A Search for Exotic Higgs Decay to the 4b Final State

A Major Qualifying Project
submitted to the faculty of
WORCESTER POLYTECHNIC INSTITUTE
in partial fulfillment of the requirements for the
degree of Bachelor of Science

BY:

Kaitlyn Morrison

DATE: May 6, 2022

ADVISORS:

Professor Raisa Trubko – *WPI*

Professor Rafael Coelho Lopes de Sá – *UMass Amherst*

WPI

UMass Amherst

Abstract

The search for exotic Higgs decay has been an active area of research since the discovery of a Higgs Boson in 2012 at the ATLAS and CMS detectors at the Large Hadron Collider (LHC) at CERN. These searches use the large number of Higgs Bosons produced in the LHC to investigate beyond the Standard Model physics. Naturally, this is a major area of research and various theories have been proposed and are being investigated. This project focuses on the possibility of a new particle (represented by "a" here), which would be similar to the Higgs Boson particle and would be produced in the decay of the Higgs boson found in 2012. The Higgs would then decay into two a bosons that would subsequently decay into four b quarks ($H \rightarrow aa \rightarrow bb\bar{b}\bar{b}$). Searching for this decay in the events recorded by the ATLAS detector is a difficult process as this decay is expected to be very rare, while the production of four b quarks is very common. The goals of this project include understanding the kinematics behind signal and background events (*ie.* Collisions that produce four b quarks but do not come from an exotic Higgs decay), creating a Binary Decision Tree (BDT) to classify events, and analyzing the effectiveness of the BDT to separate the signal from background. This work aims to provide a well-established algorithm that can be run over the Run 2 data of the LHC to determine if these signal events exist.

Acknowledgements

I would like to take the time to thank all the people who made this project a possibility and a success.

To Professor Rafael for his time, effort, and support as well his guidance throughout this entire project

To Professor Trubko for her advice and support throughout this year and over the course of this project

To the ATLAS Collaboration and the UMass Amherst ATLAS group for their partnership

To the CERN NSF REU program at University of Michigan for connecting me with Rafael, because this project would not exist without him

To my friends and family for supporting me throughout my journey and project

Table of Contents

Table of Figures

Table of Tables

Chapter 1: Project Motivation and Background

The Standard Model of Elementary Particles (SM) is the culmination of over 50 years of theory and experimentation in the second half of the 20th century and is a model of all the elementary particles that have been observed in the laboratory (Figure 1). This model is the product of physicists' efforts to combine three[1] out of the four fundamental forces (the weak force, the strong force, electromagnetic force) into a single theory along with the elementary particles. From these efforts one unified theory existing as the Standard Model emerged [1]. Since then, there has been much experimentation and validation of the Model through the construction of high energy particle accelerators, most notably the Large Hadron Collider (LHC) at CERN (European Council for Nuclear Research) in Geneva, Switzerland, thus proving the model's accuracy. The one particle predicted by the Standard Model but not experimentally proven to exist at the time of the construction of the LHC was the Higgs Boson, a spin-less boson responsible for the Higgs Mechanism. Without the Higgs Mechanism, fundamental particles would be massless. In 2012, the LHC successfully detected a Higgs Boson. Since then, new theories of various types of Higgs particles have been developed that could explain several observations that still evade an explanation using the SM, such as the existence of dark matter in the universe. My project aims to create a method by which one of these new particle like the Higgs Boson can be searched for at the LHC.

[1] The force of gravity is excluded from this model as it is governed under the theory of general relativity which had emerged before the understanding and formulation of the Standard Model.

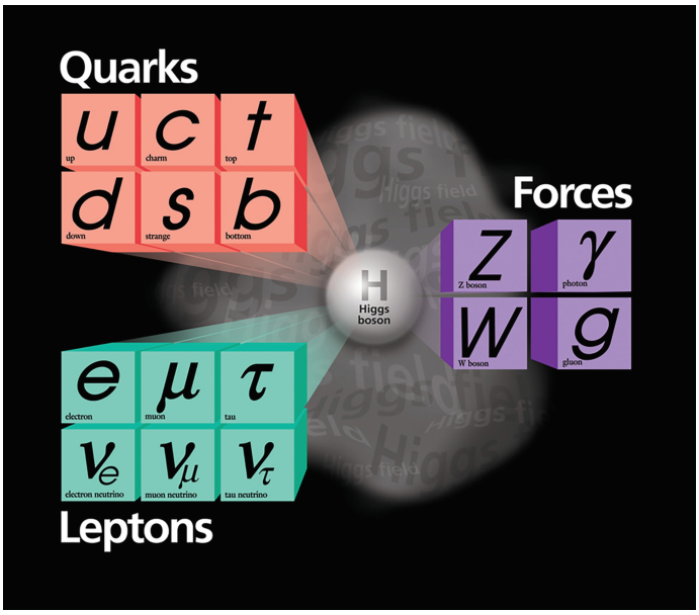

Figure 1: The Standard Model
This diagram describes the Standard Model in its totality, where quarks and leptons are devised into 6 different types of particles and the bosons are described as force particles, all of which come because of interacting with the Higgs Field and Higgs Boson.

1.1 The Higgs Boson and Its Significance

The Higgs Boson was theorized by multiple research groups around the same time searching for why certain particles have mass [2, 3, 4]. As the Standard Model was being formulated, one large problem that arose was finding an appropriate theory connecting the electromagnetic and the weak forces. The question was, "how does the photon remain massless, giving the electromagnetic force an infinite range, whilst the W and Z Bosons acquire a seemingly large mass, explaining the short-range of the weak nuclear force" [5]. Thus, people began looking for a solution to this problem to understand where and how particles obtain mass. The mechanism, proposed in those papers, would eventually become known as the Higgs Mechanism. Their research became central to bringing together the theories of the Standard Model [5]. The Higgs field is an invisible field that stretches across the entire universe and when interacted with gives fundamental particles their mass [6]. The Higgs Field creates a "spontaneous symmetry breaking of the local gauge symmetry, through the field's non-zero vacuum expectation value" which thus leads to the fundamental particles gaining their mass [5]. The physical particle associated with this field is the Higgs Boson, a spin zero, positive parity boson [5, 6]. However, without proof of the

existence of the Higgs Boson particle as described by the mechanism, it remained the one theory within the Standard Model to be experimentally proven. This was one of the biggest questions in particle physics that prompted the development of the LHC.

The first potential proof of this theory was presented in 2012, when both the ATLAS (A Toroidal LHC ApparatuS) and CMS (Compact Muon Solenoid) detectors at the LHC announced the discovery of a new particle, possibly the Higgs Boson [7, 8]. It was a huge triumph in the realm of particle physics and Standard Model research as well as the detectors themselves. Through further investigation of the data, it was uncovered that the discovered particle does follow predicted theory surrounding the Standard Model Higgs predictions. This included the results that the detected Higgs had a mass of around 125 GeV, had spin 0, and positive parity, giving validation to the Higgs Mechanism [9].

1.2 The Search for the a Particle

Every measurement of the Higgs boson so far agrees with the predictions of the Higgs Mechanism in the Standard Model. However, the measurements performed to date have limited precision and it is possible that deviations to the predicted behavior can be observed. It is possible that other particles similar to the Higgs Boson particle may exist, which could help shed light on the nature of dark matter. These potential particles would also be bosons with spin zero but would have different masses and a different role in the Higgs Mechanism. While there are many different theories being investigated, this paper focuses on the idea that there exists a lower mass particle similar to the Higgs boson that is pair-produced in exotic decays of the Higgs boson. This new light particle is referred to here as the "a" particle. It is studied by many research groups working with ATLAS, including my collaborators at the ATLAS group at University of Massachusetts – Amherst (UMass Amherst). The present method of searching for the "a" particle involves analyzing the current data that researchers have from the LHC's Run 2. However, this search is a challenge and first requires a discussion of the inner workings of the detector.

The ATLAS detector, shown in Figure 2, utilizes several magnets and consists of three major parts: an inner detector, an outer muon spectrometer, and a thick calorimeter [5, 10-12]. Each of

these elements work together to record data that is interpreted by algorithms that yield the specific information such as momentum and energy of each particle created in the collision. The inner detector consists of extensive granular pixel and microstrip silicon semiconductor detectors, and transition radiation detectors which are used to detect charged particles coming out of the collisions and to retrace their trajectories [5]. Surrounding the inner detector is a solenoid magnet and, downstream of the magnet, the calorimeters that measure the energy of almost all particle coming from the collision point [13, 14]. Finally, the muon spectrometer, located on the outer part of the detector, captures muons typically unseen by the inner detector and calorimeter, mainly muons. An additional toroidal magnet is present in the muon spectrometer. Neutrinos interact too weakly to be detected and are not registered by the ATLAS detector [15]. Since neutrinos carry energy and momentum, we can determine where and when they exist through an imbalance of linear momentum in the plane transverse to the beam pipe.

