Study of the Associated Production of $t\bar{t}$ and Higgs with the ATLAS Detector

Martina Laura Ojeda

260579578

Final Report

Department of Physics

McGill University

Montreal, Quebec

August 21st, 2015

Completed under the supervision of:

Dr. Tamara Vázquez Schröder

Dr. François Corriveau

Acknowledgments

I would like to first thank my supervisor, Dr. Tamara Vázquez Schröder who came up with the project, introduced me to many physics concepts and analysis methods, and helped me through every step of the way. I would also like to thank Dr. François Corriveau and the Institute of Particle Physics, who gave me the opportunity to come to CERN to work on this analysis through the IPP Summer Student Fellowship, as well as the Natural Sciences and Engineering Research Council of Canada (NSERC) and McGill University for supporting this research. Lastly, I would like to thank Dr. Kun Liu and his group for sharing the details of their analysis with Run 1 data so I could have a starting point in terms of variable selection, and for introducing me to the idea of category classification.

Abstract

The observation of the production cross-section of a top-antitop pair in association with the Higgs boson ($t\bar{t}H$) with the ATLAS detector is one of the highest priorities in the physics plan at 13 TeV energies. MultiVariate Analysis techniques are planned to be implemented in the $t\bar{t}H$ multi-lepton channel during Run 2 data analysis as an improvement to the "cut and count" method used in Run 1. In this work, the Boosted Decision Trees and Neural Network algorithms are tested in $t\bar{t}H$ final states with two leptons with same sign electric charge and adjusted in order to obtain better sensitivity and higher separation power between signal and background compared to the counting experiment from Run 1.

Chapter 1

Introduction

The Large Hadron Collider (LHC) is the world's largest and most powerful particle collider. It was started in September 2008 and sought to test the predictions of high-energy physics from the Standard Model. After the long shutdown of the LHC from 2013 to 2015, the second run recently started on June 3rd, 2015. The ATLAS (A Toroidal LHC ApparatuS) collaboration will continue studying the Higgs processes, this time at higher energies, to improve upon the measurements of Run 1 (2009-2013) and access rare processes to explore new physics predictions.

A considerable amount of work must be done in order to upgrade the analysis of the incoming data. The focus of this project will be specifically on the improvement of the search for the associated production of $t\bar{t}$ (top-antitop pair) and a Higgs boson. The Run 1 $t\bar{t}H$ ATLAS analysis was done without MultiVariate Analysis (MVA) techniques (described in Chapter 5); the implementation of such techniques in Run 2 could significantly increase the sensitivity of the analyses and help achieve the 5 standard-deviation excess over the background only hypothesis in the measurement of the $t\bar{t}H$ cross-section (see Chapter 4).

The project involves becoming familiar with the analysis from Run 1 and the idea of a multivariate analysis (pros/cons of different algorithms), implementing the Toolkit for MultiVariate Analysis (TMVA) and comparing the performance of Boosted Decision Trees and Neutral Network algorithms (explained in Section 5.2) within TMVA. The final goal is to study the separation power between signal ($t\bar{t}H$) and background from kinematic variables, and to build a "discriminant" (see Section 6.2) from these variables using MVA techniques.

Chapter 2

The Standard Model

The Standard Model of Particle Physics is the current most widely accepted theory of fundamental particles and their interactions. It was developed in the early 1970's and has been successful in explaining most experimental results and in predicting most observed phenomena [6]. While it remains the theory which best represents the extent of our current understanding of particle physics, its limitations (such as its inability to explain the nature of dark matter, gravity or the excess of matter vs antimatter in our universe) hint at the fact that there is a next generation of theories that might better explain the laws of physics. An in-depth study of the outcomes of high-energy collisions will give an indication as to where the Standard Model begins to fail.

In its current state, the Standard Model includes three forces (weak, strong and electromagnetic) and two types of fundamental particles, each of which is separated into subcategories.

1. **Elementary Fermions** - Fermions have fractional spin. These are the particles that constitute matter and come with an associated antiparticle (particle of same mass but opposite charge). There are two types of fermions:

   (a) **Quarks** - Quarks come in six flavors and are associated in pairs, namely up(u)/down(d), charm(c)/strange(s) and top(t)/bottom(b). The analysis presented here will mainly focus on the top quark, which has the particularity of being the heav-

iest of the fundamental particles at 173.34 ± 0.27 (stat) ± 0.71 (syst) [4] GeV[1] and thus of having the strongest coupling to the Higgs field, as couplings to the Higgs boson have the characteristic that they are proportional to the mass of the particle. Quarks interact through all three forces and make up "hadrons" such as protons, neutrons, pions, etc.

(b) **Leptons** - These consist of electrons (e), muons (μ) and taus (τ), called lepton flavors, as well as their corresponding neutrinos (ν), organized in three generations similarly to the quarks. All leptons interact through the weak force; charged leptons (e, μ, τ) also interact through the electromagnetic force.

2. **Bosons** - Bosons are the force particles and have integer spin. There also two types of bosons:

   (a) **Spin-1 Gauge Bosons** - Consist of the gluon (g) carrier of the strong force, photon (γ) carrier of the electromagnetic force and the massive vector bosons (Z, W^+ and W^-) carriers of the weak force.

   (b) **Scalar Boson** - The latest addition to the Standard Model is the Higgs boson which is responsible for giving mass to the fundamental particles. Following the Higgs boson discovery at CERN in 2012, continuous efforts have been made to study its properties in order to investigate the validity of the Standard Model's predictions.

The structure of the Standard Model can be seen on Figure 2.1 where the red, green and orange sections contain fermions and the purple and blue sections contain bosons. Antiparticles are not shown.

[1] Units are chosen such that $\hbar$,c = 1; the particle's mass may be expressed in units of GeV instead of GeV/c^2.

Figure 2.1: The fundamental particles, as described by the Standard Model. Quarks are in red, neutrinos in green, charged leptons in orange and gauge bosons in purple and blue. The Higgs boson is in grey. Original image from [10].

Chapter 3

The LHC and the ATLAS Detector

The LHC was restarted on June 3rd, 2015 and has recently been producing collisions for physics analysis at a center-of-mass energy of 13 TeV. Run 1 was started in 2009 and produced 7 and 8 TeV collisions up until the shutdown in 2013. The first run allowed for many new physics results such as the discovery of the Higgs boson - many of its production and decay modes were analysed during this time.

3.1 ATLAS Detector

The ATLAS detector is one of the four detectors built around the LHC ring. It is one of the two "multi-purpose" detectors along with CMS[1]. This means that ATLAS is built to detect many different types of long-lived particles. For that purpose, it is composed of three main sub-detectors with several layers, each fine-tuned for specific measurements, as can be seen on Figure 3.1. The layers are used to "reconstruct" particles, meaning that the information coming from the detectors can be combined to infer what kind of particles were in the collision.

[1] CMS stands for Compact Muon Solenoid

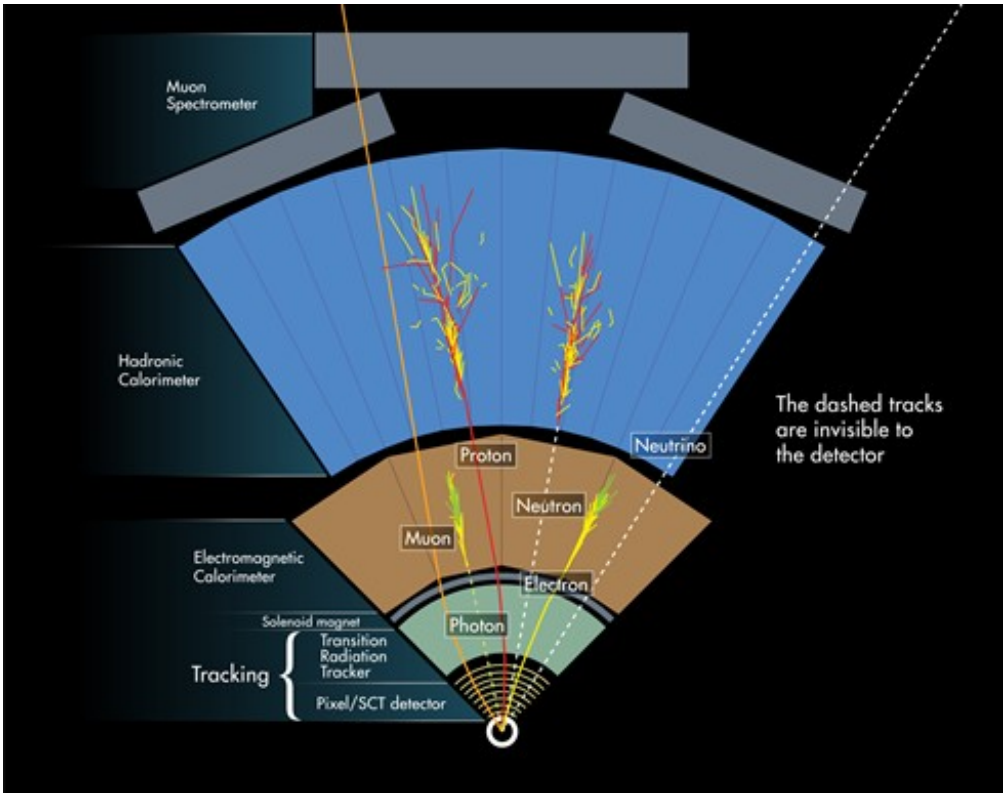


Figure 3.1: The ATLAS detector, composed of an inner detector, an elecromagnetic and a hadronic calorimeter, and a muon spectrometer. It is designed to be sensitive to almost all fundamental particles: only neutrinos go undetected. Image from [15].

Inner Detector - This is the first layer of detection, and it tracks charged particles that are deflected by a 2 Tesla magnetic field. The direction of motion and degree of curvature of the reconstructed tracks of the particles are used to infer their charge and momentum, respectively.

Calorimeters - The calorimeter is used to "shower" most of the particles by transforming the primary particle that interacts with the detector material into a cascade of secondary particles. The output is a cone of tracks (see photons, protons, neutrons and electrons in Figure 3.1) that can later be reconstructed as a single event and used to gain knowledge of the energy that the particles deposited in the calorimeter as they were stopped. The first layer is an electromagnetic calorimeter and is used to detect mainly electrons and photons via electromagnetic showers, while the second layer is called the hadronic calorimeter and

is used to detect jets[2]. In some cases it is possible to resolve the type of quark from which a jet originates through reconstruction algorithms. One of the most common examples is "b-tagging": hadrons containing a b-quark have a relatively long lifetime (about 1.5 ps [5]), and thus their tracks have a characteristic second vertex from which the cone originates and that differs from the primary vertex corresponding to the origin of the collision.

Muon Spectrometer - The last piece of the detector is used to track muons, which are the only detectable particles to pass through the calorimeter almost unaffected. The muon spectrometer functions in a similar way to the inner detector and is used to very precisely measure the momentum of the muon.

3.2 Kinematic Variables

The typical outputs of the detectors are called kinematic variables and characterize the particles' motion. All coordinates are set with respect to the center of the detector, where the collisions occur as shown in Figure 3.2.

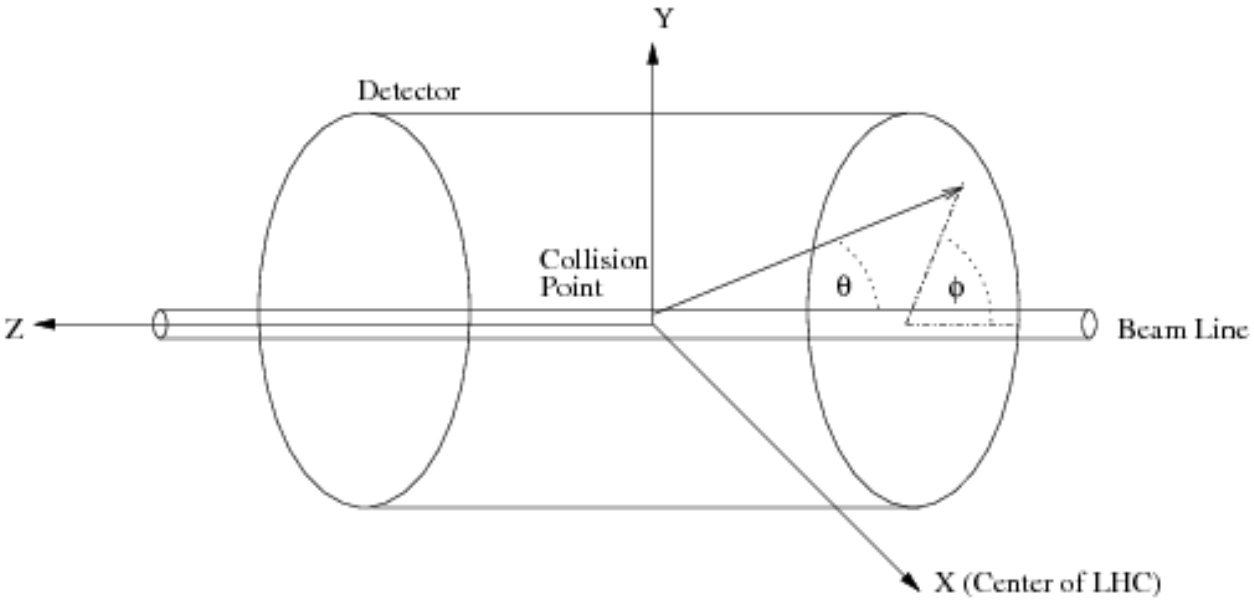


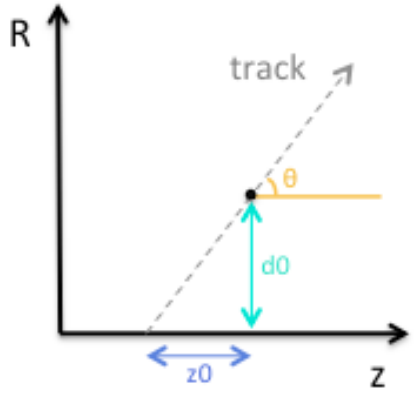
Figure 3.2: Coordinates in the ATLAS Detector. Image from [17].

