Multiclass Classification Of Leptons In Proton-Proton Collisions At $\sqrt{s}$=13 TeV Using Machine Learning

Kristoffer Langstad

Thesis submitted for the degree of
Master in Computational Science: Physics
60 credits

Department of Physics
Faculty of mathematics and natural sciences

UNIVERSITY OF OSLO

Spring 2021

Multiclass Classification Of Leptons In Proton-Proton Collisions At $\sqrt{s}$=13 TeV Using Machine Learning

Kristoffer Langstad

Multiclass Classification Of Leptons In Proton-Proton Collisions At √s=13 TeV Using Machine Learning

http://www.duo.uio.no/

Printed: Reprosentralen, University of Oslo

Abstract

The Standard Model (SM) of particle physics has been used to explain many observed phenomena in the nature with great precision. But not everything that is observed has been explained by the SM, like the non-zero mass of the neutrino. A mechanism that tries to explain the mass of the neutrino is the Inverse Seesaw mechanism (ISS) yielding heavy neutrino masses and the existence of right-handed neutrinos. This can lead to heavy pseudo-Dirac neutrinos with trilepton final states and a neutrino from the decay of a W-boson. This thesis uses two types of such neutrino signals with neutrino masses, N_1, of 150 GeV and 450 GeV with data from proton-proton collisions collected by the ATLAS detector at $\sqrt{s} = 13$ TeV.

In this thesis we use the two simulated signal samples containing particles properties with Machine Learning to train classification models on these signals. This is a supervised learning case where we use multiclass classification to classify the vertex permutations of the leptons. The leptons in each event in the original Ntuples are ordered after p_T where lepton 1 is the lepton with highest p_T. With the multiclass classification we want to find from which vertex these leptons really originate from in each event. We train several different classification models to find the best performing model to use on unseen data.

For these simulated signals we find that the Light Gradient Boosting Machine (LGBM) is the fastest and best for classifying both simulated signals with accuracy scores of 0.88 for the 150 GeV signal and 0.96 for the 450 GeV signal when evaluating with the test set.

With the LGBM model we classify simulated background and signal data containing proton-proton collision events to find the particle vertex permutations for the three leptons in the final state. The outcome is to classify the vertex permutations such that we can study the charge and flavor of the leptons in the production and decay of the heavy neutrino. We study and compare features for these backgrounds and signals with different signal regions, and compare with a more standard analysis as benchmark. If an excess is observed some time in the future, we would like to study which neutrino mass model we are dealing with.

For the two simulated signals we only get predictions for the 123 and 132 vertices of the leptons, while the simulated backgrounds have predicted vertex permutations for 123, 132 and 213. The 213 vertex is predicted much less than the other two vertex permutations. The 213 vertex er predicted a maximum 10871 number of times, while 123 and 132 are predicted between 22337 and 5821865.

In the signal regions we find that the same flavor and opposite sign state of leptons from vertex 1 and 2 for electrons and muons are more dominant with much more events than the different flavor and opposite sign cases for the backgrounds. We get different degrees of lepton flavor violation for the different vertex permutations. For the signals there is not that much difference in the flavor ratios.

For the invariant mass of the three lepton system (m_{3l}) we find that it is easier to differentiate between background and the 450 GeV neutrino signal for masses higher than

400-500 GeV. The significance of the 450 GeV for m_{3l} reach maximum above 4 σ for all signal regions after 250 GeV, except for the 213 vertex permutations with no signal events. For the missing transverse energy (MET) backgrounds and signals have similar number of events and are more difficult to differentiate. The significance was found to be higher with the 450 GeV signals compared to the 150 GeV signal, and higher for the m_{3l} compared with the MET. The significance of the signals are much less in the MET distributions compared with the m_{3l} distributions. As expected the MET does not discriminate well the signals and backgrounds and we do not expect any excess in the signal distributions for MET.

The multiclass classification of the lepton vertex permutations with the LGBM model have successfully predicted the lepton vertices, yielding better performance than the current simple benchmark analysis in general.

Searching for new physics with the LHC is demanding and it is not always clear how to get the results we want. This is where ML can be of great assistance to uncover new physics by e.g. implementing multiclass classification to classify lepton vertices like we have done in this thesis. This study has been a great help to understand how ML techniques can be used to analyze and discover new physics, especially in particle physics.

Acknowledgements

I want to thank my supervisor Heidi Sandaker for letting me pretend to be a particle physicist during the work of this Computational Science thesis. I also want to give a big thanks to my co-supervisor Eirik Gramstad for the help with my thesis, answering my many questions and providing me with the data I use. I also want to thank the HEP group at the University of Oslo for taking me in and letting me use their computing hardware and be a part of their group.

My family and friends also need some appreciation during the work with this thesis, for the support they have shown me the past year and believing in me.

I want to give a special thanks to someone near and dear to me for having the patience to be with me the last few months, and always believing in me. I could not have done this without you.

Contents

List of Figures

List of Tables

Listings

Chapter 1

Introduction

The goal of this thesis is to use Machine Learning (ML) algorithms to classify the origin vertices of final state particles from proton-proton collisions. We will study two simulated neutrino signal scenarios with a heavy neutrino mass of 150 and 450 GeV, respectively. These signals go beyond the particle physics Standard Model (SM) with three final state leptons and a neutrino. We will compare the two signals with each other and compare particle features with some chosen cuts with a simpler analysis using more standard cuts as used in Pascoli et al. [1].

Particle physicists focuses a lot on colliding particles at high energies is to produce known and possibly unknown particles. When two particles collide, they will produce new particles that move through detectors built around the collision points. At the Large Hadron Collider (LHC), huge amount of data are produced each second which is captured by detectors and stored for further analysis. Many particles are produced in each such particle collision, and it is not always trivial to identify all these particles. There are also cases where some particles are not directly detected at all, like neutrinos. By using ML algorithms, which is a study in computer science and mathematics involving, among others, pattern recognition, we can try to computationally identify the vertices of the produced particles from the collision decays.

Particle physics takes a closer look at the building blocks of the universe, the fundamental particles in the SM. This is a theory that fits well with most observations. However there are several observations in the universe that cannot fully be explained by the SM, like dark matter and dark energy, meaning that the SM is incomplete and have to be extended. One of the methods to complete or expand the SM is to find new particles. This is done at large laboratories like CERN, where one of the things they do is to collide particles at high energies to produce new particles. After colliding particles, the new particles are detected using big detectors like A Toroidal LHC ApparatuS (ATLAS). One of the major discoveries at CERN is the discovery of the Higgs boson[2][3], which was the last missing piece of the SM. In this thesis, we will analyze both "*truth*" and more realistic simulations after taking into account hadronization, showers and detector inefficiency of particle collisions. With "*truth*" we mean simulated data or the particle collisions where we know exactly which particles have been produced and their origins.

1.1 Motivation for Thesis

The Compact Muon Solenoid (CMS) collaboration published in 2014 an article "Search for heavy neutrinos and W bosons with right-handed couplings in proton-proton collisions at $\sqrt{s}$=8 TeV"[4] using an integrated luminosity of 19.7 fb^{-1} of $\sqrt{s} = 8$ TeV p-p collision data produced by the CMS experiment at CERN. They searched for 2 leptons and 2 jet final states in signal regions with only same flavor (SF) and no lepton flavor violation (LFV). They observed a 2.8 local significance in the 2 electron and 2 jet (eejj) channel with no excess in the 2 muon and 2 jet (mmjj) channel. The ratio in the eejj channel had a SS/OS event ratio of 1/14. This is not consistent with left-right symmetry model (LRSM) theory.

When more data was included, this excess disappeared. Nevertheless, the results gave motivation for new neutrino mass mechanism theories to study this lack of SS and LFV, e.g. the Inverse seesaw (ISS) mechanism[1]. There has not been discovered any significant excess in the heavy neutrino searches at the LHC since.

This thesis looks at the same neutrino models as Pascoli et al. [1] to produce the final trilepton plus missing transverse energy (MET) states, seen in the Born diagram in Figure 1.1. By applying ML techniques on simulated data of this type of trilepton final state, we want to see if we can identify the final state lepton's production vertices origins in events seen in Fig. 1.1. If we can classify the vertex origins of the leptons, we can study the charge and flavor of the leptons in the production and decay of the heavy neutrino (N_m). Different neutrino mass models gives different expectation on the LFV and Majorana component of the N_m. This gives rise to different amount of SS/OS lepton pairs as well as the ratio between SF, for lepton flavor conservation (LFC), and DF (LFV) lepton pairs. If an excess is observed some time in the future, we would like to study which models of neutrino mass mechanism the excess is compatible with by efficiently identify and study the properties of the leptons involved in the neutrino production and decay.

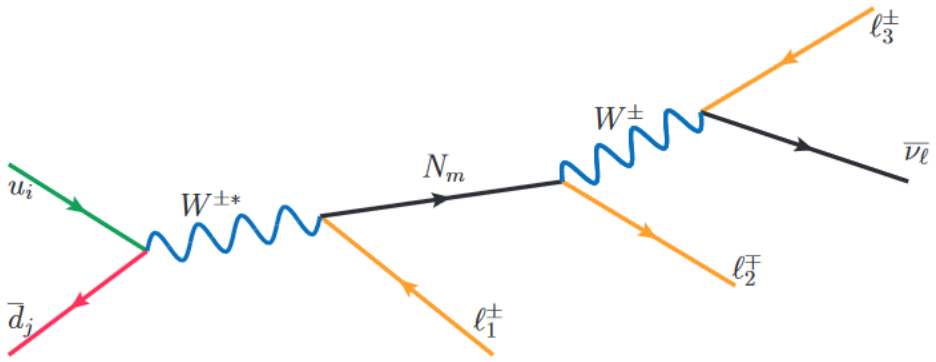


Figure 1.1: The Born diagram for the charged current Drell-Yan process of the proton-proton collision (on the left) producing a heavy pseudo-Dirac neutrino N in the inverse seesaw mechanism model, leading to a trilepton plus missing transverse energy (a light neutrino) final state. Figure is taken from ref. [1].

1.2 Motivation for Machine Learning

Machine learning and data science has had a huge growth since 2014. There are now a bigger variety of algorithms and approaches better suited for data analysis and different types of analyses depending on both the datasets and the desired goals. The datasets are as well a lot bigger than before, with samples ranging up to billions. This is both good and bad, since most machine learning models need a lot of training data to perform and predict on a good level. But with larger datasets, more time is needed to do the analyses since there are more data, obviously. This means that the models have to be fast and good.

Machine learning models have pattern recognition as one of the main focuses. The idea is to automatize and learn what normally is complex and difficult for humans to do. This can include image analysis or to learn the rules of a game. The more data the learning models have, the better they can do their tasks. Even though the algorithms can do quite complex tasks, the fundamental methods in these algorithms include normally simple methods. Many of the most used algorithms have several hyperparameters that are used to optimize the models. After a machine learning algorithm have been trained on some data, it can be exported and used on other types of similar data. This skips the step of training the algorithms with the data each time, and we can do straight to the predictions of the new data.

We will test several different classification model algorithms, and the best performing models for each signal sample will be chosen. If successful, then we can export the best models to other similar scenarios later.

1.3 Structure of Thesis

In the first chapters of part I, we take a look at an introduction to particle physics and further theories connected with the model we study. The next chapters involve the particle kinematics of particle collisions, and how they are collided and detected by instruments. Then follows theory of machine learning, the classification models and evaluation metrics to be used in this thesis.

Part II starts by looking at the most important libraries we use, the data and the features of the data prior to classification. Then we look at more detailed implementation and evaluation of the machine learning aspect leading to the best performing multiclass classification model. The ML analysis is chosen to be done in the programming language **Python**, which has many useful libraries for doing machine learning.

In part III we first present the results of the classification of simulated data with the trained classification models. Then follows the analysis of some chosen features of the data in defined signal regions, and a comparison with a more standard analysis.

Part IV consists of discussions of the results, concluding remarks of the thesis and a short look into future research based on this thesis.

Notation and Conventions

- $e = 1.6 \cdot 10^{-19}$ C : The elementary charge.
- $c = 2.998 \times 10^8$ m/s: Speed of light in vacuum.
- 1 GeV $= 10^9$ eV $= 10^9 \times 1.602 \times 10^{-19}$ J: Approximately the rest mass energy of the proton.
- $m_e = 9.109 \times 10^{-31}$ kg $= 0.511$ MeV/c^2: Mass of an electron.
- 1 barn (b) $\equiv 10^{-28}$ m^2: Interaction cross sections (dimension of area).
- $h = 6.626 \times 10^{-34}$ J·s: Planck's constant, a fundamental physical constant.
- $\hbar = \frac{h}{2\pi} = 1.055 \times 10^{-34}$ J·s: Unit of action in quantum mechanics (also called the reduced Planck constant).
- Einstein energy-momentum formula: $E^2 = p^2c^2 + m_o^2c^4$
- Coulomb force between two charged particles: $F = \frac{q_1 q_2}{4\pi\epsilon_0 r^2}$
- Natural units (from S.I. units):
  - Replace [kg, m, s] with [$\hbar$, c, GeV].
  - $\hbar c = 197$ MeV fm.
  - Use $\hbar = c = \epsilon_0 = \mu_0 = 1$.
- 1D time-dependent Schrödinger equation:

$$i\frac{\partial\psi(\mathbf{x},t)}{\partial t} = -\frac{1}{2m}\frac{\partial^2\psi(\mathbf{x},t)}{\partial x^2} + \hat{V}\psi(\mathbf{x},t)$$

- Planck scale $\sim 10^{19}$ GeV.
- GUT scale $\sim 10^{16}$ GeV.
- Magnetic fields are measured in Tesla (T).

Part I

Theory

Chapter 2

The Standard Model of Particle Physics

Throughout the years, there have been many theories in physics of what the universe is made up of and how everything fits together. For now, the best theory/model is the Standard Model (SM) of particle physics. This theory has many times through the years proven to successfully predict and explain particles and their interactions. This model has lead to the discovery of what we now call elementary particles and fundamental forces, and they are the building blocks of the universe.

In this chapter we look closer at the contents of the SM and the underlying theories and models. Much of the information in this chapter is based upon Thomson [5] and some on Elert [6].

2.1 Particle and Force Contents

The known elementary particles can be categorized into two main categories according to their spins; fermions and bosons. Fermions have half-integer spins, while bosons have integer spins. The Higgs boson is categorized as a boson but has 0 spin. In Figure 2.1 we see the categorization of the elementary particles, and the fundamental forces, in the SM. The individual categorizations will be explained in the upcoming sections. The interactions between the SM particles can be seen in Figure 2.2.

2.1.1 Gauge Bosons

From what we know of, there exists four fundamental forces. Three of these can be explained by the SM through exchange of (gauge) bosons. That is why bosons also are called force-carrier particles. The three forces are the electromagnetic, strong and weak nuclear forces, where each force has its own connected boson(s). There are five different bosons that mediates these forces, and they all have integer spins. This means that they go with vector fields, along a direction.

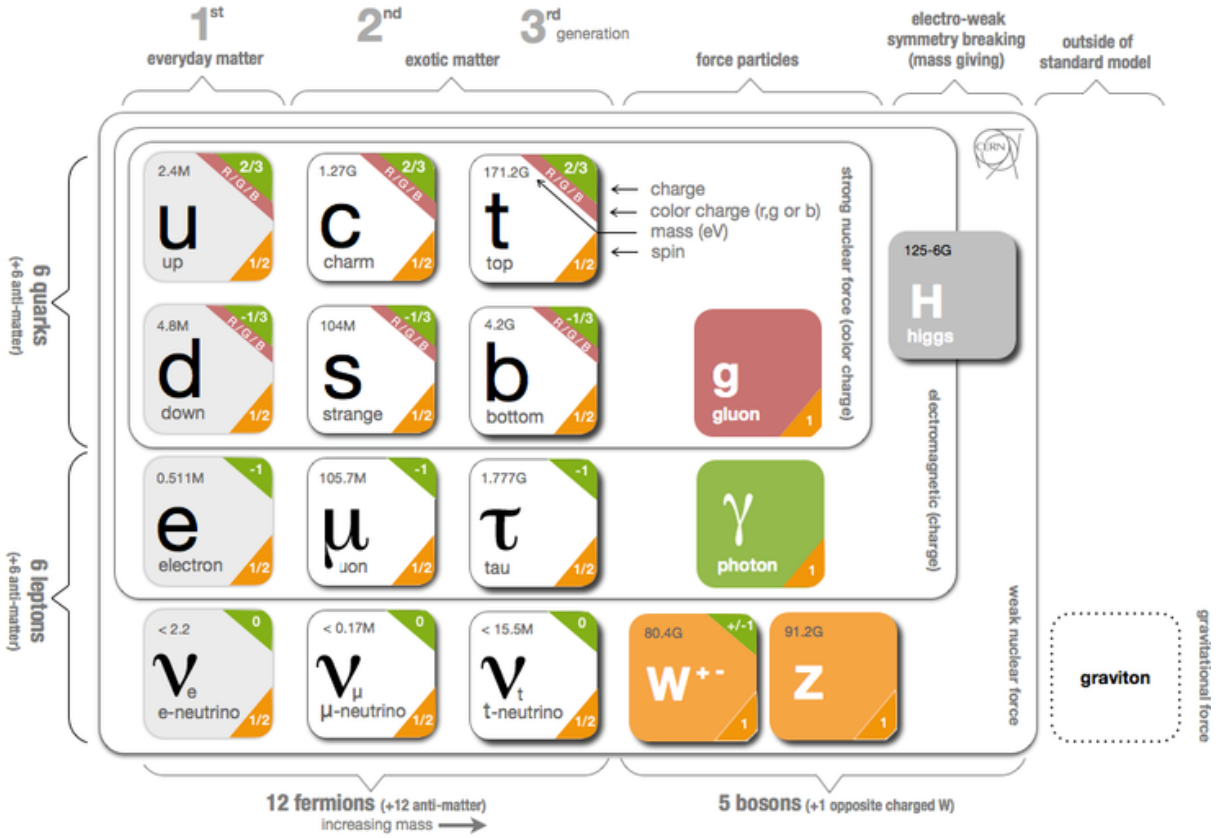


Figure 2.1: The Standard Model contents, source [7].

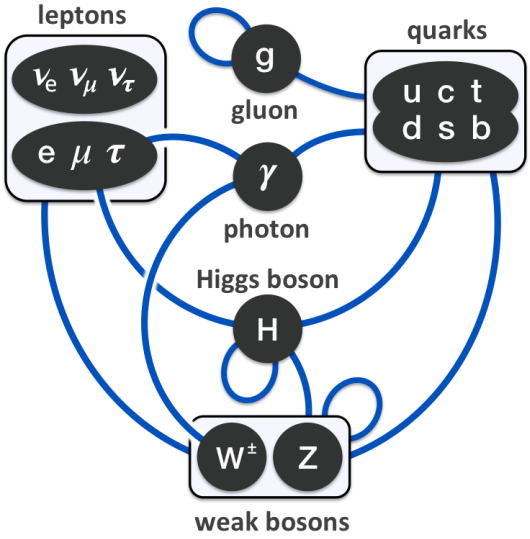


Figure 2.2: The interaction between the particles in the Standard Model. Credit: Wikipedia.

Strong nuclear force

The strong nuclear force is mediated by eight massless gluons (g). They only affect the (r,g,b) color charged quarks, and come in combinations of color and anti-color charges. Since the six gluons carry a different variation of color and anti-color combinations, they come in an octet of colored states. The color assignments of these eight physical gluon variations can be written as:

$$b\bar{g}, b\bar{r}, g\bar{b}, g\bar{r}, r\bar{b}, r\bar{g}, \frac{1}{\sqrt{2}}(r\bar{r} - g\bar{g}), \frac{1}{\sqrt{6}}(r\bar{r} + g\bar{g} - 2b\bar{b})$$

It is this strong interaction force that binds the quarks together to make e.g. protons and neutrons. The gluons can also self-interact with each other. This makes the interaction range of the strong nuclear force short, keeping the gluons within the nucleus. The exchange of gluons by interactions of colored particles is a mathematical model known as quantum chromodynamics (QCD, sect.2.4.2).

Electromagnetic force

The electromagnetic force is mediated by the massless photon (γ). Photons have interact with electrically charged particles. Since the photon is massless and electrically neutral, it has an infinite range. The electromagnetic force is responsible for holding electrons in place around the nucleus, and is not as strong as the strong nuclear force. Electrically charged particles are either attracted to each other or repelled away for each other, dependent on if the charges of the particles have the same sign or not. The exchange of photons by interactions of charged particles is a mathematical model known as quantum electrodynamics (QED, sect.2.4.2).

Weak nuclear force

The weak nuclear force is mediated by the $W^{\pm}$ and Z^0 bosons. There are two charged variants of the W with charge $+e$ or $-e$. The Z boson is electrically neutral. They er all massive which gives a short lifetime and short range. Because of the difference in charges, they act on different particles. The W boson couples the electromagnetic interactions. The W boson can decay to all flavors of quarks, except the top quark which is too massive, leptonic final states or hadronic final states. The weak interaction force can change the flavor of quarks. The exchange of W and Z bosons is explained with a more complex mathematical model that unifies both the weak and electromagnetic interactions, and it is known as electroweak theory (EWT, sect.2.4.2).

Gravitational force

The last force of nature is gravity. We have not yet found the hypothetical graviton (G) particle which should carry the gravitational force. All the other forces seem to be well explained in the SM, except for gravity. So the gravitational force is not included in the

SM. LIGO and Virgo discovered in 2015 gravitational waves from observing the merging of two black holes with ∼30 solar masses each [8], which might give insight into gravitons in the future. When looking at small objects (micro size), gravity does not seem to have any noticeable effect. But when we look at bigger objects of mass like humans or planets (macro size), then gravity has a much bigger effect and is well described by Einstein's General Theory of Relativity. Since gravity has more or less a negligible effect on particles energies so far probed in experiments, particle physicists do not have to take gravity into consideration.

2.1.2 Higgs Boson

The Higgs boson (H) is a "recently" discovered particle (2012) [2][3] theorized by Peter Higgs in 1964. This particle has intrinsic no spin, which makes it a scalar particle, and the only scalar particle discovered so far. It's electrically neutral and massive ($m_H \approx$125 GeV), and interacts with itself. Since it is so massive, the lifetime of the Higgs boson is very short and it's hard to detect directly. It can in principle decay to all massive SM particles. The heavier particle, the stronger is the coupling to the Higgs.

The discovery of the Higgs boson was a major contribution to the SM since it can explain the origin of the masses of the other elementary particles. It also confirmed the existence of the Higgs (scalar) field, which gives the other elementary particles mass when they interact with this field. This field is thought to be everywhere in the universe with a non-zero vacuum expectation value. Here, Higgs bosons appear and disappear and interact with other particles in the field giving them their masses. The gluons and photons do not interact with this field, hence they are massless.

2.1.3 Fermions

The fermion group in the SM consists of 12 elementary particles with half-integer spins. These particles are also known as matter-particles, since these particles are the building blocks of the matter in the universe. Each fermion has its own antiparticle. The antiparticles have the same mass as their particle partner, but has opposite electric charges and different quantum numbers. Fermions which acts as their own antiparticles are called Majorana particles.

The 12 fermions can be split into two groups of six quarks and six leptons. The fermions can then be categorized into three generations, which goes from lighter and more stable to heavier and less stable. As seen in the SM figure (2.1), the first generation is called the "everyday matter". This is because most of the stable (baryonic) matter is made from the first generation particles. The reason for this is that the first generation particles do not decay. Second and third generation particles are only observed in high-energy environments. None of the neutrinos decay, but they oscillate and scatter, and they rarely interact with baryonic matter.

Quarks

The six quarks are up, down, charm, strange, top and bottom. A characteristic property for the quarks is that they all have color and electric charges, and they interact through the strong nuclear force. The colors charges are denoted red, green and blue and they all have an anti-color. Quarks cannot exist as free particles. As explained in section 2.1.1, the quarks have a strong binding force between them since they are acted upon by the strong nuclear force. From this strong binding force, the quarks form particles called hadrons, like protons and neutrons. They are made up of either three quarks (baryons) or a quark and an anti-quark (mesons). A proton is made up of one down quark and two up quarks. The hadrons are color-neutral particles. Since quarks have electric charges, they also interact via the electromagnetic force and the weak nuclear force.

Leptons

The six leptons are electron (e), electron neutrino (ν_e), muon (μ), muon neutrino (ν_μ), tau (τ) and tau neutrino (ν_τ). The electron, muon and tau leptons have electric charges and are influenced by electromagnetism. They all carry a $-1e$ electric charge, while their respective antiparticle having electric charge $+1e$. Every lepton carrying a lepton number, which is conserved in all known interactions. The leptons also interact weakly. Both leptons and antileptons have their respective lepton number +1 and -1, and each flavor has its own lepton flavor number with the same values as the lepton numbers. There are three generations where the three charged leptons are paired with their respective neutrino, and the masses of these three leptons increases with the generation. Only the electron (1st gen) is stable and doesn't decay, while the muon and tau leptons decay via the weak interaction.

2.2 Neutrinos

The three neutrinos (electron, muon, tau) are a little more special than the other elementary particles. They are classified as leptons with half-integer spins, but they do not carry any charge and are thus neutral. They only interact via the weak nuclear force, making them very hard to observe since they go through almost everything without interacting much with anything. If the neutrinos are Majorana, they are the only Majorana fermions of the SM since all the other fermions have an non-zero electric charge. By detection of neutrinos and antineutrinos, only left-handed neutrinos and right-handed antineutrinos are observed. From the weak nuclear force mediator particles, the $W^\pm$ bosons, we know that they only couple to left-handed particles and right-handed antiparticles. This means that interaction of the right-handed neutrinos is not covered in the SM. Since mass terms couple both left- and right-handed states, the neutrinos are considered as massless in the SM. Through the discovery of neutrino oscillations [9], we know that the neutrinos can change flavor meaning they cannot be massless. We know that they have to have mass since the neutrinos oscillate, but the mechanism behind the masses are not known. So,

one type of neutrino can in fact change flavor to another type of neutrino when it travels over a large distance. Neutrino oscillation describes the difference between the neutrino flavor eigenstates and the neutrino mass eigenstates. This type of physics is not covered by the SM and will be looked more into later in this thesis.

2.2.1 Neutrino Oscillations

From the SM we know that for all interactions the lepton number is conserved for both the total and each lepton flavor separately. The lepton number is conserved when a $W^{\pm}$ boson decays into a lepton neutrino pair. We will in this thesis have a $W^{\pm}$ boson that decays into leptons

The discovery of neutrino oscillations was done by two experiments. Namely the Super-Kamiokande Observatory[10] and the Sudbury Neutrino Observatories (SNO)[11] experiments. They got the Nobel Prize in physics in 2015 for their contributions by detecting solar neutrinos from the Sun [12]. The Super-Kamiokande detected electron neutrinos using a big water Čerenkov detector, but they got a too low electron neutrino flux than what was expected to be produced in the Sun. The SNO experiment showed that the atmospheric neutrinos and the neutrino flux from β-decay in the Sun had strong muon and tau components by using heavy water. Since only electron neutrinos are produced by nuclear fusion in the Sun, the neutrinos must have the ability to change their flavor when moving over large distances.

The neutrino oscillation is a quantum-mechanical phenomenon, where the neutrino flavor (weak) eigenstates $(\nu_e, \nu_\mu, \nu_\tau)$ can be related to the mass eigenstates (ν_1, ν_2, ν_3) by an unitary transformation matrix U as

$$\begin{pmatrix} \nu_e \\ \nu_\mu \\ \nu_\tau \end{pmatrix} = \begin{pmatrix} U_{e1} & U_{e2} & U_{e3} \\ U_{\mu 1} & U_{\mu 2} & U_{\mu 3} \\ U_{\tau 1} & U_{\tau 2} & U_{\tau 3} \end{pmatrix} \begin{pmatrix} \nu_1 \\ \nu_2 \\ \nu_3 \end{pmatrix}. \tag{2.1}$$

The flavor eigenstates are linear combinations of the mass eigenstates. The 3×3 unitary matrix is the Pontecorvo-Maki-Nakagawa-Sakata (PMNS) matrix, and it's expressed with three mixing (rotation) angles and a complex Dirac CP violation phase if the neutrinos are Dirac particles. The unitary of the PMNS matrix implies that $U^{-1} = U^{\dagger} \equiv (U^*)^T$ and $UU^{\dagger} = I$.

If the neutrino mass eigenstates are not the same, we get neutrino oscillations from the phase differences in components of the wavefunction. Since we already know that the neutrinos change flavor from the discovery of neutrino oscillations, we know that the neutrinos need some mass, differing by flavor, to being able to change flavor. That is why the neutrinos need non-zero masses and not equal to each other for neutrino oscillations to be true. From experimental measurements, like long baseline accelerators, for the neutrino masses there is only found upper limits to the masses. The best upper limits on the neutrino masses was found to be

$$\sum_{i=1}^{3} m_{\nu_i} \lesssim 1.1\text{eV} \tag{2.2}$$

by the Karlsruhe Tritium Neutrino (KATRIN)[13] experiment in Germany. The reason why the neutrino masses seems to be so much smaller than the other fundamental particles is not known.

2.3 Symmetries

Particle dynamics are heavily influenced by symmetries and laws of conservation. From classical Newtonian physics, we know that energy (E), three-momentum ($\vec{p}$) and total angular momentum (J) are conserved quantities. This is also the case in the SM. A quantity that is not conserved is the (rest) mass (m). This is something we know according to Einstein's Special Relativity. This enables production of heavier particles than the colliding particles.

Another fundamental symmetry of physical laws is the CPT theorem. The CPT theorem is one of the results concluded by quantum field theory (QFT), and states that all physical processes are symmetric under CPT-transformation [14]. C is charge conjugation, where every particle can be replaced by its antiparticle. P is parity reflection, where everything in the universe is mirrored along the three physical axes. T is time reversal, where the direction of time is reversed in the sense of looking at the local properties of the SM. The combination of these three symmetries is predicted by the SM to be a symmetry, while each symmetry alone is only a near-symmetry. The CPT symmetry explains why particles and antiparticles have identical masses, magnetic moments, etc. The CPT is also thought to be an exact symmetry in the Universe. Only the weak interactions of quarks and leptons seems to violate the C-, P-, T- and CP-symmetries out of the three fundamental forces explained by the SM.

A topic to be further discussed later is gauge theory (sect. 2.4.2). From the connected gauge symmetry in the SM, we get a conservation of certain quantum numbers during the different interactions with the fundamental forces based on the $SU(3) \times SU(2) \times U(1)$ group. The quantities that are conserved are: the color charge for the strong nuclear interaction ($SU(3)$), the electric charge for electromagnetic interactions ($U(1)$) and the weak isospin for the weak nuclear interaction ($SU(2)$).

Other important conservation laws are the conservation of baryon number, B, and lepton number, L_x, in an interaction. x is the lepton flavor. The only case where the lepton number is not conserved is for neutrino oscillation. As we have explained earlier, neutrinos can change flavor when traveling large distances. But, this is not something we have to be concerned about in our case since we look at particles in particle detectors over short distance. This distance is not big enough for neutrino oscillations to occur.

2.4 Quantum Field Theory

The Standard Model is based on the framework of quantum field theory (QFT). This is a theory that combines quantum mechanics, special relativity and field theory. In other words, quantum field theory tries to explain the little things in the universe, like the

elementary particles, that move very fast, close to or with light speed c. This also means that every elementary particle has its own associated field. These fields can then be explained in terms of the Lagrangian density, $\mathcal{L}$, to explain the dynamics and kinematics of the fields.

The combination of quantum mechanics and special relativity does give some problems. The most important equation in quantum mechanics is the Schrödinger equation, and it's not Lorentz invariant. The problem with this is that Schrödinger's equation is not the same for two observers in different reference frames. Other problems this leads to is that we get violation of causality, negative energy states and there is no possibility for new particle creations. The good thing is that these problems can be fixed by exchanging the Schrödinger equation (see Notation and Conventions) by the Dirac equation [15][16] for $\frac{1}{2}$-spin particles and the Klein-Gordon equation [15][16] for scalar particles. With the Dirac and Klein-Gordon fields, this leads to specific (gauge) theories for different particles and associated interactions, which we have briefly mentioned earlier and will explain more soon.

2.4.1 The Lagrangian

For more simple classical mechanics cases the Lagrangian is just given as the difference between the kinetic energy, K, and the potential energy, V, $L = K - V$. This is also a baseline for the QFT. By using the Lagrangian of a system with a set of generalized coordinates q_i and their time derivatives $\dot{q}_i$, we can find the equation of motion that describes the system by using the Euler-Lagrange equation,

$$\frac{d}{dt}\left(\frac{\partial L}{\partial \dot{q}_i}\right) - \frac{\partial L}{\partial q_i} = 0. \tag{2.3}$$

A difference for QFT is that instead of kinetic and potential energies, or the generalized coordinates, we use fields with four space-time coordinates. This changes the Lagrangian L to the Lagrangian density $\mathcal{L}$ as a continuous system. This is a function of the fields, $\phi_i(t, x, y, z)$, and their derivatives, $\partial_\mu \phi_i(t, x, y, z)$. Since L is the spatial integral over $\mathcal{L}$,

$$L = \int \mathcal{L} d^3\mathbf{x}, \tag{2.4}$$

and using the principle of least action [17], the new Euler-Lagrange equation becomes

$$\partial_\mu \left(\frac{\partial \mathcal{L}}{\partial(\partial_\mu \phi_i)}\right) - \frac{\partial \mathcal{L}}{\partial \phi_i} = 0. \tag{2.5}$$

For simplicity we will just denote the Lagrangian density the Lagrangian from now on. From this new Euler-Lagrange equation, we can derive both the free-particle Dirac and the Klein-Gordon equations by imposing the Lagrangian with a free fermion field[1] and free

[1] Relativistic spin-half fields, Chapter 17.2.2 in Thomson [5]

theory[2] respectively. The Lagrangian for the spin-half (spinor) field, ψ, is

$$\mathcal{L}_D = i\bar{\psi}\gamma^\mu\partial_\mu\psi - m\bar{\psi}\psi, \tag{2.6}$$

and the Lagrangian for the non-interacting scalar field, ϕ, is

$$\mathcal{L}_S = \frac{1}{2}(\partial_\mu\phi)(\partial^\mu\phi) - \frac{1}{2}m^2\phi^2. \tag{2.7}$$

Both of these two equations for the Lagrangian contain a kinematic term and a mass term.

With perturbation theory in quantum mechanics, the Lagrangian can also be used to describe the behavior and interaction of elementary particles with Feynman diagrams for simpler visualization of usually complex particle interactions.

2.4.2 Qauge Theories

From the new Lagrangian we now know that we need some new theory to explain the interactions between the elementary particles, since these interactions vary depending on the particles and associated interactions involved. In this theory we need to require that the Lagrangian stays invariant under local transformations using symmetry or gauge groups. In special relativity, this global symmetry group is called the Poincaré group which includes spacetime symmetries.

To describe the SM we need an internal gauge invariant symmetry that represents the different elementary interactions and is independent of spacetime coordinates. This is the local $SU(3) \times SU(2) \times U(1)$ gauge symmetry group. Here each special unitary group with degree n (the number in the parenthesis) is connected to its own gauge theory and the three elementary interactions in the SM, and n is a n-dimensional space. If a symmetry group is commutative, meaning that regardless of what the order of the elements are applied the result will be the same, then it is called an Abelian group. If the group is non-commutative, it is then a non-Abelian gauge theory which implies the existence of gauge boson self-interaction.

Quantum chromodynamics (QCD)

The gauge theory that defines the strong interaction between the quarks and (eight) gluons (color charged particles) is the quantum chromodynamics sector [18]. The QCD conserves the separately conserved color charges red, green and blue, and thus works in a three dimensional color space. Another quantity which is conserved in QCD is parity. This comes from that the QCD interaction Hamiltonian is invariant under parity transformations (sect. 11.2.2 in Thomson [5]). The antiquarks carry the opposite color charge to the quarks of red, green and blue. The color states consists of color isospin and color hypercharge. It also ensures invariance under the local gauge transformation. The gauge symmetry group for this sector is $SU(3)_C$ and is represented by 3×3 matrices, where the C stands for

[2] Relativistic scalar fields, Chapter 17.2.2 in Thomson [5]

the conserved color. This symmetry group does not commute and is a non-Abelian gauge theory, or more precise it is a Yang-Mills gauge theory [19]. By using this gauge theory, we can derive a new invariant Lagrangian which does not have a mass term for the gluons:

$$\mathcal{L}_{QCD} = \bar{\psi}(i\gamma^{\mu}\partial_{\mu} - m)\psi - \frac{1}{2}g_s\bar{\psi}\gamma^{\mu}\lambda_a\psi G^a_{\mu} - \frac{1}{4}G^a_{\mu\nu}G^{\mu\nu}_a \tag{2.8}$$

ψ is a fermion (quark) field, g_s is a coupling constant of the strong interaction, γ^{μ} are Dirac matrices, $a = 1, ..., 8$ are the eight gluons, λ_a is one of the eight Gell-Mann matrices and $G^a_{\mu\nu}$ is a gauge invariant gluon field strength tensor. The last term of the Lagrangian in equation 2.8 implies that the gluons should be massless and can self-interact.

In Figure 2.3 we see the QCD vertices for quark and gluon interactions (and self-interacting gluons).

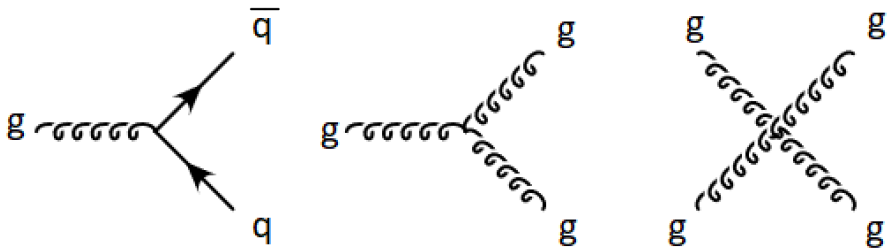


Figure 2.3: Here we see Feynman diagrams of the basic QCD vertices. From left to right we see, the coupling of gluon fields (g) interaction with quark fields (q), a triple gluon vertex and a quartic gluon vertex. Source Fig. 10.1 in Thomson [5].

Quantum electrodynamics (QED)

The gauge theory that defines the electromagnetic interaction for the electrically charged particles and photons is the quantum electrodynamics sector [20]. The QED conserves the electric charge of the particles. Like in QCD, parity is conserved in QED (sect. 11.2.2 in Thomson [5]). The gauge symmetry group for QED is $U(1)$ which is an Abelian group. By starting with a free fermion field for the Lagrangian (eq.2.6, invariant under *global* $U(1)$ transformation) and require invariance under a local phase transformation, leads to a Lagrangian with a Lorentz-invariant description where there is an electromagnetic interaction between fermions and the gauge field of the massless photon:

$$\mathcal{L}_{QED} = \bar{\psi}(i\gamma^{\mu}\partial_{\mu} - m_e)\psi + e\bar{\psi}\gamma^{\mu}\psi A_{\mu} - \frac{1}{4}F_{\mu\nu}F^{\mu\nu}. \tag{2.9}$$

ψ is the field of the spin half particles, e is a coupling constant of the electromagnetic interaction, γ^{μ} are Dirac matrices, A_{μ} is a covariant four-potential (gauge field), and $F_{\mu\nu}$ is the electromagnetic field strength tensor.

From the Lagrangian in equation 2.9, we can construct the Feynman diagram of a QED interaction vertex between a single photon and two spin-half fermions, seen in Figure 2.4.

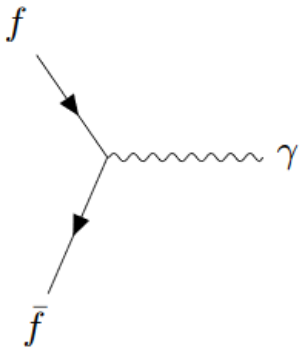


Figure 2.4: A Feynman diagram of the basic QED vertex for the interaction between fermions (f) and a massless photon (γ). Source Fig. 5.6 (and 10.10a) in Thomson [5].

Electroweak theory (EWT)

The gauge theory which defines the weak interaction for the 3rd component of isospin particles and the W and Z bosons/fields is the unified theory known as electroweak theory (EWT) [21] or Glashow-Weinberg-Salam (GWS) theory. This theory (from the 1960's) earned the three contributors Glashow[22], Weinberg[23] and Salam[24] the Nobel Prize in Physics in 1979 [25][26].

Unlike QCD and QED, it is found experimentally that parity is not conserved in the weak interaction (sect. 11.2.3 in Thomson [5]). This parity-violation makes the weak interaction treat left-handed and right-handed particles differently. The charge-current weak interaction is invariant under $SU(2)$ local phase transformations and includes weak isospin. The cross-section of W-pairs produced at higher energies, violates quantum mechanical unitarity such that particle probability is no longer conserved. This is solved because the couplings of the γ (QED), $W^{\pm}$ and Z EWT are related to each other in the unified electroweak model.

In the EWT theory fermions exists as left- and right-handed chirality states, while W-bosons only couple to left-handed fermions. The EWT conserves the flavor charge and weak isospin of the particles. It is the weak isospin quantum number that accounts for the W-boson coupling, since left-handed fermions have half-isospin and appears as isospin doublets while right-handed fermions appear as isospin singlets. Something to take notice of here is that, the weakly interacting quarks are superpositions of the mass eigenstates while the strongly interacting quarks are mass eigenstates.

The electroweak theory is based on the $SU(2)_L \times U(1)_Y$ symmetry group, where L is left-handed interaction and Y is the weak hypercharge expressed by the electric charge Q and the third component of the weak isospin I_3, $Y = 2(Q - I_3)$. This new $U(1)_Y$ local gauge symmetry is used instead of that in QED, where the charge now has been replaced by the weak hypercharge. Each gauge invariant transformation in this theory, introduce new gauge fields which as linear combinations corresponds to the photon and the W and Z bosons of the weak interaction. With these new gauge fields, we can derive yet another

new preliminary (electroweak) Lagrangian that is associated with the EWT theory:

$$\begin{aligned}\mathcal{L}_{EWT} =&\bar{\psi}_L\gamma^\mu\left[i\partial_\mu - \frac{1}{2}g\boldsymbol{\sigma}\mathbf{W}_\mu - \frac{1}{2}g'YB_\mu\right]\psi_L + \bar{\psi}_R\gamma^\mu\left[i\partial_\mu - \frac{1}{2}g'YB_\mu\right]\psi_R \\ &- \frac{1}{4}B_{\mu\nu}B^{\mu\nu} - \frac{1}{4}\mathbf{W}_{\mu\nu}\mathbf{W}^{\mu\nu}\end{aligned} \tag{2.10}$$

$\psi_{L,R}$ are the fields for left- and right-handed fields respectively, g and g' are coupling constants related to the elementary charge, γ^μ are the Dirac matrices, $\boldsymbol{\sigma}$ are the Pauli matrices, B_μ is a field strength tensor for the weak hypercharge gauge field for $U(1)_Y$, $\mathbf{W}_{\mu\nu}$ is a field strength tensor for the three weak isospin gauge fields for $SU(2)_L$.

The EWT gauge symmetry group is non-Abelian. In Figure 2.5 and 2.6 we see Feynman diagrams of the electroweak interaction vertices including fermions and gauge boson self-interactions. The photon and the Z-boson couple with both left- and right-handed fermions, while the W-bosons do not.

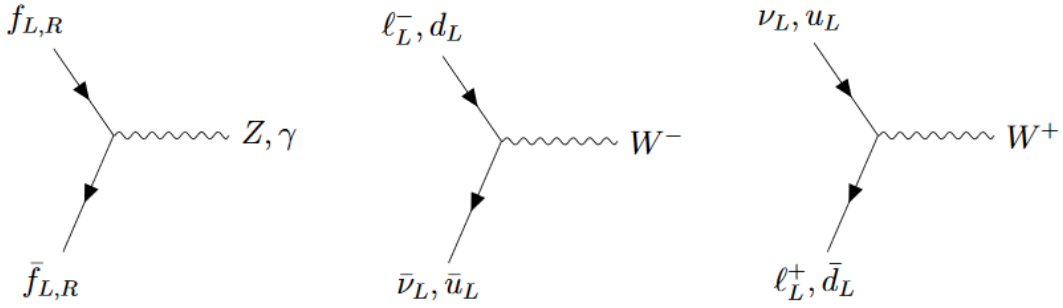


Figure 2.5: Here we see Feynman diagrams of the electroweak interaction vertices that includes fermions.

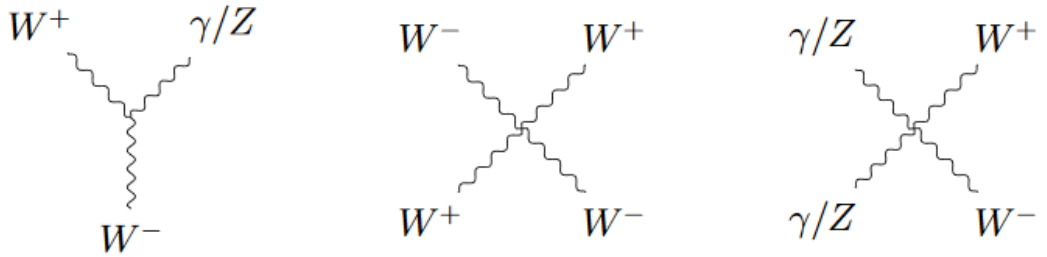


Figure 2.6: Here we see Feynman diagrams of the electroweak interaction vertices for gauge boson self-interaction.

By introducing the BEH mechanism we get, in addition to the coupling in Figure 2.5 and 2.6, couplings between the Higgs boson and the massive gauge boson as well as Higgs self-interaction. These couplings can be seen in Figure 2.7.

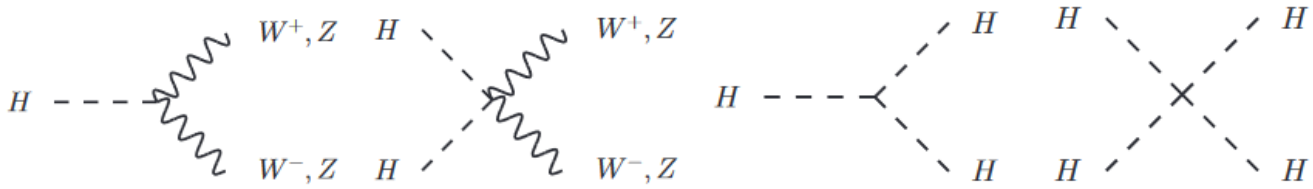


Figure 2.7: Here we see Feynman diagrams of the couplings between the Higgs boson and the massive gauge bosons and Higgs self-interaction.

Fermion masses: The Higgs mechanism can also be used to give masses to the fermions. The Higgs isospin doublet has a lower and an upper element. The lower element is used to give masses to down-type quarks and charged leptons, while the masses of the up-type quarks are constructed from the conjugate doublet. The gauge invariant mass terms of the Dirac fermions are then described as

$$m_f = \frac{g_f v}{\sqrt{2}}, \tag{2.11}$$

where g_f is the Yukawa coupling constant of the fermions to the Higgs field, as shown in Figure 2.8.

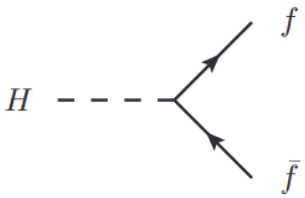


Figure 2.8: Here we see Feynman diagrams of the coupling between the Higgs boson and fermions.