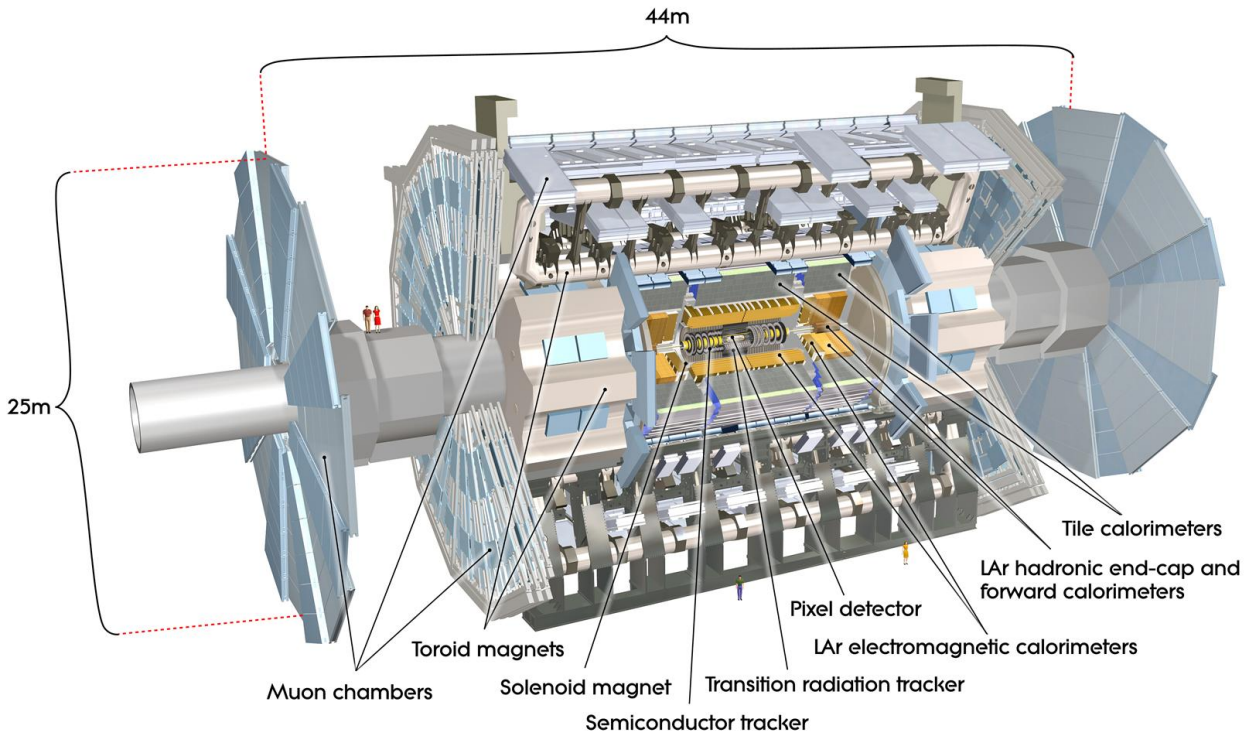

Figure 2: Cross Section of the ATLAS Detector
Shown here is a schematic of the ATLAS Detector with all the different parts labelled. While the types of equipment have remained the same since its construction, the detector has undergone multiple upgrades and will be upgraded shortly for the High-Luminosity Run

Particles collide at the center of the ATLAS detector and scatter in all directions. From there, the inner detector detects, through electrical signals, the particles that pass through. Using vertex reconstruction techniques, computers recreate the collision or decay center. While traveling through the Inner Detector, the solenoid magnets exert a magnetic force on them, bending the particles at various degrees based on the particle's momentum and charge. Finally, the particles interact with the calorimeter, which collects the energy of the particle. Using this data, the ATLAS detector calculates a particle's trajectory, collision point, momentum, charge, and energy. If the ATLAS detector can gain all of this information for a single particle, why can it not detect this potential a particle?

The lifetime of the Standard Model Higgs Boson is 10^{-23} s [5]. Because of this extremely small timeframe, no modern detectors can physically detect a Higgs Boson. They can only detect its decay products. In fact, this is true for many of the Standard Model particles, including the W boson, the Z boson, and the top quark. The theory currently being investigated by the ATLAS group at UMass Amherst suggests the existence of an a particle as a Higgs-like particle that decays from the known Higgs Boson ($H \rightarrow aa$). In some scenarios, the a boson would have a similarly short decay rate, making it impossible to be seen by any detector. In order for us to search for this a" particle, we must first determine the decay products that would be seen by the detector. While the known Higgs has numerous decay patterns, the most common decay pattern is the decay to b (bottom) quarks, which makes up 58% of all detected decays [16, 17]. The new a boson would have a similar preference for decays to b-quarks. Thus the signal decay pattern can be written as $H \rightarrow aa \rightarrow bb\bar{b}\bar{b}$. The search performed by the UMass ATLAS group requires the Higgs boson to be produced in association with a Z boson. With this pattern, the end products calculated from the detectors are two leptons l and four b quarks.

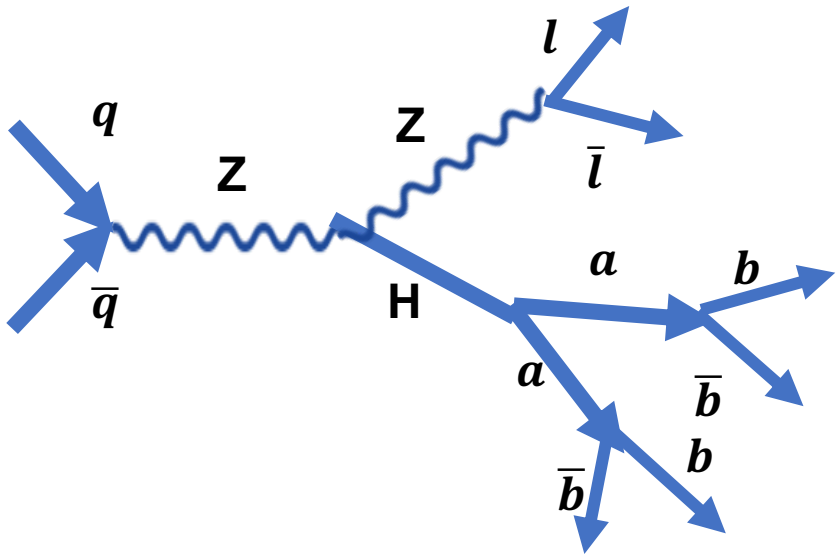

Figure 3: Decay Pattern of Signal Events
Shown here is the Feynman diagram of the signal process from collision to final decay products and can be written out as $qq \rightarrow Z \rightarrow ZH,\ Z \rightarrow l\bar{l},\ H \rightarrow aa \rightarrow bb\bar{b}\bar{b}$. For clarification the following variables are defined: q – proton, Z – Z Boson, H – 125GeV Higgs Boson, a – the new particle, l – lepton, $\bar{l}$– anti-lepton, b – bottom quark, $\bar{b}$– anti-bottom quark

These end decay products are not unique to this signal process, which is the main problem in searching for the a particle. There are many known processes that end with this set of particles, two of which are shown in Figure 4. Because the detector only detects the final products, it is challenging to distinguish this signal event from other events, which we will collectively call background events. For the purposes of this project, we looked at seven different background processes: production of single Z bosons, top-quark pair production (ttbar), production of two W and Z bosons (diboson), single top-quark events, and other rarer processes. All of these events end in the aforementioned six detected particles. The top-quark pair (ttbar) background also includes the presence of neutrinos which are seen in Figure 4 (right), however, as mentioned previously, the detector does not capture neutrinos, making it appear the same as the signal event.