[2]Because of the strong interaction, quarks and gluons are always found in confined states called hadrons. When created in LHC collisions, quarks and gluons hadronise due to color confinement (color is the quantum number of the strong interaction) which means that they produce quark-antiquark pairs from the vacuum to shield and surround them, creating the cones of tracks referred to as "jets"

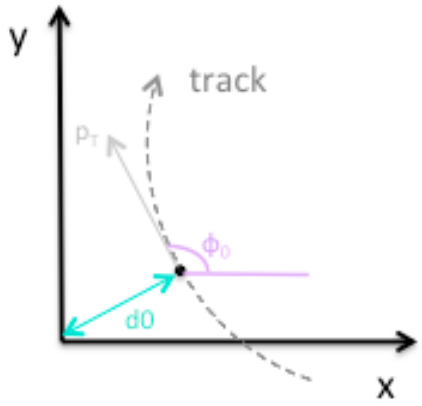
The coordinates η and ϕ denote the angular position of the particle track. ϕ is the azimuthal angle and η is the pseudorapidity related to the polar angle θ as $\eta = -ln[tan(\theta/2)]$. $\Delta\eta$ is invariant under Lorentz boosts in the direction of the beam axis, with $\eta = 0$ corresponding to a plane perpendicular to the beam axis and $\eta = \pm\infty$ corresponding to planes parallel to it. The z-coordinate corresponds to the beam axis.

Distances between tracks are calculated as $\Delta R = \sqrt{\Delta\eta^2 + \Delta\phi^2}$ and are interpreted as a solid angle from the center of the detector. R is used to denote radial distances from the collision point.

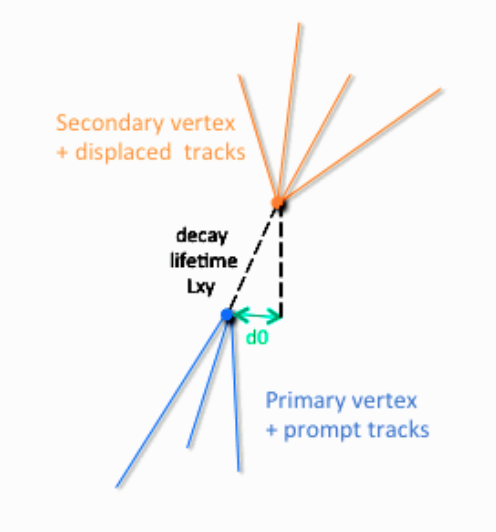
There are two impact parameters (transverse - d0, and longitudinal - z0), which can be defined as the point of closest approach to the beam axis as in Figures 3.3a and 3.3b or as the location of the principal vertex as in Figure 3.3c.



(a) d0 and z0 relative to the z-axis.



(b) d0 in the XY-plane for events with only one vertex.



(c) d0 for events with secondary vertices.

Figure 3.3: Impact parameters, as well as θ and ϕ. z0 is the z-coordinate of the track at the point of closest approach, and d0 is the signed distance to the z-axis.

Lastly, the parameters p_T and E_T characterize the particle's transverse momentum and transverse energy respectively (i.e., the projection in the transverse (ϕ) plane) [9], with $p_T = psin\theta$ and similarly for E_T. There is often an imbalance in energy in the transverse plane called the "missing transverse energy" or MET which is due to the presence in the collisions of undetectable particles such neutrinos or other non-detectable hypothetical particles from non-Standard Model theories.

Chapter 4

Top Quark

One way to probe the validity of the Standard Model's predictions is to study the couplings of the Higgs boson to other fundamental particles. The top quark is the main focus of this study and ideal for this purpose due to its very strong coupling to the Higgs (the so-called "Yukawa coupling" [16]). The quantification of the associated production of a top quark pair and a Higgs boson in high-energy collisions via the measurement of cross-sections (a measure of the rate of the process, which depends on the number of events recorded and the luminosity of the beam) would allow for the testing of the Standard Model.

At the LHC energy scales, the main production channel for $t\bar{t}H$ is through gluon-gluon fusion [14]. The $t\bar{t}H$ process, which generates a top quark, an anti-top quark and a Higgs boson, is illustrated in the Feynman diagram in Figure 4.1. This process provides the only direct measurement of the top quark Yukawa coupling, that the cross-section is proportional to.

The main challenge in performing these measurements is that the $t\bar{t}H$ process is rare in the Standard Model compared to other Higgs production modes, so very few events from the total collected events in the collisions will fulfill the requirements to be a $t\bar{t}H$ candidate. It is therefore very important to be able to select a set of data events from which most of the $t\bar{t}H$ events are selected (high signal efficiency) and few events from other processes contaminate this sample (high background rejection).

Top quarks have short lifetimes ($\approx$ 0.5 x 10^{-24} s [18]) and decay before they can be observed in the detector. Therefore, they are studied through their decay products (which

can be seen in the detector). The top quark typically decays to a b-quark and a W^+ boson, which then in turn decays to either an antilepton and a neutrino or a quark-antiquark pair, while the anti-top decays to W^- boson and an anti b-quark. This means that events containing these particles, collectively called "final states" or "signatures" of the top quark, are likely to contain a top quark. The Higgs boson is also observed through its decay products.

The final state studied in this report is the so-called "two leptons same sign" which is characterized mainly by the presence of two charged leptons (e or μ) of same sign electric charge (circled in Figure 4.1) and no taus decaying to hadrons as end products of the collision. This final state targets mostly $t\bar{t}H$ processes, where one of the top/antitop quarks decays "leptonically" (i.e., the corresponding W-boson decays in a lepton and a neutrino), the other top/antitop quark decays "hadronically" (i.e., the corresponding W-boson decays in a quark-antiquark pair), and the Higgs boson decays in two W bosons (this happens 80% of the time and is the main Higgs-boson decay mode) where at least one of them decays in a charged lepton of same sign electric charge as the one from the top quark. This state has the advantage of being rare in the Standard Model and therefore having lower background rates (i.e. only a few other processes can also result in this same final state). The events containing this signature can be split into categories with different signal-to-background ratios, for example, by the number of jets in the event or the lepton flavour which, in association to requirements on the lepton transverse momentum and isolation[1], results in a higher sensitivity to the $t\bar{t}H$ signal [7]. By searching events with two leptons of same sign, it is thus possible to "clean up" the set of all events recorded at the LHC and select only those that are the most likely to be $t\bar{t}H$ events.

[1] how much energy is deposited around the lepton track

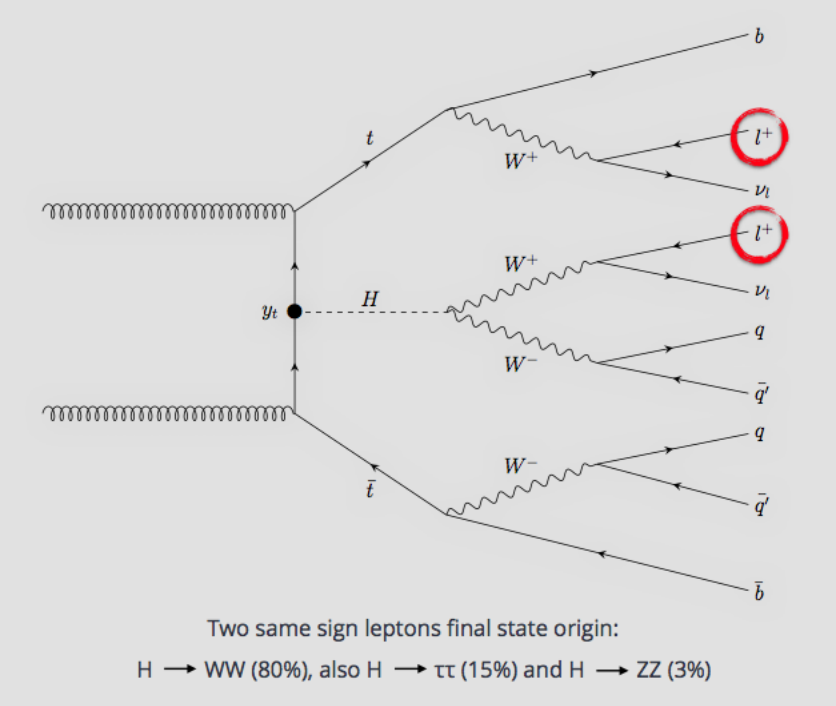
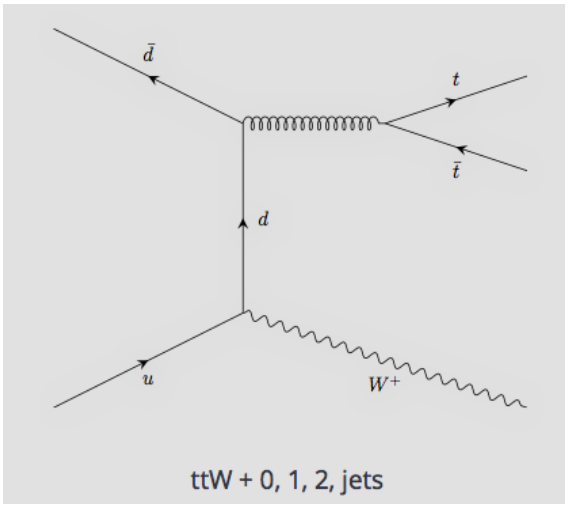


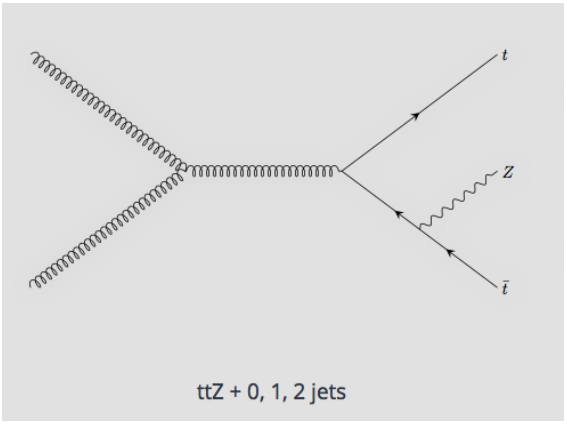
Figure 4.1: Higgs boson production in association with $t\bar{t}$ pair, decaying into a final state containing two leptons of same sign electric charge. Image from [8]

Throughout this report, processes related to $t\bar{t}H$ production and decay will be labeled as "signal" events. Processes which have the same signatures in the detectors but originate from different processes are labeled "background".

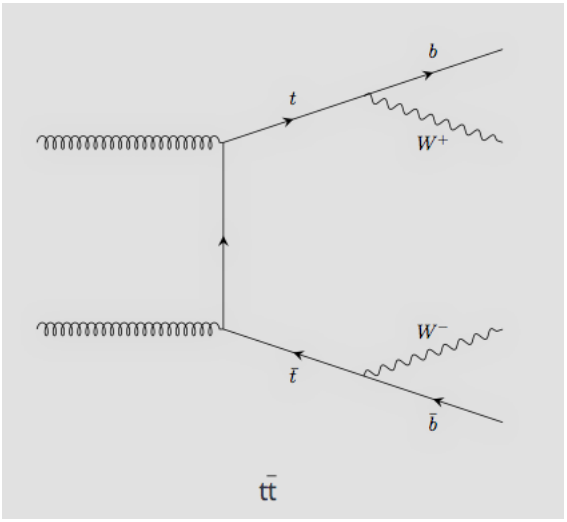
The main background processes include $t\bar{t}W$ (Figure 4.2a), $t\bar{t}Z$ (Figure 4.2b) and the $t\bar{t}$ (Figure 4.2c) events. The $t\bar{t}W$ and $t\bar{t}Z$ processes, collectively called $t\bar{t}V$, are the associated production of $t\bar{t}$ and a vector boson (W,Z), and are considered real backgrounds since these processes may also decay with two-lepton same-sign signatures. The $t\bar{t}$ process, on the other hand, contributes to background through reconstruction errors such as mis-identification of lepton charge or detection of "non-prompt leptons" (leptons coming from the decay of hadrons). Estimations for background processes rates come from simulation or auxiliary measurements using data from regions where a $t\bar{t}H$ signal is not expected.



(a) The $t\bar{t}W$ process.



(b) The $t\bar{t}Z$ process.



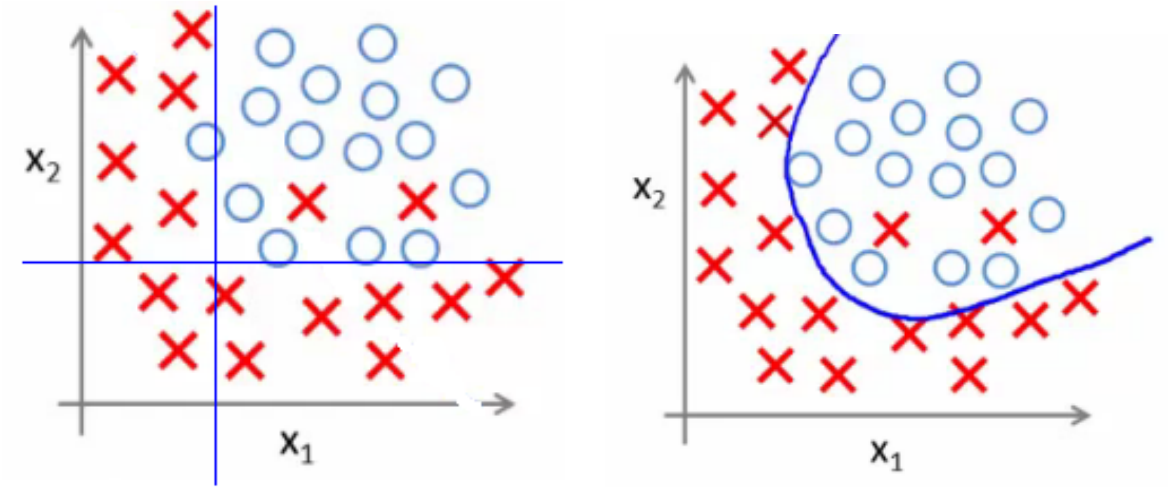
(c) The $t\bar{t}$ process.