Full EWT Lagrangian

The complete Lagrangian for the EWT is given by:

$$\begin{aligned}\mathcal{L}_{EWT} =& \overline{\psi}_L \gamma^\mu \left[i\partial_\mu - \frac{1}{2} g \boldsymbol{\sigma} \mathbf{W}_\mu - \frac{1}{2} g' Y B_\mu \right] \psi_L + \overline{\psi}_R \gamma^\mu \left[i\partial_\mu - \frac{1}{2} g' Y B_\mu \right] \psi_R \\ & - \frac{1}{4} B_{\mu\nu} B^{\mu\nu} - \frac{1}{4} \mathbf{W}_{\mu\nu} \mathbf{W}^{\mu\nu} + \left| \left(i\partial_\mu - \frac{1}{2} g \boldsymbol{\sigma} \mathbf{W}_\mu - \frac{1}{2} g' Y B_\mu \right) \phi \right|^2 \\ & - V(\phi) - (g_f \overline{\psi}_L \phi \psi_R + G'_f \overline{\psi}_L \phi_c \psi_R + h.c.)\end{aligned} \tag{2.12}$$

The first line is the couplings between the fermions and the gauge fields and kinetic terms for the fermion fields. The second line is the kinetic terms for the gauge fields and the Higgs

field, the couplings between the gauge field and the Higgs field, and the couplings between the gauge fields. The third line contains the scalar potential, the Yukawa coupling terms and the fermion mass terms, and $h.c$ stands for the corresponding Hermitian conjugate.

Chapter 3

Neutrinos Beyond the Standard Model

The SM explains most of the physics we measure in experiments. The SM has several free parameters which are chosen to match observations. Nevertheless, the SM does not explain everything. For theorists the ultimate goal is to construct a Theory of everything, which explain all the physical phenomena in a unified way (including also gravity). Particle physicists try to address the shortcomings of the SM by extending it and construct more complete models which can explain e.g. gravity or the masses of the neutrinos.

A major problem in today's particle physics is that the SM can only explain about 5% of the total energy density of the Universe. This 5% of the matter in the Universe is called baryonic matter, while the rest is something yet unknown. One theory is that about 25% is something called dark matter, that acts as matter, but we can't see it, and has a gravitational pull in the Universe. The remaining 70% is then thought to be dark energy, which has a pushing affect on the galaxies in the Universe making it expanding faster and faster with time. A dark matter candidate is neutrinos. We will not go into the dark matter aspect, but look closer at neutrinos and the neutrino masses.

From the discovery of neutrino oscillations, we know from observations and experiments, that the neutrinos need to have mass since they have the ability to change flavor over very large distances. Why the neutrinos have mass and what gives them mass, on the other hand, are not explained in the SM. The only place in the SM that allows CP-violation, is in the weak interaction domain where left-handed neutrinos are affected through neutrino mixing. Since it is not observed right-handed neutrinos nor left-handed antineutrinos, C- and P-symmetry should be violated. It has not yet been observed if CP-violations occur in neutrino oscillations, since neutrinos seem to uphold the CP-symmetry with the existence of right-handed antineutrinos. This means that some new physics is required to explain this breaking of CP-violation.

For the neutrinos to acquire mass, we have to go beyond the SM neutrino knowledge, and introduce some new theories. We will look more into the neutrino masses and the model for this thesis in this chapter.

3.1 Neutrino Masses

According to the SM, neutrinos do not have mass because only left-handed (LH) neutrinos are covered by the SM and thus, right-handed (RH) neutrinos are not involved in any of the fundamental interactions and have not yet been observed. As mentioned earlier, we know from observations and experiments of neutrino oscillations that neutrinos have a tiny, but non-zero mass. to being able to change flavor when moving over large distances. The neutrino masses are something we need to look more into.

3.1.1 Dirac Neutrinos

Dirac particles are particles which can be distinctively separated from its antiparticle. The Dirac field is described by a four-component Dirac spinor ψ and can be divided into a left-handed ψ_L and a right-handed ψ_R part as two component Weyl spinors:

$$\psi = \begin{pmatrix} \psi_L \\ \psi_R \end{pmatrix}. \tag{3.1}$$

The left-handed neutrinos in the SM are described by this left-handed Weyl field. Since the Dirac mass term require both left- and right-handed fields in the SM, there is no Dirac mass term for the neutrinos.

If we assume neutrinos as Dirac particles, the neutrino mass is added similarly to the up-type quarks as the conjugate Higgs doublet. The gauge invariant Dirac neutrino mass term after spontaneous symmetry breaking becomes

$$\mathcal{L}_D = -m_\nu(\overline{\nu_R}\nu_L + \overline{\nu_L}\nu_R), \tag{3.2}$$

with the neutrino mass still determined by the Yukawa coupling constant as for Dirac fermions (eq. 2.11):

$$m_\nu = \frac{g_\nu v}{\sqrt{2}} \tag{3.3}$$

The neutrino masses have been found to be several orders of magnitude smaller than the charged lepton masses. This leads to a Yukawa coupling constant $g_\nu \leq 10^{-12}$ for neutrino masses that are less than 1.1 eV (sect. 2.2.1). There are no reasons why the Yukawa constants should be so small, which gives reason to believe that there must be some other mechanism giving neutrinos their masses. The right-handed neutrino in the SM would be sterile and only interact with the Higgs boson.

3.1.2 Majorana Neutrinos

Another option for the neutrinos, is that they can be Majorana neutrinos. This means that they can be their own antiparticles. The result of this would mean that the lepton number no longer is conserved, which it is in the SM. To not break the gauge invariance of

the SM when adding the fields for RH neutrinos and LH antineutrinos in the Lagrangian, the LH antineutrinos appear as the CP conjugate field of the RH neutrino[5] defined by

$$\psi_L^c = \hat{C}\hat{P}\psi = C\overline{\psi}_R^T, \tag{3.4}$$

where C is the charge conjugation matrix.

For a Majorana neutrino we have $\psi^c = \psi$, which means that the neutrino field can be expressed with a Majorana spinor

$$\psi_\nu = \begin{pmatrix} \overline{\nu_R^c} \\ \nu_R \end{pmatrix} \tag{3.5}$$

for LH and RH neutrino fields and the CP conjugate of the RH field (or the LH antineutrino) $\overline{\nu_R^c}$. The local gauge invariant Majorana neutrino mass term, with Majorana mass M, becomes

$$\mathcal{L}_M = -\frac{1}{2}M(\overline{\nu_R^c}\nu_R + \overline{\nu_R}\nu_R^c). \tag{3.6}$$

This means that the Majorana mass term is not constrained by gauge symmetry and can be arbitrary large. The global baryon number minus the lepton number $(B-L)$ symmetry of the SM would be broken if the neutrino is a Majorana neutrino. From observations of the asymmetry between matter and antimatter in the Universe, it actually looks like the baryon number is not conserved.

A generic Majorana mass matrix, $\mathcal{M}$, with three neutrinos can also be expressed as

$$\mathcal{M} = \begin{pmatrix} M_L & m_D \\ m_D^T & M_R \end{pmatrix} \tag{3.7}$$

m_D is the mass for a Dirac neutrino, M_L is the Majorana mass for a LH neutrino (ν_L) and M_R is the Majorana mass for a RH neutrino (ν_R).

3.1.3 Pseudo-Dirac Neutrinos

A pseudo-Dirac neutrino[27][28] mass matrix is similar to the Majorana mass matrix in equation 3.7, except that the M_L and M_R masses are the lepton number violating Majorana masses of light neutrinos[1]. When the Dirac mass is $m_D >> M_L, M_R$, we get a pseudo-Dirac mass matrix where the eigenvalues of the resulting mass eigenstates are close to each other. This means that the two light neutrinos can form a Dirac-like/pseudo-Dirac neutrino.

3.1.4 The Seesaw Mechanism

One of many theories for the light masses of the neutrinos is to add RH neutrinos that couple to the LH neutrinos. However, this would lead to a disparity problem regarding mass scale. To solve this, a seesaw mechanism is introduced where the observed (light Dirac) LH neutrinos couple with very heavy (sterile) Majorana RH neutrinos. This would

[1] A RH neutrino is also called a sterile neutrino, ν_s.

explain the small masses of the observed SM left-handed neutrinos and the absence of observation of RH neutrinos. The problem is that the mass scale of the RH neutrinos is unknown, since the masses of the Dirac neutrinos are still uncertain. So they could be somewhere between a few keV, and possibly be light dark matter particle candidates, or have higher masses near the unification energy (GUT scale), where the electromagnetic, weak and strong forces have equal strength.

Type-I seesaw mechanism

There are several varieties of the seesaw mechanism which extends the SM, but the simplest one is the **Type-I seesaw mechanism**[29]. This involves the mix of LH Dirac neutrinos and RH Majorana neutrinos. In this theory, a right-handed neutrino is added for each of the SM LH neutrinos, in total three. When involving neutrinos as Majorana, we get that $\overline{\nu_L}\nu_R$ is equivalent to $\overline{\nu_R^c}\nu_L^c$. The Lagrangian after the spontaneous electroweak symmetry breaking with both the Dirac and Majorana mass terms becomes:

$$\mathcal{L}_{DM} = -\frac{1}{2}\left(m_D\overline{\nu_L}\nu_R + m_D\overline{\nu_R^c}\nu_L^c + M\overline{\nu_R^c}\nu_R\right) + h.c. \tag{3.8}$$

m_D is the Dirac mass and M is the Majorana mass. This seesaw mechanism is characterized by $M_L << m_D << M_{(R)}$. This equation can also be written in terms of a 2×2 mass matrix ($\mathcal{M}$) for the neutrinos:

$$\mathcal{L}_{DM} = -\frac{1}{2}(\overline{\nu_L}\,\overline{\nu_R^c})\begin{pmatrix} 0 & m_D \\ m_D & M \end{pmatrix}\begin{pmatrix} \nu_L^c \\ \nu_R \end{pmatrix} + h.c. \tag{3.9}$$

By looking at the eigenvalues (λ) of the mass matrix $\mathcal{M}$ we get the physical masses of the neutrinos (in this model) as (sect.17.8.1 in Thomson [5])

$$m_\pm = \lambda_\pm = \frac{M \pm M\sqrt{1 + 4m_D^2/M^2}}{2}. \tag{3.10}$$

If we assume the Majorana mass much larger than the Dirac mass, $M >> m_D$, we get a light LH neutrino state (ν) and a heavy RH neutrino state (N) with masses

$$|m_\nu| \approx \frac{m_D^2}{M} \quad \& \quad m_N \approx M. \tag{3.11}$$

The physical neutrino states are in this case

$$\nu \approx (\nu_L + \nu_R) - \frac{m_D}{M}(\nu_R + \nu_R^c) \quad \& \quad N \approx (\nu_R + \nu_R^c) + \frac{m_D}{M}(\nu_L + \nu_L^c). \tag{3.12}$$

By looking at equation 3.11, we see that the lightness of the SM neutrinos are explained by the existence of much heavier right-handed neutrinos.

Inverse seesaw mechanism

The model we will be studying in the following section involves a slightly different seesaw (ISS) theory, namely the so-called **Inverse seesaw mechanism** [1][30]. This is a low-scale Type-I neutrino mass model and yields heavy neutrino masses and allows large Yukawa couplings. While the ordinary (Type-I) seesaw predict very heavy RH neutrinos ($\sim 10^{14}$ GeV), from the ISS predicts TeV-scale RH neutrinos. Masses of 10^{14} GeV is out of range for experiments, which is not so attractive.

Besides the addition of three right-handed neutrinos, this model also adds three LH singlet fermions as well as three light LH neutrinos. These three added particle "groups" make a 3×3 matrices for each group. The ISS Lagrangian is a 9×9 matrix given as:

$$\mathcal{L}_{\text{ISS}} = -\nu_L m_D N_R - S_L M N_R - \frac{1}{2}\bar{S}_L \mu S_L^c + h.c. \tag{3.13}$$

ν_L is the (SM) LH neutrino, N_R is the RH neutrino, S_L is a new light singlet neutrino and μ is a lepton violating parameter ($\mu << m_D, M$). The light neutrino mass matrix can be written as a 3×3 matrix:

$$m_\nu = m_D^T (M^T)^{-1} \mu M^{-1} m_D.^{2} \tag{3.14}$$

These nine neutrinos form three heavy pseudo-Dirac neutrino pairs with small lepton number violations in the singlet mass terms. This comes from the decay of a $W_R^\pm$ to a pseudo-Dirac neutrino, since a neutrino coupled to a $W_R^\pm$ is a pseudo-Dirac fermion. It is during this process that the lepton number is approximately conserved, and accounts for missing same-sign electron events.

Our base model is the $SU(2)_L \times SU(2)_R \times U(1)_{B-L}$ left-right symmetry group which involves the ISS mechanism, and is based on the

$$SU(3)_C \times SU(2)_L \times SU(2)_R \times U(1)_{B-L} \tag{3.15}$$

gauge symmetry. The main difference from the Type-I seesaw mechanism is that instead of a heavy Majorana mass eigenstate neutrino, we have a heavy pseudo-Dirac neutrino mass eigenstate. The mass difference (mixing) between the left- and right-handed neutrinos probe small neutrino masses. This leads to Left-Right symmetric models with the same final state as for a heavy Majorana neutrino.

3.2 The Charge Current Drell-Yan Process

The model in this thesis is based on the works of Pascoli et al. [1] with the inverse seesaw mechanism. Here, two protons are accelerated and collided to produce a heavy pseudo-Dirac neutrino, and a left-right symmetric model. Since the inverse seesaw mechanism allows a large left-right neutrino mixing, while keeping the neutrino masses tiny, the W boson may decay into a charged lepton l and a heavy pseudo-Dirac neutrino N_m. The

[2] Scales: $m_\nu \sim$ eV, $m_D \sim$ eV, $\mu \sim$ keV, $M \sim$ TeV.

pseudo-Dirac neutrino then decays into another lepton with opposite sign and another W, which then decays into another lepton and MET/a (light) neutrino:

$$q\overline{q'} \to W^{\pm(*)} \to l_1^{\pm} N_m \to l_1^{\pm} l_2^{\mp} W^{\pm(*)} \to l_1^{\pm} l_2^{\mp} l_3^{\pm} \bar{\nu}_l \tag{3.16}$$

The final state is then three charged leptons (trilepton) plus a neutrino which goes undetected through the detector, and is observed indirectly through large missing transverse energy in the event (like ATLAS). This decay process can be seen in Figure 3.1, and is produced through the charged current Drell-Yan (CCDY) process [1]. In this model, the lepton number is almost conserved. This is set by the mixing parameter μ. For the mixing of N1's couplings to electrons and muons, the mixings are set to $\mu = |V_e N| = |V_\mu N| = \frac{1}{\sqrt{2 \cdot 10^{-2}}}$ and $|V_\tau| = 0$[3] for no mixing to tau in the simulation models for charged lepton flavor violation (LFV). This means that the amount of opposite-sign and same-sign events for the first two leptons may differ from e.g. the normal seesaw model. The mixings allow LFV between vertex 1 and 2, i.e. an electron at vertex 1 and a muon at vertex 2 or vice versa, while the W boson decays according to the SM. As seen in equation 3.16, the leptons in the lepton pairs 1 and 2 and 2 and 3 must always have opposite sign (OS) while 1 and 3 always have same sign (SS).

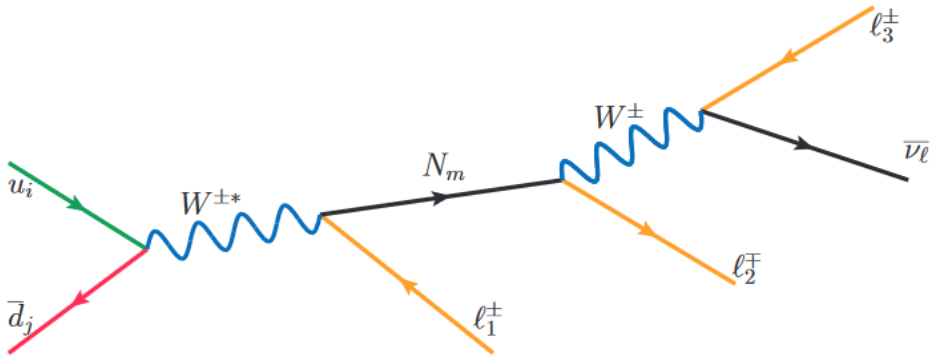


Figure 3.1: The Born diagram for the charged current Drell-Yan process of the proton-proton collision (on the left) producing a heavy pseudo-Dirac neutrino N in the inverse seesaw mechanism model, leading to a trilepton plus missing transverse energy (a light neutrino) final state. Figure is taken from ref. [1].

The decay products of such particle collisions can be detected in experiments like the LHC and ATLAS (sect.5.3). These events can also be simulated, meaning that we can simulate proton-proton collision events and the decay processes. For each decay final state product, we can measure many properties like momentum, the transverse momentum, the polar angle and the azimuthal angle. We can also detect which final state particles are produced. With these particle properties we can calculate the angles and angular distances between each produced particle, and for truth we have all the information about

[3]Equation 3.18 in Pascoli et al. [1].

the neutrino (MET). In a real detector, we only have the transverse information[4]. We can also calculate the invariant masses of pairs of combined final state particles. We should then be able to find out which lepton comes from which decay branch (vertex) in the decay process in Figure 3.1 computationally.

In Figure 3.2 we see a distribution of the lepton flavor between vertex 1 and 2 for the 150 and 450 GeV signals. We either have two electrons, or two muons (SF) or an electron and a muon (DF). This distribution shows that we expect more DF events and LFV for the 150 Gev signal than for the 450 GeV signal. The LFV is important if we were to discover some excess to better understand which neutrino mass model we are dealing with.

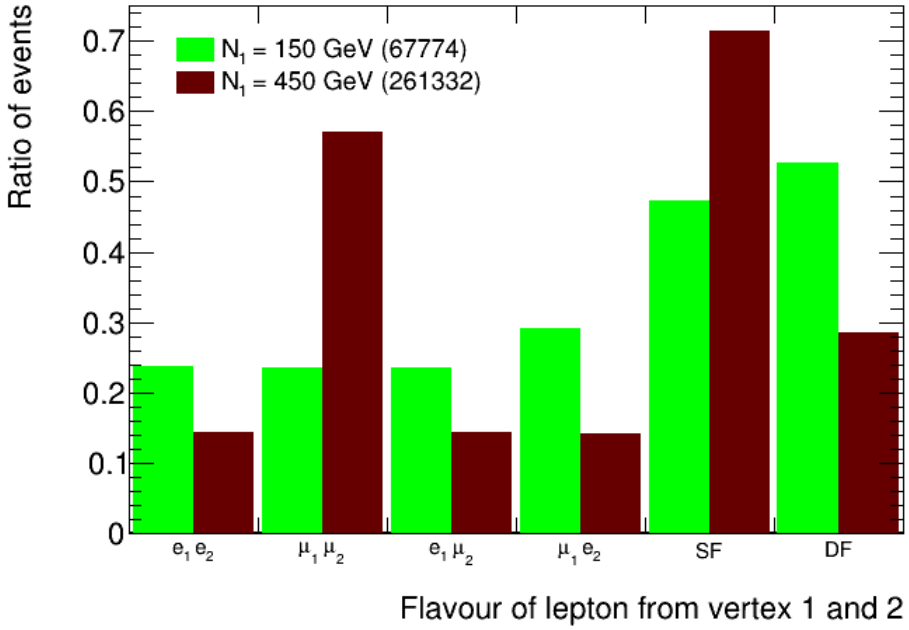


Figure 3.2: Distribution of the lepton flavor between vertex 1 and 2 for the 150 and 450 GeV signals where we either have two electrons, or two muons (SF) or an electron and a muon (DF).

The end goal is to identify the decay vertices (according to Fig. 3.1), by utilizing the particle properties in various machine learning algorithms. We will look more into machine learning in chapter 6. The data we are analyzing are covered in section 7.2.

[4]I.e. no p_z and no θ.

Chapter 4

Proton-Proton Collisions

In this thesis we study the proton-proton (p-p) collisions from LHC (sect.5.2). Protons consists of quarks and this makes proton-proton collisions somewhat complex. When two hadrons collide, it is the constituents of the hadrons[1] which collide. The colliding partons only carry fractions of the total momentum of the protons. We use the center-of-mass (CM) frame of the p-p collision system and not the CM frame of the patrons that collide. This chapter explains the basics of high energy proton-proton collisions.

4.1 Particle Kinematics

To describe the kinematics of what happens in p-p collisions, we need the momentum, energy and rest mass of the particles. The Einstein energy-momentum relation in natural units becomes

$$E^2 = p^2 + m^2. \tag{4.1}$$

Since the protons will reach very high velocities when they collide, we need to include special relativity into the equations[2]:

$$E = \gamma m \quad \text{and} \quad \mathbf{p} = \boldsymbol{\beta}\gamma m \tag{4.2}$$

These equations depend on the Lorentz factor

$$\gamma = \frac{1}{\sqrt{1-\beta^2}} \quad \text{and} \quad \beta = \frac{v}{c}.$$

We then introduce the momentum as a four-vector momentum

$$P^\mu = (E, \mathbf{p}) = (E, p_x, p_y, p_z).$$

[1] The partons, i.e. quarks and gluons.
[2] In natural units.

The scalar product of the four-momentum is then a Lorentz-invariant quantity

$$\begin{aligned} P^2 = P^\mu P_\mu &= E^2 - \mathbf{p}^2 \\ &= \gamma^2 m^2 - \beta^2 \gamma^2 m^2 \\ &= m^2, \end{aligned} \tag{4.3}$$

since the momentum and energy are conserved separately, the four-momentum is also conserved. By rearranging this equation, we just end up with the Einstein energy-momentum relationship in equation 4.1. This is a very useful relation in particle collisions.

4.1.1 Colliding Particles

The reference frame of choice for colliding particles, is as mentioned the CM frame of the two colliding particles. This is defined where the sum of the three-momenta $\mathbf{p}$ is zero. When two particles collide, this means that $\mathbf{p}_1 = -\mathbf{p}_2$. And when these two particles have the same rest mass $E_1 = E_2 = E$, we get

$$(P_1 + P_2)^\mu = (2E, \mathbf{0}). \tag{4.4}$$

Now we introduce what is called a Mandelstam variable[5], s , which is defined as the squared sum of the four-momenta

$$s = (P_1 + P_2)^2. \tag{4.5}$$

This we have already found out is a Lorentz-invariant quantity. We can then draw two conclusions; 1) s is a Lorentz-invariant quantity as well, and 2), the $\sqrt{s} = 2E$ can be interpreted as the total energy of the CM system. This is a key quantity in particle physics for particle colliders.

From equation 4.3, we got that $P^2 = m^2$. This means that if the colliding particles were elementary particles, $\sqrt{s}$ could be interpreted as the possible energy available for heavier particle production. This would then be an upper limit for producing a heavy particle with mass M, as $M \leq \sqrt{s}$. But since protons are not elementary particles and the p-p collisions are really collisions between partons, this limit changes. We denote the momenta carried by the two partons colliding as $\mathbf{q}_1$ and $\mathbf{q}_2$. The associated four-momenta for the partons are Q_1^μ and Q_2^μ. Since we mentioned that the partons only carry fractions of the momenta, these fractions will be defined as x_1 and x_2 for the two colliding partons. By using what is called the Drell-Yan process[3] (explained and derived in Thomson [5]) for a quark and an antiquark, we get the fractions given as

$$x_1 = \frac{q_1}{E} \quad \text{and} \quad x_2 = \frac{q_2}{E}. \tag{4.6}$$

[3]This is not restricted to Drel-Yan processes, but yields for any 2-¿1 process.

To get the mass M of a produced particle from the collision with the partons, we use the same limit as for an elementary particle collision and equation 4.5 for s:

$$\begin{aligned} M &\leq \sqrt{s} \\ M^2 &\leq s \\ M^2 &\leq (Q_1 + Q_2)^2 = E^2 \left[(x_1 + x_2)^2 - (x_1 - x_2)^2\right] \\ &= 4x_1 x_2 E^2 \\ &= x_1 x_2 s \end{aligned}$$

This leads to that the produced invariant mass is equal to the CM energy of the colliding partons.

The actual values of the fractions are described by the parton distribution functions (PDFs). These PDFs can be interpreted as the probability of a parton with a special flavor to carry the fraction x of the proton momentum when the parton participates in a hard scattering process.

From this section, we can see that the event kinematics in hadron-hadron collisions have to be explained by the three independent kinematic variables, Q^2, x_1 and x_2.

4.1.2 Products of Particle Collisions

In particle colliders, like at the LHC, the direction of the particle beams are normally defined in the z-direction which gives $\mathbf{p} = (0, 0, p)$. This plane is the longitudinal plane. The positive y-direction is defined upwards, and the positive x-direction is defined towards the center of the ring. We can then define the transverse momentum p_T perpendicular to the z-axis as

$$p_T = \sqrt{p_x^2 + p_y^2}. \tag{4.7}$$

The corresponding transverse energy is given as

$$E_T = \sqrt{p_T^2 + m^2}. \tag{4.8}$$

The total momentum can then be derived as

$$p = \sqrt{p_T^2 + p_z^2}. \tag{4.9}$$

The reason for working in the transverse (xy) plane of the initial beam direction, is that the initial momentum is zero in this direction. We want to express the kinematics in spherical coordinates in terms of the polar angle θ and the azimuthal angle ϕ.

After the collisions, not just the parton jets, but the whole system will get a boost along the beam direction. That is why we introduce a *rapidity* variable y that is used to express the jet angles:

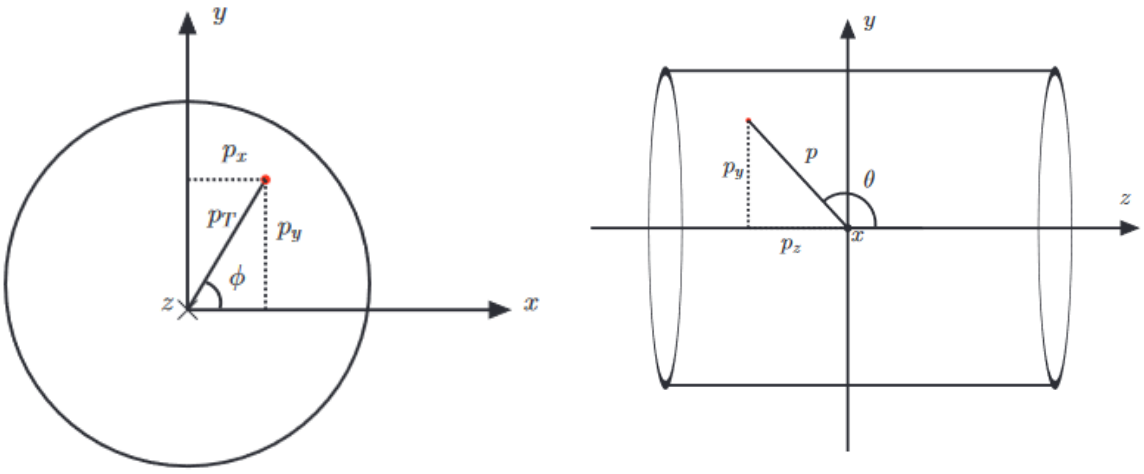
$$y = \frac{1}{2} \ln \left(\frac{E + p_z}{E - p_z} \right) \tag{4.10}$$

What is useful with this rapidity variable, is that the rapidity differences are invariant under Lorentz boosts along the beam direction. This does not apply for the polar angle θ.

If the particle mass is small compared to the particle energy, $p_z \approx E\cos\theta$. We can then rewrite the rapidity as

$$y \approx \frac{1}{2}\ln\left(\frac{1+\cos\theta}{1-\cos\theta}\right) = \frac{1}{2}\ln\left(\cot^2\frac{\theta}{2}\right) = -\ln\left(\tan\frac{\theta}{2}\right) \equiv \eta \tag{4.11}$$

This new variable η is called the *pseudorapidity*. The pseudorapidity also has the following relation with the polar angle: $\eta(\theta) = -\eta(180^\circ - \theta)$. We now have the most used set of variables (p_t, ϕ, θ) for describing the kinematics of particles in a detector. In Figure 4.1 we see the illustration of the transverse and longitudinal planes. The cylindrical shape shows how particle accelerators will be situated around the collision point.



(a) The transverse plane.

(b) The longitudinal plane.

Figure 4.1: Illustrations of the (a) transverse plane and the (b) longitudinal plane. The collision point is at the origin. Figures are both from ref. [31].

Another useful variable associated with hadron colliders, is the angular distance between two particles

$$\Delta R = \sqrt{(\Delta\eta)^2 + (\Delta\phi)^2}. \tag{4.12}$$

The angular distance defines how much two particles are moving in the same direction or as the separation in the $\phi\eta$-space, and is invariant under longitudinal boosts.

4.2 Proton-Proton Interactions

When proton-proton collisions take place in colliders, the interactions can roughly be divided into three groups:

i) elastic (el) ii) diffractive (di) iii) non-diffractive (nd)

These three groups are also components that make up the total cross-section at proton-proton colliders:

$$\sigma_{\text{total}} = \sigma_{el} + \sigma_{di} + \sigma_{nd} \tag{4.13}$$

For elastic processes, both the colliding protons remain unchanged. For the diffractive processes (di and nd), the collisions/interactions are inelastic and one or both protons will be fragmented. This leads to multi-particle final states.

The elastic and diffractive interactions have cross-sections that can not be calculated using perturbation theory, meaning they are non-perturbative processes. In these cases we get so-called *pomerons*, which are color singlet states that do not exchange color between the protons. These interaction processes at high-p_T proton-proton collisions are normally not interesting, since they will produce particles with low transverse momentum close to the beam line. They are thus difficult to detect, but important for luminosity measurements since they contribute to the total p-p cross-section. These events are detected in special experiments that use *minimum bias* events, where the final state has no requirements or special triggers.

4.2.1 Hard Scattering Events

The more interesting events to look at in high-p_T p-p collisions, are the non-diffraction events. With non-diffractive events, there is an exchange of color between the partons in the interaction. These are called hard scattering events. Hard scattering events with high momentum transfers, Q^2, may create heavy particles. This is the main interest in particle colliders.

A hard scattering event can be expressed as

$$A + B \to c + X, \tag{4.14}$$

where the collision between the partons are expressed as

$$a + b \to c. \tag{4.15}$$

A and B are the two colliding protons, and a and b are the corresponding colliding partons. c are the interesting high p_T objects. X are underlying products which are mostly remnants after the original collision.

In Figure 4.2 we see how a hard scattering p-p collision may look like, with outgoing partons, underlying events, initial- and final-state radiation. The initial-state radiation is mean radiation of gluons or photons from partons before the hard scattering. Final-state radiation is the mean radiation from the produced partons after the hard interaction. The underlying events are the further interactions between partons beyond the hard scattering. These interactions will often go out of reach of the detector, and is another reason why we look at the transverse plane.

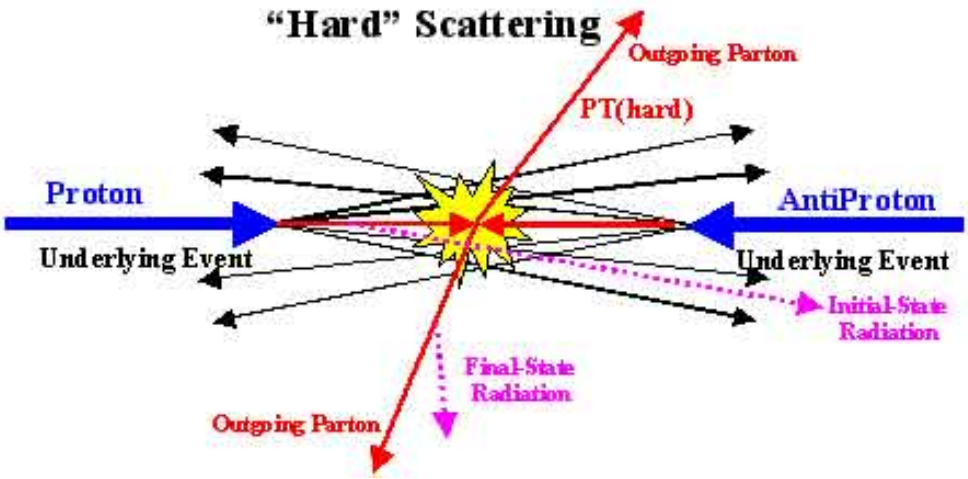


Figure 4.2: Illustration of a hard scattering proton-proton collision. Figure is taken from ref. [32].

4.2.2 Parton Distribution Function

The parton distribution function (PDF)[4] is used to describe the probability density of the two partons, a in proton A and b in proton B, to carry the proton momentum fractions x_a and x_b. These PDFs are also dependent on the squared of the momentum scale indicating the total four-momentum transfer in the collisions Q^2 as $F_{a/A}(x_a, Q^2)$ and $F_{b/B}(x_b, Q^2)$. These PDFs must be found experimentally in Deep Inelastic Scattering (DIS) experiments of leptons against hadrons, since they cannot be calculated from QCD theory. The PDFs are also used to get the cross-section of the collisions.

With the measured PDFs $f(x, Q^2)$, a structure function $F_2^{ep}(x, Q^2)$ can be determined

$$F_2^{ep}(x, Q^2) = 2xF_1^{ep}(x, Q^2) = x \sum_i Q_i^2 f_i(x), \tag{4.16}$$

where i is a quark in the proton and Q_i is the charge of the quark. The interesting here are the $f(x)$ of each of the partons. So results of measurements from several DIS experiments of varying structure functions, which are superpositions of the same $f_i(x)$'s, are combined to get the $f(x)$ for each parton.

4.2.3 Hadronization

We already have covered that quarks and gluons carry color charge (sect. 2.1), and that they are not observed as free particles[5]. They can only be found in colorless objects like hadrons.

We also talked about the strong force, which increases in strength when increasing the distance between (elementary) particles. So if we separate a quark from a hadron, the

[4] See chapter 8 in Thomson [5] for more in depth explanations.

[5] Only exception is the top quark with shorter lifetime than the QCD interaction time scale.

color field will increase and the emerged energy will enable creation of new quark-anitquark pairs or gluons. These will be observed as jets of colorless particles. As this production of partons continue, the energy will decrease until it is low enough to produce hadrons. This process of high-energy quarks (and gluons) that produce new jets until we get hadrons, is called *hadronization*. The jets can also be called hadronic showers, since many hadrons are usually produced in hadronization processes.

Jets are not only produced in p-p collisions with hard scattering, but also in the underlying events and from initial- and final-state radiation. This makes p-p collisions very complicated and messy when trying to study them, compared to electron-positron collisions.

Chapter 5

Particle Accelerators and Collider Experiments

To fully understand the physics of the particles around us and what the Universe is made of, we need some way of looking at the subatomic world. This is done in huge particle accelerators where particles are accelerated to high velocities and energies, and collided with each other to make other particles. Here the aftermath of the collisions result in new particles with new energies that are detected as they move through detectors.

There are various accelerators and detectors which produce and accelerate different particles in the world. In this chapter we will look at the biggest particle physics laboratory in the world, namely the European Organization for Nuclear Research (CERN[1]), and some of its components like particle accelerators and detectors.

5.1 CERN

The CERN laboratory lies near Geneva, on the border between France and Switzerland, and was founded in 1954 [33]. It is a multinational collaboration between 23 (mostly) European countries. They also have several international relations with other countries both inside and outside of Europe. CERNs main focuses today is particle physics and particle accelerator experiments. Many of the biggest discoveries in particle physics have come from particle experiments at CERN. This includes, among others, the discovery of the Higgs boson and discovery of the W and Z bosons. At the main site of CERN in Meyrin, and in the World LHC Computing Grid (WLCG) scattered around the world, data of simulations of particle collisions are stored. CERN is the place where Tim Berners-Lee invented the World Wide Web in the late 1980s [34].

CERN consists of several particle accelerators, experiments and facilities in different shapes and sizes. The two main types of accelerators are linear and circular. They are located at various sites, and they accelerate particles to high energies before they send the particles to be collided with other accelerated particles or particles with stationary

[1]The name CERN is originally from French; Conseil Européen pour la Recherche Nucléaire.

targets, or are sent to more powerful accelerators. They are built differently to accelerate different kinds of particles with different masses. In Figure 5.1 we see the CERN accelerator complex. Some of the accelerators are mostly used to pre-accelerate the particles before they are sent to another accelerator where they are accelerated even more. This repeats until the particles reach the desired energy to collide with at one of the detectors.

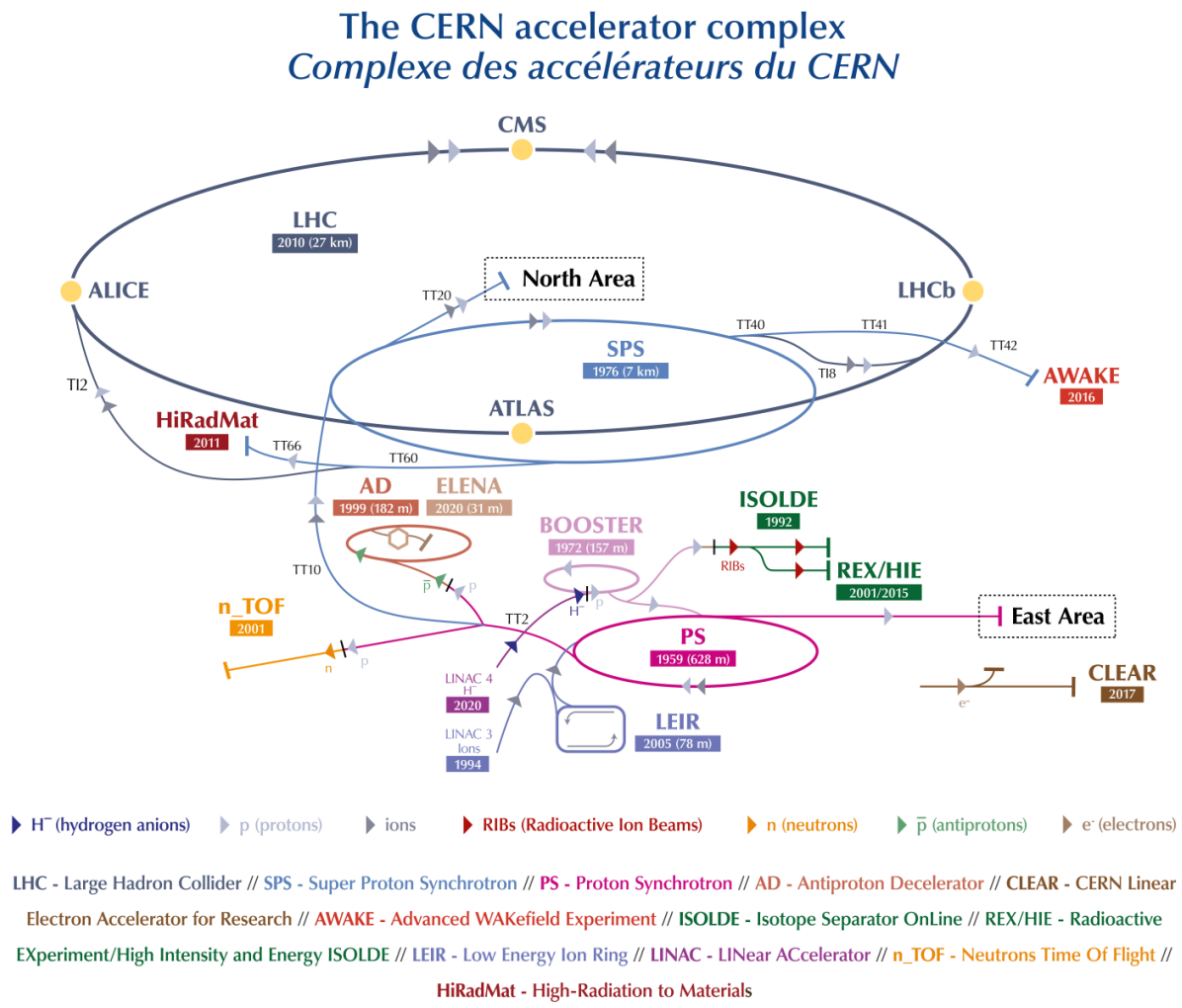


Figure 5.1: The CERN accelerator complex as of 2019. Credit: CERN[35].

For the more important discoveries, like the ones we have mentioned above, the W and Z bosons were discovered by the Super Proton Synchrotron (SPS) in 1983. The SPS delivered an energy between 300-450 GeV. It was then later used to accelerate high energy electrons and positrons into the Large Electron-Positron Collider (LEP). LEP is the largest and most powerful lepton collider built to this date, and was functional between 1989 and 2000. LEP was then replaced by the Large Hadron Collider (LHC) in 2008 to collide protons and heavy ions.

5.2 The LHC and Accelerator Experiments

Today's largest and most powerful particle accelerator is the Large Hadron Collider (LHC) [36], which we easily can see in Figure 5.1 as the biggest gray circle around the North Area. The particles are sent in bunches up to 10^{11} protons and are accelerated using radio frequency cavities in a 27 km ring consisting of superconducting magnets, where the particles are boosted in several structures along the ring to the desired energies. The LHC is designed to have 2808 bunches at the same time traveling in the ring. The ring lies 100 m underground in a tunnel beneath the French-Swiss border. Along the ring, there are 4 main crossing points (ATLAS, CMS, ALICE, LHCb) with detectors that register the particle collisions and the following particle decays. At these collision points, the total collision energy, or center-of-mass energy $\sqrt{s}$, can reach 13 TeV[2]. There are in total seven detectors along the ring, each designed for different experiments.

The LHC was first used for proton-proton (hadron) collisions in 2010 (run 1), where it reached a record high energy of 3.5 TeV per beam. After upgrades, during run 2, it reached an even higher energy of 6.5 TeV per beam. It is currently stopped for another upgrade, which started in 2018 and is during operation. The accelerator sends two high-energy beams, in separate tubes and directions, near the speed of light before they collide at one of the detectors. To reach these high energies, the particle beams are accelerated in several systems which increase the energies before injected into the main LHC ring [37]. Inside the tubes, there is an ultrahigh vacuum. To make sure that the particles are directed correctly through the ring, superconducting electromagnets are used to bend the particle trajectories. The magnets vary in strengths and sizes to direct the beams properly. Since the particles are incredibly tiny, the precision of the magnets have to be extremely good to make the particles hit each other at the collision points. That is also why beams of 10^{11} protons are accelerated and not single particles. Since the construction of the accelerator is a ring they can continue around again when some of them do not collide. A beam can typically go around in the ring for about 10 hours before the beam has lost too much intensity.

As mentioned earlier, there are seven detector experiments at the LHC [38]. The four main, and biggest, detectors in the LHC, have different objectives. The ATLAS and CMS experiments are two large and similar general-purpose particle detectors that looks for new physics and more precise study of the SM. The ALICE and LHCb experiments have more specific roles, and study the quark-gluon plasma from heavy ion collisions and missing antimatter connected to CP-violation after the Big Bang, respectively. The remaining detectors are much smaller and are used in more specialized research. We will look more at the ATLAS detector later (sect.5.3).

The LHC is used to explore many different open questions in physics, like to further study the SM and theories beyond it. In addition to proton-proton collisions, the LHC can also collide heavy ion collisions at some of the detectors.

[2]The LHC is theorized to a limit of 14 TeV.

5.2.1 Important Parameters

One of the most important parameters of measurements at particle accelerators, is the CM energy $\sqrt{s}$ we already have mentioned. For two particles colliding, the Lorentz invariant quantity s (the squared invariant mass) is formed as

$$s = \left(\sum_{i=1}^{2} E_i\right)^2 - \left(\sum_{i=1}^{2} \mathbf{p}_i\right)^2. \tag{5.1}$$

There are also other important parameters used to describe the performance of particle colliders:

Luminosity

Another important parameter in particle collider performance is the *luminosity*, $\mathcal{L}$. The design luminosity of the LHC is $\mathcal{L} = 10^{34}$ cm^{-2}s^{-1}. The bunches at the LHC are separated by 25 ns, which corresponds to a frequency of $f = 40$ MHz. The (instantaneous) luminosity is used to describe the number of collisions per area per second as[3]

$$\mathcal{L} = f\frac{n_1 n_2}{4\pi\sigma_x\sigma_y}, \tag{5.2}$$

where f is the frequency of the particle beam bunches colliding (bunch crossing rate), n_1 and n_2 are the number of particles in the colliding bunches and σ_x and σ_y are the root-mean-square (rms) horizontal and vertical beam sizes.

The complete collider luminosity at the LHC can be written in terms of colliding beam parameters [39]

$$\mathcal{L} = f\frac{n_1 n_2 n_b}{4\pi\sigma_x\sigma_y}F(\sigma_x, \sigma_y, \sigma_s, \Phi). \tag{5.3}$$

This equation has the same parameters as in equation 5.2, except for two additional parameters. n_b is the number of proton bunches. F is a geometrical reduction factor accounting for the non-zero-crossing angle at the interaction point, depending on the two rms beam sizes, the beam length σ_s and the crossing angle Φ.

Rate

The cross-section, σ, for a given collision process is given by the SM (or any other new model). The cross-section can be used to compute the (event) *rate*, R, after accumulating many such collisions. The rate is calculated as

$$R = \sigma\mathcal{L}. \tag{5.4}$$

[3] With the assumption of Gaussian profile beams and head-on collisions.

Number of interactions

The total number of expected events of a given process with cross-section, σ, over a given time, is the time integration of the event rate

$$N = \sigma \int \mathcal{L} dt. \tag{5.5}$$

The time-integral of the luminosity, $\int \mathcal{L} dt$, is often called the *integrated luminosity*, and is given in inverse femtobarns $[fb^{-1}]$.

Pile-up

In particle collisions, we want a high instantaneous luminosity. This means that the intensity of the proton beam need to be high. But with high intensity proton beams, the probability of having more than one proton undergoing an inelastic interaction per bunch crossing is increased. This leads to what is called *pile-up* events, where there are several collisions from the same bunch crossing. This means we need very accurate measurements in detection of the particle tracks to distinguish which new particles comes from which collisions. The main event that is normally used in detection, and this corresponding vertex is called the *primary vertex*.

Since we want higher and higher luminosity to get more collisions, we also get more pile-ups. This need to be controlled to be able to use the data efficiently. The additional collisions do normally have smaller momentum transfers, which means we can characterize them as minimum bias events.

5.3 The ATLAS Experiment and Particle Detection

To detect the particles produced at particle colliders, we need different instruments that can detect the various types of particle interactions. The largest detector at the LHC is the ATLAS (A Toroidal LHC ApparatuS) experiment. In Figure 5.2 we see a computer generated image of the ATLAS detector with pointers to the main components. It is 25 m in diameter, 46 m long and weights about 7000 tons. The cylindrical shape of ATLAS is optimized to detect as many particles as possible, and covers almost a 4π angle with detectors. Like we mentioned earlier for particle collisions in the LHC, ATLAS uses the same Cartesian coordinate system with the z-direction in the direction of the beam, y-direction is upward and x-direction is towards the center of the accelerator circle. It also uses a spherical coordinate system with the azimuthal angle ϕ in the xy-plane around the beam axis, and the polar angle θ being the angle from the beam axis. To measure the distance between the particles, the angular distance ΔR (eq.4.12) in the $\phi\eta$-plane is used.

The ATLAS detector is designed to be a general-purpose detector, covering a wide range of signals. The particle properties the ATLAS detector can detect is the mass, momentum and energies of the particles. For ATLAS to detect these properties, it has a layered design of detectors that is optimized in observing specific properties of the various particles. The

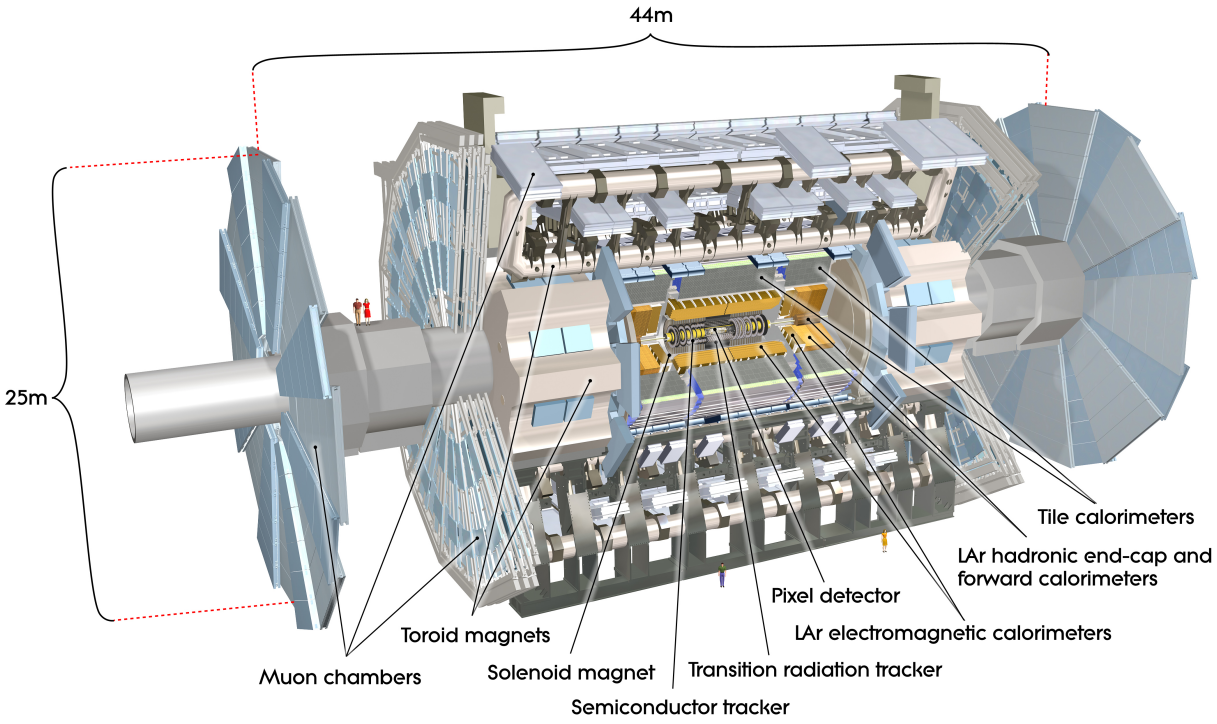


Figure 5.2: The ATLAS detector and its components. Credit: CERN [40].

