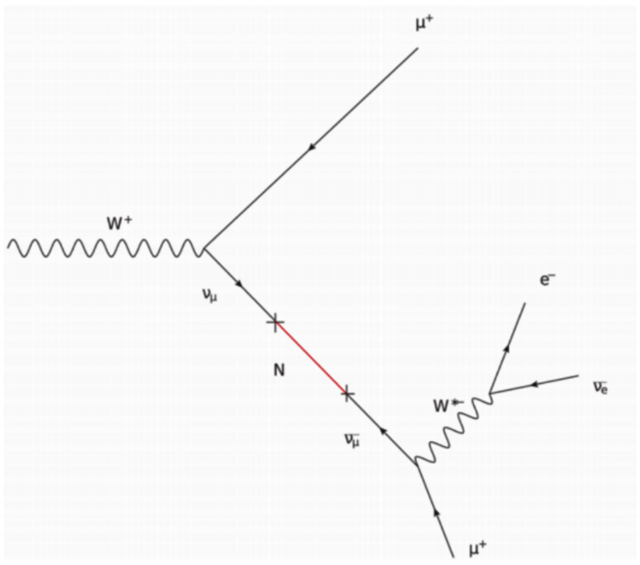
University Of Copenhagen

Assessing Machine-Learning methods for optimizing signal detection in the search for Theoretical Particles.

An Experimental Project



Bachelor Project

Author: Svend V. Korsgaard

Date: February 4th

Supervisor: Stefania Xella

Institute:	Niels Bohr Institute
Department	Department for Experimental Particle Physics
Author(s):	Svend Ventura Korsgaard
Description:	This project is about investigating the use of Supervised Machine-Learning Classifiers for the purpose of event-selection in data from particle colliders. The goal is to maximally decrease the proportion of background events that will be left in the data after selection. We will asses how well this can be done by adjusting the decision threshold in supervised machine learning models, that were trained using simulations of particle collisions.
Advisor:	Stefania Xella
Date:	04 02 2022

Table of content

Part I
Introduction

Since the dawn of time, humanity has pondered about our existence, searching for meaning and structure in the world around us. We have for the longest time used myths to explain our reality, but since the inception of the scientific method, our understanding of the world has vastly deepened, so where has this lead us?

The Ancient Greeks believed that the world was made up of ”atoms”, which were small indivisible spheres. It took us around 2000 to come to the understanding that they were right. However, the atoms of today are not indivisible, in fact they are made of even smaller entities, which we call ”particles”. But that is just the beginning of the story.

1 The Standard Model

What we thus so far has discovered, is that the universe is made up of *Quantum Fields*. The things we can observe, such as light, matter and forces, are localized excitation's of the quantum-fields, and these excitation's is what we call *particles*. This theory of matter, called *Quantum Field Theory*, has a specific version that describes all of the interactions between matter and forces that are known so far, and is what is known as *The Standard Model*.

The Standard Model represents an accumulation of all knowledge about how the world works (with the exception of gravity). It describes how the particles can be organized into several different classes, which is represented in figure 1.1, we have 12 particles that constitute matter, know as *fermions*. There are 2 types of fermions, namely the *Quarks* and *Leptons*. We have 4 particles that are force carriers, namely the *Bosons*. And lastly we have the *Higgs-boson*, which was discovered in 2012, which was the last piece missing to confirm the Standard Model. Besides that, all particles have a corrosponding anti-particle, which is identical to the particle, except they have the opposite electrical charge. Our universe consists mainly of particles.

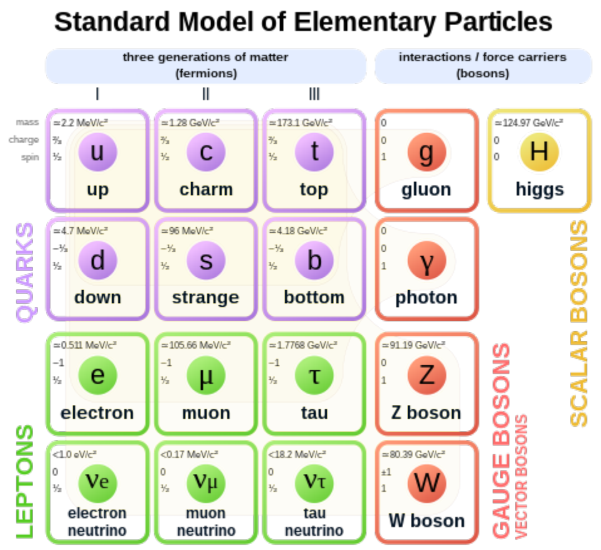


Figure 1.1: The Standard Model: An overview of particles and forces

All experiments we have done so far has only served to confirm the predictions of the Standard Model, but we know that the Standard Model can not be the complete picture of reality. For once, it predicts that in the beginning of time, matter and anti-matter was created in equal amounts, which is a problem, because when matter meets antimatter they annihilate each-other and become *photons*, the carriers of light. If that was the case, we would not exist, because there would be no matter. It also predicts that neutrinos are massless, but we know that neutrinos **DO** have mass, because of a phenomenon called *Neutrino Oscillations*.

The Standard Model also doesn't explain Dark Matter, Dark Energy, and doesn'tinclude gravity. Because of this, we know that the Standard Model is an incomplete picture of reality.

1.1 Extending The Standard Model

There are many proposed solutions to solve the problems presented above, one of them is that the world is made of tiny strings, instead of particles, another proposed explanation is that space and time itself only exists in increments. But another approach is to take the Standard Model, and extend it by adding to it new functionality, new particles and etc...

1.2 Majorana Particle, Sterile Neutrino, and Heavy Neutral Lepton

Particles can normally be either left-handed or right-handed, this is a deep theoretical concept. but the weak force only reacts with left-handed particles. Since the neutrino only ever interacts via the weak force, this means that we have never observed a right-handed neutrino. So if they exists, we call them a *Sterile Neutrino.*

One of the proposed extensions to The Standard Model, is that the neutrino, is its own antiparticle. A Fermion that is its own antiparticle is called a *Majorana Particle* . The Neutrino is the only particle in The Standard Model that has this possibility, because it is the only fermion with zero charge. This is motivated by the fact that we have never observed a right-handed neutrino. Another variation of this proposal is that that neutrinoes can turn into their anti-particle, through first transforming into a new kind of particle called a **Heavy Neutral Lepton** [6] or **HNL**, for short. In either case, if these extensions were proved to be True, this would create a mechanism for anti-matter to turn into matter, thus explaining one of the great problems in physics: The Matter-Antimatter Asymmetry. Or in other words, why is there *something* instead of *nothing*?

The Focus of this Project is on the **Heavy Neutral Lepton**, and how to find it (if it even exists). We focus on a small part of the many steps necessary to take in the hopes of finding it, which is on selecting the correct events from particle collider's.

2 Strategy for Observation

Now suppose the **Heavy Neutral Lepton** exists in the universe, how would we go about detecting it? Well all hopes lie currently in Particle Colliders, such as the Large Hadron Collider (LHC) [2] at *CERN, Geneva.* What particle colliders do, is that they accelerate particles to extremely high speeds and high energies inside a tube, then the SMASH them together. Because of how nature works, the particles that gets smashed together than transforms into several other particles, that get scattered in all possible directions, then those particles decay into other and lighter particles that we then detect in the ATLAS experiment [7]. We then count how many particles were detected, check their types, measure the energies, momentum, angles and etc... to determine which particles were created during the collision. It is akin to smashing two cars together then looking at the pieces that fell off to determine whether they were a Ferrari or a Lamborghini.

2.1 Signature Decay

Unfortunately the HNL is extremely rare and extremely short lived, such as is the case with many particles, we will never be able to detect them directly. But these particles decay into other, lighter and more common particles, which we can detect. And the exact combination of what particles come out of the decay, their energies, momentum and angles with respect to the collider beam, are usually very distinct from decays of other particles. The combination of particles that come out from a decay, their energies, momentum and etc... is what we call a **Signature Decay** of that interaction. Every particle, theoretical or real, has a very specific signature, and it is through that signature that we can recognize and know exactly which particles where creates during the collision.

2.2 Signal Signature

The signature decay of whatever particle we are searching for is what we call a **Signal Signature**. One of the possible Signal Signatures for the HNL is shown in 2.1. There are many more possible Signal Signatures for the HNL, but for the scope of this project i will only focus on this one.