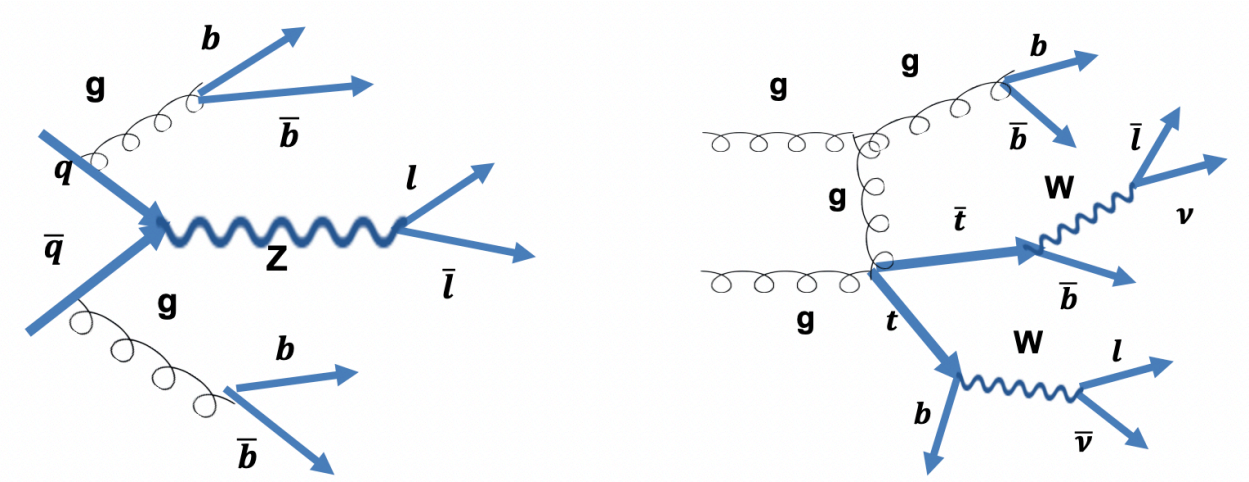

Figure 4: Example Background Decay Patterns
These Feynman diagrams represent the two most important types of backgrounds processes used in this project: to the left is the Z-jets background and to the right is the ttbar background. For clarification the following variables are defined: q – proton, Z – Z Boson, W – W Boson, g – gluon, l – lepton, $\bar{l}$ – anti-lepton, b – bottom quark, $\bar{b}$ – anti-bottom quark, t – top quark, $\bar{t}$ – anti-top quark, ν – neutrino, $\bar{\nu}$ – anti-neutrino

1.3 Project goals

To determine if the a particle does exist, we need to identify specific events consistent with the decay $H \rightarrow aa \rightarrow bb\bar{b}\bar{b}$. Since we need to narrow down the very large dataset of collected events to those with the potential to contain evidence of the a particle, we will utilize machine learning algorithms. Machine learning is a promising analysis technique because these types of algorithms can adapt to data overtime, making them more efficient than other programs. Machine learning for data analysis is further described in Chapter 3: Creating a BDT.

The goals of this project were to:

1. Understand the kinematics variables in the $H \rightarrow aa \rightarrow bb\bar{b}\bar{b}$ decay
2. Create a set of parameters to distinguish signal events from background events
3. Create a binary decision tree (BDT) to separate the signal and background events

Chapter 2: Distinguishing Signal and Background Events

The first challenge in the search for the a particle is to distinguish between background and the signal events using the information present in the data collected by ATLAS. As discussed in section 1.2 The Search for the $\boldsymbol{a}$ Particle, the difficulty to finding these potential events is that multiple processes produce the same particles. To accurately distinguish between signal and background events, we must uncover differences to separate these decay patterns. There are several variables that we can use to differentiate the number of events of interest.

2.1 Mass

The first, most obvious candidate for differentiating between events is the mass of the related decay particles. The mass can be reconstructed from the momentum and the energy of the particle using Equation 1. The ATLAS detector is able to collect the final momentum and energy of the end decay particles. Using this, we can determine the mass of the parent decay particles moving backwards up the decay pattern to the collision vertex. In terms of separating out given signal and background, the first variable that we define is that the mass of the Higgs boson equals approximately 125 GeV.

$$E^2 = (pc)^2 + (m_0c^2)^2$$

Equation 1: The Energy-Momentum Relation with energy E, momentum p, rest mass m_0, and speed of light c.

Additionally, since the a particle has never been observed, we do not know the value of its mass. With the decay pattern we are looking at, its mass must be less than or equal to half of the mass of the Higgs to maintain conservation of energy and momentum in a decaying process. In a general sense, this adds one more variable, the mass of the a particle, as a discriminating variable. This will be further discussed in Chapter 3: Creating a BDT.

2.2 Missing Transverse Momentum

The next distinguishing variable involves the conservation of momentum. As discussed in section 1.2 The Search for the $\boldsymbol{a}$ Particle, the background from top-quark pairs has neutrinos in the final state, small fundamental particles that are not detected by ATLAS. In ATLAS, their existence is accounted for through noting an imbalance of linear momentum in the transverse plane. Thus, the second variable which we can use is labeled MET and accounts for the presence of missing transverse momentum. If this missing transverse momentum exists, then the process is a background event.

2.3 Spin and Parity

The last avenue for telling the difference between signal and background is from the spin zero nature of the Higgs boson. While the end products seen by the ATLAS detector are the same, the spin of the final state is only a pure spin zero state when the Higgs boson is present. This subject has been studied heavily when searching for other types of events and is used in many different differentiating algorithms. This project uses the techniques of Bolognesi et al, which is described in brief below [18].

The direction of particles produced in a decay depends on the spin and parity of the unstable particle. Figure 5 shows a diagram of the production of a single unstable particle in a collision and the decay sequence of this unstable particle (X) into other bosons (represented by the directions q_1 and q_2). Each of these two bosons further decay into two particles (represented by the directions $q_{11}, q_{12}, q_{21},$ and q_{22}). The angles in the diagram fully characterize these directions. As shown in Ref. [18], the distributions of these angles in data depend on the spin and parity of the particle X. Therefore, each of these angles can be used to distinguish background from signal. The cosine $\cos\theta^*$ (Equation 2) is particularly sensitive to the spin of X and therefore can be used to discriminate signal events with a Higgs, which has spin 0, from background processes with a signal Z, which has spin 1.

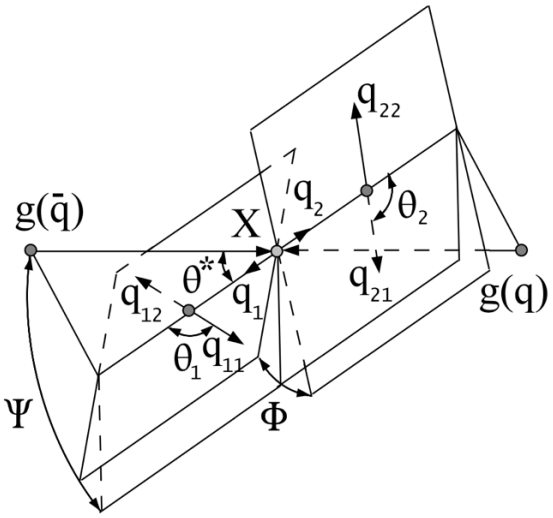

Figure 5: Diagram of a Decay Sequence

This diagram describes all angles present in a generic decay sequence $g(\bar{q})g(q) \to X \to V_1(q_1)V_2(q_2)$ *where* $V_1 \to f(q_{11})\bar{f}(q_{12})$, $V_2 \to f(q_{21})\bar{f}(q_{22})$. *Each of these variables can be calculated using the momentum components of the final decay outputs. Full calculations can be found in the appendix of Bolognesi et al. [*18*]*

$$\hat{q}_i = (\sin\theta^*\cos\Phi^*, \sin\theta^* \sin\Phi^*, \cos\theta^*)$$

Equation 2: Formula for $\cos\theta^*$

Shown here is the formula as written by Bolognesi et al. which describes the $\cos\theta^*$ *variable within the decay sequence shown in* Figure 5 [18]. *The angle is found using the momentum vector* $\hat{q}_i$ *where i is a stand in for the 2 momenta of the two parent particles*

2.4 Final Variables

Bringing together all the variables discussed in the previous sections, there are a total of seven discriminating variables. We also include an eighth variable, namely the event reconstruction quality, a measurement of how well the four b-quarks were identified in the detector. Therefore in total we have eight different variables with which we will use to separate these events: MET, Higgs mass (m_H), Higgs momentum (p_T^H), Z Boson production direction $\cos\theta_{CS}^{\ell\ell}$, a boson mass ($m_a^{(1)}$), a boson mass ($m_a^{(2)}$), Higgs $\cos\theta^*$, and Event Quality.

The distribution of each of these variables and their correlation are taken into account in the machine learning algorithm and will be discussed further in the next section.

