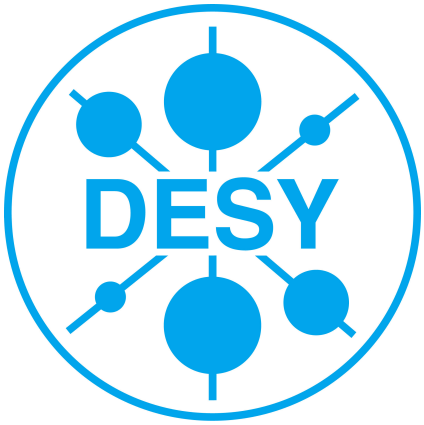


Optimising Variable Selection for Machine Learning Analysis in ATLAS ttH Search

Sitong An*, University of Cambridge, U.K.

Supervisors: Dr. Paul Glaysher, Dr. Judith Katzy

DESY ATLAS Group

September 7, 2017

Abstract

Variable selection is a non-trivial problem in the Machine Learning community, and even less research is done in the context of High Energy Physics. Methods of variable selection for training of Machine Learning algorithms are presented for the ATLAS ttH search. Focus was placed on iterative feature removal procedures, addressing information redundancy among training variables. A new optimised variable list for the ttH classification BDT was established. A generic wrapper algorithm is provided that can easily be applied to other HEP problems.

*sa747@cam.ac.uk

Contents

1. Introduction

Since the discovery of Higgs boson in 2012 at the Large Hadron Collider (LHC), its properties are being studied to ever greater precision. Measuring the the production of Higgs boson in association with pairs of top quark (ttH) is one of the primary goals of the LHC over the next decade. An observation of ttH (figure 1) provides a direct measurement of the top quark Yukawa coupling and probes the Standard Model (SM).

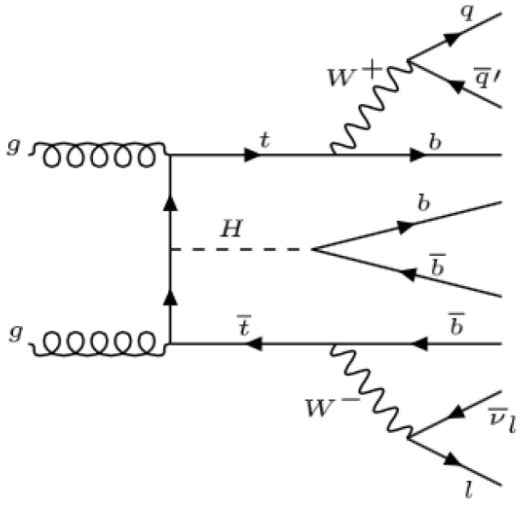


Figure 1: Feynman graph of ttH production with $H \to b\bar{b}$ decay.

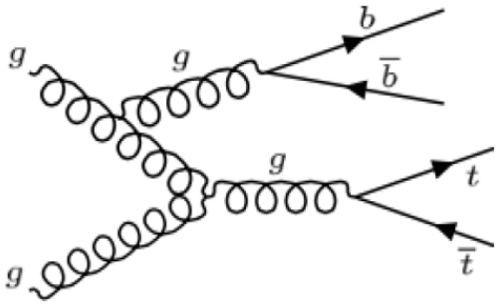


Figure 2: Feynman graph of top pair production in association with b, which is the main irreducible background to ttH.

The Feynman graph for the signal event in ttH search is shown on the left in figure 1. The Higgs boson decays predominantly to two bottom quarks (bb) 58% of the time, yielding signatures that are similar to tt + b-jets (figure 2), which is the main irreducible background in this investigation. The ttH process has not yet been observed yet at the LHC.

The Standard Model predicts that only about one out of every 100 Higgs bosons at the LHC is produced via ttH [2] and the dominant $t\bar{t}$ background is about 3 orders of magnitude more common [3]. The similar kinematic signatures of signal and background events make it difficult to establish signal/background discrimination with cuts on a single variable, thus motivating the use of multivariate techniques to distinguish between these processes.

For this search, the most popular multivariate analysis technique in use is Boosted Decision Trees (BDT). Both Neural Network (NN) and Support Vector Machine (SVM) should theoretically perform better than BDT, but in practice BDT usually outperform them[4]. This is because both NN and SVM need fine-tuning (optimal architecture for NN, optimal kernel for SVM). In comparison, BDT is very effective and relatively

simple, with less hyperparameters to adjust, hence its popularity in the HEP community.

1.1. Motivation

Theoretically, with more variables, we have more information about the event and the BDT should gain greater separating power. However, in practice we want to reduce the number of variables used to train the BDT while retaining its performance as much as possible. This is mainly due to the fact that we use Monte Carlo simulated data to train these Classification BDTs. As Monte Carlo simulations are not always perfectly reliable for all variables, each variable used in the training needs to be checked for validity and bias towards different Monte Carlo generators. With less variables used, less human effort is needed on these checks and systematic error can be potentially reduced.

Minimising the number of training variables also has the potential benefit of removing information redundancy and reducing training time. Moreover, by looking at which variables are ranked as the most important by the variable search algorithms, we could potentially gain greater insight into the ttH search from their physical significance.

This motivates us to produce a ranking of the variables and select the most important of them in the training[1].

1.2. Current Analysis Technique

The ttH analysis is split by the decay mode of the Higgs boson and the top quarks. Only the most sensitive channel is considered here, namely H→bb, tt→1 charged lepton + jets.

We consider only Higgs bosons that decay via the dominant decay mode H→bb. The signal has eight final state objects, two intermediate W bosons, two top quarks. In particular a dijet resonance at the Higgs boson mass is of importance in separating signal and background. The neutrino is not directly detected, but instead inferred from applying conservation of momentum to the objects in the detector.

The signal extraction is aided by sub-categorising [1] the analysis further into signal regions on the basis of the number of jets and the number of b-tagged jets. The most sensitive region is defined by >=6jets and 3b tags. This represents a compromise between signal/background purity and sufficient data statistics .

A Classification BDT is trained independently for each signal region. Training is performed on kinematic variables such as invariant mass or ΔR quantities, some of which

[1]It is worth noting that this is different from the concept of feature extraction in machine learning, as the goal is not to avoid the curse of dimensionality. Popular methods for feature extraction, like Principal Component Analysis, are not applicable for our purpose.

are calculated by a separate, prior Reconstruction BDT that matches final state objects to either Higgs or top decays. This additional reconstruction step is only up to 43% efficient but substantially aids the final discrimination power. Only the final Classification BDT is discussed here.

2. Multivariable Analysis

2.1. Boosted Decision Trees

Boosted decision trees (BDT) is the Machine Learning technique widely used in high energy and ttH search. Therefore we also adopt this technique in this investigation.

A decision tree is a binary tree network for data categorisation, with structure shown in figure 3. It starts from a root node and grows successive layers formed by binary nodes. At each node, a cut on a particular variable is applied to split the data. When a new node is generated, the variable and the cut value that can achieve maximal separation between signal and background on this node is selected to make the cut [4]. As the decision tree grows in layer, the phase space is then split into multiple signal-rich and background-rich regions. Each event will start from the root node and go down the decision tree, eventually being classified as signal or background depending on which node in the bottom layers (also called "leaf") it reaches.

Some of the hyperparameters that define the structure of a decision tree can be found in table 1.

As a single decision tree is limited in its separating power (i.e. a "weak classifier"), multiple (usually in the order of hundreds) decision trees are trained together to form a "forest". When each new tree is generated, more emphasis is given to previously misclassified events. This process of assembling multiple weak classifiers into a strong classifier is called "boosting". In the end the each decision tree will classify an event as signal/background and the weighted average of the individual tree classification is given as a final BDT score (as shown in figure 5). This BDT score shows the likelihood of an event being signal or background.

2.2. Performance Benchmark

In order to evaluate the performance of each set of variables we use, we use the the Receiver Operating Characteristic (ROC) curve, which is a plot (figure 4) of background rejection vis-a-vis signal efficiency for cuts made on BDT output score (figure 5). In this analysis, signal efficiency is defined as the proportion of signal retained for a particular cut on the BDT score. Background rejection is defined as $1 -$ background efficiency, or equivalently the proportion of background rejected by that same cut. If a machine

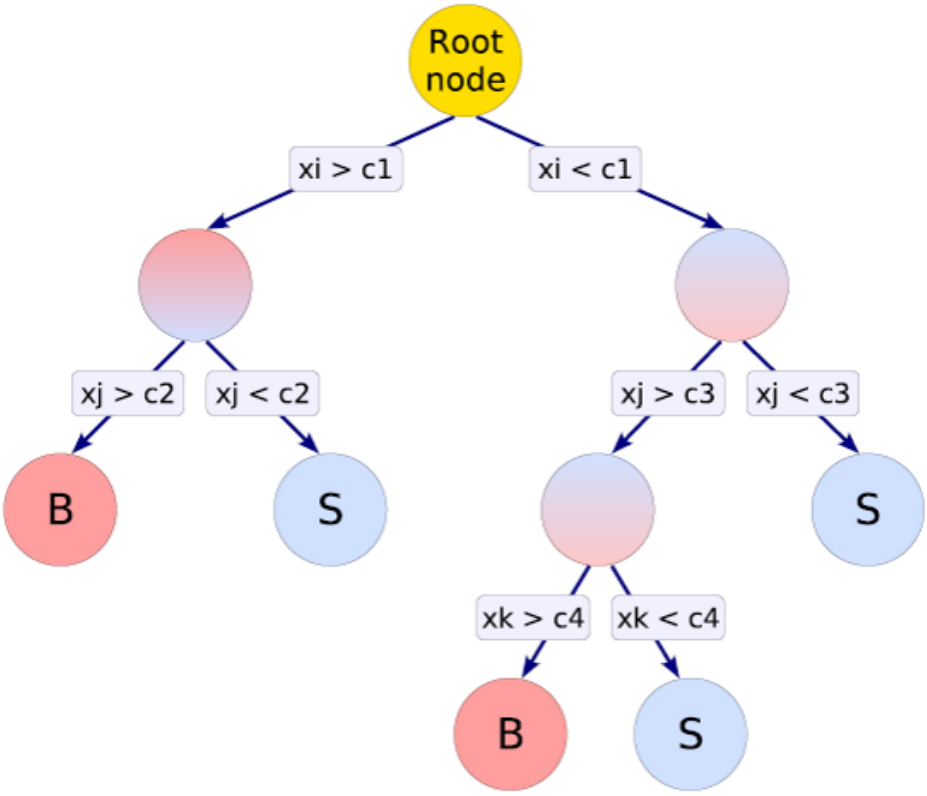


Figure 3: Schematic view of a decision tree (source: TMVA user manual [4]). xi, xj and xk represent variables in each node and c1,c2 and c3 are cuts made on each node. As data goes down the tree structure, the purity of background(B)/signal(S) events increases. Which node in the bottom layer ("leaf") the event eventually reaches decides whether it is classified as signal or background by this decision tree.

learning method, such as BDT, performs better, then for a particular signal efficiency there will be higher background rejection. The curve then will convex more.