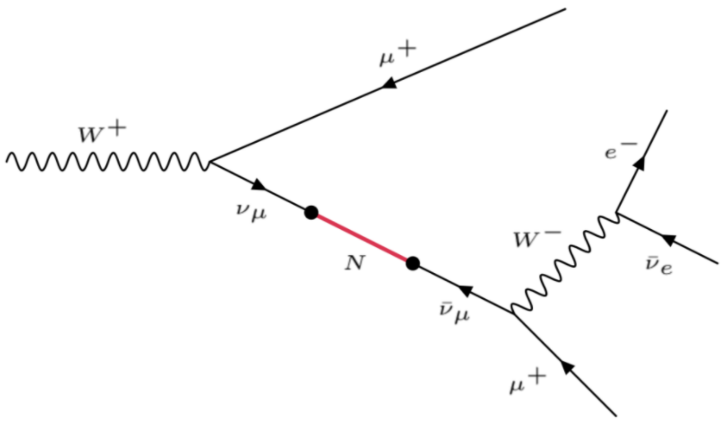


Figure 2.1: The Feynman Diagram of a possible HNL signature.

This is what can be seen happening in figure 2.1: a W^+ decays into a muon-neutrino (ν_μ^+) and a muon (μ^+). The muon neutrino then transforms into a Heavy Neutral Lepton (N) which then transforms into an anti-muon-neutrino ($\hat{\nu_\mu}$), which then further decays into a muon by emitting a W^- Boson that then decays into an anti-electron (e^-) and an anti-electron-neutrino ($\hat{\mu_e}$). The end product is then a pair of muons, an electron and an anti-electron-neutrino ($\hat{\mu_e}$).

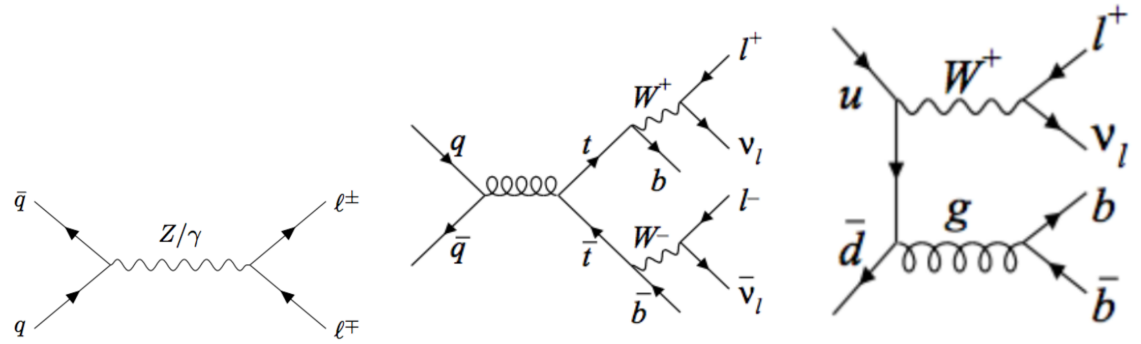
In practice, we will not observe the anti-electron-neutrino because they cant be observed. But we can infer their existence by calculating the total energy that went in, seeing the total energy that went out, and calculate the difference to see what is missing. This missing energy we call **Missing Transverse Energy** (TME)

We thus end up with the set (e^-, μ^+, μ^+) of particles. An alternative decay signal is where we start with a W^- Boson, then we und up with the conjugate particles: (e^+, μ^-, μ^-). In short, the decay signal is two Muons of the same charge, and one electron of the opposite charge, together with some missing energy.

2.3 Background Signatures

One of the things that further complicate our search, besides the rarity of the event, is that several other reactions, **not** originating from a HNL, that can imitate the exact same signal from a reaction involving the HNL. In some cases it may simply be because some particles have a trajectory such that they miss the detector, other cases are just very similar to the signal. In figure 2.2 we are shown a few examples:

In figure 2.2a we see a quark-antiquark annihilation reaction that produces either a Z-Boson or a Photon, they then decay into two a lepton and an anti-lepton. If two of these processes occurs simultaneously, there is a chance that two Muons of same charge and two Electrons of opposite charge can be created (e^-, e^-, μ^+, μ^+). If one of the electrons by chance escapes the detector, then we have left only (e^-, μ^+, μ^+), which is exactly the same signal as for the HNL.



(a) Drell-Yan Pair Production (b) t$\hat{t}$-production decay (c) $W^+ Background$

Figure 2.2: Some Possible Background Processes.

Another Decay Channels that can Imitate the HNL Signal, is when a top-antitop pair is created from a quark-antiquark interaction. As see in figure 2.2b, the top quark decays into a W^+ Boson and a bottom-quark, the W^+ Boson the decays into an anti-lepton and its corresponding anti-neutrino. The analogous decay happens for the anti-top quark. The b-quarks can then decay semi-leptonically and then we produce the same signal as for the HNL. We can filter out a great deal of this background by eliminating all events that has created jets of b-quarks (see section 9).
Lastly and most importantly for us, is the decay arising from the W^+-Boson, as seen in figure 2.2c, Here a W boson decays leptonically to produce a real Electron or Muon in addition to Missing Transverse Energy. The two other leptons can then arise from b-jets or photon-jets produced by way of the strong interaction. *This is the background that will be our focus of the project*
There are many more backgrounds that can happen in a particle collision, and all of them are much more likely to happen than the HNL signal. Even worse, as we have seen, many even look very similar to the HNL Signal. In fact, just the W^+ is 1000 times more likely to occur than the HNL Signal. So how would we go about finding the HNL if it exists? Well there is still hope, as will be proved in this project. One of the things is that, since the particles that are produced in the HNL Signal and the Background Processes are the same, the only distinguishing features are the kinematics of each Particle. By kinematics we mean the Energy, Momentum, Angles and time of each individual particle. We must look at these to find what distinguishes the HNL signal from the background processes.

3 The Goal of the Project

The aim of this project, is to use **Machine Learning** to remove as much Background events in the data as possible, while keeping the greatest amount of Signal. Thus increasing the ratio $\frac{Signal}{Background}$ to be as high as possible. This is equivalent to amplifying the Signal to a level we can observe. This is necessary because the Signal is much weaker than the background. In our case the background process of the W^+ Boson Happens 1000 times more often than the HNL Signal. This means the signal must be amplified at least 1000 times in order to be relevant.

The other goal is to do this with 2 different algorithms of Machine Learning, We will be comparing the performance of a **Neural-Network** method with a **Boosted Decision Tree** method, in how well each one can amplify the Signal-to-Background-Ratio as much as possible, as well as evaluating some other metrics. Both the **Neural-Network** and the **Boosted Decision Tree** will be explained in more detail later.

Part II

Methodology and Data

Here in part II, our approach to solving the problem of distinguishing Signal from Background Signatures will be explained. The focus will be on developing methods to best distinguish the W^+ Background from the HNL Signal. But the method used is very generalisable and can be used with every type of Background imaginable.

4 Obtaining Data

4.1 Simulations

Since the HNL is a theoretical particle and has never been observed, we must know what to look for. Since the calculations for something so complex and random as particle collisions are so difficult and impractical to make, we must use Numerical Simulations to imitate what would happen in a collision. That is Standard Practice in the field of Particle Physics. By running simulations we can simulate exactly what we would happen during the collisions of particles in colliders. We can then use that simulated data to develop methods of how to best search for the events that would be seen in real data from particle colliders.

The Simulations that were produced simulated Proton-Proton Collisions at 13 TeV, from those simulations were then selected only data where W-Boson's were produced. From those data, only the data of W^+ Bosons with 50-GeV were selected. From that data we could then select the simulation in which a HNL were created, and we could also specifically choose the events in which resulted in a specific Background Process. These two were then separated into their own data-sets, which I then got from my Supervisor in order to use for this project.

4.2 The Data

The data for the HNL signal and The Background Process contained many variables, but only 24 were selected. As both the HNL signal and background process produces only 3 leptons, we have variables for each of the 3 leptons. The selected Variables for each lepton that each Event contains were as follows:

For each of the 3 leptons, the variables for each event included:

- pdg-id: The Identity of the Lepton. Whether it is an Electron of Muon.
- Charge: Whether the Charge of the Leptons were +1 or -1 (In other words, was it an antiparticle or particle)
- p_t: The transverse momentum of the particle, (or rather, the momentum in the direction perpendicular to the collider beam)
- Energy: The energy of the particle.
- η: The *Pseudorapidity*, calculated by $\eta = arctan(\frac{p_L}{|\mathbf{p}|})$, where $\mathbf{p}$ is the 3-momentum, and p_L is the momentum along the beam axis.
- ϕ: The Azimuthal angle Around the beam axis

Besides that, we have following information:

- E_{MET}: The missing transverse energy (Pertains the Neutrino)

- ϕ_{MET} The Azimuthal Angle at which the MET flied off.
- n-bjets: Number of B-Quark jets produced

All of these variables are what was included for each of the two files, the one for the HNL Signal and the one for the W^+ Background. However the amount of events obtained in each file were different. For the W^+ Background there was exactly 248.234 Events Recorded, and for the HNL Signal we had 39.323 Events. So around 6 times more data for the W^+ Background then for the HNL Signal.