Chapter 3: Creating a BDT

For this project, a binary decision tree was used as the method of machine learning and was carried out using the Python programming languages. Before going into the specifics of the BDT created as a part of the research, a brief discussion of how a BDT works is given.

3.1 The Binary Decision Tree

A binary decision tree is one of the simpler types of machine learning algorithms and uses a series of splittings to reduce a set of events. Taking into account the "significance" of different variables, it randomly selects an input variable and a threshold which are used to divide the data into two sets. It then repeats the process, splitting the data into smaller and smaller subsets that can have a significant larger fraction of signal events than the original dataset. Once this process is complete, the BDT can then create a file with the information of which splittings were most successful to select more signal than background. For this process to work, a BDT must be trained using simulated data where each event is labelled as signal or background. By running the BDT over simulated data, it goes through its splitting process and then evaluates at the end which groups contain mostly signal data. It is important to note that when creating a BDT to search for signal data, one can possibly overtrain or undertrain the algorithm. If too little data is used or the number of variables looked at is too high, there is the possibility that the program becomes overtrained and will miss valuable events when running over actual data. Similarly, if there is too much data and there are too few variables looked at, then too many events may pass as signal events. To avoid this, it is crucial that the BDT be tested using various total amounts of data as well as different numbers of variables to ensure that no signal event is missed while also narrowing down the data set.

The BDT for this project was assembled in Python with the functions in the TMVA ROOT library made available to Python via the PyROOT package as well as the XGBoost package. Python is a user-friendly programming language which is widely used for the purposes of machine learning. ROOT, an open-source language established by CERN for the purposes of high energy physics, is optimized for large data sets and has built-in machine learning functions. Both reasons

made the choice of programing languages simple. In addition, with the large amount of open-source tutorials on ROOT and its integration with Python, much of this project was able to base its code off of written tutorials [19]. These tutorials made the project good smoother and allow for faster basic structure creation.

3.2 The Coding of the BDT

As stated in the previous section, a BDT must go through a training process before working through experimental data. Thus, before the code was written, specific simulation data had to be chosen with which to train the BDT with. Dedicated simulation data was prepared with the decay of Higgs boson to a boson. The ATLAS collaboration provided standard simulation of all background processes. One single a boson mass hypothesis was chosen for the BDT in this work We choose to work with the hypothesis of the a boson mass equaling 30 GeV. The simulated events provided by the ATLAS collaboration were skimmed into smaller TTrees containing only the variables that we discussed in section 2.4 Final Variables (see Appendix A: Get m30 Events Code). After the skimming, a total of 1,566,900 events (Table 1) among signal and background were available for training. Using the pre-selected data, a training program and testing program (detailed in

Appendix B: Binary Decision Tree Creation Code) were written based on given tutorials in References [20] and [21].

Event Name	Number of Events
signal	75,100
single-top	2,127
Z-jets	570,022
ttbar	490,769
ttH	67,050
ttV	297,752
rare	12,331
diboson	51,749

Table 1: Table of Simulated Events

The above table lists all eight different types of events used in training the BDT program and the number of events present for each type of event. As can be calculated, the signal represented roughly 4.7% of all events used.

The algorithms written and shown in

Appendix B: Binary Decision Tree Creation Code, facilitate both the training and testing of the BDT. Using the python program NumPy and ROOT RDataFrames, the training program defines a load data function which combines the signal and all of the background data into two arrays. From there the function then splits these arrays into smaller ones separating for each variable that we defined previously. Lastly, using a variable defined as "weight", the load data function places probabilistic quantities to both signal and background to give significance to the signal events. Once complete, the load data function returns arrays of all the variables for signal and background. Then the training code invokes the XGBClassifier function from xgboost[2] which takes in a max depth, or how many variables the BDT goes through, and creates the BDT function. Finally, it fits the data given in the load data function and saves the model as a BDT. By running this program over all background and signal, a full model is created which can be used to test and eventually run over experimental data.

The purpose of the second script, the testing script, is to allow the user to test different sets of data to understand the BDT model and its optimal max depth. It calls the load data function from the training program and runs it over the established model. Finally, using the roc function from another python package, sklearn, a graph is created indicating the efficiency of the trained model. The graph plots the true positive rate (tpr) against the false positive rate (fpr) to understand the quality of the model and its predictions. For this project's roc graphs, tpr is labelled on the y-axis as "Signal Efficiency" and fpr is labelled on the x-axis as "Background Efficiency". The script also calculates the Area Under the Curve (AUC) which is a numerical indicator of the BDT's performance. An AUC equal to one represents perfect performance where an AUC of zero represents a chance performance. With both the training and testing scripts written, the next step was to create the BDT model.

3.3 The Preliminary Results

As part of the training process, a BDT model was created using only one type of background event and then another using all types of background events. For the first trained

[2] Documentation on the functions within xgboost are given in reference [22].

model, the training program was run using the data from the signal events and from the Z-jets background event (the largest of the background events) and a max depth of 3 variables. From this, the testing script created the graph shown in Figure 6, which had an AUC of 0.90. While still being the largest background, this model only represented one of the seven possible background events making it not practical for use on real data. Despite that, this result proved a large step in the overall project progress and was used as a baseline for how it could be made better.

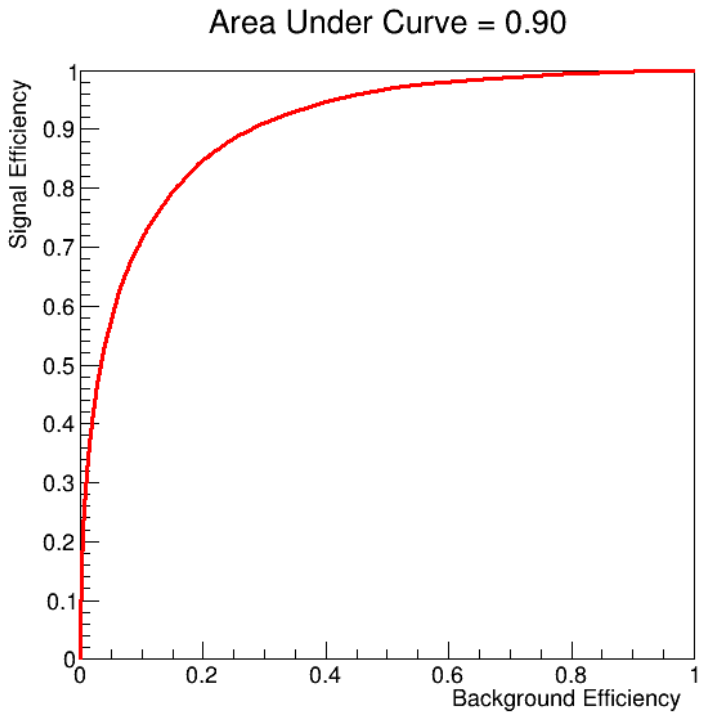

Figure 6: BDT Efficiency of Signal vs. Z-jets Background
Shown here is the ROC curve for the BDT model trained with only signal and Z-jets background. As shown in the title, the AUC of this model was 0.90, a fairly good number considering the smaller amount of data and max depth. The y-axis represents the signal efficiency, and the x-axis represents the background efficiency, with both being rates.

Given the baseline of the previous model, a new model was created using all the potential backgrounds described in Section 1.2 The Search for the $\boldsymbol{a}$ Particle and in Table 1 and again a max depth of three. Once created, the model was tested over all pre-selected background and signal events, yielding the graph in Figure 7. The AUC is equal to 0.99, which is much closer to one and thus a closer approximation of correctly chosen signal events. It is worth mentioning that this graph

does encounter more noise, seen by the rougher curvature of the roc. This is most likely caused by the addition of multiple types of background events that were not present in Figure 6. Figure 7 represents a vastly more accurate model than the previous model as it incorporates all possible backgrounds known and signal, making it able to discern more background out of the final events of interest.