ROC shows how much background is rejected at each possible point of signal efficiency. Since we are using full spectrum of the BDT output score in the analysis in ttH search instead of making a single cut on it, the total area under the ROC curve (AUROC) provides a better representation of the separating performance of the BDT model trained on a particular set of variables, rather than a certain point on the ROC curve. This is the primary reason why we decided to use AUROC to benchmark the variable list performance in this investigation, but other performance benchmarks are possible.

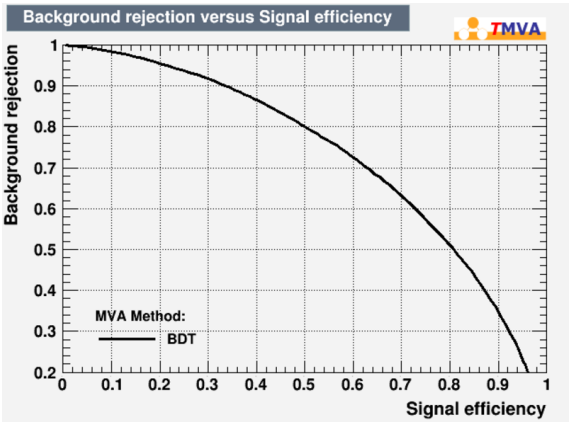


Figure 4: Receiver Operating Charateristic (ROC) curve, generated by TMVA framework. It plots background rejection against signal efficiency for different cuts that can be made on the BDT output score. As we aim to have maximal signal efficiency with minimal background contamination, the higher (more convex) the curve the better.

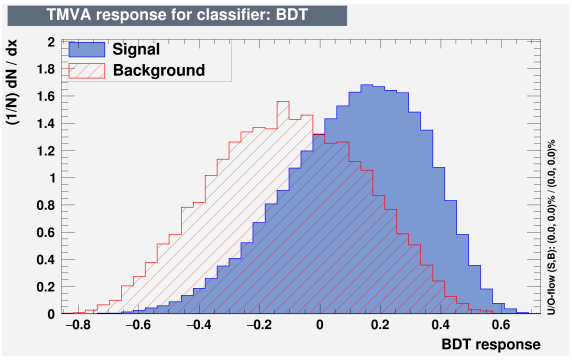


Figure 5: BDT output score. If a cut of a certain value is made on this score, some proportion of all signal events will be selected together with some background contamination. The ratio of selected signal events to total signal events for this cut is called "signal efficiency", and the ratio of rejected background events to total background events is defined as "background rejection".

2.3. Overfitting[2]

In general overfitting is a major concern for BDT training. When overfitting ocurrs the trained BDT starts to describe the statistical fluctuations in the data set used for training and lose generality in the model, resulting in a seemingly high performance that does not hold if data sets are changed. In this investigation we have guarded against this problem by using half of the available data set for training and the other half for testing, as per the standard procedure.

In this investigation we have the additional risk of possible overfitting of the variable search methods to the particular original variable list used. To verify the generality of the variable search methods we have conducted validation test on an alternative variable list, as elaborated in section 9.

3. BDT and Computing Setup

We used the following hyperparameters for the BDT training in this investigation:

[2]Also called "Overtraining".

Table 1: Hyperparameters

Hyperparameter	Value	Explanation
Nvariables	21	Number of variables used to train the BDT
Ntrees	400	Number of decision trees in the forest
MinNodeSize	4%	Lower bound on the amount of events that pass through each node, as a proportion of total events
MaxDepth	5	Upper bound on the layers of a single decision tree
nCuts	80	Number of test points across the range of a variable when testing variables and cut values for a new node
BoostType	AdaBoost	Define algorithm to assign weights to individual tree

Among the hyperparameters, MinNodeSize and MaxDepth are features to prevent a single decision tree from overfitting on the training data.

The data sets used in this investigation are "ttH aMC@NloPythia8" and "ttbar PowhegPythia8", both with roughly 250,000 reconstructed Monte Carlo training events.

Only events with at least 6 jets and among which at least 3 are tagged as b-jets are selected and included in the data sets. The b-tagging score is cut at 60%, which is a very tight selection. An additional selection is performed to keep events with Pt_jets $>$ 25 GeV.

The ICHEP 2016 Summer variable list is used primarily for the investigation of this project. This list is available in appendix A. These variables that we are using are derived high-level quantities, derived on the basis of significant prior knowledge of the underlying physics. An alternative variable list of the same size is used for validation, which can be looked up in appendix B.

Toolkit for Multivariable Data Analysis with ROOT (TMVA) Framework is used for the training and testing of BDTs.

When possible, algorithms are run in parallel on DESY BIRD cluster to speed up the search. Each job takes about 3 minutes to complete with our settings and data sets, and the BIRD Cluster allows a maximum of 300 jobs to be run in parallel at the same time. For example, an iterative removal algorithm (explained in details in section 7) on 21 variables will take about 250 jobs in total to run. If run in serial this would take more than 12 hours, but if parallelised it only takes about 3 hours. The advantage of

parallelisation is greater with more variables.

While training, we used half of our data as the training set and the other half for the test to mitigate overfitting to statistical fluctuations.

We recognise the risks to systematically bias the training due to the particular model used in the Monte Carlo generator chosen. This certainly necessitates further studies, but due to time constraint it is not addressed in this investigation.

4. Case Study: Visual Inspection by Signal/Background Separation

4.1. Motivation

The distribution of both signal and background data for each variable can be plotted, such as shown here in figure 6, for a visual inspection of the signal-to-background separating power of each variable. Intuitively, the variable that gives greater separation between signal and background, such as the one shown on the left of figure 6, should be more important and useful than variables with little separating power, such as the one on the right.

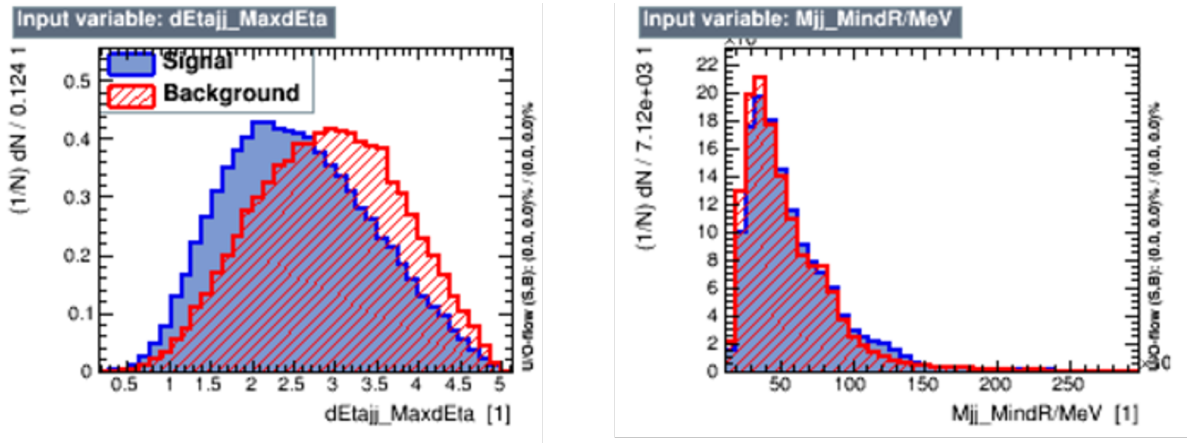


Figure 6: Signal-background separation plots for variables dEtajj_MaxdEta(left) and Mjj_MindR(right). By simple visual inspection, the variable on the left should provide greater signal and background separation than the one on the right.

4.2. Method

To test this idea, a set of 8 "useful" variables with greater apparent separating power is selected by visual inspection. This list and the complementary set of 13 "less useful"

variables is given in table 2. Their signal-background separation plots are shown in figure 7 and figure 8 respectively.

Table 2: Variable sets selected according to apparent separating power

Variable Name
dEtajj_MaxdEta
NHiggs_30
Njet_pt40
Centrality_all
semilepMVAreco_bbhiggs_dR
dRbb_avg
dRbb_MaxPt
pT_jet5

(a) List of "useful" variable

Variable Name
HT_jets
H1_all
bb_MindR
Mbj_MaxPt
Aplan_jets
Mjj_MindR
semilepMVAreco_higgsttbar_withH_dR
semilepMVAreco_higgsbleptop_mass
semilepMVAreco_higgs_mass
semilepMVAreco_higgsleptop_dR
semilepMVAreco_higgsbhadtop_withH_dR
semilepMVAreco_BDT_withH_output
semilepMVAreco_higgsbhadtop_withH_dR

(b) List of "less useful" variable

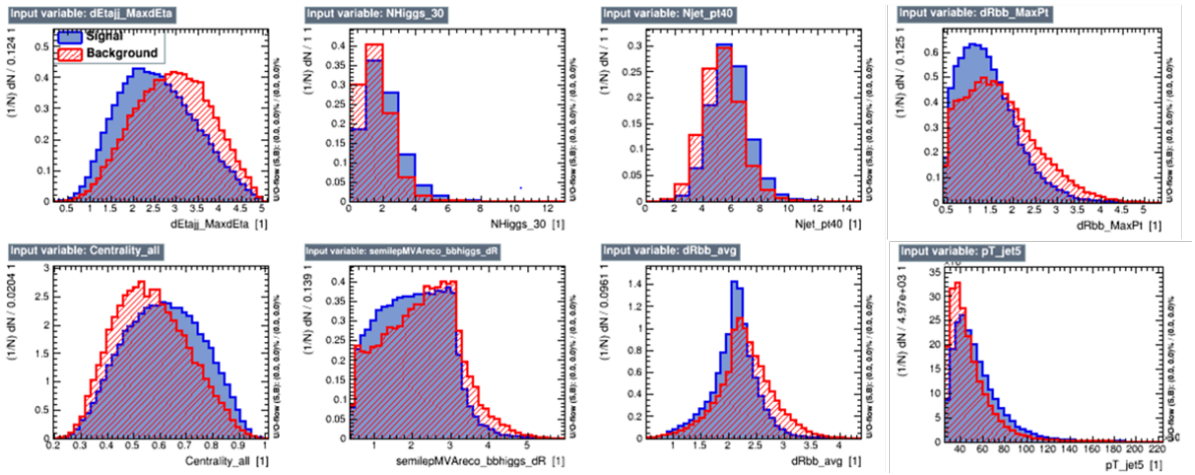


Figure 7: Signal-background separation plots for variables with greater apparent separating power, selected by visual inspection.