ATLAS detector consists of several main systems; the inner detector (ID), calorimeters, a muon spectrometer (MS), a magnet system and a trigger and data acquisition system. The main systems consists of smaller sub-systems, which we will take a brief look at next. In Figure 5.3, we see a sketch of the detector layout systems and how some particles behave in these different systems. Only the neutrinos should now go undetected through the detectors, in principle, and they are normally identified as missing momentum, or MET. This comes from the energy conservation law, where the sum of the measured transverse momenta of the all particles produced should be zero.

5.3.1 Inner Detector

The inner detector tracks charged particles that leaves traces of ionized atoms when traveling through a medium. The tracks, momentum and charges of the particles can be traced in a 2 T magnetic field that makes the charged particles curve. The degree of the curvature is used to determine the charge and the momentum.

The inner detector consists of three sub-systems. The inner most part is a silicon Pixel Detector that is used for extremely precise tracking near the interaction point of the particle collisions. The second part is a Semiconductor Tracker (SCT) that covers a bigger area than the pixel detector for the particle tracking and uses long and narrow strips instead of

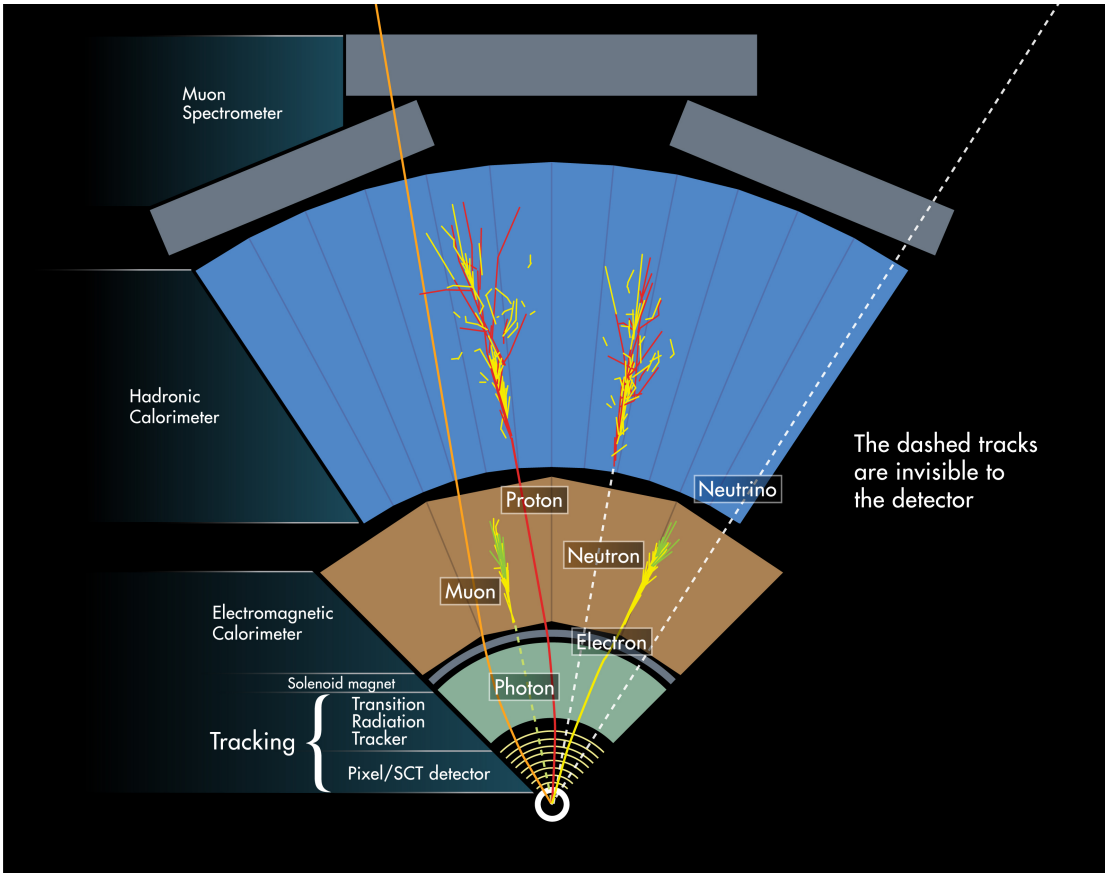


Figure 5.3: An illustration of the main tracking systems in the ATLAS detector, including how some particles behave in the various systems. Credit: ref. [41].

pixels. The third part is a Transition Radiation Tracker (TRT) that covers an even larger area with lower spatial resolution, and can detect transition radiation photons by using gas filled drift/straw tubes. The TRT provides the capability of electron identification for a variety of energies since the transition radiation gives out a stronger signal than ionization signal.

5.3.2 Calorimeters

Outside the ID and the solenoid magnet system, there follows two types of calorimeters; an (inner) electromagnetic calorimeter and a (outer) hadronic calorimeter. Their purpose is to measure the energy of the passing particles and particle showers especially.

The **electromagnetic** calorimeter (ECal) measures particles that interact electromagnetically, like charged leptons and photons. The ECal is made of layers of lead absorbing plates and liquid argon, and covers the whole ϕ angle around the beam axis. The energy is measured in the liquid argon, and free electrons are picked up by electrodes. The ECal is covered by cryostats to keep it at the correct low temperature.

The **hadronic** calorimeter (HCal) measures hadrons and hadronic showers[14]. The HCal is made of several layers of steel absorbers and plastic scintillator tiles that alternates. The iron in the detector both slows down and traps hadrons. The HCal is a lot bigger than the ECal.

5.3.3 Muon Spectrometer

Outside the calorimeters, we find the muon spectrometer. Here high-energy muons are detected. This detector is very large, 11 m radius [42], and consist of three parts; a magnetic field with several toroidal magnets, a set of chambers measuring the tracks of the muons and a set of triggering chambers with accurate time-resolution. The detection of the muons happens the same way as before, by measuring their momentum as they are bent in the detector. They should also be simpler to identify since all other identifiable particles should not reach this far out from the interaction point.

5.3.4 Magnet System

ATLAS uses two types of superconducting magnet systems to measure the momentum from the bending of the particles through the Lorentz force. The magnet system consists of a central solenoid, a barrel toroid and two end-cap toroids. The central solenoid is located between the inner detector and the electromagnetic calorimeter, which produces the 2 T magnetic field for the ID. The barrel toroid produces a magnetic field of 0.6 T, and is located around the middle cylinder of the MS barrel outside the calorimeters. The two end-cap toroids produce magnetic fields of 1 T, and are located at the end-cap regions of the Muon System.

5.3.5 Trigger System

The detector produces a huge amount of data, which need to be stored and processed. The output event storage rate have to be reduced from an initial bunch crossing of 40 MHz to ∼200 Hz. To only get the most interesting data for further analysis, a trigger system is used to extract these relevant events. The ATLAS Trigger and Data Acquisition system (TDAQ) has three levels for reducing the amount of stored data [43]; the Level 1 (LVL1) trigger is hardware-based and makes quick decisions of which events to store, the Level 2 (LVL2) and the Event filters (EF) are software-based and are often combined to and referred to as the High Level Triggers (HLT). Only the events passing both the LVL1 and HLT are stored for further analysis.

The LVL1 trigger uses information from the calorimeters and the muon spectrometer to choose interesting events. These interesting events passed on to the next trigger. The LVL1 trigger also defines regions based on the ϕ and η coordinates from the interesting events.

[14]It measures the energy of particles that interact via the strong force, which is mainly hadrons.

The LVL2 trigger uses all the information within the regions of interest (ROIs) defined by the LVL1 trigger to further reduce the amount of event data. The accepted events are then assembled put together into a full event. The EF uses an offline analysis to even further reduce the data used to store and further analysis at the WLCG.

Chapter 6

Machine Learning

6.1 Introduction

Machine learning (ML) has recently become widely used in many fields of research. The meaning of machine learning is to train computational algorithms to automatically determine an outcome from specific patterns in data the algorithms have not seen before by using pre-trained algorithms with a given input set of hyperparameters. When training an algorithm, one tries to teach patterns using large amounts of data. ML goes in under what is called artificial intelligence, which is where the computer takes its own decisions to produce and predict solutions to problems.

The machine learning algorithms build a model based on some given data and general rules. The data may often need to be processed in some way, like when there are missing values in the dataset. The models are then fit and trained on sample data, which is a subset of the full dataset. The remaining data, which is a smaller part than the training data, are used to make predictions and do an evaluation of the trained model. There is a huge variety of different evaluation metrics which are used to check the performance of the algorithms on data. When we have a good enough trained model, we can save it and use it later on similar unseen data.

There is a plethora of usages for machine learning, and it is often divided into estimation or prediction problems. An example of a machine learning problem can be to identify objects in images of animals, which may be easy to humans. Algorithms can be trained to identify various animals by the algorithms given some features to best distinguish the animals from each other. This may be the shape of ears or the tail of the animals. Computationally this means we choose some observable quantity $\mathbf{x}$ in the data we look at which are related to some parameter θ. The model $p(\mathbf{x}|\theta)$ is describing the probability of observing $\mathbf{x}$ given θ. A dataset $\mathbf{X}$, also called a design matrix, is produced to fit the model. The design matrix only consists of feature data, while the class variables are stored in a target vector $\mathbf{y}$. These two datasets are often split into training and test sets, and sometimes even into training, test and validation sets. The fitting of the model then tries to find the parameters $\hat{\theta}$ which best explains the data. In this thesis, it is the accuracy of the model that we want to optimize and focus on. Optimizing the accuracy of $\hat{\theta}$ is often the concern

with estimation problems, where as prediction problems focuses more on how the model makes new predictions.

Most machine learning problems consists of the same ingredients, starting with a dataset $\mathcal{D} = (\mathbf{X}, \mathbf{y})$, where $\mathbf{X}$ is the matrix containing the independent variables $\mathbf{x}$ and $\mathbf{y}$ is a vector containing the dependent variables. Then there is a model as a function $\mathbf{f} : \mathbf{x} \to \mathbf{y}$ with the parameters θ. The function is used to predict the outputs given vectors of input variables. For the predictions to take place, we need a cost function $\mathcal{C}(\mathbf{y}, \mathbf{f}(\mathbf{X}; \theta))$ that judges how well the model performs on the observations. When fitting the model, we want the $\hat{\theta}$ which best explains the data. When considering a linear regression case with the sum of least squares as the cost function,

$$\mathcal{C}(\mathbf{y}, f(\mathbf{X}; \theta)) = \sum_{i}^{N} (y_i - f(\mathbf{x}_i; \theta))^2, \tag{6.1}$$

we get the best fit with the set of parameters that minimize the cost function:

$$\hat{\theta} = \underset{\theta}{\operatorname{argmin}} \{\mathcal{C}(\mathbf{y}, f(\mathbf{X}; \theta))\} \tag{6.2}$$

The ML approaches are usually divided into supervised, unsupervised and reinforcement learning [1]. **Supervised learning** already has the answers or outputs before we do anything to the model. The dataset needs to be labeled and have the answers to the problem such that the algorithms know what is correct. During training, the algorithm predicts the answers from what it has learned. If we are not satisfied with the accuracy the algorithm provides, we change the hyperparameters or the algorithm until we are satisfied with the results. **Unsupervised learning** does not have any labeled data or correct answers, meaning that it has to find its own structure in the inputs. The algorithms can only use predefined metrics to make a conclusion. This can then be used to discover hidden data patterns or to reproduce the given input. **Reinforcement learning** uses a dynamic environment that has a specific goal. As the problem is solved through trial, error and experience, the program tries to maximize its rewards from feedback during the problem solving. The program then trains itself to make decisions.

This chapter takes a closer look at the supervised learning category in machine learning and some of the basics of statistical learning, as well as classification and multiclass classification, which is used in this thesis. The theory is mostly based on the works of Hastie et al. [45] and Mehta et al. [46].

6.2 Supervised Learning

For supervised learning, we already mentioned that we need the outputs, labeled data and the need to tune hyperparameters for optimization. The inputs may also be called

[1] There exists other approaches that goes beyond these three mentioned approaches. The most dominant approach today of these is called deep learning. See Goodfellow et al. [44] for more on deep learning and other possible machine learning tasks.

independent variables, while the outputs can be called dependent variables. Supervised learning can be divided into different learning algorithms; classification, regression and active learning. **Active learning** algorithms uses a source with information to label data points with some desired output. **Regression** algorithms uses a given set of features and inputs, and estimates the relationship between the features and an outcome variable. Regression is mostly used for problems with a variation of outcome values, or a continuous output, within a range of values. **Classification** algorithms has a limited set of values as outputs, which can be categories, numbers or names. Classification uses pattern recognition in sets of categories of discrete variables to identify new observations or to group unseen data based on the inputs. We will take a closer look into the basics of statistical learning with a focus on supervised learning next.

6.2.1 Basics of Statistical Learning

In statistical learning, the goal is to find a function h in a hypothetical set $\mathcal{H}$ such that $h \in \mathcal{H}$ approximates an unknown function $y = f(x)$ as best as possible. $\mathcal{H}$ consists here of all possible functions that are defined in the domain of f and are of interest for the problem at hand. With the newly developed function $h(x)$, we would then get $h \approx f$. The *expected error* for a particular function h over all inputs x and outputs y is given by the cost function $\mathcal{C}$ and the joint probability distribution for x and y as:

$$\mathbb{E}[h] = \int_{X \times Y} \mathcal{C}(h(x), y)\rho(x, y)dxdy. \tag{6.3}$$

In this case we need knowledge of the probability distribution, which we in most cases do not. For n data points, we can instead use the *empirical error*:

$$\mathbb{E}_E[h] = \frac{1}{n}\sum_{i}^{n} \mathcal{C}(h(x_i), y_i). \tag{6.4}$$

With the expected and empirical errors, we can compute the *generalization error* as the difference between those two:

$$G = \mathbb{E}[h] - \mathbb{E}_E[h]. \tag{6.5}$$

In the limit of the generalization error goes towards zero,

$$\lim_{n\to\infty} G = 0,$$

we say that an algorithm can learn or generalize from the data. In general, we cannot compute the generalization error since we in general cannot compute the expectation error. To solve this we can divide our dataset into training and test sets, and then use cross-validation to estimate the generalization error. The values on the cost function on the training and test sets are called the *in-sample* error, E_{in}, and *out-of-sample* error, E_{out}, respectively. The in-sample error can be an appropriate approximation to the generalization error if the dataset is large enough and is representative of the function f.

In Figure 6.1 we see how the errors in general behave when the training set size, or number of data points, increases. We have assumed here that the number of data points is large and that the true function $f(x)$ can't be exactly fit. As the number of data points increase, we see that the in-sample error increases while the out-of-sample error decreases. The sampling noise decreases since the error difference between the two errors decreases. The out-of-sample error we get from this sampling noise is called the *variance*, which goes towards zero in the infinite data limit. As the training dataset approaches the infinity limit, we can conclude that the two errors must go to the same value. This is called the model *bias*. The bias is a representation of the best our model can do with infinite data size.

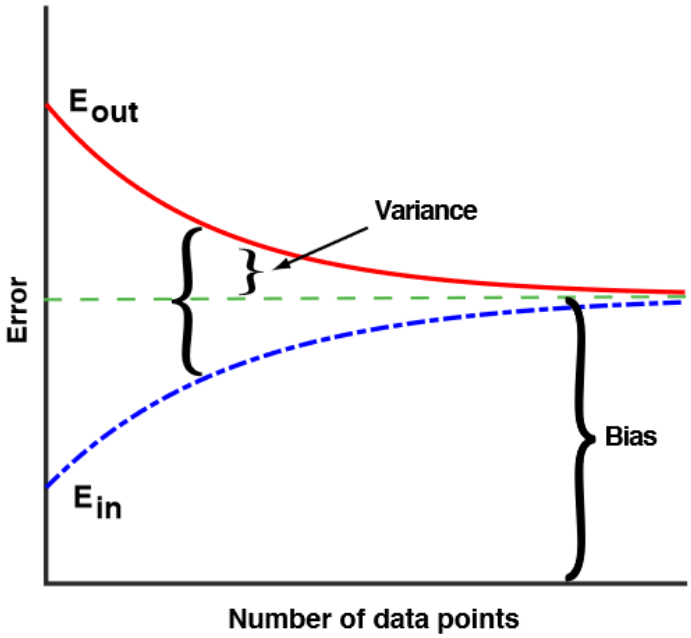


Figure 6.1: Illustration of the in-sample error, E_{in}, out-of-sample error, E_{out}, variance, bias and difference of errors as function of the training set size. It is assumed that the number of data points is not small, and that we cannot exactly fit the true function $f(x)$. The training error increases while the test error decreases as the training set size increases. Figure is taken from ref. Mehta et al. [46].

6.2.2 Bias-Variance Decomposition

We will now go a bit further into the bias and variance that is an important aspect of machine learning. Lets consider a dataset $\mathcal{D}(\mathbf{X}, \mathbf{y})$ with N pairs of independent and dependent variables. We then assume that the true data is created from a noise model

$$y = f(x) + \epsilon, \tag{6.6}$$

where ϵ is a normally distributed noise with mean zero and standard deviation σ_ϵ. A chosen estimator $f(\mathbf{x};\hat{\theta})$ is trained by minimizing the cost function, lets say the sum of squared errors[2],

$$\mathcal{C}(\mathbf{y}, f(\mathbf{X};\theta)) = \sum_i (y_i - f(\mathbf{x}_i;\theta))^2. \tag{6.7}$$

Our best estimates for the model parameters,

$$\hat{\theta}_{\mathcal{D}} = \underset{\theta}{\operatorname{argmin}} \{\mathcal{C}(\mathbf{y}, f(\mathbf{X};\theta))\}, \tag{6.8}$$

are functions of the dataset $\mathcal{D}$. Then we make another set of datasets $\mathcal{D}_n = (\mathbf{y}_n, \mathbf{X}; n)$, where all sets have N samples. We want the expectation value, $\mathbb{E}_{\mathcal{D}}$, of the cost function of all these datasets. We also want the expectation value of the average over different noise instances $\mathbb{E}_\epsilon$. The expected generalization error can be found to be (full derivation can be seen in Appendix A):

$$\begin{aligned}\mathbb{E}_{\mathcal{D},\epsilon}[\mathcal{C}(\mathbf{y}, f(\mathbf{X};\hat{\theta}_{\mathcal{D}}))] = &\sum_i (f(\mathbf{x}_i) - \mathbb{E}_{\mathcal{D}}[f(\mathbf{x}_i;\hat{\theta}_{\mathcal{D}})])^2 \\ &+ \sum_i \mathbb{E}_{\mathcal{D}}[\{f(\mathbf{x}_i;\hat{\theta}_{\mathcal{D}}) - \mathbb{E}[f(\mathbf{x}_i;\hat{\theta}_{\mathcal{D}})]\}^2] \\ &+ \sum_i \sigma_\epsilon^2\end{aligned} \tag{6.9}$$

The first term in equation 6.9 is the bias

$$\text{Bias}^2 = \sum_i (f(\mathbf{x}_i) - \mathbb{E}_{\mathcal{D}}[f(\mathbf{x}_i;\hat{\theta}_{\mathcal{D}})])^2, \tag{6.10}$$

and is a measure of the deviation of the expectation value of the model estimator from the true value. This is the best we can do in the infinity limit as we have already discussed. The second term is the variance

$$\text{Var} = \sum_i \mathbb{E}_{\mathcal{D}}[\{f(\mathbf{x}_i;\hat{\theta}_{\mathcal{D}}) - \mathbb{E}[f(\mathbf{x}_i;\hat{\theta}_{\mathcal{D}})]\}^2], \tag{6.11}$$

and measures the fluctuation in the estimator due to finite-sample effects. The last term is just a noise term $\text{Noise} = \sum_i \sigma_\epsilon^2$. By combining these three terms we can decompose the out-of-sample error as

$$E_{\text{out}} = \text{Bias}^2 + \text{Var} + \text{Noise}. \tag{6.12}$$

It is often much simpler to train a very complex model than it is to obtain sufficient good data. Therefore it is normally more useful to use a less complex model with higher bias, since it is less sensitive to noise in the sampling data from having a finite-sized training dataset.

[2]This is used in regression cases. For classification we could use cross-entropy for instance.

6.2.3 Bias-Variance Tradeoff

Before we look into classification, we need to be aware of a few problems with supervised learning. First is the balance of variance and bias. This is called the **bias-variance tradeoff** in statistics and machine learning. We want to minimize both the variance and bias such that our model both works well on unseen data and captures the relations between the features and classes, but when one of them is lowered the other has a tendency to increase. High bias may lead to underfitting between the features and the classes, while high variance may lead to overfitting. When a model is overfit, it is excessively complex and will then model noise in the data as well. Overfit models will then do a great job during fitting, but worse on data outside of the training domain. Underfit models do not have the power to capture important variations in the data. With today's improved machinery, it is often easier to make a model too complex rather than to not.

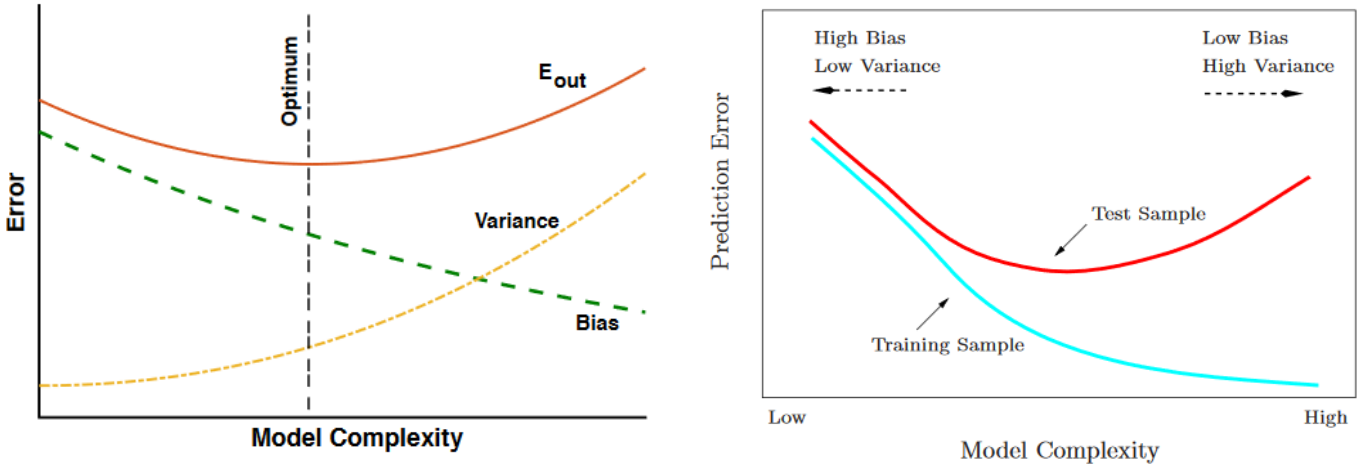
Second is the amount of training data that is available depending on the real function. For a more simple real function, the model does not need that much training data to learn on. While for a more complex[3] real function, the model needs a lot of training data.

Third is the dimensionality of the features. If there are a lot of features with high dimensionality, the model may be confused and cannot separate out the most important features that defines the output. One way to fix this is to manually remove irrelevant features in the data that can confuse the model. The method for doing this is called **dimensionality reduction**, and there are several strategies for doing this.

The fourth and final major concern is noise or incorrect values in the desired output values. This often comes from human error or errors in sensors which can lead to overfitting. This can be fixed by e.g., remove noise training data or use early stopping. There also exists other factors that one need to consider, but these four bias-variance related issues are some of the biggest.

In Figure 6.2 we see illustrations of the bias-variance tradeoff for training error, E_{in}, and test error, E_{out}, as the model complexity increases. In Figure 6.2a we see that as the model complexity increases, the model fits the training data well leading to high variance. For a low complexity model the bias is high. This is exactly as we have already look at above. So we want a model that has a compromise between the variance and the bias, as seen by the optimal line in Figure 6.2a. This optimal line is also where we have a minimum in E_{out}. For the prediction error for test and training samples in Figure 6.2b as function of the model complexity, we see the variance and bias areas for low and high model complexities. From the gap between the two prediction error samples we see the same argument for choosing a optimal compromise between variance and bias. This will lead to a predicted error difference between training and test samples that is not to0 big and not to0 similar to each other. Often we want to use a more biased model with small variance to minimize E_{out} and maximize the predictions.

[3]When we talk about simple and complex real function, we mean the complexity of interactions between the features and the number of features we use to approximate the true function.



(a) (b)

Figure 6.2: Illustrations of the bias-variance tradeoff as function of model complexity. From these two illustrations we see that we want to find the optimal compromise between variance and bias that gives the best model, which does not underfit nor overfit the data. Figures are taken from ref. Hastie et al. [45] and Mehta et al. [46].

6.2.4 Regularization

With increasing data power and amount of data collected, the datasets we can gather can be quite complex. This means that we need better machine learning models. With these better models we can solve more complex problems than before. As we mentioned earlier, this also gives rise to more problems, especially overfitting models. Overfitting is a more common issue than underfitting, since overfitting comes from models fitting functions and training data too well, making it perform worse on unseen (test) data. This is not something we desire to get, since machine learning is all about training model to analyze new data.

Finding good methods to reduce overfitting has been an important aspect in machine learning a long time. That is the reason for developing *regularization* techniques that reduces overfitting problems without significantly worsen the performance on the training data. Regularization techniques try to improve the generalization error of the test set. There are several different regularization methods that can be used, depending on the type of models which are used.

One way is to tune the model complexity to be better at predicting. This is done by introducing a penalty for individual weights, w . There are two types of norms of

regularization that is often used; L1 and L2:

$$L_{1,\text{norm}} = \sum_i |w_i| \tag{6.13}$$

$$L_{2,\text{norm}} = \sum_i ||w_i||^2 \tag{6.14}$$

The L1 penalty will yield sparse feature vectors from the fact that most features weights will be zero. That means that the L1 norm can be seen as a kind of feature selection that removes irrelevant features in datasets with higher dimensionality that would only confuse the model when training. This feature reduction can also be done manually by removing the irrelevant features that makes the model underperform, making it less complex. The L2 norm also acts on the weights of the loss function. These two regularization norms are set in the models as *hyperparameters*.

Other ways to avoid overfitting is to *prune* the models which use *trees*[4], affecting the splitting of trees. *Sampling* and *early stopping* are other ways to control overfitting, by making boosted trees less correlated or stop training when a chosen training metric of a model no longer improves. All these ways to control overfitting are controlled by various input parameters numerically.

6.2.5 Hyperparameters

As we have already mentioned, hyperparameters are something which need to be manually chosen before fitting a model. Hyperparameters help to tune and optimize the models in order to do a better fit of the data, and used to control the algorithms. These hyperparameters have no strict solution and change depending on the dataset we are looking at. The same type of parameter may not have the same value in different models. For a small set of hyperparameters we could simply use trial and error to test the parameters. Most modern models require a lot of different hyperparameters. When there are a lot of parameters to tune, we may want to use some learning algorithm that searches through some given sets of hyperparameter values. An efficient method for doing this is to do a random search that uses the fact that not all hyperparameters are equally important. Searching for parameters are often computationally expensive since they require that the model is re-trained each time we change a configuration of hyperparameters.

During the hyperparameter optimization, we want the test set to be isolated until the model is fully optimized. This is where the validation set becomes useful. The purpose of the validation set is to be used when training the model and optimize the hyperparameters. The first split of the original dataset is into training and test sets. The training set can be further split into a smaller training set and a validation set. This means that we loose some training data which we need to take into consideration. The evaluation of the validation set will not be the exact same as evaluating the test set. The generalization error of the test set will be underestimated by the validation set error since the hyperparameters are trained on the validation set.

[4]We will come back to what this is later.

6.3 Classification

Classification is one of the most used and successful tasks in machine learning. Classification uses algorithms to decide which category the input belongs to. The function that produces an output value can be used to produce a probability distribution over the different outcomes. The simplest and probably most common classification problems are binary outcomes like True or False, Yes or No, Cat or Dog etc, where the outcomes are either the one or the other. When there are more than two outcomes, or classes, we use multiclass classification algorithms. Not all classification algorithms are made to classify instances with more than two outcomes, and cannot be used to classify problems other than binary outcomes. On the other hand, they can be turned into multiclassifiers by using various strategies. There are also other types of classifiers that are similar to multiclass classifiers, like multilabel and multioutput classification. They are similar, but are used in different cases with different outcomes. For example, multiclass classification labels a sample as one class only, meaning that it cannot be classified with two classes. This means that an image of a cat can only be classified as either a "cat" or a "dog" by the algorithm. The other two may categorize the image as both a "cat" and as "small" for instance.

In this thesis, we use multiclass classification to classify different particles in event decay chains produced by colliding protons at the LHC. In this thesis we will test different classification models and algorithms with various values for the hyperparameters for the respective models, to try and optimize and find the most accurate model. We will also study various evaluation metrics used to both find and evaluate the performance of the best model.

6.4 Classification Models

A so-called "hard" classifier will assign each datapoint to a category, while a "soft" classifier will give the probability of a given category. The simplest classification algorithm is the "perceptron". It is given by the same transformation as linear regression with a weight matrix $\mathbf{w}$,

$$\mathbf{y} = \mathbf{X}\mathbf{w}. \tag{6.15}$$

The classes are then determined by the sign of the predictions by using sign functions or boundary thresholds. The perceptron is an example of a "hard" classifier. Sometimes it may be useful to use a "soft" classifier yielding category probabilities instead.

There are a lot of different classification models in machine learning with their own strengths and weaknesses. This is why we in this thesis will test a few different approaches and algorithms to find the best model for the analysis. In this section, we will briefly look at the classification methods we will test in this thesis.

6.4.1 Logistic Regression

A simple "soft" model in statistical analysis for classifying discrete outcomes is logistic regression (LR)[46]. It uses linear regression to fit data and a logistic function[5], usually the Sigmoid function

$$\sigma(s) = \frac{1}{1 + e^{-s}}, \tag{6.16}$$

to predict the outcomes into categories using probabilities. A threshold for the predicted values is chosen which determines which classes the data belongs to. These boundary thresholds can be complex and doesn't have to be linear. The cost function is the usually cross-entropy with added L_1 (eq.6.13) and L_2 (eq.6.14) regularization terms. The cross-entropy is the negative log-likelihood of the prediction being in the dataset. The cross-entropy is derived from the fact that the Maximum Likelihood Estimator (MLE) is the set of parameters that maximize the log-likelihood.

The most basic model is a Binary Logistic Regression that yields two possible outcomes. However, it can be extended to more than two outcomes by using Multinomial Logistic Regression (MLR. Both LR and MLR can be combined with cross-validation, using various optimization solvers supporting the regularization parameters as input.

6.4.2 Multi-Layer Perceptron

A Multi-Layer Perceptron (MLP)[47] is an artificial feed-forward neural network (FFNN) model consisting of interconnected nodes, and is similar to LR in that it has an *input* and an *output layer*, but differs in that between these layers, the MLP can have several non-linear layers called *hidden layers*. In a FFNN the information only goes one way. The inputs are called neurons and are transformed in the hidden layers by a weighted linear summation of the inputs and a non-linear activation function to determine the outputs for each layer. The hidden layers often have some bias to ensure non-zero values. The output layer transforms the values from the last hidden layer into output values. In Figure 6.3a we see how each node in a neural network is connected to all the nodes in the previous layer with a weight value. Then it goes to a non-linear activation function that transforms the node to an output either to a new node in a hidden layer or to the output layer. The nodes will have some bias term individually connected to them. In Figure 6.3b we see a fully connected neural network since all nodes are connected to all nodes in the next layer.

The MLP trains the model using *backpropagation* with initial guesses for the biases and weights. Backpropagation is a method used to optimize the weights and biases to minimize he cost function. The backpropagation iterates backwards from the last layer to the first layer using gradient descent of the weights and biases to start a new feed-forward process from the input layer. This process is repeated until the cost function is sufficiently minimized.

[5]It can also be called an activation function.

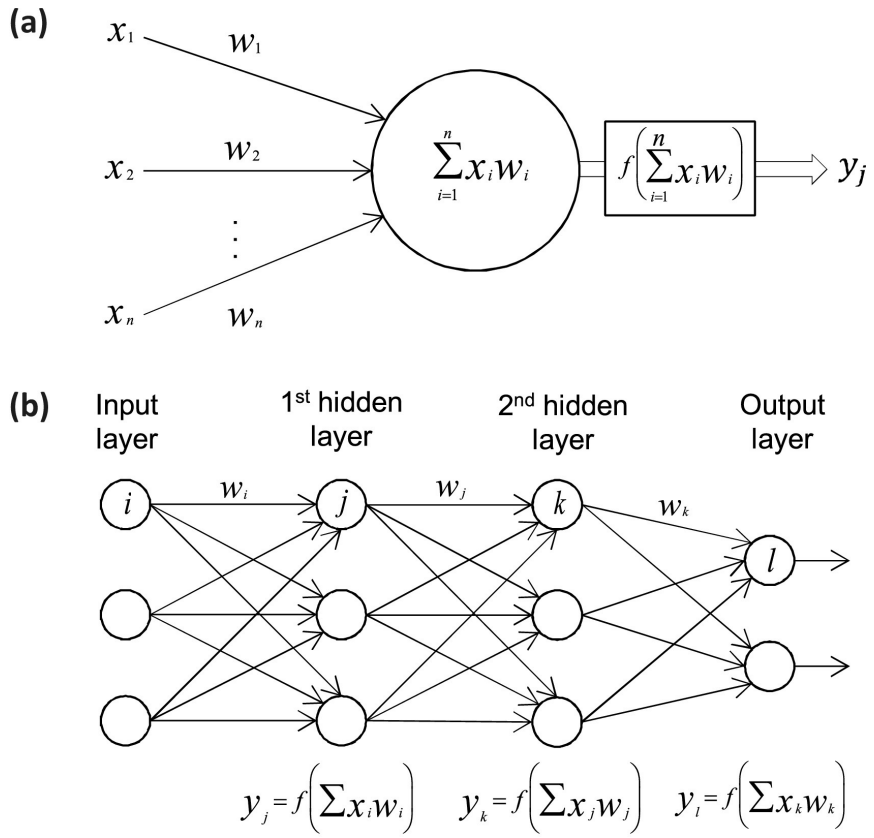


Figure 6.3: (a) Each node in a neural network has some input acted on by some associated weights. Then the weighted inputs are summed together inside the node and passed to a non-linear activation function which transforms it to an output. (b) This is a FFNN/MLP with 3 inputs, 2 hidden layers and 2 classes. All the nodes in one layer is connected to all the nodes in the next layer. This is then a fully connected neural network. Figure is taken from ref. Vieira et al. [48].

Typical choices of activation functions are the hyperbolic tangent function, the sigmoid and the rectified linear unit function (ReLU). The choice of cost function also needs to be considered. The MLP library in *Scikit-Learn* only supports the cross-entropy loss function as the cost function.

Neural networks typically have a large amount of parameters which often leads to overfitting. That is why we add a L_2 regularization penalty to the weights. This hyperparameter have to be tuned. Another hyperparameter that is needed is the *learning rate*. This parameter is used to control the step length in the optimization of the cost function with a gradient descent method. In neural networks the weights and biases are the parameters to be adjusted, while it can be different in other models. There are several gradient descent methods, e.g. the stochastic gradient descent with minibathces, which can be used to avoid interpreting a local minimum as a global minimum. A more modern method is the *adam* solver proposed by Kingma and Ba [49]. It is a stochastic gradient-based optimizer

which combines an adaptive learning rate[6] with other functions, and thus adds a few more hyperparameters to be tuned, i.e. β_1, β_2 and ϵ.

For multiclass classification, the softmax function is used as the last hidden layer activation function. It normalizes the output of the network into a probability distribution over the predicted output classes.

6.4.3 Decision Tree

Decision Trees (DTs)[50] tries to learn simple decision rules from the data features by constructing tree-like models to predict the target values. The models simply break down a complex decision into several simpler decisions. Each tree starts of with a single *root* node containing all the class labels. The root node then splits into several smaller *internal* nodes, which then splits into *leaf* nodes in the end representing the class targets. The splits are decided by some chosen *criterion* function that uses a certain strategy to do the splits. The root node is chosen as the feature with the highest information gain value by the criterion function. The path from the root node to an internal node or a leaf is always unique, and the leaves do not have any descendants.

The DT uses a cost function to determine the the most homogeneous branch when splitting. The stopping point for splitting is something we can set as an input parameter to the model by choosing a maximum depth of the tree from the root to the leaves, or by setting the maximum number of leaves at the end. Other parameters for controlling the size and splitting of the tree should be considered since the DT is prone to overfitting with many features. This can be fixed by pruning the tree, i.e. to remove nodes with low importance features, use dimensionality reduction with e.g. principal component analysis (PCA)[51] or decrease some of the controlling parameters.

There are a few different DT algorithms to generate the optimal trees. The algorithm that is implemented in Scikit-Learn is an optimized version of the Classification and Regression Trees (CART) algorithm that construct binary trees from the features and thresholds for giving the highest information gain at each node. The Scikit-Learn DT classifier automatically supports multiclass classification.

6.4.4 Random Forest

Another classification ensemble method is the Random Forest (RnF)[52] algorithm. It produces a number of DT classifiers on bootstrapped training samples with a low correlation to each other, and uses their average like the bagging method. This improves the accuracy score and helps control overfitting. One can also choose to bootstrap samples.

Like with the DT algorithm, we can control the size and splitting of the tree. The DT and RnF algorithms are very similar and have many of the same input parameters and same procedure for building the trees. The main difference is that with the RnF algorithm, we produce many trees with sources of randomness. This randomness is very important in

[6] It adjusts the learning rate as it iterates towards the minimum.

that it decreases the variance when combining and taking the average of many trees, and can cancel out some prediction errors. This normally yields a better model.

6.4.5 AdaBoost

Instead of using average ensembles where we use many independent bootstrap samples, we can use something called *boosting*. This is a type of method which keeps the weight for each iteration, and the base estimators are built sequentially. The boosting model then builds a combined estimator that reduces the bias and the variance. The result is to get a powerful ensemble from several weaker models combined.

One such boosting method is the AdaBoost classifier[53][54]. The AdaBoost uses adaptive boosting and weaker classifiers as estimators to sequentially combine them into a single better classifier with a weighted majority. It will fit weaker classifiers sequentially such that the next classifier will have a different weight than the previous to adjust for incorrect classification in the previous. Data which are difficult to predict will then have an increasing influence since the next classifier will learn from the mistakes of the previous weaker classifier. The final prediction is the result of a weighted majority vote of the combination of the weaker classifier predictions. A weaker classifier can then be boosted to a stronger classifier that is more accurate.

The AdaBoost algorithm in Scikit-Learn takes a weaker classification algorithm as input together with the maximum number of estimators to be boosted before stopping. If the model is to be perfectly fit, the executing will also be stopped. It also takes a learning rate parameter for the shrinking of the classifiers. The AdaBoost algorithm can naturally detect and adapt to a multiclass problem.

6.4.6 Gradient Boosting

Gradient Boosting Decision Tree (GBDT)[55] is another boosting method like AdaBoost. The GBDT is an additive model that tries to identify the shortcomings of the weak classifiers. While AdaBoost uses high weight data points, the GBDT uses the same for gradients in the loss function. This allows the cost function to become better for optimizing the fitting. The K number of regression trees[7] at each stage are fit on the negative gradient of the binomial (binary class) or multinomial (multiclass) deviance loss function.

The algorithm is well suited for both binary and multiclass classification, and takes the maximum number of estimators and the learning rate as input parameters. Since it is a boosted tree method, it can also take the maximum depth of the trees and maximum number of leaves as inputs. It is also quite robust against overfitting. In a multiclass problem, the algorithm will create K trees for each iteration when we have K classes. The loss function for multiclass also have to be "deviance" to give probabilistic outputs (similar to LR).

When dealing with larger datasets (n_samples>10 000) or a large number of classes, a histogram-based gradient estimator can be more useful. Scikit-learn has an experimental

[7]For a binary classification case, only a single tree is fit.

implementation of GBDTs called **HistGradientBoostingClassifier** which is inspired by Ke et al. [56] on a **LightGBM** algorithm. This estimator can be orders of magnitude faster than the original GBDT estimator. To reduce the computation time and number of splitting points, the algorithm bins the input samples into integer-valued bins. They share most of the same parameter inputs which controls the models, except that the histogram-based estimator gets a parameter for controlling the number of bins. This can act as another regularization parameter.

6.4.7 Extreme Gradient Boosting

Another highly efficient, flexible and portable tree boosting method is the Extreme Gradient Boosting (XGBoost)[57]. It is a scalable end-to-end tree boosting system using an optimized distributed gradient boosting algorithm and provides fast and accurate parallel tree boosting. It is one of the most used and highly recognized machine learning algorithms today together with deep neural networks. The XGBoost algorithm won the Kaggle Higgs ML challenge in 2014[58]. One of the most important aspects of the XGBoost is its scalability, making it several times faster than other algorithms combined with parallelization.

The XGBoost algorithm uses the GBDT framework as its core. It looks at distributions of the features for all data points in a leaf to build trees using potential loss for the possible splits to make a new branch. This decreases the space of possible feature splits search. The algorithm chooses features and split-points based on the criteria to maximize the gain. The splits are binary such that it splits according to if a value is bigger or lower than a threshold set by the algorithm. The gain is different depending on the type of loss function which is used. With a small dataset, the XGBoost algorithm tries all split points gained by the data values for each feature. The feature and threshold combination with the highest gain is then chosen. For a larger dataset, the algorithm uses fewer candidate splits given by the quantiles of the data.

Since XGBoost is more complex than other algorithms, it also requires more parameters to be tuned to control the model properly. The parameters can be sorted into *general parameters* for choosing the booster method, *booster parameters* which are dependent on the boosting method and *task parameters* which specify learning task parameters and learning objectives. We are using a tree booster which has many of the same tree boosting parameters as the DT and RnF algorithms, i.e. regularization terms, hyperparameters for tree controlling, pruning and others. The task parameters include the type and size of the classes we have, e.g. multiclass classification, and types of evaluation metrics to use.

6.4.8 Light Gradient Boosting Machine

Light Gradient Boosting Machine (LGBM)[56] is a distributed gradient boosting framework for machine learning. It is similar to the XGBoost algorithm, but made to be faster, around 7 times faster, with higher efficiency, lower memory usage and better accuracy. This is a huge advantage when dealing with larger datasets. The LGBM algorithm uses a gradient based one-side sampling and exclusive feature bundling for filtering the data samples to

find the split value in the trees, while the XGBoost uses a histogram based algorithm to find the best splits. This means that the LGBM algorithm will keep features with higher absolute values, regarding information gain, than a pre-defined threshold and drop the features with small absolute values. This will improve the accuracy. The features that rarely have non-zero values simultaneously will be combined into a single feature, to reduce the number of features in the dataset. XGBoost and LGBM have very similar input parameters.

6.4.9 Multiclass Classification Models

To do multiclass classification, there are several existing techniques. We will look more into two of those techniques[8]; transformation to binary and extension from binary. These are all meta-estimators. This means that they all need a base estimator, most often a binary classifier, which is extended to do multiclass classification when they are implemented in the constructors.

The extension from binary technique is rather trivial. We simply use already existing binary classifiers and modify them to do multiclass classification. Not all binary classifiers can be extended to multiple classes. The classification models we have looked at, this far, can either do this automatically, or have input parameters and constraints in the models to tell the models to do multiclass classification.

Transformation to binary reduces our multiclass problem down to several binary classification problems. This technique can also be split into more strategies, which we will look more into.

One-Vs-Rest Classifier

The first strategy is the one-vs-rest (OvR) classifier. Each class in this model has its own classifier which does the fitting, and the classifier fits the single class against the rest of the classes. This means we only need n classifiers for the n classes. This also improves interpretability, since we can get information about a specific class by looking at its classifier.

The OvR takes a input a binary classifier along with samples and targets and outputs a list of the classifiers for each class. When doing predictions, it uses all the classifiers on unseen data and picks the class with the highest confidence score.

One-Vs-One Classifier

The second strategy is the one-vs-one (OvO) classifier. This takes one classifier and a pair of classes at a time. For each pair of classes, the classifier trains on data containing these classes and learns to distinguish them. This happens between all the classes. It then uses a voting scheme to select the class with the most votes. For n number of classes in the multiclass problem, the OvO trains $n(n-1)/2$ binary classifiers. All the classifiers that

[8]There is also a third technique, hierarchical classification, that we will not cover.

are trained will be applied when doing the prediction on unseen data, and the one with the highest number of predictions will be predicted by the combination of classifiers.

This method is slower than the OvR since it has a $\mathcal{O}(n^2)$ complexity. Both the OvO and OvR methods suffer from the fact that there may be regions where the input space can get the same number of votes.

6.5 Evaluation Metrics

To evaluate the performance of the classification models properly and decide which model best fits the data, we need to have some evaluation metrics. In this section we will take a look at the evaluation metrics used for the classification[9].

6.5.1 Mutual Information

To look closer at the correlations in the dataset, we can use the entropy and information gain. The entropy can be calculated using the probability $P(j)$ of a value j occurring, where j is a value which a feature group x_i can take;

$$H(x_i) = -\sum_{j \in x_i} P(j) \log_2 P(j) \tag{6.17}$$

With a given target $\mathbf{y}$, we can calculate the conditional entropy of a feature x_i:

$$H(x_i|\mathbf{y}) = -\sum_{y \in \mathbf{y}} P(y) \sum_{j \in x_i} P(j|y) \log_2 P(j|y) \tag{6.18}$$

Now we compute the information gain, or *mutual information* in the context of variable selection, for a given feature as the difference between these two entropies:

$$I(x_i : \mathbf{y}) = H(x_i) - H(x_i|\mathbf{y}) \tag{6.19}$$

With the information gain we get a measure of the correlation between a feature and the target, which shows dependencies between features and the amount of information that one feature provides about others.

6.5.2 Accuracy Score

To measure the performance of the models, we use the the accuracy score for classification. This is a measure on how well the models can predict the classes. It is defined as the number of correct predictions divided by the total number of predictions, giving a value between 0 and 1.

$$\text{Accuracy} = \frac{\sum_{i=1}^{n} I(\tilde{y}_i = y_i)}{n}, \tag{6.20}$$

[9] See Scikit-Learn[59] for more details on metrics.

where $\tilde{y}_i$ is the predicted target by the model, y_i is the actual class target, n is the total number of predictions and I is and indicator function

$$I = \begin{cases} 1, & \text{if } \tilde{y}_i = y_i \\ 0, & \text{if } \tilde{y}_i \neq y_i \end{cases} \tag{6.21}$$

When the model prediction fits the data perfectly we get an optimal score of 1.