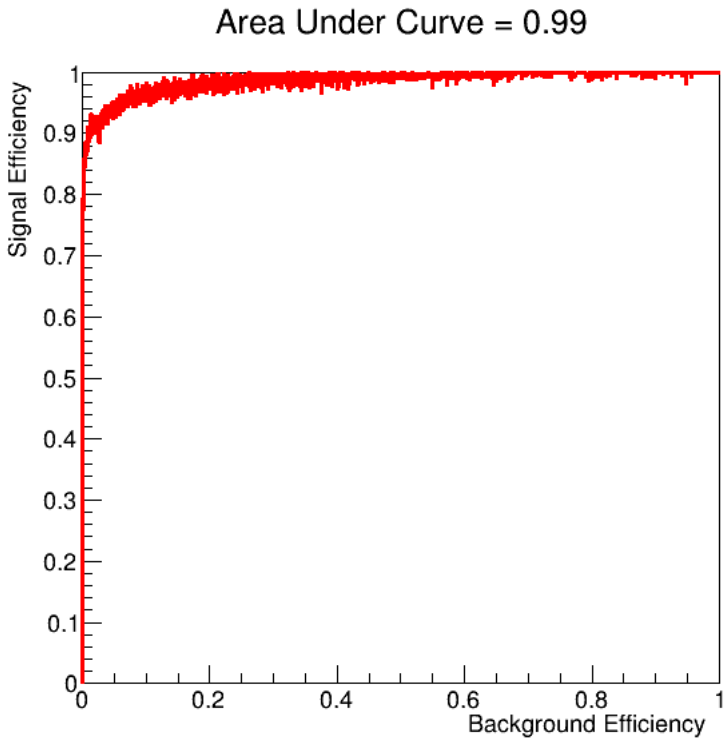

Figure 7: BDT Efficiency Signal vs. All Background
Shown here is the ROC curve for the BDT model trained with signal and all possible background events. As shown in the title, the AUC of this model was 0.99, an almost perfect signal distinction. The y-axis represents the signal efficiency, and the x-axis represents the background efficiency, with both being rates. As can be seen, this graph does not have as smooth of a curve as the previous one, due most likely to the addition of multiple types of background events.

3.4 Testing the BDT

It is important to recognize that while an AUC equal to 0.99 indicates an almost perfect distinction, it must be tested to ensure that this is not an indicator of overtraining. In the end, this model was created based on simulation data which carries with it uncertainty. This uncertainty comes from the uncertainty of the statistics behind which the simulation data is curated. Because

of this, it is important to confirm that this model is not specified to the given data set and produces similar results for different sized data sets. With this in mind, the last step of this project was to test for overtraining by running the trained model over a subset of the simulation data and creating comparison histograms. These comparison histograms combine the histogram of the signal and background from the trained model with a scatter plot of the test including error bars. If the plotted test aligns with the histogram, then a BDT is confirmed to work on other datasets. Once verified to work for different amounts of data, a BDT can then be further refined to bring the AUC closer to one.

To test the BDT created in this project, the testing code was modified to include functions that randomly selected a designated percentage of the data and produce the histograms mentioned in the previous paragraph as well as the roc curves (Appendix C: Testing BDT Code). The model was then tested using first 30% of the simulated data and then 70% (Test size equal to 0.3 and 0.7). From these parameters, graphs of the roc curves as well as the test histograms and comparison histograms were created and are detailed in (Figure 8). Focusing on the comparison histograms, the scatter plot successfully lines up in both the 70% and the 30% cases with the trained model. With this result, we are confident that the AUC of 0.99 is an almost perfectly trained model that is not overtrained or specific to one set of data.

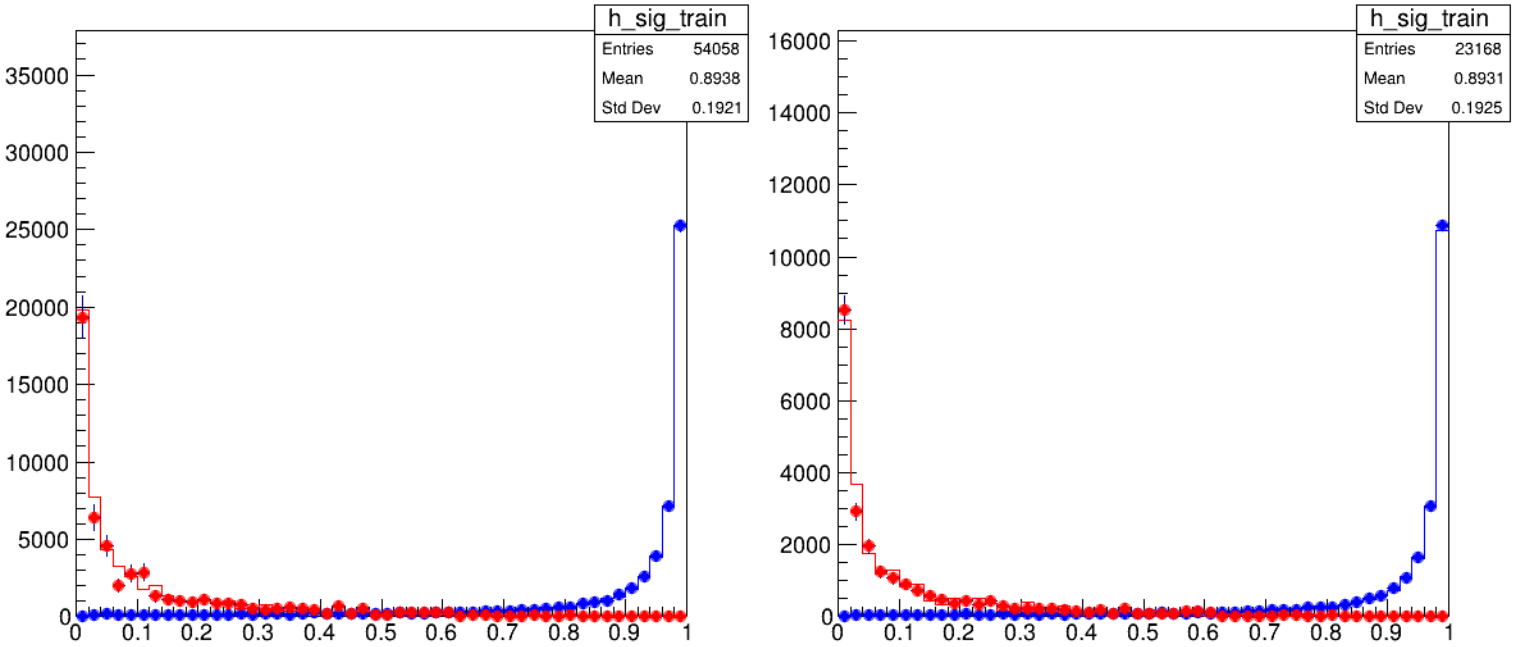

Figure 8: BDT Testing Comparison Histograms

Shown here are the comparison histograms created in two testing rounds of the trained BDT (left – 30% of the simulation data, right – 70% of the simulation data). These graphs were created using the code detailed in Appendix C: Testing BDT Code. Here, red marks the background events while blue marks the signal events. The histograms created detail the actual signal and background events where the dots represent the tested data. The markers for both test align with the histograms, validating proper training of the BDT.

Chapter 4: Conclusion and Next Steps

The goals of this research project were to:

1. Understand the kinematics behind Higgs Boson decay
2. Create a set of parameters to distinguish signal events from background events
3. Create a binary decision tree (BDT) to separate the signal and background events

Through efforts made in both literature research and programming, a full understanding of the defining variables distinguishing was established and a Binary Decision Tree was produced. This BDT is the basis code which successfully can distinguish signal and background events in simulated data with high accuracy and precision. As with any research, there is more work to be done. While this BDT is functional, it still requires more testing and refining which is left to future researchers. This includes possible changes to the max depth which can enhance the accuracy further. In addition to this, the author suggests that research into other machine learning algorithms, like a neural network, be done and compared with the established BDT. While the BDT is an appreciable machine learning method, it does carry with it certain limitations from its cut process. Other algorithms, such as neural networks, could provide an even stronger model for distinguishing background and signal. By completing a comparison, the most effective method can be used to separate LHC Run 2 data and further the search for the a boson.

References

[1] Cottingham, W. N. (1998). *An introduction to the standard model of particle physics*. Cambridge University Press.

