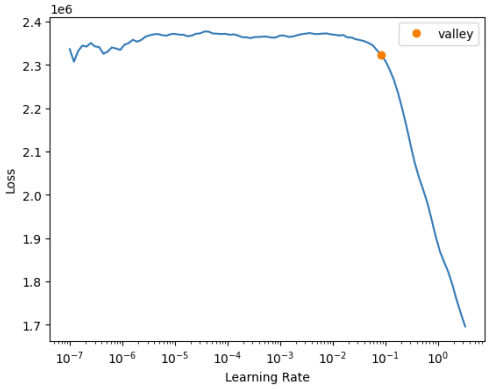


Figure 3.3.21: Results after training with a learning rate of 10^{-2}. The valley indicates a starting point for finding the optimal learning rate for the model.

Epoch	Train Loss	Valid Loss	RMSE	Time
0	2350989.00	2329793.50	1526.37	00:03
9	7301.82	1095.69	33.10	00:04
19	811.43	290.64	17.05	00:03
29	737.59	196.13	14.00	00:04
39	569.11	73.29	8.56	00:05
49	563.09	42.12	6.49	00:04

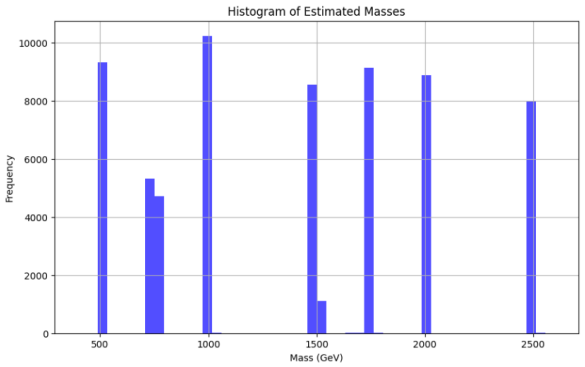
Table 3.3.24: Selected epoch training results for the regression model, indicating the train and validation losses, RMSE, and time per epoch. Early epochs show higher RMSE values which decrease as training progresses.

Metric	Value
Mean Squared Error (MSE)	42.1217
Root Mean Squared Error (RMSE)	6.4901
Mean Absolute Error (MAE)	2.2942
R-squared	0.9998

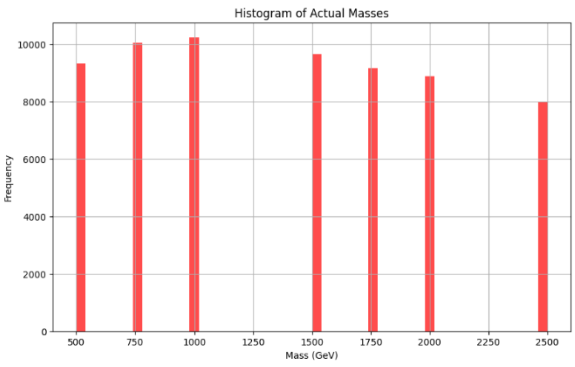
Table 3.3.25: Regression Model Performance Metrics for the model training.

Histograms of mass estimations and mass parameters on signal files

Figures 3.3.22i and 3.3.22ii present the distributions of estimated masses and actual mass parameters used in the study, respectively. The histograms facilitate a side-by-side visual comparison, serving as an assessment of the model's estimation accuracy against the actual mass values. Given that the regression model was trained exclusively on signal data, it was accordingly first tested on signal files with mass parameters set at 500 GeV, 750 GeV, 1000 GeV, 1500 GeV, 1750 GeV, 2000 GeV, and 2500 GeV. This approach ensured that the model's performance was evaluated on the same type of data on which it was trained.



(i) Histogram of mass estimations from the regression model.



(ii) Histogram of actual mass parameters for the dataset.

Figure 3.3.22: Comparative histograms of mass estimations and mass parameters.

Initial assessment of model accuracy

The scatter plot presented in Figure 3.3.23 offers an initial overview of the model's prediction accuracy by comparing estimated masses against actual masses. Points that lie close to the diagonal line, which represents perfect predictions, indicate where the model performs well. This plot serves as a baseline assessment, helping to identify general trends in prediction accuracy.

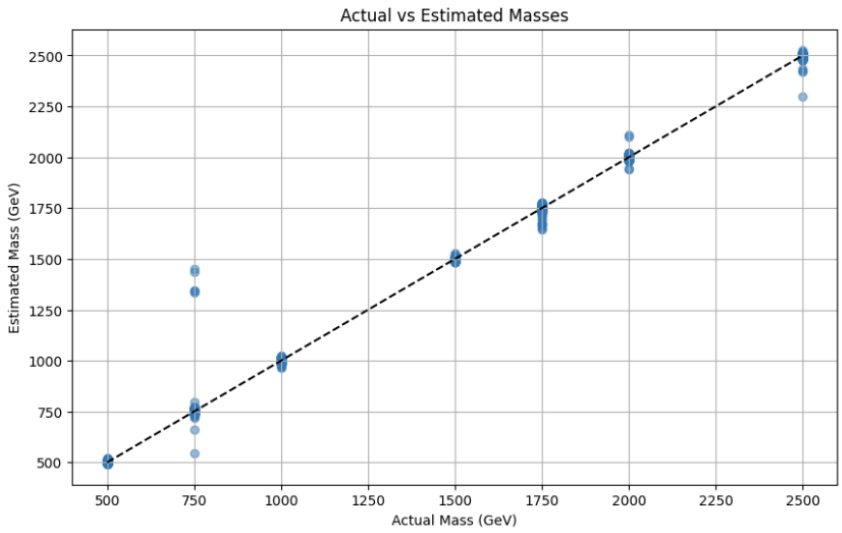
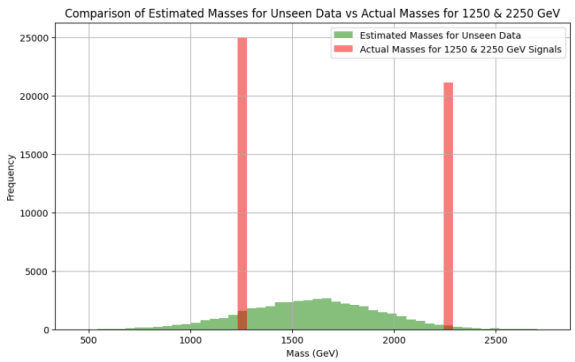
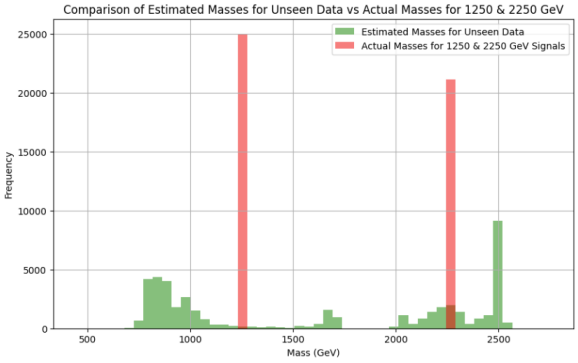


Figure 3.3.23: Scatter plot of actual versus estimated masses, where each point represents a test case with the x-coordinate indicating the actual mass (GeV) and the y-coordinate showing the estimated mass by the model. Points close to the diagonal line, which depicts perfect estimations, demonstrate high accuracy, while deviations from this line highlight areas needing improvement. For example, for an actual mass of 750 GeV, some estimations are misclassified around approximately 1400 GeV.

Figures 3.3.24i and 3.3.24ii show the model's estimations of masses for unseen data (1250 GeV and 2250 GeV) before and after training the model with the new features.



(i) Initial estimation of masses on unseen data before feature updates. (From Kaggle version 43 [48])



(ii) Comparison on how well the model estimated masses on unseen data after feature updates. (From Kaggle version 46 [48])

Figure 3.3.24: Comparative histograms showing the model's estimations of masses on unseen data before and after feature updates. These plots shows the significant improvement in model accuracy following the incorporation of new features calculating the invariant mass from jet decay.

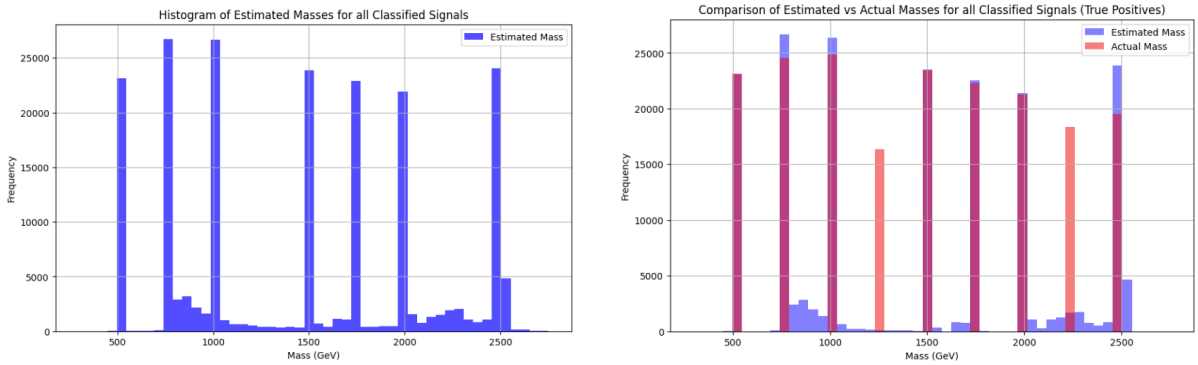
By integrating invariant mass calculations directly into the feature set, the regression model gains a refined understanding of the underlying physical phenomena, resulting in more precise mass estimations. This is particularly evident in the clear differentiation of the two mass signals in the updated model's predictions, showing the potential of tailored feature selection in improving the outcomes of machine learning applications in particle physics. The comparison of the models performance is shown in Table 3.3.26

Metric	Before	After	Improvement (%)
Mean Squared Error (MSE)	288897.9169	90919.6797	68.5%
Root Mean Squared Error (RMSE)	537.4922	301.5289	43.9%
Mean Absolute Error (MAE)	446.0714	269.0298	39.8%
R-squared	-0.1639	0.6337	-

Table 3.3.26: Regression Model Performance Metrics specifically for the unseen data (1250 GeV and 2250 GeV) before and after adding invariant mass calculated features. (From Kaggle version 43 vs 46 [48])

Estimated masses for all classified signal samples

To filter out signal from background events for mass estimation, the classifier was applied to all the data, including the new unseen datasets, resulting in an imbalance ratio of 1:107. Figure 3.3.25 presents the model's estimated mass distribution for all the files classified as signal (including misclassified background).