The accuracy score can be computed for all datasets, i.e. training validation and test sets. If there is a big difference between the accuracy score for either validation and training or test and training, we might under- or overfit the data. When the training score is much better, we most likely overfit the data.

Another way to balance out the accuracy scores is to use *cross-validation*. It's a very useful technique against overfitting, and can be used to tune hyperparameters. There are several cross-validation techniques, but the main idea of cross-validation is to divide samples into subsets. The cross-validation will do the analysis on one subset and compute the accuracy on that subset. Then it will do another analysis with another subset and compute the accuracy again. After many iterations, dividing the data into subsets and computing several accuracy scores, the average score is used as an estimate of the model performance.

6.5.3 Cohen Kappa Score

Another scoring statistic is the Cohen Kappa Score (CKS)[60]. The CKS accounts for uncertainties in the predictions, comparing a random classifier against a more accurate and tuned classifier. The CKS is calculated by using the rate of agreement for random guessing, p_e, and the rate of agreement for the actual prediction, p_a. The CKS ranges from -1 to 1, where 1 is the optimal score representing perfect agreement, 0 represents agreement that can be expected by random guess and -1 represents no agreement, and is calculated as

$$\kappa = \frac{p_a - p_e}{1 - p_e} \tag{6.22}$$

6.5.4 Error Evaluation

We will use several different error metrics to get a good overall error estimate of the classification models. These will also help to discover any over- or underfitting of the data.

Error Rate

With the accuracy score, we can compute the *error rate*. The error rate is defined as the fraction of misclassifications:

$$\text{error} = 1 - \text{accuracy} \tag{6.23}$$

This is an often used metric in classification. Both the error and the accuracy score can be computed in multiclass classification cases.

Log Loss

Instead of using discrete predictions, we can evaluate probability outputs of classifiers. We can use the *log loss* function, also called the cross-entropy or logistic regression loss, to evaluate the probabilities. When dealing with a binary case with a probability estimate $p = P(y = 1)$, the log loss is defined as the negative log-likelihood given a true output for each sample. It is computed as

$$L_{\log} = -\log P(y|p) = -(y \log(p) + (1 - y) \log(1 - p)). \tag{6.24}$$

For a multiclass case, the log loss is taken over a whole set of size n with K labels, a binary indicator matrix $\mathbf{Y}$ and a matrix $\mathbf{Pr}$ of probability estimates as

$$L_{\log}(\mathbf{Y}, \mathbf{Pr}) = -\log P(\mathbf{Y}, \mathbf{Pr}) = -\frac{1}{n} \sum_{i=1}^{n} \sum_{k=1}^{K} y_{i,k} \log p_{i,k}. \tag{6.25}$$

Variance

Previously in section 6.2.2, we defined the variance and the bias of a model. These two are used to check for possible under- and overfitting. The variance is a measure of how far the spread of our predictions are from their average values. Given the predictions, $\tilde{\mathbf{y}}$, of a model, the variance is calculated as

$$\text{Var}(\tilde{\mathbf{y}}) = \frac{1}{n} \sum_{i=1}^{n} (\tilde{y}_i - \frac{1}{n} \sum_{j=1}^{n} \tilde{y}_j)^2. \tag{6.26}$$

Bias

The bias error is a measure of the difference between the true values, $\mathbf{y}$, and the average of the predicted values. To get the out-of-sample error in equation 6.12. The bias squared can be calculated as

$$\text{Bias}^2(\mathbf{y}, \tilde{\mathbf{y}}) = \frac{1}{n} \sum_{i=1}^{n} (y_i - \frac{1}{n} \sum_{j=1}^{n} \tilde{y}_j)^2. \tag{6.27}$$

6.5.5 Classification Report

With Scikit-Learn, we can easily build what is called a *classification report*. This is a text report containing some useful classification metrics using the true targets and predictions of the model.

First we will look at some useful prediction results used to compute some of the report metrics:

1. Positive (P) - The observation is positive.
2. Negative (N) - The observation is negative.

3. True Positive (TP) - Observation is positive, and the prediction is positive.

4. True Negative (TN) - Observation is negative, and the prediction is negative.

5. False Positive (FP) - Observation is negative, but the prediction is positive.

6. False Negative (FN) - Observation is positive, but the prediction is negative.

With the last four outcomes above (3-6), we can compute some useful metrics in the report:

Precision - The fraction of a sample classified correctly as positive of all positive predicted samples by the model:

$$\frac{TP}{TP + FP}$$

Recall - The fraction of a sample classified correctly as positive of all positive observations (true positive rate):

$$\frac{TP}{TP + FN}$$

The recall of the positive class is also called the sensitivity. The recall of the negative class (true negative rate) is called the specificity.

F1-score - A weighted average of the precision and recall:

$$2 \times \frac{\text{Precision} \times \text{Recall}}{\text{Precision} + \text{Recall}}$$

In the multiclass case, these metrics are computed for each class independently.
The classification report also includes for various classification cases:

Support - The number of true classes in the dataset for each class.

Accuracy - The accuracy score of the model (binary case).

Macro avg - Average of the unweighted mean for each class.

Micro avg - Average of the total true positives, false negatives and false positives (multiclass or multilabel cases).

Weighted avg - Average of the support-weighted mean for each class.

Sample avg - Average of samples (multilabel case).

6.5.6 Confusion Matrix

With the four outcomes in the classification report (3-6), we compute a *confusion matrix*. For a binary case, the confusion matrix looks like Figure 6.4. Here we see the predictions versus the true values. It gives a better understanding of the accuracy of a classification model. The accuracy score shows the overall accuracy, whereas the confusion matrix shows the predictions and accuracy of each class. It is easily extended for multiclass classification as a matrix with dimension $k \times k$ for k classes. When the confusion matrix is normalized the total values of the rows are equal to 1. In the optimal case with all predictions correctly guessed, we should have 1's along the diagonal and 0 elsewhere.

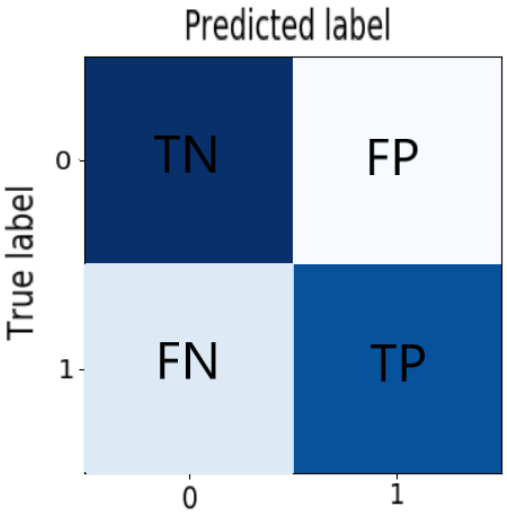


Figure 6.4: The confusion matrix is used to evaluate the accuracy of a classification model by using the four true and false observation and prediction outcomes (TP, TN, FP, FN. See sect. 6.5.5.).

6.5.7 Precision-Recall Curve

Scikit-Learn provides a useful function for plotting precision versus recall. In Figure 6.5 we see an example of how a precision-recall curve can look like in a multiclass case with 10 classes. This lets us see how the precision and recall behaves for different thresholds. A large AUC is the result of both high precision and high recall, which is preferable. The range of values will be between 0 and 1, as for accuracy. When the area under the curve (AUC) of a class is close to 1, the classification model can predict this class with a good accuracy. For a multiclass-case, the precision and recall are computed for each class as binary cases. A large area for each class is the optimal case here as well.

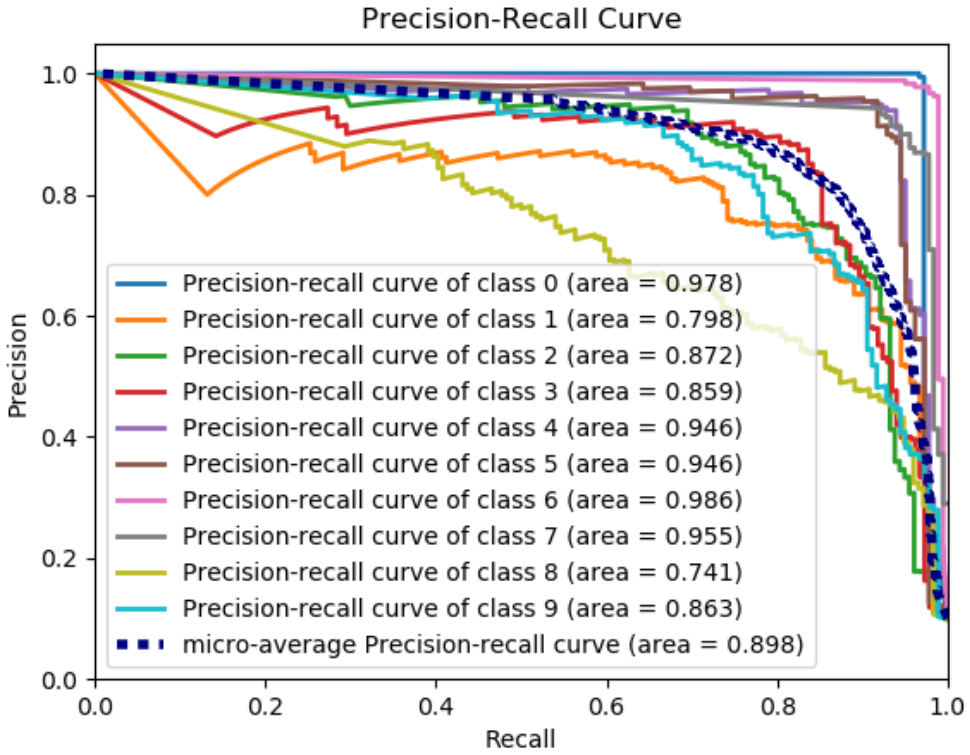


Figure 6.5: Example of a precision-recall curve for a multiclass classification case with 10 classes and a micro-average curve plotted. Most of the classes have a large AUC, showing that the classifier can predict these classes with good accuracy. Credits Scikit-Learn [59].

6.5.8 Balanced Accuracy

If we are dealing with imbalanced datasets, we can use *balanced accuracy*. It uses a macro-average of the recall for each class. When we have a balanced dataset, this just becomes the standard classification accuracy. It is computed as the mean of the sensitivity and the specificity:

$$\text{Balanced-accuracy} = \frac{1}{2}\left(\frac{TP}{TP+FN} + \frac{TN}{TN+FP}\right) \tag{6.28}$$

The balanced accuracy ranges from 0 to 1.

6.5.9 ROC Curve

The Receiver Operating Characteristic (ROC) curve utilizes the AUC to summarize the overall performance of classification models. An example of a ROC curve plot can be seen in Figure 6.6 with a multiclass case with 10 classes. This results in 10 ROC curves, a random model curve and two different average curves, as seen in the figure. The ROC curve function in Scikit-Learn plots the sensitivity versus the specificity for a model. A

totally random model would result in an AUC of 0.5, showing as the straight dashed line from the left bottom corner to the right top corner in the figure. The optimal model would show an infinitely quick incline in the ROC curve at the beginning, before flattening out with AUC close to 1. A good classifier would typically have an AUC larger than 0.8.

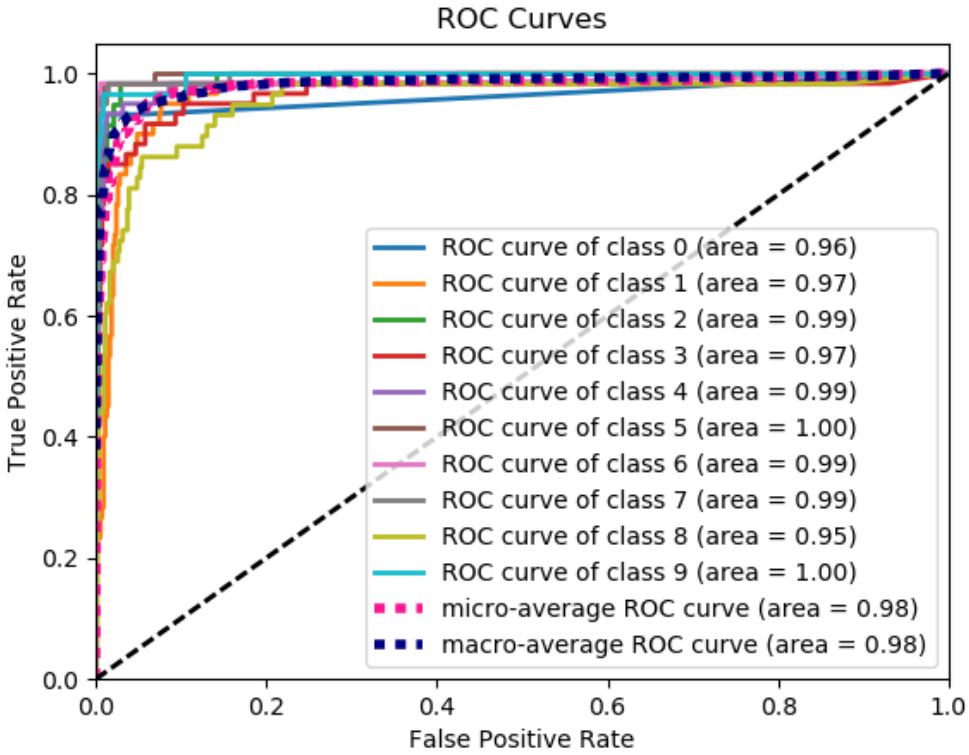


Figure 6.6: Example of a ROC curve for a multiclass classification case with 10 classes, a micro-average curve, a macro-average curve and a random model (black dashed line) plotted. All classes and averages have large areas under the curves, showing that the classifier can predict the classes with good accuracy. Credits Scikit-Learn [59].

Part II

Implementation

Chapter 7

Preparing for Machine Learning

In this part we will look at the framework of the classification exploiting a range of different models. First we will look at the data we will use and how it is made and converted to fit our purposes. The data is already produced beforehand as ROOT files (more in sect. 7.2) and will be converted into dataframes with Python where we will make new features as well as the target values. The data will then be analyzed and preprocessed (sect. 8.1) with different methods before it is split into training, validation and test sets. We will then go through some of the tuning which is done with the models on the validation set using the evaluation metrics from section 6.5, before we apply the best fit model on new similar data. The best model will be used to classify the vertices of the leptons in background and signal data. The results will be presented in part III and discussed in part IV.

Python is used for easy implementation of machine learning libraries with Scikit-Learn[59] and for plotting, using the Matplotlib library. In the following section, we will give a brief presentation of the most important Python libraries we use in our code.

7.1 Python Libraries

Many of the libraries we use require other libraries to be installed, but they do not have to be explicitly imported in the code itself. The code and necessary software requirements are found in the GitHub repository[1]. It contains explanations on how to setup and run the code.

When we present code snippets in this thesis, we will leave out some parts, noted by "\\...", since it will only be used for visualization of the code. The full source codes can be found in the GitHub repository.

- **NumPy**: NumPy, or Numerical Python, is one of the most used packages in Python. It handles arrays, matrices, has functions for working with high-level mathematics, can dump data to files and more.

[1] https://github.com/krilangs/ComPhys—Master

- **Pandas**: Pandas is a powerful and easy Python made library for handling data manipulation and analysis. This library is very useful with machine learning for handling the data for visualization, since it creates data structures that are flexible, efficient, customizable and easy to use and read.
- **Matplotlib**: Matplotlib is a plotting library for Python and NumPy which creates graphs and visualizations.
- **Seaborn**: Seaborn is a more high-level visualization library, and is based upon Matplotlib. It is most used for statistical graphics to understand data better, and is closely connected to pandas.
- **ROOT**: ROOT, or PyROOT, is ROOT's Python C++ bindings. It lets us use ROOT in Python. This is very much used in particle physics. This also lets us use Python libraries like NumPy and Pandas combined with ROOT.
- **Uproot**: Uproot is a library for converting ROOT files to e.g. dataframes by combining Uproot and Pandas.
- **Scikit-Learn**: Scikit-Learn is a library used for data analysis and machine learning in Python. It contains a lot of useful tools for statistical modeling and machine learning. Most of the classification models we use, are imported from this library.
- **Imblearn**: Imblearn, or Imbalanced-learn, is used with Scikit-Learn to handle imbalanced datasets in machine learning.
- **XGBoost**: XGBoost is a library that provides a powerful, scalable and distributed gradient boosting framework for machine learning.
- **LightGBM**: LightGBM is another distributed gradient boosting library for machine learning. It is made to be efficient and faster than XGBoost for larger datasets.

7.2 Data

The inputs we are using in this thesis consists of Monte Carlo (MC) simulated background data and neutrino signals as well as data from p-p collisions at $\sqrt{s} = 13$ TeV. The following ATLAS data processing chain information is taken from Catmore [61]. The MC and data go through the same chain of Reconstruction, Derivation and Analysis, shown in Figure 7.1. For the MC simulations (right side of Fig. 7.1) we have an additional step of Generation, Simulation and Digitization before Reconstruction. The (event) Generation step is a simulation of the interaction between quarks and gluons in proton-proton collisions, parton showering and hadronization and subsequent decays into stable particles. Next step is detector Simulation which simulate how the particles interact with the detector. The Digitization step turns simulated energy deposits into detector responses looking like real raw data. The MC are from here on treated similarly as real data. In the Reconstruction

step histograms of the raw data are made. The Derivation step reduces the size of the datasets from PB to TB, before the datasets will be even further reduced to MB-GB sized ROOT Ntuples.

In this thesis the background data are made to best represent all possible production-mechanisms that may give a three lepton final state with a given transverse momentum plus MET.

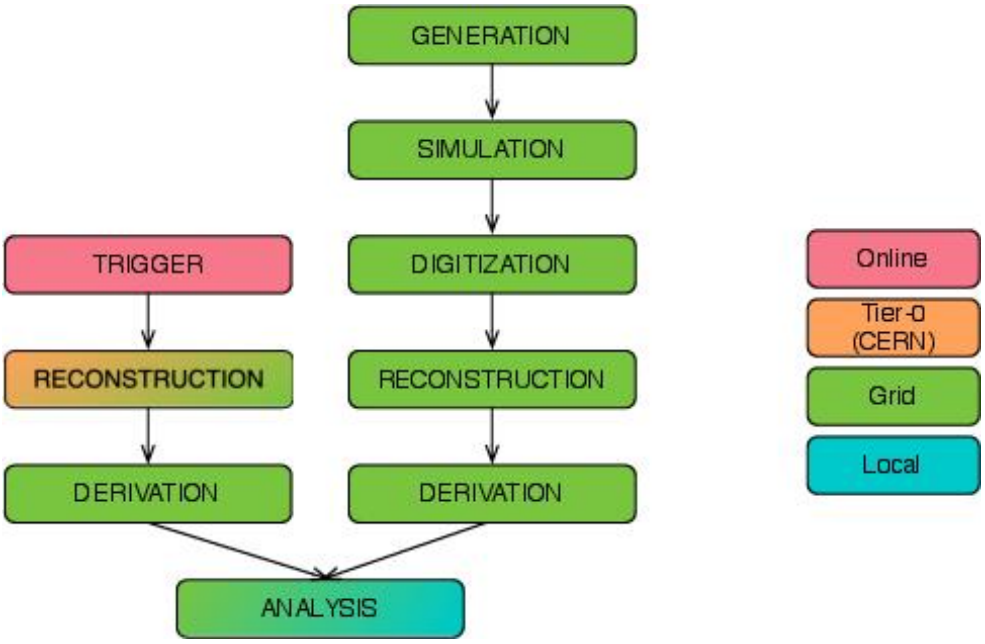


Figure 7.1: The data flow for producing data and MC/background simulation. MC start with Generation (right side) going through all the steps ending up with Analysis. Data start with a Trigger (left side) that picks out interesting events and information from a detector, e.g. the ATLAS detector. Credit: Catmore [61].

The data (left side of Fig. 7.1) we use is proton-proton collisions at $\sqrt{13} = 13$ TeV from the LHC from 2018. For the data we have a Trigger step that picks out interesting events. What is regarded as interesting is defined by the physicists and the collaboration depending on the analysis they want to perform. In our case we trigger on three leptons with a given transverse momentum. This reduces the rate of writing data to disk.

The files we use for the MC training in this thesis have only gone through the Generation step. We are then in full control over the truth origin, type (electron, muon, tau, quark etc.), p_T etc. of the particles. The simulation is done using MadGraph[62] with Pythia[63] for showering/hadronization. The features we plot with MC and data are after the Analysis step in Figure 7.1 with three leptons of good quality. For the ML we want to use the ML models on simulated backgrounds[2] and (neutrino) signal after the Analysis step. Now we no longer have the "truth" information about the particle types, the vertex it comes from etc. E.g. instead of a "true" electron we now have an electron object classified as an electron

[2]Following the SM processes.

after it has passed a set of detector cuts. In some cases we get an electron is classified as a muon or a jet, causing some inefficiency with respect to the "true" distributions. Moreover, the energy and momentum resolution of the detector will smear the measurement of these quantities. In the case of the neutrino we know the full 4-vector at the truth level, while after the Reconstruction step we only see it as missing transverse energy (MET) with no information of its longitudinal component. We will just say we have three leptons and a neutrino/MET in each event to distinguish easier in this thesis, even though the neutrino is a lepton.

7.3 Feature Validation

We will now take a look at the features in the MC and signal Ntuples we will be using in this thesis. The Ntuples all contain the same features with separate variables for the three leptons, e.g. Charge, Flavor, Pt, Eta and Phi. We also use a feature to define tw0 cuts on the events that only contain 3 leptons, *nLep_base*==3 and *nLep_signal*==3. Since we do not know the truth information of the leptons in these Ntuples, the leptons in each event are arranged such that lepton 1 of an event has the highest momentum, lepton 2 has the second highest and so on. This does not mean that lepton 1 actually comes from vertex 1. For the neutrino we only have the met_Et and met_Phi features. In the Ntuple distributions we merge the *higgs* and *topOther* backgrounds into *ttbar+X*, *diboson3L* is named *WZ* and *diboson4L* is named *ZZ*.

In Figure 7.2 we see the data, MC and signal plots for the Flavor and Charge of the three leptons. The Flavor figures show either 1 or 2 as lepton flavor values corresponding to electron and muon, respectively. The Charge figures show that the leptons can either have -1 or 1 as values.

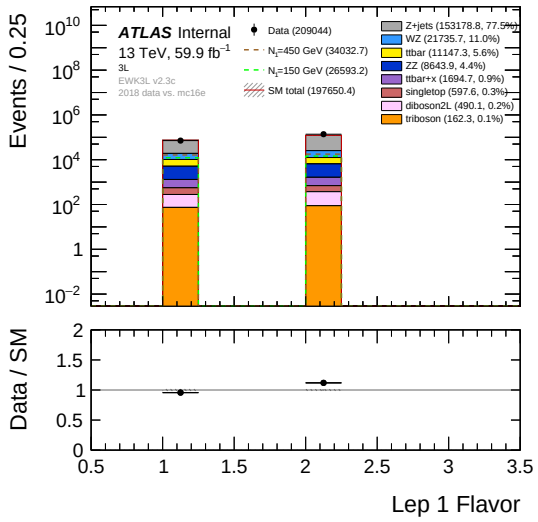
In Figure 7.3 we see the data, MC and signal plots of Eta and Phi for the three leptons. The eta values are between -2.5 and 2.5, while the phi values are between $-\pi$ and π. All distributions show very little dependence on phi and eta.

In Figure 7.4 we see the Pt of the three leptons. As expected, since the arrangement of the leptons are after p_T lepton 1 has a long tail towards high p_T reaching beyond 800 GeV. Lepton 2 has only a few events around 800 GeV, while p_T of lepton 3 reaches only to about 500 GeV. All distributions have most events at low p_T and decreasing as p_T increases. The smaller signal with mass $N_1 = 150$ GeV reaches its peak much earlier than the 450 GeV signal in all p_T figures. This makes sense since the signal with higher mass neutrino will have more events with high energy leptons.

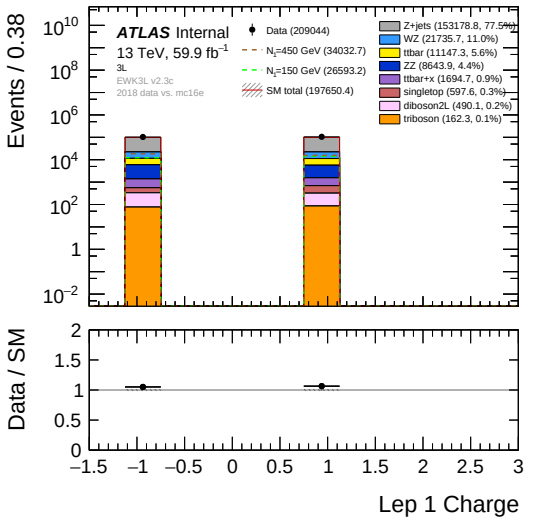
Met_Et and met_Phi are seen in Figure 7.5. The Phi feature is very similar to the Phi feature for the three lepton, where the number of events is more or less equal for all values of ϕ. The transverse energy, or E_T^{miss}, reaches to 600 GeV. Like for the p_T of the leptons, the smaller signal reaches its peak much earlier and has fewer events with higher p_T.

A thing to notice from these figures is the remarkably good compliance between data and MC. The optimal case would have all the dots in the lower frames on the 1-marked line for Data/SM, and our data and MC looks to give a ver ygood agreement. The Data/SM

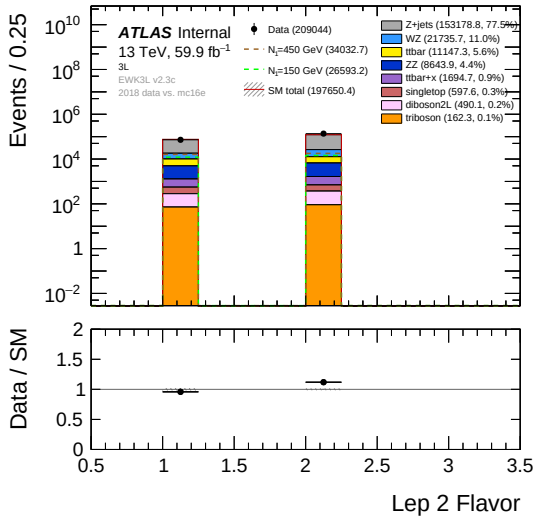
(lower) plots show how well the simulated MC fit with the data for different features values.



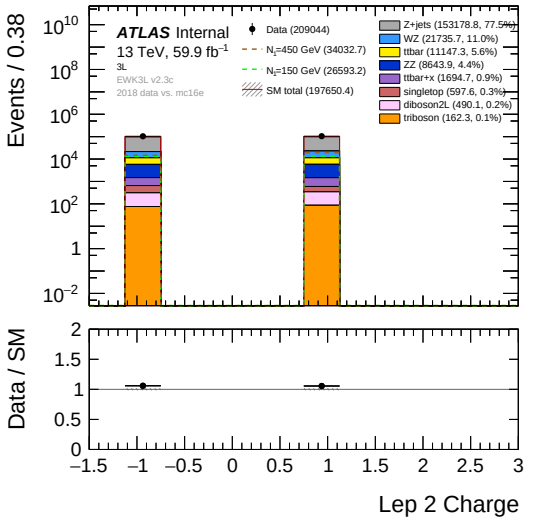
(a) Flavor lepton 1



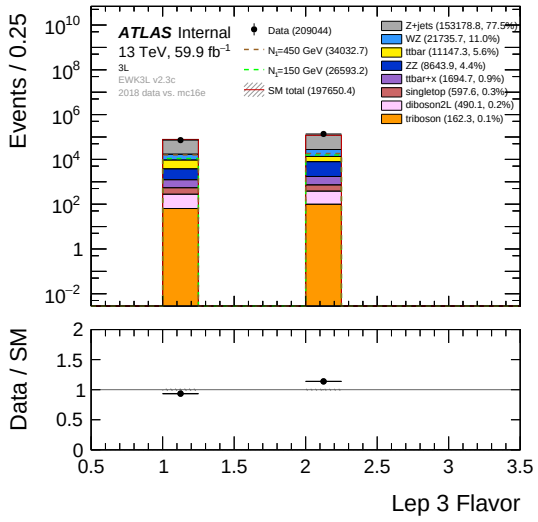
(b) Charge lepton 1



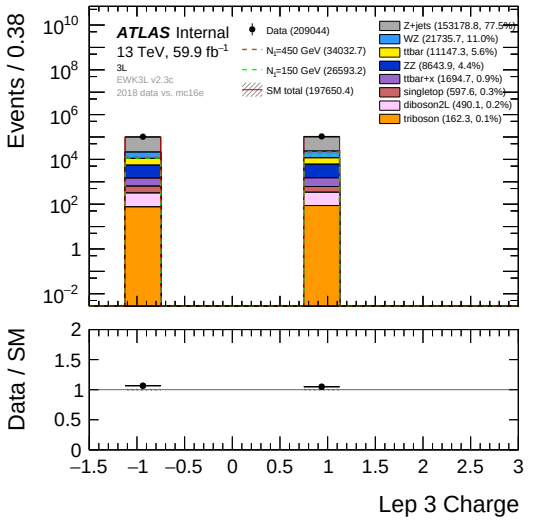
(c) Flavor lepton 2



(d) Charge lepton 2

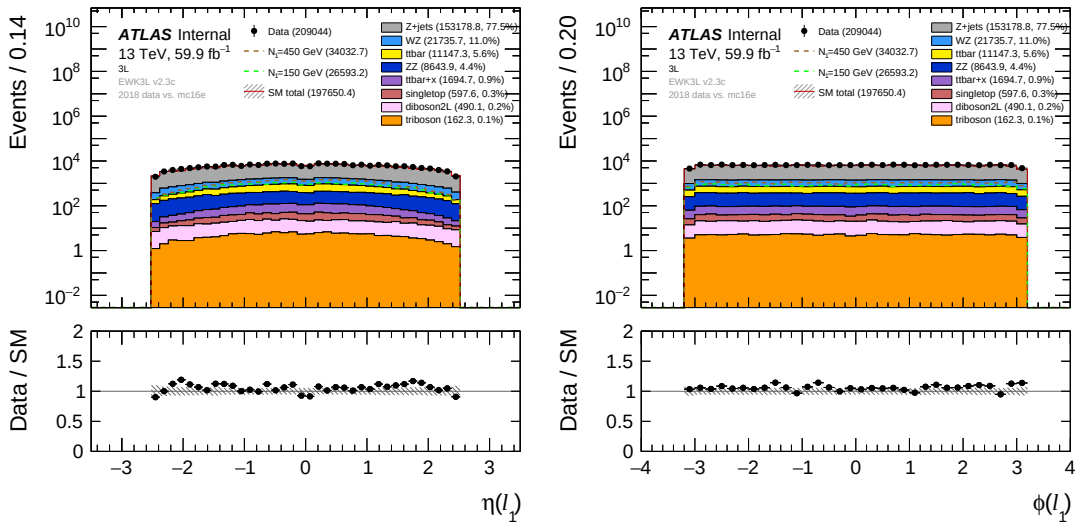


(e) Flavor lepton 3



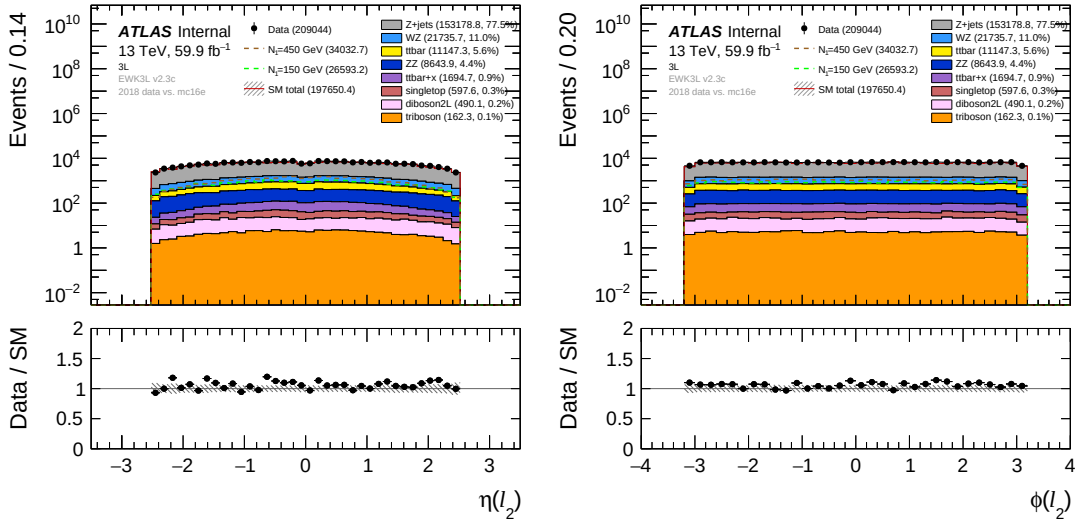
(f) Charge lepton 3

Figure 7.2: Plot of the Flavor and Charge features for the three leptons with data, MC and two neutrino signals. The flavor of the leptons can either be 1 (an electron) or 2 (a muon), while the charge of the leptons can be either -1 or 1.



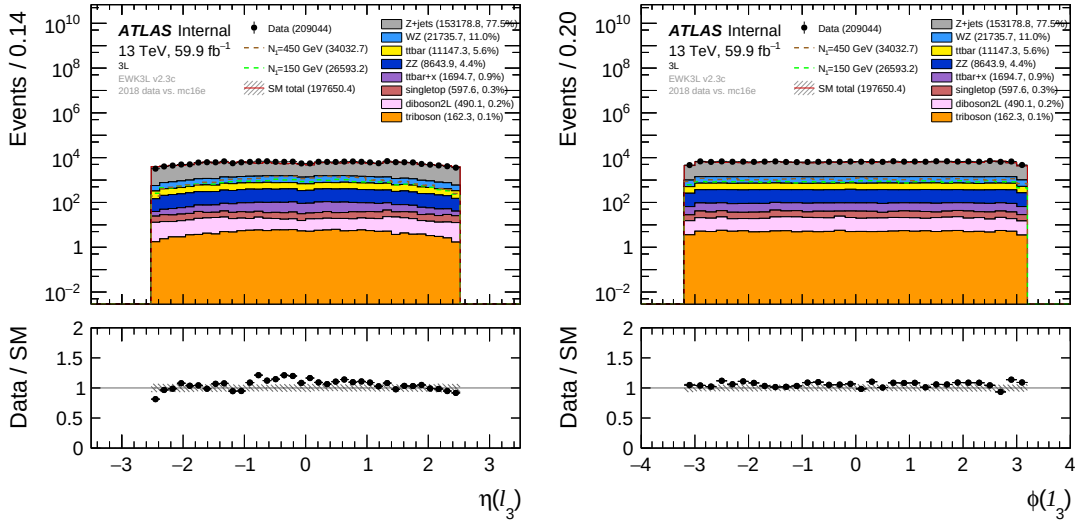
(a) Eta lepton 1

(b) Phi lepton 1



(c) Eta lepton 2

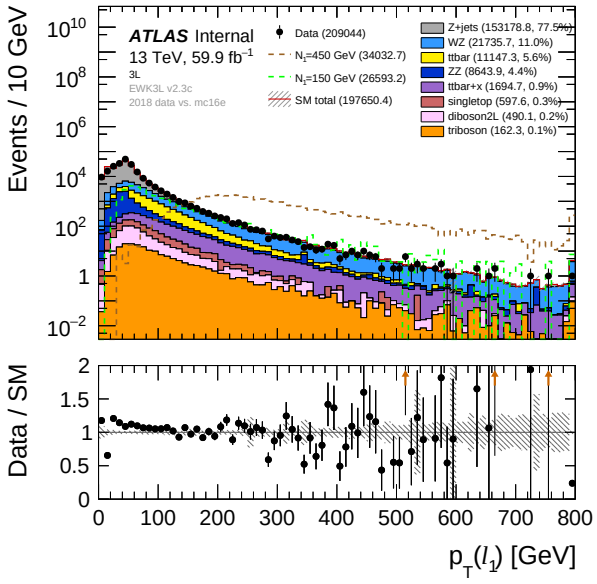
(d) Phi lepton 2



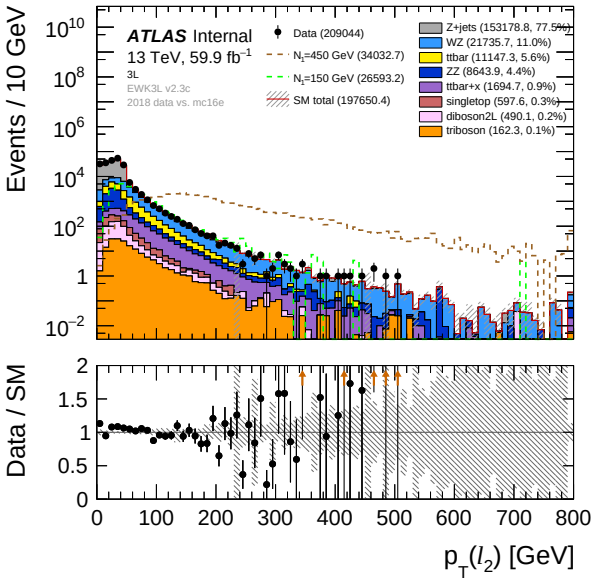
(e) Eta lepton 3

(f) Phi lepton 3

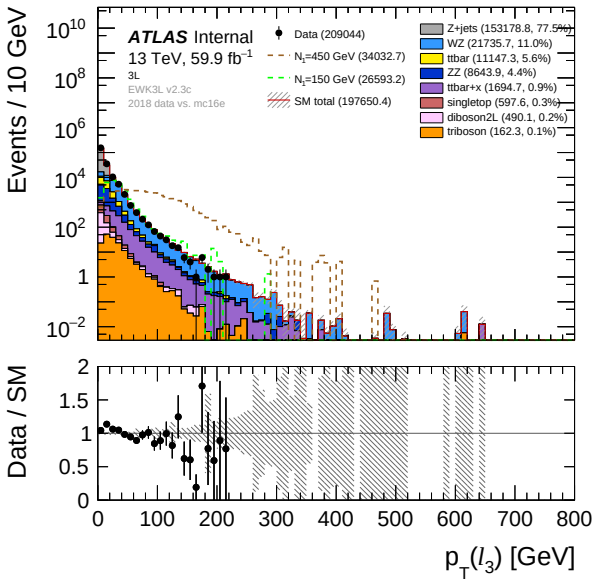
Figure 7.3: Plot of the Eta and Phi for the three leptons with data, MC and two neutrino signals. All distributions show very little dependence on phi/eta.



(a) Pt lepton 1

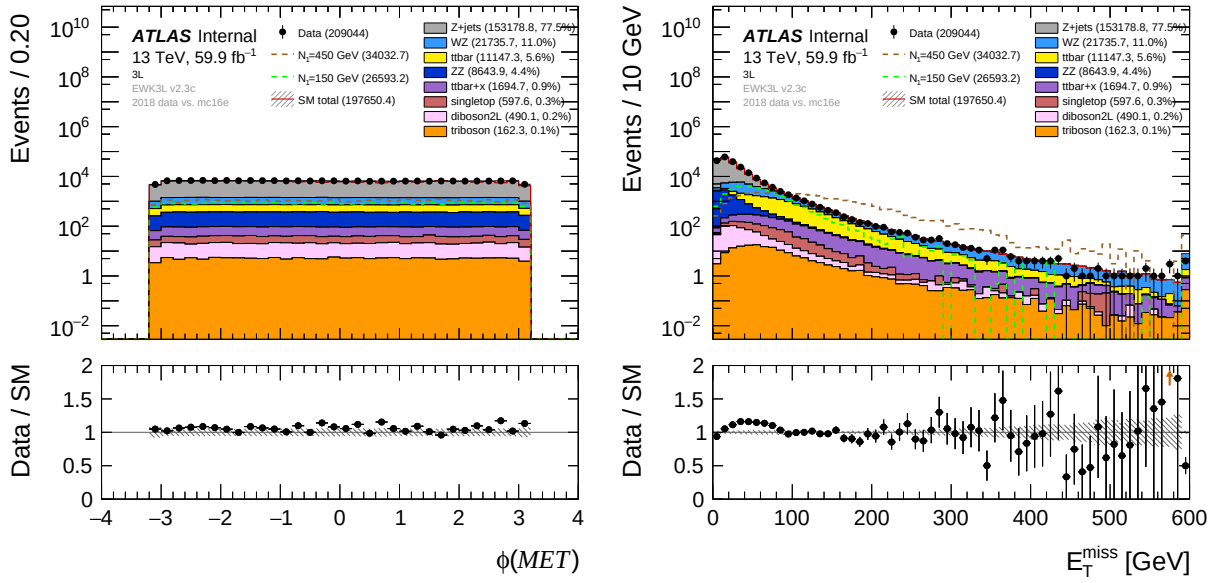


(b) Pt lepton 2



(c) Pt lepton 3

Figure 7.4: Plot of the Pt of the three leptons with data, MC and two neutrino signals. The number of events with higher transverse momentum decreases for each lepton, where lepton 1 has most events reaching above 800 GeV.



(a) Phi neutrino

(b) Missing transverse momentum

Figure 7.5: The figure shows the Phi and E_t of the missing transverse momentum for data, MC and two neutrino signals. The Phi feature is similar to the three lepton Phi features, and the MET is similar to the Pt feature for the leptons.

7.4 Making New Variables

We start by converting the desired datasets with proton-proton collision events from ROOT to Python using the Uproot library. The files are then stored as dataframes with Pandas. This is done in the script called *Trilepton_read_root.py*. By using the momentum and energy of the leptons in each event we compute new useful variables, which are added to a new dataframe. The new variables we make are the angular variables for each particle (three leptons and a neutrino) (θ, ϕ, η), angular variables between pairs of particles ($d\phi$, dR) and the invariant masses of pairs of leptons (m_{ll}). We also have p_x, p_y, p_z, p_t and E for all four particles. For benchmark cuts we will use later (sect. 9.2), we also make a m_{3l} variable for the invariant mass of the three lepton system. The new dataframes can now be imported by other scripts to be used further with ML.

The reason we make new variables for eta, phi and p_t that already exists is that during the making of the invariant masses, we get some errors for some events where the $p > E$. From the Einstein energy-momentum relation in equation 4.1, the invariant masses becomes negative. This is not (physically) correct, and since this does not happen too often we simply drop these events in all features. This could be a simulation error, but is not known or explored further in this thesis.

In the Born diagram in Figure 3.1, the first lepton ($l_1^{\pm}$) and the pseudo-Dirac neutrino (N) comes from the first vertex. The second lepton ($l_2^{\mp}$) and the W-boson comes from the second vertex, while the third lepton ($l_3^{\pm}$) and the neutrino (ν) comes from the third vertex. By using the identity traits, the particle vertex and particle ID, we classify the events by constructing a *target* variable as permutations of the vertexes the leptons come from for the two signals samples we will use to train the classification models. The leptons are ordered by decreasing p_T, for both signals and backgrounds. The neutrino will always come from the fourth vertex in the decay chain and is not considered in the targets. This leads to the following vertex permutations for the leptons:

$$[123, 132, 213, 231, 312, 321] \tag{7.1}$$

This means that e.g. the 132 vertex has the highest p_T lepton coming from vertex 1, second lepton from vertex 3 and thrid lepton from vertex 2.

The function for making the new variables can be seen in Listing 7.1. The new dataframe is then exported as a .h5-file.

```
# Method for flattening and adding additional variables
def lepaugmentation(df, nlep):
    px  = awkward.fromiter(df['px'])
    py  = awkward.fromiter(df['py'])
    pz  = awkward.fromiter(df['pz'])
    E   = awkward.fromiter(df['E'])
    vtx = awkward.fromiter(df['vtxid'])
    pid = awkward.fromiter(df['pdgid'])

    # Make tlv - handy when computing angular variables
    tlv = uproot_methods.classes.TLorentzVector.TLorentzVectorArray.↩
        from_cartesian(px, py, pz, E)
```

```
            \\...

            # Make the lepton variables
            for i in range(1,nlep+1):
                df['lep%i_pt'%i]  = pt[pt.argmax()].flatten()
                df['lep%i_phi'%i]  = phi[pt.argmax()].flatten()
                df['lep%i_eta'%i]  = eta[pt.argmax()].flatten()
                df['lep%i_theta'%i]  = theta[pt.argmax()].flatten()
                df['lep%i_px'%i]  = px[pt.argmax()].flatten()
                df['lep%i_py'%i]  = py[pt.argmax()].flatten()
                df['lep%i_pz'%i]  = pz[pt.argmax()].flatten()
                df['lep%i_E' %i]  = E[pt.argmax()].flatten()
                df['lep%i_vtx'%i]   = vtx[pt.argmax()].flatten()
                df['lep%i_pid'%i]   = pid[pt.argmax()].flatten()
                df['lep%i_tlv'%i]   = tlv[pt.argmax()].flatten()

                \\...

        # Compute variables for all combinations of 2 leptons
        pairs = pt_org.argchoose(2)
        print("pairs:", pairs)
        left  = pairs.i0
        right = pairs.i1

        \\...

        for ilep in range(len(left[0])):
            i = left[0][ilep]
            j = right[0][ilep]
            print('i = %i, j = %i'%(i,j))
            idx1 = left[0][i]
            idx2 = right[0][i]

            df['mll_%i%i'%(i+1,j+1)]   = (df['lep%i_tlv'%(i+1)]+df['lep%i_tlv'%(j+1)]).↩
                apply(get_invmass)
            df['dphi_%i%i'%(i+1,j+1)] = df.apply(lambda x : get_deltaPhi(x['lep%i_tlv'%(↩
                i+1)],x['lep%i_tlv'%(j+1)]), axis=1)
            df['dR_%i%i'%(i+1,j+1)]   = df.apply(lambda x : get_deltaR(x['lep%i_tlv'%(i↩
                +1)],x['lep%i_tlv'%(j+1)]), axis=1)

        if Truth:
            df['target'] = df.apply(lambda x : classify_event(x['lep1_vtx'],x['lep2_vtx'↩
                ],x['lep3_vtx'],x['lep4_vtx'],x['lep1_pid'],x['lep2_pid'],x['lep3_pid'],↩
                x['lep4_pid']), axis=1)


        df = df.drop(['px', 'py', 'pz', 'pt', 'E', 'vtxid', 'pdgid', 'evnum', 'onshell_w↩
            ', 'tlv', 'phi', 'theta', 'eta', 'lep1_tlv', 'lep2_tlv', 'lep3_tlv', '↩
            lep4_tlv'], axis=1)

        return df
```

Listing 7.1: Function for making new variables.

7.4.1 Plotting New Variables

To plot the newly produced variables, we will convert the dataframes back into ROOT to make similar plots as shown in section 7.3. First is to convert the dataframes into comma separated values files, .csv (CSV), before converting them into ROOT. In Listing 7.2 we see how we convert from CSV to ROOT. We only look at the MC and signal data, which

contain the samples of interest to us in this thesis. With the new background and signal Ntuples, we can use the same plotting scripts as earlier (sect. 7.3) to make plots of the newly produced features.

```
TFile *f = new TFile(Filename_ROOT, "RECREATE") # Create file
TTree *tree = new TTree(Name_of_Tree, Title_of_Tree) # Create tree
tree->ReadFile(Filename_CSV) # Read the .csv-file
tree->Fill()
tree->Write()
```

Listing 7.2: Convert from CSV to ROOT.

The momentum features (p_x, p_y, p_z) for all four particles are seen in Figures 7.6 and 7.7. They all peak around 0 GeV and decreases as the absolute value of the momenta increases. p_x and p_y are very similar, while p_z have more spiked peaks around 0 GeV and have more events with higher momentum than the other two momentum coordinates. The z-direction is the direction the particles travel initially, thus it makes sense to have somewhat higher momentum along the z-direction. The only differences between these particle momentum plots are more or less the width of the event peaks and how much momentum the events reach. Like before, lepton 1 reaches higher momentum and decrease for the other particles. Lepton 1 also has the broadest peak around 0 GeV which gets more narrow for each lepton.