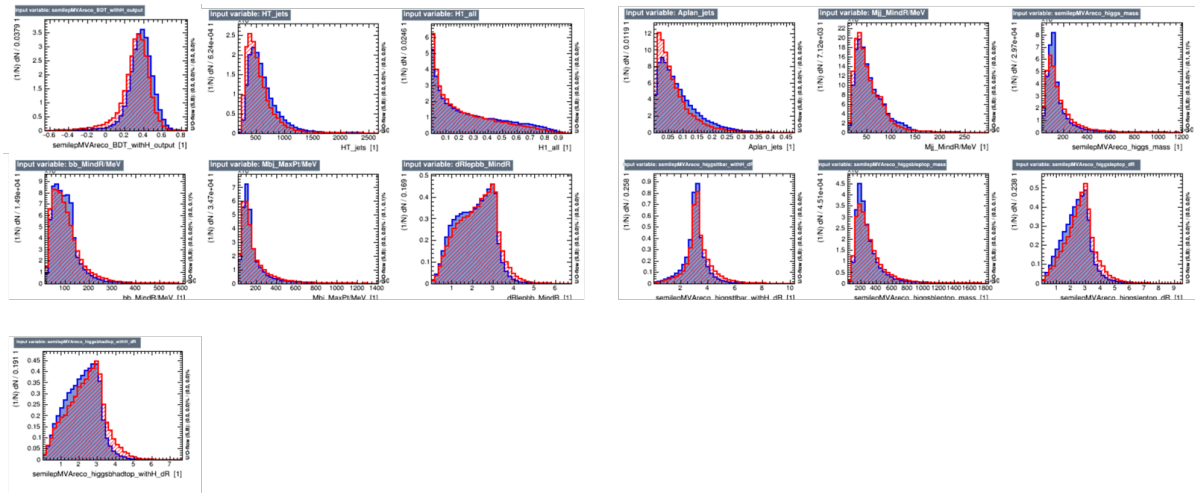


Figure 8: Signal-background separation plots for variables with apparently less separating power, selected by visual inspection.

4.3. Results

Both the "Useful" and "Less useful" sets of variables are tested, with results summarised in table 3.

Table 3: Results for the two variable sets selected by visual inspection of apparent separating power

	All 21 Variables	8 "Useful" Variables	13 "Useless" Variables
AUROC	76.05	72.97	73.85

With "Useful" set performing worse than "Useless" set, this surprising result seems to imply that the number of variables used in BDT training is more important than the choices of which variables to use.

The result makes the case that this method of visual inspection of the signal-background separation does not take into account the correlation or shared information between variables. If two or more variables are correlated, their apparently strong separating power may have the same physical origin. If they both are used in the training of the BDT, there will be much less extra information about the underlying physical process for the algorithm to exploit.

Another possible explanation for this is that, despite the seemingly significant difference between the signal-background separating power of different variables, they are equally weak classifier as far as the BDT training algorithm is concerned. Due to the inherent

logic of the Boosted Decision Trees algorithm, number of weak classifiers available could be more important than the strength of each individual weak classifier in the process of aggregating an assembly of weak classifiers into a single strong classifier.

While this is an interesting result, it is also slightly worrying, as it casts doubts on the quality of the original list of variables as recommended by ICHEP Summer 2016. This recommendation was made based on our theoretical understanding of the variables and their apparent stand-alone separating power as seen in the signal-background separation plots, but this result implies that this is not the optimal method for selecting the best variables for training the BDT. What if there are unused variables that actually perform better for training the BDT for ttH analysis, but just did not look good to us?

5. Case Study: Correlation Based Selection

5.1. Motivation

Linear correlation coefficient is a simple measure of shared information content between variables. In order to minimise information redundancy, a method to iteratively remove variables according to their linear correlation coefficient is investigated.

5.2. Method

The TMVA framework readily provides a matrix of linear correlation coefficient for variables used in the BDT training, as seen in figure 9. The method starts with all 21 variables. At each step, one variable from the pair with the highest linear correlation coefficient (absolute value) is removed and the performance of the subset is tested. This process is iterated until only one variable is left. Among the two variables in a pair, if one has had more pair partners removed in previous iterations, that variable is kept and its current pair partner discarded. After each variable is removed in each iteration, the BDT is retrained again and the resulting AUROC calculated.

5.3. Result

In figure 10, the plot in red represents the performance of variable lists as produced by the previously described correlation-based selection method, whereas the plot in blue represents the average performance of 28 randomly selected variable lists, which is used here as a baseline for comparison. The error bar on the blue plot represents the standard deviation of the random variable lists performance. Note though that the performance of randomly selected variable lists does not necessarily have a Gaussian distribution, so the error bar here is only an indication of the spread.

As shown in the plot, correlation-based selection method can in most cases produce variable lists that perform better than those generated randomly. Therefore, it is shown

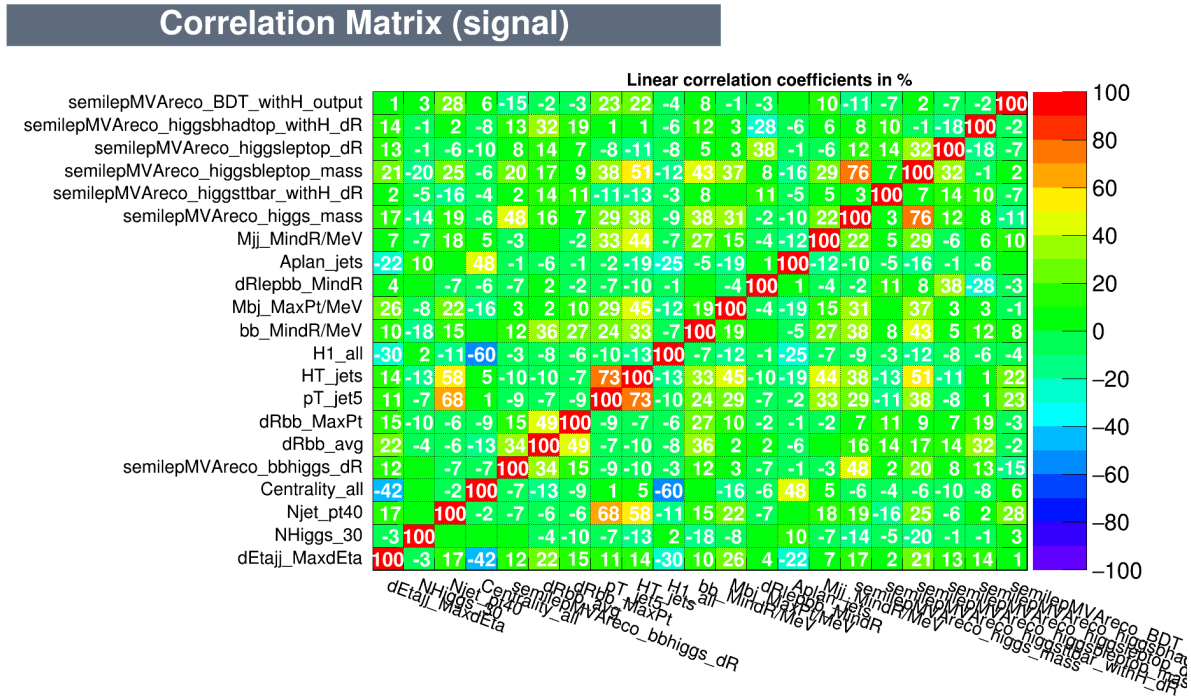


Figure 9: A matrix of linear correlation coefficient of all 21 variables, generated by TMVA framework.

here that correlation can be used as a measure of shared, or redundant, information in the variables, and correlation-based selection method could be a good way at reducing the information redundancy.

The idea of correlation-based selection could potentially be expanded as part of a future investigation into variable selection for HEP Machine Learning analysis, as the method adopted here is relatively elementary and only shown here as a proof of concept. There could be groups of more than two variables strongly correlated with one another, and the correlation relationship between variables could be non-linear. For example, instead of focusing on the pair of variables with the highest linear correlation coefficient, weaker but still significant correlation relations with other variables should also be accounted for in a better method. A possible topic for future investigation could be devising an algorithm that better utilises all the information provided by the linear correlation coefficient matrix. Other statistical measure of variable interdependence, such as mutual information or other types of correlation coefficients, could also be investigated.