(i) Estimated Masses for all Classified Signals

(ii) Comparison of Estimated vs Actual Masses for all Classified Signals

Figure 3.3.25: Distributions and comparisons of estimated masses post-classification.

In Figure 3.3.26, the scatter plot contrasts actual and estimated masses for classified true positives. Each point, aligned with a theoretical ideal line representing perfect matches, provides insight into the model's estimation accuracy. Notably, the points for 1250 GeV and 2250 GeV, which represent new unseen data, deviate from this ideal line, revealing a reduced performance in these regions. Particularly for the 1250 GeV estimations, subtle variations in color intensity suggest a higher concentration of points around 750 GeV to 900 GeV, indicating that the cluster is centered lower than it might initially appear.

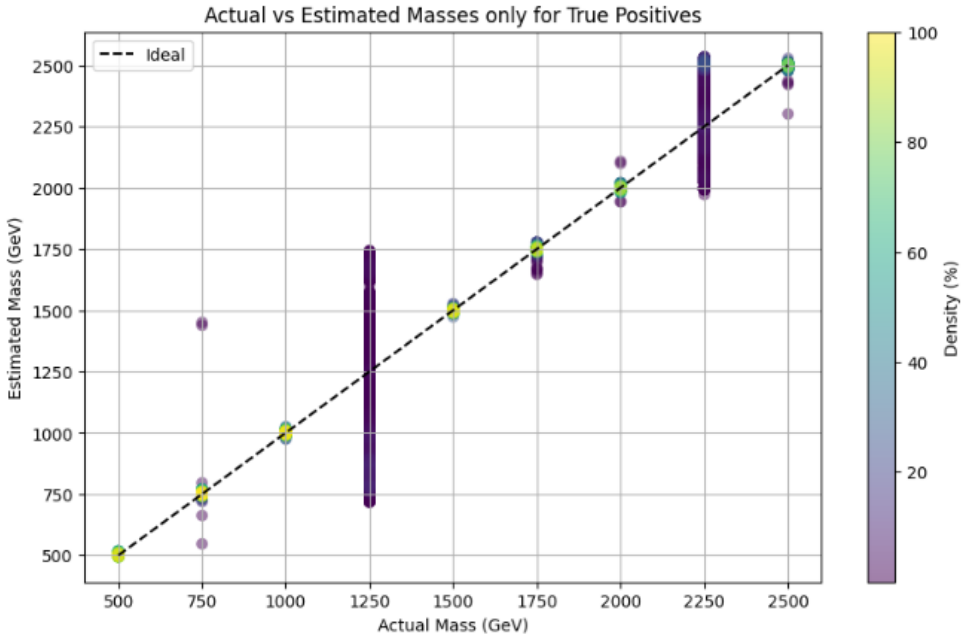


Figure 3.3.26: This scatter plot compares actual and estimated masses for true positives. The color intensity varies, indicating density levels of estimations. Estimations for 1250 GeV seem to center around 750 to 900 GeV, while those for 2250 GeV have a large number of estimations clustered around 2500 GeV. These patterns align with observations from previous analyses (see Figures 3.3.24ii and 3.3.25).

Classification performance on all data

The confusion matrix (Figure 3.3.27) shows the classification performance of the undersampled neural network model on all data, including the new unseen signal data. Performance metrics are listed alongside.

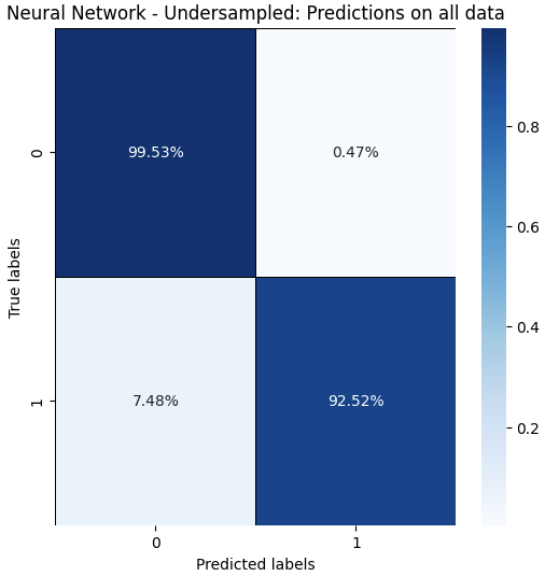


Figure 3.3.27: Confusion matrix of the model showing percentages of true positives (92.52%), false positives (0.47%), false negatives (7.48%), and true negatives (99.53%).

Metric	Performance
Accuracy	0.9910
Precision	0.9280
Recall	0.9252
F1 Score	0.9266
ROC Score	0.997

Table 3.3.27: Performance metrics of the neural network classifier, with an imbalance of 1:107.

Figure 3.3.28 shows the model's predicted confidence, showing a clear distinction between signal and background events. Most background events cluster at lower confidences, and signal events skew towards higher values, affirming the model's discriminative power.

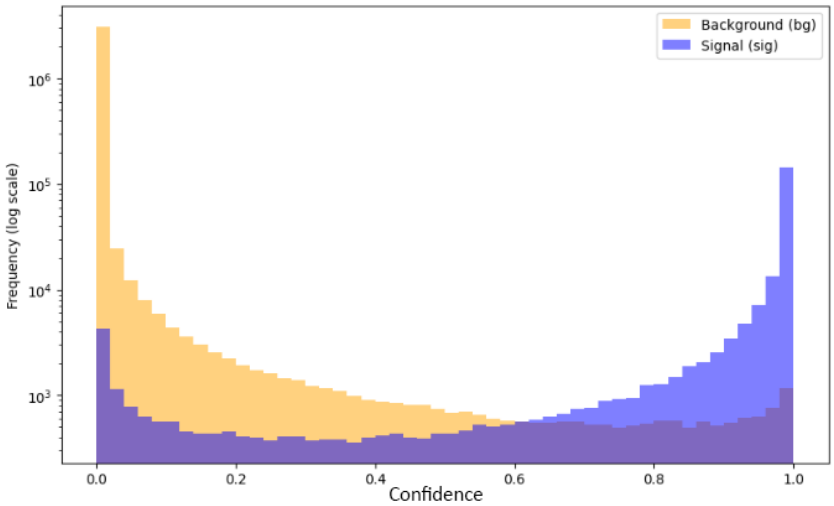


Figure 3.3.28: Histogram showing the distribution of confidence for signal and background classes. The histogram is plotted with a logarithmic scale on the y-axis to account for the disparity in class sizes.

4 Discussion

In this study, advanced machine learning methods were employed to investigate the identification and measurement of the hypothetical Z' particle using simulated data from the ATLAS experiment at the LHC. The models developed in this study highlighted significant insights and implications regarding the robustness and limitations of machine learning in high-energy physics, especially in handling imbalanced datasets and predicting rare event signals.

a) Data exploration

The initial analysis of the Monte Carlo datasets from the ATLAS experiment involved detailed statistical evaluations and visualizations, including histograms and correlation heatmaps for jet and lepton features. Key findings included skewed distributions and, more importantly, differences in scale between features, which necessitated normalization to enhance model training. The feature importance analysis, significantly influenced by insights from the ATLAS Research Group, uncovered several key variables that impacted the model performance. For instance, features like "met_et", "jet_1_m", "jet_1_pt", and "lep_1_pt" were identified as the most influential variables in optimizing the classifier's performance, with their respective feature-importance scores ranging between approximately 4% and 7%.

Moreover, the comprehensive preparation of the datasets was evident as they exhibited no missing values and displayed a high level of compatibility, facilitating seamless integration and manipulation during the analysis. This level of data integrity not only simplified the preprocessing steps but also enhanced the robustness of the subsequent machine learning models. During this phase, we focused on studying metadata like "eventWeight" and "mcWeight" to understand their implications for later phases, particularly in significance calculations. However, it was determined that scale factors such as "scaleFactor_BTAG", "scaleFactor_ELE", and "scaleFactor_MUON", while crucial for aligning simulations with experimental conditions, were excluded from the training process to prevent model overfitting and ensure that the training focused solely on inherent data characteristics.

b) Classification

Our classifier models initially exhibited artificially high accuracy due to severe data imbalance. To address this, we employed various sampling techniques. The initial use of oversampling techniques effectively equalized class distribution but led to overtraining, as evident from the perfect recall scores in the oversampled models. These models, though initially promising due to their high performance metrics, were ultimately disregarded because they had become overtrained on too many repetitive duplicates of signal data. On the other hand, the undersampling technique proved superior; it employed a signal-to-

background sample ratio of 1:10, which significantly enhanced the models' generalization capabilities, striking a better balance between precision and recall, as detailed in Section 3.

The baseline models (with a traditional approach of not adjusting the class imbalance) showed a high overall accuracy but significantly lower recall rates, indicating a tendency to favor the majority class excessively. This was further evidenced by the performance metrics of the baseline random forest model, which had a recall of only 37.95%, signifying poor identification of the minority class. Conversely, the undersampled model displayed a much higher recall of 74.71%, reflecting an improved ability to identify both classes equally well without sacrificing precision, which remained commendably high at 90.2%. The undersampled model not only demonstrates high accuracy but also maintains high levels of other critical metrics like F1-Score and ROC, indicating robustness against varying data scenarios.