Now that i have explained the data and where they come from, i will from here on Refer to the W^+ Background data just as ”Background”, and the HNL signal data just as ”Signal”.

5 Motivations for Using Machine Learning

5.1 What is Machine Learning?

Machine Learning is a class of algorithms that can organize data, draw conclusions from data, and learn patterns in data sets such that it is able to make predictions about data it has not seen before. We are interested in the latter.

Machine Learning is a very powerful tool that has been growing in its wide range of uses around the world, it is a sub-area of artificial intelligence. What makes machine learning so powerful is its ability to effectively learn patterns in high-dimensional data that humans are not able to spot. Our eyes and brain are very good at observing patters in 2, maybe even 3-dimensional graphs, but anything beyond that is beyond our capability.

This makes Machine Learning ideal for use in particle physics, where we often have very large data-sets with many correlated variables. And the more data you have, the better machine learning performs.

5.2 Traditional Methods

Traditional Methods of removing backgrounds in data involve primarily making cuts in the distribution of data, for example, look at figure 5.1, it represents the distribution of the transverse momentum for the third lepton. Normally you would choose a point on the X-axis, for example at the red line on fig 5.1 and accept remove all of the data for signal and background above or below a certain point. The amount of background removed depends on the location of that point. Usually this process would be Repeated for Several Distributions of Variables, and you could also draw distributions of certain combinations of variables, and perform cuts. The goal is of course to remove as much background While keeping as much Signal As possible

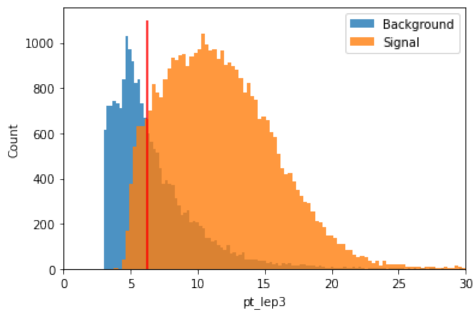


Figure 5.1: Histogram of Transverse momentum for Signal and Background data: Red line is a cut. Blue is Background, Orange is Signal

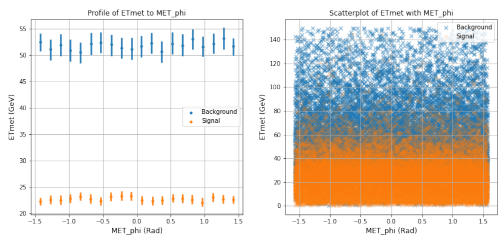
Other Traditional Methods Involve finding linear correlations in data and using advanced statistical methods to make a seperation. Usually a researcher combines many methods together to make the separation.

5.3 Disadvantages of Traditional Methods

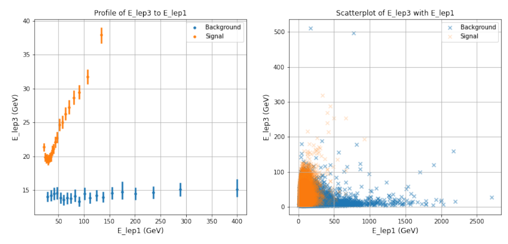
The disadvantage of Traditional Statistics and The Cut Based Method is that they often are not able to take advantage of nonlinear correlations in data. Other Disadvantage is that the huge amount of data obtained in particle colliders require an anormous amount of work and manual inspection. This is a Problem That is Solved with Machine Learning, as you just Feed the algorithm some data, and it learn automagically.

5.4 Correlations in Data

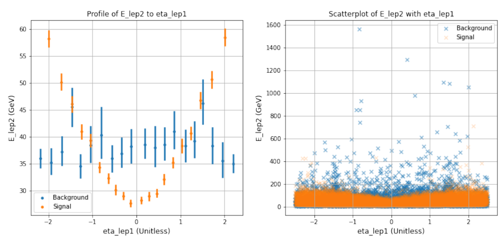
Before we even decide if it is worth it to investigate further in the data sets, we must find out if there is a significant difference in the distributions and relationships between the Signal and Background. To see this, you would manually plot every combination of variables and see is the data looks random of if there are some kind of relationship between them. This is done for 4 different combinations of variables in figure 5.2.



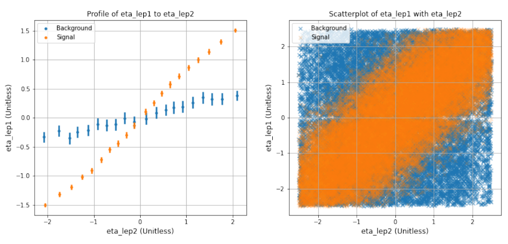
(a) X= ϕ_{MET}, Y= Missing Transverse Energy



(b) X= Energy of Lepton 1, Y= Energy of Lepton 3



(c) X= η of lepton 1, Y= Energy of Lepton 2



(d) X= η of lepton 2, Y= η of lepton 1

Figure 5.2: Scatterplot and Profile Plots for 4 combinations of Variables of our data: Left= ScatterPlot, Right= Profile Plot

On the Right we have a scatterplot of the different points in the X-Y combination of data, on some it looks random and chaotic. But if you bin together the Y-values for bins in X-axis, you can see suprising pettersn emerge. This is plotted on the left of each plot, and is called a profile plot. You can clearly see interesting correlations, and even nonlinear ones. This is supports the claim that Machine Learning would have an effective use case here.

6 A deeper dive into Machine Learning Models

There are Several different algorithms of machine learning, the types we are going to use falls into the class of **Supervised Learning** for classification, this means that the model works by getting a large data-set. The data-set is then split into a 2 different set. 1 set for *Training* and another set for *Testing*. The sets are usually an array that has Rows for its datapoints, and columns for variables, with the last column being the **Classification**. For example, in our case we have 24 Variables, several thousand data-points, and 2 classifications (Signal or Background). The algorithm then processes the Training-data, it generally works like this:

First it looks at all of the datapoints, takes a note of the variables. It then assigns a class to each datapoint. Then it compares the prediction it made about the class of the datapoint, to the actual classification. Based on this result, it then adjust its internal parameters using complicated mathematics. After this it repeats the process, multiple times. It's aim is to predict the class as accurately as possible, for as many datapoints as possible. It only stops when the score for the accuracy of its prediction stops improving, or until some other criteria is met. This entire process is called **Training** of the machine learning model.

Now let us look at the two different types of machine learning algorithms i have decided on using for this project.

6.1 Neural-Networks

What you see in figure 6.1 is a Representation of a **Neural Network**. A Neural Network is a Machine Leanring Model that Consists of Several Layers of Nodes. In each Node is an *Activation Function*, which is normally a sigmoid, like the logistic curve. We have an input layer, where variables are entered, then everything in the middle consists of several layers of nodes that are connected to each other. Each Layer between the input layer and output layer are called **Hidden Layers**, each hidden layer can have a certain amount of nodes. The connections between the nodes have certain weighs, which are the parameters that get altered during Training. The Output layer consist of the set of predictions made by the neural network for each datapoint.

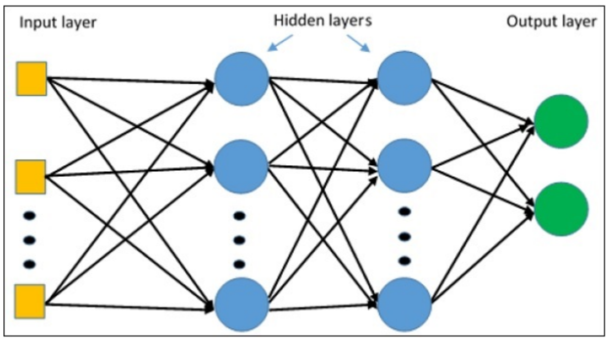


Figure 6.1: Illustration of a Neural Network with 2 Hidden Layers

A **Deep Neural Network** is a neural network with many hidden layers, and a lot of nodes in each layer. Other than that, there are different types of neural network, which is determined by how the nodes are connected and the order in which the data passes through the nodes. For this Project i will be using the type of Neural Network called a **MultiLayer Perceptron**, which is a Deep Neural Network with back-propagation [1] more information on MultiLayer-Perceptron Neural Networks can be read here: `https://scikit-learn.org/stable/modules/neural_networks_supervised.html`.

6.2 Boosted Decision Trees

6.2.1 Decision Trees

A Decision Tree is another type os Machine Learning Model, that has the same uses cases as a neural network, but they are structured differently. In figure 6.2 we see a very simple example.