Figure 4.2: Background processes for $t\bar{t}H$.

Chapter 5

Data Analysis

Once the two-lepton same-sign events have been selected, signal and background events need to be separated. In Run 1, events were classified as signal or background through the "cut and count" method which corresponds to applying cuts on kinematic variables, discarding the events that do not fulfil the requirements. It can be understood as splitting "signal"-like from "background"-like events through straight lines in the phase space built by these variables, as illustrated in Figure 5.1a. The stricter the cuts are, the fewer events will survive the selection, which can lead to large statistical uncertainties on the measurements done on the final sample. For this reason, the separation method is planned to be changed in Run 2 to what is called a "MultiVariate Analysis" (MVA) technique. These analyses consist of the use of algorithms to estimate the best way to separate the signal from the background in the phase space defined by a set of discriminating variables, as shown in Figure 5.1b. The signal samples' purity is highly increased and the classification mistakes are reduced.



(a) The "cut and count" method consists of creating a straight-line separation for signal (circles) and background (crosses) in some phase space defined by variables x1 and x2.

(b) MVA techniques estimate the signal/background-like regions better using a set of variables (x1,x2) and can reduce the classification error.

Figure 5.1: Comparison of classification methods. Red crosses represent background events and blue circles are signal events. Original images from [13] (Figure 5.1a has been changed from the original).

An in-depth study of different MVA algorithms is required to find the algorithm with the best performance. The goal of this project is to compare the performance of different MVA algorithms in distinguishing signal and background events. The data analysis is done through ROOT, which is an object-oriented program and library developed by CERN (for documentation on ROOT see [1]). TMVA is a toolkit to be used within ROOT and allows for the use and comparison of MultiVariate algorithms. Here, Boosted Decision Trees (BDTs) and Neural Networks via the commercial package NeuroBayes® (NB®) will be investigated in detail and optimized to get the best possible tool for a $t\bar{t}H$ analysis with Run 2 data.

5.1 MultiVariate Analysis

The idea of a MultiVariate Analysis is to figure out where the events lie in the phase space, and produce an output called the "final discriminant" with values between -1 and 1 that indicates this location. Events that are signal-like according to the information of the input kinematic variables will be given a final discriminant value close to 1 while events in the background region will be given a value close to -1. A threshold x_0 is selected and all events

with a discriminant value above x_0 are labeled as signal events while all events with a value below x_0 are labeled background.

Because signal and background events tend to have different kinematic variable distributions, the location in the phase space defined by these variables of any event is determined through the values that the variables take for that particular event.

In the next two sections, Boosted Decision Trees and Neural Networks will be described. Note that the algorithms are applied to a "training sample" to create the trees and the networks, meaning that they are provided with a sample with known signal and background events. This sample consists of Monte Carlo simulations of signal ($t\bar{t}H$) processes and of the different background processes at 8 TeV (Run 1), re-weighted to 13 TeV for Run 2 studies.

5.2 Boosted Decision Trees

Boosted Decision Trees consist of a series of binary splits of the initial set of events such that for each variable x, a specific value x_0 is chosen and all events with values above x_0 are separated from the events with a value below x_0. The value of x_0 is chosen such that the two split sub-samples will be purer in either signal or background than the original sample. The sub-samples are plotted against another variable y and a value y_0 is similarly chosen for this variable, and another split is made. Many splits can be made and a "tree" is grown. At the end, the small sub-samples are classified as signal or background samples according to what type of events they are mainly made up of. In Figure 5.2, the first variable chosen is the transverse momentum (p_T) of one of the leptons; the value of x_0 here is 6.08 x 10^4 MeV. Events with lepton p_T of less than that are put on the right sub-sample and those with higher values are sent to the sub-sample to the left. Other variables are chosen and more splits are made, for example by number of jets with p_T above 25 GeV (nJets25) or p_T of the other lepton.

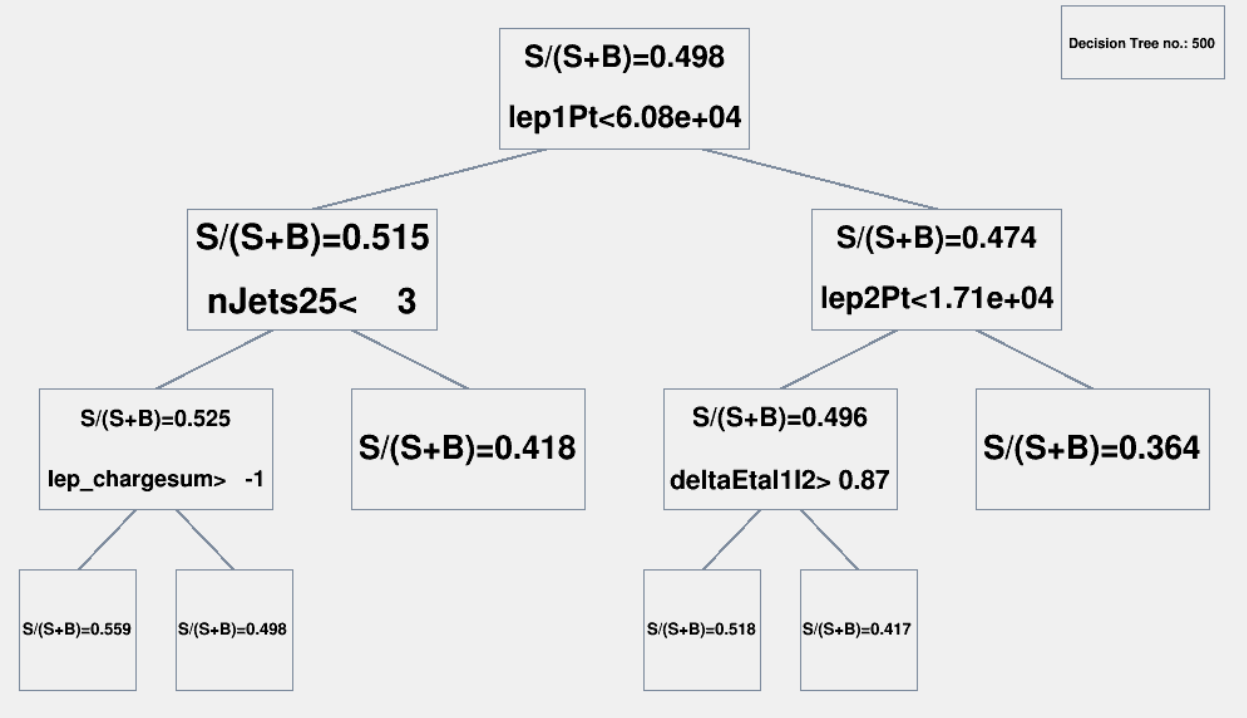


Figure 5.2: Decision trees: binary splits for separation of the signal from the background. This is a tree for the $t\bar{t}H$ vs $t\bar{t}$ training. The separation is illustrated as $S/(S+B)$ where S is the number of signal events and B the number of background events. Nodes with values closer to 1 contain more signal-like events while nodes with values closer to 0 contain more background-like events. For values "close enough" to 1 or 0, the nodes are labelled S or B. This is the tree #500 out of 850 trees grown during the training; other parameters were set such that the nodes must contain no less than 2.5% of the total number of events, a maximum amount of 3 separations may be made in a row (see method parameters on line 231 in Appendix 2).

Many trees are grown with cuts on variables in different orders (i.e., the split on y could be performed before the split on x) and then compared to each other. Events that end up in signal branches on most trees are given a discriminant value close to 1 and those that end up in background branches are given a value of about -1 [3]. The algorithm is applied on the events and the outcome is compared to the "true" classification from the Monte-Carlo simulation while the splits are created; events that have been mis-classified are given more weight after each tree is built so that they may be correctly classified in the next tree. This is called the "learning" phase.

5.3 NeuroBayes®

The NeuroBayes® tool consists of a neural network of non-linear decisions similar to those of firing neurons. Unlike decision trees where variables are analysed one by one ("linearly"), neural networks take in the information from all variables at once and decide if an event is more signal- or background-like based on the combination of the variable information. A typical Neural Network can be seen in Figure 5.3: the first layer of neurons contains the values of the input variables for each event, while the "hidden" layer consists of their combination, assigning a weight to each connection. Finally, the output neuron contains the discriminant value that has been assigned to the event. Some neurons are connected more tightly than others, in order to get the best "chain" possible, meaning that some nodes (variables) are given more weight in the decision process. Not much is known about the inner workings of the NeuroBayes® tool due to its non-linearity, although its performance can be easily checked through the outputs provided by the tool (see Section 6.2) [2], [19].

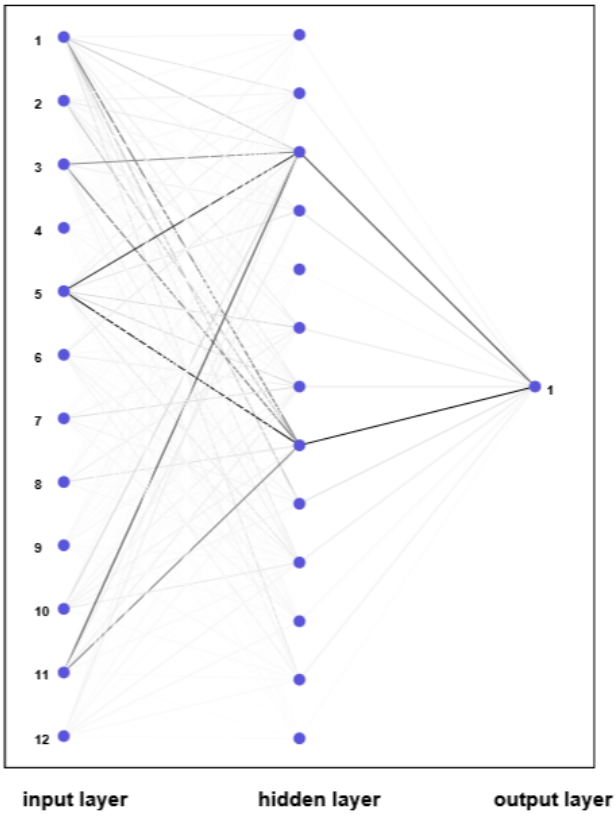


Figure 5.3: Neural Networks are made up of multiple layers. The input layer receives the inputs directly from the individual variables. The weights are shown as lighter or darker connections between nodes.

5.4 Classification

After the trees or networks are grown using the training sample, they are applied to a "test sample" (also from Monte-Carlo) but where the true classification is hidden. The events are allowed to go down the trees or through the neural networks created in the previous step and they are labeled as either signal or background depending on the value of discriminant that they receive. Then, the classification provided by the algorithms in the training phase is compared to the real classification of the events. If the algorithm performs well (i.e. it correctly classifies events as signal/background most of the time), it may be used for analysis. Otherwise, its parameters must be checked or new variables must be investigated, and the learning period restarted.

Chapter 6

Event and Object Selection

The input files consist of ROOT TTrees. A TTree contains information about all the events selected, and is made up of branches. Each branch contains information about the distribution of the events with respect to a certain variable. For example, one branch can contain information about the energy of the particles that came out of the collision for each event, while another branch contains information about the number of jets in each event. One can use these variables to distinguish signal and background events: the BDT and NeuroBayes® algorithms "learn" to distinguish the type of event based on these variables. TTrees are useful as they can be used to "cut" on the variables and see the effect of those cuts on other distributions. For example, events with leading lepton energy of less than 20GeV can be removed to reduce background; the branches are automatically updated and only the distributions for events which satisfy this requirement are kept. The cuts used in this analysis are looser than those used in Run 1 to increase statistics of the available simulated dataset and to use those variables in the MVA training instead of performing cuts on them. An example of a TTree is provided in Appendix 1.

6.1 TMVA Classification code

TMVA comes with a framework for comparing different algorithms' performance. One can manually select which variables to use for the training, and then decide if any preprocessing is to be done on them (for example, to label them as discrete or continuous variables). Once

this code is run, it provides the user with a Graphical User Interface, or GUI, that allows for the visualization of several training parameter outputs. The most important outputs for this analysis are presented in Section 6.2. Before any real analysis can be done, the code must be optimized with simulation data; examples of this step are shown below. Some of the code is provided in Appendix 2.

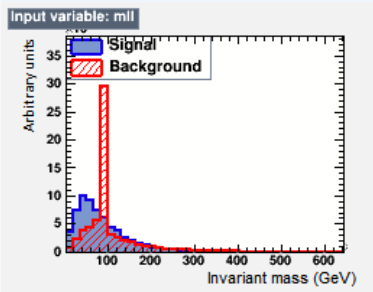
6.2 Outputs from the GUI

A. Input Variable Distributions

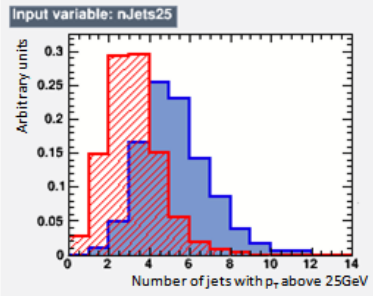
Often variables have different values, or their distributions have different shapes, for signal vs background events due to kinematic differences in the processes. It is thus important to understand the physics behind the differences to decide which variables are to be used. An example of variables with significant separation between signal and background is presented in Figure 6.1. For the $t\bar{t}Z$ background events in Figure 6.1a, note the peak in the invariant mass of the two leptons (mll) for background events, due to the two leptons coming from a Z boson: the peak is at the Z boson mass of about 90 GeV. This peak does not appear on signal event distributions. Signal events also tend to have more jets due to the Higgs decaying into two W bosons, one of them possibly decaying hadronically into jets, as illustrated in Figure 6.1b where the signal distribution (blue) shows that these events contain more jets on average. If an event has few jets and the invariant mass of the two leptons in it is of about 90 GeV this event is much more likely to be $t\bar{t}Z$ than $t\bar{t}H$. An event of this type would be assigned a value close to -1 by the algorithms.