The individual angle features, η, θ and ϕ, of the four particles are seen in Figures 7.8 and 7.9. The η and ϕ features are similar to the ones in the original Ntuples with more or less an equal amount of events for each eta and phi value, except now a peak in events around $\eta = 0$ appears. θ shows similar traits like η with equal amount of events except for a peak around $\theta = \pi/2$.

The transverse momentum features are seen in Figure 7.10 for all four particles. They all have most events below $p_T = 100$ GeV, decreasing as p_T increases. Lepton 1 has most events reaching higher p_T values around 800 GeV, while the neutrino with smallest p_Ts has only most events reaching p_T around 400 GeV. The E features in Figure 7.11 show the same type of behavior like the p_T features, where lepton 1 has much higher energy than the other particles and it decreases more and more for each particle. Lepton 1 has events reaching around 1 TeV, while the neutrino only has events reaching around 200 GeV. The invariant mass of the three lepton system is similar to the lepton 1 energy, reaching around 1 TeV for all backgrounds. The 450 GeV signal has more events for higher mass $>$ 450 GeV, while the 150 GeV signal has more events for lower masses.

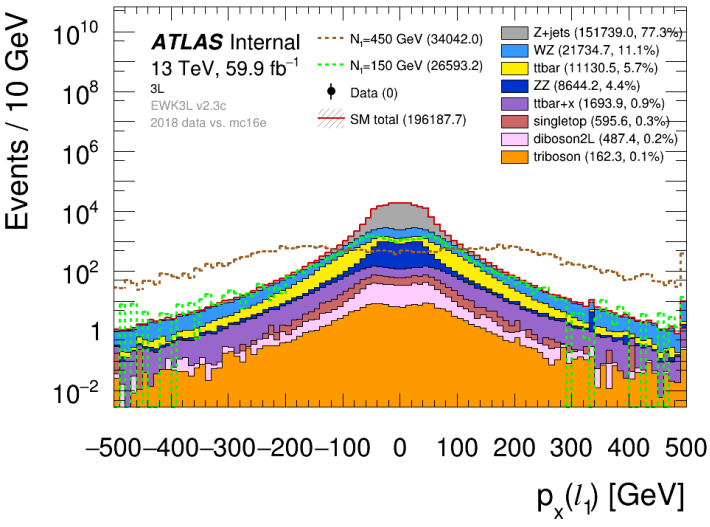
The invariant mass pair plots in Figure 7.12 look similar to the transverse momentum plots, where the event peaks are between 0 and 100 GeV depending on the particle combinations, and decreases as the invariant masses increases. The 150 GeV signal follows the MC for the number of events while the 450 GeV signal first reaches the peak around 400 GeV for the more massive combinations.

The azimuthal angle difference between pairs of particles are seen in Figure 7.13. Most of the pairs have small peaks around $\Delta\phi = \pm\pi$ and $\Delta\phi = 0$ with not much difference in the number of events elsewhere. The two signals have fewer events around $\Delta\phi = 0$,

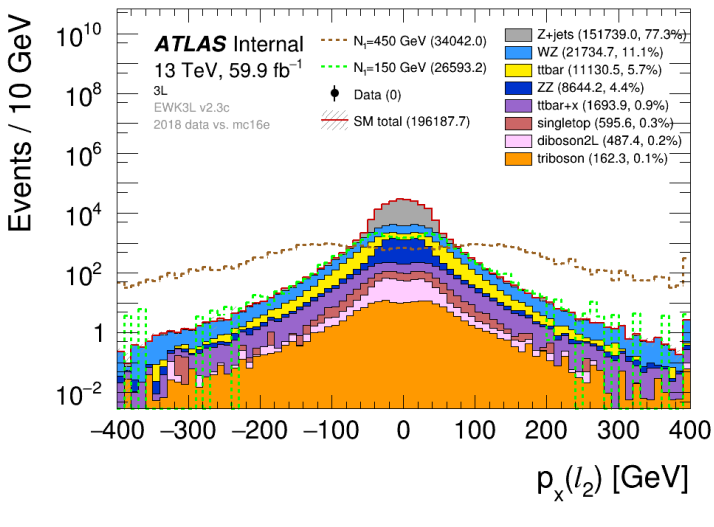
especially for the lepton 1 and 2 and 1 and 3 pairs.

The angular distance features for all four particles in Figure 7.14 have the number of events increasing slowly until the angular distance is around 3.2 before it decreases more rapidly when the angular distance approaches 6. This happens for all combinations of angular distances. The main differences is how steep the increase and decrease are when the angular distance approaches 3.2 and 6, respectively.

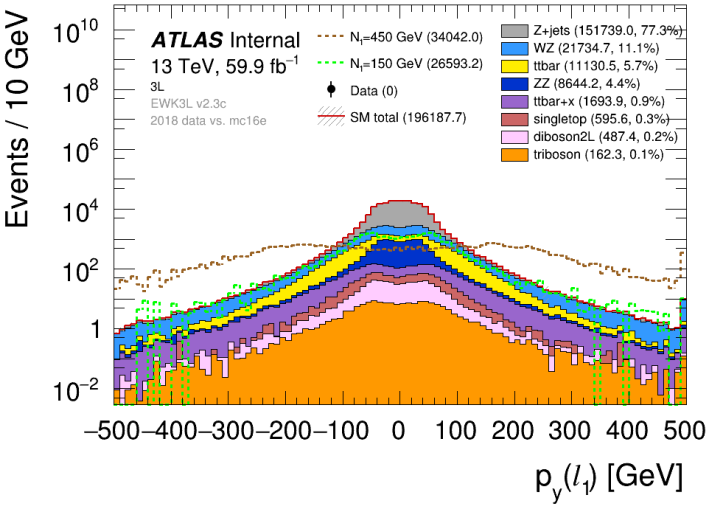
The plots of the features we have produced are as expected, except for the spiked event peaks for p_z, θ and η for all four particles. We do not know why we get more events at these feature values.



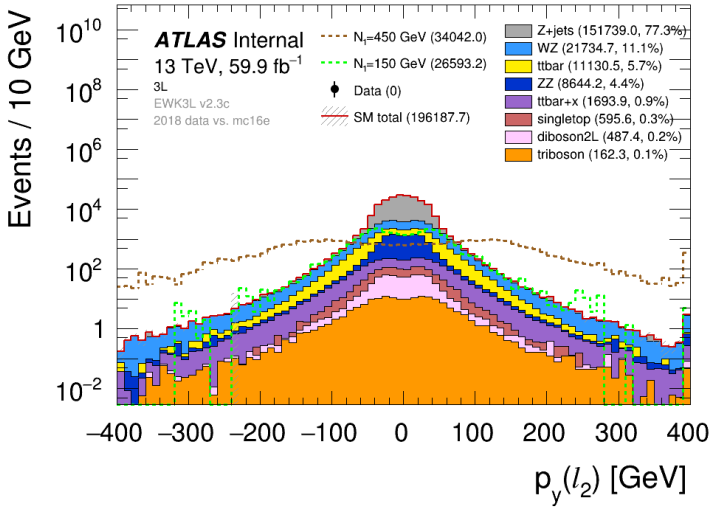
(a) p_x lepton 1



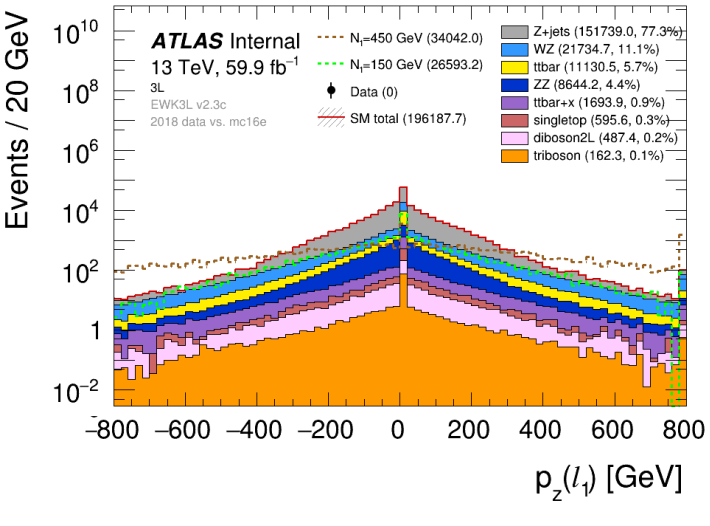
(b) p_x lepton 2



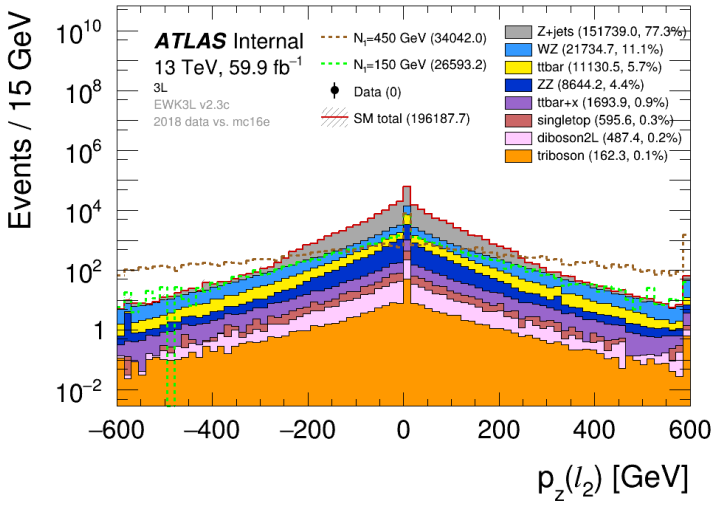
(c) p_y lepton 1



(d) p_y lepton 2

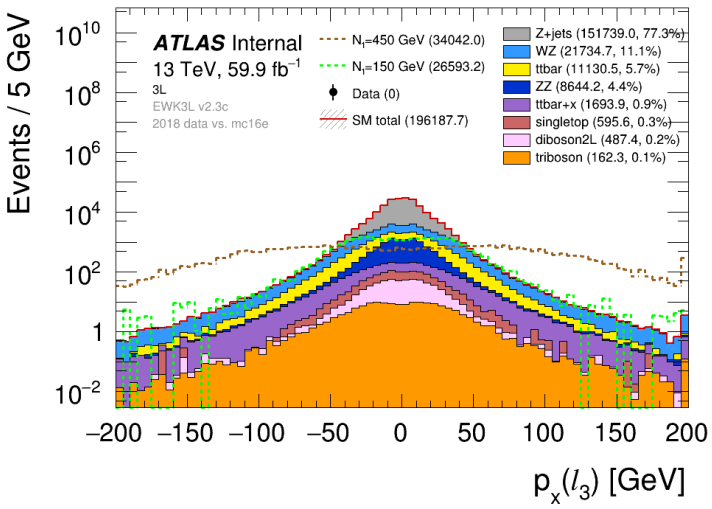


(e) p_z lepton 1

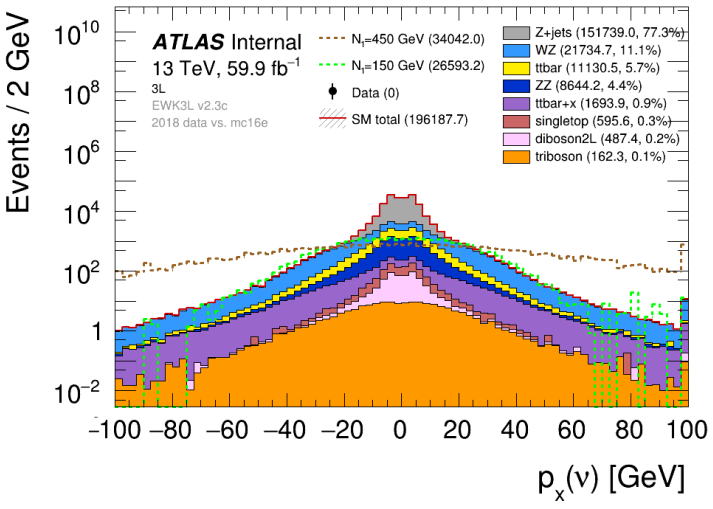


(f) p_z lepton 2

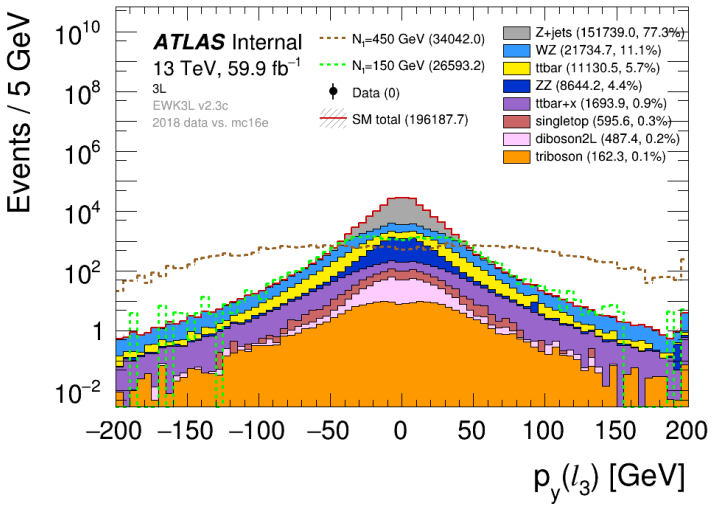
Figure 7.6: The momentum features of lepton 1 and 2. They both have number of event peaks around 0 GeV, but lepton 1 has a broader peak reach higher (absolute) energies.



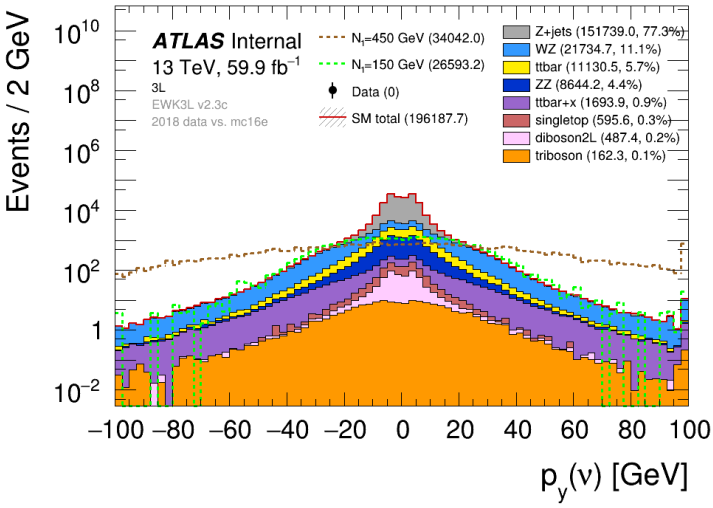
(a) p_x lepton 3



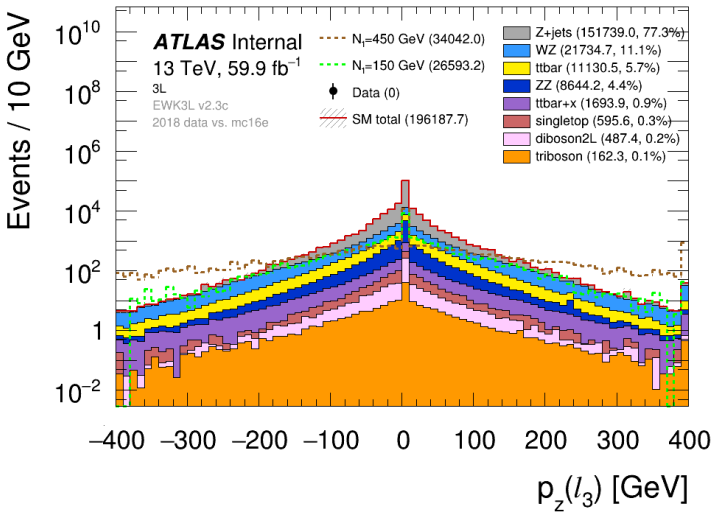
(b) p_x neutrino



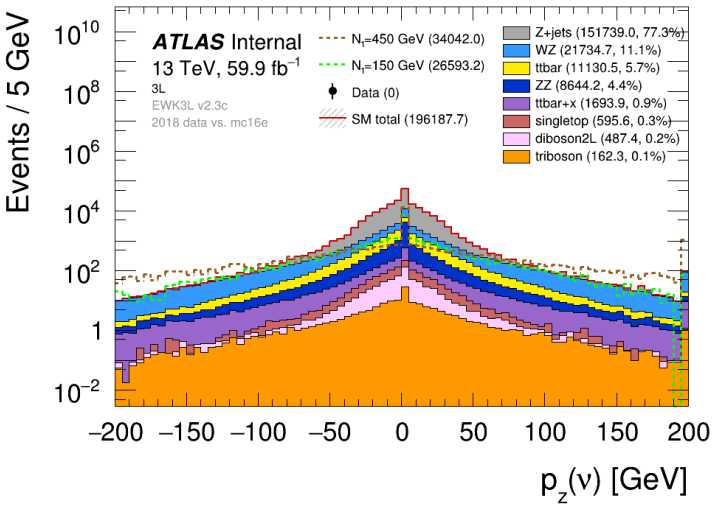
(c) p_y lepton 3



(d) p_y neutrino

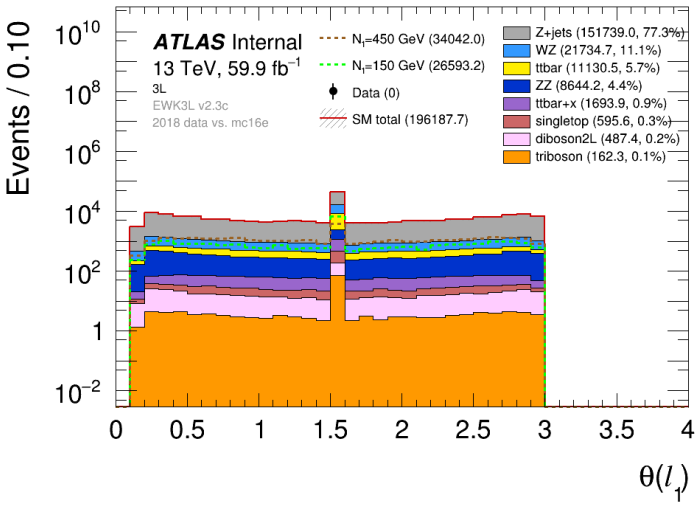


(e) p_z lepton 3

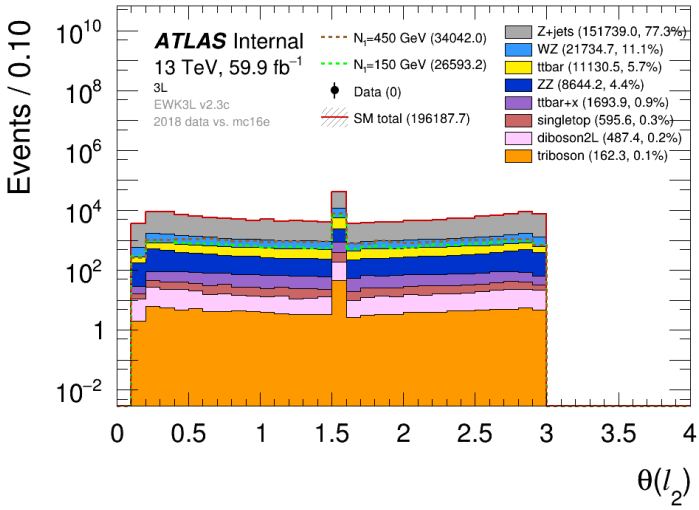


(f) p_z neutrino

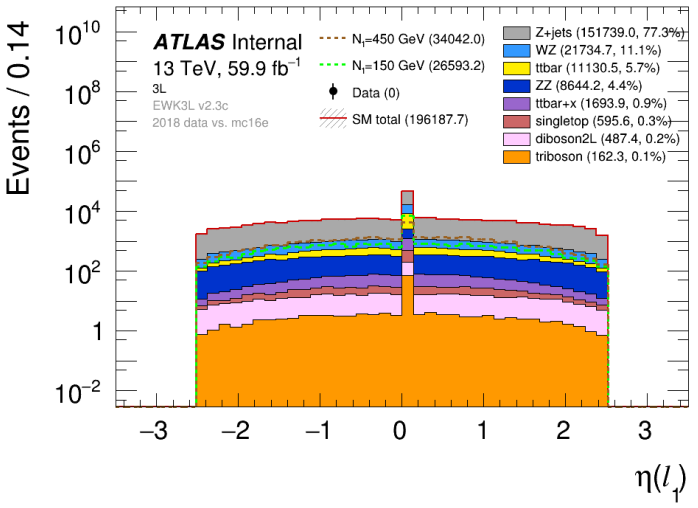
Figure 7.7: The momentum features of lepton 3 and the neutrino. These plots are similar to lepton 1 and 2, except for the lower highest momentum for the events.



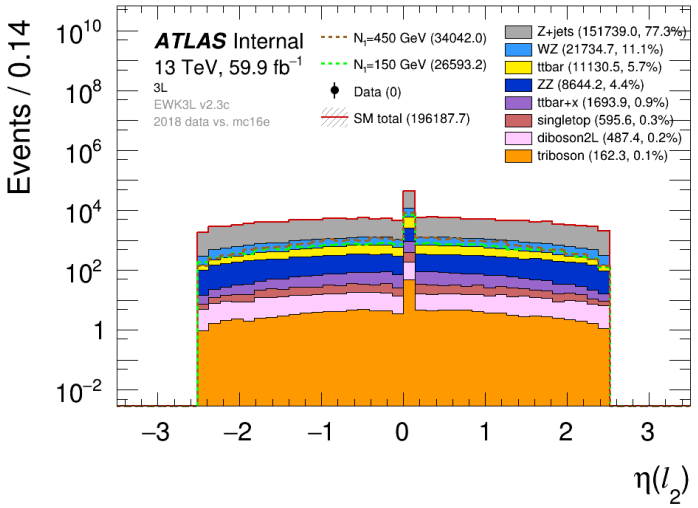
(a) θ lepton 1



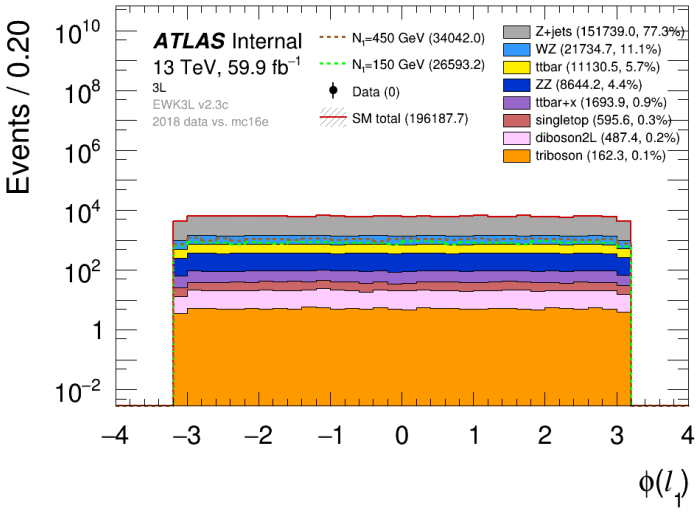
(b) θ lepton 2



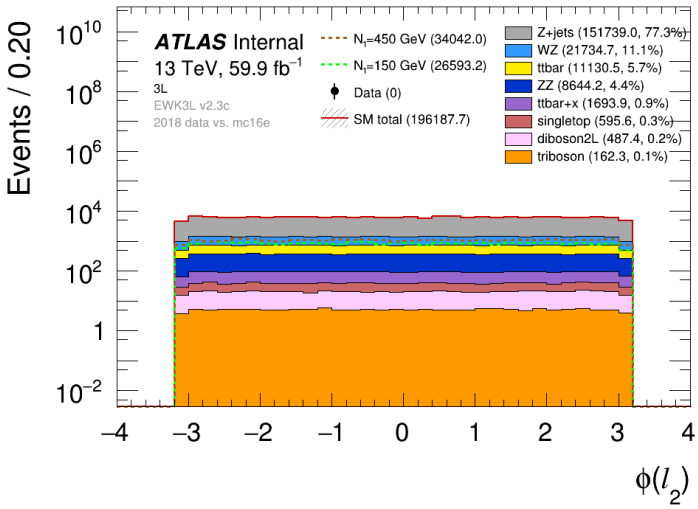
(c) η lepton 1



(d) η lepton 2

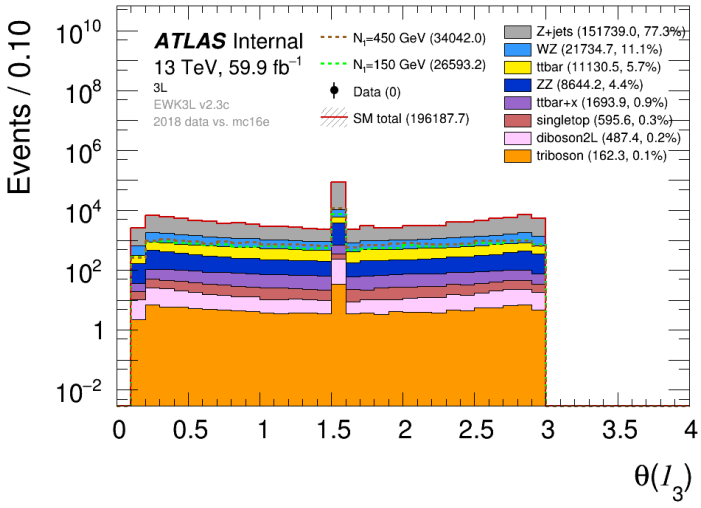


(e) ϕ lepton 1

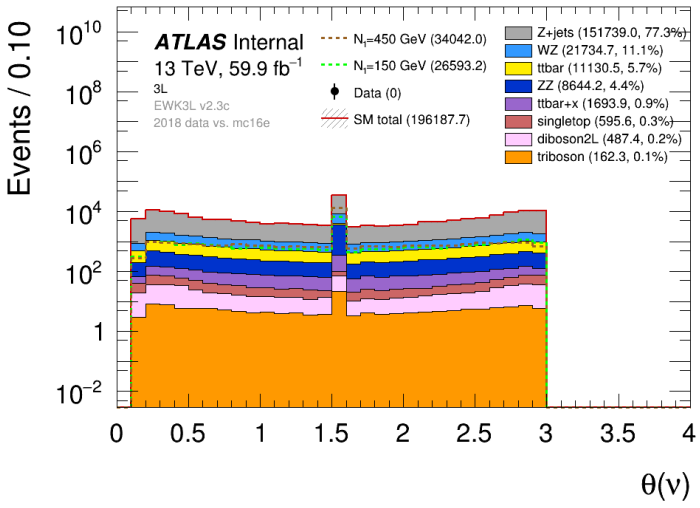


(f) ϕ lepton 2

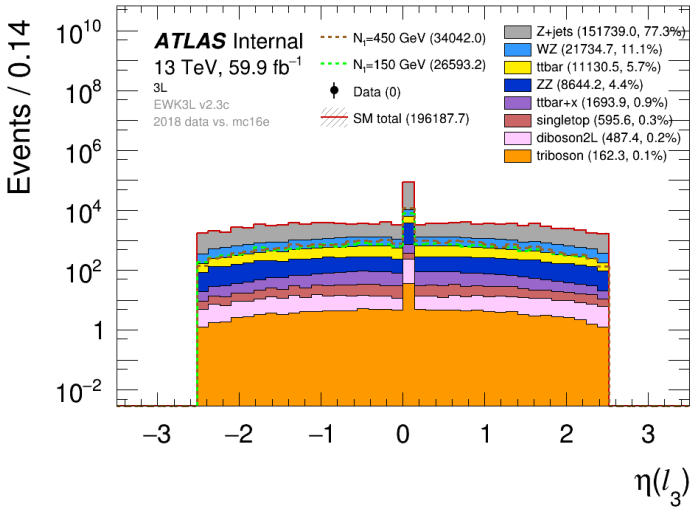
Figure 7.8: The angular features (θ, η, ϕ) for lepton 1 and 2. Almost all the angular values have an equal amount of events, except for θ with a peak around $\theta = \pi/2$ and η with a peak around $\eta = 0$.



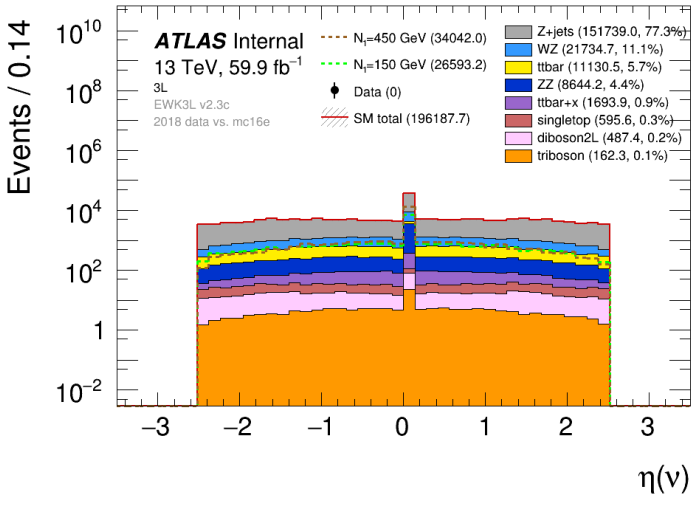
(a) θ lepton 3



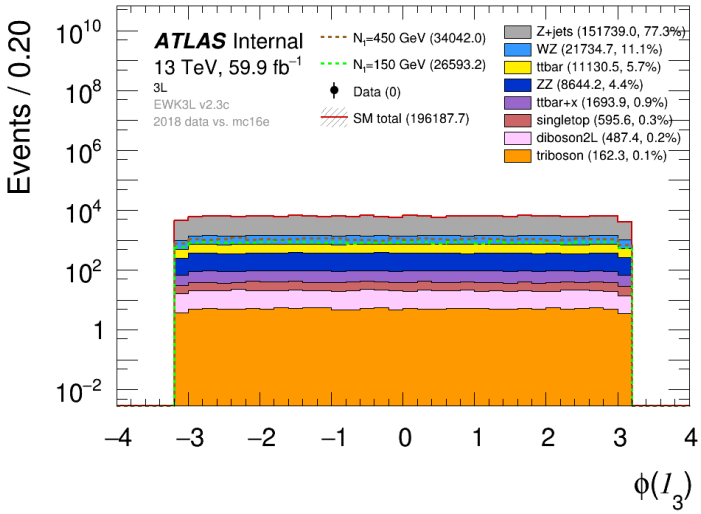
(b) θ neutrino



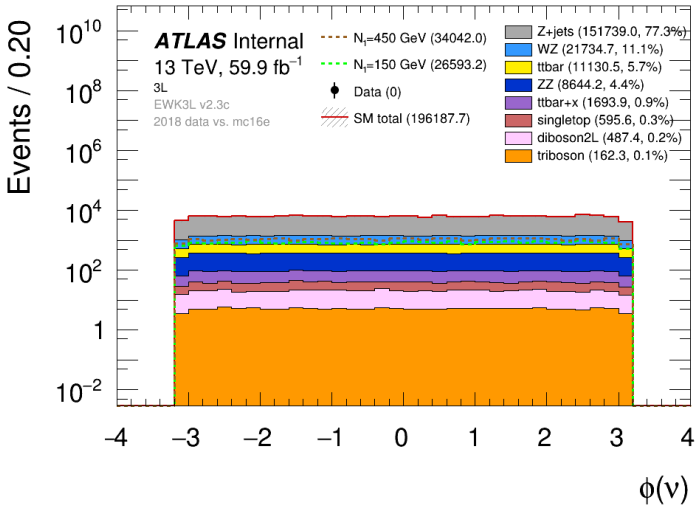
(c) η lepton 3



(d) η neutrino

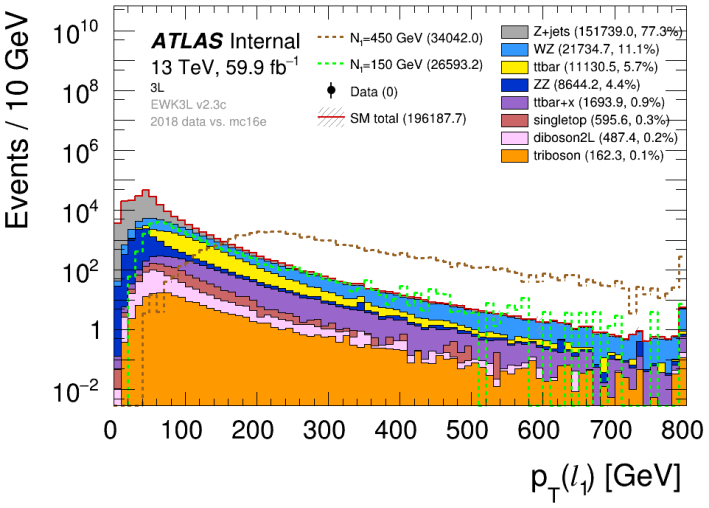


(e) ϕ lepton 3

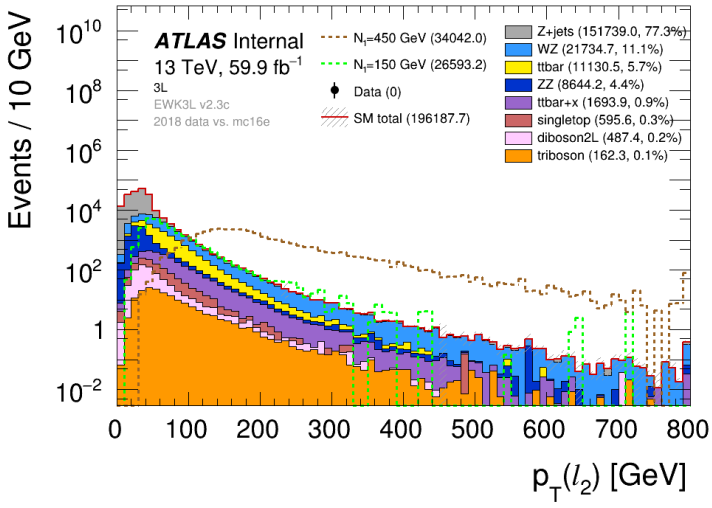


(f) ϕ neutrino

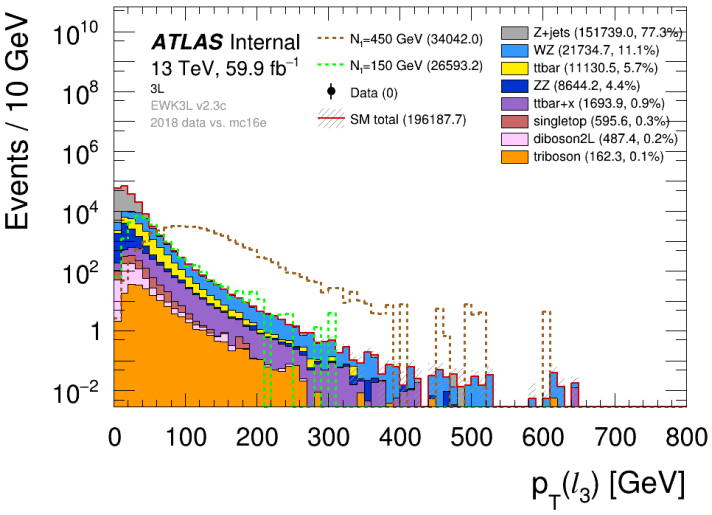
Figure 7.9: The angular features (θ, η, ϕ) for lepton 3 and the neutrino. Almost all the angular values have an equal amount of events, except for θ with a peak around $\theta = \pi/2$ and η with a peak around $\eta = 0$.



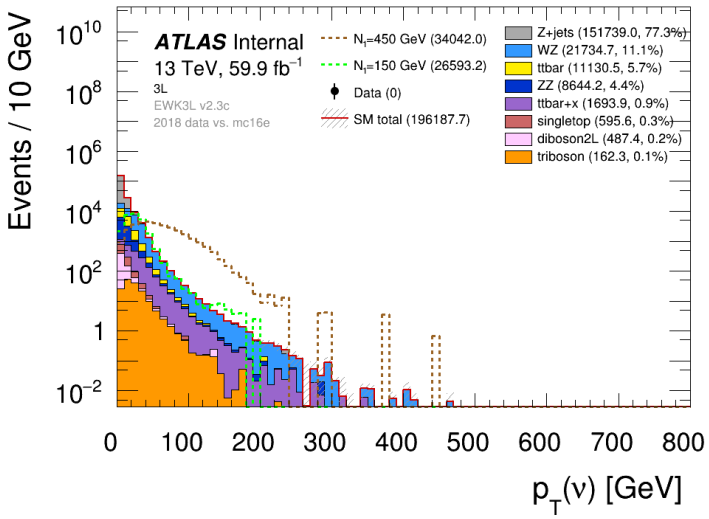
(a) p_T lepton 1



(b) p_T lepton 2

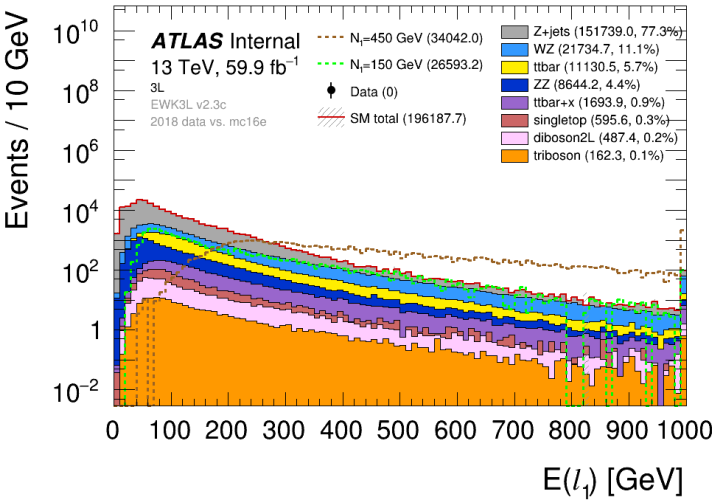


(c) p_T lepton 3

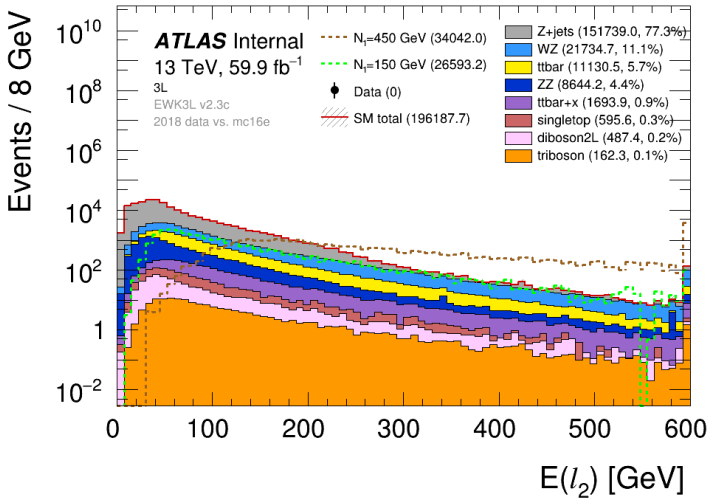


(d) p_T neutrino

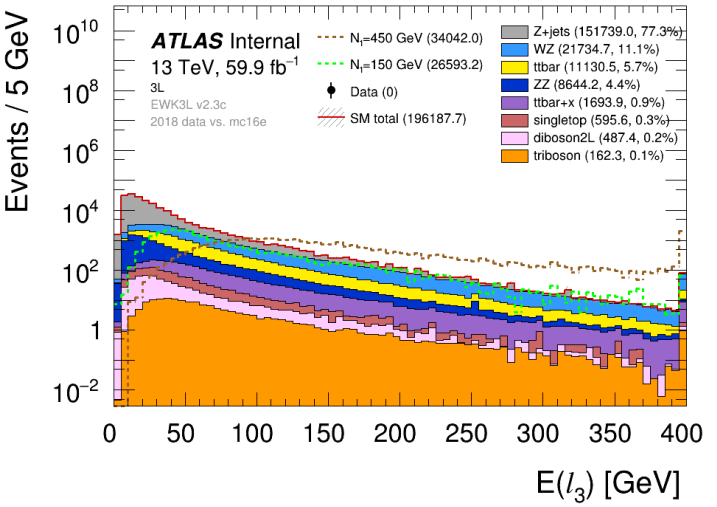
Figure 7.10: The transverse momentum for all four particles showing decreasing number of events when the momenta increases. Lepton 1 has events reaching highest p_T around 800 GeV.



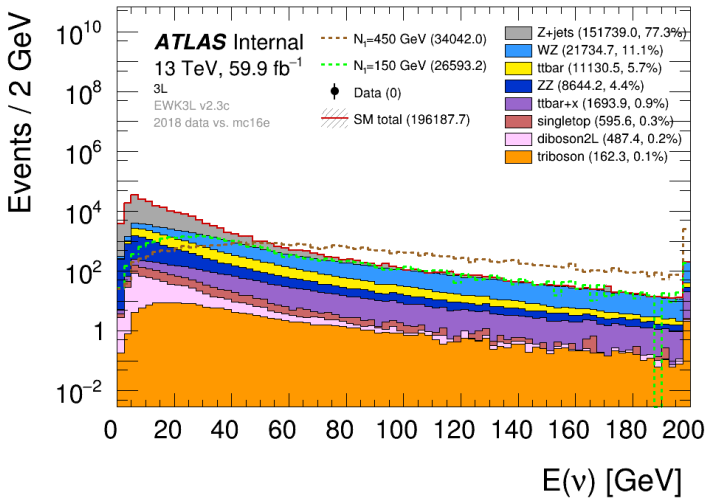
(a) E lepton 1



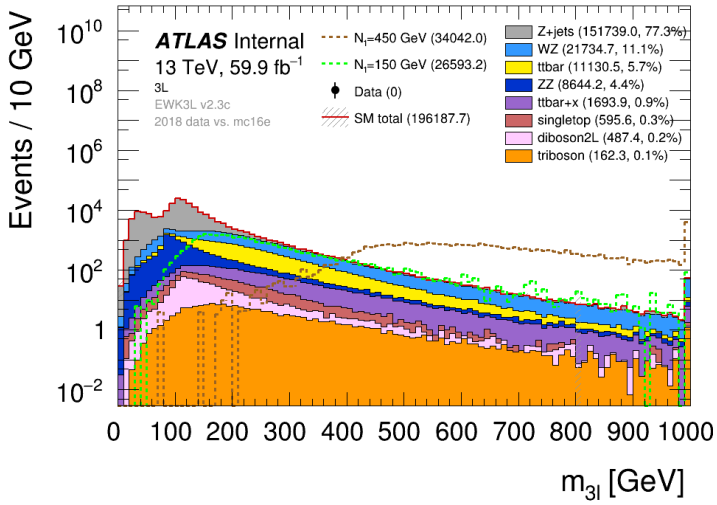
(b) E lepton 2



(c) E lepton 3

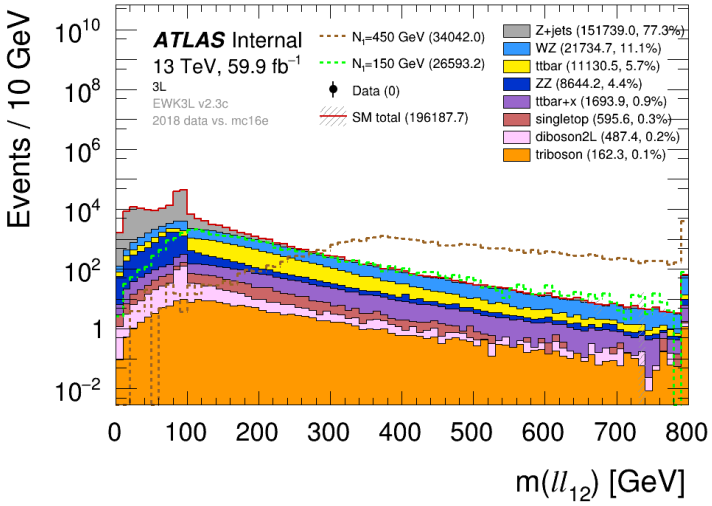


(d) E neutrino

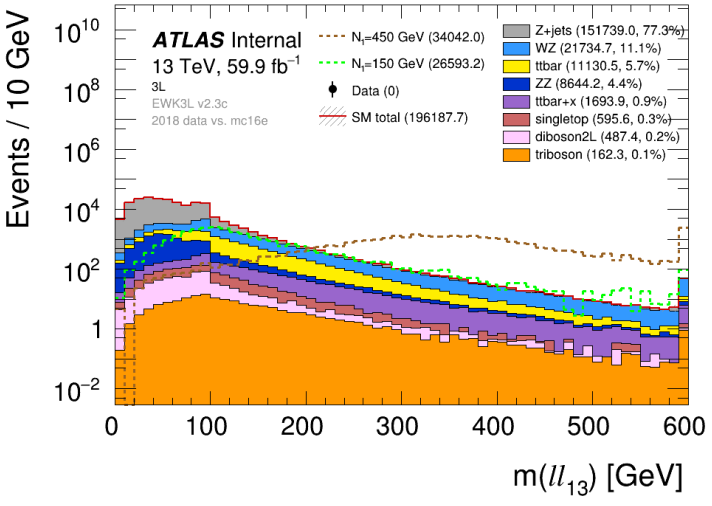


(e) Invariant mass of three lepton system

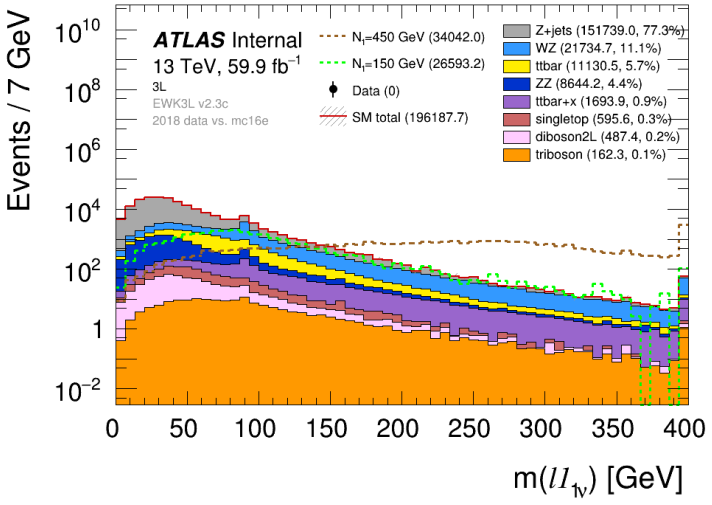
Figure 7.11: Energy for all particles in Figures 7.11a to 7.11d, and the invariant mass for the three lepton system in 7.11e. Similar behavior for lepton 1 and invariant mass with high amount of events reaching around 1 TeV.