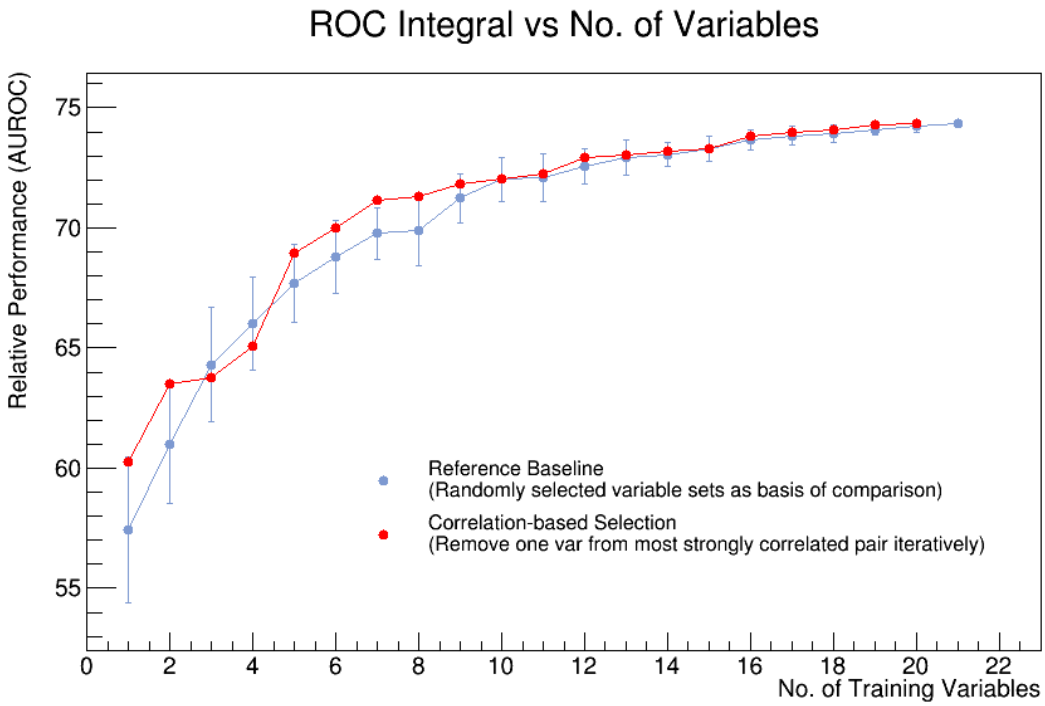


Figure 10: Performance for correlation-based selection method. Each data point on the red curve represents the ROC Integral of the variable list with a particular number of variables produced by this method. The method starts with all 21 variables and goes through an iterative subtractive process. Note the plateau of the plot formed at high number of training variables. To have a better trade-off between performance and number of variables used, choose to use the number of variables that corresponds to the start of this plateau.

6. Case Study: TMVA Ranking

6.1. Motivation

A preliminary ranking of the variable is produced by the TMVA Framework after each training. In the case of BDT, the ranking is produced "by counting how often the variables are used to split decision tree nodes, and by weighting each split occurrence by the separation gain-squared it has achieved and by the number of events in the node", according to the TMVA User Manual[4].

This ranking is known to be unstable and sub-optimal, but widely used within the community. It is thus studied in this project as a standard method.

6.2. Method

All 21 variables are used to train the BDT once with TMVA, and the TMVA ranking is acquired from the TMVA output (method specific ranking, as seen in figure 11) produced by this training. For N = 1 to 21, the top N variables according to this ranking are used in the training, and their performance is plotted.

```
--- Factory                  :
--- Factory                  : Ranking input variables (method specific)...
--- BDT                      : Ranking result (top variable is best ranked)
--- BDT                      : ------------------------------------------------------------------------
--- BDT                      : Rank : Variable                                : Variable Importance
--- BDT                      : ------------------------------------------------------------------------
--- BDT                      :    1 : dRbb_avg                                : 1.007e-01
--- BDT                      :    2 : semilepMVAreco_BDT_withH_output         : 6.646e-02
--- BDT                      :    3 : dEtajj_MaxdEta                          : 5.787e-02
--- BDT                      :    4 : HT_jets                                 : 5.725e-02
--- BDT                      :    5 : NHiggs_30                               : 5.425e-02
--- BDT                      :    6 : Centrality_all                          : 5.069e-02
--- BDT                      :    7 : H1_all                                  : 4.887e-02
--- BDT                      :    8 : Mjj_MindR                               : 4.746e-02
--- BDT                      :    9 : pT_jet5                                 : 4.729e-02
--- BDT                      :   10 : dRbb_MaxPt                              : 4.692e-02
--- BDT                      :   11 : semilepMVAreco_higgsbhadtop_withH_dR    : 4.626e-02
--- BDT                      :   12 : Mbb_MindR                               : 4.568e-02
--- BDT                      :   13 : Aplan_jets                              : 4.232e-02
--- BDT                      :   14 : semilepMVAreco_higgsttbar_withH_dR      : 4.111e-02
--- BDT                      :   15 : semilepMVAreco_bbhiggs_dR               : 3.948e-02
--- BDT                      :   16 : dRlepbb_MindR                           : 3.867e-02
--- BDT                      :   17 : semilepMVAreco_higgs_mass               : 3.719e-02
--- BDT                      :   18 : semilepMVAreco_higgsbleptop_mass        : 3.585e-02
--- BDT                      :   19 : Mbj_MaxPt                               : 3.320e-02
--- BDT                      :   20 : semilepMVAreco_higgsleptop_dR           : 3.277e-02
--- BDT                      :   21 : Njet_pt40                               : 2.971e-02
--- BDT                      : ------------------------------------------------------------------------
--- Factory                  :
```

Figure 11: Sections of TMVA Training output that shows TMVA Method Specific Ranking

6.3. Result

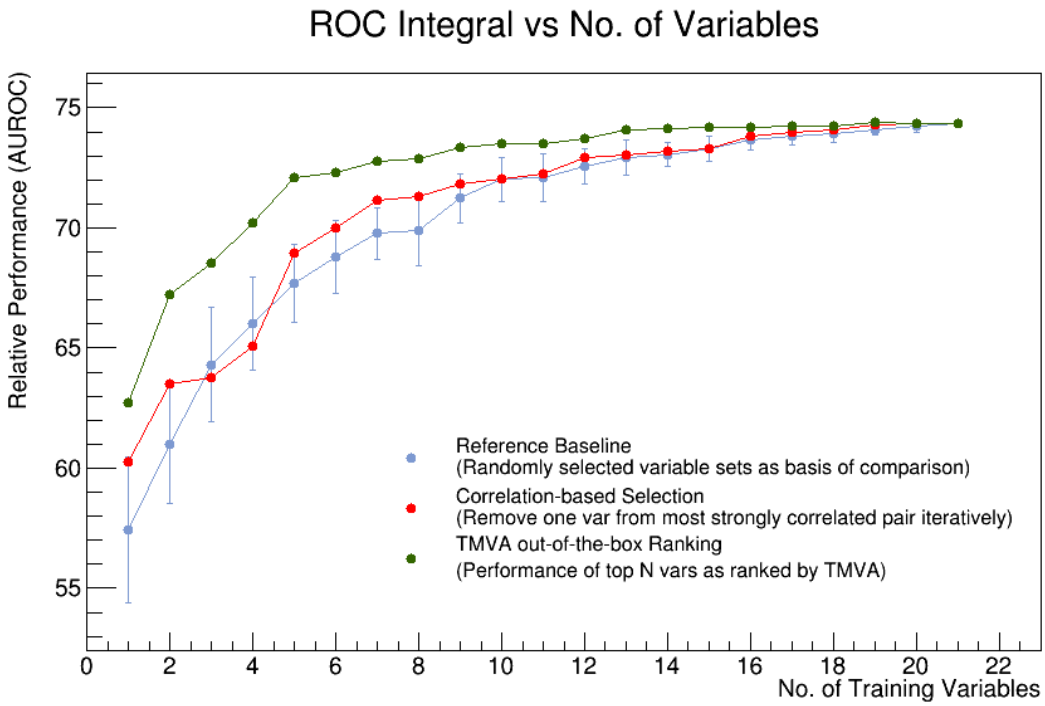


Figure 12: Performance for TMVA out-of-the-box ranking. The dark green plot represents the performance of variable sets chosen based on the TMVA ranking. As before, the reference baseline refers to the performance of randomly selected variable sets.

The result for TMVA Ranking (plot in dark green) is presented in figure 12. It performs much better than the reference baseline (plot in blue).

6.4. Derived Methods

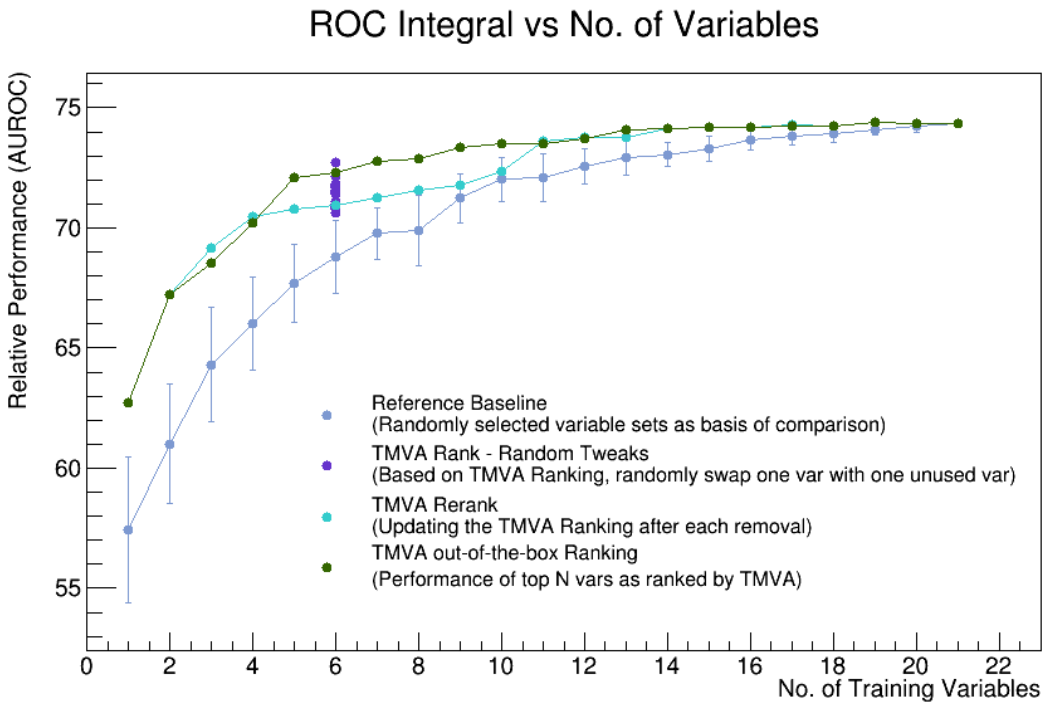


Figure 13: Performance for TMVA out-of-the-box ranking and related methods. The dark green plot represents the performance of variable sets chosen based on the TMVA ranking. The light blue plot represents the performance the the same method, but with the TMVA ranking updated after every time the lowest ranked variable is removed and the BDT is trained with the reduced variable set. The purple dots represent the performance of variable sets generated from randomly selecting one variable from the Nvar = 6 TMVA variable set and swapping it with one randomly selected unused variable. As before, the reference baseline refers to the performance of randomly selected variable sets.