Other backgrounds exhibit these differences in other variable distributions. For example, the W-boson from the $t\bar{t}W$ background is created through the interaction of $u\bar{d}$ (that creates a W^+ boson decaying leptonically, resulting in two ℓ^+ final signature) or $\bar{u}d$ (a W^- decaying leptonically resulting in two ℓ^- final signature) quarks. Protons consist of two u and one d valence quarks making the background ratio higher in the $2\ell^+$ category than the $2\ell^-$ one. This can be seen in Figure 6.1c where the background (red) peak at -2 is about half the height of the peak at +2 (meaning that there are about twice as many events with two ℓ^+)

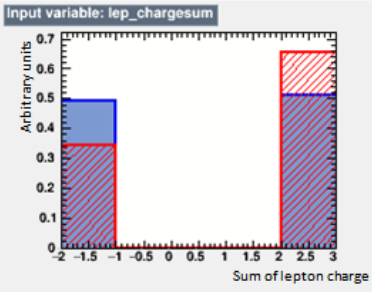
while for signal (blue) the two peaks have the same height.



(a) Invariant mass of the two leptons for $t\bar{t}H$ signal and $t\bar{t}Z$ background, expressed in [GeV] units. The background distribution is characterized by a peak at 90GeV.



(b) Number of jets per event for $t\bar{t}H$ signal and $t\bar{t}Z$ background.



(c) Sum of lepton electric charge for $t\bar{t}H$ signal and $t\bar{t}W$ background.

Figure 6.1: Distributions for a few selected variables with $t\bar{t}Z$ as background (Figures 6.1a and 6.1b) and $t\bar{t}W$ as background (Figure 6.1c).

B. Overtraining and Final Discriminant

The final discriminant (or simply discriminant) shows how signal- or background-like the events are. The trained algorithm is applied to two orthogonal samples: the training sample (used to train the MVA and create the trees/networks) and the test sample. The goal is to have signal (background) events with discriminant values as close to 1 (-1) as possible, and to have training and testing distributions similar in shape. It is often the case that the test sample does not yield results that are as good as the training samples due to what is called "overtraining": the algorithm learns from the statistical fluctuations of the training sample instead of from real physics information [11]. This can be seen in Figure 6.2, where the phase space is divided into signal and background regions in a way that could not be generalized to other samples (i.e., this shape is defined by the statistical fluctuations of the location of certain events in the phase space and not the real kinematic distinction between the two phase space regions).

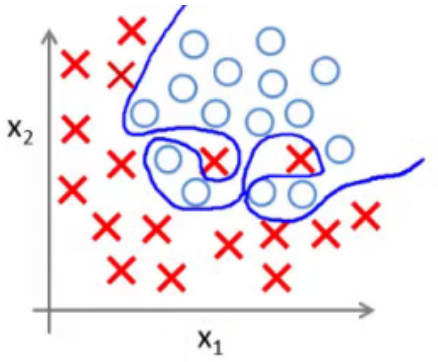
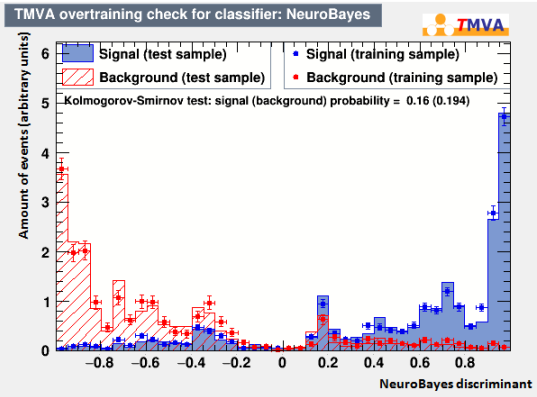
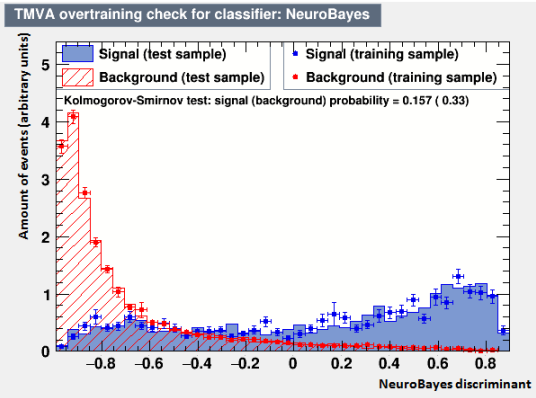


Figure 6.2: Overtraining in the (x1,x2) phase space.

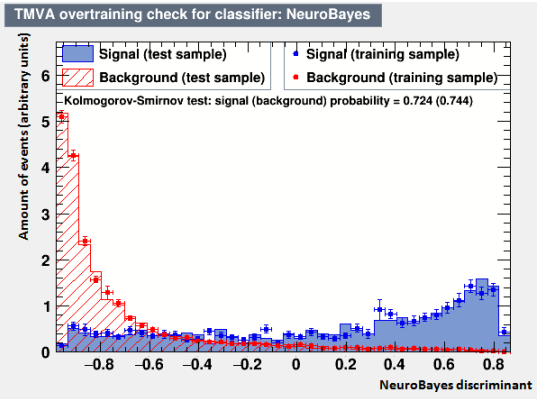
Often this may be solved by loosening the constraints on the training parameters, reducing the separation requirements or by increasing the size of the training sample. If training and testing events have statistically similar distributions (quantified by the Kolmogorov-Smirnov test [12], with a value close to 1 indicating identical distributions), then the algorithm is expected to perform well when applied to data. Some of the plots generated during the updates of the NB® training parameters are shown in Figure 6.3 to illustrate the improvement. The final discriminant is plotted normalized to the number of events. The goal here is to have all signal (blue) events accumulated near 1 and all the background events (read) near -1. In Figure 6.3a the events are not smoothly distributed and one can see quite a few signal events in the background region (negative discriminant). After a few adjustments of the parameters and a better selection of input variables, it was possible to get the distribution to look like the one in Figure 6.3d, where the events are well separated into two peaks focused at -1 and 1 while the center of the graph is relatively empty. This would correspond to two distinct regions in the phase space of Figure 5.1.



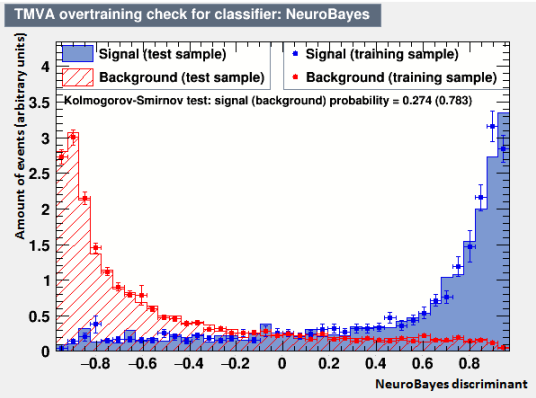
(a) One of the first attempts yielded an output with a highly irregular shape; NeuroBayes®default parameters had to be changed.



(b) The shape was then fixed but the "signal" distribution was too wide near 1; preprocessing had to be adjusted.



(c) The shape of the signal and background distributions were dissimilar; signal-to-background ratios had to be adjusted.



(d) Final output, all issues fixed. A clear separation can be seen, and signal can finally be distinguished from background.

Figure 6.3: Separation of signal and background events for training/testing.

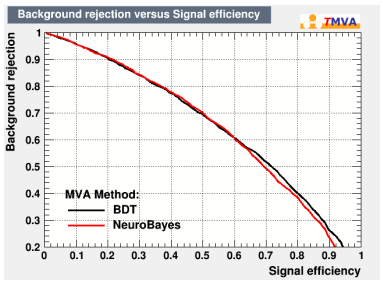
Overtraining checks can also be performed through these plots by comparing the testing (solid bands) and training (dots) sample distributions - matching shapes indicating less overtraining since it implies the algorithm always assigns similar values to similar events - or by looking at the result of the Kolmogorov-Smirnov test printed under the legend. Once again, values of about 1 for this test indicate similar training/testing sample distributions while values close to 0 indicate overtraining issues.

C. ROC curve

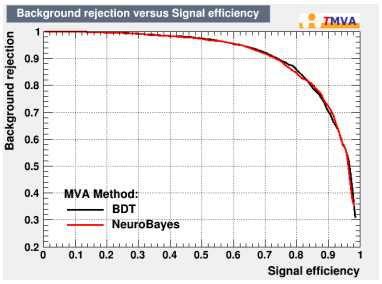
The receiver operating characteristic (ROC) curve shows the performance of different algorithms. It plots the signal efficiency (accepting as many true signal events as possible) and the background rejection (accepting least amount of background events as signal) as a function of the discriminant threshold. This means that a threshold is selected for the final discriminant and varied; events with final discriminant above the threshold are selected as signal events and those below the threshold are rejected. Signal efficiency is measured through $\epsilon_{sig} = \frac{P(selected\ and\ true)}{P(true)}$ where ϵ is the efficiency, P(selected and true) is the probability of a signal event being classified as signal and P(true) is the probability of any event being a signal event. An efficiency of 1 implies all signal events have been classified as signal. Similarly, background rejection is calculated as $(1 - \epsilon_{bkg})$, where ϵ_{bkg} is the probability of misclassifying a background event as signal over the probability of an event being a background event. Once again, a rejection of 1 implies all background events have been correctly classified as background. ROC curves are used to compare algorithms' performance (in this case BDT vs NB®), or to look at the impact on the performance of including additional variables. Curves closer to the (1,1) point of the graph at the top right (i.e., 100% signal efficiency and 100% background rejection) distinguish better algorithms for classification. Algorithms with ROC curves far from (1,1) can be improved through parameter tuning, in parallel to the study of training plots and improving the input set of variables. Examples are shown below where parameter optimization and a change of the input variables provided to the algorithms cause the curve to change from an almost diagonal line in Figures 6.4a (50% chance of getting the classification right) to the curve in Figure 6.4b which is much closer to the (1,1) point, corresponding to an algorithm capable of sorting signal events into a signal sample and background events into a different background sample. In this case, the red (NB) and blue (BDT) curves are very close to each other, i.e., the algorithms have comparable performances.

Given the different types of background processes, different decision trees can be built for different background classifications: for example, one separating $t\bar{t}H$ and $t\bar{t}V$ and another separating $t\bar{t}H$ and $t\bar{t}$. The $t\bar{t}W$ background is harder to separate from signal. In

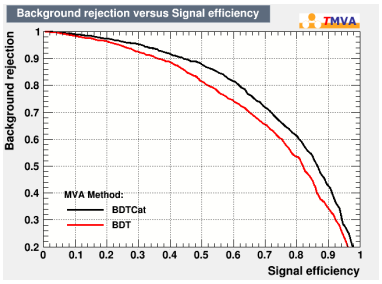
order to increase the separation of the $t\bar{t}H$ vs $t\bar{t}W$ final discriminant, final states with two positively charged leptons can be separated from those with two negatively charged leptons (see separation of the sum of lepton charges in Figure 3-2 (right)) as signal-to-background ratios differ in those two sections. The sample of events can be split into two sub-samples according to the charge of the leptons in them. A BDT is trained in each category and then the two are combined. This type of classification is called "category" (since it requires two different categories out of the sample of events: here, positively- and negatively-charged leptons) and has a considerably larger separation power, as is illustrated by the ROC curve in Figure 6.4c where the black curve (Category BDT) is closer to the top right corner than the original red (simple BDT) curve.



(a) ROC curve for the parameters as they were in Figure 6.3a ($t\bar{t}$ background).



(b) ROC curve with parameters as in Figure 6.3d ($t\bar{t}$ background).



(c) ROC curve showing the improvement in separation for BDT category classification ($t\bar{t}W$ background).

Figure 6.4: ROC Curves for BDT and NeuroBayes®. Note: the training against each type of background has a different maximal ROC curve associated to it; Figure 6.4a can be compared to Figure 6.4b since they both correspond to discriminants separating $t\bar{t}H$ and $t\bar{t}$. From the comparison of the two plots one can conclude that the parameter selection for the latter is advantageous for the separation, and the performance of the algorithms is better. Figure 6.4c can only be compared to other $t\bar{t}W$ outputs, so what one can conclude in this case is that the Category BDT classifier performs better than the standard BDT.

Chapter 7

Variable Ranking and Algorithm Performance

The NB® and BDT algorithms were optimized against $t\bar{t}W$ and $t\bar{t}$ backgrounds. From the original 46 variables, the following variables were found to be the ones with the most separation power (i.e., they provided the best classification when used together). All 46 variables were provided to the algorithms, outputting a raking by separation power taking into account the correlation between the variables, with the top variables being the most important ones for classification. The less powerful variables were removed and the ranking process repeated until the performance of the algorithms was observed to decrease due to the removal of more variables. The results shown in previous sections were obtained with the use of the 9 variables selected for $t\bar{t}$, as presented below.

7.1 $t\bar{t}$ Background

The following variables were used to separate $t\bar{t}$ background from $t\bar{t}H$ signal events. Some of these variables are individual particles' kinematic variables, such as `lep1Pt` (see Table 7.1 for variable descriptions) while others are general event variables, such as `nJets25` or even variables relating two particles, like the distance in R or η of the particles. All of these variables are provided in the input TTrees.

Ranking	Variable Name	Definition
1	mll	invariant mass of the two leptons
2	deltaRl1l2	ΔR between the two leptons
3	deltaEtal1l2	$\Delta\eta$ between the two leptons
4	nJets25	number of jets with p_T of at least 25 GeV in the event
5	nBJets25	number of jets with p_T of at least 25 GeV coming from a b-quark
6	MET	missing transverse energy
7	Ht	sum of the p_T of the two leptons and of the jets in the event
8	lep1Pt	p_T of the lepton with the highest energy (leading lepton)
9	lep2Pt	p_T of the other lepton (subleading lepton)

Table 7.1: Variables used in the classification of $t\bar{t}H$ signal events vs $t\bar{t}$ background events.