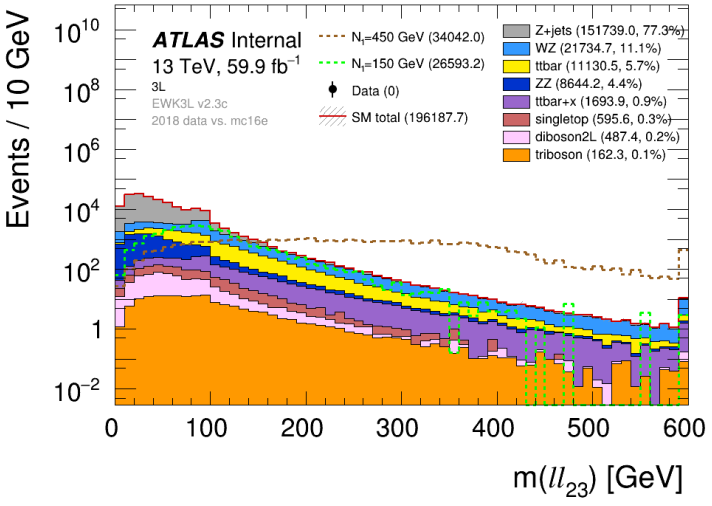
(a) m_{ll} lepton 1 and 2



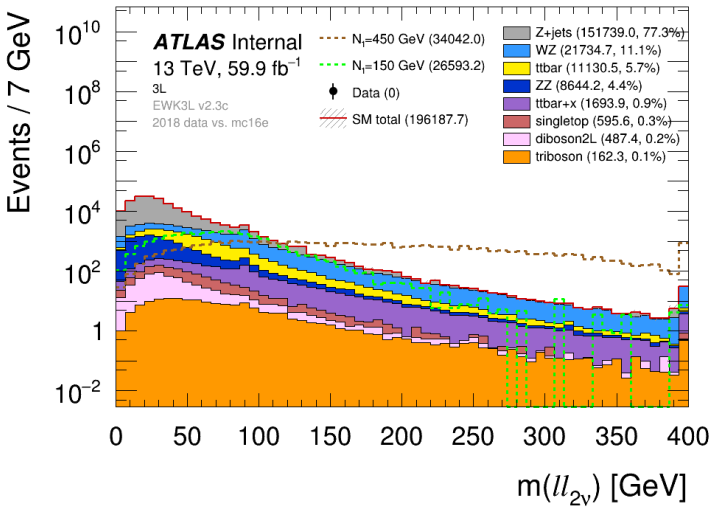
(b) m_{ll} lepton 1 and 3



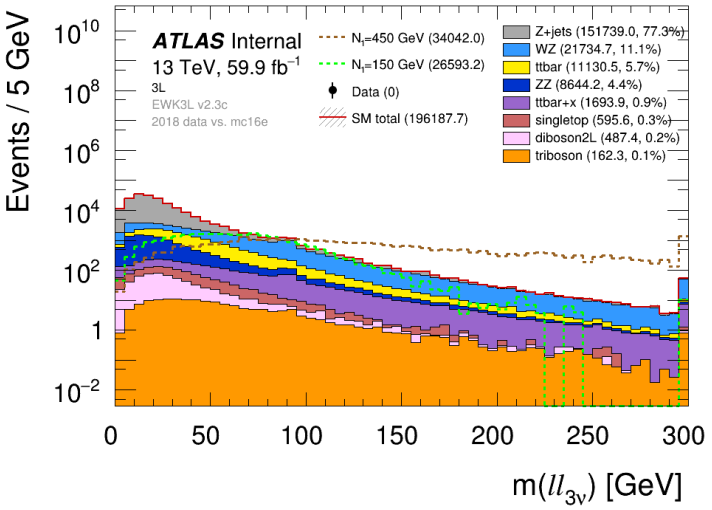
(c) m_{ll} lepton 1 and neutrino



(d) m_{ll} lepton 2 and 3

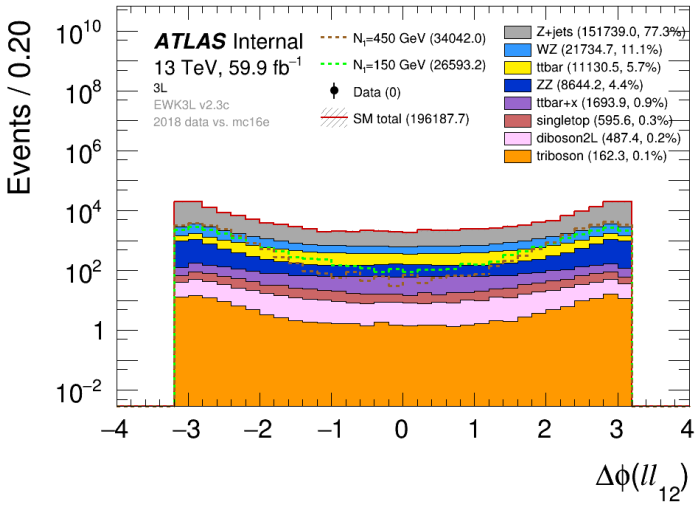


(e) m_{ll} lepton 2 and neutrino

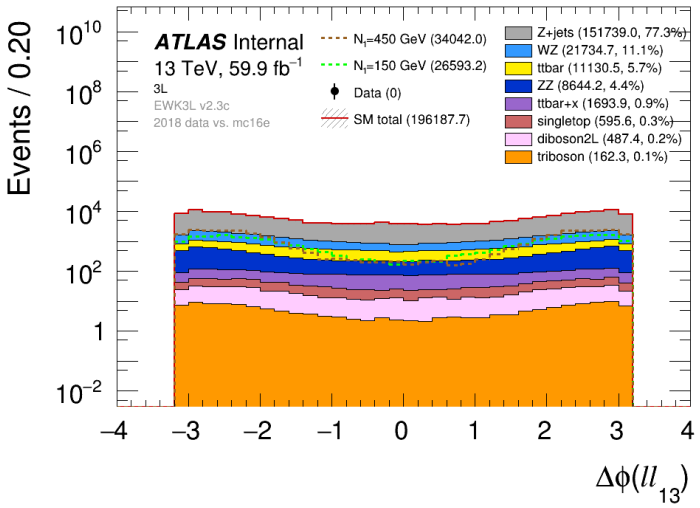


(f) m_{ll} lepton 3 and neutrino

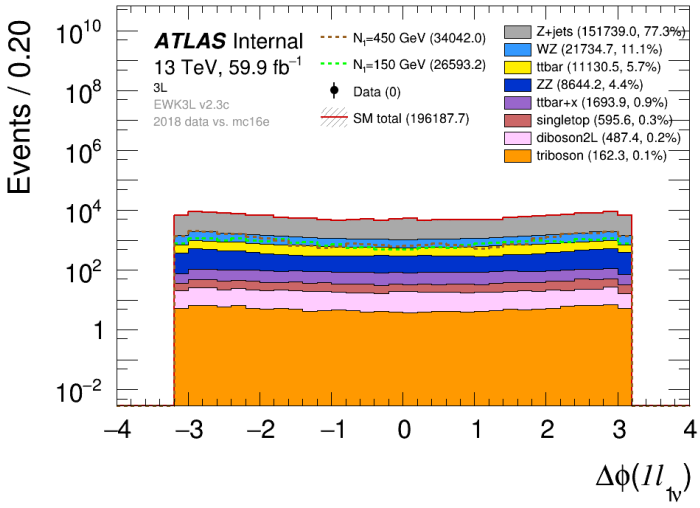
Figure 7.12: Invariant masses between pairs of particles. The further out in the vertices the particles appear, the less is the invariant mass of the combinations of those particles.



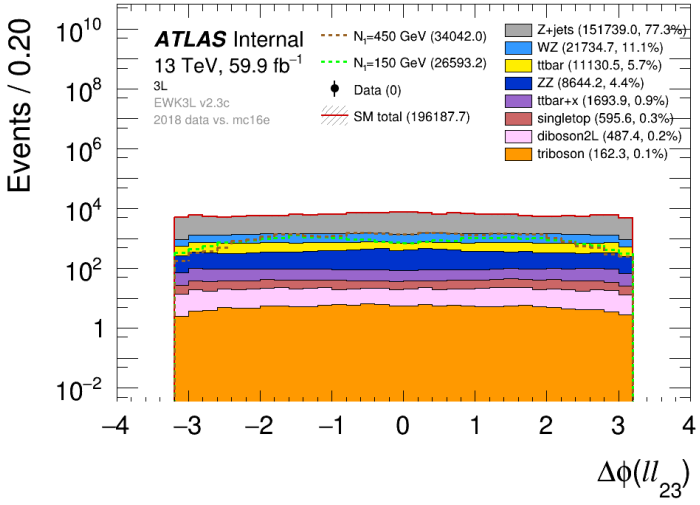
(a) $\Delta\phi$ lepton 1 and 2



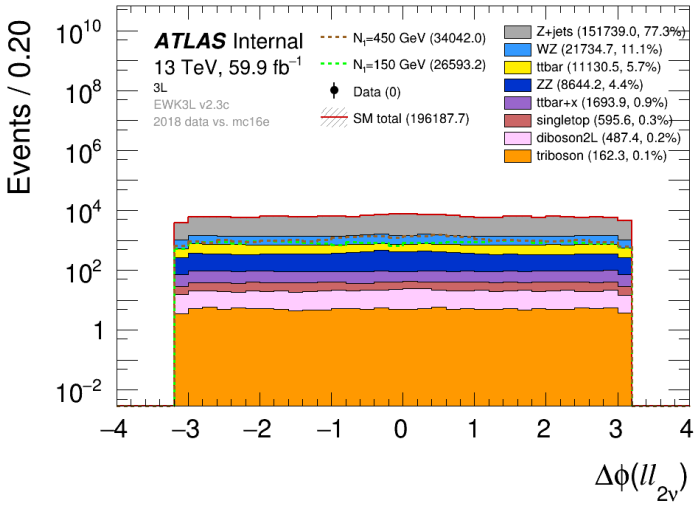
(b) $\Delta\phi$ lepton 1 and 3



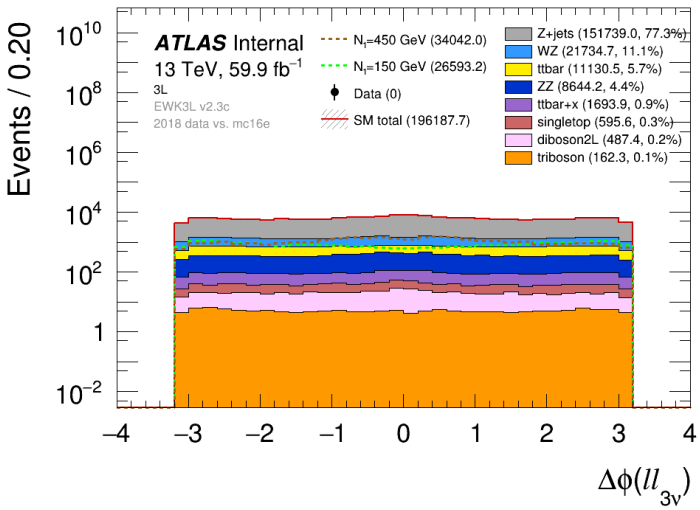
(c) $\Delta\phi$ lepton 1 and neutrino



(d) $\Delta\phi$ lepton 2 and 3

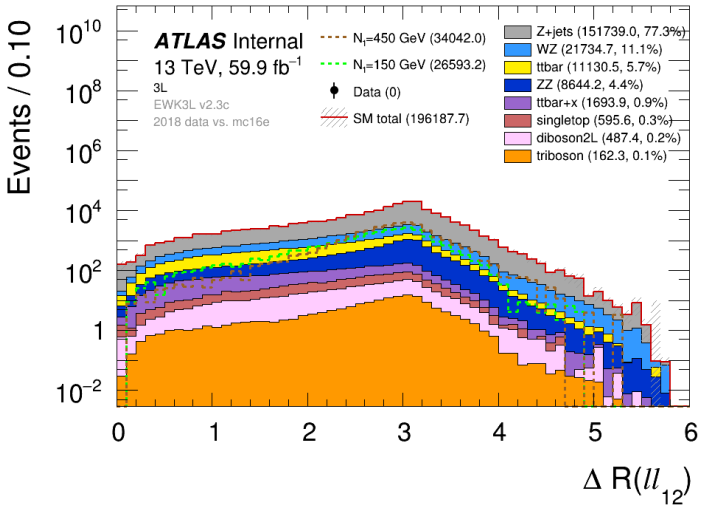


(e) $\Delta\phi$ lepton 2 and neutrino

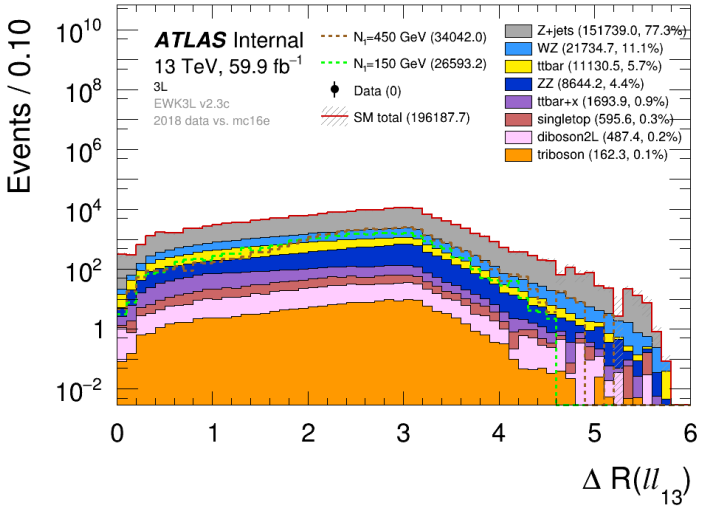


(f) $\Delta\phi$ lepton 3 and neutrino

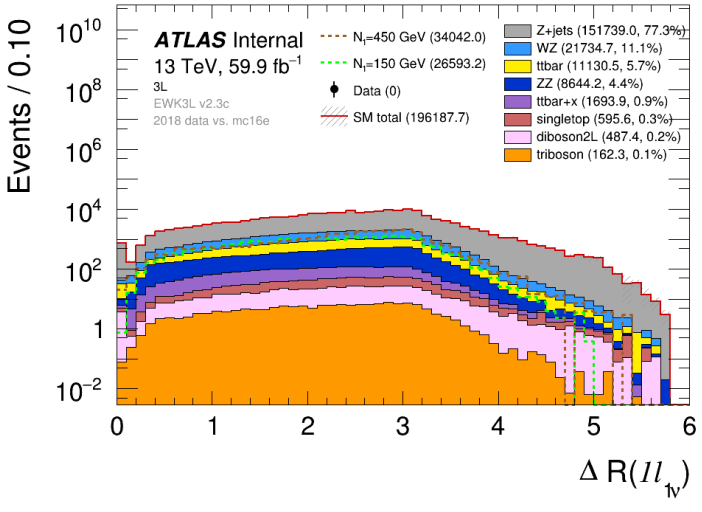
Figure 7.13: The azimuthal angular difference features between pairs of particles. Most of the combinations have small peaks around $\Delta\phi = \pm\pi$ and $\Delta\phi = 0$.



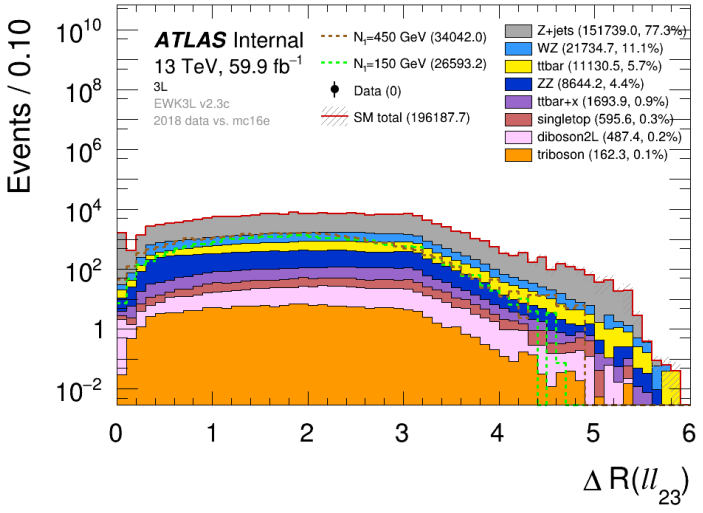
(a) ΔR lepton 1 and 2



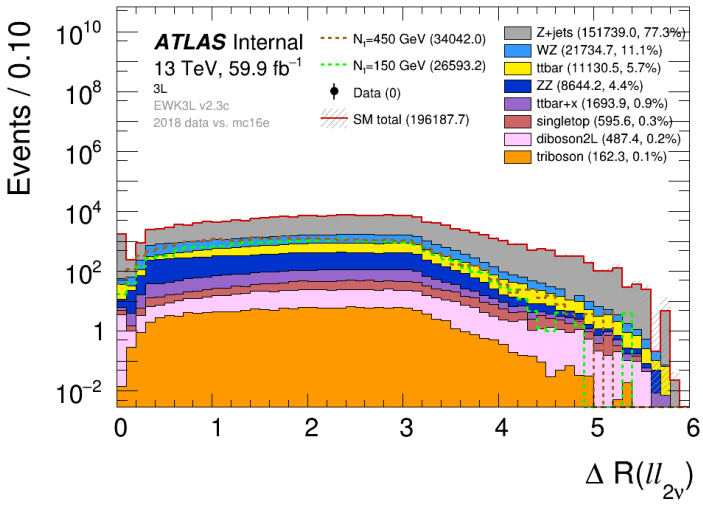
(b) ΔR lepton 1 and 3



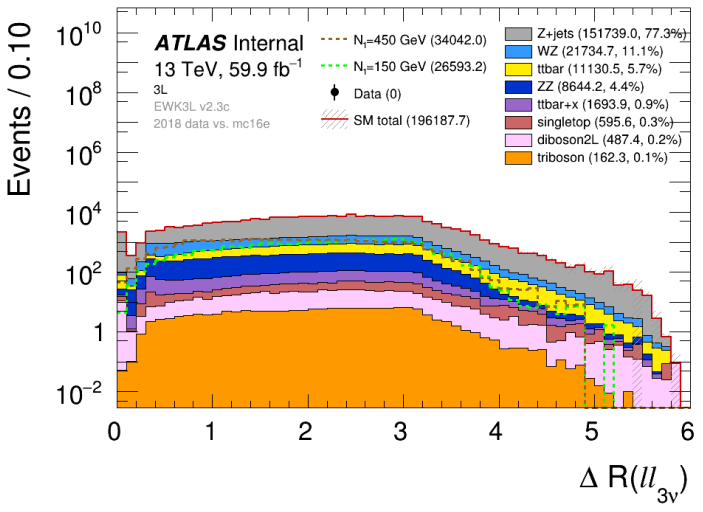
(c) ΔR lepton 1 and neutrino



(d) ΔR lepton 2 and 3



(e) ΔR lepton 2 and neutrino



(f) ΔR lepton 3 and neutrino

Figure 7.14: The angular distance features between pairs of particles. The number of events increase mostly as ΔR approaches 3.2, and the decreases as ΔR approaches 6.

Chapter 8

Evaluation of ML Models

8.1 Preprocessing of the Data

After importing all the necessary libraries mentioned in section 7.1 in a new script, *Trilepton_classifier.py*, we load the dataframes and drop unnecessary features that we have not explained at this point and events with e.g. NaN values we will not consider in the classification. Using the properties of Pandas dataframes and utilizing Seaborn and Scikit-Learn we will do some preprocessing of the data. One thing we have to consider for our dataset is NaN, or NULL, values. This have to be done since we in section 7.4 got errors in some events when $p > E$ which we gave NaN values and were dropped. Not all classification models can deal with NaN values so we cannot have those in the datasets. We double-check for any remaining NaN values and can drop them easily from the dataframe by doing the following:

```
df.isnull()   # Returns a boolean matrix, if the value is NaN then True ↩
    otherwise False.
df.isnull().sum()   # Returns the column names along with the number of NaN ↩
    values in that particular column.
#df.dropna(inplace=True)  # Removes rows in the dataframe containing NaN values.
```

Listing 8.1: Check NaN values in the datasets to be removed if existing.

8.1.1 Inspect Data

Pandas dataframes lets us easily print a few lines and a summary of the dataframe. We then get a quick overview of what the data looks like. The $N1 = 150$ GeV data summary:

```
<class 'pandas.core.frame.DataFrame'>
Int64Index: 66885 entries, 0 to 67773
Data columns (total 55 columns):
 #   Column       Non-Null Count  Dtype
---  ------       --------------  -----
 0   lep1_pt      66885 non-null  float32
 1   lep1_phi     66885 non-null  float32
```

```
 2   lep1_eta     66885 non-null  float32
 3   lep1_theta   66885 non-null  float32
 4   lep1_px      66885 non-null  float32
 5   lep1_py      66885 non-null  float32
 6   lep1_pz      66885 non-null  float32
 7   lep1_E       66885 non-null  float32
 8   lep1_tlv     66885 non-null  object
 9   lep2_pt      66885 non-null  float32
 10  lep2_phi     66885 non-null  float32
 11  lep2_eta     66885 non-null  float32
 12  lep2_theta   66885 non-null  float32
 13  lep2_px      66885 non-null  float32
 14  lep2_py      66885 non-null  float32
 15  lep2_pz      66885 non-null  float32
 16  lep2_E       66885 non-null  float32
 17  lep2_tlv     66885 non-null  object
 18  lep3_pt      66885 non-null  float32
 19  lep3_phi     66885 non-null  float32
 20  lep3_eta     66885 non-null  float32
 21  lep3_theta   66885 non-null  float32
 22  lep3_px      66885 non-null  float32
 23  lep3_py      66885 non-null  float32
 24  lep3_pz      66885 non-null  float32
 25  lep3_E       66885 non-null  float32
 26  lep3_tlv     66885 non-null  object
 27  lep4_pt      66885 non-null  float32
 28  lep4_phi     66885 non-null  float32
 29  lep4_eta     66885 non-null  float32
 30  lep4_theta   66885 non-null  float32
 31  lep4_px      66885 non-null  float32
 32  lep4_py      66885 non-null  float32
 33  lep4_pz      66885 non-null  float32
 34  lep4_E       66885 non-null  float32
 35  lep4_tlv     66885 non-null  object
 36  mll_12       66885 non-null  float64
 37  dphi_12      66885 non-null  float64
 38  dR_12        66885 non-null  float64
 39  mll_13       66885 non-null  float64
 40  dphi_13      66885 non-null  float64
 41  dR_13        66885 non-null  float64
 42  mll_23       66885 non-null  float64
 43  dphi_23      66885 non-null  float64
 44  dR_23        66885 non-null  float64
 45  mll_14       66885 non-null  float64
 46  dphi_14      66885 non-null  float64
 47  dR_14        66885 non-null  float64
 48  mll_24       66885 non-null  float64
 49  dphi_24      66885 non-null  float64
 50  dR_24        66885 non-null  float64
 51  mll_34       66885 non-null  float64
 52  dphi_34      66885 non-null  float64
 53  dR_34        66885 non-null  float64
 54  target       66885 non-null  object
dtypes: float32(32), float64(18), object(5)
memory usage: 20.4+ MB

              lep1_pt  lep1_phi  lep1_eta  ...    dphi_34     dR_34 target
entry                                      ...
0       364078.281250  1.312494 -1.321615  ...   0.891000  0.892144          (1, ↩
    2, 3)
2        43565.238281  1.124601  1.340168  ...   2.664658  3.030723          (1, ↩
    3, 2)
3        62504.234375 -3.002433 -0.343577  ...   0.209667  1.262884          (3, ↩
    2, 1)
4        77743.296875 -1.776769 -1.809337  ...  -2.731673  3.427304          (2, ↩
```

```
        3, 1)
5       65388.453125  2.266318  2.015514  ...  2.228343  2.480432          (1, ↩
        2, 3)

[5 rows x 55 columns]
```

Listing 8.2: Inspecting the 150 GeV data set.

The data summary for the $N1 = 450$ GeV signal can be found in Appendix B. The two data summaries are very similar, but the 450 GeV signal has more events. The prints show the names and values of the features for a few events as well as their data type. They also count and print the number of non-null values and the memory usage.

Then we make a design matrix $\mathbf{X}$, containing all the features for each event, and a target vector $\mathbf{Y}$, containing all the targets for each event. The targets are at this point of type *tuples*. This makes classification more difficult, which is why we convert each event target in $\mathbf{Y}$ into an *integer* and make a new target vector $\mathbf{y}$ with the different vertex permutations as in equation 7.1.

Another useful thing to print is the individual target counts in the target vector $\mathbf{y}$ to check the number of each target in the dataframe. With this check, we quickly get an overview to see if we have a balanced or imbalanced dataset. This can be very important to check, since it might lead to problems later. The target counts for both signals are seen in Table 8.1. We easily see that there is an imbalance in both datasets where the number of target counts vary a lot between the classes, and that the 450 GeV signal has a lot more events. Vertex permutation 123 has the highest count for the 150 GeV signal, which is where the highest p_T lepton comes from the N_1 production vertex. For the 450 GeV signal 231 has the highest count, which corresponds to the case where the highest p_T lepton comes from the N_1 decay. The second highest p_T lepton comes from the final W, and the third highest comes from the N_1 production vertex. When the mass of $N_1 = 450$ GeV, a lot of momentum is released into the lepton when it decays. When $N_1 = 150$ GeV, the first lepton has more phase space and thus can typically have larger momentum.

Vertex permutations	$N1$	
	150 GeV	450 GeV
123	26801	34303
132	9716	10863
213	12871	65308
231	8454	139686
312	4013	3938
321	5030	5338

Table 8.1: The target counts for both signal samples. For the 150 GeV signal, the highest target counts is for vertex permutation 123. For the 450 GeV signal, 231 has the highest count.

Correlations

An important descriptive statistic for data analysis with multi-variable data is the *correlation matrix* using Seaborn[64]. It is a symmetric table of size $k \times k$, for k features (this includes the targets as well), with pairwise correlations between the features in the data. It summarizes the relationships between the features. With machine learning, it is also an early preprocessing step that can give some information to whereas dimensionality reduction might come in handy when dealing with high-dimensionality data. The closer to 1 the correlation coefficients are, the more correlated they are. We don't want a value close to -1, since this indicates a strong negative correlation. The diagonal will of course always be 1, since it is the correlation between the feature itself. This helps us exclude features that worsen the predictions. The optimal case is then to have as many feature correlations around 0 as possible

By using the correlations between the features, we print the feature pairs with strong correlation (magnitude greater than 0.7) for the $N1 = 150$ GeV signal:

```
lep1_theta  lep1_eta     -0.982589
lep2_theta  lep2_eta     -0.980482
lep3_eta    lep3_theta   -0.979638
lep4_eta    lep4_theta   -0.967329
lep2_py     lep1_py      -0.861290
lep2_px     lep1_px      -0.832358
lep1_pz     lep1_theta   -0.821220
lep2_pz     lep2_theta   -0.798809
lep3_theta  lep3_pz      -0.788853
lep4_pz     lep4_theta   -0.701025
lep3_phi    lep3_py       0.707593
lep1_pt     lep3_pt       0.712439
mll_13      lep2_pt       0.719664
            lep3_pt       0.753440
mll_12      mll_13        0.784307
lep4_pz     lep4_eta      0.812571
lep1_pt     mll_13        0.827285
lep2_pz     lep2_eta      0.863876
lep3_eta    lep3_pz       0.868294
lep1_pz     lep1_eta      0.872948
lep1_pt     mll_12        0.900048
mll_12      lep2_pt       0.900361
lep2_pt     lep1_pt       0.914949
```

Listing 8.3: Correlation between the features with magnitude grater than 0.7 for 150 GeV signal dataset.

We print the same for the 450 GeV signal yielding similar results:

```
lep1_eta    lep1_theta   -0.991685
lep4_eta    lep4_theta   -0.987881
lep3_eta    lep3_theta   -0.984183
lep2_eta    lep2_theta   -0.984020
lep1_pz     lep1_theta   -0.894602
lep1_py     lep2_py      -0.890322
lep2_px     lep1_px      -0.851323
lep2_pz     lep2_theta   -0.849920
lep3_pz     lep3_theta   -0.805985
lep4_theta  lep4_pz      -0.778867
```

```
        dR_14       dR_13       −0.759559
        dR_13       dR_23       −0.715678
                    dphi_23     −0.709584
        lep2_py     lep1_phi    −0.707875
        lep4_phi    lep4_py      0.704739
        lep3_py     lep3_phi     0.718642
        dR_24       mll_24       0.730663
        lep2_py     lep2_phi     0.735340
        lep3_pt     mll_13       0.752684
        mll_14      dR_14        0.775010
        lep1_phi    lep1_py      0.783622
        dphi_23     dphi_12      0.783714
        dR_13       mll_13       0.805792
        lep4_eta    lep4_pz      0.806726
        lep1_pt     lep2_pt      0.850018
        mll_12      lep1_pt      0.862141
                    lep2_pt      0.864872
        lep3_pz     lep3_eta     0.872818
        lep2_pz     lep2_eta     0.898086
        lep1_pz     lep1_eta     0.924052
```

Listing 8.4: Correlation between the features with magnitude grater than 0.7 for 450 GeV signal dataset.

The features we have printed for the two signals shows features that have strong correlations, either positive or negative correlations, between the pairs. These feature correlations may affect the predictions and is something we do not want. For this reason we may want to remove some of these features. The full correlation matrices can be seen in Figures C.1 and C.2 in Appendix C.

To take a closer look at the correlations in the dataset, we use the mutual information of the features from section 6.5.1. By using the *mutual_info_classif* function by Scikit-Learn, we can easily compute the information gain of the features and the targets. We want the features that maximizes the information gain. This helps us find which unnecessary features to remove before classification. The full information gains for all features can be seen in Appendix C.

From the strong correlation pairs, correlation matrices and the mutual information we choose to remove the eta features for all the four particles in both signals. They show high correlations to other features and have low mutual information, as seen in Table 8.2. lep1_pt shows a high correlation with other features, but has one of the highest values for mutual information with the target, lep1_pt ≈ 0.2766, for the 150 GeV signal. This is why we do not remove it. In the case of decision trees, the information gain is used for the splitting of trees.

8.1.2 Resampling

After the feature selection, we make a function using the Imblearn[65] library for imbalanced data. We want a balanced dataset to ensure that the classification model does not favor some classes due to an insufficient amount of data. With resampling we ensure that the classes have almost the same amount of data to be trained on. The Imblearn library allows us to choose between the options of both oversampling and undersampling, or only one of them. This function can be seen in Listing 8.5 and will balance the data by first

$N1$	150 GeV	450 GeV
lep1_eta	0.0617	0.4363
lep2_eta	0.0604	0.4218
lep3_eta	0.0560	0.4210
lep4_eta	0.0649	0.4259

Table 8.2: Table for the mutual information of the eta variables for the leptons for both $N1 = 150$ GeV and $N1 = 450$ GeV signals. They all show very low values, indicating low correlations between these features and the target. For 450 GeV the values are higher, but they are still among the smaller compared to the highest with mll_{23} : 0.6203

sampling the information we already have, then resample the dataset. Undersampling is used to decrease the size of the samples for one or more classes, while the oversampler increase the size of the samples for one or more classes. The *random_state* is used to reproduce the data when necessary, since the sampling algorithms will differ each time they are run.

```
    """Resample the data to make the datasets more balanced."""
    def Resample(X, y, under=False, over=False):
        if under == True:
            print("Undersample")
            undersample = RandomUnderSampler(sampling_strategy="majority", ↩
                random_state=42)
            X, y = undersample.fit_resample(X, y)

        if over == True:
            print("Oversample")
            oversample = ADASYN(sampling_strategy="not majority", random_state=42)
            X, y = oversample.fit_resample(X, y)

        #print(y.target.value_counts())  # Print the counts of the different classes↩
             after resampling
        return X, y
```

Listing 8.5: Function for resampling and balancing the amount of data.

By looking at the target counts in section 8.1, we see that both datasets are imbalanced. For the 150 GeV data we use only oversampling with the ADASYN algorithm in Scikit-Learn to create more data depending on the distribution of the classes we will oversample. All the classes except the class with highest count, the majority class, will be sampled with ADASYN. The new target counts after resampling are seen in Table 8.3.

For the 450 GeV signal we have a lot more data than the other signal, but the data is still very imbalanced. We balance the data by first undersample the majority class with a RandomUnderSampler algorithm and then oversample the minority classes with the ADASYN algorithm. The RandomUnderSampler takes random samples from the majority class to produce a subset of the data with approximately the size of the biggest minority class. We then get two majority classes with approximately the same size before the ADASYN algorithm oversample the minority classes. The new target counts after

resampling are seen in Table 8.3.

Vertex permutations	$N1$	
	150 GeV	450 GeV
123	26801	60667
132	25198	65352
213	26088	65308
231	26122	65263
312	27109	64924
321	27527	64600

Table 8.3: The target counts for both signal samples after using resampling techniques on the datasets. Oversampling is used for the 150 GeV signal, while both undersampling and oversampling is used on the 450 GeV signal.

8.1.3 Train, validation and test sets

Regardless of resampling or not, one important thing we have to do with our data when doing classification is to split the data into multiple sets. We split the design matrix $\mathbf{X}$, containing the features, and the target vector $\mathbf{y}$ into three new sets each. This is done by using a Scikit-Learn function called *train_test_split*, as seen in Listing 8.6. First we split $\mathbf{X}$ and $\mathbf{y}$ into training and test sets. The training sets are then further split into new smaller training sets and validation sets. We choose the splits to have 60% of the data as training data, 20% are validation data and 20% are test data. The validation set is used to tune the classification models, while the test set is only used as unseen data in the end when we have a good enough trained model.

```
        """Split events into training, validation and test sets."""
        X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, ↩
            random_state=42, stratify=y)

        X_train, X_val, y_train, y_val = train_test_split(X_train, y_train, test_size↩
            =0.25, random_state=42)
```

Listing 8.6: Splitting the data.

8.1.4 Scaling

The next technique we will apply is scaling of the data. We will use *standardization* of the data, which means we transform the values with a mean of 0 and a standard deviation of 1. This will fix any unwanted weighting favoring some features. Scikit-Learn has a function for doing this called *StandardScaler*. We will both fit and transform the training

data, meaning that we both compute the mean and standard deviation to standardize the training set. We have to transform both the validation and test sets with the scaler, but we don't fit them. In Listing 8.7 we use the *fit_transform* on the training set, while only using *transform* on the validation and test sets. Note that we only scale the features, since scaling the targets will assign a distribution to the categorical features.

```
        """Scale the data when called."""
        def scaler(X_train, X_val, X_test):
            sc = StandardScaler()
            X_train = sc.fit_transform(X_train)
            X_val = sc.transform(X_val)
            X_test = sc.transform(X_test)
            return X_train, X_val, X_test
```

Listing 8.7: Function for scaling data.

8.2 Training the Classification Models

With the input data properly balanced and scaled we will use the two signals to train several classification models to find the one with the best performance. We start with the training of the classification models on the validation set with various hyperparameters. We create a useful function that uses the *RandomizedSearchCV* function in Scikit-Learn to test several different values of hyperparameters for some chosen model using a randomized search with cross-validation. This is much easier than changing one hyperparameter at a time for each run, since the randomized search function can test several hyperameters in one run. Our function in Listing 8.8 prints the results of the randomized search given some set(s) of hyperparameters. The results include the mean test scores for each set of hyperparameters, and the best mean test score, with the corresponding hyperparameters.

```
        def getTrainScores(gs):
            # Function that prints the RandomizedSearchCV best parameters and mean ↩
                scores
            print("Start getTrainScores:")
            gs.fit(X_train, np.ravel(y_train))
            results = {}
            runs = 0
            for x,y in zip(list(gs.cv_results_['mean_test_score']), gs.cv_results_['↩
                params']):
                results[runs] = 'mean:' + str(x) + 'params' + str(y)
                runs += 1
            best = {'best_mean': gs.best_score_, "best_param":gs.best_params_}
            print(results)
            print(best)
            return results, best
```

Listing 8.8: Function for training models using a randomized search function.

After training all the models with the randomized search function and implementing the best hyperparameters for each model, we use the validation set to test and compare the classification models. To evaluate and compare the models, we use the accuracy score

on both the training and validation sets. This lets us see if we have any overfitting when the training accuracy score is much higher than the validation accuracy score. It is the accuracy score and the confusion matrices we will use as the main evaluation methods to check the model performances on the validation set. We also look at the variance and bias of the models. The XGBoost and LGBM models lets us plot the errors and log losses to see the convergence and fitting of these two models. The best overall performing model for each signal sample will be chosen for further use.

8.2.1 Choosing the Best Performing Models

In Table 8.4 the values of the evaluation metrics after tuning each model of the 150 GeV model is shown. There we see the accuracy score of both the validation and training sets, the balanced accuracy score, the variance and the bias for each classifier. We see the same evaluation metrics for the same classifiers trained on the 450 GeV signal in Table 8.5. From the tables we see that the most accurate classifiers for both signals are the XGBoost with accuracy scores 0.863 and 0.950 and LGBM classifiers with accuracy scores 0.877 and 0.954.

Model	Score	Score_train	BAcc	Var	Bias
LogRegCV	0.410274	0.412666	0.409277	5937.2669	6102.1446
DecisionTree	0.608518	0.795671	0.607949	6033.3890	6086.2548
AdaBoost	0.851900	1.000000	0.850790	5796.7615	6088.7001
RandomForest	0.765558	0.911612	0.764617	5751.0957	6136.3113
OvR	0.774592	0.930866	0.773828	5819.3619	6123.2995
OvO	0.778810	0.929900	0.777738	5841.6059	6135.7178
MLP	0.822657	0.949238	0.822251	6045.2875	6044.7911
HGBC	0.786301	0.899881	0.785906	6023.6668	6058.7670
XGBoost	0.863106	0.999769	0.862487	6016.7689	6053.3081
LGBM	0.877868	0.999926	0.877134	6046.5238	6055.6849

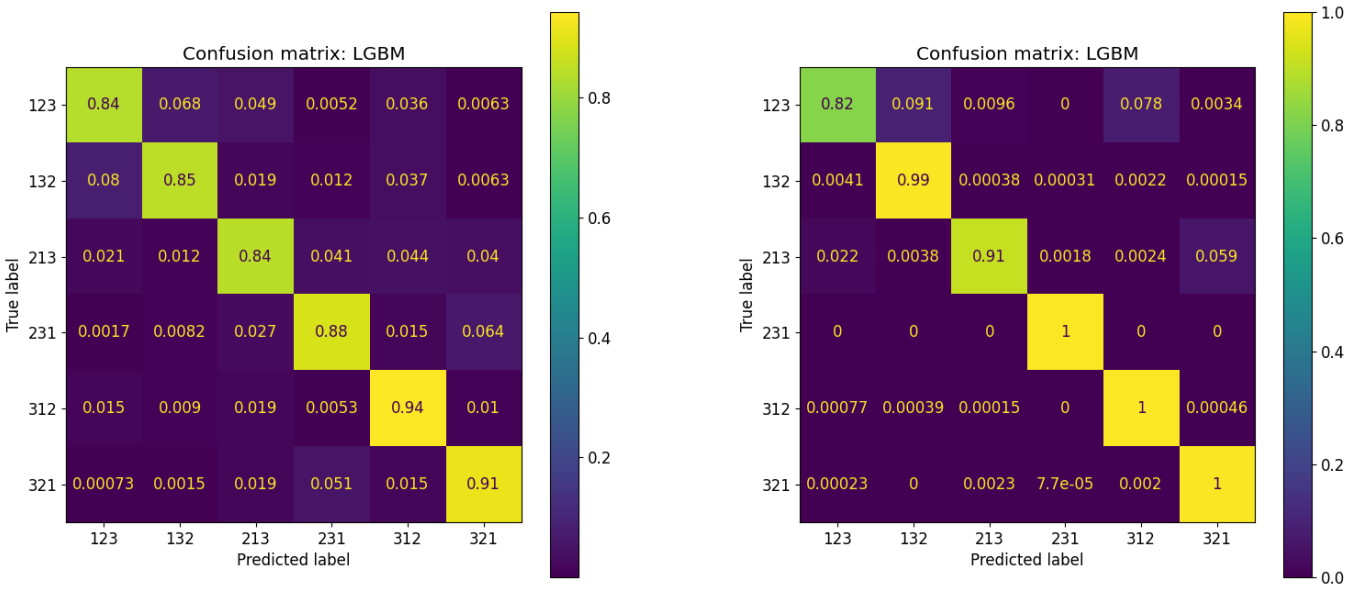
Table 8.4: Table containing evaluation values with the 150 GeV signal validation set of the classification models in section 6.4. From left to right: The classification model (names), accuracy score of validation set, accuracy score of training set, balanced accuracy score, variance and bias.

Model	Score	Score_train	BAcc	Var	Bias
LogRegCV	0.699027	0.698413	0.696966	5994.8327	5975.0053
DecisionTree	0.850938	0.916329	0.848865	6048.3255	5953.8820
AdaBoost	0.938490	1.000000	0.937000	6107.4315	5948.8834
RandomForest	0.899343	0.922821	0.896808	6057.5101	5972.7990
OvR	0.906038	0.931635	0.903435	6045.4265	5972.2990
OvO	0.908771	0.931885	0.906208	6046.0539	5971.0928
MLP	0.934993	0.960599	0.933578	5984.1467	5950.7039
HGBC	0.928027	0.957107	0.925760	5980.3233	5961.7443
XGBoost	0.950883	0.999879	0.949377	6005.1208	5951.0907
LGBM	0.954081	0.999922	0.952588	5999.1799	5950.8842

Table 8.5: Table containing evaluation values with the 450 GeV signal validation set of the classification models in section 6.4. From left to right: The classification model (names), accuracy score of validation set, accuracy score of training set, balanced accuracy score, variance and bias.

The confusion matrix for each classifier is also considered when we choose the best performing model for each signal. Like the evaluation metric tables of the classifiers, the LGBM has the best confusion matrices for both signals. The confusion matrices for the two signals are seen in Figure 8.1. The confusion matrices for the rest of the classifiers are found at the GitHub-repository in the *Plots*-folder. With these confusion matrices we get to see the individual prediction accuracy for each class, which gives more info than just the accuracy score on the whole set. The confusion matrices for the LGBM all show individual class accuracy scores bigger than 0.8, and most of them are bigger than 0.84 which is a good indication for a good classification model.

Based on this evaluation of the validation set on the models, we will use the LGBM as the preferred model further for both signals.



(a) 150 GeV signal: Confusion matrix

(b) 450 GeV signal: Confusion matrix

Figure 8.1: Left: Validation set confusion matrix of the LGBM classifier trained on the 150 GeV signal. The 150 GeV model seems to be better at predicting the 231, 312 and 321 classes with predicted accuracy scores bigger than 0.88 for these three classes. Overall for all classes, the model predicts all classes with accuracy scores bigger than 0.84. Right: Validation set confusion matrix of the LGBM classifier trained on the 450 GeV signal. The 450 GeV signal model predicts all the classes at 0.91 and better, except the 123 class with only 0.82.

8.3 Classification with Test Set

After choosing the best performing model in section 8.2 for each signal, we do a new evaluation using the test set. We use the evaluation metrics from section 6.5 for evaluating the classification model performance to check that the performances of the best model is satisfying.

8.3.1 Evaluation of the Best Models

From the the model evaluation with the validation set in section 8.2, we choose the LGBM as the best performing model for both signal samples since it has the highest accuracy scores of the classes. The LGBM model is then evaluated on the test set for both signals.

When the performances of the best model is good enough, we use a Scikit-Learn module

called *Pickle* to save the model to separate .pkl-files for the signals. These files can be loaded and exported to be used on new unseen data with the same features we have trained on. This quick and easy way of loading already trained models lets us skip the training of the model such that we can go straight to predicting the outcomes on the new data.

150 GeV signal

With the test set we use more evaluation metrics to evaluate the LGBM model. We start by looking at the classification report of the LGBM model trained on the 150 GeV signal in Table 8.6. All values for precision, recall and f1-score are higher than 0.8, which is an indicator that the LGBM model is a good classifier for the 150 GeV signal. All classes seems to be predicted satisfactory and the model has a high accuracy score of 0.88. This can also be seen in the confusion matrix in Figure 8.2 which has very similar prediction scores compared to the validation set confusion matrix (Fig. 8.1).

Vertex permutation	Precision	Recall	F1-score	Support
123	0.88	0.84	0.86	5360
132	0.90	0.85	0.87	5040
213	0.86	0.85	0.85	5218
231	0.89	0.88	0.88	5224
312	0.87	0.94	0.91	5422
321	0.89	0.91	0.90	5505
accuracy			0.88	31769
macro avg	0.88	0.88	0.88	31769
weighted avg	0.88	0.88	0.88	31769

Table 8.6: Classification report of the LGBM model trained on the 150 GeV signal with the test set. All classes show high scores for precision, recall and f1-score, except for recall on the 123 vertex. The high scores indicate a good classification model.

The evaluation metrics of the model are seen in Table 8.7. The accuracy score, CKS and balanced accuracy score all have score higher than 0.85, showing that the LGBM model is satisfactory trained and performs well. The high accuracy score of the training set might indicate some overfitting and the log loss is at a respectable level, but the model still performs good enough.

We also plot the precision-recall curve and ROC curve. The ROC curves in Figure 8.3 all show AUC scores around 0.99. An AUC score higher than 0.8 is usually considered a good model. The precision-recall curve is seen in Figure 8.4 and shows AUC scores higher than 0.92.

Finally, we take a look at the 20 most important features decided by the LGBM model, seen in Figure 8.5. The invariant mass pairs are clearly the most important features when

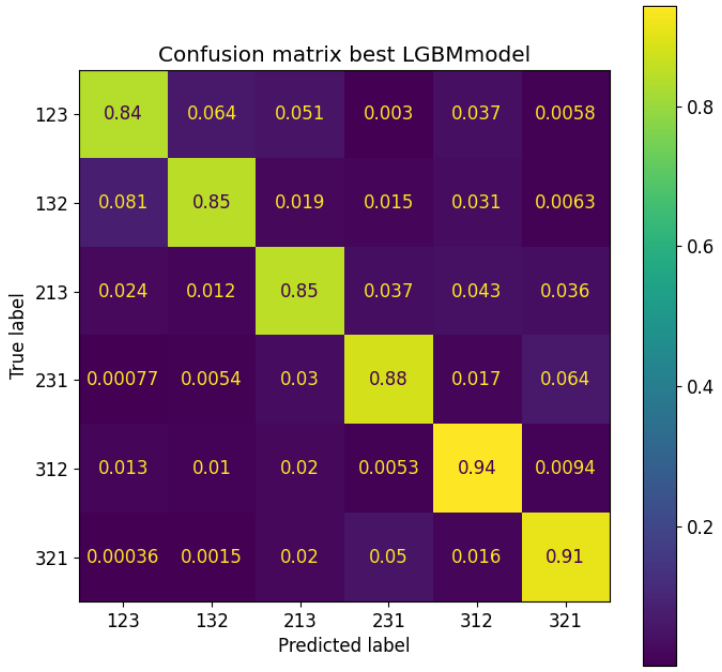


Figure 8.2: Test set confusion matrix of the LGBM classifier trained on the 150 GeV signal. The model on the test set shows similar accuracy scores for the classes like it did with the validation set, where all classes are predicted with 0.84 or higher.

Score	Score_train	CKS	BAcc	LogLoss	Var	Bias
0.879285	0.999935	0.855092	0.878572	0.333487	6033.5629	6054.7185

Table 8.7: Table containing evaluation values with the 150 GeV signal test set of the LGBM model in section. From left to right: The accuracy score of test set, accuracy score of training set, the CKS score, balanced accuracy score, log loss, variance and bias.

predicting with the model, with mll_{12} being the most important one. Other important features are $dPhi$, dR and E. Thus the variables we have added to the data with the angular variables and invariant masses of pairs of particles, have high importance when predicting.

The performance of the LGBM on the 150 GeV signal is proven to be very good, and is saved with Pickle to be used later.

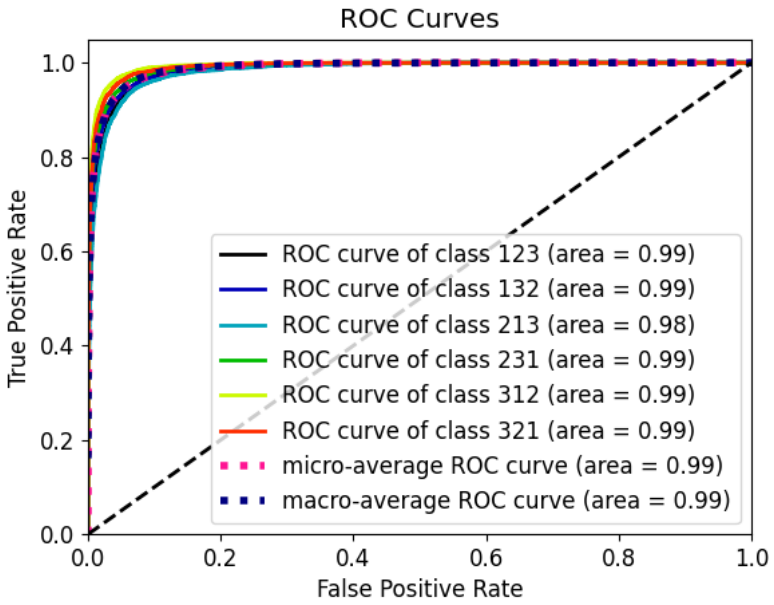


Figure 8.3: ROC curve plot for the LGBM model with the 150 GeV signal test set. The AUC for all classes are around 0.99, which shows that the model is a very good model for predicting the classes.