6.4.1. Random Tweaking on TMVA

In order to test whether TMVA Ranking produces the optimal variable set, random tweaking is performed on the TMVA Ranking variable list with 6 variables. One of the six variables as chosen by TMVA Ranking is randomly selected and swapped with one random variable that is not used. 30 randomly tweaked variable sets in total are generated and their performance shown here as the dots in purple in figure 12. The one dot above the dark green TMVA Ranking plot demonstrates the existence of variable sets that perform better than TMVA Ranking, thus showing that TMVA Ranking is not optimal. However, since this is only produced with a probability of approximately 1 in 30, it also validates the TMVA Ranking as a decent preliminary guidance for variable selection.

6.4.2. TMVA with Reranking

As explained in section 6.1, TMVA Ranking is produced according to the frequency of each variable being selected and used in the BDT training process, weighted by its respective separation and the amount of events separated by this variable at this node. This means that correlation information is not accounted for in TMVA Ranking.

In order to improve the TMVA method and include the correlation in the ranking, it is proposed that a dynamic subtractive process could be used. The algorithm starts with all 21 variables and the original TMVA out-of-the-box ranking, removes the lowest ranked variable to produce a reduced variable set, and update the ranking according the the TMVA output from the training of this subset. This process is iterated until only one variable is left.

The result of this TMVA Reranking method is presented in figure 12 as the plot in light blue. To our surprise, it does not show an improvement over the original TMVA Ranking (in dark green). For some No. of variables, it actually performs worse. This might be a result of the inherent instability of the TMVA Ranking method.

6.5. Conclusion

TMVA Ranking is shown to be decent as a preliminary guidance for variable selection, but sub-optimal (as shown by the result of random tweaking) and unstable (since the rerank option did not perform better). In the quest for a better variable selection method, we must look beyond TMVA Ranking.

7. Case Study: Iterative Removal

7.1. Motivation

The variable selection problem can be viewed as a classical search problem in a hyperspace with extremely high dimensionality. Each dimension in this hyperspace represents a variable, and is only binary in its range (0 or 1, representing whether that variable is used in the training or not). Then our goal is transformed into a search for a maximum value of performance in this hyperspace. An exhaustive search in this hyperspace is impractical as the number of possible configurations scale with number of possible variables as 2^n. For 21 variables this evaluates to slightly more than two million. We could nevertheless approach this problem with classical algorithms well established in the field of computer science.

A hill-climbing algorithm solves the search problem by always going in the direction with the highest gradient. It can be naïve in the sense that it might get stuck in a local

maximum instead of a global one, but it is still a valuable and intuitive method and is thus investigated here in our project.

7.2. Method

The algorithm starts with all 21 variables and tests all 21 possibilities of removing each variable. Among the 21 variable lists produced this way, the one that performs the best indicates which variable, if removed, has the least impact on the performance. This variable is ranked as the least important, and this particular variable list is used to generate a new batch of variable lists by again removing each variable once. This process is iterated until only one variable is left, thus acquiring a ranking of the variables.

7.3. Result

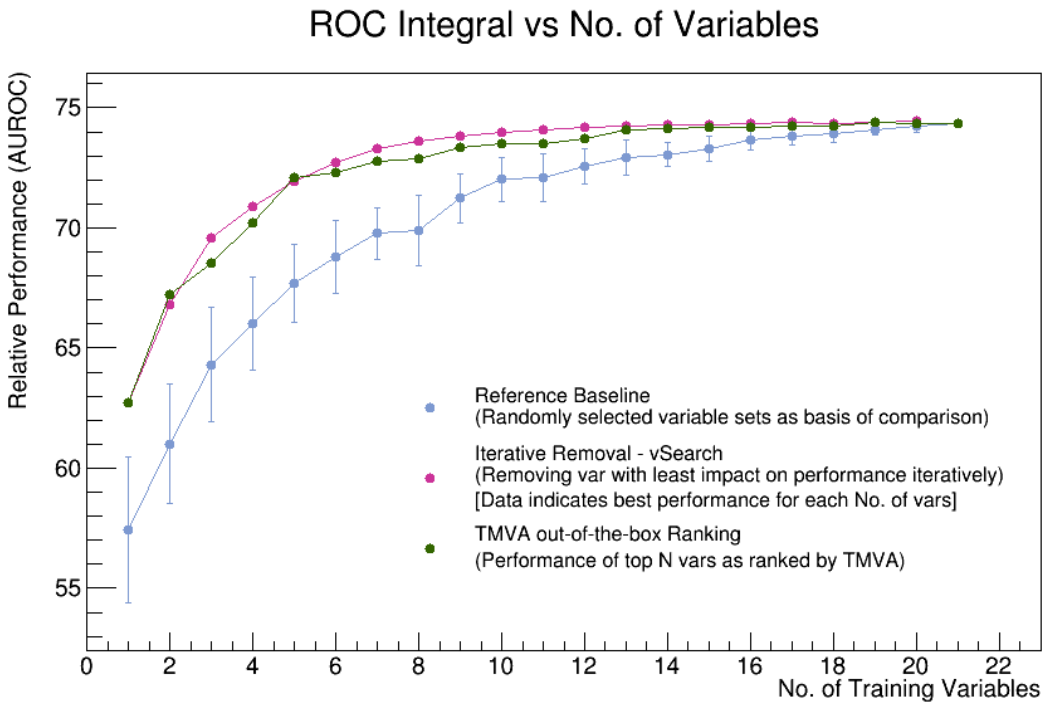


Figure 14: Performance for iterative removal algorithm is shown as the plot in pink. As before, the reference baseline refers to the performance of randomly selected variable sets. Note that the plateau shape of the plot is very pronounced, which shows that this search method is promising.

As seen in figure 14, iterative removal method performs the best for almost all number of variables. However in this case, it performs slightly worse than TMVA Ranking (in dark green) when the variable list is of the size 2 and 5. This could be attributed to the tendency for iterative removal algorithm to get stuck in local maxima, as discussed in section 7.1.

7.4. Derived Method: Beam Search

One potential method to reduce this tendency towards local maxima is to implement a beam search of a certain width $\mathcal{W}$. The idea is that instead of taking the best performing child-variable list in each iteration as in iterative removal, we could take the top $\mathcal{W}$ performing child-variable lists. It is as if the search is being conducted in a beam of fixed width, thus the name. Variable lists that perform slightly worse than the best in each iteration have the chance to survive longer in the search, thus allowing for the possibility that they might give rise to child-variable lists that perform better in subsequent iterations.

This, of course, requires several times more computational resources than iterative removal. However, the time complexity of these two algorithms scale the same as No. of variables, and when run in parallel beam search takes only slightly longer actual time than iterative removal due to the extra overheads.

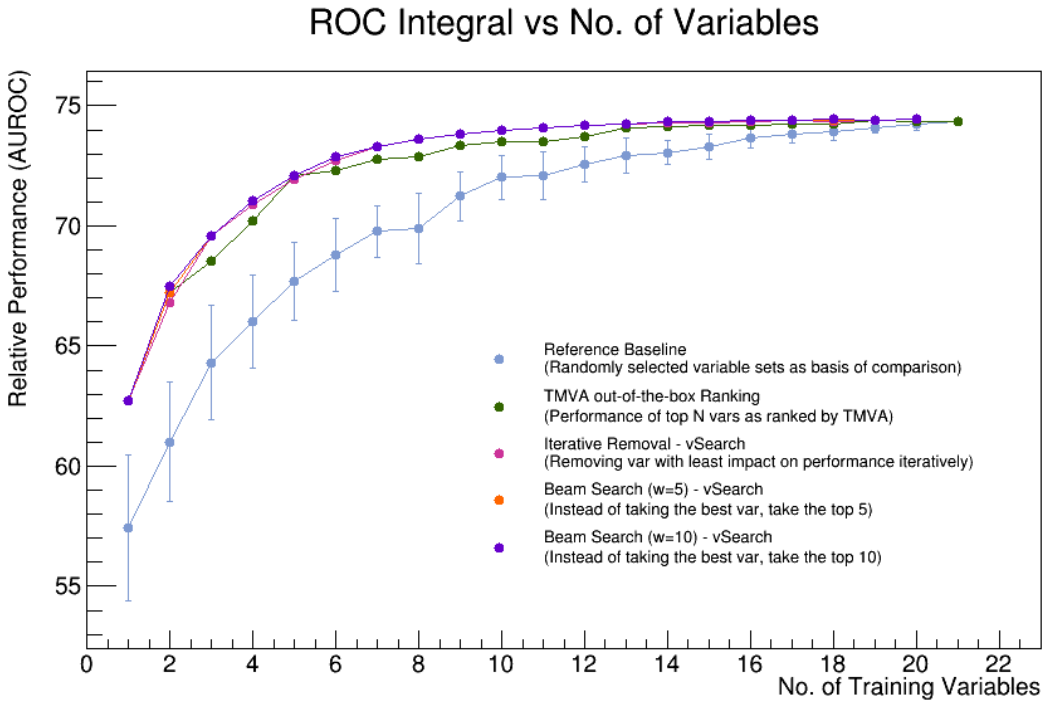


Figure 15: Performance for beam search algorithm is shown as the plot in orange ($\mathcal{W} = 5$) and blue ($\mathcal{W} = 10$). As before, the plot in pink represents the performance of iterative removal algorithm and the reference baseline in light blue refers to the performance of randomly selected variable sets.