This balanced approach in the undersampling method underscores the importance of precision in handling data imbalances, facilitating models that are more reflective of the real-world distributions they are meant to interpret. Additionally, the neural network models, particularly the undersampled neural network, exhibited even more promising results. The undersampled neural network achieved a precision of 95.79%, a recall of 96.48%, and an F1-score of 96.13%, outperforming the random forest models and demonstrating enhanced capability in handling imbalances and maintaining high performance across various metrics.

c) Classification for different Z' mass hypotheses

The performance of the undersampled neural network model, when employed to classify signal datasets over a mass range from 500 GeV to 2500 GeV for the Z' particle, led to the following observations:

The recall metric demonstrated a non-linear response; it initially peaked at 500 GeV with a recall of 0.8304, then declined through 1000 GeV before progressively increasing up to 2500 GeV. This pattern suggests that while the model faced difficulties in differentiating signal from background at intermediate masses, it adjusted better at higher masses. Furthermore, the observed significance, starting at a high of 1.7751 at 500 GeV, indicated intriguing findings but did not meet the gold standard of 5σ. The significance then dropped at 750 GeV before gradually increasing, mirroring the recall's trend and suggesting an enhanced model confidence with higher masses, though still not achieving definitive discovery.

Additionally, the optimal cutoff values utilized for determining significance were consistent across the different mass settings, ranging closely around 0.99. This uniform

approach in setting cutoff values might have influenced the results, particularly in the context of significance, where despite a similar number of true positives before weighting, the significance should theoretically decrease with increasing mass. This expectation stems from the principle that with a growing mass, the challenge in distinguishing between increasingly rare signal events and background noise should intensify, theoretically diminishing the significance unless the model's sensitivity markedly improves. The consistency in significance levels across various masses, therefore, raises questions about the cutoff strategy's impact on the measured outcomes, suggesting that adjustments or a more dynamic approach to setting these thresholds might provide a clearer understanding of the model's performance across the studied mass spectrum.

d) Mass estimation

The regression model developed to estimate the Z' particle's mass demonstrated variable performance across the mass spectrum. Initially, it performed exceptionally well on familiar mass values from the training set, such as 500 GeV, 750 GeV, etc. The inclusion of invariant mass calculations in the new variables significantly improved the model's accuracy, especially on the new unseen data, as evidenced by a dramatic 44% reduction in Root Mean Squared Error and a substantial increase in R-squared value from -16.39% to 63.37%.

Building on these initial evaluations, the regression model was then applied to all files classified as signals by the undersampled neural network model, including misclassified background files. In this broader application for mass estimation, the model yielded a Mean Squared Error (MSE) of 8480, a Root Mean Squared Error (RMSE) of 92.1, a Mean Absolute Error (MAE) of 41.3, and an R-squared of 0.979. Although these metrics indicate a slight performance degradation compared to the initial training phase, they still affirm the model's high accuracy in practical testing scenarios. This phase of analysis highlights the effectiveness of the neural network in classifying signal files and underscores the regression model's capability to reliably estimate mass from the data classified as signals.

That being said, Figure 3.3.25i illustrates a clear distinction between the mass estimations from trained data and those from new, unseen data. The discrepancies observed on unseen data, with predictions for the new mass points (1250 GeV and 2250 GeV) being notably more spread, underscore the challenges and the necessity for continuous refinement of both the feature set and model parameters.

Future directions and enhancements

This paper follows a framework established by the Atlas research group with focus on utilizing machine learning techniques to analyze particle collision data. Grefsrud et al. (2024) applied advanced models like XGBoost and ResNet for image classification, converting complex particle interactions into image data for improved detection at the LHC [9].

On the other hand, our research diverges significantly by leveraging simulated tabular data derived from the ATLAS detector. Rather than converting complex particle interactions into image data, we analyze structured, high-dimensional tabular data that includes a variety of physical measurements such as energy levels, momenta, and other particle identifiers. This approach allows for a more direct interpretation of the raw numerical data, facilitating detailed statistical analysis and feature selection that are not typically focused on in image-based analysis. By focusing on new particle physics phenomena through the novel use of deep learning, we believe our approach, however humble, makes a valuable contribution to the progress of this field.

It is important to acknowledge that this investigation encountered several limitations. Interpreting high-energy particle physics data was challenging without specialized knowledge in the field. Furthermore, the selection of machine learning algorithms might have been biased, which could distort the results. Moreover, the significant imbalance within the dataset added complexity to the training and evaluation processes of the modela. While methods such as oversampling were employed, they led to overtraining, evident from the perfect recall scores in oversampled models. This indicates that models likely memorized the signal characteristics rather than learning to generalize, a limitation that could potentially be mitigated by employing more sophisticated techniques like Synthetic Minority Over-sampling Technique (SMOTE) [49]. Further testing and development of our undersampling technique could also enhance the model's performance. Alternatively, techniques such as weighted loss functions, the focal loss function, and balanced random forests could further enhance the robustness and fairness of the model training process, as detailed in the study on class imbalance techniques for high energy physics by Christopher W. Murphy [50].

Additionally, the scalability of our approaches was limited by the large sizes of the datasets and the high processing needs. Assessing the benefits of ensemble methods, such as XGBoost [51] —a highly efficient implementation of Gradient Boosting, might prove useful in enhancing the performance of the classifiers. These methods have shown promise in improving model robustness and generalization capabilities in several fields. Incorporating physics-based constraints on the models, such as conservation laws and symmetry properties, may improve accuracy and reliability on both classifier and regression models.

Furthermore, exploring new features based on domain knowledge of particle physics, like derived variables that may capture underlying physical processes better than raw measurements,

could enhance both classification and regression model performance, as our regression model demonstrated. Finally, more testing can be done on the benefits of using a transfer learning approach, potentially accelerating the model's training time and improving its performance, especially when dealing with limited labeled data or complex feature spaces [52].

5 Conclusion

Our research utilized advanced machine learning techniques to enhance the detection of the Z' particle and estimate its parameters using simulated data from the ATLAS experiment. The classifiers, while initially showing high accuracy due to data imbalance, were effectively optimized through techniques like undersampling, which improved their ability to generalize across different scenarios. Despite the artificial inflation of accuracy, the classifiers performed commendably across other metrics, establishing their efficacy in high-energy particle physics applications. The regression model, tailored for mass estimation, excelled with training data but encountered challenges with new, unseen data points. The subsequent integration of features based on invariant mass calculations markedly improved performance, showcasing the model's adaptability and the potential for further refinement. As we look forward, it becomes apparent that further advancements could be achieved by exploring ensemble methods and integrating physics-based constraints to enhance the robustness and accuracy of the models. Collaborative efforts and further research are essential to overcome existing limitations and fully harness the potential of machine learning in this complex domain.

6 References

[1] *Large Hadron Collider (LHC) | Definition, Discoveries, & Facts | Britannica*. Accessed: 12/02/2024. URL: https://www.britannica.com/technology/Large-Hadron-Collider.

[2] *The Large Hadron Collider | CERN*. Accessed: 12/02/2024. URL: https://home.web.cern.ch/science/accelerators/large-hadron-collider.

[3] *The Standard Model | CERN*. Accessed: 10/02/2024. URL: https://home.web.cern.ch/science/physics/standard-model.

[4] Paul Langacker. "The Physics of Heavy Z Gauge Bosons". In: (2009). Accessed: 12/02/2024. URL: https://inspirehep.net/literature/777086.

[5] Matthew D. Schwartz. "Modern Machine Learning and Particle Physics". In: *Harvard Data Science Review* (Mar. 2021). Accessed: 13/02/2024. DOI: 10.1162/99608f92.beeb1183. URL: http://arxiv.org/abs/2103.12226%20http://dx.doi.org/10.1162/99608f92.beeb1183.

[6] *HVL-ATLAS-Group (n.d), 'Learning dark matter'*. Accessed: 08/02/2024. URL: https://learningdarkmatter.com/.

[7] Vladimirs Sergejevics Civilgins and Sunniva Storetvedt Lothe. "Utnytting av symmetri til å kunstig øke mengden treningsdata for klassifisering av mikroskopiske sorte hull og sfaleroner – ved bruk av simulerte kollisjonsdata fra ATLAS (HVL)". In: (2023). Accessed: 03/03/2024. URL: https://hvlopen.brage.unit.no/hvlopen-xmlui/handle/11250/3082970.

[8] Sayna Ganjei and Daniel Kristiansen Gunleiksrud. "Deep learning approach for binary classification of microscopic black holes and sphalerons. Optimization by employing custom loss function". In: (2023). Accessed: 03/03/2024. URL: https://hvlopen.brage.unit.no/hvlopen-xmlui/handle/11250/3082038.

[9] Aurora Singstad Grefsrud et al. "Machine Learning Classification of Sphalerons and Black Holes at the LHC". In: (Oct. 2023). Accessed: 03/03/2024. URL: https://arxiv.org/abs/2310.15227v1.

[10] Lars E. Risholm and Marius S. Hauger. "Search for new physics at the Large Hadron Collider (LHC)," in: (2024). Accessed: 27/04/2024. URL: https://github.com/L596241/Search-for-new-physics-at-the-LHC.

[11] *MC Simulations - CMS Open Data Guide*. URL: https://cms-opendata-guide.web.cern.ch/analysis/datasim/mcsimulations/.

[12] *scikit-learn: machine learning in Python — scikit-learn 1.4.1 documentation*. Accessed: 03/03/2024. URL: https://scikit-learn.org/stable/.

[13] *pandas documentation — pandas 2.2.1 documentation*. Accessed: 03/03/2024. URL: https://pandas.pydata.org/docs/.

[14] *Practical Deep Learning for Coders - Practical Deep Learning*. Accessed: 03/03/2024. URL: https://course.fast.ai/.

[15] *Matplotlib documentation — Matplotlib 3.8.3 documentation*. Accessed: 03/03/2024. URL: https://matplotlib.org/stable/index.html.

[16] *Kaggle: Your Home for Data Science*. Accessed: 03/03/2024. URL: https://www.kaggle.com/.

[17] Klaus Rabbertz. "Jet Physics at the LHC". In: 268 (2017). Accessed: 29/02/2024. DOI: 10.1007/978-3-319-42115-5. URL: http://link.springer.com/10.1007/978-3-319-42115-5.

[18] F. Close. *Particle Physics: A Very Short Introduction*. EBSCO ebook academic collection. Accessed: 29/02/2024. OUP Oxford, 2004. ISBN: 9780192804341. URL: https://books.google.no/books?id=hwYTDAAAQBAJ.

[19] *International Physics Masterclasses*. Accessed: 13/02/2024. URL: https://atlas.physicsmasterclasses.org/en/zpath_protoncollisions.htm.