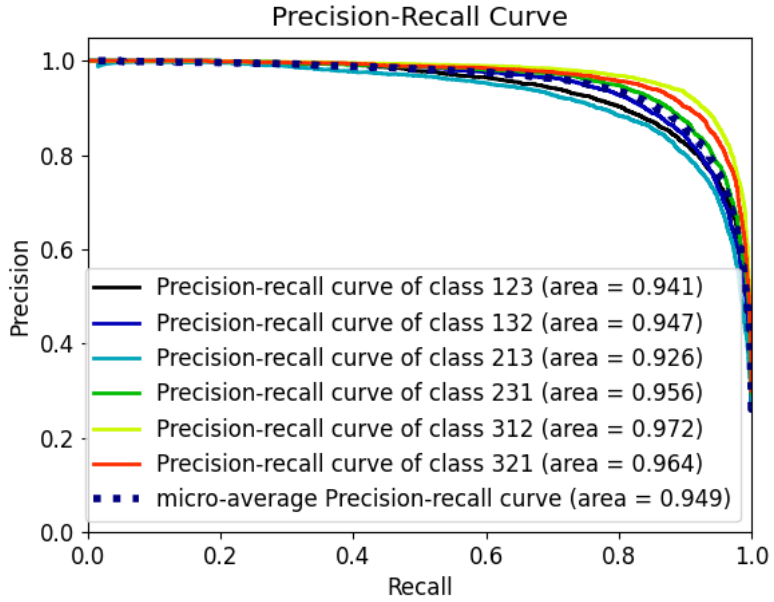


Figure 8.4: Precision-recall curve plot for the classes predicted by the LGBM model with the 150 GeV signal test set. All classes and the average have AUC higher than 0.92, once again showing the good performance of the model.

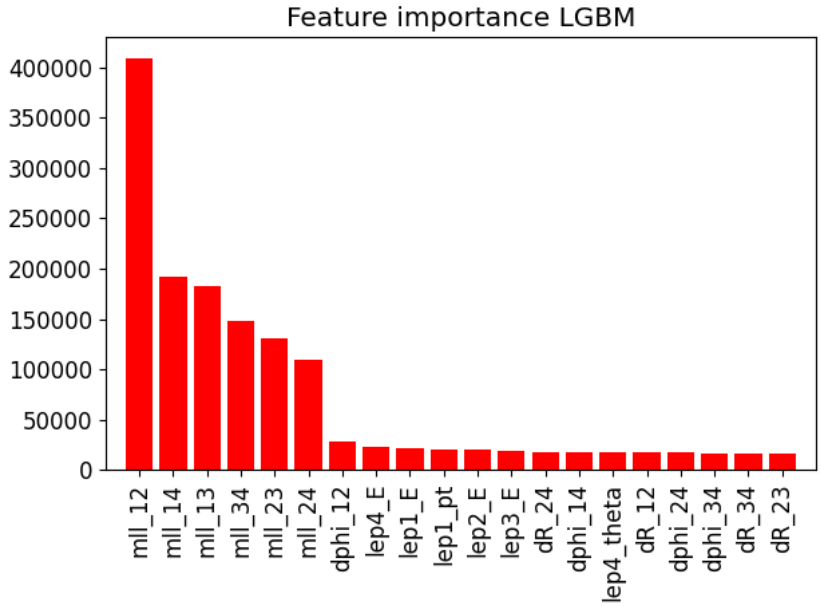


Figure 8.5: The most important features decided by the LGBM model trained on the 150 GeV signal for predicting new data. The features we added to the data with the angular variables and invariant masses of pairs of particles show high importance when making predictions.

450 GeV signal

For the 450 GeV signal, the classification report in Table 8.8 shows similar results like the 150 GeV signal, but better. Almost all values are around 0.9 or higher, except for the recall for the 123 vertex, as seen for the 150 GeV signal. This indicates that the LGBM model trained on the 450 GeV is a very good classifier. The confusion matrix for the 450 GeV signal model in Figure 8.6 is very similar to the validation set, with all classes higher than 0.92 and an accuracy score of 0.96. Compared with the 150 GeV confusion matrix, the 450 GeV trained model is better on predicting all the classes except the 123 vertex, for which the two models perform equally.

The evaluation metrics in Table 8.9 show a high accuracy score, CKS and balanced accuracy, not to far away from the accuracy score of the training set. This means that we are less likely of having overfitting since the two accuracy scores are close in value. The log loss is also less than for the 150 GeV signal.

The ROC curves in Figure 8.7 are better than the 150 GeV signal, with AUC around 1.0 for all classes. The precision-recall curve in Figure 8.8 is also better, with all AUC values higher than 0.97. These plots indicate a very good classification model on these data.

The 20 most important features decided by the LGBM model as seen in Figure 8.9. It is still the invariant mass pairs that are clearly the most important features when predicting with the model, with mll_{14} now the most important one. Other important features are

Vertex permutation	Precision	Recall	F1-score	Support
123	0.97	0.83	0.89	12133
132	0.92	0.99	0.96	13070
213	0.99	0.92	0.95	13062
231	1.00	1.00	1.00	13053
312	0.93	1.00	0.97	12985
321	0.94	0.99	0.97	12920
accuracy			0.96	77223
macro avg	0.96	0.96	0.96	77223
weighted avg	0.96	0.96	0.96	77223

Table 8.8: Classification report of the LGBM model trained on the 450 GeV signal with the test set. All classes show high scores for precision, recall and f1-score, except for recall on the 123 vertex. The high scores indicate a very good classification model.

Score	Score_train	CKS	BAcc	LogLoss	Var	Bias
0.956982	0.999914	0.948356	0.955582	0.111992	5997.1512	5948.8764

Table 8.9: Table containing evaluation values with the 450 GeV signal test set of the LGBM model in section. From left to right: The accuracy score of test set, accuracy score of training set, the CKS score, balanced accuracy score, log loss, variance and bias.

here $dPhi$, dR and E.

The performance of the LGBM on the 450 GeV signal is proven to be very good and even better than the 150 GeV signal. This model is also saved with Pickle to be used later.

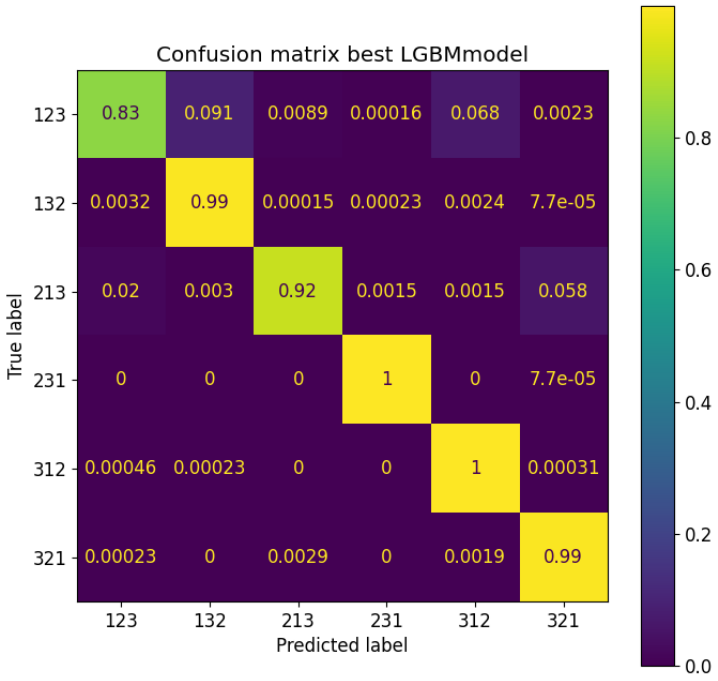


Figure 8.6: Test set confusion matrix of the LGBM classifier trained on the 450 GeV signal. The model on the test set shows similar accuracy scores for the classes like it did with the validation set, where all classes are predicted with 0.9 or higher except for the 123 vertex permutation class.

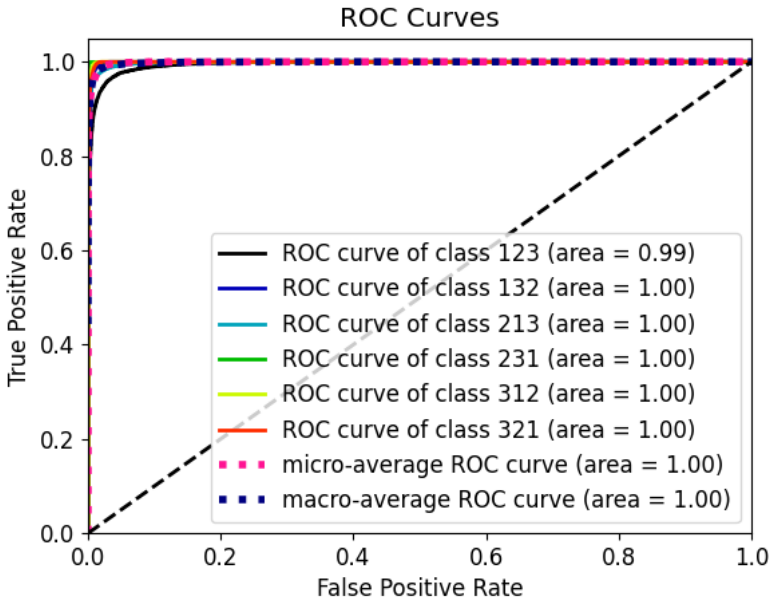


Figure 8.7: ROC curve plot for the LGBM model with the 450 GeV signal test set. The AUC for all classes and averages are around 1.0, which shows that the model is a very good model for predicting the classes.

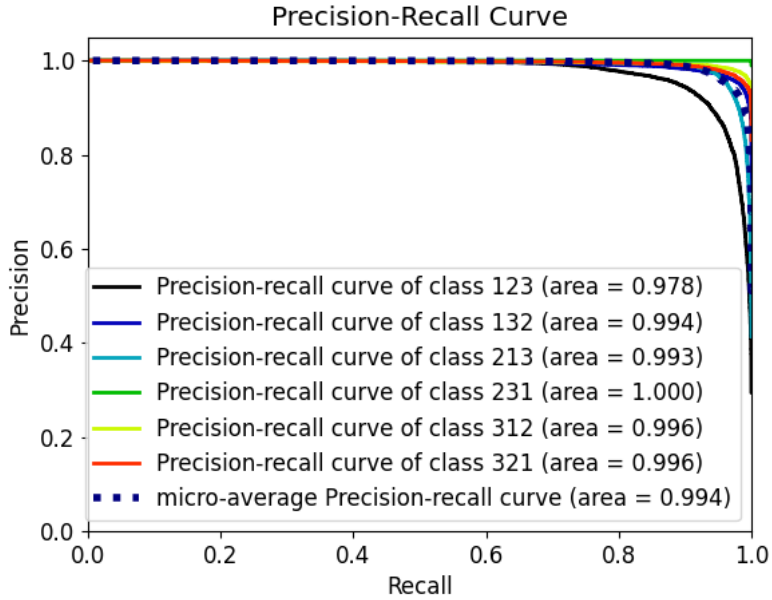


Figure 8.8: Precision-recall curve plot for the classes predicted by the LGBM model with the 450 GeV signal test set. All classes and the average have AUC higher than 0.97, once again showing the good performance of the model.

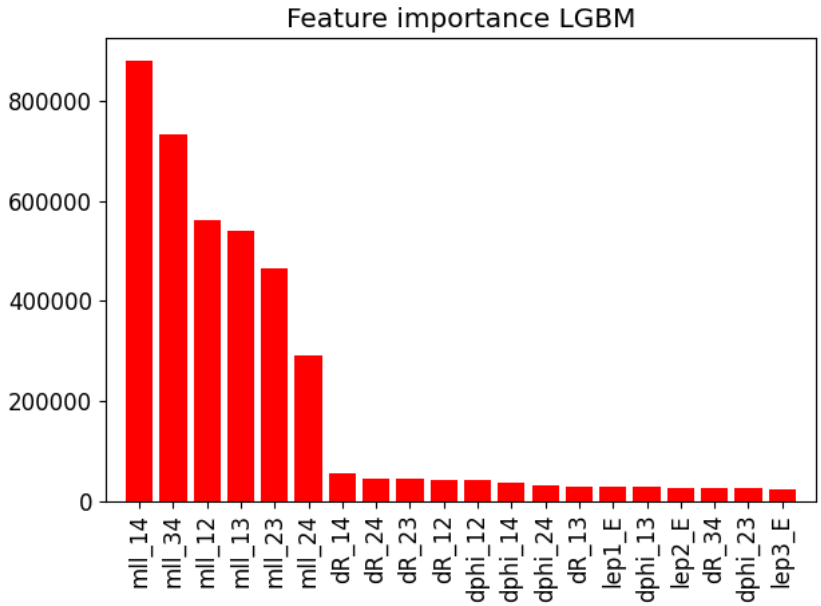


Figure 8.9: The most important features decided by the LGBM model trained on the 450 GeV signal for predicting new data. The features we added to the data with the angular variables and invariant masses of pairs of particles show high importance when making predictions also in this case.

Part III

Results

Chapter 9

Classification Results

To classify leptons in background and signal Ntuples, we first trained classification models on two signals samples with truth data and chose the best performing model for each signal sample. Then we evaluated the performance of these models with classification evaluation metrics. With the models saved, we can classify background and signal Ntuples from section 7.2 after they've been through all the steps of the data and MC processing chain in Fig. 7.1. We present the classification results in this chapter. Discussions of the results are done in chapter 10.

9.1 Ntuple Classification

The models have been trained on two samples of signal data with truth variables for classifying leptons. Now we want to classify and predict the lepton vertices in various background data which has been simulated, digitized and paved through detector reconstruction to give a good representation of the data. The background Ntuples are originally ROOT-files which have to be converted to dataframes with the same features like the signals, except for the truth. We use a new script for classification of the backgrounds and signal Ntuples, called *Trilepton_classify_Ntuples.py*. This time we do not create the target variables since the backgrounds do not contain truth data. We also do this with the two signal samples, containing the same events as the samples used in training/validation/test, but containing data after reconstruction.

After converting the backgrounds and signals to dataframes, we load the files containing the trained best models for the two signals and use them to classify the background and signal lepton vertices. The predicted vertex permutations for the two signals are seen in Tables 9.1 and 9.2. Table 9.1 contains the predicted counts of the two signals with the 150 GeV trained model, while Table 9.2 contains the predicted counts of the two signals with the 450 GeV trained model. These two tables have different counts for the vertex classes, indicating that the two models predict differently. The most predicted vertices are the 123 and 132 vertices. This makes sense since the leptons are ordered after highest p_t, and in both of these cases the highest p_T lepton comes from the N_1 production vertex. The p_T of the produced particles decreases after the first vertex decay. We get one 213 predictions

of the 450 GeV signal with the 450 trained model. This may look like a misclassification case since it only happens once, but we will see that the backgrounds also have the 213 vertex predicted. The signals look to only contain events with the 123 and 132 vertices.

Classes	123	132	213	231	312	321
150 GeV	7358	4879	0	0	0	0
450 GeV	7464	5450	0	0	0	0

Table 9.1: The target counts of the predicted classes of the signal Ntuples with the LGBM model trained on the 150 GeV signal. Only predicted classes are 123 and 132.

Classes	123	132	213	231	312	321
150 GeV	7435	4802	0	0	0	0
450 GeV	7856	5057	1	0	0	0

Table 9.2: The target counts of the predicted classes of the signal Ntuples with the LGBM model trained on the 450 GeV signal. Mostly predicted classes are 123 and 132.

For the background Ntuples, we see the predicted vertex permutations in Table 9.3 for the 150 GeV trained model, and in Table 9.2 for the 450 GeV trained model. Most of the predicted vertices are still 123 and 132, like for the signals. In these cases we now get more predictions of the 213 vertex as well, and still almost none of the last three vertices in the tables. Like for the signals it makes sense that since the leptons are ordered after highest p_T, lepton 1 should at least be in the first two vertices. This is very clear for the backgrounds where lepton 1 never is predicted coming from the third vertex.

For each dataframe, the predicted outcomes are saved in the dataframe before the dataframe is converted to a (CSV) file, and then converted back into ROOT like we did earlier in section 7.4.1. The ROOT-files now contain all original variables, as well as the variables we produced and used for classification and the predicted vertex permutations.

Classes	123	132	213	231	312	321
diboson2L	547161	518339	595	0	5	0
diboson3L	2878169	2688290	844	0	2	0
diboson4L	2554621	2352298	746	0	4	0
higgs	821006	807451	437	0	5	0
singletop	410908	261725	252	0	21	0
topOther	2573115	2382427	941	0	1	0
triboson	27629	22337	3	0	0	0
ttbar	4752100	3847796	2455	0	10	0
Zjets	5157323	5821865	7923	0	62	0

Table 9.3: The predicted target counts of the backgrounds for the 150 GeV trained LGBM model. 123 and 132 are the most predicted classes with 213 predicted much less. 312 is just predicted a few times, while 231 and 321 are never predicted.

Classes	123	132	213	231	312	321
diboson2L	558950	506051	1098	0	1	0
diboson3L	2754490	2812135	677	0	1	2
diboson4L	2535186	2371997	482	0	1	3
higgs	731278	897241	380	0	0	0
singletop	330649	341791	465	0	1	0
topOther	2390539	2565389	553	0	0	3
triboson	26268	23700	1	0	0	0
ttbar	4288182	4311015	3155	0	4	5
Zjets	5596455	5379835	10871	0	7	5

Table 9.4: The predicted target counts of the backgrounds for the 450 GeV trained LGBM model. 123 and 132 are the most predicted classes with 213 predicted much less. 312 and 321 are just predicted a few times, while 231 is never predicted.

9.2 Signal Regions

We then plot the background and signal variable distributions again using the first ROOT-plotting scripts. We define some signal regions and cuts in the variables used for the plotting. The outcome classes (lepton vertices) from the classification are used to define cuts. We will use the lepton flavors, same flavor (SF) and different flavor (DF), and signs of the leptons, opposite sign (OS), for the three leptons as cuts as well since these scenarios are of most interest to us from equation 3.16. All these cuts are seen in Table 9.5. From Tables 9.1 to 9.4, the most predicted vertices are 123, 132 and 213. The other three vertices are predicted so much less, close to zero in comparison, that we would not get anything by using them as cuts. That is why we leave them out of the analysis. The DF, SF and OS are to be applied between leptons i and j depending on the vertex permutation, e.g. SF would be lep$i$_Flavor == lep$j$_Flavor. The leptons i and j are the first and second vertices in each vertex permutation, e.g. vertex 123 has i=1 and j=2.

Signal region cuts:	
Baseline leptons	nLep_base == 3
Signal leptons	nLep_signal == 3
SF & OS	lep$i$_Flavor == lep$j$_Flavor & lep$i$_Charge != lep$j$_Charge
DF & OS	lep$i$_Flavor != lep$j$_Flavor & lep$i$_Charge != lep$j$_Charge
Lepton vertices	pred_class == [123, 132, 213]

Table 9.5: Signal region cuts used for plotting variable distributions of Ntuples for backgrounds and signals with classification variables and predicted lepton vertices. Cuts to be applied where leptons i and j to have same flavor (SF) and opposite sign (OS), and leptons i and j to have different flavor (DF) and OS. The leptons i and j are the first and second vertices in each vertex permutation, e.g. vertex 123 has i=1 and j=2. Combine them with the cuts for lepton vertices.

Then we will use some (benchmark) cuts for a more "standard" analysis at $\sqrt{s} = 14$ TeV from Pascoli et al. [1] for comparison. These cuts are seen in Table 9.6. [1]

Benchmark "Standard" Analysis at $\sqrt{s} = 14$ TeV:
$m_{l_i,l_j} > 10$ GeV, $\quad |m_{l_i,l_j} - M_Z| > 15$ GeV, $\quad |m_{3l} - M_Z| > 15$ GeV, $\quad p_T^{l_1} > 55$ GeV, $\quad p_T^{l_2} > 15$ GeV, $\quad m_{3l} > 80$ GeV

Table 9.6: Cuts used for a benchmark analysis to be compared with our cuts from Table 9.5. The combinations of $l_i l_j$ are for l_1, l_2 and l_3. $M_Z = 91.2$ GeV is the mass of the Z-boson and m_{3l} is the invariant mass of the three lepton system. Reference: Table 6 in Pascoli et al. [1].

[1] We have left one cut out, $p_T^{b-Tagged}$, since this variable is not available to use in our files.

9.3 Distributions

The most interesting features we want to look at are the invariant mass of the three lepton system and the MET for both the 150 GeV and 450 GeV signals with cuts defined in Table 9.5 and 9.6. In the following plots we have the histogram distributions we have plotted earlier, e.g. like in section 7.3, in the upper part of the plots and in the lower part of the plots we have the Significance Z plots of the simulated signals to look for excess. Significance plots show the expected significance Z of the signals and quantifies the separation between the backgrounds and the signals. When the significance is high it means we have good sensitivity to the signal. The significance plots are used to see where we should cut in the variable we're looking at to maximize the significance. Typically we want to cut where the significance distribution has it's maximum. If we can reach $> 5\ \sigma$ it is possible to discover the model. If it reaches 1.37 we have sensitivity to possibly exclude the model if the data follows the SM background expectation.

In Table 9.7 and 9.8 we see the number of events for the different signal regions from table 9.5 for each backgrounds, the total number of events for the backgrounds and the two neutrino signals. After the cuts we see that the number of events for the MC and signals are much less compared to Table 9.3 and 9.4. There is also a difference between the flavors of the first and second vertex leptons SF and DF where the SF plots have distributions with much more events. The total number of background event ratios for DF/SF are seen in Table 9.9 for the three vertex permutations.

Ntuples	150 GeV model Cuts					
	123		132		213	
	SF	DF	SF	DF	SF	DF
diboson2L	189.3	55.3	65.2	52.8	0.0	0.0
diboson3L	5924.9	1963.7	4841.0	1522.2	0.7	0.2
diboson4L	3519.6	398.5	1649.1	694.0	0.2	0.1
ttbar+X	409.3	197.9	358.2	178.7	0.1	0.0
singletop	143.6	135.8	82.3	72.8	0.1	0.0
triboson	35.1	25.1	25.6	19.2	0.0	0.0
ttbar	2730.9	2718.0	1557.8	1519.4	0.7	1.0
Zjets	74606.5	2065.8	23289.2	14726.9	11.4	0.2
SM total	87559.2	7560.1	31868.3	18786.0	13.2	1.5
150 GeV	6392.4	6309.4	2869.1	2768.5	0.0	0.0
450 GeV	8658.1	8890.8	5401.7	5162.3	0.0	0.0

Table 9.7: Table for the number of events for each background, the total for backgrounds and signals with the combinations of vertex and flavor cuts for 150 GeV model. OS, baseline and signal lepton cuts are applied as before in Table 9.5.

Ntuples	450 GeV model Cuts					
	123		132		213	
	SF	DF	SF	DF	SF	DF
diboson2L	193.5	51.7	67.2	53.6	0.1	0.0
diboson3L	5634.3	1939.9	4397.4	1616.1	0.0	0.1
diboson4L	3578.3	398.2	1639.6	676.4	0.0	0.0
ttbar+X	389.2	186.3	375.9	190.0	0.0	0.0
singletop	132.4	128.1	88.5	83.0	0.0	0.0
triboson	32.7	24.6	28.4	20.3	0.0	0.0
ttbar	2532.9	2526.4	1679.5	1638.2	0.4	0.5
Zjets	75098.6	2171.9	23404.5	14226.3	69.0	0.0
SM total	87591.8	7427.1	32221.1	18504.0	69.7	0.7
150 GeV	6529.2	6359.7	2876.1	2714.0	0.0	0.0
450 GeV	9349.0	9264.7	4698.2	4706.4	0.0	0.0

Table 9.8: Table for the number of events for each background, the total for backgrounds and signals with the combinations of vertex and flavor cuts for 450 GeV model. OS, baseline and signal lepton cuts are applied as before in Table 9.5.

Model	123	132	213
150 GeV	0.0863	0.589	0.114
450 GeV	0.0848	0.574	0.042

Table 9.9: Flavor ratio, DF/SF, for the two signal models with vertex cuts showing that there are a lot more SF events than DF events for the backgrounds.

The plots for the invariant masses with DF between the leptons are seen in Figure 9.1, and the plots with SF for the leptons are seen in Figure 9.2. There are less events with the 213 vertex permutation compared to the 123 and 132 vertices. There are only a few backgrounds in the 213 vertex cut plots (Fig. 9.1e, 9.1f, 9.2e, 9.2f) that has events with this predicted vertex permutation compared with the other vertex cuts, about 1-70 events. Most of the events are around 100-200 GeV where more or less all events are below 500 GeV. The signal models favor the 123 and 132 vertex permutations with no signals in the 213 vertex permutation. In total for the backgrounds the 450 GeV model for the 213 vertex has more events compared to the 150 GeV model, $\approx$ 70 vs 13 events. There is a clear difference between the DF and SF plots where there are a lot more events in total in the SF plots as seen in Table 9.9 for the flavor ratios. Mainly because Z can not decay into electron-muon (emu) events and thus reduces the large backgrounds such as WZ and Z+jets.

The 150 GeV simulated signal in the plots have around the same number of events

like the backgrounds and where the number of events decrease when the masses increase, with some fluctuations mostly after reaching around 500 GeV. This makes the 150 GeV simulated signal more difficult to differentiate from the backgrounds, however it reaches a significance of > 2 in the DF channels. The 450 GeV simulated signal is easier to split from the backgrounds since it has more events for higher masses, from around the m_{3l} mass of 400 GeV the heavier signal becomes dominant over the backgrounds in terms of events. For the simulated signals there is not much difference between the flavors for the first and second vertex leptons with only maximum 300 events in difference for the same vertices.

The significance plots for the m_{3l} plots with the SF channel, 123 and 132 vertex regions are very similar for the 450 GeV simulated signal where it starts at 0 GeV and reaches the maximum significance over 4 σ around 250 GeV. The 150 GeV simulated signal significance increases after 100 GeV but varies a lot more depending on the signal regions. For the 123 vertex in 450 GeV model the significance of the 150 GeV signal reaches its maximum around 800 GeV, while in the 150 GeV model it stays around 2 σ from 300-950 GeV. For the 132 vertex plots the 150 GeV signal never goes beyond 2 σ. For the DF channel and the 123 vertex region the significance of the 450 GeV signal starts at almost 3 σ and reaches over 4 σ around 160 GeV for both signal models. The 150 GeV signal in the 150 GeV model starts at 2 σ and reaches beyond 4 σ around 850 GeV. In the 450 Gev model the 150 GeV signal reaches its maximum with $\sigma > 4$ between 400-800 GeV. In the 132 vertex plots the maximum significance is close to 4 σ in the 150 GeV model and 3 σ in the 450 GeV model for the 150 GeV signal in the region around 600-700 GeV. There are no signals in the 213 vertex plots which give no significance to look at.

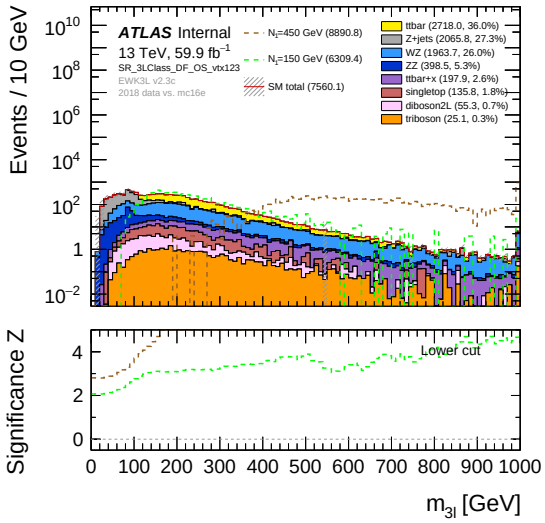
The standard analysis plot in Figure 9.3 have cuts on the invariant mass higher than 100 GeV. This plot has a total number of 214780.4 background events which is much more than the SF plots. The 450 GeV simulated signal is easier to differentiate from the backgrounds after 450 GeV while the 150 GeV simulated signal is more difficult, similar to the plots with the vertex cuts. The significance of the 150 Gev signal increases slowly towards 1 σ while the invariant mass increases to 1 TeV. For the 150 GeV model we reach higher sensitivity with our ML model than with this simplified model. The 450 GeV goes over 4 σ at around 380 GeV and stays there.

Also for MET we get that the SF plots have much more events than the DF plots. For this feature we can't differentiate as easily between the backgrounds and signals we could before for both the vertex cuts and for the benchmark cuts. All the signals seem to have the same amount of events like the backgrounds, and we do not have the same difference between the two signals either. They are much closer in number of events for this feature. For DF and vertex 123 for both signal models we have that the 450 GeV simulated signal has more events that the MC around 100-500 GeV. The only real source of MET for our signal models comes from the neutrino thus we do not expect any excess of high MET in the signal distributions. However, one does see a slightly longer tail in the MET distribution of the 450 GeV signal since it typically has more momentum available for the neutrino.

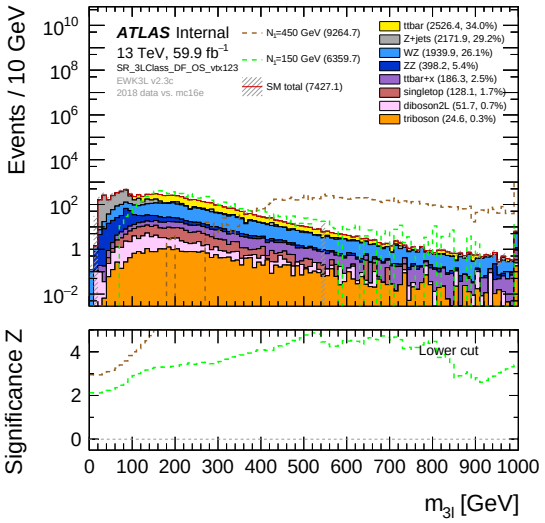
The significance for the DF and 123 vertex plots goes quickly from 3 σ to over 4 σ and

stays there for the 450 GeV signal. The 150 GeV signals starts at 2 σ and ends at 0 around 410 GeV in the 150 GeV model and around 280 GeV in the 450 GeV model because we run out of events in the signal model. The significances of the signals are much less in the MET distributions compared with the m_{3l} distributions and, as expected, does not discriminate very well the signals and backgrounds.

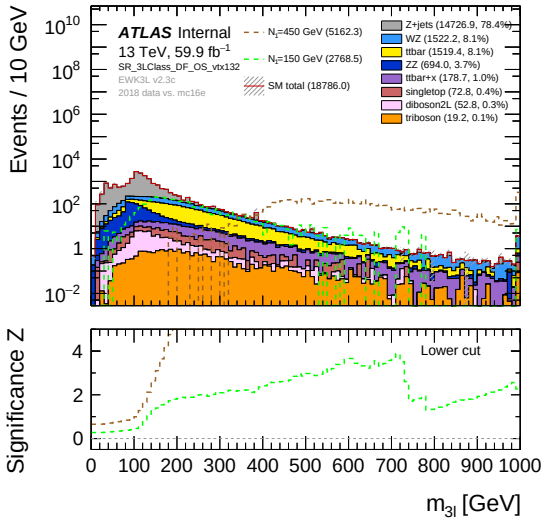
In the benchmark plot for MET the significance for the 150 GeV signal stays at 0 while it increases and ends up at 4 σ at 600 GeV for the 450 GeV signal.



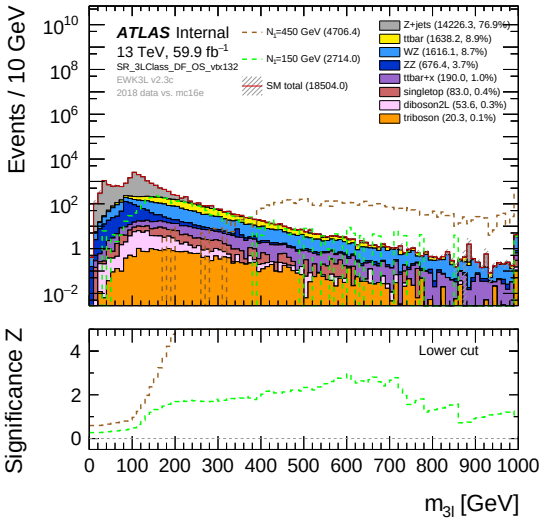
(a) 150 GeV signal: DF,OS,123



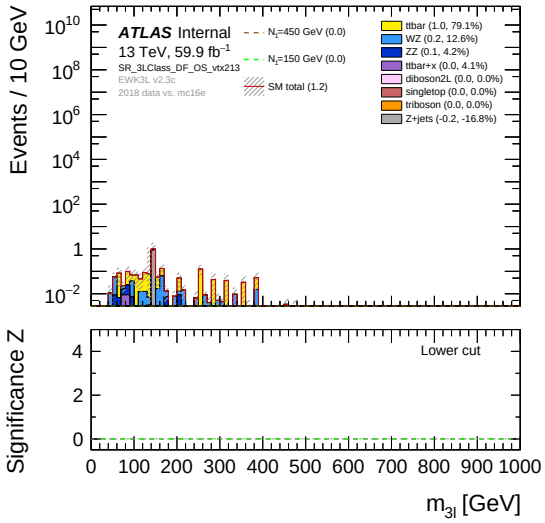
(b) 450 GeV signal: DF,OS,123



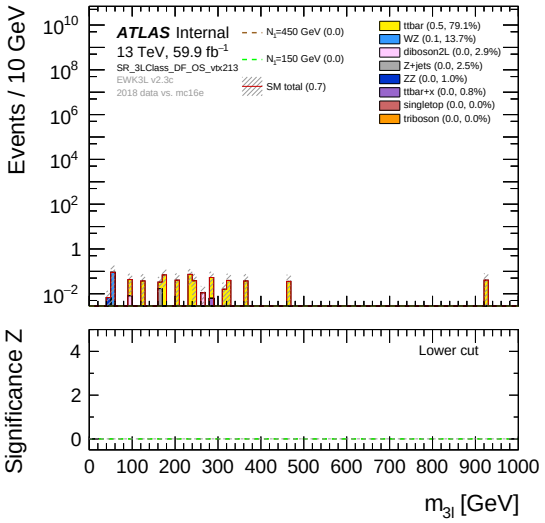
(c) 150 GeV signal: DF,OS,132



(d) 450 GeV signal: DF,OS,132

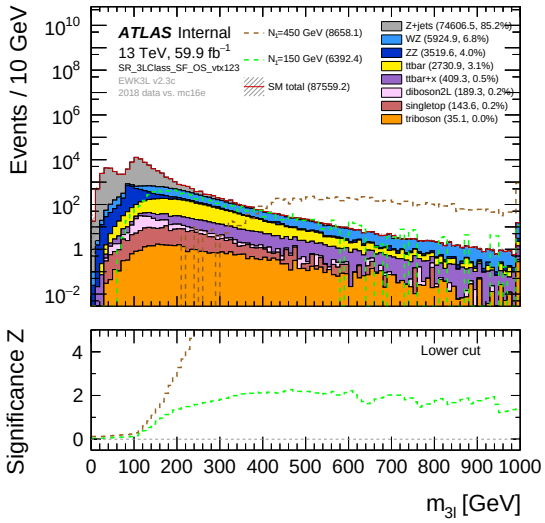


(e) 150 GeV signal: DF,OS,213

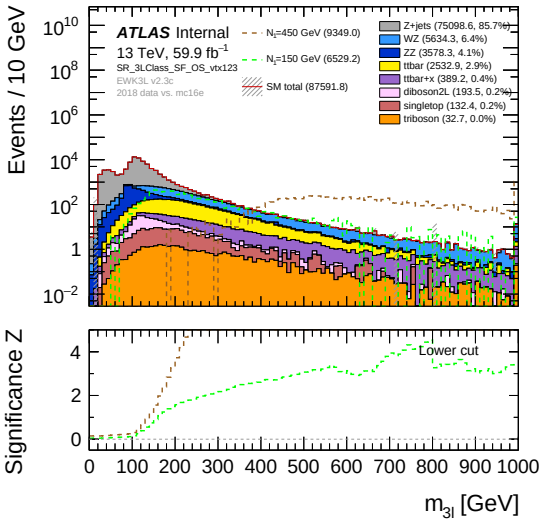


(f) 450 GeV signal: DF,OS,213

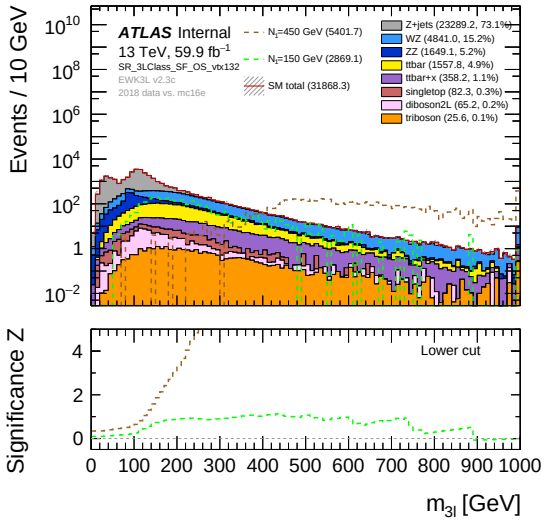
Figure 9.1: The invariant mass of the three lepton system with DF and OS cuts between lepton 1 and 2 and different vertex cuts for the two signals with masses 150 GeV (left side plots) and 450 GeV (right side plots) defined in the subcaptions.



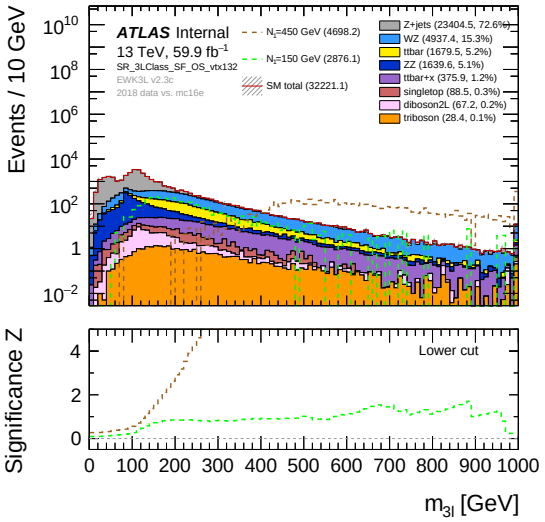
(a) 150 GeV signal: SF,OS,123



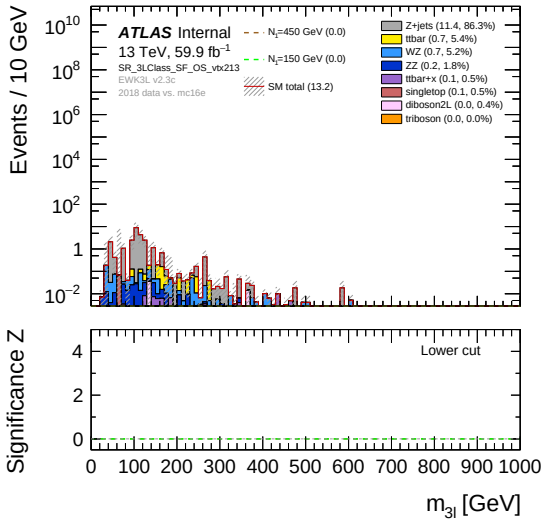
(b) 450 GeV signal: SF,OS,123



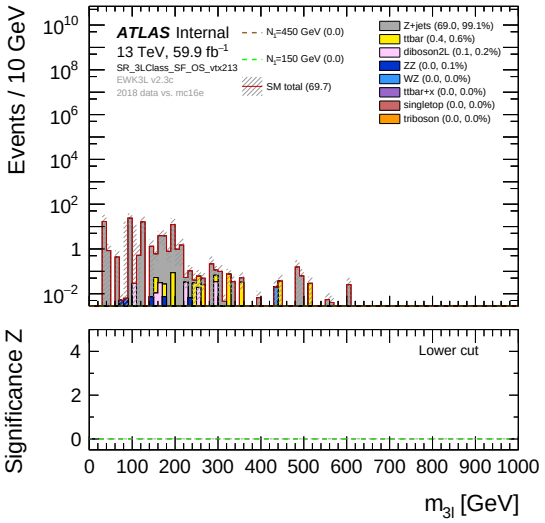
(c) 150 GeV signal: SF,OS,132



(d) 450 GeV signal: SF,OS,132



(e) 150 GeV signal: SF,OS,213



(f) 450 GeV signal: SF,OS,213

Figure 9.2: The invariant mass of the three lepton system with SF and OS cuts between lepton 1 and 2 and different vertex cuts for the two signals with masses 150 GeV (left side plots) and 450 GeV (right side plots) defined in the subcaptions.

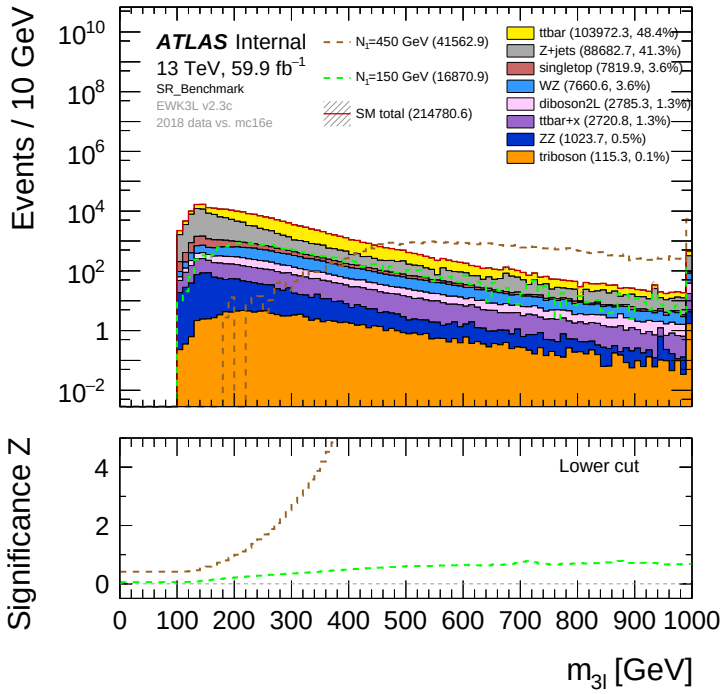
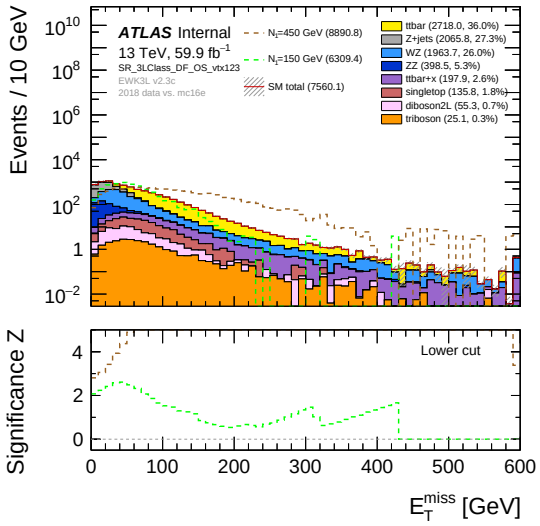
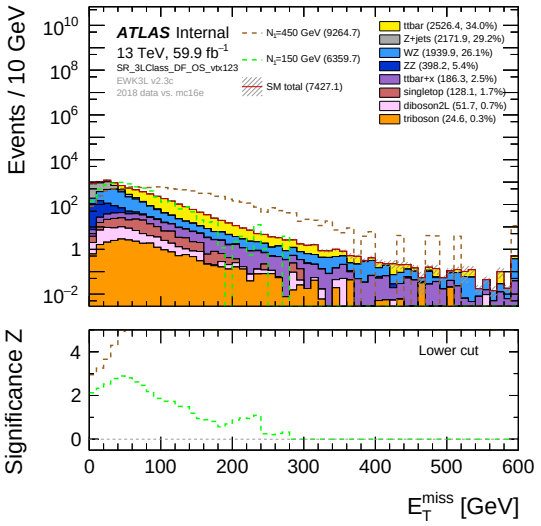


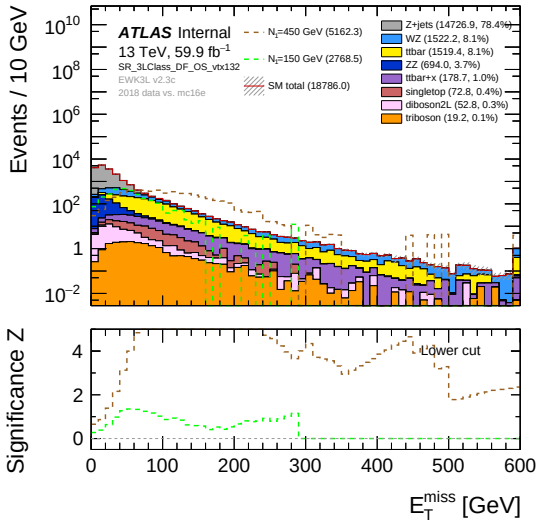
Figure 9.3: The invariant mass of the three lepton system with the benchmark cuts for a standard analysis with MC and two signals.