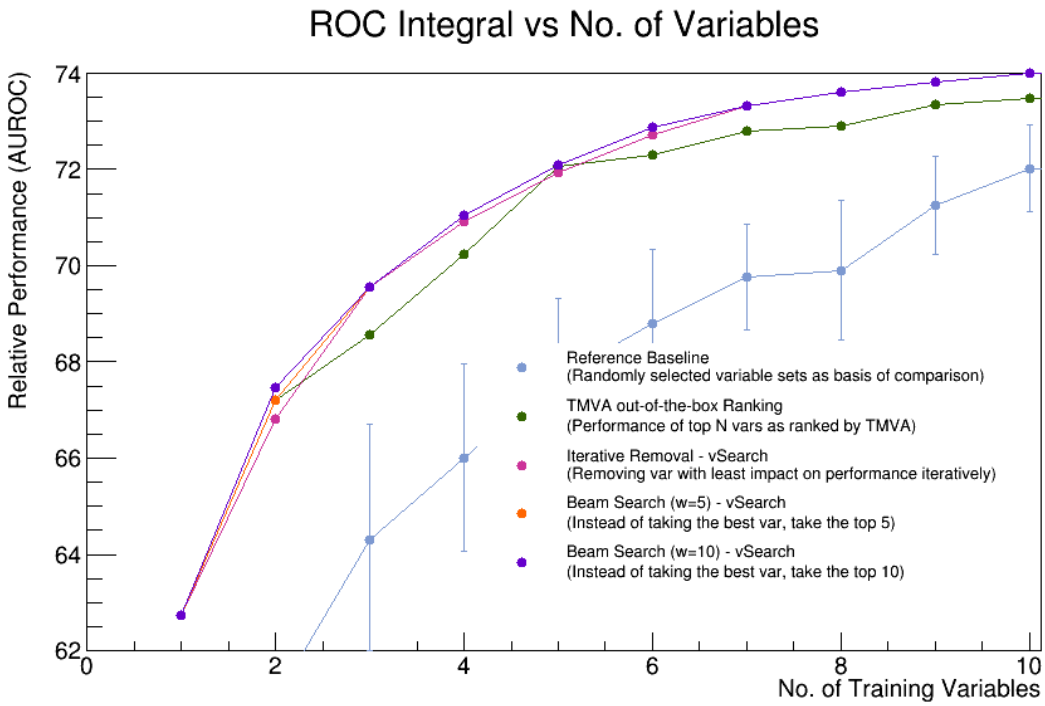


Figure 16: Same plot as figure 15, zoomed in to the range where performance improves with $\mathcal{W}$ of the beam search algorithm.

Performance of beam search algorithm is shown in figure 15. The Beam Search algorithm does not show a significant improvement from the iterative removal algorithm, but it is guaranteed to perform at least as well as iterative removal. At places where iterative removal shows sub-optimal behaviour, such as shown in figure 16, it can be seen clearly that the better performance can be achieved with beam search, and the improvement scales with the width of the search $\mathcal{W}$ as expected.

In essence, beam search is a quasi-exhaustive search in the hyperspace of variable selection. By adjusting the hyperparameter $\mathcal{W}$, we could adjust the trade-off between performance and computational resource needed for the search.

7.5. Conclusion

We have shown that despite its tendency to get stuck in local maxima, iterative removal performs reasonably well for the problem of variable selection and much better than the standard TMVA Ranking method in most cases. If performance is extremely important, beam search could be implemented to circumvent local maxima and achieve slightly higher performance at greater cost of computational resources. It is also worth noting the in most cases beam search only performs better at low number of variables. Iterative removal, however, should be good enough for most purposes.

8. Case Study: Random Walk

8.1. Motivation

To better address the problem of local maxima, we should introduce more randomness into the search.

8.2. Method

Since TMVA Ranking has been proved to be a decent guidance for variable selection as previously mentioned, the algorithm uses variable lists as selected by TMVA Ranking as starting points, thus utilising the easily obtainable information contained in the ranking.

For each iteration and each particular number of variable in a predefined search range, the algorithm does a number of "random tweaks" on the current best performing variable list (using TMVA Ranking produced variable lists as the initial variable lists). Specifically, "random tweaking" means removing N variables at random from the current list and replacing the with N previously unused variables. N is determined with a predefined probability density function (for example, letting $P(N = 1) = 0.4, P(N = 2) = 0.4, P(N = 3) = 0.1, P(N = 4) = 0.1$). By adjusting this probability density function, we could fine tune the behaviour of the algorithm. Greater probability for higher N means the algorithm will tend to explore more, and greater probability for lower N means the algorithm will tend to exploit the current local maxima more. In this investigation, we used a PDF that emphasises low N more. This is because the starting point for the random walk algorithm, i.e. TMVA Ranking variable lists, are already quite good, and more exploitation performs better than exploration in this case.

After producing these next-generation variable lists in this random process, they are tested and the best performing variable list for each number of variables is kept as the parent variable lists for the next generation.

The algorithm is coded such that previously checked combinations of variables will not be generated in the "random tweak" process.

It is also worth noting that this algorithm has more hyperparameters than previous investigated methods. While this could allow greater freedom in algorithmic behaviour, it also requires more fine-tuning.

8.3. Result

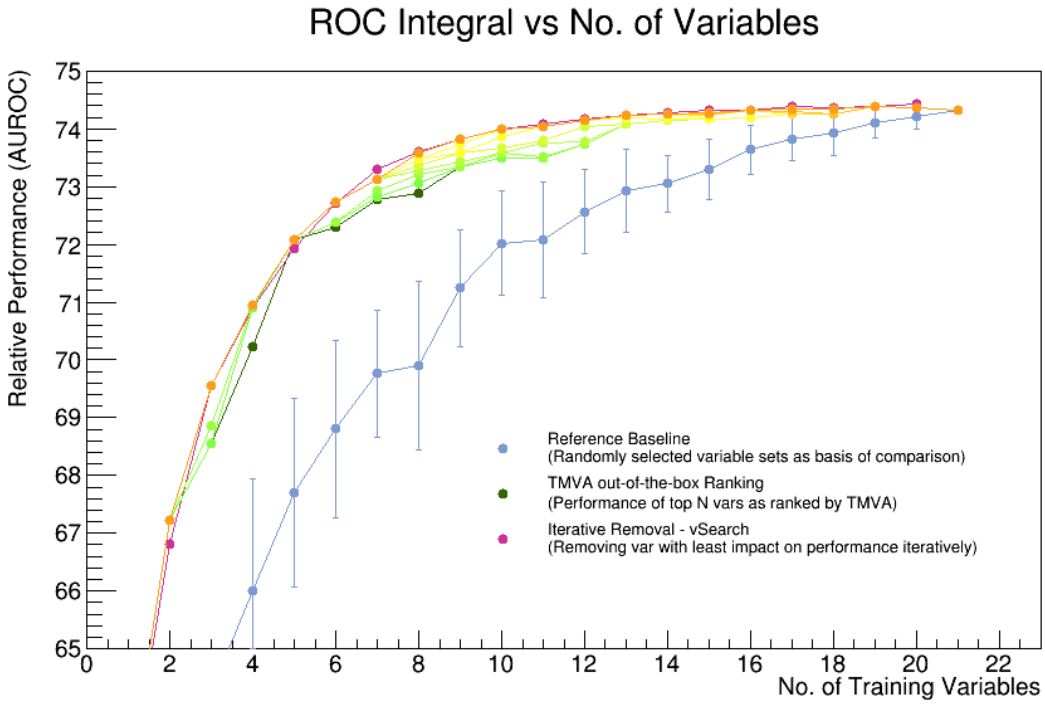


Figure 17: Performance for the random walk algorithm. It starts from the TMVA Ranking variable lists as shown in dark green, and pushes outwards in an envelope. The trace of the envelope in each generation are represented by plots in the colour gradient from light green to yellow, and eventually to orange. 9 generations of random walk is run in this case. After the last generation as shown in plot in orange, the algorithm has achieved comparable or even better performance as that for iterative removal (plot in pink). Plot of performance for each generation of the Random Walk algorithm is available in appendix C.

As figure 17 shows, random walk algorithm improves the performance of variable lists asymptotically. In this particular test, in each iteration, 4 top performing variable lists are chosen, each giving rise to 20 child-variable lists by "random tweaking" for testing in the next iteration. At the end of the ninth iteration, the algorithm has achieved performance comparable to that of iterative removal. The improvement is asymptotic; as the performance approaches that of the iterative removal, the magnitude of improvement for each successive iteration decreases.

This is a good validation of results previously obtained, showing that the variable lists obtained by iterative removal method is at least very close to the optimal configuration at each possible number of variables. The random walk algorithm also has the advantage that its computational complexity does not scale with number of total variables as iterative removal or beam search does. Its performance only depends on the computational

resources available. With more computational resources, more variable lists can be randomly generated from the current best and tested, the closer to optimal configuration we can achieve.

9. Validation with Alternative Variable List

As the different variable selection methods are developed and tested in this project, we used the ICHEP 2016 Summer variable list (Appendix A), 21 variables in total, as the original variable list. While this provides a consistent performance benchmark across different algorithms, it also leads to the risk of over-optimisation. In other words, there is a possibility that, as we optimise search algorithms and draw conclusions about them, the results we obtained might not be generic, but specific to this particular variable list only.

In order to validate the search algorithms against over optimisation, we come up with an alternative variable list (Appendix B) that is completely different from the ICHEP 2016 Summer variable list. Different search algorithms are run again on this variable list with the same data to verify their performances.

As before, for each number of variables, variables are randomly selected to establish a reference baseline for comparison. TMVA out-of-the-box Ranking, Iterative Removal and Beam Search ($\mathcal{W} = 5$) are tested on this alternative variable list.