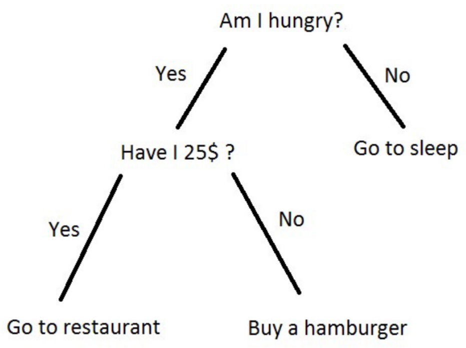


Figure 6.2: A Very Simple Decision Tree

The decision tree works by chossing a variable, or logical combination of variables, and defines a condition for which there is a "yes" or "no". Based on that it will then make a decision. This then repeats itself. A Decision tree is then a tree of decisions, of paths it can take. The example in figure 6.2 is extremely simple, has only 3 layers. But a The trees generated for our data can have hundreds of layers. The Advantage of DecisionTrees Compared to Neural Networks is that they are "Transparent". You can easily know how it works by followijg each decision branch. Where a Neural-Network is so complex and interconnected that it is hard to gain any understanding of how it makes the predictions. Another advantage is that they are very wuick to Train. The Disadvantage is that it can easily **Overfit** the data, which means that it performs very well on the data it trained on, but can not be generalized to data it has not seen before. This is called the **Bias-Variance Tradeoff**

6.2.2 Boosted Decision Trees

The process of Boosting in machine learning is when you take several ”Weak Learners” together to make a prediction. In the case of decision trees, rather than have 1 large tree that is Hundreds of Layers Deep, you take many trees with just 1 or 2 layers, and make them ”Vote” and a prediction. The Prediction with most number of votes will be chosen.

6.2.3 LightGBM

The Algorithm, or Model i am Going to use for the Boosted Decision Tree is one called LightGBM. More information about it can be read at: `https://lightgbm.readthedocs.io/en/latest/#` . LightGBM is currently the best algorithm based on a boosted decision tree, it has won numerous machine learning competitions, which is why i am using it for the project.

6.3 Using Supervised Machine Learning in Practice

The Way that machine is used, whether a Neural Network (NN) or a Boosted Decision Tree (BDT), is that first you Feed it Data From the Training Set, to train the model, you can also say you ”fit” the model to the data. When the model is then trained you evaluate it on the **Testing Set**, which is data it has not seen before. The model will then make predictions for the probability that the datapoint belongs to a certain clas (Signal or Background), you then compare the prediction with the truth-value of the class. You can evaluate the model based on various metrics. For simple cases the metric used for evaluating the perfomance of the Model is its **accuracy**, which is the number of predictions predictions it got right, divided by total number of predictions. But for our purpose we need more advanced ways to evaluate the models, which is the topic of this next Section.

7 Evaluating Machine Learning Models

7.1 Perfomence Metrics

7.1.1 Confusion Matrix

When you take the predictions of the Model on a test-set and compare it to the actual classes of the test set, you can get 4 different Results, which are represented by a **Confusion Matrix**, as seen in fig 7.1. Usually the class we are searching for we call *Positive Class*, in this case ”Signal” is the positive class, and the class we want to filter out is called the *Negative Class*, in this case ”Background”.

As you see in figure 7.1, Positivies that were predicted to be Positive are Called **True Positives (TP)**, Positives predicted to be negative are called **False Negatives (FN)**, Negatives Predicted to be Negative are Called **True Negatives (TN)**, Negatives Predicted to be Positive are called **False Positives (FP)**.

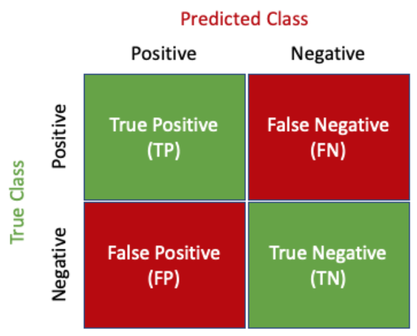


Figure 7.1: Confusion Matrix

7.1.2 Accuracy, Precision, True Positive Rate and False Positive Rate

The accuracy of the model, as written in section 6.3, is the amount of predictions it got right divided by total number of Predictions. So in other words $accuracy = \frac{TP+TN}{TP+TN+FP+FN}$.

The **True Positive Rate (TPR)**: $TPR = \frac{TP}{P} = \frac{TP}{TP+FN}$ represents how many positives were correctly predicted among the populations of positives.

The **False Positive Rate (FPR)**: $FPR = \frac{FP}{N} = \frac{FP}{FP+TN}$ Represents the amount of Wrongly Predicted Positives Among The Negative Population.

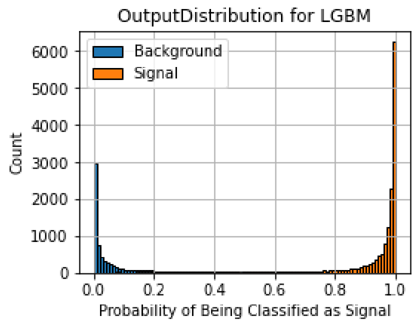
The **Positive Predicted Value (PPR)** $PPV = \frac{TP}{TP+FP}$ also known as *Precision* Represents the amount of Correctly Predicted Positives of the Positive Population

These are important because the FPR is a measure of how much Background we will expect to Wrongly classify as Signal after the data gets processed by the model. Or how much pollution remains in the data. Therefore it is also know as **Background Efficiency**. The TPR is important because it represents how much data of the Signal will be kept after being processed by the Machine Learning. Therefore it is also known as the **Signal Efficiency**.

7.2 Explaining the ROC-AUC-Curve

7.2.1 Decision Threshold

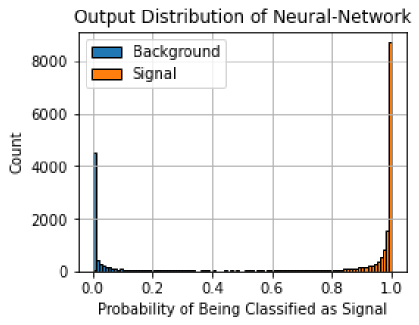
As mentioned briefly in in section 6.3, what the Machine Learning Models output for each datapoint is a Probability that the point belongs to a certain class. The decision to actually classify an event as "Signal" or "Background" will then actually depend on a value we can decide. The default is always set to 50%. Such that is the Probability of being signal is higher than 50%, the model will classify it as Signal, if less, it will classify it as background. The value for that cutoff point is called a **Decision Threshold**, or just **Threshold** for short.



(a) Class Seperation for LightGBM before Optimization

7.2.2 Output Distribution

If we take all of the predicted probabilities that the model has put out, make a histogram of all the probabilities for the background, and another one for the signal, and plot both of these histograms on the same graph, we get an **Probability-Output-Distribution** that measures **Class Separability**

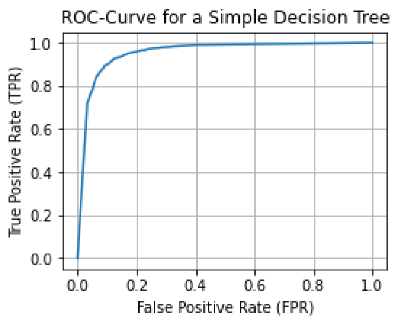


(b) Class Seperation for NeuralNetwork before Optimization

7.2.3 ROC-AUC Curve

[3]

Finally, the ROC-AUC curve. This is the most prevalent metric in this Project. The ROC curve is simply a plot of the Signal Efficiency vs the Background Efficiency for different Thresholds. How? Imagine you in the beginning set the Decision Threshold (Which i will call just as Threshold from now on) at 0. Then it will classify all data points as Signal! So the TPR will be 100% and the FPR will be 0%.



(c) A ROC-Curve

If we then make it a bit higher, say 1%, then the TPR will decrease and the FPR will Decrease. If you do this for all thresholds, then plot the TPR and FPR for each threshold value, you get the ROC CURVE!

What the ROC-Curve tells us, is, for a given threshold,
what Efficiencies for signal and Background should we expect from a given model? And by inspecting the ROC-Curve, we can check what Background Efficiency to expect if we chose a certain Signal Efficiency. In figure 7.2c you can see a ROC-Curve plotted for the perfomance of a simpe decision tree on our data.

8 Hyper-Parameters

8.1 What are Hyper-Parameters?

A model, such as a neural network, still has some flexibility in precise details of how it is structured. For Example, a Multi-Layer Perceptron consists of many layers of nodes, where each node in one layer is connected to each node in the next layer. The Number of nodes in each Layer, and the number of Layers, is a parameter that can be adjusted. It is not a parameter that is adjusted during training of the model, but it stays fixed, and affects how the model gets Trained. These types of Parameters is what we call Hyper-Parameters. Hyper-Parameters are basically parameters that decide the Architecture of the Model. For the LightGBM, examples of hyper-parameters are for example the max number of leaf nodes in a tree, the total number of trees produced, the max depth of each tree and etc...