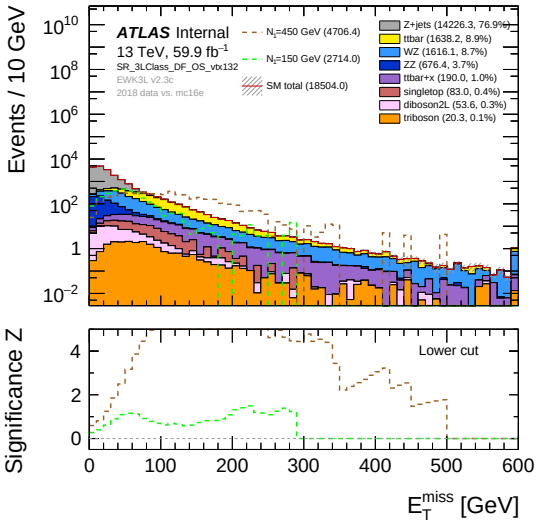
(a) 150 GeV signal: DF,OS,123



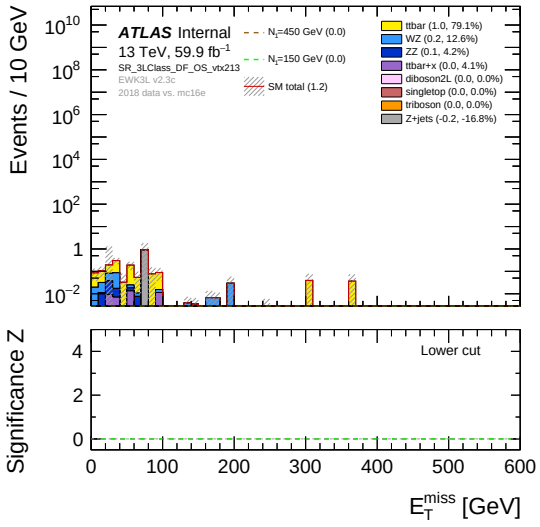
(b) 450 GeV signal: DF,OS,123



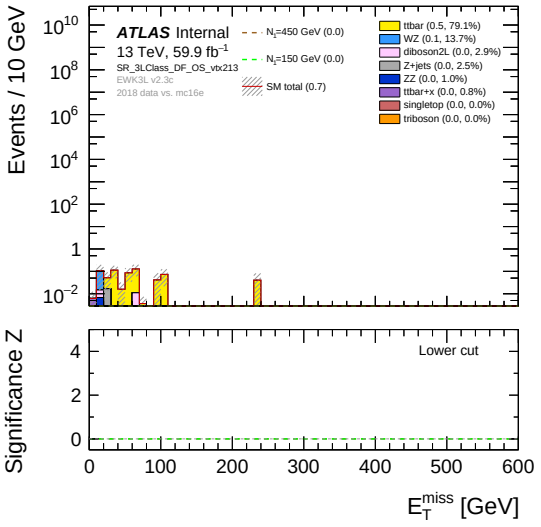
(c) 150 GeV signal: DF,OS,132



(d) 450 GeV signal: DF,OS,132

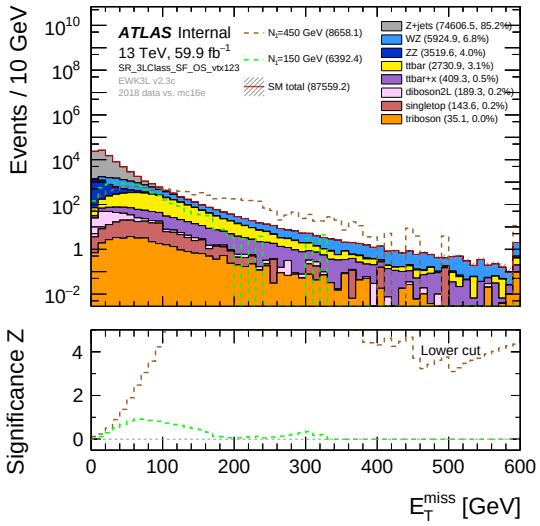


(e) 150 GeV signal: DF,OS,213

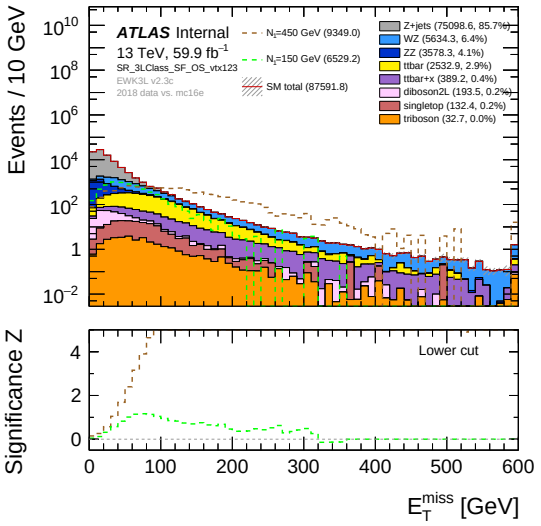


(f) 450 GeV signal: DF,OS,213

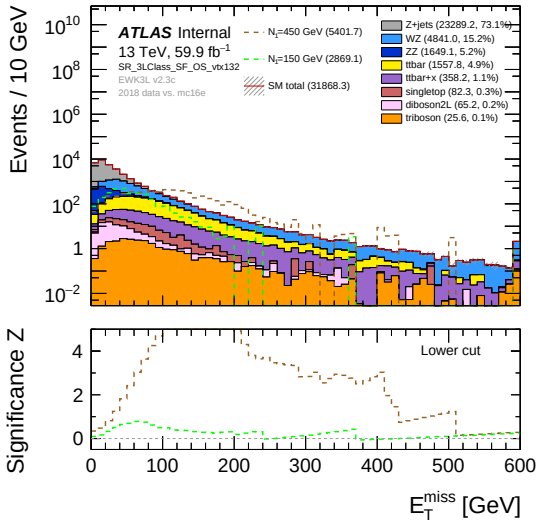
Figure 9.4: The MET with DF and OS cuts between lepton 1 and 2 and different vertex cuts for the two signals with masses 150 GeV (left side plots) and 450 GeV (right side plots) defined in the subcaptions.



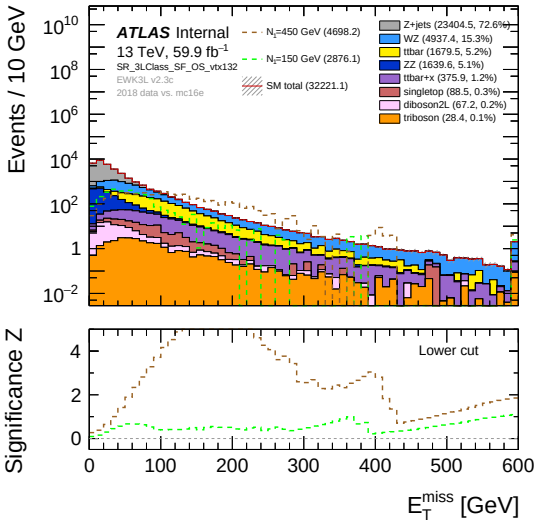
(a) 150 GeV signal: SF,OS,123



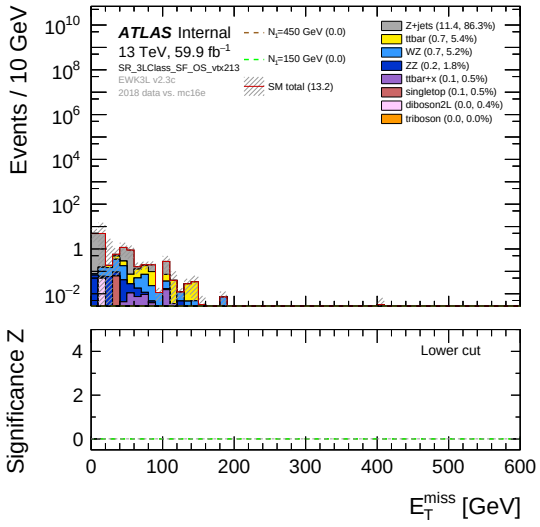
(b) 450 GeV signal: SF,OS,123



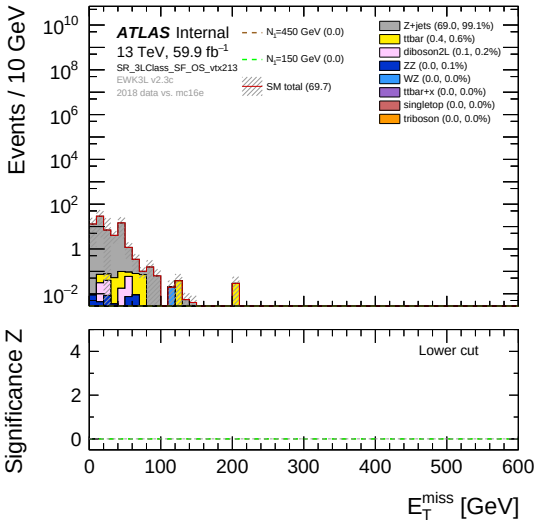
(c) 150 GeV signal: SF,OS,132



(d) 450 GeV signal: SF,OS,132



(e) 150 GeV signal: SF,OS,213



(f) 450 GeV signal: SF,OS,213

Figure 9.5: The MET with SF and OS cuts between lepton 1 and 2 and different vertex cuts for the two signals with masses 150 GeV (left side plots) and 450 GeV (right side plots) defined in the subcaptions.

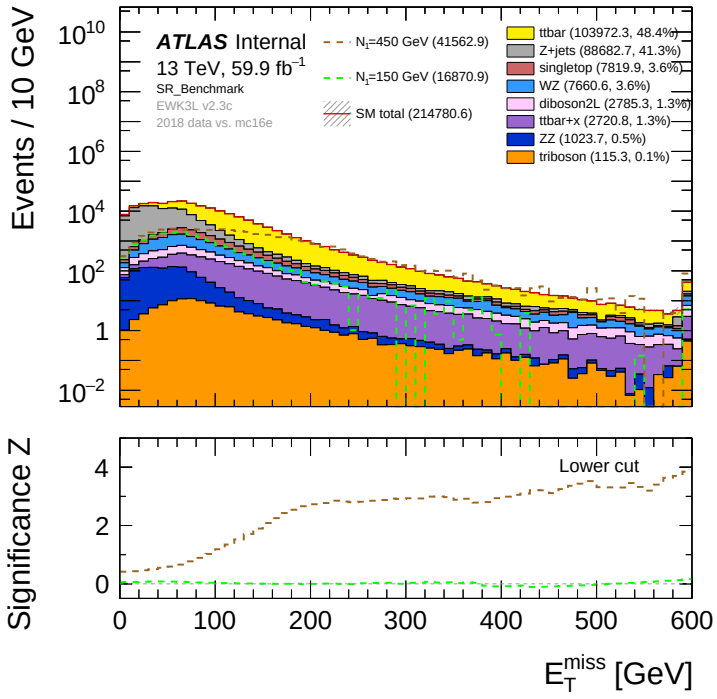


Figure 9.6: The MET with the benchmark cuts for a standard analysis with MC and two signals.

Part IV

Discussion, Conclusion and Future Prospects

Chapter 10

Discussion

This chapter is dedicated to the discussion of the findings we get working with this thesis as presented in part III. The discussion of the results are ranged by tasks where we first discuss the classification performances of the models we have used, how we choose the best performing model and the classification of the Ntuples. Second, we compare and discuss the new feature distributions with different cuts applied.

10.1 Performance of the Classification Models

When we used the correlations and mutual information of the features in section 8.1.1, we came to the conclusion to remove the eta variables for each lepton from the dataframes. There were other features with high correlations, greater than —0.7—, or low mutual information, lower than 0.1 for 150 GeV and lower than 0.45 for 450 GeV, as well. Removing some of these features together with the etas gave only worse results. This is why we only removed the eta features from the dataframes.

During resampling (sect. 8.1.2) of the imbalanced datasets we used different techniques of resampling depending on which signal was to be trained on. From the target counts of the signals in Table 8.1, we see that the 450 GeV signal has from 3938 to 139686 events while the 150 GeV signal only has from 4013 to 26801 events. For the 150 GeV signal, undersampling the majority class with the RandomUnderSampler would not be a great idea since we would not have that much data left to train on afterwards. We would then get something around 12800 events for each class, instead of around 26000 events for each class using only oversampling. With the 450 GeV signal, we still have enough data that we could use the RandomUnderSampler to undersample the majority class to around 65000 events. Using both resampling techniques gave better results since we got overfitting on the trained data when only using oversampling to around 130000 events per class.

With the 150 GeV signal we trained ten different classification models with various results. In Table 8.4 we see the evaluation metrics used for the validation set. Most of the classifiers have an accuracy score below 0.8, which is not good enough to be used more. The classifiers that stand out are the AdaBoost, XGBoost and LGBM models with at least an accuracy score of 0.85. The LGBM model has the highest accuracy score both

overall with score of 0.88, and individually for each class as seen in the LGBM confusion matrix in Figure 8.1. We get the same kind of results with the 450 GeV signal, but all the accuracy scores are much higher. Most of the classifiers get a score higher than 0.9, but the LGBM model is the best also here with accuracy score of 0.95. The reason why the accuracy scores are higher for the 450 GeV model might be because we now have a lot more events ($\approx$ 65000 events for each class) than for the 150 GeV model ($\approx$ 26000 events for each class). The models have a lot more data to train on and detect important differences in the features for deciding a class. The accuracy scores and confusion matrices are the reasons why we pick the LGBM as the best performing model to be used more. The LGBM was also the fastest model to train.

The test set results are very similar to the validation results for the LGBM model with accuracy scores of 0.88 and 0.96 for the 150 GeV and 450 GeV trained models, respectively. This is good since it means that the model can give similar accuracy scores for different unseen data. Once again, the 450 GeV trained model gives better results most likely since it has been trained on more data. The most important features for training the LGBM model on both signals are the invariant masses between the pairs of particles. All the invariant masses show high feature importance for predictions as seen in Figure 8.5 and 8.9. For the 150 Gev model the invariant masses are chosen between 100000 and around 400000 times, while the rest of the features are chosen under 50000 times each. For the 450 GeV model the invariant masses are chosen between 250000 and 900000 times each, while the rest of the features are chosen less than 100000 each.

10.2 Comparing Ntuple Distributions

From the analysis of the different cuts applied to the invariant mass of the three lepton system and the MET, we see that we have a lot more events in the SF plots by comparing the total number of MC events for m_{3l} in Figures 9.1 and 9.2, the MET Figures 9.4 and 9.2 and the number of events for each plot with cuts shown in Tables 9.7 and 9.8 for the 150 and 450 GeV models, respectively. The DF/SF ratios for the two signal models and each vertex permutation are seen in Table 9.9 also showing that there are a lot more SF events. This ratio difference does not apply for the two simulated signals, where the number of SF and DF events are more equal for each vertex and between the models. The 450 GeV signal has on average between 2000-3000 more events in each plot than the 150 GeV signal, except for the plots with the 213 vertex where there are no signal events. From the plot of the expected flavor ratios and LFC in Figure 3.2 for the two models, the 150 GeV shows clearly a larger DF component. This is also what we see in Table 9.9, and is most prominent for the 123 and 132 vertex permutations. We can conclude that SF is favored over DF with two electrons or two muons in the first and second vertices.

The standard analysis plot for the invariant mass has significance similar to the SF and 132 vertex permutation where the 450 GeV significance increases to over 4 σ while the 150 GeV stays below 1 σ. The sensitivity looks to be bigger in the DF channel with almost the same amount of signal events, but the backgrounds have much less events. This

is as expected since the DF cut removes Z-decay which happens in the WZ and Z+jets backgrounds. From the flavor ratios we get different degrees of LFV for the different vertex permutations. This is interesting to look at for the future if we were to discover excess to better understand what type of neutrino mass model we are dealing with. This would also be interesting for the SS vs. OS like the CMS saw.

The 450 GeV signal looks easier to split from the backgrounds when looking at higher invariant masses than 400-500 GeV for the three lepton system with the 123 and 132 vertex permutations. The 450 GeV signal is more dominant in this region in terms of events. The standard analysis looks to have the same results for dividing background and signal where the 450 GeV signal has more events for $m_{3l} > 500$ GeV. In general it seems like our trained model performs better then the simple benchmark analysis. However, one should bare in mind that the benchmark model from Table 9.6 has not been optimized on the signal models under study.

For MET it is not as easy to differentiate between the backgrounds and signals since both signals have more or less the same amount of events like the backgrounds have. The 450 GeV can be differentiated to some extent between 100 and 400 GeV for the 123 vertex plots. The signals do also have very similar amount of events for energies below 400 GeV. So to differentiate the signals from the models we might need to look in the higher energy regions. For the significance with the MET the 150 GeV signal were only as high as 2 σ with this signal region for 0-100 GeV. For the other signal regions the 150 GeV signal significance stays mostly between 0-1 σ. The significances are much less in the MET distributions. As expected the MET does not discriminate well the signals and backgrounds.

Chapter 11

Conclusion and Future Work

11.1 Conclusion

In this thesis we have tested a new approach in particle physics by utilizing ML and multiclass classification of lepton vertices to be used to differentiate simulated signal and backgrounds and looking for excess. We first trained multiclass classification models on two neutrino signal scenarios with different neutrino masses of 150 and 450 GeV. These models have been used to find the classification models that best predicts the lepton vertex permutations for a trilepton plus a neutrino final state system. The proton-proton decay model is based on the decay of a SM W-boson decaying to a heavy pseudo-Dirac neutrino through the Inverse seesaw mechanism, leading to the trilepton final state. The Light Gradient Boosting Machine (LGBM) was found to be the best performing model with accuracy score of 0.88 for the 150 GeV trained signal and accuracy score of 0.96 for the 450 GeV trained signal.

We then classified the vertex permutations of simulated background MC and two similar neutrino signals corresponding to the data recorded by the LHC in 2018. The most predicted vertex permutations, 123, 132 and 213, were used as cuts for a feature analysis to check for LFV between the leptons coming from vertex 1 and 2 for electrons and muons. The invariant mass of the three lepton system and the MET were used for comparing our selected signal region cuts with a more standard analysis applying different cuts from Pascoli et al. [1].

The standard analysis was found to give similar results for the signals significance for the SF with OS cuts and vertex permutation 132 for both signals. The classification models for the backgrounds seemed to favor the SF and OS case more with much more events for the 123 and 132 vertex permutations for these signal region cuts. The two simulated signals had number of events quite similar for the SF and DF. With the 450 GeV signal we found it easier to differentiate against background when we looked at higher masses than 400-500 GeV of the invariant three lepton system mass for both the flavor, charge and vertex permutation signal regions and the standard analysis signal region. This differentiation between the backgrounds and the 450 GeV signal could also somewhat be seen with the MET for the DF, OS and 123 vertex signal regions between 100 and 400

GeV. The significance were found to be higher with the 450 GeV signal compared with the 150 GeV signal, and higher for the m_{3l} compared with the MET.

In general it seems like our trained models perform better than the simple benchmark analysis. However, one should bare in mind that the benchmark model in Table 9.6 has not been optimized on the signal models under study.

Modern ML techniques are rapidly modified and developed in the field of particle physics and high energy physics (HEP) for analysis. The article by Feickert and Nachman [66] presents a list of interesting literature into the development and application of ML techniques for HEP analyses. The article is constantly updated and made for the community to follow the development of different ML techniques. It only contains a fraction of the papers about the use of ML with particle physics that are out there.

11.2 Future Work

We have tested a few classification models yielding various results and performances on the datasets. There are other ML models to test and other useful libraries for ML that could be used to train classification models, e.g. Tensorflow or Keras. Another ML technique to be tested could be deep learning or other networks that may give different results than the ones we have used in this thesis. Only the MC and two neutrino signals were classified with the LGBM model. The data could also be classified and analyzed together with the MC and signals.

The work on ML has been the main focus of this thesis. A natural continuation of this work is to extend it with more detailed particle physics studies, such as more in depth of the particle physics and the deviation of the number of SS vs. OS the CMS looked at in the article [4].

Another possible continuation of this work would be to extend the analysis to include other data periods, like data from the whole Run 2 by the LHC. This is easily implemented in the scripts we have used to plot the feature distributions. It would also be interesting to test the models on the data to see how it fits with the SM background expectations.

Other neutrino signals with other masses could also be interesting to test. The invariant mass production errors with $p > E$ is something to further studied as well.

The most interesting results of this work would be if an excess is observed to try and see if one can understand more about the events and which underlying theoretical model would fit the observed excess best.

Part V

Appendices

Appendix A

Bias-Variance Decomposition

Here we do the full derivation of the expected generalization error in equation 6.9:

$$\begin{aligned}
\mathbb{E}_{\mathcal{D},\epsilon}[\mathcal{C}(\mathbf{y}, f(\mathbf{X}; \hat{\theta}_{\mathcal{D}}))] &= \mathbb{E}_{\mathcal{D},\epsilon}\left[\sum_i (y_i - f(\mathbf{x}_i; \hat{\theta}_{\mathcal{D}}))^2\right] \\
&= \mathbb{E}_{\mathcal{D},\epsilon}\left[\sum_i (y_i - f(\mathbf{x}_i; \hat{\theta}_{\mathcal{D}}) - f(\mathbf{x}_i) + f(\mathbf{x}_i))^2\right] \\
&= \sum_i \mathbb{E}_{\epsilon}[(y_i - f(\mathbf{x}_i))^2] + \mathbb{E}_{\mathcal{D},\epsilon}[(f(\mathbf{x}_i) - f(\mathbf{x}_i; \hat{\theta}_{\mathcal{D}}))^2] \\
&\quad + 2\mathbb{E}_{\epsilon}[y_i - f(\mathbf{x}_i)]\mathbb{E}_{\mathcal{D}}[f(\mathbf{x}_i; \hat{\theta}_{\mathcal{D}})] \\
&= \sum_i \sigma_\epsilon^2 + \mathbb{E}_{\mathcal{D}}[(f(\mathbf{x}_i) - f(\mathbf{x}_i; \hat{\theta}_{\mathcal{D}}))^2]
\end{aligned}$$

Here we have used the fact that the noise has zero mean and variance σ_ϵ^2. We also further decompose the second expectation term:

$$\begin{aligned}
\mathbb{E}_{\mathcal{D}}[(f(\mathbf{x}_i) - f(\mathbf{x}_i; \hat{\theta}_{\mathcal{D}}))^2] &= \mathbb{E}_{\mathcal{D}}\left[(f(\mathbf{x}_i) - f(\mathbf{x}_i; \hat{\theta}_{\mathcal{D}}) - \mathbb{E}_{\mathcal{D}}[f(\mathbf{x}_i; \hat{\theta}_{\mathcal{D}})] + \mathbb{E}_{\mathcal{D}}[f(\mathbf{x}_i; \hat{\theta}_{\mathcal{D}})])^2\right] \\
&= (f(\mathbf{x}_i) - \mathbb{E}_{\mathcal{D}}[f(\mathbf{x}_i; \hat{\theta}_{\mathcal{D}})])^2 + \mathbb{E}_{\mathcal{D}}[\{f(\mathbf{x}_i; \hat{\theta}_{\mathcal{D}}) - \mathbb{E}[f(\mathbf{x}_i; \hat{\theta}_{\mathcal{D}})]\}^2]
\end{aligned}$$

Putting these two equation together leads to the expected generalization error:

$$\begin{aligned}
\mathbb{E}_{\mathcal{D},\epsilon}[\mathcal{C}(\mathbf{y}, f(\mathbf{X}; \hat{\theta}_{\mathcal{D}}))] &= \sum_i (f(\mathbf{x}_i) - \mathbb{E}_{\mathcal{D}}[f(\mathbf{x}_i; \hat{\theta}_{\mathcal{D}})])^2 \\
&\quad + \sum_i \mathbb{E}_{\mathcal{D}}[\{f(\mathbf{x}_i; \hat{\theta}_{\mathcal{D}}) - \mathbb{E}[f(\mathbf{x}_i; \hat{\theta}_{\mathcal{D}})]\}^2] \\
&\quad + \sum_i \sigma_\epsilon^2
\end{aligned}$$

Appendix B

450 GeV Signal Data Summary

The $N1 = 450$ GeV data summary:

```
<class 'pandas.core.frame.DataFrame'>
Int64Index: 259436 entries, 0 to 261331
Data columns (total 55 columns):
 #   Column       Non-Null Count   Dtype
---  ------       --------------   -----
 0   lep1_pt      259436 non-null  float32
 1   lep1_phi     259436 non-null  float32
 2   lep1_eta     259436 non-null  float32
 3   lep1_theta   259436 non-null  float32
 4   lep1_px      259436 non-null  float32
 5   lep1_py      259436 non-null  float32
 6   lep1_pz      259436 non-null  float32
 7   lep1_E       259436 non-null  float32
 8   lep1_tlv     259436 non-null  object
 9   lep2_pt      259436 non-null  float32
 10  lep2_phi     259436 non-null  float32
 11  lep2_eta     259436 non-null  float32
 12  lep2_theta   259436 non-null  float32
 13  lep2_px      259436 non-null  float32
 14  lep2_py      259436 non-null  float32
 15  lep2_pz      259436 non-null  float32
 16  lep2_E       259436 non-null  float32
 17  lep2_tlv     259436 non-null  object
 18  lep3_pt      259436 non-null  float32
 19  lep3_phi     259436 non-null  float32
 20  lep3_eta     259436 non-null  float32
 21  lep3_theta   259436 non-null  float32
 22  lep3_px      259436 non-null  float32
 23  lep3_py      259436 non-null  float32
 24  lep3_pz      259436 non-null  float32
 25  lep3_E       259436 non-null  float32
 26  lep3_tlv     259436 non-null  object
 27  lep4_pt      259436 non-null  float32
 28  lep4_phi     259436 non-null  float32
 29  lep4_eta     259436 non-null  float32
 30  lep4_theta   259436 non-null  float32
 31  lep4_px      259436 non-null  float32
 32  lep4_py      259436 non-null  float32
 33  lep4_pz      259436 non-null  float32
 34  lep4_E       259436 non-null  float32
 35  lep4_tlv     259436 non-null  object
 36  mll_12       259436 non-null  float64
```

```
 37  dphi_12      259436 non-null  float64
 38  dR_12        259436 non-null  float64
 39  mll_13       259436 non-null  float64
 40  dphi_13      259436 non-null  float64
 41  dR_13        259436 non-null  float64
 42  mll_23       259436 non-null  float64
 43  dphi_23      259436 non-null  float64
 44  dR_23        259436 non-null  float64
 45  mll_14       259436 non-null  float64
 46  dphi_14      259436 non-null  float64
 47  dR_14        259436 non-null  float64
 48  mll_24       259436 non-null  float64
 49  dphi_24      259436 non-null  float64
 50  dR_24        259436 non-null  float64
 51  mll_34       259436 non-null  float64
 52  dphi_34      259436 non-null  float64
 53  dR_34        259436 non-null  float64
 54  target       259436 non-null  object
dtypes: float32(32), float64(18), object(5)
memory usage: 79.2+ MB

              lep1_pt  lep1_phi  lep1_eta  ...    dphi_34     dR_34 target
entry                                      ...
0      247239.234375  1.705486 -0.009060  ... -0.059092  0.872886           (2, ↩
    3, 1)
1      242870.343750 -2.694518 -1.187190  ...  0.105023  0.717894           (2, ↩
    1, 3)
2      275632.000000 -0.628263 -0.185416  ...  1.024500  1.044483           (2, ↩
    1, 3)
3      203956.265625  2.971124 -2.185891  ... -0.364194  0.678069           (2, ↩
    1, 3)
4      106095.476562 -0.685928 -0.315220  ... -0.983589  1.397503           (2, ↩
    1, 3)

[5 rows x 55 columns]
```

Listing B.1: Inspecting the 450 GeV data set.

Appendix C

Correlations

Signal $N1 = 150$ GeV:

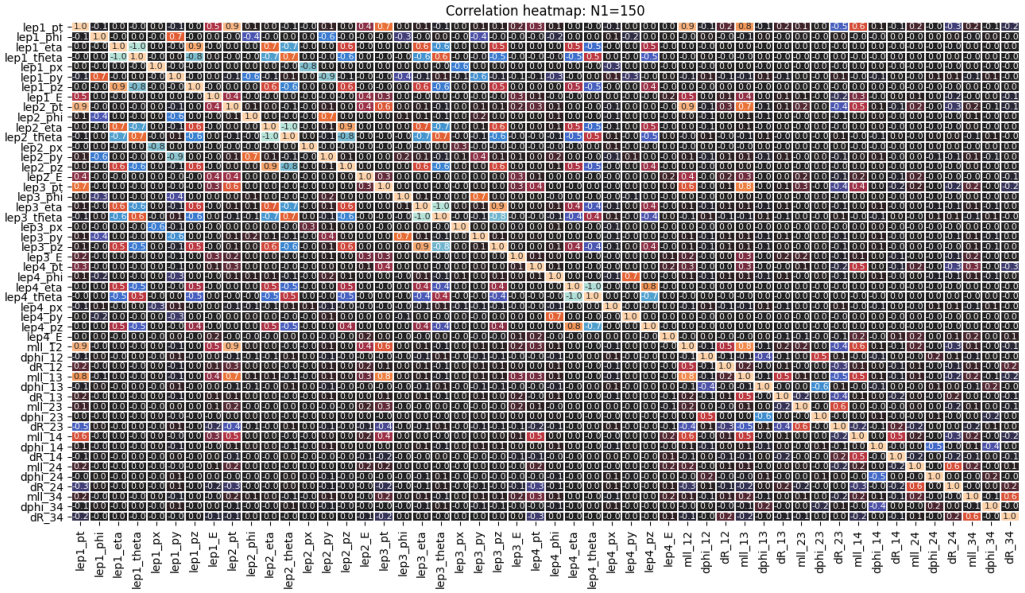


Figure C.1: Correlation matrix of the features in the 150 GeV signal dataset. Values close to 1 or -1 show high correlations between the features.

```
Information gain of the features:
[('lep2_phi', 0.05178672925582184),
('lep1_phi', 0.053745432520311276),
('dphi_24', 0.05485490369918944),
('lep3_phi', 0.05502894645503176),
('lep3_eta', 0.05594756636754683),
('lep3_theta', 0.05598787413359707),
('lep4_phi', 0.05641456053531346),
('lep4_pz', 0.05947799347697558),
('lep2_eta', 0.06035343605141508),
```

```
('lep2_theta', 0.060373423669804804),
('dphi_14', 0.06083513368874938),
('dphi_34', 0.06115565193619199),
('lep1_eta', 0.06166261021577125),
('lep1_theta', 0.06168403392066146),
('lep3_pz', 0.06422467081537953),
('lep4_theta', 0.06485782851335387),
('lep4_eta', 0.0648977837020408),
('lep4_py', 0.06604010400547766),
('lep4_px', 0.06752198284777089),
('lep4_E', 0.06793095808433014),
('dR_34', 0.06849304962655012),
('lep1_pz', 0.06885166365937634),
('lep2_pz', 0.06906061765196592),
('dR_13', 0.0716082982162769),
('dR_14', 0.07197872976190567),
('dphi_13', 0.07423421063493052),
('dR_24', 0.07595752322276583),
('lep3_E', 0.08410378910700622),
('lep4_pt', 0.09082301709001017),
('dphi_23', 0.09258874841582809),
('dphi_12', 0.09770848996184833),
('lep1_E', 0.10512713284596842),
('lep3_px', 0.10532072166635142),
('lep2_E', 0.10629293534281459),
('lep3_py', 0.10734840100318799),
('dR_12', 0.10964280256859293),
('dR_23', 0.11751795736407011),
('mll_24', 0.12528976189890173),
('lep1_px', 0.14667014422688363),
('lep2_px', 0.14672888551516206),
('lep2_py', 0.15222854675813702),
('lep1_py', 0.1559645954915414),
('lep3_pt', 0.17443874057539732),
('mll_14', 0.19302957284767652),
('mll_34', 0.20736617389081324),
('mll_23', 0.2081857239040863),
('lep2_pt', 0.27355582731666006),
('lep1_pt', 0.276609594149106),
('mll_13', 0.28691694495451703),
('mll_12', 0.3932658736116754)]
```

Signal $N1 = 450$ GeV:

```
Information gain of the features:
[('lep4_theta', 0.4191552527318376),
('lep4_phi', 0.42027977972758146),
('lep3_eta', 0.42045370659465386),
('lep1_phi', 0.4210809806293345),
('lep2_pz', 0.42162772292849215),
('lep2_eta', 0.42201102378344824),
('lep2_phi', 0.4230067197296721),
('lep1_pz', 0.424541501741019),
('lep4_eta', 0.4257280226520528),
('lep3_pz', 0.42672816312478656),
('lep3_phi', 0.427599991411618),
('lep3_theta', 0.42907751026540675),
('lep4_pz', 0.4301374641613722),
('lep2_theta', 0.43121039167465 48),
('lep1_E', 0.43129937565133747),
('lep4_px', 0.43132544346123725),
('dR_12', 0.4322203904822728),
('lep4_E', 0.43372185011808995),
```

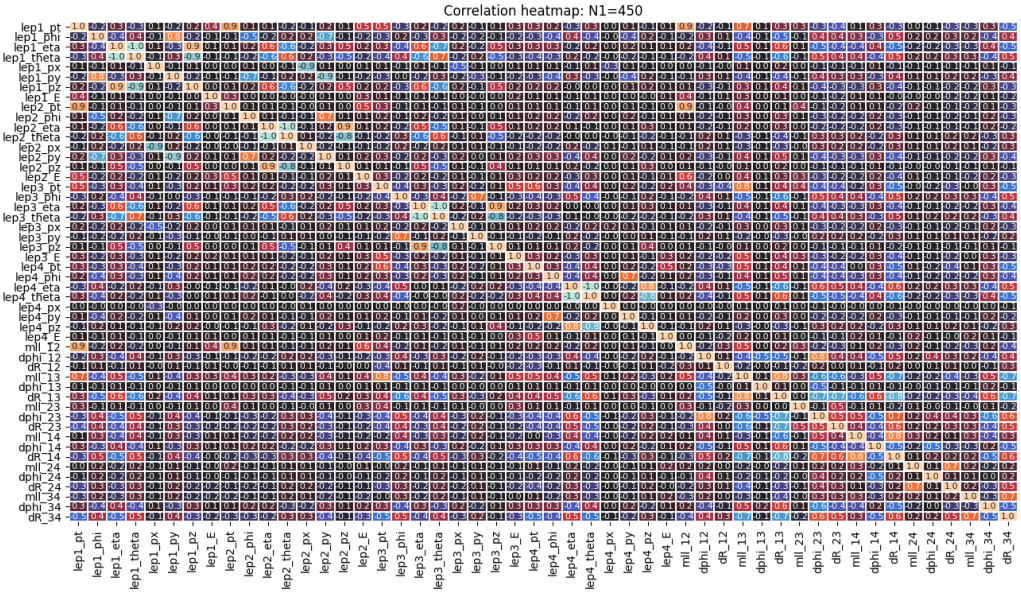


Figure C.2: Correlation matrix of the features in the 450 GeV signal dataset. Values close to 1 or -1 show high correlations between the features.

```
        ('dphi_12', 0.43389037653643614),
        ('lep4_py', 0.43395756196715096),
        ('dR_13', 0.4347988629887223),
        ('dphi_13', 0.43616113224927466),
        ('lep1_eta', 0.43616248827916504),
        ('lep1_theta', 0.4386034683797735),
        ('lep3_E', 0.44167047831807293),
        ('lep2_E', 0.44415728982148406),
        ('lep3_px', 0.4471424012534033),
        ('dphi_24', 0.4487297203828642),
        ('lep3_py', 0.44887857656083185),
        ('lep1_px', 0.45032798796079376),
        ('lep1_py', 0.4515757089823731),
        ('lep4_pt', 0.45213792012843435),
        ('lep2_px', 0.4580357692816419),
        ('lep2_py', 0.4610110981533644),
        ('dphi_14', 0.4611574842208519),
        ('dphi_23', 0.4624262746988771),
        ('dR_24', 0.4692019081992844),
        ('dphi_34', 0.4784639532967272),
        ('dR_14', 0.4809026477868572),
        ('lep3_pt', 0.48225046197217947),
        ('lep1_pt', 0.4998335491304495 3),
        ('dR_34', 0.5095008908298231),
        ('lep2_pt', 0.5120865029463617),
        ('dR_23', 0.5122830173546382),
        ('mll_24', 0.5187574240316617),
        ('mll_13', 0.5404532121755724),
        ('mll_14', 0.5424036353399868),
        ('mll_12', 0.5602705982085625),
        ('mll_34', 0.6049717229179872),
        ('mll_23', 0.6204488569318036)])
```

Acronyms

ALICE A Large Ion Collider Experiment

ATLAS A Toroidal LHC ApparatuS

AUC area under the curve

BEH Brout-Englert-Higgs

CART Classification and Regression Trees

CCDY charged current Drell-Yan

CERN European Organization for Nuclear Research (EN)

CKS Cohen Kappa Score

CM center-of-mass

CMS Compact Muon Solenoid

CSV comma separated values

DF different flavor

DIS Deep Inelastic Scattering

DT Decision Tree

ECal electromagnetic calorimeter

EF Event filter

EWT electroweak theory

FFNN Feed-Forward Neural Network

GBDT Gradient Boosting Decision Tree

GWS Glashow-Weinberg-Salam

HCal hadronic calorimeter

HEP high energy physics

HLT High Level Trigger

ID inner detector

ISS Inverse Seesaw

KATRIN Karlsruhe Tritium Neutrino

LEP Large Electron-Positron Collider

LFC lepton flavor conservation

LFV lepton flavor violation

LGBM Light Gradient Boosting Machine

LH left-handed

LHC Large Hadron Collider

LHCb LHC-beauty

LR Logistic Regression

LRSM Left-Right Symmetric Model

MC Monte Carlo

MET missing transverse momentum

ML machine learning

MLE Maximum Likelihood Estimation

MLP Multi-Layer Perceptron

MLR Multinomial Logistic Regression

MS muon spectrometer

OS opposite sign

OvO one-vs-one

OvR one-vs-rest

PCA Principal Component Analysis

PDF parton distribution function

PMNS Pontecorvo-Maki-Nakagawa-Sakata

QCD quantum chromodynamics

QED quantum electrodynamics

QFT quantum field theory

ReLU Rectified Linear Unit

RH right-handed

RnF Random Forest

ROC Receiver Operating Characteristic

ROI region of interest

SCT Semiconductor Tracker

SF same flavor

SM Standard Model

SNO Sudbury Neutrino Observatories

SPS Super Proton Synchrotron

SS same sign

TDAQ The ATLAS Trigger and Data Acquisition system

TRT Transition Radiation Tracker

WLCG World LHC Computing Grid

XGBoost Extreme Gradient Boosting

Bibliography

[1] Silvia Pascoli, Richard Ruiz, and Cedric Weiland. Heavy neutrinos with dynamic jet vetoes: multilepton searches at $\sqrt{s}$= 14, 27, and 100 TeV. *Journal of High Energy Physics*, 2019(6):49, 2019.

[2] Georges Aad, Tatevik Abajyan, B Abbott, J Abdallah, S Abdel Khalek, Ahmed Ali Abdelalim, R Aben, B Abi, M Abolins, OS AbouZeid, et al. Observation of a new particle in the search for the standard model higgs boson with the atlas detector at the lhc. *Physics Letters B*, 716(1):1–29, 2012.

[3] Serguei Chatrchyan, Vardan Khachatryan, Albert M Sirunyan, Armen Tumasyan, Wolfgang Adam, Ernest Aguilo, Thomas Bergauer, M Dragicevic, J Erö, C Fabjan, et al. Observation of a new boson at a mass of 125 gev with the cms experiment at the lhc. *Physics Letters B*, 716(1):30–61, 2012.

[4] CMS Collaboration, Vardan Khachatryan, Albert M Sirunyan, Armen Tumasyan, Wolfgang Adam, Thomas Bergauer, Marko Dragicevic, Janos Erö, Christian Fabjan, Markus Friedl, Rudolf Fruehwirth, et al. Search for heavy neutrinos and W bosons with right-handed couplings in proton-proton collisions at $\sqrt{s}$= 8TeV. *The European Physical Journal C*, 74(11):3149, 2014. doi: https://doi.org/10.1140/epjc/s10052-014-3149-z.

[5] Mark Thomson. *Modern particle physics*. Cambridge University Press, 2013.

[6] Glenn Elert. The physics hypertextbook, 1998-2020. URL https://physics.info/standard/.

[7] Andrew Purcell. Go on a particle quest at the first cern webfest. le premier webfest du cern se lance à la conquête des particules. page 10, Aug 2012. URL https://cds.cern.ch/record/1473657.

[8] Benjamin P Abbott, Richard Abbott, TD Abbott, MR Abernathy, Fausto Acernese, Kendall Ackley, Carl Adams, Thomas Adams, Paolo Addesso, RX Adhikari, et al. Observation of gravitational waves from a binary black hole merger. *Physical review letters*, 116(6):061102, 2016.

[9] Y Fukuda, T Hayakawa, E Ichihara, K Inoue, K Ishihara, Hirokazu Ishino, Y Itow, T Kajita, J Kameda, S Kasuga, et al. Evidence for oscillation of atmospheric neutrinos. *Physical Review Letters*, 81(8):1562, 1998.

[10] Christopher W Walter and Super-Kamiokande collaboration. The super-kamiokande experiment. In *Neutrino Oscillations: Present Status and Future Plans*, pages 19–43. World Scientific, 2008.

[11] Alain Bellerive, JR Klein, AB McDonald, AJ Noble, AWP Poon, SNO Collaboration, et al. The sudbury neutrino observatory. *Nuclear Physics B*, 908:30–51, 2016.

[12] Elizabeth Gibney and Davide Castelvecchi. Neutrino flip wins physics prize, 2015.

[13] Max Aker, K Altenmüller, M Arenz, M Babutzka, J Barrett, S Bauer, M Beck, A Beglarian, J Behrens, T Bergmann, et al. Improved upper limit on the neutrino mass from a direct kinematic method by katrin. *Physical review letters*, 123(22): 221802, 2019.

[14] Alan Kostelecky. The status of cpt. *arXiv preprint hep-ph/9810365*, 1998.

[15] Franz Mandl and Graham Shaw. *Quantum field theory*. John Wiley & Sons, 2010.

[16] Michel Le Bellac. *Quantum and statistical field theory*. Clarendon Press, 1991.

[17] Richard P Feynman. The principle of least action in quantum mechanics. In *Feynman's Thesis—A New Approach To Quantum Theory*, pages 1–69. World Scientific, 2005. doi: https://doi.org/10.1142/9789812567635_0001.

[18] Gerhard Ecker. Quantum chromodynamics. *arXiv preprint hep-ph/0604165*, 2006.

[19] C. N. Yang and R. L. Mills. Conservation of isotopic spin and isotopic gauge invariance. *Phys. Rev.*, 96:191–195, Oct 1954. doi: 10.1103/PhysRev.96.191. URL https://link.aps.org/doi/10.1103/PhysRev.96.191.

[20] Richard Phillips Feynman. *QED: The strange theory of light and matter*. Princeton University Press, 2006.

[21] Guido Altarelli. The standard electroweak theory and beyond. *arXiv preprint hep-ph/9811456*, pages 27–93, 2000.

[22] Sheldon L Glashow. Partial-symmetries of weak interactions. *Nuclear physics*, 22(4): 579–588, 1961. doi: https://doi.org/10.1016/0029-5582(61)90469-2.

[23] Steven Weinberg. A model of leptons. *Physical review letters*, 19(21):1264, 1967. doi: https://doi.org/10.1103/PhysRevLett.19.1264.

[24] A Salam. N. svartholm, ed. elementary particle physics: Relativistic groups and analyticity. In *Eighth Nobel Symposium. Stockholm: Almquvist and Wiksell*, page 367, 1968.

[25] *Press release: The Nobel Prize in Physics 1979*. Nobel Media AB 2020, 1979. URL https://www.nobelprize.org/prizes/physics/1979/press-release/.

[26] Abdus Salam and Steven Weinberg. Nobel Prize for Physics, 1979. *CERN Courier*, page 395, 1979.

[27] Darwin Chang and Otto CW Kong. Pseudo-dirac neutrinos. *Physics Letters B*, 477 (4):416–423, 2000.

[28] KRS Balaji, Anna Kalliomäki, and Jukka Maalampi. Revisiting pseudo-dirac neutrinos. *Physics Letters B*, 524(1-2):153–160, 2002.

[29] Rabindra N Mohapatra. Seesaw mechanism and its implications. In *SEESAW 25*, pages 29–44. World Scientific, 2005.

[30] Arindam Das, Natsumi Nagata, and Nobuchika Okada. Testing the 2-TeV resonance with trileptons. *Journal of High Energy Physics*, 2016(3):49, 2016.

[31] Eirik Gramstad. Searches for Supersymmetry in di-Lepton Final States with the ATLAS Detector at $\sqrt{s}$= 7 TeV. 2013.

[32] Richard D Field. The underlying event in hard scattering processes. *arXiv preprint hep-ph/0201192*, 2002.

[33] About CERN. CERN, (10.12.20). URL https://home.cern/about.

[34] Stephanie Sammartino McPherson. *Tim Berners-Lee: Inventor of the World Wide Web*. Twenty-First Century Books, 2009.

[35] Esma Mobs. The CERN accelerator complex - 2019. Complexe des accélérateurs du CERN - 2019. Jul 2019. URL https://cds.cern.ch/record/2684277. General Photo.

[36] The Large Hadron Collider. CERN, (11.12.20). URL https://home.cern/science/accelerators/large-hadron-collider.

[37] Accelerators. CERN, (11.12.20). URL https://home.cern/science/accelerators.

[38] Experiments: LHC experiments. CERN, (11.12.20). URL https://home.cern/science/experiments.

[39] A Airapetian, V Dodonov, L Micu, D Axen, V Vinogradov, D Akerman, B Szeless, P Chochula, C Geich-Gimbel, P Schacht, et al. *ATLAS detector and physics performance: Technical Design Report, 1*, volume 1. ATLAS-TDR-014, 1999. URL http://atlas.web.cern.ch/Atlas/GROUPS/PHYSICS/TDR/access.html.

[40] CERN. Computer generated image of the whole ATLAS detector. CERN Document Server, (01.01.21). URL https://cds.cern.ch/record/1095924?ln=en.

[41] How ATLAS detects particles: diagram of particle paths in the detector . CERN Document Server, (01.01.21). URL https://cds.cern.ch/record/1505342?ln=en.

[42] CERN. Overall detector concept. *ATLAS Technical Proposal*, 1994.

[43] DA Scannicchio. Atlas trigger and data acquisition: Capabilities and commissioning. *Nuclear Instruments and Methods in Physics Research Section A: Accelerators, Spectrometers, Detectors and Associated Equipment*, 617(1-3):306–309, 2010.

[44] Ian Goodfellow, Yoshua Bengio, and Aaron Courville. *Deep Learning*. MIT Press, 2016. http://www.deeplearningbook.org.

[45] Trevor Hastie, Robert Tibshirani, and Jerome Friedman. *The elements of statistical learning: data mining, inference, and prediction*. Springer Science & Business Media, 2009.

[46] Pankaj Mehta, Marin Bukov, Ching-Hao Wang, Alexandre GR Day, Clint Richardson, Charles K Fisher, and David J Schwab. A high-bias, low-variance introduction to machine learning for physicists. *Physics reports*, 810:1–124, 2019.

[47] Dennis W Ruck, Steven K Rogers, and Matthew Kabrisky. Feature selection using a multilayer perceptron. *Journal of Neural Network Computing*, 2(2):40–48, 1990.

[48] Sandra Vieira, Walter HL Pinaya, and Andrea Mechelli. Using deep learning to investigate the neuroimaging correlates of psychiatric and neurological disorders: Methods and applications. *Neuroscience & Biobehavioral Reviews*, 74:58–75, 2017.

[49] Diederik P Kingma and Jimmy Ba. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*, 2014.

[50] S Rasoul Safavian and David Landgrebe. A survey of decision tree classifier methodology. *IEEE transactions on systems, man, and cybernetics*, 21(3):660–674, 1991.

[51] Svante Wold, Kim Esbensen, and Paul Geladi. Principal component analysis. *Chemometrics and intelligent laboratory systems*, 2(1-3):37–52, 1987.

[52] Leo Breiman. Random forests. *Machine learning*, 45(1):5–32, 2001.

[53] J Ross Quinlan et al. Bagging, boosting, and c4. 5. In *Aaai/iaai, Vol. 1*, pages 725–730, 1996.

[54] Gunnar Rätsch, Takashi Onoda, and K-R Müller. Soft margins for adaboost. *Machine learning*, 42(3):287–320, 2001.

[55] Jerome H Friedman. Stochastic gradient boosting. *Computational statistics & data analysis*, 38(4):367–378, 2002.

[56] Guolin Ke, Qi Meng, Thomas Finley, Taifeng Wang, Wei Chen, Weidong Ma, Qiwei Ye, and Tie-Yan Liu. Lightgbm: A highly efficient gradient boosting decision tree. *Advances in neural information processing systems*, 30:3146–3154, 2017.

[57] Tianqi Chen and Carlos Guestrin. Xgboost: A scalable tree boosting system. In *Proceedings of the 22nd acm sigkdd international conference on knowledge discovery and data mining*, pages 785–794, 2016.

[58] Abha Eli Phoboo. Machine learning wins the higgs challenge. Technical report, 2014.

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

[60] Mary L McHugh. Interrater reliability: the kappa statistic. *Biochemia medica*, 22(3): 276–282, 2012.

[61] James Catmore. The atlas data processing chain: from collisions to papers. *University of Oslo, presentation slides*, 2020.

[62] J. Alwall, R. Frederix, S. Frixione, V. Hirschi, F. Maltoni, O. Mattelaer, H.-S. Shao, T. Stelzer, P. Torrielli, and M. Zaro. The automated computation of tree-level and next-to-leading order differential cross sections, and their matching to parton shower simulations. *Journal of High Energy Physics*, 2014(7), Jul 2014. ISSN 1029-8479. doi: 10.1007/jhep07(2014)079. URL http://dx.doi.org/10.1007/JHEP07(2014)079.

[63] Torbjörn Sjöstrand, Stephen Mrenna, and Peter Skands. Pythia 6.4 physics and manual. *Journal of High Energy Physics*, 2006(05):026, 2006.

[64] Michael L. Waskom. seaborn: statistical data visualization. *Journal of Open Source Software*, 6(60):3021, 2021. doi: 10.21105/joss.03021. URL https://doi.org/10.21105/joss.03021.

[65] Guillaume Lemaître, Fernando Nogueira, and Christos K. Aridas. Imbalanced-learn: A python toolbox to tackle the curse of imbalanced datasets in machine learning. *Journal of Machine Learning Research*, 18(17):1–5, 2017. URL http://jmlr.org/papers/v18/16-365.html.

[66] Matthew Feickert and Benjamin Nachman. A living review of machine learning for particle physics. *arXiv preprint arXiv:2102.02770*, 2021.