[20] Thomas SchÖRner-Sadenius. *The large Hadron Collider: Harvest of run 1*. Accessed: 29/02/2024. Springer International Publishing, Jan. 2015, p. 146. ISBN: 9783319150017. DOI: 10.1007/978-3-319-15001-7/COVER.

[21] *International Physics Masterclasses | Z boson*. Accessed: 13/02/2024. URL: https://atlas.physicsmasterclasses.org/en/zpath_lhcphysics2.htm.

[22] M. Aaboud et al. "Constraints on mediator-based dark matter and scalar dark energy models using

$$\sqrt{\{s\}}$$

= 13 TeV pp collision data collected by the ATLAS detector". In: *Journal of High Energy Physics 2019 2019:5* 2019 (5 May 2019). Accessed: 04/04/2024, pp. 1–87. ISSN: 1029-8479. DOI: 10.1007/JHEP05(2019)142. URL: https://link.springer.com/article/10.1007/JHEP05(2019)142.

[23] Siddharth Dwivedi et al. "Associated zroduction in the flavorful U(1) scenario for RK($*$)". In: *The European Physical Journal C 2020 80:3* 80 (3 Mar. 2020). Accessed: 05/03/2024, pp. 1–13. ISSN: 1434-6052. DOI: 10.1140/EPJC/S10052-020-7810-4. URL: https://link.springer.com/article/10.1140/epjc/s10052-020-7810-4.

[24] *What is AI? | Quick Start Guide for Beginners*. Accessed: 23/03/2024. URL: https://www.datacamp.com/community/blog/what-is-ai-quick-start-guide-for-beginners.

[25] *Schematic illustration of Artificial Intelligence*. Accessed: 23/03/2024. URL: https://www.mdpi.com/2504-4990/3/3/32.

[26] *Divisions and subdivisions of machine learning algorithms | Download Scientific Diagram*. Accessed: 23/03/2024. URL: https://www.researchgate.net/figure/Divisions-and-subdivisions-of-machine-learning-algorithms_fig1_374010223.

[27] Pádraig Cunningham, Matthieu Cord, and Sarah Jane Delany. "Supervised learning". In: *Cognitive Technologies* (2008). Accessed: 04/03/2024, pp. 21–49. ISSN: 16112482. DOI: 10.1007/978-3-540-75171-7_2/COVER. URL: https://link.springer.com/chapter/10.1007/978-3-540-75171-7_2.

[28] *Classification in Machine Learning: A Guide for Beginners | DataCamp*. Accessed: 04/03/2024. URL: https://www.datacamp.com/blog/classification-machine-learning.

[29] Keiron O'shea and Ryan Nash. "An Introduction to Convolutional Neural Networks". In: (). Accessed: 04/03/2024.

[30] *Epoch in Machine Learning - GeeksforGeeks*. Accessed: 05/03/2024. URL: https://www.geeksforgeeks.org/epoch-in-machine-learning/.

[31] *callbacks.lr$_{finder|fastai}$*. Accessed: 10/05/2024. URL: https://fastai1.fast.ai/callbacks.lr_finder.html.

[32] *The Complete Guide on Overfitting and Underfitting in Machine Learning*. Accessed: 04/03/2024. URL: https://www.simplilearn.com/tutorials/machine-learning-tutorial/overfitting-and-underfitting.

[33] *The Complete Guide on Overfitting and Underfitting in Machine Learning*. Accessed: 27/04/2024. URL: https://www.simplilearn.com/tutorials/machine-learning-tutorial/overfitting-and-underfitting.

[34] Shimon Whiteson and Daniel Whiteson. "Machine learning for event selection in high energy physics". In: *Engineering Applications of Artificial Intelligence* 22 (8 Dec. 2009). Accessed: 04/03/2024, pp. 1203–1217. ISSN: 0952-1976. DOI: 10.1016/J.ENGAPPAI.2009.05.004.

[35] *Confusion Matrix in Machine Learning - GeeksforGeeks*. Accessed: 18/03/2024. URL: https://www.geeksforgeeks.org/confusion-matrix-machine-learning/.

[36] *Precision-Recall — scikit-learn 1.4.1 documentation*. Accessed: 04/03/2024. URL: https://scikit-learn.org/stable/auto_examples/model_selection/plot_precision_recall.html#precision-recall.

[37] *sklearn.metrics.f1$_{s}core^{-}scikit - learn1.4.1documentation$*. Accessed: 04/03/2024. URL: https://scikit-learn.org/stable/modules/generated/sklearn.metrics.f1_score.html.

[38] *Understanding AUC - ROC Curve | by Sarang Narkhede | Towards Data Science*. Accessed: 04/03/2024. URL: https://towardsdatascience.com/understanding-auc-roc-curve-68b2303cc9c5.

[39] *AUC ROC Curve in Machine Learning - GeeksforGeeks - image*. Accessed: 24/03/2024. URL: https://www.geeksforgeeks.org/auc-roc-curve/.

[40] *Statistical Significance | ATLAS Experiment at CERN*. Accessed: 23/03/2024. URL: https://atlas.cern/node/38039.

[41] *Why do physicists mention "five sigma" in their results?* Accessed: 23/03/2024. URL: https://home.cern/resources/faqs/five-sigma.

[42] *ZparticleSearch - monte carlo simulation datasets*. Accessed: 15/02/2024. URL: https://www.kaggle.com/datasets/larserikrisholm/zparticlesearch.

[43] *Variable names | ATLAS Open Data*. Accessed: 02/03/2024. URL: https://opendata.atlas.cern/docs/8TeVDoc/variable_names/.

[44] E Bothmann et al. "A standard convention for particle-level Monte Carlo event-variation weights The MCnet Community". In: (2022). Accessed: 24/03/2024.

[45] J.-C Walter and G T Barkema. "An introduction to Monte Carlo methods". In: (2014). Accessed: 24/03/2024.

[46] *Search for new physics at the LHC: Part 1/3*. Accessed: 04/04/2024. URL: https://www.kaggle.com/code/larserikrisholm/search-for-new-physics-at-the-lhc-part-1-3.

[47] *Search for new physics at the LHC: Part 2/3*. Accessed: 04/04/2024. URL: https://www.kaggle.com/code/larserikrisholm/search-for-new-physics-at-the-lhc-part-2-3.

[48] *Search for new physics at the LHC: Part 3/3*. Accessed: 04/04/2024. URL: https://www.kaggle.com/code/larserikrisholm/search-for-new-physics-at-the-lhc-part-3-3.

[49] *Synthetic Minority Oversampling (SMOTE) in ML: Techniques Examples*. Accessed: 09/05/2024. URL: https://domino.ai/blog/smote-oversampling-technique.

[50] Christopher Murphy. "Class imbalance techniques for high energy physics". In: *SciPost Physics* 7 (Dec. 2019). Accessed: 09/05/2024. DOI: 10.21468/SciPostPhys.7.6.076.

[51] *XGBoost Documentation — xgboost 2.0.3 documentation*. Accessed: 09/05/2024. URL: https://xgboost.readthedocs.io/en/stable/.

[52] *A Gentle Introduction to Transfer Learning for Deep Learning - MachineLearningMastery.com*. Accessed: 09/05/2024. URL: https://machinelearningmastery.com/transfer-learning-for-deep-learning/.

7 Appendix

A Gantt Chart

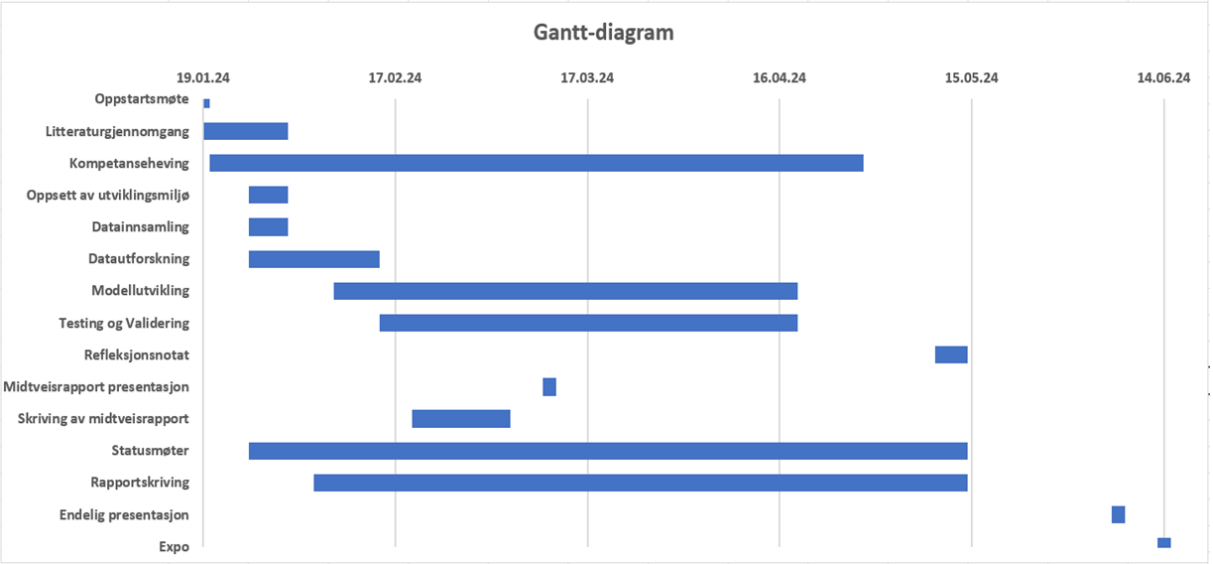


Figure 7.A.1: Gantt chart illustrating the scheduled tasks and their durations as planned for the project.