[2] Guralnik, G. S., Hagen, C. R., & Kibble, T. W. B. (1964). Global Conservation Laws and Massless Particles. *Physical Review Letters*, *13*(20), 585–587. https://doi.org/10.1103/PhysRevLett.13.585

[3] Higgs, P. W. (1964). Broken Symmetries and the Masses of Gauge Bosons. *Physical Review Letters*, *13*(16), 508–509. https://doi.org/10.1103/PhysRevLett.13.508

[4] Englert, F., & Brout, R. (1964). Broken Symmetry and the Mass of Gauge Vector Mesons. *Physical Review Letters*, *13*(9), 321–323. https://doi.org/10.1103/PhysRevLett.13.321

[5] Jenni, P., & Virdee, T. S. (2020). The Discovery of the Higgs Boson at the LHC. In H. Schopper (Ed.), *Particle Physics Reference Library: Volume 1: Theory and Experiments* (pp. 263–309). Springer International Publishing. https://doi.org/10.1007/978-3-030-38207-0_6

[6] *The Higgs boson*. (n.d.). CERN. Retrieved March 30, 2022, from https://home.cern/science/physics/higgs-boson

[7] Aad, G., Abajyan, T., Abbott, B., Abdallah, J., Abdel Khalek, S., Abdelalim, A. A., Abdinov, O., Aben, R., Abi, B., Abolins, M., AbouZeid, O. S., Abramowicz, H., Abreu, H., Acharya, B. S., Adamczyk, L., Adams, D. L., Addy, T. N., Adelman, J., Adomeit, S., … Zwalinski, L. (2012). 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. https://doi.org/10.1016/j.physletb.2012.08.020

[8] *Observation of a new boson at a mass of 125 GeV with the CMS experiment at the LHC | Elsevier Enhanced Reader*. (n.d.). https://doi.org/10.1016/j.physletb.2012.08.021

[9] *New results indicate that particle discovered at CERN is a Higgs boson*. (n.d.). CERN. Retrieved May 6, 2022, from https://home.web.cern.ch/news/press-release/cern/new-results-indicate-particle-discovered-cern-higgs-boson

[10] *CERN experiments observe particle consistent with long-sought Higgs boson*. (n.d.). CERN. Retrieved October 12, 2021, from https://home.cern/news/press-release/cern/cern-experiments-observe-particle-consistent-long-sought-higgs-boson

[11] George, S. (1997, April 30). *ATLAS: Inner detector technical design report; 7, level-2 trigger* (CERN-OPEN-99-155). CERN Document Server. https://cds.cern.ch/record/398955

[12] ATLAS muon spectrometer: Technical Design Report. (1997). In *CERN Document Server* (CERN-LHCC-97-022). CERN. https://cds.cern.ch/record/331068

[13] *Calorimeter*. (n.d.). ATLAS Experiment at CERN. Retrieved April 27, 2022, from https://atlas.cern/Discover/Detector/Calorimeter

[14] *Magnet System*. (n.d.). ATLAS Experiment at CERN. Retrieved April 27, 2022, from https://atlas.cern/Discover/Detector/Magnet-System

[15] Reines, F., & COWANjun., C. L. (1956). The Neutrino. *Nature*, *178*(4531), 446–449. https://doi.org/10.1038/178446a0

[16] Djouadi, A., Kalinowski, J., & Spira, M. (1998). HDECAY: A program for Higgs boson decays in the Standard Model and its supersymmetric extension. *Computer Physics Communications*, *108*(1), 56–74. https://doi.org/10.1016/S0010-4655(97)00123-9

[17] Aaboud, M., Aad, G., Abbott, B., Abdinov, O., Abeloos, B., Abhayasinghe, D. K., Abidi, S. H., AbouZeid, O. S., Abraham, N. L., Abramowicz, H., Abreu, H., Abulaiti, Y., Acharya, B. S., Adachi, S., Adam, L., Adamczyk, L., Adelman, J., Adersberger, M., Adiguzel, A., … Zwalinski, L. (2018). Observation of $H \rightarrow b\bar{b}$ decays and VH production with the ATLAS detector. *Physics Letters B*, *786*, 59–86. https://doi.org/10.1016/j.physletb.2018.09.013

[18] Bolognesi, S., Gao, Y., Gritsan, A. V., Melnikov, K., Schulze, M., Tran, N. V., & Whitbeck, A. (2012). On the spin and parity of a single-produced resonance at the LHC. *Physical Review D*, *86*(9), 095031. https://doi.org/10.1103/PhysRevD.86.095031

[19] *ROOT: TMVA tutorials*. (n.d.). Retrieved April 29, 2022, from https://root.cern/doc/master/group__tutorial__tmva.html

[20] *ROOT: tutorials/tmva/tmva101_Training.py Source File*. (n.d.). Retrieved April 29, 2022, from https://root.cern/doc/master/tmva101__Training_8py_source.html

[21] *ROOT: tutorials/tmva/tmva102_Testing.py File Reference*. (n.d.). Retrieved April 29, 2022, from https://root.cern/doc/master/tmva102__Testing_8py.html