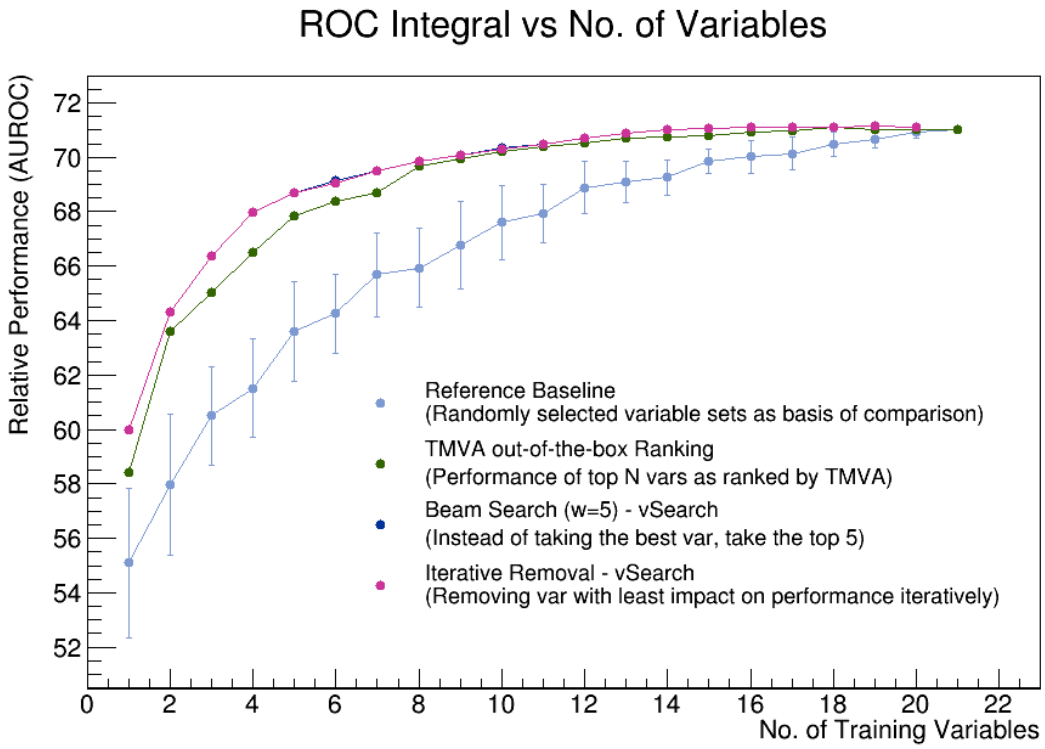


Figure 18: Performance of different search algorithms for the validation variable set.

For all search algorithms, we observe similar performance trends as for the original variable list. The coarse plot in dark green again demonstrates the inherent instability of the TMVA out-of-the-box Ranking. Iterative Removal produces a relatively smooth plot as before, and performs better than TMVA out-of-the-box Ranking for all possible number of variables. Beam Search (plot in dark blue) achieves only very insignificant improvement over iterative removal at certain number of variables, again corroborating our previous conclusion that iterative removal is the most cost-effective in terms of performance vis-a-vis computational complexity, and is good enough for most purposes.

This similarity in the performance is a strong validation for the previously developed different search algorithms and conclusions drawn thereof.

10. Full Feature Analysis

To produce a final recommendation for ttH search, both ICHEP 2016 Summer variable list (Appendix A) and the alternative variable list for validation (Appendix B) are used in this analysis. Iterative removal algorithm is implemented on this combined variable list.
The top ten variables as recommends by iterative removal algorithm is presented in table 4.

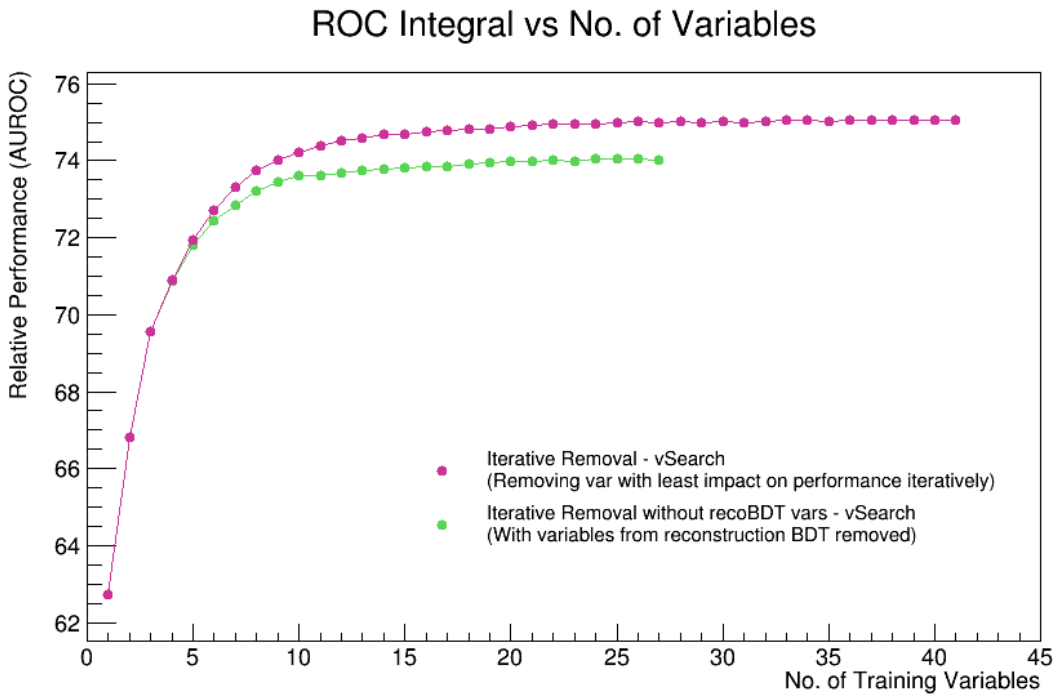


Figure 19: Analysis for all 42 variables by iterative removal is shown as plot in pink.

Table 4: Ten Most Important Variables in Full Feature Analysis

Variables
dRbb_avg
semilepMVAreco_BDT_withH_output
HT_jets
NHiggs_30
Centrality_all
H1_all
Mbb_MindR
Aplan_jets
semilepMVAreco_Ncombinations*
semilepMVAreco_ttH_Ht_withH*

Ranking recommended by Iterative Removal Algorithm

* Asterisks denote variables not from the original ICHEP 2016 Summer variable list

We have also tested the variable list with all variables generated from the reconstruction BDT removed. This is to see if these reconstructed variables, if removed, have a significant impact on the performance of the BDT. If it is possible to remove them, future analysis work can be simplified. This is, however, not the case. Removing all reconstructed variables leads to a significant drop in performance, as shown by the plot

in green in figure 19.

11. vSearch: Parallelised Script for Variable Selection

Running search algorithms manually can be tedious, time-consuming and error-prone. For example, running iterative removal on 21 variables requires about 200 BDT training runs.

During this project we have developed a script in Python which the author has named as "vSearch" for the lack of a better name. It is a "launch and forget" script for variable selection. It is capable of automatic training and testing of different variable lists and automatic selection of best variable lists with predefined search strategies in the iterative process for each search algorithm. It has a human-readable output format so that the best variable list at a certain number of variable can be easily read out after the search is concluded. It also comes with an analysis script that conveniently produces plots for performance as used in this report.

vSearch is designed to run to DESY BIRD Cluster in parallel to save running time, but it could be easily adapted to run on any computing grid that uses Sun Grid Engine. It is coded to guard against job loss or corruption that occur occasionally, so that if the computing grid returns unexpected result file for BDT training the search algorithm is not interrupted.

It is also designed to be highly modular and as generic as possible. It can run different search strategies such as those investigated in this project. Alternatively users can conveniently define search strategies of their own, as the script provides interface for custom-written search strategies. It can also use performance benchmark different from the one used in this project (ROC Integral). It can be adapted to search optimal variables for completely different physics processes that use similar machine learning techniques. As of the time this report is written, it has been successfully applied to a SUSY search [5] within DESY ATLAS group.

This script is available on GitHub[6].

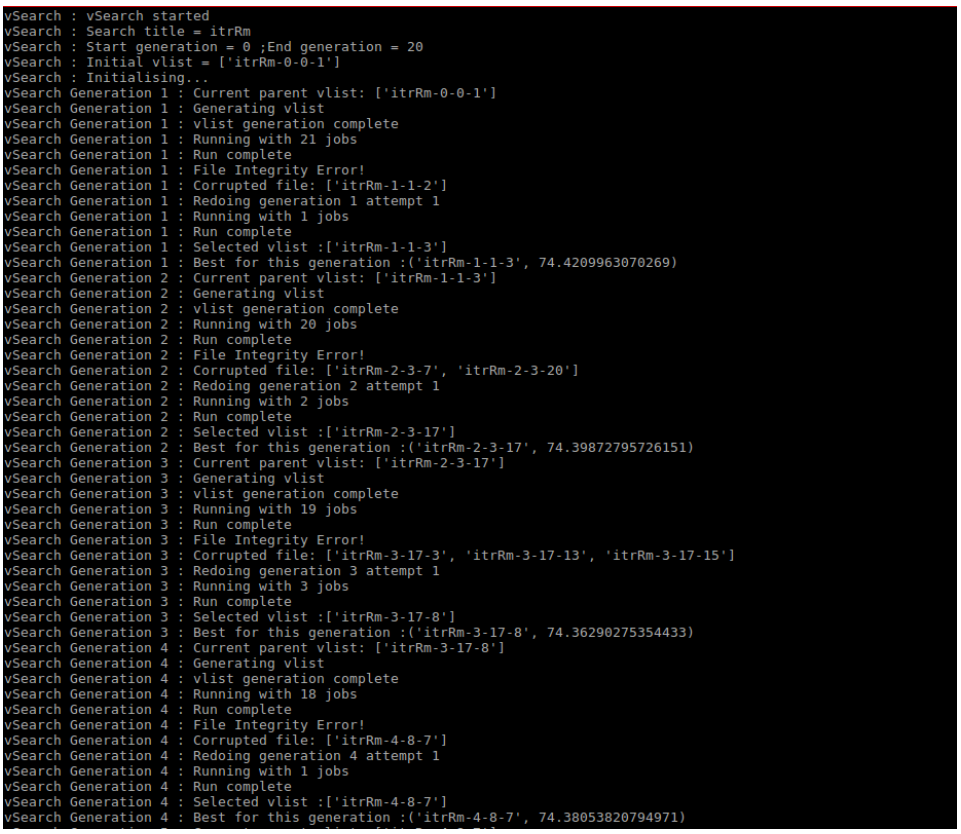


Figure 20: Screenshot of vSearch running iterative removal algorithm.