B Risk assessment

	Hendelser	Årsak(er)	Sannsynlighet	Virkning(er)	Konsekvens	Risikoprodukt	Risikoreduserende tiltak
	Hvilke uønskede hendelser kan inntreffe og hindre oppnåelse av målet?	Hva er årsaken(e) til at denne hendelsen kan inntreffe?	Sannsynligheten for hver hendelse	Hva er virkningen(e) av hendelsen du har identifisert?	Grad av konsekvens		Tiltak som reduserer risiko
1	Kompetansegap i prosjektteamet.	Manglende erfaring med komplekse datanalyser eller partikkelfysikk.	Sannsynlig	Redusert produktivitet og mulige misforståelser.	Svært høy	20	Studere relevant literatur, teknikk og konsepter.
2	Feiltolkning av resultater for deteksjon av Z'-partikkelen.	Utilstrekkelig forståelse av fysiske data og maskinlæringsmodeller.	Sannsynlig	Negative konsekvenser for prosjektets vitenskapelige nøyaktighet.	Høy	16	Regelmessig gjennomgang og kvalitetssikring av dataanalyseprosessen.
3	Feiltolkning av data fra datautforskningen.	Manglende evne til å forstå de fysiske prosessene registrert i de simulerte datasettene.	Sannsynlig	Feilaktige resultater som kan lede til feil konklusjoner.	Høy	16	Bruk av avanserte teknikker for støyreduksjon og forbedring av datarensing
4	Tekniske problemer i utviklingsmiljø.	Maskinvarefeil eller programvarekompatibilitets-problemer.	Middels	Prosjektets fremdrift blir forsinket.	Høy	12	Anvend riktig utviklingsmiljø og opprett sikkerhetskopier.
5	Ineffektiv modelltrening på grunn av begrenset datakraft. (CPU/RAM)	Store mengde data som skal trenes og evalueres.	Middels	Prosjektets fremdrift blir forsinket.	Høy	12	Benyttelse av datadelsett for å optimalisere minnebruk og CPU-effektivitet under modelltrening.
6	Utilstrekkelig samarbeid og kommunikasjon i teamet.	Dårlig definerte roller eller ansvarsområder.	Middels	Redusert produktivitet og mulige misforståelser.	Middels	9	Regelmessige teammøter, klar kommunikasjon.

Figure 7.B.2: Overview of key project risks and mitigation strategies.