8.2 Hyper-parameter Optimization

Since the Hyper-Parameters of the model determines how the training-dataset ends up adjusting the models internal parameters, different sets of Hyper-Parameters thus determines its performance. There is no unique set of Hyper-Parameters that generally does best for all data-sets, so the models Hyper-Parameters must be adjusted, through various methods, so that it can optimize its performance on the dataset it is being designed to train and test on. The Process of adjusting the Hyper-Parameters is called **Hyper-Parameter Optimization**. There are various methods of Optimizing Hyper-Parameters, the old way of doing it would be to adjust them one at a time, train the model an see if there is any improvement. Luckily today we have methods of adjusting hyper-parameters automatically. The one method that was used to find following hyper-parameters is one called **Bayesian Optimization**, which is based on Bayesian Probability.

The best parameters found for the BDT were:

('boosting type'= 'gbdt'),('learning rate'= 0.08070423147486032),('max depth'= 10),('minimum child samples'= 315),('minimum child weight'= 17),('n estimators' = 869), ('number of leaves'= 20,)'reg alpha'= 2, ('reg lambda'= 5)

And the best parameters for the neural network were:

('activation'= 'tanh'), ('alpha'= $1e-07$), ('beta1'= 0.5545561677659581), ('beta2'= 0.5713485660228933), ('epsilon'= 0.000492036188902597), ('hidden layer sizes'= 200), ('initial learning rate '= 0.02199012604107307), ('max iterations'= 537), ('solver'= 'adam'), ('tol'= 0.049795769219370685)

Information of what each hyperparameter mean can be found at:
`https://lightgbm.readthedocs.io/en/latest/pythonapi/lightgbm.LGBMClassifier.html`
for LightGBM.
And at:
`https://scikit-learn.org/stable/modules/generated/sklearn.neural_network.MLPClassifier.html`. for the Neural Network.

9 Pre-Selection

Before we Train the Machine Learning Models, we must see if there is a quick way to remove as much Background as possible, all while reducing the number of variables. This procedure is called pre-selection.

Since the signature signal for the HNL Decay is either a (e^-, μ^+, μ^+) or a (e^+, μ^-, μ^-) combination. We can select only the data events that produced this combination in the signal and background.

Another thing we can do is to remove all events that produced b-jets, since we know that the HNL signal produces almost no B-jets. Below is a table of the datapoints left after doing this selection.

	Signal Events Kept	Background Events Kept
Before Pre-Selection	39323	248234
After Removing Bjets	38129	196913
After Selecting Lepton Combination	38987	24845
After Applying Both	37814	19710

As we can see, just by doing this we have Removed 92.1% While keeping 96.2% of signal!

Part III
Final Analysis

Now Having Optimized the Hyper-Parameters, we are ready to get the results, including the ROC-Curve, and the OutPutDistribution for the two Models, which can be seen in figure 10.1

10 Preliminary Results

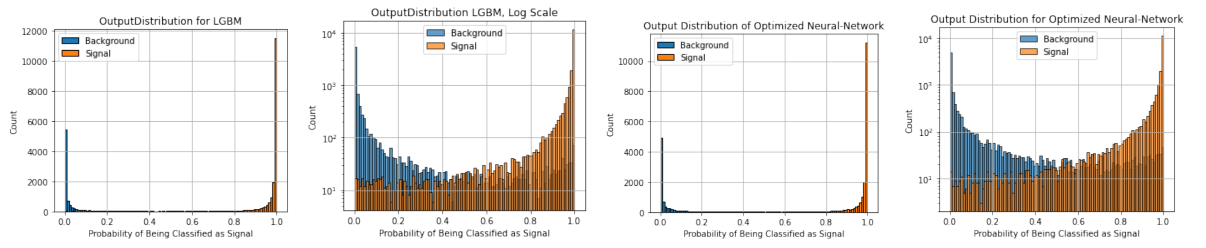


Figure 10.1: OutPut Distributions for The Neural Network, and the Boosted Decision Trees. In both normal and Logarithmic Scale.

As can be seen in figure 10.1, the separation between signal and background is very good! On the logarithmic plot we can see around 3 orders of magnitude in difference between True Positives and False Positives, as the threshold approaches 1! This already signifies that at very high threshold values, we can get the Background Efficiency down to around 0.01%

10.1 ROC-Curves of the Neural Network and LightGBM models with Optimised HyperParameters

Finally, here is the resulting ROC-Curves. I done 5 iterations for each Model. Since we are Interested in what happens at the lower left corner of the ROC-Curve, i have here plotted them with the x-axis logarithmic.

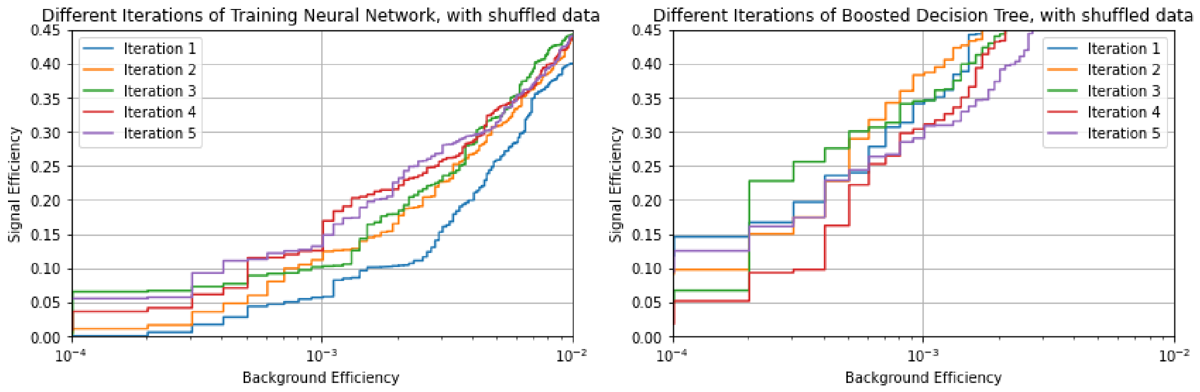


Figure 10.2: On the Left: Neural Network, On the Right: LightGBM

What you see in the two figures, is the roc-curves. You see 5 ROC-Curves in each figure, because the machine learning models were trained 5 times and graphed. With each training we made sure to shuffle the data around. This is important because, in the real world, the data we have available is only a sample of a much larger population. This is the case with our data as well. The Simulation of the decay of the particles is essentially a random process. And particle decays in reality are also a random process. So by training on a random subset of the data, we can see tiny variations in the ROC-Curve. We should imagine the data we have available, and data we CAN generate, as being a small sample of an infinitely large distribution of possible outcomes.

10.2 Stability and Variability of the ROC-Curves

Lending into the idea discussed in 10.1, we know that the Trained Models vary in performance, depending on what subset of the data that is used during training. Therefore, in order to properly assess the models, we must have a certain level of confidence in its prediction, the question then becomes: ”How confident can we be in the prediction of the model?”

The way this problem was addressed, was to make an ”Average ROC-Curve” of many possible ROC-Curves, where we train each of the two models many times, each time on a different random subset of the data (this is called **Resampling**), and produce a ROC-Curve for each iteration. The Average ROC-Curve would then be obtained by averaging over all of the ROC-Curves Produced.

Of course, a normal person would just make 1 ROC-Curve, and call it a day. But that would not be very useful for real-world applications of machine learning, especially not in particle physics, where the reduction of background noise is crucial to claim something a discovery. Therefore, the rest of the analysis is based on estimating uncertainties, expected values and confidence around these models.

11 Mean ROC-Curves and their Variance

11.1 The path to 1000: The Algorithm

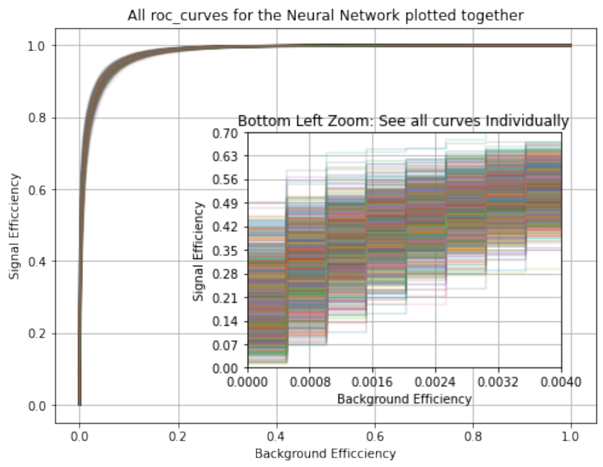
It was decided that 1000 ROC-Curves for each model would give us enough statistics to estimate the statistical properties we are after. Now, a ROC-Curve is just a plot of the Signal Efficiency (True Positive Rate) against the Background Efficiency (False Positive Rate) for every given threshold between 0 and 1. Therefore it is also possible to make other types of plots with this algorithm.