The use of these variables results in the ROC curve presented in Figure 7.1 for the two algorithms. The red (NB®) and black (BDT) curves are almost perfectly overlaid, meaning that the two algorithms perform equally well; they are both able to separate signal and background with similar signal efficiency and background rejection.

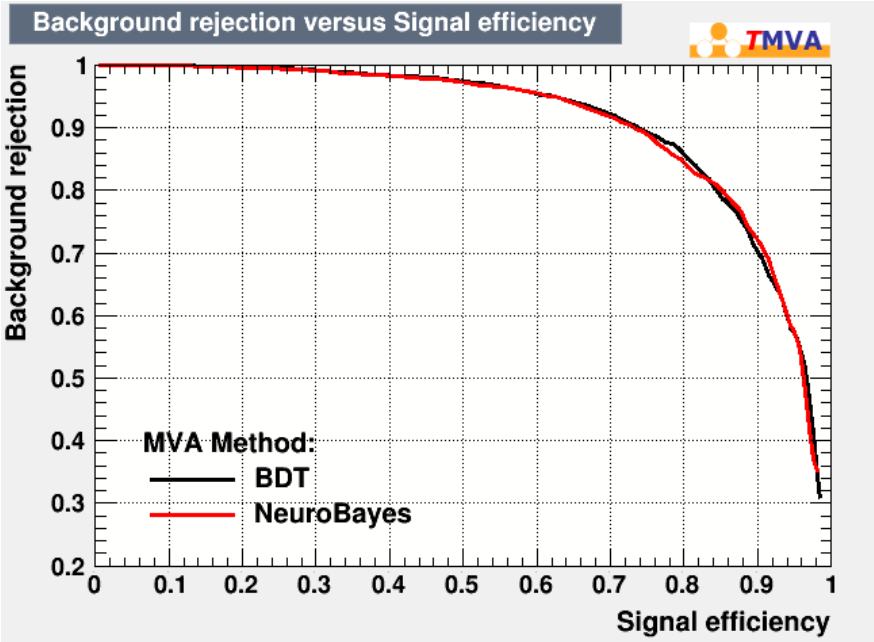
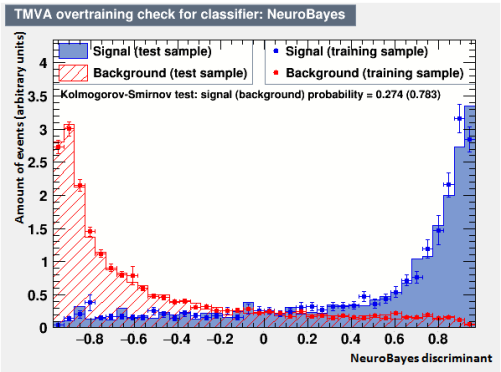


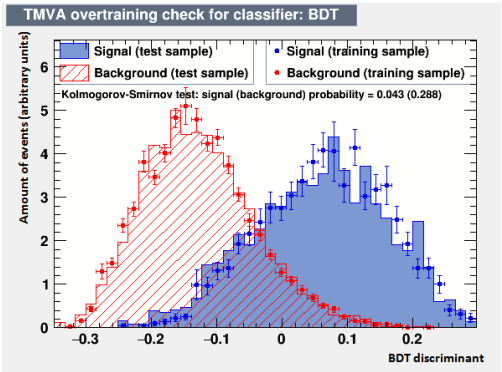
Figure 7.1: Signal efficiency vs background rejection for the $t\bar{t}$ background for Neural Networks and Boosted Decision Trees built from the 9 variables presented above. They both perform extremely well, as is illustrated by the fact that the curve approaches the (1,1) point.

It is important to note, nevertheless, that the Neural Network shows less overtraining than BDT, i.e. NB® is less likely to be affected by statistical fluctuations. This can be seen

in Figure 7.2 where the testing and training distributions are much more similar to each other for NeuroBayes® than for BDT, as indicated by the Kolmogorov-Smirnov test for signal and background. Therefore, NeuroBayes® is expected to perform better than BDT on data.



(a) Overtraining for Neural Networks - distributions look similar to each other and Kolmogorov-Smirnov test statistic is close to 1.



(b) Overtraining for Boosted Decision Trees - distributions look slightly different in shape and Kolmogorov-Smirnov test statistic value is close to 0.

Figure 7.2: Overtraining check for $t\bar{t}$ background shows that NeuroBayes® might perform better on data.

7.2 $t\bar{t}W$ Background

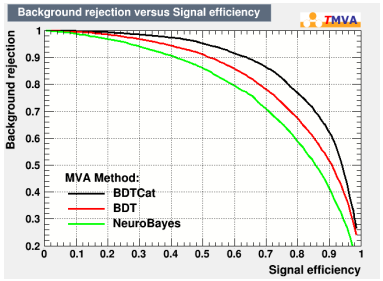
For the $t\bar{t}H$ vs $t\bar{t}W$ background MVA training, good separation was more difficult to achieve. In this case, different variables had to be used, and a Category classifier had to be introduced.

The variables had different rankings for the different algorithms; NeuroBayes® was able to separate signal and background better with a certain set of variables while BDT performed better with a different set, as can be seen in the rankings shown in Table 7.2.

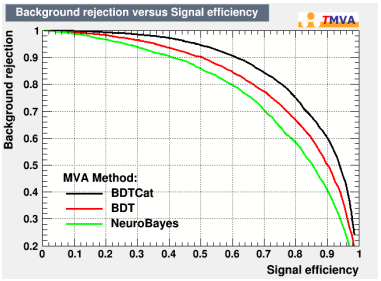
NB Ranking	BDT Ranking	Variable Name	Definition
1	1	nJets25	number of jets with p_T of at least 25 GeV in the event
2	2	mll	invariant mass of the two leptons
3	4	mindeltaRl2jet	distance in R between the subleading lepton and the closest jet
4	3	mindeltaRl1jet	distance in R between the leading lepton and its closest jet
5	17	MET	missing transverse energy
6	15	SumOfEtal1l2	sum of η-position value of the two leptons
7	13	nJets_total	total number of jets with p_T above 10 GeV in the event
8	5	deltaEtal1l2	distance in η between the two leptons
9	6	lep1Pt	p_T of the leading lepton
10	14	lep2caloiso	isolation of the subleading in the calorimeter
11	16	meff	sum of the p_T of all the leptons, jets and MET in the event
12	8	lep1Eta	η-location of the leading lepton
13	19	met_phi	ϕ-location of the MET
14	10	mt	transverse mass, with $mt(W) = \sqrt{2 \cdot lep1Pt \cdot MET \cdot (1 - cos(\Delta\phi_{l1,MET}))}$, of the leptonically-decaying W-boson resulting in the leading lepton
15	9	lep2Pt	p_T of the subleading lepton
16	11	lep2Eta	η location of the subleading lepton
17	12	lep1caloiso	isolation of the leading lepton in the calorimeter
18	7	deltaRl1l2	distance in R between the two leptons
19	22	jet_eta	η-location of all jets in the event
20	20	deltaPhil1l2	distance in ϕ between the two leptons
21	18	lep1Phi	ϕ location of the leading lepton
22	21	lep2Phi	ϕ location of the subleading lepton

Table 7.2: Variable ranking for NB® and BDT. Note that the ranking differs greatly for the two algorithms. For the distributions, see Appendix 3.

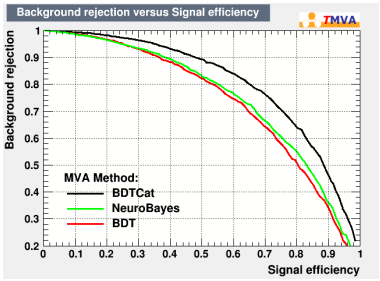
This variable list was chosen by first using all variables available (see Figure 7.3a) and then removing the lowest ranked variables until the ROC curve was seen to fall due to the removal of these variables (see Figure 7.3c). Using too many variables takes too much computing time and is inefficient, while using too few does not allow for proper signal/background separation and classification. The optimal point is the one with the fewest variables for the highest amount of separation. This point is the one with the 22 variables presented below and can be seen in Figure 7.3b, with curves almost identical to those of Figure 7.3a. Taking out only 4 extra variables and using the algorithms with 18 variables leads to the curves in Figure 7.3c showing a lower performance (it is closer to a diagonal line, and the curve gets "bumpy"), especially for the BDT algorithm (red curve).



(a) ROC curves for Neural Networks and Boosted Decision Trees when using 41 variables. The separation is very good as the curves are close to the top right corner.



(b) ROC curves for Neural Networks and Boosted Decision Trees when using 22 variables. The separation is still very good and the algorithm is now much faster since it deals with about half the amount of variables.



(c) ROC curves for Neural Networks and Boosted Decision Trees when using 18 variables. The separation power is decreased.

Figure 7.3: ROC Curves for different sets of input variables.

Using only the top 10 variables (to make the algorithms faster), the outputs in Figure 7.4 were produced to investigate the specific type of BDT that should be used. The types available in TMVA include BDTM (takes into account misclassifications of Monte-Carlo events and attempts to correct them), BDTS (optimizes tree growth by taking into account the separation at each branch using the formula $\frac{S}{\sqrt{S+B}}$ where S is the number of signal events and B is the number of background events) and BDTD (decorrelates variables before using them), as well as the Category classifier versions of some of them. The BDT and BDTM algorithms were found to be the most effective at classifying events when used with

the Category option; this can be seen from the fact that their corresponding curves are the closest to the top right corner of the plot. Neural Networks with categories were not explored as NeuroBayes® does not have a Category option.

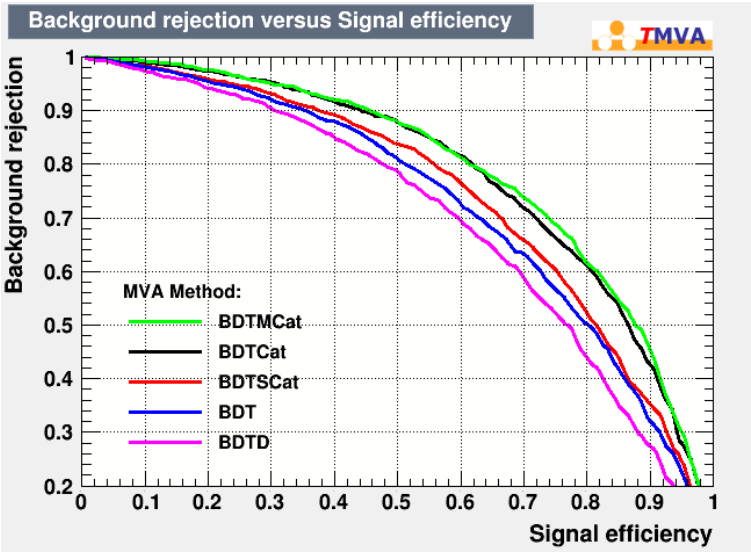
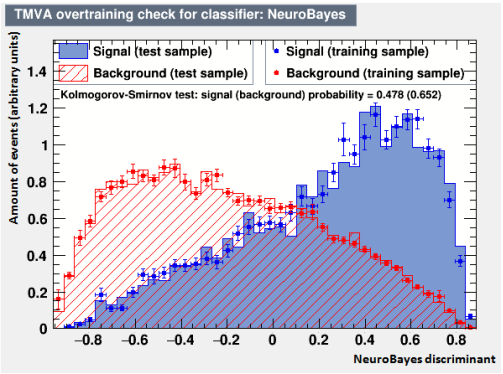


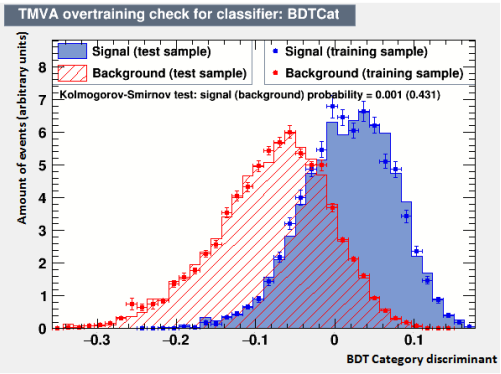
Figure 7.4: Different types of BDT classification algorithms were explored; BDT Category (black) and BDTM Category (green) perform better than the other versions of this algorithm.

Using the set of 22 variables, BDT and BDT Category were compared to the NeuroBayes® algorithm with the latter performing significantly less well. As can be seen in Figure 7.3b, BDT Category (black curve) was able to separate signal and background events with the highest efficiency, indicating that for this background BDT Category works much better.

The overtraining check outputs, on the other hand, follow the same trend as the ones for $t\bar{t}$ meaning that NeuroBayes® might still work better than BDT when applied to data as it is less overtrained.



(a) Overtraining check for NeuroBayes.



(b) Overtraining check for BDTCat (BDT Category).

Figure 7.5: Overtraining checks show that NeuroBayes is less influenced by statistical fluctuations, as illustrated by the higher value of the Kolmogorov-Smirnov test.

7.3 $t\bar{t}Z$ Background

The expected contribution of the $t\bar{t}Z$ process at 13 TeV is approximately $\frac{1}{3}$ of the $t\bar{t}W$ contribution in the two-lepton same-sign channel [8]. For simplicity, and because $t\bar{t}Z$ events tend to have similar kinematic distributions to $t\bar{t}W$ events, the two are often combined into a single $t\bar{t}V$ background and trained together or, alternatively, the trees/networks are trained against $t\bar{t}W$ and then applied to the $t\bar{t}Z$ background.