[22] *XGBoost*. (n.d.). Retrieved May 1, 2022, from https://xgboost.ai/

Appendices

Appendix A: Get m30 Events Code

```
import ROOT
import numpy as np
import os
import sys

def get_recoCategory(pred):

    if pred >= 1 and pred <= 7: return 1 # 2B
    if pred >= 21 and pred <= 26: return 2 # 1B2b
    if pred >= 31 and pred <= 32: return 3 # 1B1b1v
    if pred >= 41 and pred <= 55: return 4 # 4b
    if pred >= 61 and pred <= 63: return 5 # 3b1v

    return -1

def calculate_cs(leps):

    l1 = ROOT.TLorentzVector(leps[0].Px(), leps[0].Py(), leps[0].Pz(), leps[0].E())
    l2 = ROOT.TLorentzVector(leps[1].Px(), leps[1].Py(), leps[1].Pz(), leps[1].E())
    z = l1+l2

    l1plus = 1.0 / ROOT.TMath.Sqrt(2.0) * (l1.E() + l1.Z())
    l1minus = 1.0 / ROOT.TMath.Sqrt(2.0) * (l1.E() - l1.Z())
    l2plus = 1.0 / ROOT.TMath.Sqrt(2.0) * (l2.E() + l2.Z())
    l2minus = 1.0 / ROOT.TMath.Sqrt(2.0) * (l2.E() - l2.Z())

    costheta = 2.0 * (l1plus * l2minus - l1minus * l2plus) / (z.M() * ROOT.TMath.Sqrt(z.M()*z.M() + z.Pt()*z.Pt()))
    return costheta


def get_weight(event):
    weight =  (event.xsec*event.lumi/event.total_weight_sum)*event.weight_mc*event.weight_pileup*event.weight_jvt*event.weight_leptonSF*event.weight_event_ftag_SF
    return weight

def get_Bjet(event, thr = 1):

    count = 0
    ret = ROOT.TLorentzVector(0,0,0,0)
    for i in range(3):
        jet_pt = getattr(event,'jet_AntiKt8_{}_pt'.format(i+1))
        if jet_pt < 0: break

        if thr == 0 and ord(getattr(event,'jet_AntiKt8_{}_isDexter_tagged_50'.format(i+1))) == 0:
            count = count + 1
        if thr == 1 and ord(getattr(event,'jet_AntiKt8_{}_isDexter_tagged_50'.format(i+1))) == 1:
            count = count + 1
    return count

def get_bjet(event, thr = 1):

    count = 0
    ret = ROOT.TLorentzVector(0,0,0,0)
    for i in range(9):
        jet_pt = getattr(event,'jet_{}_pt'.format(i+1))
        if jet_pt < 0: break
        if thr == 0 and (getattr(event,'jet_{}_tagWeightBin_DL1r_Continuous'.format(i+1)) == 3 or getattr(event,'jet_{}_tagWeightBin_DL1r_Continuous'.format(i+1)) == 2):
            count = count + 1
        if thr == 1 and (getattr(event,'jet_{}_tagWeightBin_DL1r_Continuous'.format(i+1)) == 5 or getattr(event,'jet_{}_tagWeightBin_DL1r_Continuous'.format(i+1)) == 4):
            count = count + 1
    return count


sourceDirectory = '/eos/atlas/atlascerngroupdisk/phys-hdbs/hlrs/SOLARB/Ntuples/Level2/21/merged_SOLARBv5p4/'
mass1 = 30
mass2 = 30
procname = sys.argv[1]
signal = ''
if mass1 == mass2: signal = 'signal_Haa'
else: signal = 'signal_Haa'
backgrounds = ['diboson', 'rare', 'single_top', 'ttH', 'ttV', 'ttbar', 'zjets']
```

```
signalFile = ROOT.TFile('signal_region_{}.root'.format(procname), 'recreate')
signalTree = ROOT.TTree('signalTree', 'signalTree')
controlFile = ROOT.TFile('control_region_{}.root'.format(procname), 'recreate')
controlTree = ROOT.TTree('controlTree', 'controlTree')

lep_category = np.empty((1), dtype='int32')
controlTree.Branch('lep_category', lep_category, 'lep_category/I')
ttbb_category = np.empty((1), dtype='int32')
controlTree.Branch('ttbb_category', ttbb_category, 'ttbb_category/I')
vbb_category = np.empty((1), dtype='int32')
controlTree.Branch('vbb_category', vbb_category, 'vbb_category/I')
hf_category = np.empty((1), dtype='int32')
controlTree.Branch('hf_category', hf_category, 'hf_category/I')
nb_tight = np.empty((1), dtype='int32')
controlTree.Branch('nb_tight', nb_tight, 'nb_tight/I')
nb_loose = np.empty((1), dtype='int32')
controlTree.Branch('nb_loose', nb_loose, 'nb_loose/I')
nB_tight = np.empty((1), dtype='int32')
controlTree.Branch('nB_tight', nB_tight, 'nB_tight/I')
nB_loose = np.empty((1), dtype='int32')
controlTree.Branch('nB_loose', nB_loose, 'nB_loose/I')

weight = np.empty((1), dtype='float32')
controlTree.Branch('weight', weight, 'weight/F')
signalTree.Branch('weight', weight, 'weight/F')
sample = np.empty((1), dtype='int32')
controlTree.Branch('sample', sample, 'sample/I')
signalTree.Branch('sample', sample, 'sample/I')
reco_category = np.empty((1), dtype='int32')
signalTree.Branch('reco_category', reco_category, 'reco_category/I')
reco_score = np.empty((1), dtype='float32')
signalTree.Branch('reco_score', reco_score, 'reco_score/F')
higgs_a1m_red = np.empty((1), dtype='float32')
signalTree.Branch('higgs_a1m_red', higgs_a1m_red, 'higgs_a1m_red/F')
higgs_a2m_red = np.empty((1), dtype='float32')
signalTree.Branch('higgs_a2m_red', higgs_a2m_red, 'higgs_a2m_red/F')
higgs_costhetastar = np.empty((1), dtype='float32')
signalTree.Branch('higgs_costhetastar', higgs_costhetastar, 'higgs_costhetastar/F')
z_costhetacs = np.empty((1), dtype='float32')
signalTree.Branch('z_costhetacs', z_costhetacs, 'z_costhetacs/F')
ttbar_met = np.empty((1), dtype='float32')
signalTree.Branch('ttbar_met', ttbar_met, 'ttbar_met/F')
zh_hmass_red = np.empty((1), dtype='float32')
signalTree.Branch('zh_hmass_red', zh_hmass_red, 'zh_hmass_red/F')
zh_hpt = np.empty((1), dtype='float32')
signalTree.Branch('zh_hpt', zh_hpt, 'zh_hpt/F')


for ibkgd, bkgd in enumerate([signal] + backgrounds):
    print(bkgd)
    if not bkgd in procname: continue
    inputTree = ROOT.TChain('nominal_Loose')
    if 'signal' in bkgd:
        if mass1 == mass2:
            inputTree.Add('{}/{}/2ji3bobji/*m{}*.root'.format(sourceDirectory, bkgd, mass1))
            print '{}/{}/2ji3bobji/*m{}*.root'.format(sourceDirectory, bkgd, mass1)
        else:
            inputTree.Add('{}/{}/2ji3bobji/*m1_{}_m2_{}*.root'.format(sourceDirectory, bkgd, mass2, mass1))
    else:
        inputTree.Add('{}/{}/2ji3bobji/*.root'.format(sourceDirectory, bkgd))
        print '{}/{}/2ji3bobji/*.root'.format(sourceDirectory, bkgd)
    nevents = inputTree.GetEntries()
    for ievent, event in enumerate(inputTree):
        if ievent % 10000 == 0: print(ievent, nevents)
        if ievent == nevents: break

        passMuTrigger = False
        passElTrigger = False
        passEMuTrigger = False
        if ord(event.mumu_2015)==1 or ord(event.mumu_2016)==1 or ord(event.mumu_2017)==1 or ord(event.mumu_2018)==1 :
            passMuTrigger = True
        if ord(event.ee_2015)==1 or ord(event.ee_2016)==1 or ord(event.ee_2017)==1 or ord(event.ee_2018)==1:
            passElTrigger = True
        if ord(event.emu_2015)==1 or ord(event.emu_2016)==1 or ord(event.emu_2017)==1 or ord(event.emu_2018)==1:
            passEMuTrigger = 2
        if (not passMuTrigger) and (not passElTrigger) and (not passEMuTrigger): continue

        lep1 = ROOT.TLorentzVector(0,0,0,0)
        lep2 = ROOT.TLorentzVector(0,0,0,0)
        lep1.SetPtEtaPhiM(event.lep_1_pt, event.lep_1_eta, event.lep_1_phi, event.lep_1_m)
        lep2.SetPtEtaPhiM(event.lep_2_pt, event.lep_2_eta, event.lep_2_phi, event.lep_2_m)
        if min([lep1.Pt(),lep2.Pt()]) < 1000: continue
        leps = None
        if event.lep_1_charge < 0:
            leps = [lep1, lep2]
        else:
            leps = [lep2, lep1]

        z = lep1+lep2
        if (z.M() > 111000. or z.M() < 71000.) and (passElTrigger or passMuTrigger): continue

        reco_score[0] = getattr(event, 'highest_hypothesis_score_m1_30_m2_30'.format(mass2,mass1))
        predicted_hypothesis = getattr(event, 'predicted_hypothesis_m1_30_m2_30'.format(mass2,mass1))
        reco_category[0] = get_recoCategory(predicted_hypothesis)
        weight[0] = get_weight(event)
        sample[0] = ibkgd # leave 0 for signal

        isSignalRegion = ((reco_score[0] > 0.05 and passMuTrigger) or (reco_score[0] > 0.05 and passElTrigger)) and getattr(event, 'higgs_costhetastar_m1_30_m2_30'. format(mass2
,mass1)) > -900
```

```
        if not isSignalRegion:
            ttbb_category[0] = getattr(event, 'ttbb_categories')
            vbb_category[0] = getattr(event, 'vbb_categories')
            hf_category[0] = getattr(event, 'HF_Classification')
            lep_category[0] = 0
            if passEMuTrigger:
                lep_category[0] = 1
            nb_tight[0] = get_bjet(event, 1)
            nb_loose[0] = get_bjet(event, 0)
            nB_tight[0] = get_Bjet(event, 1)
            nB_loose[0] = get_Bjet(event, 0)
            controlTree.Fill()
        else:
            higgs_costhetastar[0] = getattr(event, 'higgs_costhetastar_m1_30_m2_30'. format(mass2,mass1))
            higgs_a1m_red[0] = getattr(event, 'higgs_a1m_red_m1_30_m2_30'. format(mass2,mass1))
            higgs_a2m_red[0] = getattr(event, 'higgs_a2m_red_m1_30_m2_30'. format(mass2,mass1))
            zh_hmass_red[0] = getattr(event, 'higgs_m_red_m1_30_m2_30'. format(mass2,mass1))
            zh_hpt[0] = getattr(event, 'higgs_pt_m1_30_m2_30'. format(mass2,mass1))
            ttbar_met[0] = getattr(event, 'met_met')
            z_costhetacs[0] = calculate_cs(leps)
            signalTree.Fill()

signalFile.cd()
signalTree.Write()
signalFile.Close()
controlFile.cd()
controlTree.Write()
controlFile.Close()
```