The way the algorithm to 1000 ROC-Curves works is as follows:

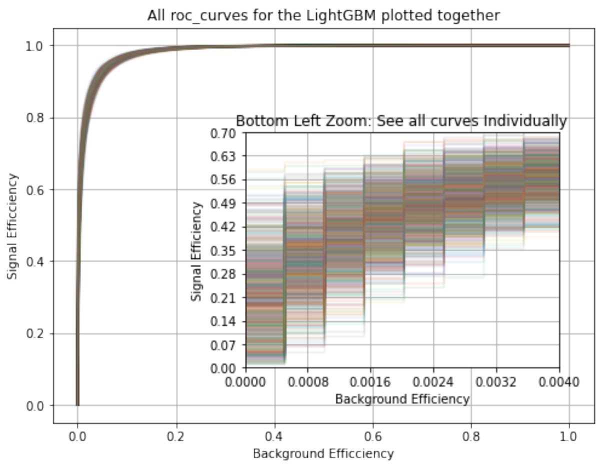
1. Seperate the Data into 10 equal sized bins, called Folds.
2. Train on 9 of the Folds, and test on the remaining Fold. Then Calculate the Threshold values, The Signal Efficiency and Background Efficiency for each Threshold. Then save the Result.
3. Repeat step 2 for each of the other folds.
4. Re-Shuffle The data around randomly and restart from step 1. do this 100 times.

More information on generating ROC-Curves can be read in: [3]. And the exact Algorithm can be found on the GitHub Repository for this Project [5]. Next section explores the results.

11.2 Results: Mean ROC-Curve



(a) ROC CURVES from the Neural Network

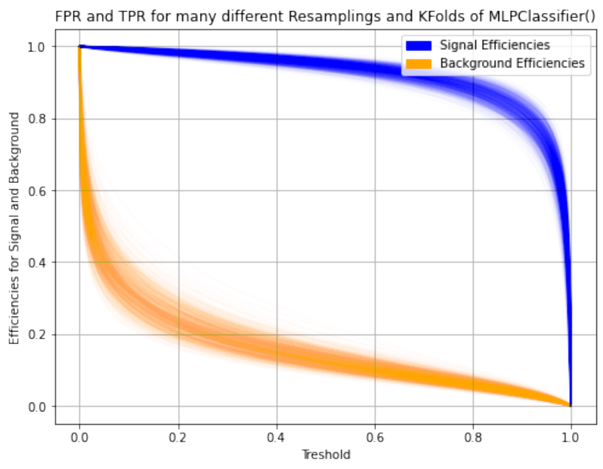


(b) ROC CURVES for Boosted Decision Trees

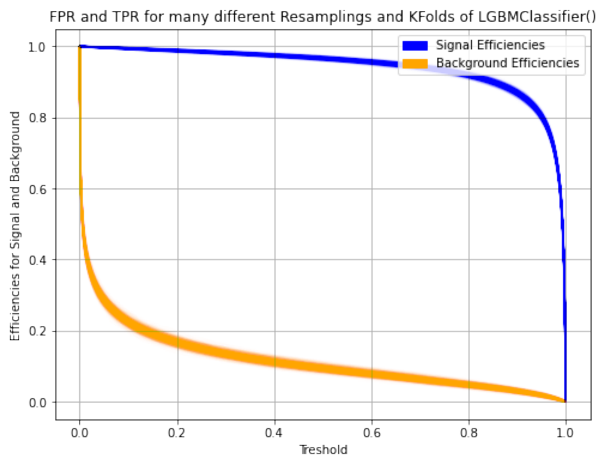
Figure 11.1: ROC CURVES from the Neural Network (Left) and Boosted Decision Trees(Right): In the zoomed in panel we can see all of the different roc-curves and approximately sense how dense they are.

Figure 11.1 Shows all of the 1000 ROC-Curves Plotted together, both the entire ROC-Curve and, in the Zoomed Window you can actually see all of the 1000 curves individually, because each one is coloured differently. It is already possible to sense that the Boosted Decision Trees are outperforming the neural Network, because it is higher on the plot. Just this first plot already gives us a good idea of the Variance of the ROC-Curves. But what is needed is to take the Quantitative Mean and Standard Deviation from all of these ROC-Curves.

In figure 11.2, we have a different plot. We have Plotted the Signal Efficiency and Background Efficiency as a function of threshold values, for all of the 1000 iterations. the X-axis is the threshold values, the blue and yellow lines are the Signal and Background Efficiencies



(a) Neural Network

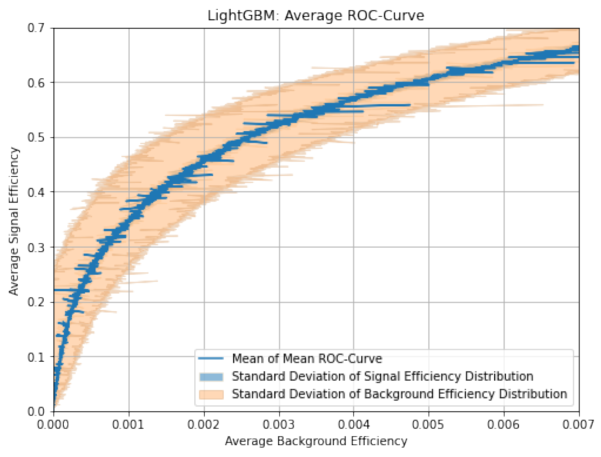


(b) Boosted Decision Trees

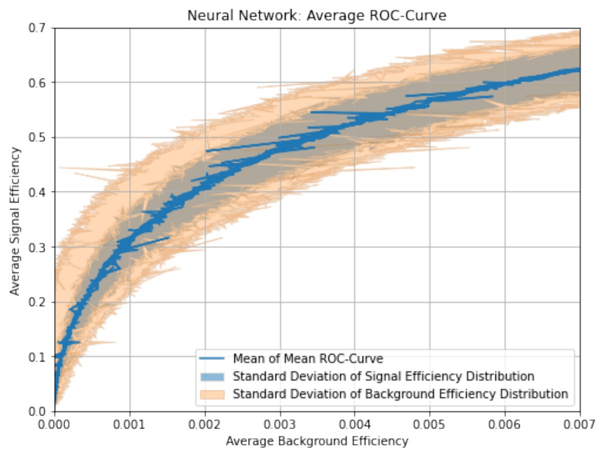
Figure 11.2: Background Efficiency (FPR) and Signal Efficiency (TPR) as a function of Threshold Values for both Neural Network and Boosted Decision Tree

Respectively. (Already from this plot you can see a difference between the Neural Network and The Boosted Decision Trees. The Variance of the Neural Network seem to be much Higher, you can see on the plots that the yellow and blue bands in the plot for neural network has a larger spread compared to the one for the boosted decision trees.)

The way to create the average ROC-Curve with a Standard Deviation, is by iterating over all of the thresholds, point by point, averaging all of the False Positive Rates, and True Positive Rates (TPR), at that Point. Point by Point you also calculate the Standard Deviation. Doing This, you end up with 4 arrays, one array containing the Mean False Positive Rates (FPR), another containing its Standard Deviations. And one array containing the Mean False Positive Rates, and another containing its Standard Deviations.



(a) Mean ROC-Curve for the Boosted Decision Tree



(b) Mean ROC-Curve for the Neural Network

Figure 11.3: Plot of the Mean ROC-Curve with it's Variance: produced by doing 10-fold Cross Validation, with 90% Training and 10% Testing set size, and then taking the ROC-Curve for each of those folds, We Illustrate the Standard Deviation of the FPR and TPR as coloured bands around the mean in the X and Y axis, respectively.

By plotting The Mean TPR and mean FPR on a graph, you get the Mean ROC-Curve as seen in figure 11.3. This method of finding the mean ROC-Curve is called *Threshold Averaging*[3]. In figure 11.3 we then represent the Standard Deviations in the TPR and FPR by colouring the space between the mean and 1σ. The brown band is 1σ only 1n the X-axis, for the TPR we look at the light-blue band, and it is only valid for the Y-axis.