Chapter 8

Conclusion

The BDT and Neural Network algorithms were optimized for the separation of $t\bar{t}H$ signal from $t\bar{t}$ and $t\bar{t}W$ backgrounds separately, showing different separation power for the different backgrounds. For $t\bar{t}$ backgrounds, Neural Networks are the ideal candidate as they perform as well as BDT on Monte Carlo events but are less affected by statistical fluctuations (less likely to be overtrained). For $t\bar{t}W$, Boosted Decision Trees have a better performance as seen on the ROC curves in Figure 7.3b, with the BDT Category classifier significantly better at discriminating between signal and background events. Neural Networks are nevertheless less affected by statistics and might still be a valid candidate towards the $t\bar{t}H$ Run 2 data analysis.

Before these algorithms are ready to be used in the Run 2 $t\bar{t}H$ analysis, the $t\bar{t}V$ background should be investigated and trained as a single background, and new topological variables (which describe the shape of the event, e.g., whether the particles were projected along the beam axis or in a spherical shape after the collision) could be created in order to test their impact on the separation power of the final discriminant.

Bibliography

[1] *ROOT Data Analysis Framework: Documentation*, https://root.cern.ch/drupal/content/documentation.

[2] *The NeuroBayes® User's Guide*, Version April 6, 2010.

[3] *TMVA4 Users Guide*, (October 4, 2013), tmva.sourceforge.net/docu/TMVAUsersGuide.pdf.

[4] The ATLAS, CDF, CMS, and D0 Collaborations, *First combination of Tevatron and LHC measurements of the top-quark mass*, (March 18, 2014), http://arxiv.org/abs/1403.4427.

[5] J. A. Atwood, W. B.and Jaros, *B-hadron lifetimes*, (October 1991), http://www.slac.stanford.edu/cgi-wrap/getdoc/slac-pub-5671.pdf.

[6] CERN, *Standard model*, (2015), http://home.web.cern.ch/about/physics/standard-model.

[7] The ATLAS Collaboration, *Search for the associated production of the Higgs boson with a top quark pair in multi-lepton final states with the ATLAS detector*, ATLAS-CONF-2015-006 (March 17, 2015).

[8] Kevin De Vasconcelos, Fabrice Hubaut, Kun Liu, and Pascal Pravalorio, *Selection optimization using MVA for ttH signal in same-sign leptons channel*, Internal ttH->leptons meeting (June 16, 2015).

[9] David Layden, *Measuring 2012 photon trigger efficiency in the atlas experiment*, (2013).

[10] Symmetry Magazine, *Standard Model Discoveries*, (May 2009), http://www.symmetrymagazine.org/article/may-2009/deconstruction-standard-model-discoveries.

[11] Daniel Martschei, Michael Feindt, and Simon Honc, *Different benchmarks for MVA methods - comparison of methods from TMVA with NeuroBayes*, (December 13, 2010).

[12] Mathworks, *Two-sample kolmogorov-smirnov test*, http://ch.mathworks.com/help/stats/kstest2.html#btn37ur.

[13] Andrew Ng, *The problem of overfitting*, (2011), http://www.holehouse.org/mlclass/07_Regularization.html.

[14] Mark Owen, *Search for ttH production at the LHC*, PowerPoint Presentation on behalf of the ATLAS and CMS collaborations, University of Glasgow (2014).

[15] Joao Pequenao and Paul Schaffner, (January 16, 2013), https://cds.cern.ch/record/1505342.

[16] Arcadi Santamaria, *Masses, Mixings, Yukawa Couplings and their Symmetries*, (February 1993), https://cds.cern.ch/record/246574/files/9302301.pdf.

[17] Mathias Schott et al., *Review of single vector boson production in pp collisions at* $\sqrt{s}$*=7 tev*, Eur.Phys.J. C74 (2014), http://inspirehep.net/record/1294662/plots, arXiv:1405.1160.

[18] Christian Schwanenberger, *TopSummary_TOP2014*, Experimental Summary, PowerPoint Presentation, University of Manchester (2014).

[19] Daniel Shiffman, *The Nature of Code: Chapter 10. Neural Networks*, 2012.

Appendices

Appendix 1: ROOT TTrees

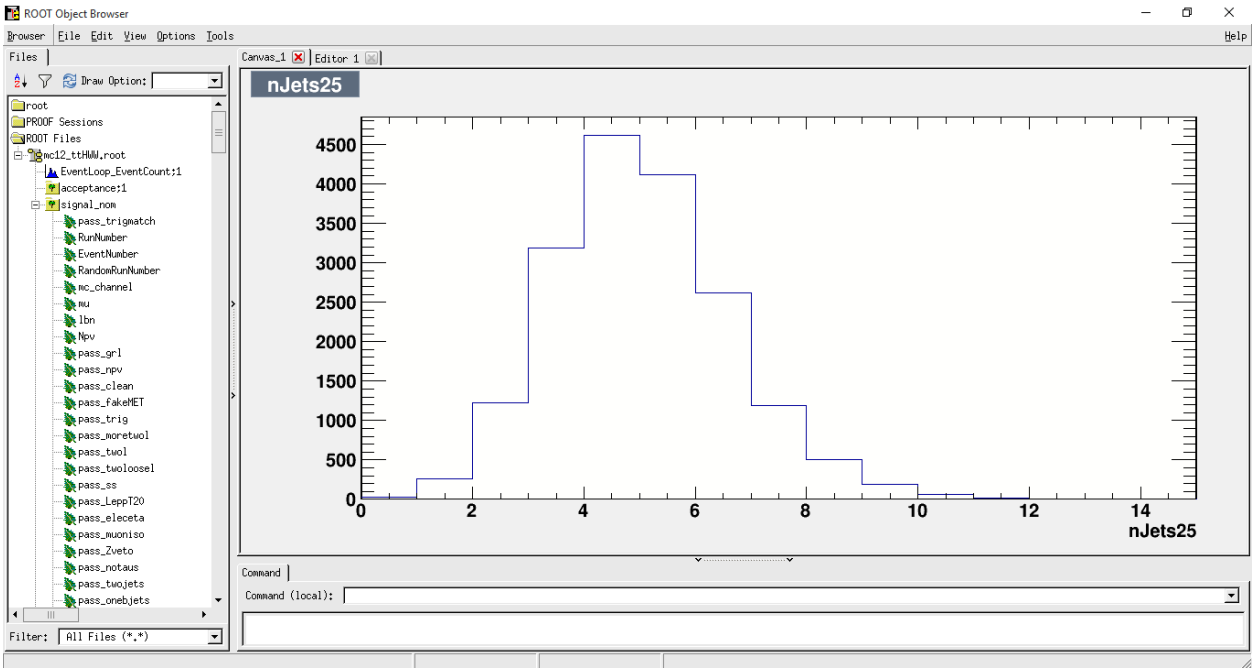


Figure 8.1: A TTree called `signal_nom` for the $t\bar{t}H$ events, with branches (to the left) and the variable `nJets25` plotted (right).

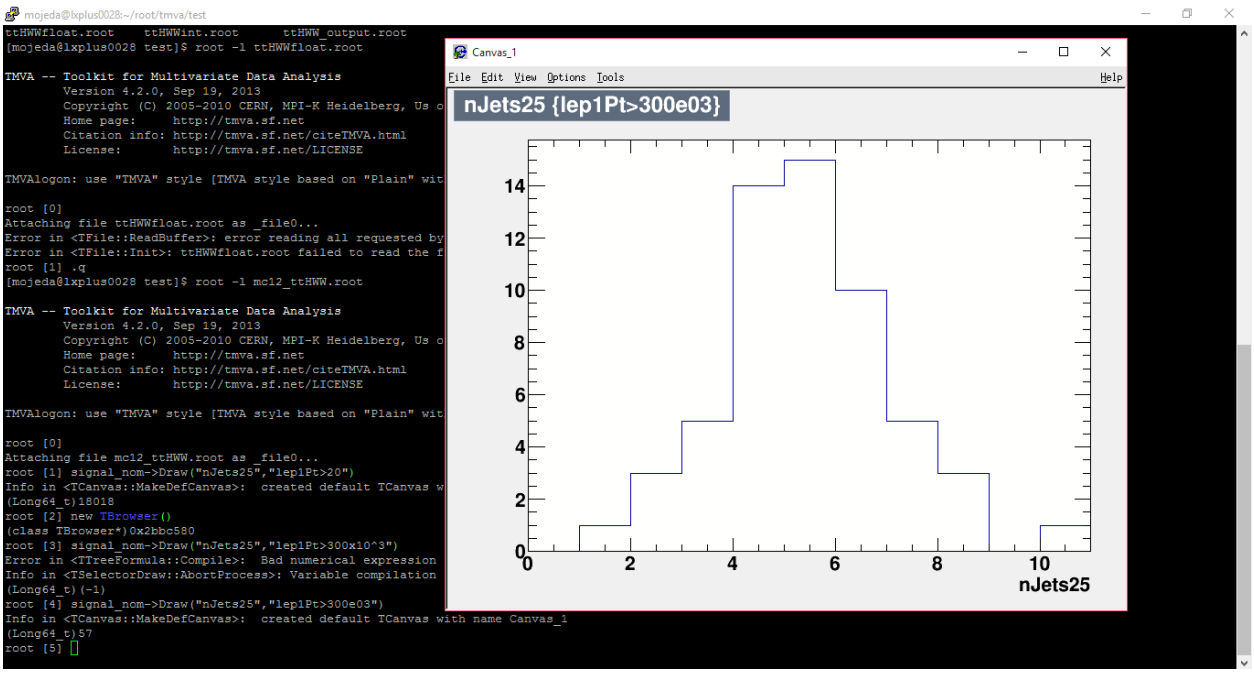


Figure 8.2: Using the TTree from Figure 8.1, the variable `nJets25` is plotted after applying a cut that selects only events with leading lepton p_T of at least 300GeV (`lep1Pt>300e3`).