C Jet and Lep feature histograms

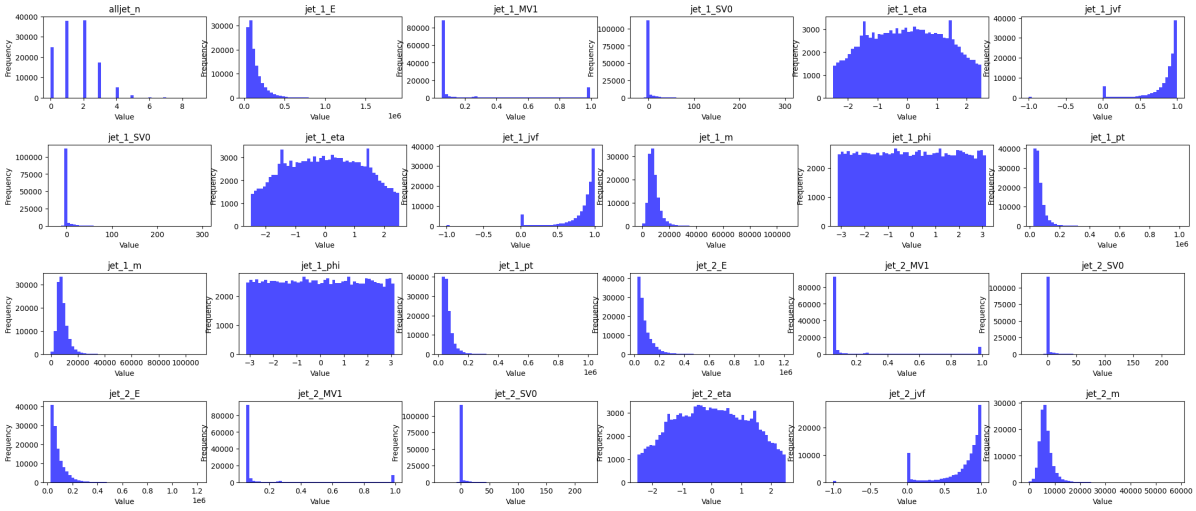


Figure 7.C.3: Feature histograms

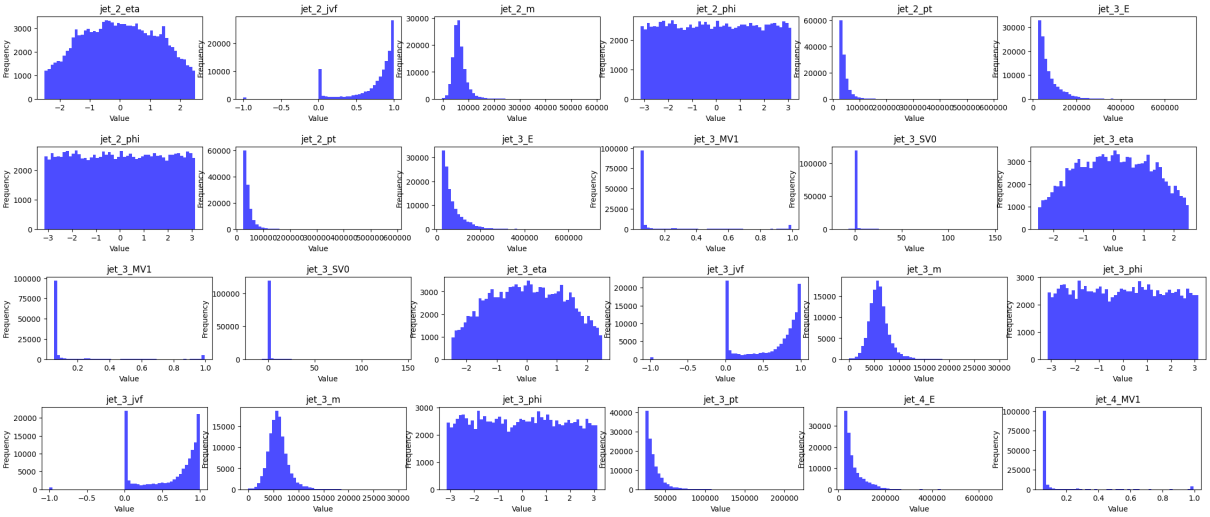


Figure 7.C.4: Feature histograms

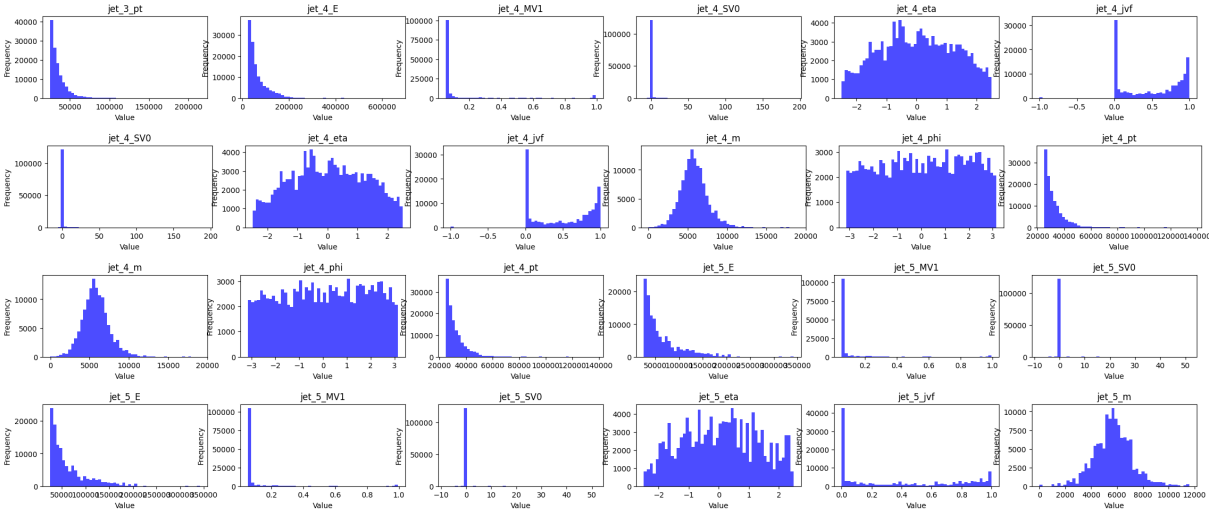


Figure 7.C.5: Feature histograms

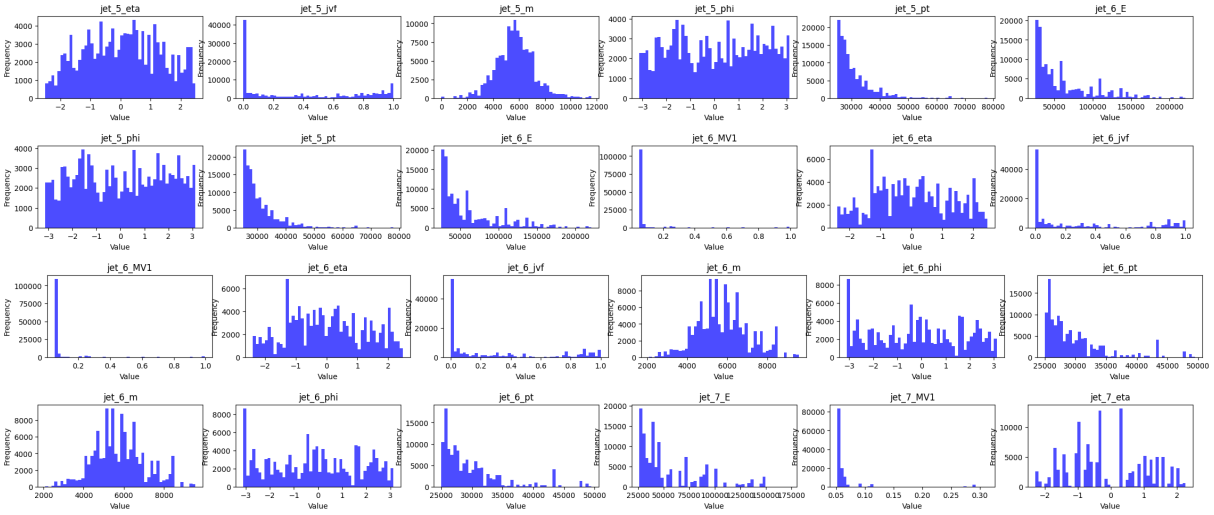


Figure 7.C.6: Feature histograms

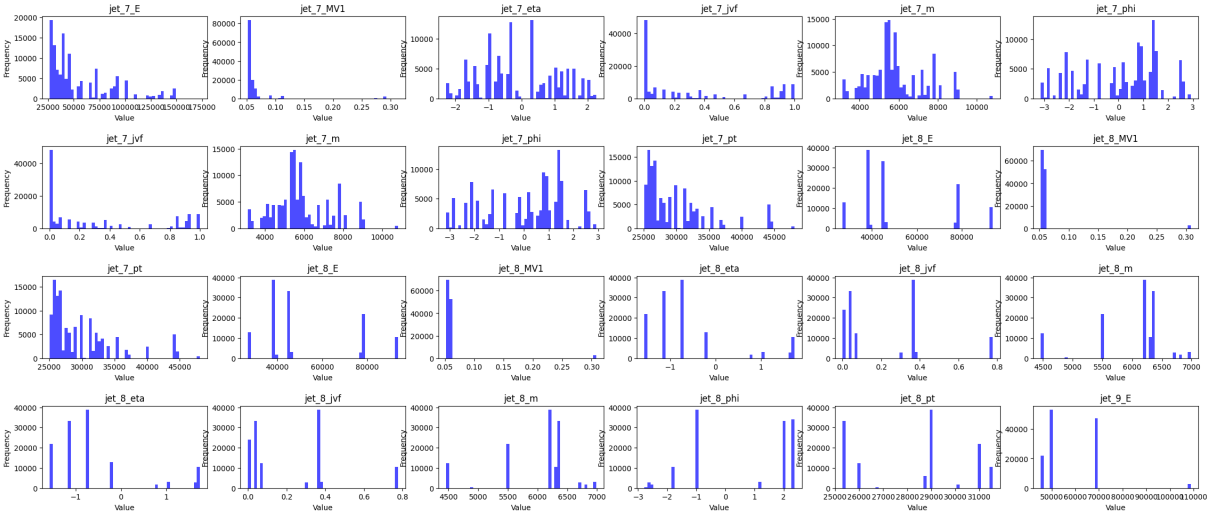


Figure 7.C.7: Feature histograms

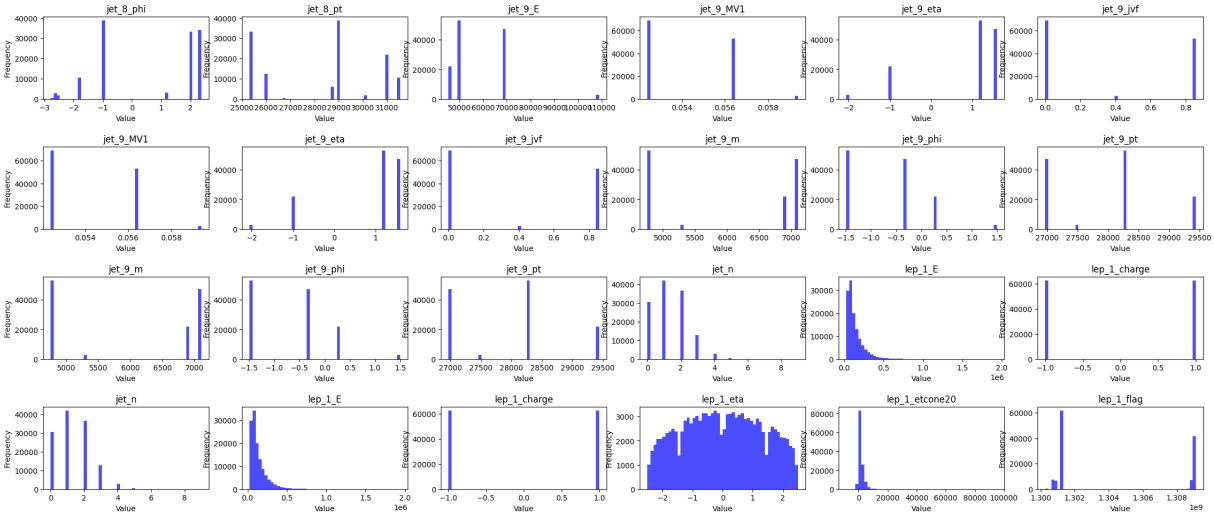


Figure 7.C.8: Feature histograms

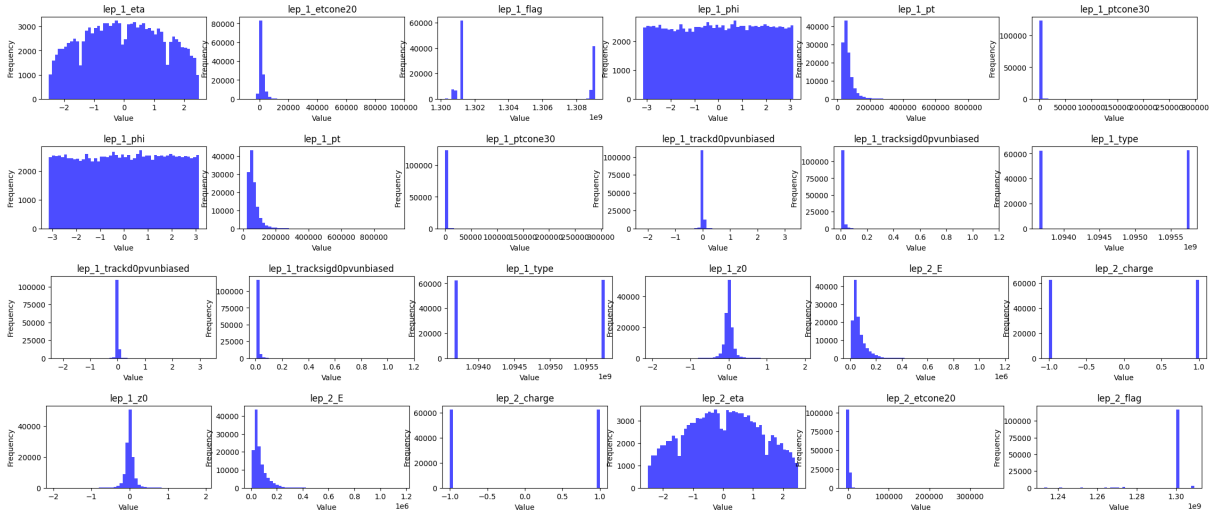


Figure 7.C.9: Feature histograms

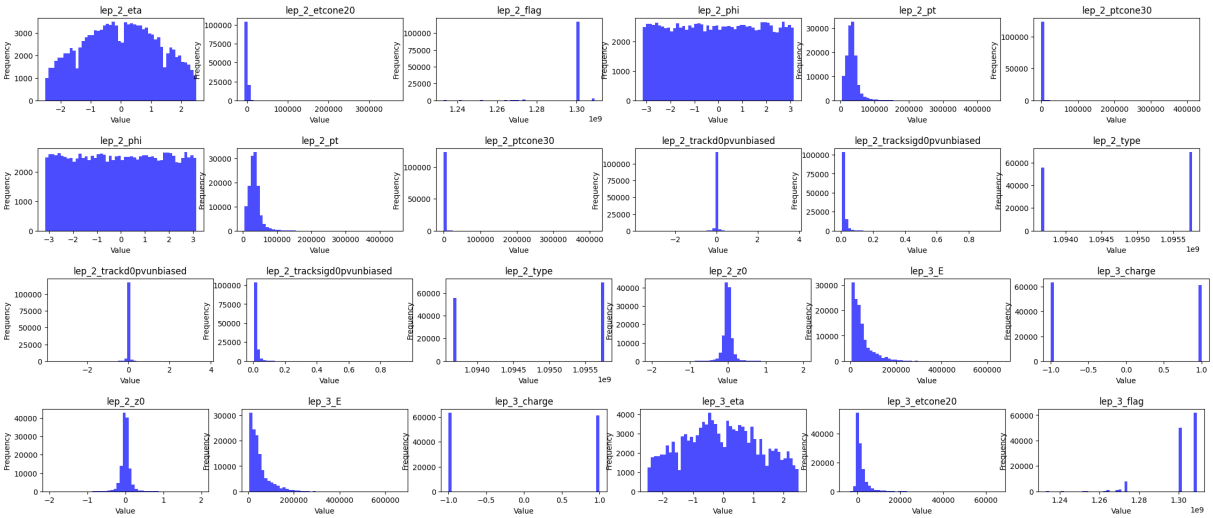


Figure 7.C.10: Feature histograms

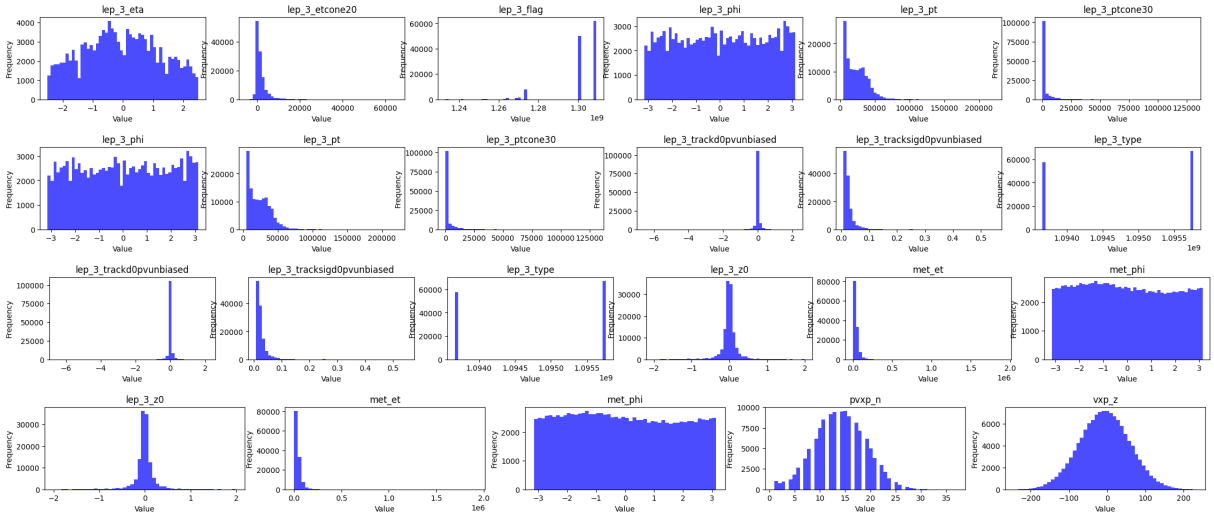


Figure 7.C.11: Feature histograms

D Complete training and validation with 50 epochs for the regression model

Table 7.4.1: Complete Training and Validation with 50 Epochs for the Regression Model