As can be seen on the plots, the average ROC-curve (dark blue line) seems to behave very erratically, jumping around all over the place. But the brown and blue areas nonetheless seems to give us a very good estimate on the variance. This will be investigated further in the next section. The blue line in the middle is the calculated average. One of the things that becomes apparent is that the curve for the Neural Network has a slightly larger variance, but both seem occupy the same place, which suggests equally good performance.

In the next section, i will be doing an in-depth analysis of the distribution of Background and Signal Efficiencies for Certain Threshold Levels, and try to estimate a confidence level for the Prediction of the models.

12 Model Assessment and Final Results

Now for the final assesment and comparision of the two models. First lets have a look at the distributions we get for choosing a threshold and seeing how the FPR and TPR is Distributed.

12.1 Figures of Efficiency Distributions for LightGBM

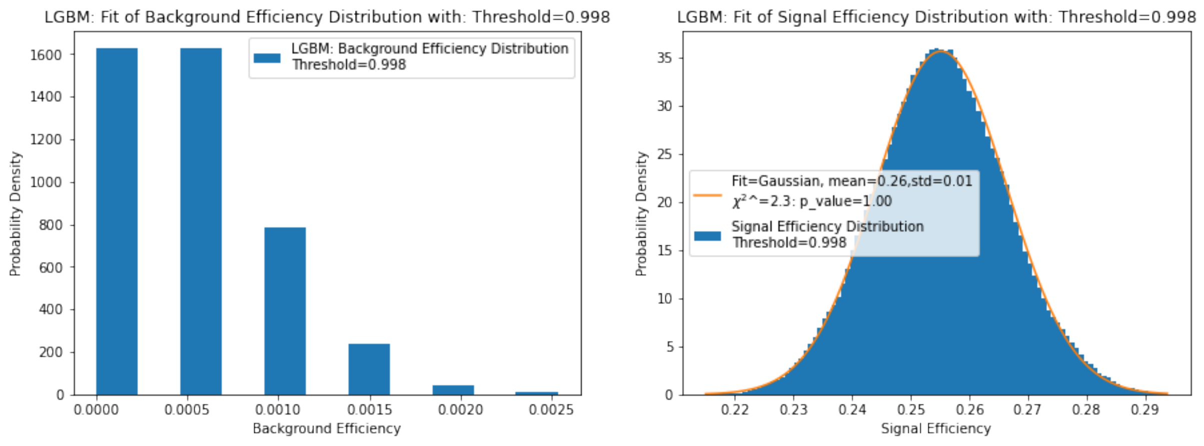


Figure 12.1: Distributions of the LGBM Background and Signal Efficiencies for threshold value of 99.8%

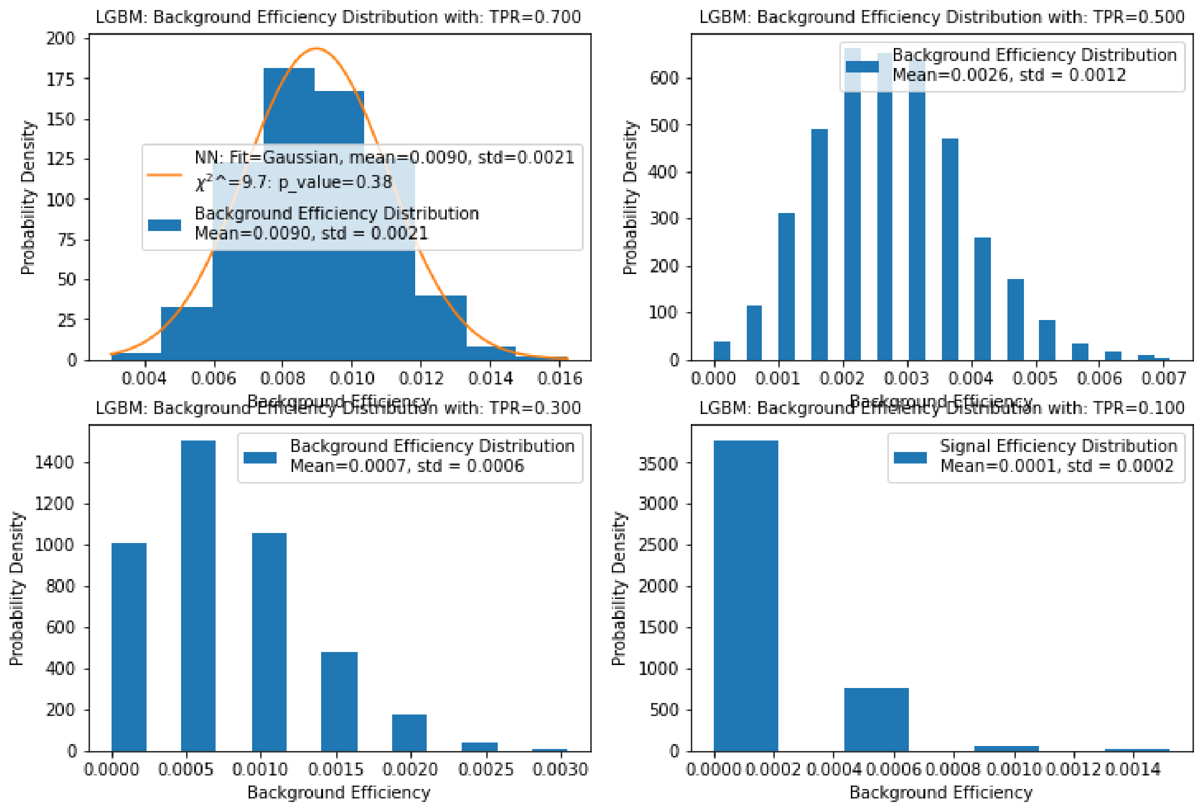


Figure 12.2: By choosing a value for TPR for the LightGBM, these were the Distributions of the Background Efficiencies

12.2 Figures of Efficiency Distributions for MultiLayer-Perceptron

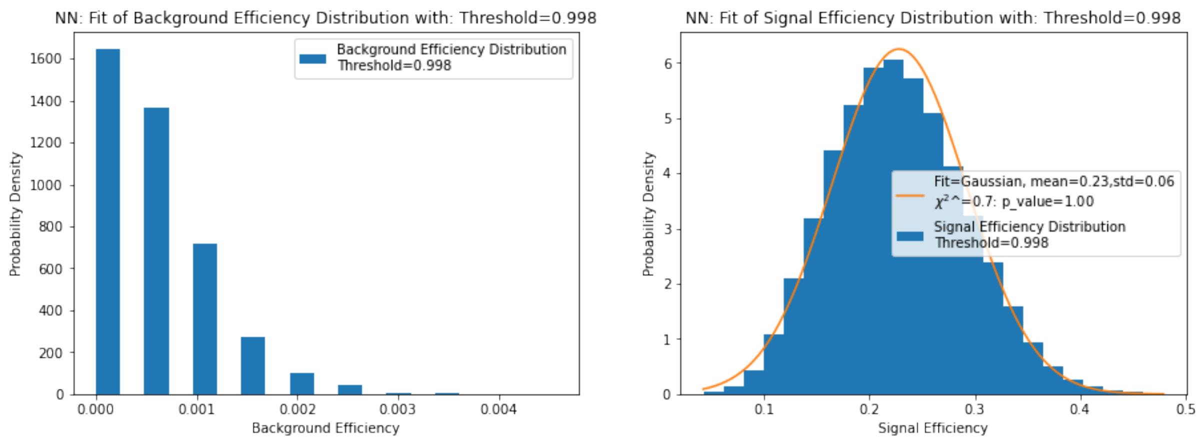


Figure 12.3: Distributions of the MultiLayer-Perceptrons Background and Signal Efficiencies for threshold value of 99.8%