Appendix 2: TMVAClassificationCategory.C

Code for loading input trees, declaring methods used (BDT and NN) and setting the testing parameters as well as declaring variables.

```
1 // @(#)root/tmva $Id: TMVAClassificationCategory.C,v 1.36 2009-04-14 13:08:13
    andreas.hoecker Exp $
2 /*
    **********************************************************************************

3  * Project    : TMVA - a Root-integrated toolkit for multivariate data analysis
         *
4  * Package    : TMVA
         *
5  * Root Macro: TMVAClassificationCategory
         *
6  *
         *
7  * This macro provides examples for the training and testing of the
         *
8  * TMVA classifiers in categorisation mode.
         *
9  *
         *
```

```
 * As input data is used a toy-MC sample consisting of four Gaussian-
    distributed  *
 * and linearly correlated input variables with category (eta) dependent
      *
 * properties.
      *
 *
      *
 * For this example, only Fisher and Likelihood are used. Run via:
      *
 *
      *
 *    root -l TMVAClassificationCategory.C
      *
 *
      *
 * The output file "TMVA.root" can be analysed with the use of dedicated
      *
 * macros (simply say: root -l <macro.C>), which can be conveniently
      *
 * invoked through a GUI that will appear at the end of the run of this macro.
      *

    **********************************************************************
    */

#include <cstdlib>
#include <iostream>
#include <map>
#include <string>

#include "TChain.h"
#include "TFile.h"
#include "TTree.h"
#include "TString.h"
#include "TObjString.h"
#include "TSystem.h"
#include "TROOT.h"
#include "TPluginManager.h"


#if not defined(__CINT__) || defined(__MAKECINT__)
// needs to be included when makecint runs (ACLIC)
#include "TMVA/MethodCategory.h"
#include "TMVA/Factory.h"
#include "TMVA/Tools.h"
#endif

// two types of category methods are implemented
Bool_t UseOffsetMethod = kTRUE;

void TMVAClassificationCategory()
{
   //---------------------------------------------------------------
```

```
51    // Example for usage of different event categories with classifiers
52
53    std::cout << std::endl << "==> Start TMVAClassificationCategory" << std::
      endl;
54
55    // This loads the library
56    TMVA::Tools::Instance();
57
58    // to get access to the GUI and all tmva macros
59    TString tmva_dir(TString(gRootDir) + "/tmva");
60    if(gSystem->Getenv("TMVASYS"))
61       tmva_dir = TString(gSystem->Getenv("TMVASYS"));
62    gROOT->SetMacroPath(tmva_dir + "/test/:" + gROOT->GetMacroPath() );
63    gROOT->ProcessLine(".L TMVAGui.C");
64
65
66    bool batchMode = false;
67
68    // Create a new root output file.
69    TString outfileName( "TMVA.root" );
70    TFile* outputFile = TFile::Open( outfileName, "RECREATE" );
71
72    // Create the factory object (see TMVAClassification.C for more information
      )
73
74   std::string factoryOptions( "!V:!Silent:Transformations=I" );
75   if (batchMode) factoryOptions += ":!Color:!DrawProgressBar";
76
77    TMVA::Factory *factory = new TMVA::Factory( "TMVAClassificationCategory",
      outputFile, factoryOptions );
78
79    // Define the input variables used for the MVA training
80
81    // L2SS criteria
82    //  factory->AddSpectator( "pass_ss", 'F' );
83    // factory->AddSpectator( "pass_notaus", 'F' );
84    //  factory->AddVariable( "pass_twojets", 'I');
85
86    // weights
87    //  factory->AddVariable( "mcWgt14TeV", 'F');
88    //  factory->AddVariable( "mcWgt13TeV", 'F');
89    //  factory->AddVariable( "pileupWgt", 'F');
90    //  factory->AddVariable( "mc_weight", 'F');
91    //  factory->AddVariable( "l1_weight", 'F');
92    //  factory->AddVariable( "l2_weight", 'F');
93    //  factory->AddVariable( "trigger_weight", 'F');
94    //  factory->AddVariable( "event_weight", 'F');
95
96    // Other variables
97    // All integer variables were changed to floats since the Application code
      does not work with int
98    //  factory->AddVariable( "pass_LeppT20", 'I' );
99    //  factory->AddVariable( "pass_eleceta", 'I' );
100   //  factory->AddVariable( "pass_muoniso", 'I' );
```

```
101     //  factory->AddVariable( "pass_fourjets", 'I' );
102     //  factory->AddVariable( "pass_onebjets", 'I' );
103     //  factory->AddVariable( "nLep10", 'I' ); // constant after cuts!
104     //  factory->AddVariable( "ntaus", 'I' ); // not v. good
105     factory->AddVariable( "mll", 'F' );
106     //  factory->AddVariable( "chanDilep", 'I' );
107     // factory->AddSpectator( "lep_chargesumFloat", 'F' );
108     factory->AddVariable( "deltaRl1l2", 'F' );
109     factory->AddVariable( "deltaEtal1l2", 'F' );
110     factory->AddVariable( "deltaPhil1l2", 'F' );
111     factory->AddVariable( "SumOfEtal1l2", 'F' );
112     factory->AddVariable( "nJets_totalFloat", 'F' );
113     factory->AddVariable( "nJets25Float", 'F' );
114     // factory->AddVariable( "nBJets25Float", 'F' );
115     // factory->AddVariable( "nJets30Float", 'F' );
116     // factory->AddVariable( "nJets40Float", 'F' );
117     // factory->AddVariable( "nJets50Float", 'F' );
118     // factory->AddVariable( "nJets60Float", 'F' );
119     // factory->AddVariable( "nJets70Float", 'F' );
120     // factory->AddVariable( "jet_pt", 'F' );
121     factory->AddVariable( "jet_eta", 'F' );
122     // factory->AddVariable( "jet_phi", 'F' );
123     // factory->AddVariable( "Isbjet", 'I' ); //boolean
124     factory->AddVariable( "met", 'F' );
125     // factory->AddVariable( "met_phi", 'F' );
126     // factory->AddVariable( "mt", 'F' );
127     factory->AddVariable( "meff", 'F' );
128     // factory->AddVariable( "ht", 'F' );
129     factory->AddVariable( "lep_flavourFloat", 'F');
130
131     //lepton #1
132     // factory->AddVariable( "lep1Ele", 'I' ); //boolean
133     factory->AddVariable( "lep1Pt", 'F' );
134     factory->AddVariable( "lep1Eta", 'F' );
135     // factory->AddVariable( "lep1Phi", 'F' );
136     // factory->AddVariable( "lep1trkiso", 'F' );
137     // factory->AddVariable( "lep1caloiso", 'F' );
138     // factory->AddVariable( "lep1z0", 'F' );
139     // factory->AddVariable( "lep1z0sin", 'F' );
140     // factory->AddVariable( "lep1d0sig", 'F' );
141     // factory->AddVariable( "lep1chargeFloat", 'F' );
142     // factory->AddVariable( "l1_weight", 'F' );
143     factory->AddVariable( "mindeltaRl1jet", 'F' );
144
145     //lepton #2
146     // factory->AddVariable( "lep2Ele", 'I' ); //boolean
147     factory->AddVariable( "lep2Pt", 'F' );
148     factory->AddVariable( "lep2Eta", 'F' );
149     // factory->AddVariable( "lep2Phi", 'F' );
150     // factory->AddVariable( "lep2trkiso", 'F' );
151     // factory->AddVariable( "lep2caloiso", 'F' );
152     // factory->AddVariable( "lep2z0", 'F' );
153     // factory->AddVariable( "lep2z0sin", 'F' );
154     // factory->AddVariable( "lep2d0sig", 'F' );
```

```
155    //factory->AddVariable( "lep2chargeFloat", 'F' );
156    // factory->AddVariable( "l2_weight", 'F' );
157    factory->AddVariable( "mindeltaRl2jet", 'F' );
158
159    // You can add so-called "Spectator variables", which are not used in the
       MVA training,
160    // but will appear in the final "TestTree" produced by TMVA. This TestTree
       will contain the
161    // input variables, the response values of all trained MVAs, and the
       spectator variables
162    //  factory->AddSpectator( "eta" );
163
164    // Load the signal and background event samples from ROOT trees
165    // TFile *input(0);
166    //TString fname( "" );
167    //if (UseOffsetMethod) fname = "data/toy_sigbkg_categ_offset.root";
168    //else                 fname = "data/toy_sigbkg_categ_varoff.root";
169    //if (!gSystem->AccessPathName( fname )) {
170       // first we try to find tmva_example.root in the local directory
171    //   std::cout << "--- TMVAClassificationCategory: Accessing " << fname <<
       std::endl;
172    //   input = TFile::Open( fname );
173    //}
174
175
176    //
       ---------------------------------------------------------------------------

177    // Load Input TTrees
178    //
       ---------------------------------------------------------------------------

179
180    TString signalname = "./ttHWWfloat.root";
181    TString backgroundname = "./ttWfloat.root";
182    //other backgrounds: mc12_ttZ.root, mc12_ttHtautau.root, mc12_ttHZZ.root,
       mc12_ttbar.root
183
184    TFile *signalinput = TFile::Open( signalname );
185    TFile *backgroundinput = TFile::Open( backgroundname );
186
187    std::cout << "--- TMVAClassificationCategroy    : Using input files: " <<
       signalinput->GetName() << " and " << backgroundinput->GetName() << std::
       endl;
188
189
190    TTree *signal     = (TTree*)signalinput->Get("signal_nom");
191    TTree *background = (TTree*)backgroundinput->Get("ttw_nom");
192
193    /// Global event weights per tree (see below for setting event-wise weights
       )
194    Double_t signalWeight     = 1.0;
195    Double_t backgroundWeight = 1.0;
196
```

```
197     /// You can add an arbitrary number of signal or background trees
198     factory->AddSignalTree    ( signal,     signalWeight     );
199     factory->AddBackgroundTree( background, backgroundWeight );
200
201     factory->SetSignalWeightExpression    ( "mcWgt13TeV*pileupWgt*mc_weight*
        lumiScaling" );
202     factory->SetBackgroundWeightExpression( "mcWgt13TeV*pileupWgt*mc_weight*
        lumiScaling" );
203
204     // Apply additional cuts on the signal and background samples (can be
        different)
205     // Cuts for 2LSS channel
206     TCut mycuts = "pass_ss && pass_notaus";
207     TCut mycutb = "pass_ss && pass_notaus";
208
209     // Tell the factory how to use the training and testing events
210     factory->PrepareTrainingAndTestTree( mycuts, mycutb,
211                                          "nTrain_Signal=0:nTrain_Background=0:
        SplitMode=Random:NormMode=EqualNumEvents:!V" );
212     // no requirements on training samples except #signal events must equal #
        background events (EqualNumEvents)
213
214
215
216
217
218
219
220     // ---- Book MVA methods (NOT Category)
221
222
223     // Boosted Decision Trees
224
225     // Gradient Boost
226     //    factory->BookMethod( TMVA::Types::kBDT, "BDTG",
227     //                         "!H:!V:NTrees=1000:MinNodeSize=2.5%:BoostType=
        Grad:Shrinkage=0.10:UseBaggedBoost:BaggedSampleFraction=0.5:nCuts=20:
        MaxDepth=2" );
228
229     // Adaptive Boost
230        factory->BookMethod( TMVA::Types::kBDT, "BDT",
231                             "!H:!V:NTrees=850:MinNodeSize=2.5%:MaxDepth=3:
        BoostType=AdaBoost:AdaBoostBeta=0.5:UseBaggedBoost:BaggedSampleFraction
        =0.5:SeparationType=GiniIndex:nCuts=20" );
232
233        //850 trees with nodes of at least 2.5% the size of the original dataset
        , max of 3 splits per tree, separate by GiniIndex.
234
235     // Bagging
236     //    factory->BookMethod( TMVA::Types::kBDT, "BDTB",
237     //                         "!H:!V:NTrees=400:BoostType=Bagging:
        SeparationType=GiniIndex:nCuts=20" );
238
239     // Decorrelation + Adaptive Boost
```

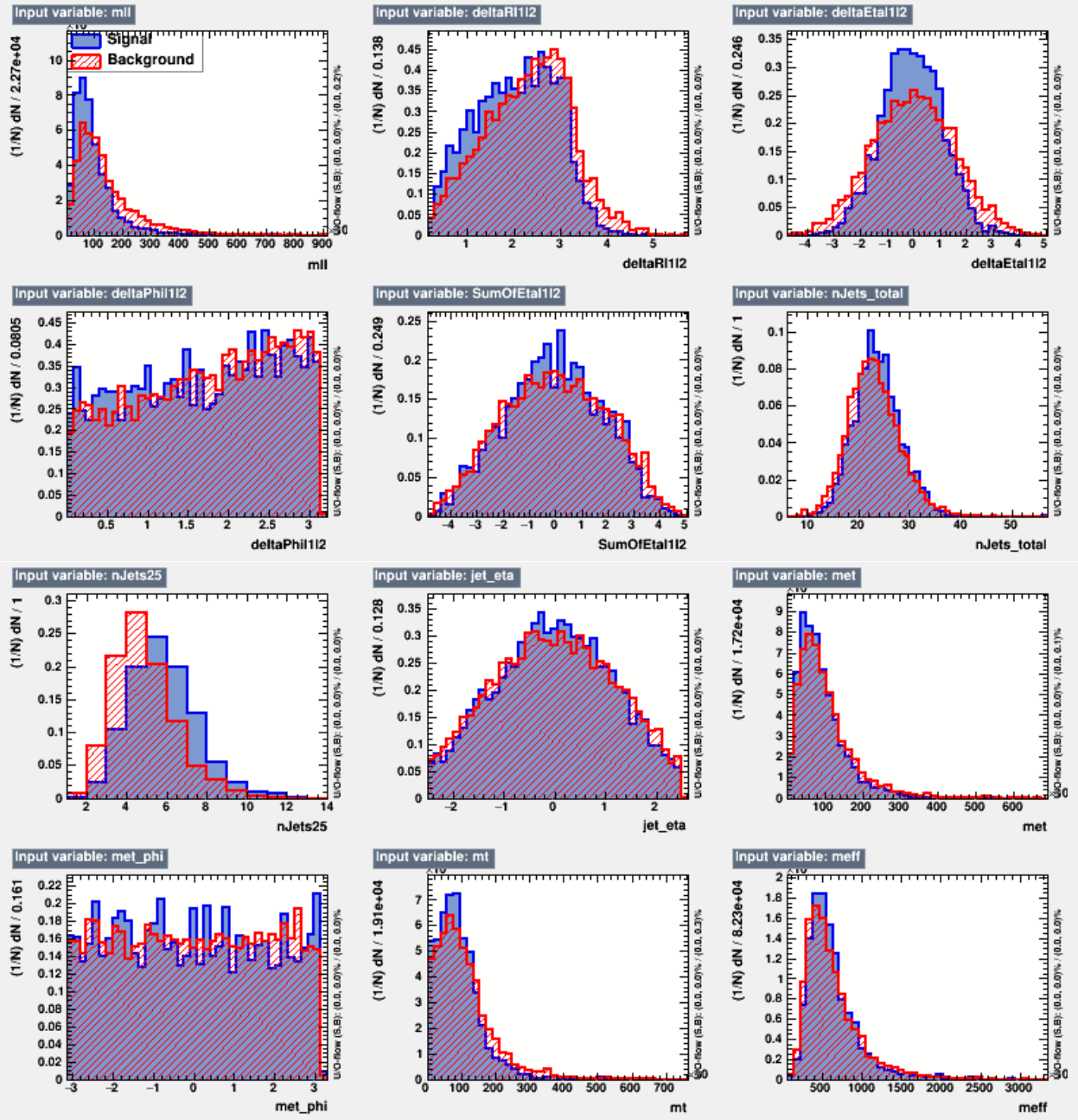
```
240     //   factory->BookMethod( TMVA::Types::kBDT, "BDTD",
241     //                        "!H:!V:NTrees=850:MinNodeSize=2.5%:MaxDepth=3:
        BoostType=AdaBoost:AdaBoostBeta=0.5:UseBaggedBoost:BaggedSampleFraction
        =0.5:SeparationType=GiniIndex:nCuts=20:VarTransform=D" );
242
243     // Allow Using Fisher discriminant in node splitting for (strong) linearly
        correlated variables
244     //     factory->BookMethod( TMVA::Types::kBDT, "BDTMitFisher",
245     //          "!H:!V:NTrees=50:MinNodeSize=2.5%:UseFisherCuts:MaxDepth=3:
        BoosType=AdaBoost:AdaBoostBeta=0.5:SpearationType=GiniIndex:nCuts=20");
246
247       // if (Use["NeuroBayes"]){ //NB
248       TString NBPreproString=":NBIndiPreproFlagByVarname=";
249
250       //Preprocessing for variables: use 14 for continuous and 18 for discrete
        variables
251
252       NBPreproString += "mll=14,";
253       //NBPreproString += "lep_chargesumFloat=18,";
254       NBPreproString += "deltaRl1l2=14,";
255       NBPreproString += "deltaEtal1l2=14,";
256       NBPreproString += "deltaPhil1l2=14,";
257       NBPreproString += "SumOfEtal1l2=14,";
258       NBPreproString += "nJets_totalFloat=18,";
259       NBPreproString += "nJets25Float=18,";
260       // NBPreproString += "nBJets25Float=18,";
261       // NBPreproString += "nJets30Float=18,";
262       // NBPreproString += "nJets40Float=18,";
263       // NBPreproString += "nJets50Float=18,";
264       // NBPreproString += "nJets60Float=18,";
265       // NBPreproString += "nJets70Float=18,";
266       // NBPreproString += "jet_pt=14,";
267       NBPreproString += "jet_eta=14,";
268       // NBPreproString += "jet_phi=14,";
269       NBPreproString += "met=14,";
270       // NBPreproString += "met_phi=14,";
271       // NBPreproString += "mt=14,";
272       NBPreproString += "meff=14,";
273       // NBPreproString += "ht=14,";
274       // NBPreproString += "lep1Ele=18,";
275       NBPreproString += "lep1Pt=14,";
276       NBPreproString += "lep1Eta=14,";
277       // NBPreproString += "lep1Phi=14,";
278       // NBPreproString += "lep1trkiso=14,";
279       // NBPreproString += "lep1caloiso=14,";
280       // NBPreproString += "lep1z0=14,";
281       // NBPreproString += "lep1z0sin=14,";
282       // NBPreproString += "lep1d0sig=14,";
283       // NBPreproString += "lep1chargeFloat=18,";
284       // NBPreproString += "l1_weight=14,";
285       NBPreproString += "mindeltaRl1jet=14,";
286       // NBPreproString += "lep2Ele=18,";
287       NBPreproString += "lep2Pt=14,";
288       NBPreproString += "lep2Eta=14,";
```