Epoch	Train Loss	Valid Loss	RMSE	MSE	MAE	R2 Score	Time
0	2350989.000	2329793.500	1526.37	2329793.500	1387.73	-4.5314	00:03
1	2326283.250	2292158.250	1513.99	2292158.250	1383.94	-4.4421	00:04
2	2266187.750	2205153.250	1484.98	2205153.250	1370.77	-4.2355	00:03
3	2123028.000	1981692.250	1407.73	1981692.250	1323.32	-3.7049	00:03
4	1801505.750	1570904.000	1253.36	1570904.000	1218.38	-2.7296	00:03
5	1296204.375	1011602.375	1005.78	1011602.375	1002.64	-1.4018	00:03
6	720953.812	406873.781	637.87	406873.781	636.10	0.0340	00:03
7	235471.594	53064.793	230.36	53064.793	222.94	0.8740	00:03
8	42175.813	1665.836	40.81	1665.836	31.59	0.9960	00:03
9	7301.821	1095.694	33.10	1095.694	23.87	0.9974	00:04
10	2278.098	1448.708	38.06	1448.708	26.95	0.9966	00:03
11	1489.174	486.911	22.07	486.911	15.62	0.9988	00:03
12	1202.062	696.083	26.38	696.083	18.50	0.9983	00:03
13	1184.137	529.725	23.02	529.725	16.42	0.9987	00:03
14	1186.375	530.868	23.04	530.868	15.82	0.9987	00:03
15	1075.815	946.685	30.77	946.685	23.12	0.9978	00:03
16	1238.171	1014.205	31.85	1014.205	24.35	0.9976	00:03
17	840.954	311.035	17.64	311.035	11.89	0.9993	00:04
18	843.714	519.926	22.80	519.926	15.69	0.9988	00:03
19	811.432	290.643	17.05	290.643	12.73	0.9993	00:03
20	840.657	327.261	18.09	327.261	12.41	0.9992	00:03

Table 7.4.1 continued from previous page

Epoch	Train Loss	Valid Loss	RMSE	MSE	MAE	R2 Score	Time
21	1119.188	243.160	15.59	243.160	9.93	0.9994	00:03
22	868.210	322.330	17.95	322.330	12.76	0.9992	00:04
23	778.519	379.718	19.49	379.718	14.82	0.9991	00:03
24	778.582	444.723	21.09	444.723	14.35	0.9989	00:03
25	879.049	275.875	16.61	275.875	11.47	0.9993	00:04
26	618.338	216.696	14.72	216.696	11.03	0.9995	00:03
27	890.185	264.560	16.27	264.560	10.72	0.9994	00:04
28	814.186	274.843	16.58	274.843	12.96	0.9993	00:03
29	737.586	196.125	14.00	196.125	8.97	0.9995	00:04
30	666.618	175.555	13.25	175.555	9.83	0.9996	00:03
31	718.041	225.475	15.02	225.475	11.07	0.9995	00:03
32	606.718	140.033	11.83	140.033	8.02	0.9997	00:04
33	540.264	90.059	9.49	90.059	5.64	0.9998	00:05
34	670.219	89.842	9.48	89.842	5.58	0.9998	00:05
35	518.222	99.042	9.95	99.042	5.90	0.9998	00:04
36	667.512	144.818	12.03	144.818	7.77	0.9997	00:04
37	575.715	120.792	10.99	120.792	7.20	0.9997	00:05
38	606.067	64.216	8.01	64.216	3.75	0.9998	00:05
39	569.112	73.287	8.56	73.287	4.48	0.9998	00:05
40	527.358	48.506	6.96	48.506	2.76	0.9999	00:05
41	459.714	56.229	7.50	56.229	3.49	0.9999	00:04
42	451.556	93.122	9.65	93.122	6.81	0.9998	00:04
43	494.874	54.064	7.35	54.064	3.71	0.9999	00:04
44	449.734	70.747	8.41	70.747	4.84	0.9998	00:04
45	486.534	45.304	6.73	45.304	2.49	0.9999	00:04
46	429.109	57.048	7.55	57.048	3.84	0.9999	00:05
47	437.374	54.326	7.37	54.326	3.65	0.9999	00:04
48	482.992	46.684	6.83	46.684	2.49	0.9999	00:05
49	563.088	42.122	6.49	42.122	2.29	0.9999	00:04

E Correlation heatmaps for features

Correlation Heatmaps for Jets

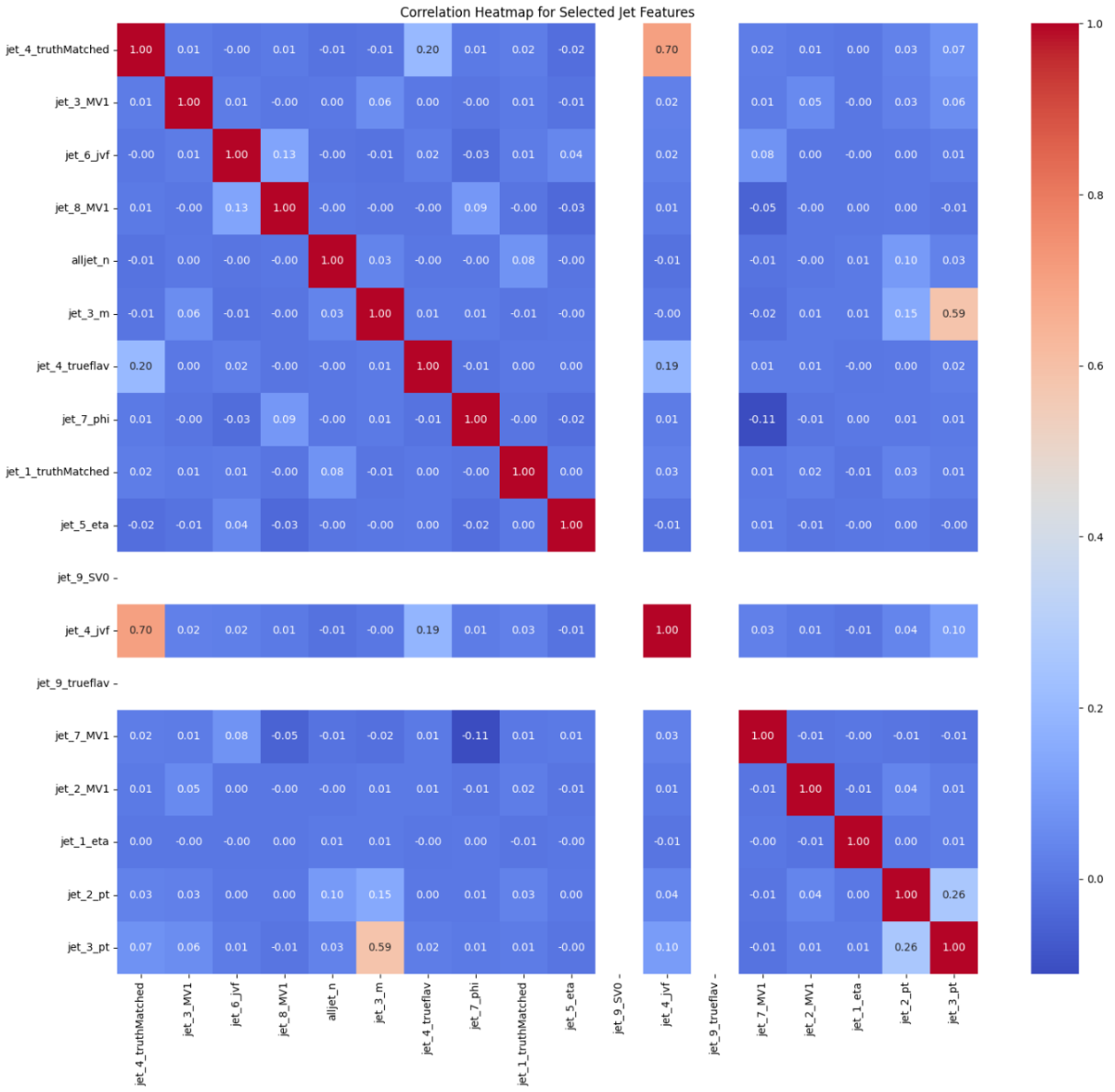


Figure 7.E.12: Heatmap illustrating the correlation between various jet features in the dataset. The white spaces are feeatures that offer no variability. These are removed before training.