Appendix B: Binary Decision Tree Creation Code

Training Code

```
import ROOT
import numpy as np
import pickle

variables = [ 'ttbar_met', 'zh_hmass_red', 'zh_hpt', 'z_costhetacs', 'higgs_a1m_red', 'higgs_a2m_red', 'higgs_costhetastar', 'reco_score']

def Merge(dict1, dict2, dict3, dict4, dict5, dict6, dict7):
    res = {**dict1, **dict2, **dict3, **dict4, **dict5, **dict6, **dict7}
    return res

data_bkg1 = ROOT.RDataFrame("signalTree", "signal_region_zjets.root").AsNumpy()
data_bkg2 = ROOT.RDataFrame("signalTree", "signal_region_ttbar.root").AsNumpy()
data_bkg3 = ROOT.RDataFrame("signalTree", "signal_region_ttH.root").AsNumpy()
data_bkg4 = ROOT.RDataFrame("signalTree", "signal_region_ttV.root").AsNumpy()
data_bkg5 = ROOT.RDataFrame("signalTree", "signal_region_diboson.root").AsNumpy()
data_bkg6 = ROOT.RDataFrame("signalTree", "signal_region_rare.root").AsNumpy()
data_bkg7 = ROOT.RDataFrame("signalTree", "signal_region_single_top.root").AsNumpy()

Full_Bkg_Array = Merge(data_bkg1, data_bkg2,  data_bkg3, data_bkg4, data_bkg5, data_bkg6, data_bkg7)

def load_data(signal_filename, background_Array):
    # Read data from ROOT files
    data_sig = ROOT.RDataFrame("signalTree", signal_filename).AsNumpy()
    data_bkg = background_Array

    # Convert inputs to format readable by machine learning tools
    x_sig = np.vstack([data_sig[var] for var in variables]).T
    x_bkg = np.vstack([data_bkg[var] for var in variables]).T
    w_sig = np.nan_to_num(np.array(data_sig['weight']))
    w_bkg = np.nan_to_num(np.array(data_bkg['weight']))
    x = np.vstack([x_sig, x_bkg])
    wo = np.hstack([w_sig, w_bkg])
    z = np.hstack([w_sig/np.sum(w_sig), w_bkg/np.sum(w_bkg)])

    # Create labels
    num_sig = x_sig.shape[0]
    num_bkg = x_bkg.shape[0]
    y_true = np.hstack([np.ones(num_sig), np.zeros(num_bkg)])

    # Compute weights balancing both classes
    num_all = num_sig + num_bkg
    w = z * num_all

    return x, y_true, w, wo

if __name__ == "__main__":
    # Load data
    x, y_true, w, wo = load_data("signal_region_signal_Haa.root", Full_Bkg_Array)

    # Fit xgboost model
    from xgboost import XGBClassifier
    bdt = XGBClassifier(max_depth=3, n_estimators=500)
    bdt.fit(x, y_true, w)

   # Save model in TMVA format
   ROOT.TMVA.Experimental.SaveXGBoost(bdt, "myBDT", "tmvaAll_3_500.root")
```

Testing Code

```
import ROOT
import pickle

from BDT_Training import load_data
from BDT_Training import variables



# Load data
x, y_true, w = load_data("signal_region_signal_Haa.root", "signal_region_zjets.root")

# Load trained model
bdt = ROOT.TMVA.Experimental.RBDT[""]("myBDT", "tmva102.root")

# Make prediction
y_pred = bdt.Compute(x)

# Compute ROC using sklearn
from sklearn.metrics import roc_curve, auc
fpr, tpr, _ = roc_curve(y_true, y_pred, sample_weight=w)
score = auc(fpr, tpr, reorder=True)

# Plot ROC
c = ROOT.TCanvas("roc", "", 600, 600)
g = ROOT.TGraph(len(fpr), fpr, tpr)
g.SetTitle("Area Under Curve = {:.2f}".format(score))
g.SetLineWidth(3)
g.SetLineColor(ROOT.kRed)
g.Draw("AC")
g.GetXaxis().SetRangeUser(0, 1)
g.GetYaxis().SetRangeUser(0, 1)
g.GetXaxis().SetTitle("Background Efficiency")
g.GetYaxis().SetTitle("Signal Efficiency")
c.Draw()
c.SaveAs("BDT_m30_signal_zjets_Trial.png")
```

Appendix C: Testing BDT Code

```
import ROOT
import pickle
import numpy as np
from sklearn.model_selection import train_test_split

from BDT_Training import load_data
from BDT_Training import variables
from BDT_Training import Merge

data_bkg1 = ROOT.RDataFrame("signalTree", "signal_region_zjets.root").AsNumpy()
data_bkg2 = ROOT.RDataFrame("signalTree", "signal_region_ttbar.root").AsNumpy()
data_bkg3 = ROOT.RDataFrame("signalTree", "signal_region_ttH.root").AsNumpy()
data_bkg4 = ROOT.RDataFrame("signalTree", "signal_region_ttV.root").AsNumpy()
data_bkg5 = ROOT.RDataFrame("signalTree", "signal_region_diboson.root").AsNumpy()
data_bkg6 = ROOT.RDataFrame("signalTree", "signal_region_rare.root").AsNumpy()
data_bkg7 = ROOT.RDataFrame("signalTree", "signal_region_single_top.root").AsNumpy()

Test_Bkg_Array = Merge(data_bkg1, data_bkg2,  data_bkg3, data_bkg4, data_bkg5, data_bkg6, data_bkg7)

seed = 8675389
test_size = 0.7
ntestbins = 50

# Load data
x, y_true, w, wo = load_data("signal_region_signal_Haa.root", Test_Bkg_Array)

x_train, x_test, y_train, y_test, w_train, w_test = train_test_split(x, y_true, w, test_size=test_size, random_state=seed)

# Load trained model
bdt_pred = ROOT.TMVA.Experimental.RBDT[""]("myBDT", "tmvaAll_3_500.root")

# Make prediction
y_pred = bdt_pred.Compute(x)
y_pred_test = bdt_pred.Compute(x_test)
y_pred_train = bdt_pred.Compute(x_train)

h_sig = ROOT.TH1D('h_sig', '', ntestbins, 0., 1.)
h_bkg = ROOT.TH1D('h_bkg', '', ntestbins, 0., 1.)
h_sig_train = ROOT.TH1D('h_sig_train', '', ntestbins, 0., 1.)
h_bkg_train = ROOT.TH1D('h_bkg_train', '', ntestbins, 0., 1.)
h_sig_test = ROOT.TH1D('h_sig_test', '', ntestbins, 0., 1.)
h_bkg_test = ROOT.TH1D('h_bkg_test', '', ntestbins, 0., 1.)
for wgt, yt, yp in zip(wo, y_true, y_pred):
     h_sig.Fill(yp, wgt*yt)
     h_bkg.Fill(yp, wgt*(1-yt))
for wgt, yt, yp in zip(w_test, y_test, y_pred_test):
     h_sig_test.Fill(yp, wgt*yt*(1-test_size)/test_size)
     h_bkg_test.Fill(yp, wgt*(1-yt)*(1-test_size)/test_size)
for wgt, yt, yp in zip(w_train, y_train, y_pred_train):
     h_sig_train.Fill(yp, wgt*yt)
     h_bkg_train.Fill(yp, wgt*(1-yt))

sig_eff = []
inv_bkg_eff = []
for i in range(ntestbins-1):
     sig_eff.append(1-h_sig.Integral(0,i+1)/h_sig.Integral(0,ntestbins+1))
     inv_bkg_eff.append(1-h_bkg.Integral(0,i+1)/h_bkg.Integral(0,ntestbins+1))
tpr = np.array(sig_eff)
fpr = np.array(inv_bkg_eff)

# Plot ROC
ln = ROOT.TLine()

c = ROOT.TCanvas('roc', '', 600, 600)
g = ROOT.TGraph(len(fpr), fpr, tpr)

g.SetLineWidth(3)
g.SetLineColor(ROOT.kRed)
g.Draw('AC')
g.GetXaxis().SetRangeUser(0, 1)
g.GetYaxis().SetRangeUser(0, 1)
g.GetXaxis().SetTitle('Background efficiency')
g.GetYaxis().SetTitle('Signal efficiency')
c.SaveAs('roc_70.png')

c2 = ROOT.TCanvas('histogram', '', 600, 600)
c2.SetLogy()
h_sig.SetLineColor(ROOT.kBlue)
h_bkg.SetLineColor(ROOT.kRed)
maxval = 1.5*max([h_sig.GetMaximum(), h_bkg.GetMaximum()])
h_sig.SetMaximum(maxval)
h_sig.SetMinimum(0)
ln.DrawLine(0.9, 0, 0.9, maxval)
ln.DrawLine(0.8, 0, 0.8, maxval)
h_sig.SetMinimum(0.1)
h_sig.Draw('hist')
h_bkg.Draw('hist,same')
c2.SaveAs('hist_70.png')
```

```
c3 = ROOT.TCanvas('comp', '', 600, 600)
h_sig_train.SetLineColor(ROOT.kBlue)
h_bkg_train.SetLineColor(ROOT.kRed)
h_sig_test.SetMarkerColor(ROOT.kBlue)
h_bkg_test.SetMarkerColor(ROOT.kRed)
h_sig_test.SetMarkerStyle(20)
h_bkg_test.SetMarkerStyle(20)
h_sig_train.Draw('hist')
h_sig_train.SetMaximum(1.5*max([h_sig_train.GetMaximum(), h_bkg_train.GetMaximum(), h_sig_test.GetMaximum(), h_bkg_test.GetMaximum()]))
h_bkg_train.Draw('hist,same')
h_sig_test.Draw('e0,same')
h_bkg_test.Draw('e0,same')
c3.SaveAs('comp_70.png')
```