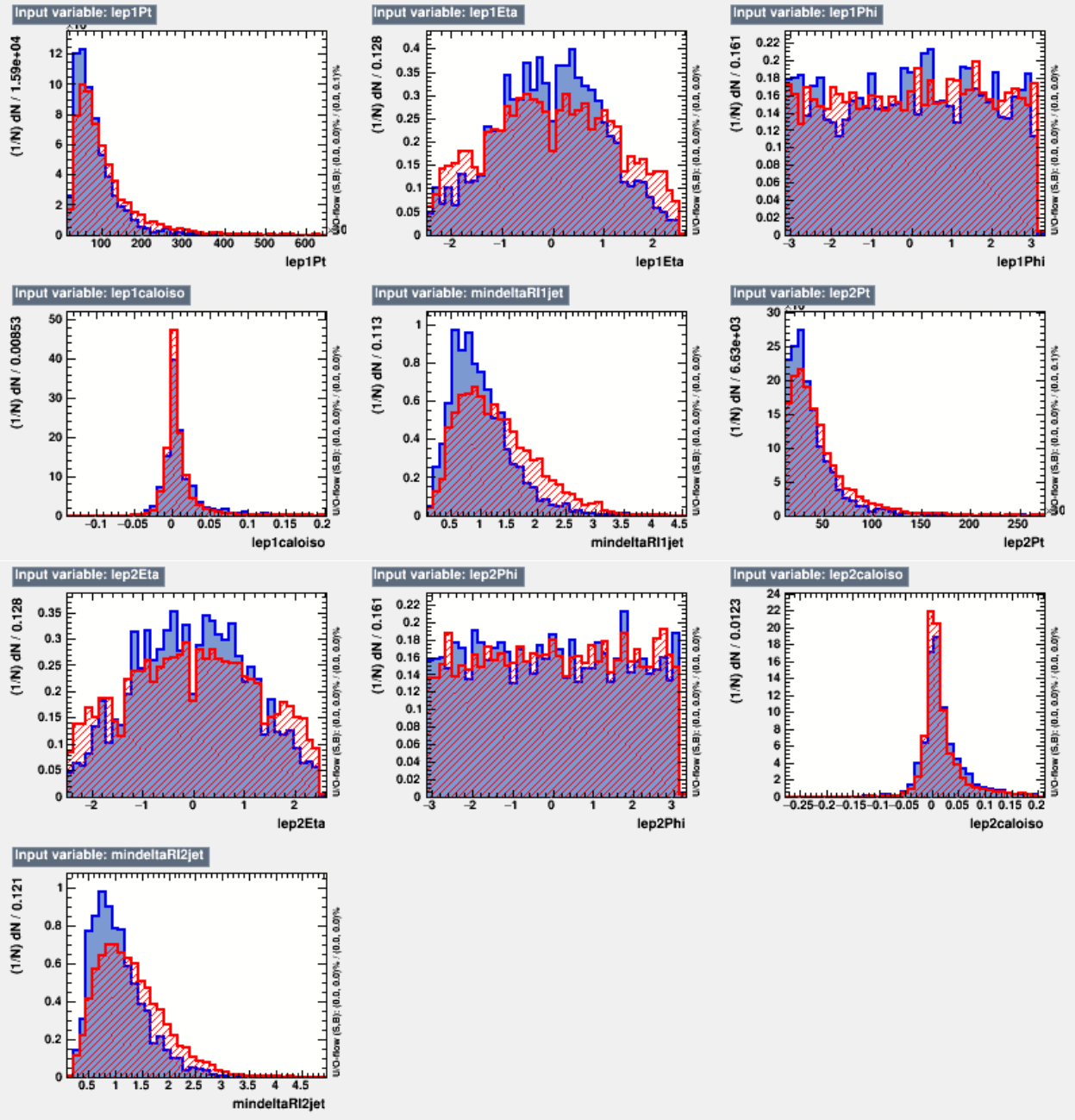
```
289       // NBPreproString += "lep2Phi=14,";
290       // NBPreproString += "lep2trkiso=14,";
291       // NBPreproString += "lep2caloiso=14,";
292       // NBPreproString += "lep2z0=14,";
293       // NBPreproString += "lep2z0sin=14,";
294       // NBPreproString += "lep2d0sig=14,";
295       //NBPreproString += "lep2chargeFloat=18,";
296       // NBPreproString += "l2_weight=14,";
297       NBPreproString += "mindeltaRl2jet=14,";
298       NBPreproString += "lep_flavourFloat=18,";
299
300
301       factory->BookMethod( TMVA::Types::kPlugins, "NeuroBayes",
302          "!H:V:Analysis:NTrainingIter=100:Preprocessing=112:ShapeTreat=INCL:
    TrainingMethod=BFGS"
303           + NBPreproString );
304
305       // parameters as in Run1 analysis
306
307
308
309
310
311
312       //CATEGORIES
313     // --- Categorised classifier
314     //TMVA::MethodCategory* mcat = 0;
315
316      //  The variable sets for the Category Classifier
317       TString theCat1Vars = "mll:deltaRl1l2:deltaEtal1l2:deltaPhil1l2:
    SumOfEtal1l2:nJets_totalFloat:nJets25Float:met:meff:lep1Pt:lep1Eta:
    mindeltaRl1jet:lep2Pt:lep2Eta:mindeltaRl2jet:jet_eta:lep_flavourFloat";
318       TString theCat2Vars = "mll:deltaRl1l2:deltaEtal1l2:deltaPhil1l2:
    SumOfEtal1l2:nJets_totalFloat:nJets25Float:met:meff:lep1Pt:lep1Eta:
    mindeltaRl1jet:lep2Pt:lep2Eta:mindeltaRl2jet:jet_eta:lep_flavourFloat";
319
320
321
322     // BDT
323     TMVA::MethodBase* BDTCat =  factory->BookMethod( TMVA::Types::kCategory, "
    BDTCat", "" );
324     mcat = dynamic_cast<TMVA::MethodCategory*>(BDTCat);
325     mcat->AddMethod("lep_chargesumFloat>0.0", theCat1Vars, TMVA::Types::kBDT, "
    Category_BDT_1",
326                                 "!H:!V:NTrees=850:MinNodeSize=2.5%:MaxDepth=3:
    BoostType=AdaBoost:AdaBoostBeta=0.5:UseBaggedBoost:BaggedSampleFraction
    =0.5:SeparationType=GiniIndex:nCuts=20" );
327     mcat->AddMethod("lep_chargesumFloat<0.0", theCat2Vars, TMVA::Types::kBDT, "
    Category_BDT_2",
328                               "!H:!V:NTrees=850:MinNodeSize=2.5%:MaxDepth=3:
    BoostType=AdaBoost:AdaBoostBeta=0.5:UseBaggedBoost:BaggedSampleFraction
    =0.5:SeparationType=GiniIndex:nCuts=20" );
329
330     //same parameters as before, two categories
```

```
331
332     //NB DOESN'T WORK WITH CATEGORY CLASSIFICATION
333     /* TMVA::MethodBase* NeuroBayesCat = factory->BookMethod( TMVA::Types::
        kCategory, "NeuroBayesCat", "" );
334     mcat = dynamic_cast<TMVA::MethodCategory*>(NeuroBayesCat);
335     mcat->AddMethod("lep_chargesumFloat>0.0", theCat1Vars, TMVA::Types::
        kPlugins, "Category_NeuroBayes_1",
336           "!H:V:Analysis:NTrainingIter=100:Preprocessing=112:ShapeTreat=INCL:
        TrainingMethod=BFGS"
337            + NBPreproString );
338     mcat->AddMethod("lep_chargesumFloat<0.0", theCat2Vars, TMVA::Types::
        kPlugins, "Category_NeuroBayes_2",
339           "!H:V:Analysis:NTrainingIter=100:Preprocessing=112:ShapeTreat=INCL:
        TrainingMethod=BFGS"
340
341           + NBPreproString );
342     */
343
344      // BDTS
345     // worst one
346     /* TMVA::MethodBase* BDTSCat =  factory->BookMethod( TMVA::Types::kCategory
        , "BDTSCat", "" );
347     mcat = dynamic_cast<TMVA::MethodCategory*>(BDTSCat);
348     mcat->AddMethod("lep_chargesum==2.0", theCat1Vars, TMVA::Types::kBDT, "
        Category_BDTS_1",
349                                 "!H:!V:NTrees=850:MinNodeSize=2.5%:MaxDepth=3:
        BoostType=AdaBoost:AdaBoostBeta=0.5:UseBaggedBoost:BaggedSampleFraction
        =0.5:SeparationType=SDivSqrtSPlusB:nCuts=20" );
350     mcat->AddMethod("lep_chargesum==-2.0", theCat2Vars, TMVA::Types::kBDT, "
        Category_BDTS_2",
351                                "!H:!V:NTrees=850:MinNodeSize=2.5%:MaxDepth=3:
        BoostType=AdaBoost:AdaBoostBeta=0.5:UseBaggedBoost:BaggedSampleFraction
        =0.5:SeparationType=SDivSqrtSplusB:nCuts=20" );
352     */
353
354     // BDTD
355     // not so helpful
356     /*TMVA::MethodBase* BDTDCat = factory->BookMethod( TMVA::Types::kCategory,
        "BDTDCat", "" );
357     mcat = dynamic_cast<TMVA::MethodCategory*>(BDTDCat);
358     mcat->AddMethod("lep_chargesum==2.0", theCat1Vars, TMVA::Types::kBDT, "
        Category_BDTD_1",
359                   "!H:!V:NTrees=850:MinNodeSize=2.5%:MaxDepth=3:BoostType=
        AdaBoost:AdaBoostBeta=0.5:UseBaggedBoost:BaggedSampleFraction=0.5:
        SeparationType=GiniIndex:nCuts=20:VarTransform=D" );
360     mcat->AddMethod("lep_chargesum==-2.0", theCat2Vars, TMVA::Types::kBDT, "
        Category_BDTS_2",
361                                "!H:!V:NTrees=850:MinNodeSize=2.5%:MaxDepth=3:
        BoostType=AdaBoost:AdaBoostBeta=0.5:UseBaggedBoost:BaggedSampleFraction
        =0.5:SeparationType=GiniIndex:nCuts=20:VarTransform=D" );
362     */
363
364     // BDTM
365     // best one for overtraining and separation!
```

```
366     /*TMVA::MethodBase* BDTMCat =  factory->BookMethod( TMVA::Types::kCategory,
          "BDTMCat", "" );
367     mcat = dynamic_cast<TMVA::MethodCategory*>(BDTMCat);
368     mcat->AddMethod("lep_chargesum==2.0", theCat1Vars, TMVA::Types::kBDT, "
        Category_BDTM_1",
369                                 "!H:!V:NTrees=850:MinNodeSize=2.5%:MaxDepth=3:
        BoostType=AdaBoost:AdaBoostBeta=0.5:UseBaggedBoost:BaggedSampleFraction
        =0.5:SeparationType=MisClassificationError:nCuts=20" );
370     mcat->AddMethod("lep_chargesum==-2.0", theCat2Vars, TMVA::Types::kBDT, "
        Category_BDTM_2",
371                                 "!H:!V:NTrees=850:MinNodeSize=2.5%:MaxDepth=3:
        BoostType=AdaBoost:AdaBoostBeta=0.5:UseBaggedBoost:BaggedSampleFraction
        =0.5:SeparationType=MisClassificationError:nCuts=20" );
372     */
373
374     // ---- Now you can tell the factory to train, test, and evaluate the MVAs
375
376     // Train MVAs using the set of training events
377     factory->TrainAllMethods();
378
379     // ---- Evaluate all MVAs using the set of test events
380     factory->TestAllMethods();
381
382     // ------ Evaluate and compare performance of all configured MVAs
383     factory->EvaluateAllMethods();
384
385     // --------------------------------------------------------------------
386
387     // Save the output
388     outputFile->Close();
389
390     std::cout << "==> Wrote root file: " << outputFile->GetName() << std::endl;
391     std::cout << "==> TMVAClassificationCategory is done!" << std::endl;
392
393     // Clean up
394     delete factory;
395
396     // Launch the GUI for the root macros
397     if (!gROOT->IsBatch()) TMVAGui( outfileName );
398 }
```

Appendix 3: Distribution of 22 Variables Used for the Training Against $t\bar{t}W$




Input variable: lep1Pt
(1/N) dN / 1.59e+04
lep1Pt
Input variable: lep1Eta
(1/N) dN / 0.128
lep1Eta
Input variable: lep1Phi
(1/N) dN / 0.161
lep1Phi
Input variable: lep1caloiso
(1/N) dN / 0.00853
lep1caloiso
Input variable: mindeltaRl1jet
(1/N) dN / 0.113
mindeltaRl1jet
Input variable: lep2Pt
(1/N) dN / 6.63e+03
lep2Pt
Input variable: lep2Eta
(1/N) dN / 0.128
lep2Eta
Input variable: lep2Phi
(1/N) dN / 0.161
lep2Phi
Input variable: lep2caloiso
(1/N) dN / 0.0123
lep2caloiso
Input variable: mindeltaRl2jet
(1/N) dN / 0.121
mindeltaRl2jet
U/O-flow (S,B): (0.0, 0.0)% / (0.0, 0.0)%

Appendix 4: Distribution of 9 Variables Used for the Training Against $t\bar{t}$