Correlation Heatmaps for Leptons

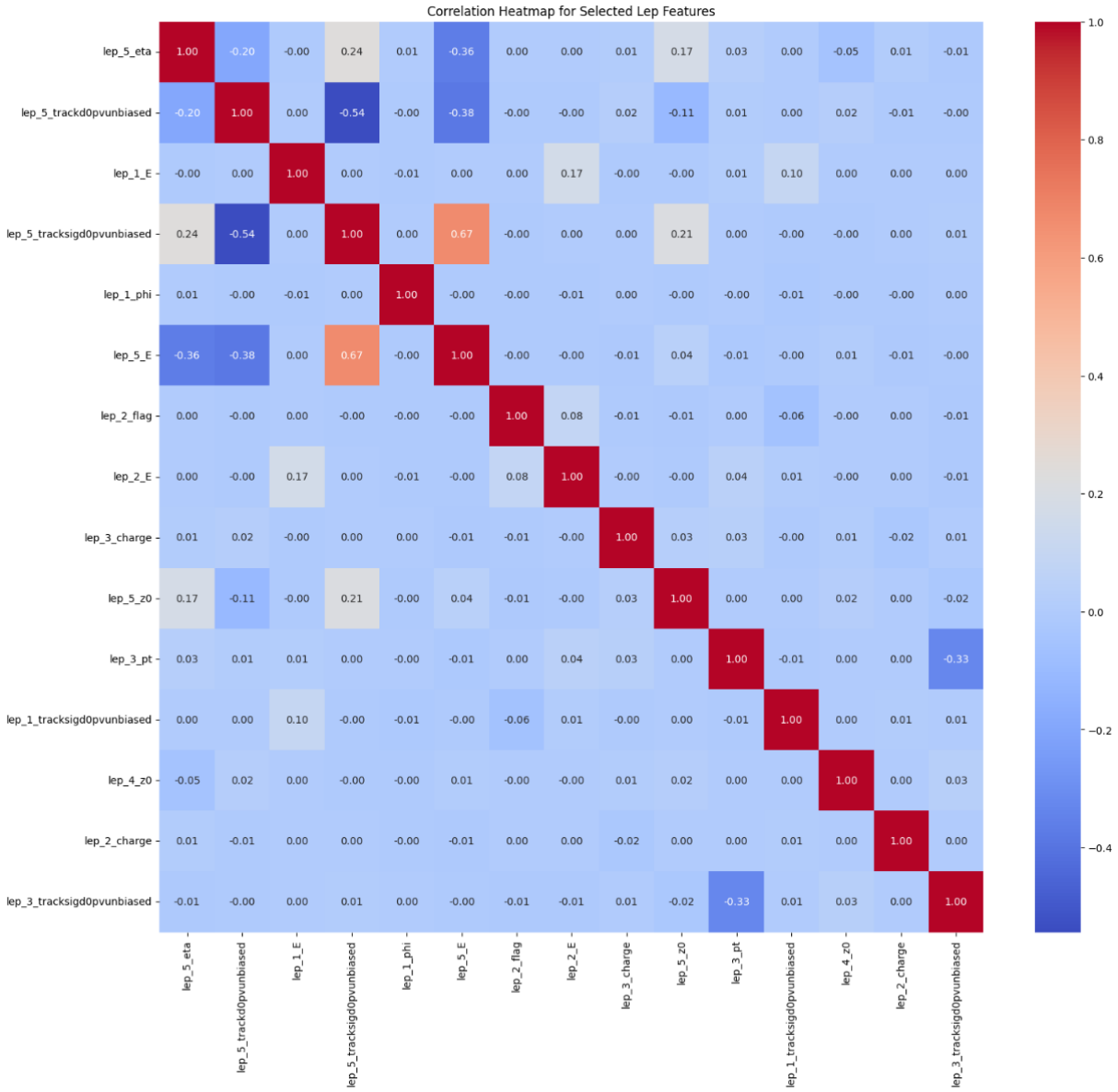


Figure 7.E.13: Heatmap illustrating the correlation between various lepton features in the dataset.

Correlation Heatmap for Specific Jet Features

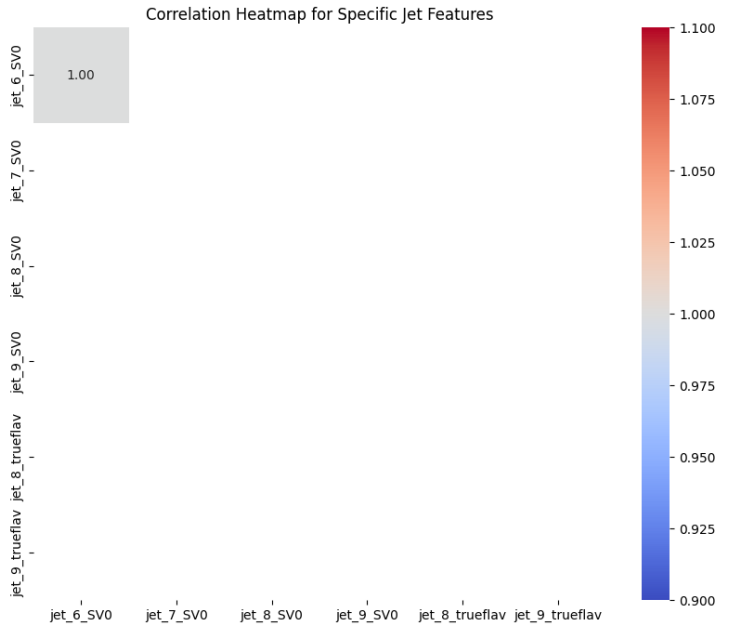
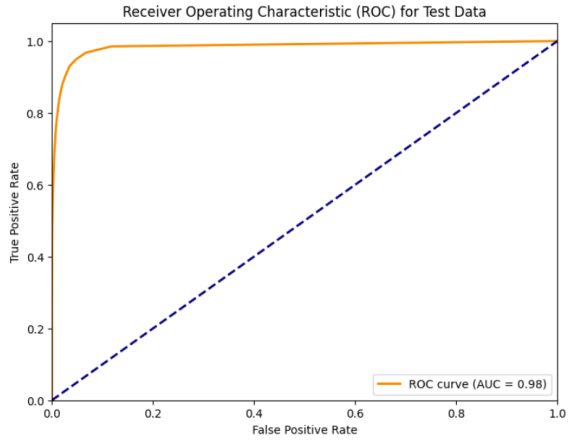
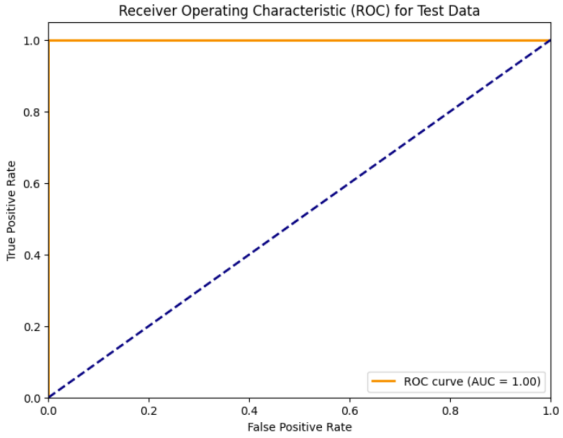


Figure 7.E.14: The heatmap is entirely white, indicating that all these features show no variability. The features involved are: jet_6_SV0, jet_7_SV0, jet_8_SV0, jet_9_SV0, jet_8_trueflaw, and jet_9_trueflaw. These are all excluded during feature selection.

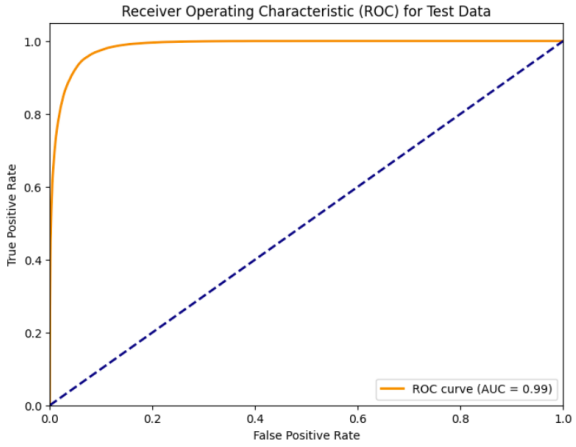
F ROC Curves



(i) ROC curve for the baseline random forest model.

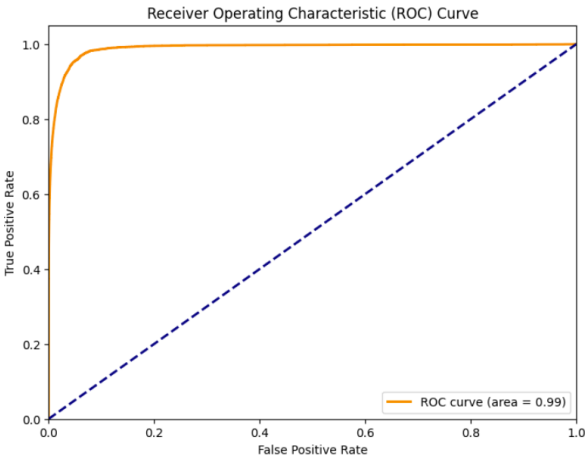


(ii) ROC curve for the oversampled random forest model.

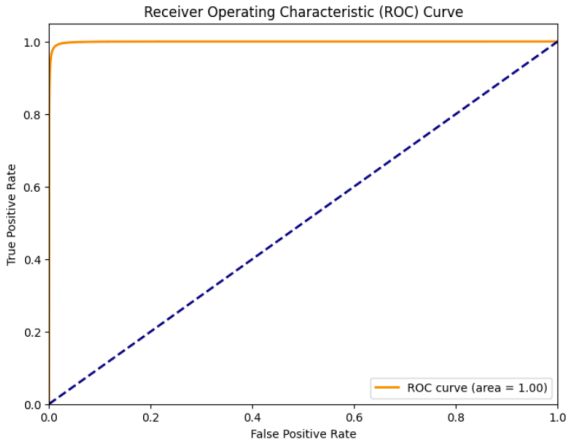


(iii) ROC curve for the undersampled random forest model.

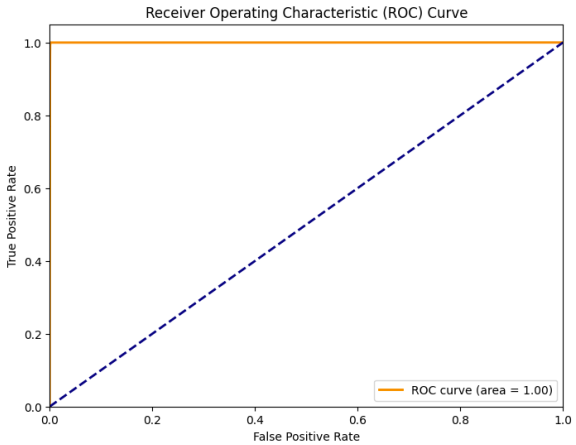
Figure 7.F.15: ROC curves for various random forest models.



(i) ROC curve for the baseline neural network model.



(ii) ROC curve for the undersampled neural network model.



(iii) ROC curve for the oversampled neural network model.

Figure 7.F.16: ROC curves for various neural network models.

G Prosjekthåndbok

Prosjekthåndbok is uploaded as a single pdf file together with this document.