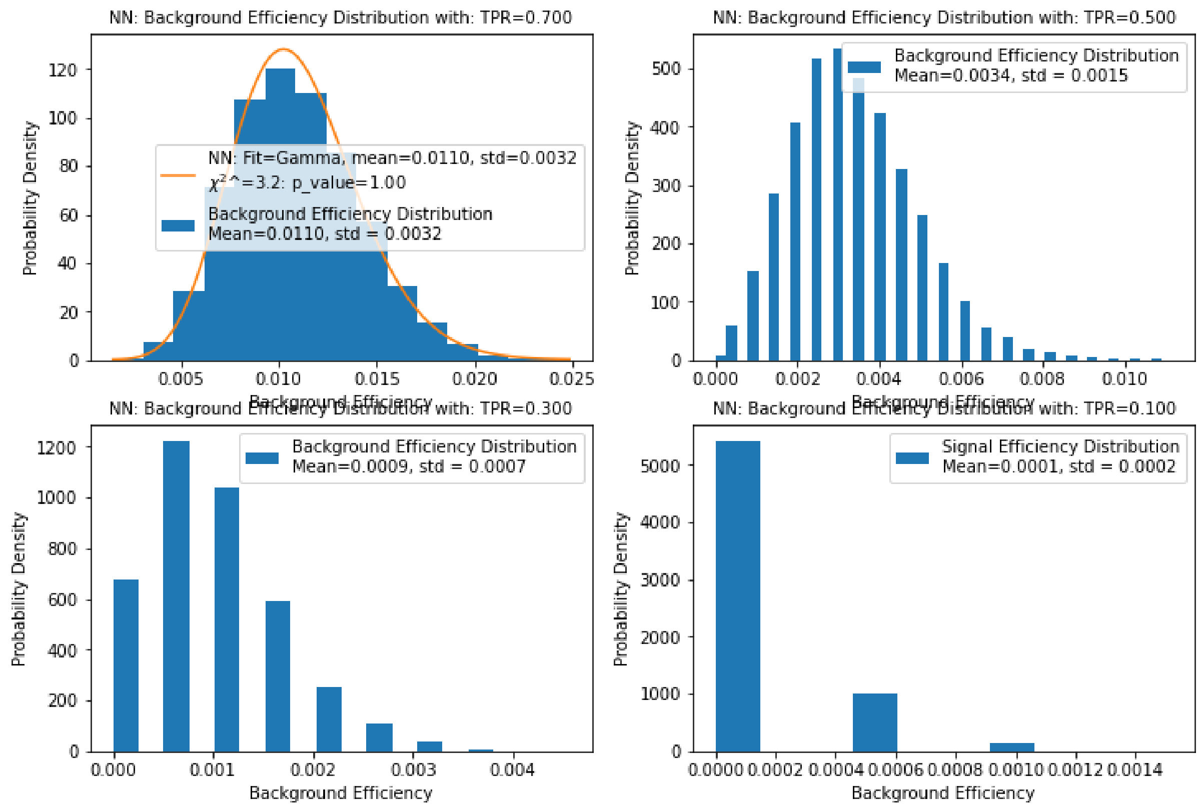


Figure 12.4: By choosing a value for TPR for the MulitLayer-Perceptron, these were the Distributions of the Background Efficiencies

12.3 What can we conclude from the Distributions?

The first things we notice is that they appear to be Poisson Distributed, which is good, because the equation Signal Significance $\frac{S_{expected}}{\sqrt{B_{expected}}}$ assumes the expected Background is Poisson Distributed. This gives us confidence in using the mean of the distributions as a measure for the

Signal-to-Background ratio ($\frac{S_{expected}}{B_{expected}}$), which measures the ratio between expected signal and expected background after applying the preselection and Machine Learning.

From the figures 12.1 and 12.3 we can conclude that the the Signal Efficiencies are Normally Distributed even at a very high threshold value. (p-value = 1 on the χ^2 test).

The χ^2 test only works when at there are at least 5 different values in each bin [4], but because of our low statistics, i can't use the χ^2 test. Maybe there are other statistical methods to test how well the FPR follows a poisson. But it looks very Poisson!

Now, to calculate our Signal-To-background-ratio and Signal-Efficiency, we must first know how many Total number of Signal And background events will happen in a Run, this is calculated below

$$\begin{aligned} S_{Total} &= (W^{+}\text{0-jets cross-section }) \times (\text{running time in Run2 in } fb^{-1}) \\ &\quad \times (\text{Current experimental upper limit on HNL mixing for M = 20-GeV}) => \\ &= 20nb \times 10^{-5} \times 140fb^{-1} \times 10^{6} = \underline{\text{28.000 Signal Events}} \end{aligned}$$

$$\begin{aligned} B_{Total} &= (W^{+}\text{2-jets cross-section}) \times (\text{running time in Run2 in } fb^{-}1) => \\ &= 0.2 \times 140 \times 10^{6} = \underline{\text{28.000.000 Background Events}} \end{aligned}$$

Now to get the number of expected Signal and Background Events, we must multiply the S_{Total} and B_{Total} by their respective efficiencies, that we got in this project.

$$S_{expected} = S_{Total} \times S_{eff}(\text{Machine Learning}) \times S_{eff}(\text{Preselection})$$

$$B_{expected} = B_{Total} \times \overline{B_{eff}}(\text{Machine Learning}) \times B_{eff}(\text{Preselection})$$

Where S_{eff} and B_{eff} are the signal an background efficiencies found in the machine learning or the preselection. and $\overline{B_{eff}}$ is the mean of the background efficiency distribution(s), as seen in figures 12.2 and 12.4. The results of these calculations are presented in the two tables below.

Tables 1 and 2 show us the values we would expect to get by applying the machine learning algorithms new data. Table 1 show us the perfomance of the LightGBM, table 2 shows us the perfomance of the Neural-Network. The First Column are some chosen values for the Signal Efficiency. The Second and Third columns shows us how much signal and background we would expect for ach of the signal efficiencies. The Fourth Column is our Signal-to-Background Ratio, while the fifth column is the Signal Significance.

S_{eff}(LightGBM)	$S_{expected}$	$B_{expected}$	$\frac{S_{expected}}{B_{expected}}$	$\frac{S_{expected}}{\sqrt{B_{expected}}}$
0.5	13468.00	5841.93	2.31	176.21
0.4	10774.40	3107.83	3.47	193.27
0.3	8080.80	1534.27	5.27	206.30
0.2	5387.20	620.99	8.68	216.18
0.1	2693.60	209.66	12.85	186.03

Table 1: Values we would expect to get by applying the Trained LigthGBM model on Data for Distinguishing HNL Signal and the W^{+} Background.

We can see that the models are very similar in performance, with the neural network tending to get values that are only slightly worse. But i judge this to be an artifact of random statistical fluctuations. We can't know for certain without more statistics. But due to the fact that neural

S_{eff}(Neural-Network)	$S_{expected}$	$B_{expected}$	$\frac{S_{expected}}{B_{expected}}$	$\frac{S_{expected}}{\sqrt{B_{expected}}}$
0.5	13468.00	7360.22	1.83	156.98
0.4	10774.40	3925.18	2.74	171.97
0.3	8080.80	1947.64	4.15	183.10
0.2	5387.20	803.49	6.70	190.05
0.1	2693.60	198.64	13.56	191.12

Table 2: Values we would expect to get by applying the Trained MLPClassifier() model on Data for Distinguishing HNL Signal and the W^+ Background.

networks are much slower to train, and the training time scaled badly with more data, it is best to use the LightGBM Algorithm, Since it wins in Training Speed.

However, for both methods, these results are marvelous! Before we underwent this process of preselection and machine learning, the $\frac{S}{B}$ ratio was $\frac{1}{1000}$, but by first cutting some events based on the pre-selection procedure, the training either the MLPClassifier or the LightGBM-Classifier on the simulated data, We can successfully bring up the $\frac{S}{B}$ upto around 13! That is an improvement by a factor of 12.000!

But this may only be the beginning, with more statistics, training data, and maybe even other algorithms, we may be able to improve it even further. In principle, we could set the Decision Threshold even higher to obtain a greater amplification of the results. But we need more statistics to measure exactly how much more improvement can be made.

Conclusion

I have proved that machine learning is an extremely powerful tool for amplifying the HNL signal, and i have showed that there is hope for finding the HNL in the future by using these methods. Furthermore, this method provides hope for finding the HNL in the future, as I have proved that these methods can significantly improve the Signal To Background Ratio of the HNL Signal, and may even be generalised to use of distinguishing the HNL from other types of background!

Bibliography

[1] URL: `https://scikit-learn.org/stable/#`.

[2] *CERN: The Accelerator Complex.* URL: `https://home.cern/science/accelerators`.

[3] Tom Fawcett. "An introduction to ROC analysis". In: *Pattern Recognition Letters* 27 (2006), pp. 861–874. URL: `https://www.sciencedirect.com/science/article/pii/S016786550500303X`.

[4] Mary L. McHugh. "The Chi-square test of independence". In: *Biochemia Medica* 23(2) (2013), pp. 143–149. DOI: `10.11613/BM.2013.018`.

[5] *Project Repository.* URL: `https://github.com/svendstar/BachelorProject`.

[6] M. E. Shaposhnikov S. N. Gninenko D. S. Gorbunov. "Search for GeV-scale sterile neutrinos responsible for active neutrino oscillations and baryon asymmetry of the Universe". In: *arXiv* (2013). URL: `https://arxiv.org/abs/1301.5516`.

[7] "The ATLAS experiment at the CERN large hadron collide". In: *Journal of Instrumentation 3* (2008). DOI: `10.11613/BM.2013.018`. URL: `https://www.researchgate.net/publication/224936536_The_ATLAS_experiment_at_the_CERN_large_hadron_collider`.