12. Conclusion

Multiple methods for variable selection for multivariate analysis in the ttH search were investigated. It was shown that visual inspection of the signal/background separation of individual variables is not a good basis for variable selection, as interactions between variables used together in the training are not considered by this method. The effect of correlation was further examined and proved significant with a correlation-based selection method. We examined the current standard practice of selection based on TMVA Ranking and showed that it is sub-optimal and unstable. Following this, three variable selection algorithms were proposed and implemented, all of which performed better than TMVA Ranking. Among these, Iterative Removal algorithm, which is a greedy subtractive iteration algorithm, was shown to be most cost-effective in terms of performance against computational complexity. If needed, Beam Search can be implemented for limited further optimisation by avoiding local maxima at a higher computational cost. With even more computational resources, Random Walk algorithm can be used to probe a wider parameter space. A general-purpose parallelised wrapper script was developed to implement these algorithms and can be used for other search strategies and physics processes. We finally established a new variable list for the ttH classifica-

tion BDT, with addition of variables previously not considered by ICHEP 2016 Summer recommendation.

In future the investigation could be extended to account for systematic errors on variable selection due to Monte Carlo generator bias.

13. Acknowledgement

Dr. Paul Glaysher and Dr. Judith Katzy, supervisors for this project, provided invaluable guidance during the investigation and feedback for this report, for which the author is very grateful. He would also like to thank DESY ATLAS Group for providing a conducive and enjoyable environment. This investigation is carried out as part of the DESY Summer Student Programme 2017.

References

[1] *The ATLAS Collaboration.* Search for the SM Higgs boson produced in association with top quarks and decaying into b bb pair in pp collisions as sqrt(s)=13 TeV with the ATLAS detector, ATLAS-CONF-2016-080, 4 Aug 2016, CERN Document Server.

[2] *LHC Higgs Cross Section Working Group.* Handbook of LHC Higgs Cross Sections: 1. Inclusive Observables, 10.5170/CERN-2011-002, arXiv:1101.0593 [hep-ph].

[3] *M. Czakon, P. Fiedler, and A. Mitov.* The total top quark pair production cross-section at hadron colliders through $O(alpha_S^4)$, Phys. Rev. Lett. 110, 252004 (2013), arXiv:1303.6254

[4] *A. Hoecker, P. Speckmayer, J. Stelzer, J. Therhaag, E. von Toerne, H. Voss.* TMVA 4 Users Guide, arXiv:physics/0703039.

[5] *E. Thompson, C. Sander.* Search optimization using boosted decision trees for a vector boson fusion SUSY signature, DESY Hamburg ATLAS Summer Students Presentations 2017 [`https://indico.desy.de/conferenceDisplay.py?confId=18569`]

[6] *S. An.* GitHub vSearch repository. [`https://github.com/wxjtwxjt/vSearch`]

A. ICHEP 2016 Summer Variable List

Table 5: ICHEP 2016 Summer Variable List

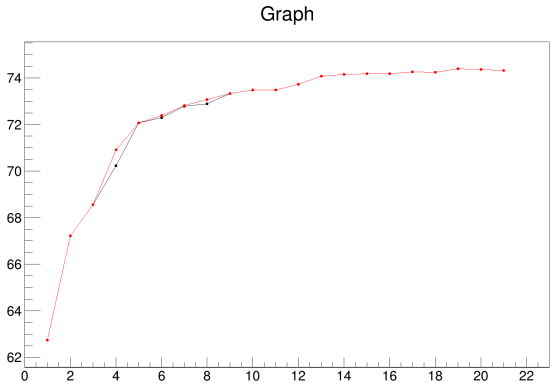
Variables
dEtajj_MaxdEta
NHiggs_30
Njet_pt40
Centrality_all
semilepMVAreco_bbhiggs_dR
dRbb_avg
dRbb_MaxPt
pT_jet5
HT_jets
H1_all
Mbb_MindR
Mbj_MaxPt
dRlepbb_MindR
Aplan_jets
Mjj_MindR
semilepMVAreco_higgs_mass
semilepMVAreco_higgsttbar_withH_dR
semilepMVAreco_higgsbleptop_mass
semilepMVAreco_higgsleptop_dR
semilepMVAreco_higgsbhadtop_withH_dR
semilepMVAreco_BDT_withH_output

B. Alternative Variable List for Validation

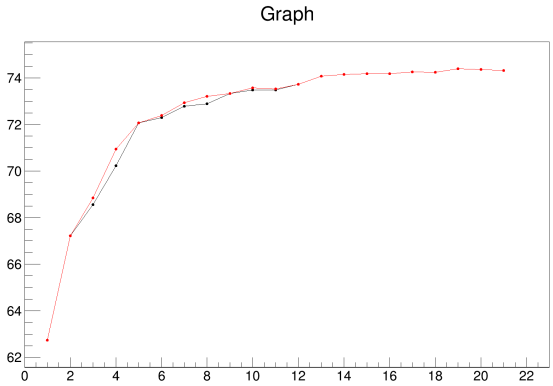
Table 6: Alternative Variable List for Validation

Variables
semilepMVAreco_Ncombinations
HT_all
dRbj_Wmass
Mbj_Wmass
Mbj_MindR
dRHl_MaxdR
dRlj_MindR
pT_jet3
dRbb_MaxM
dRjj_min
H4_all
Aplan_bjets
Mjjj_MaxPt
Mbb_MaxM
Mjj_MinM
semilepMVAreco_higgslep_dR
semilepMVAreco_leptophadtop_withH_dR
semilepMVAreco_b1higgsbhadtop_dR
semilepMVAreco_ttH_Ht_withH
semilepMVAreco_BDT_output
semilepMVAreco_higgsbleptop_withH_dR

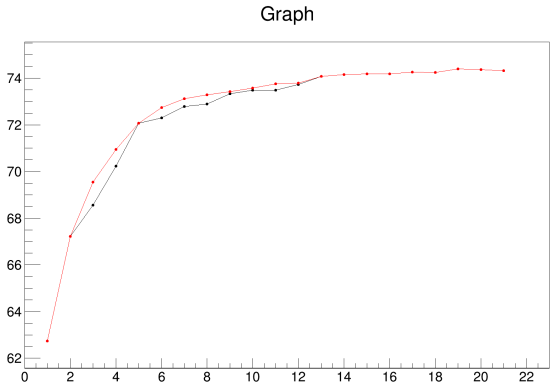
C. Generational Performance for the Random Walk algorithm



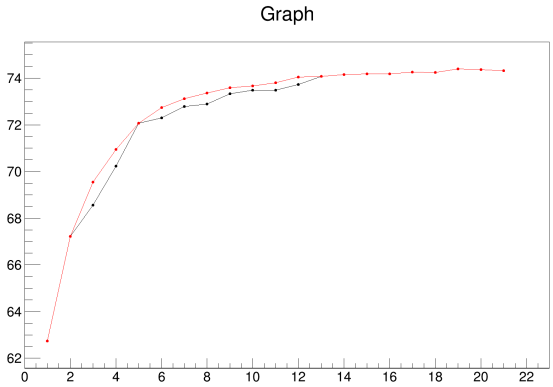
(a) Generation 1



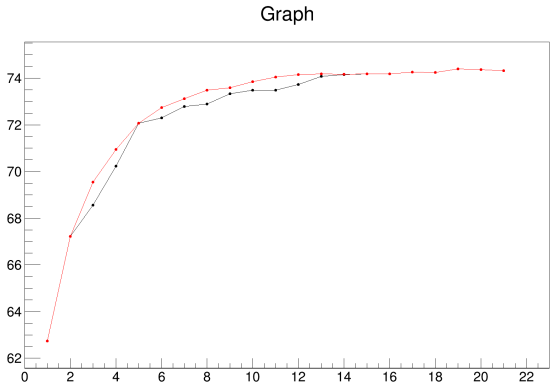
(b) Generation 2



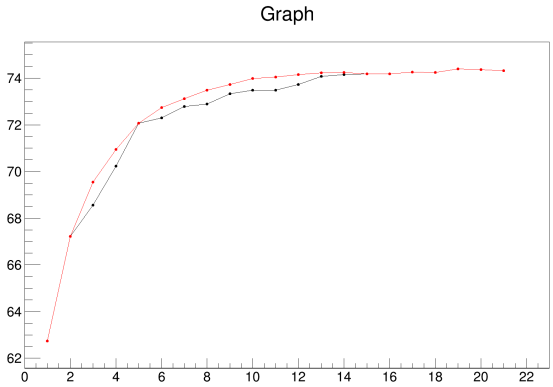
(c) Generation 3



(d) Generation 4

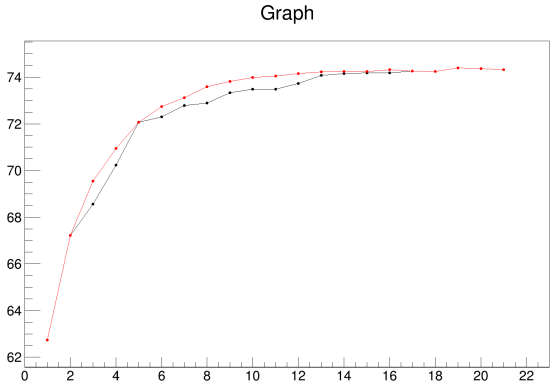


(e) Generation 5

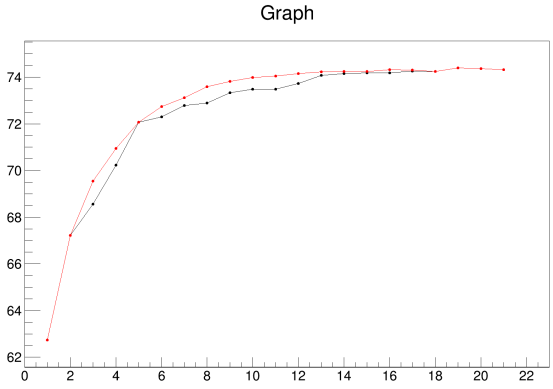


(f) Generation 6

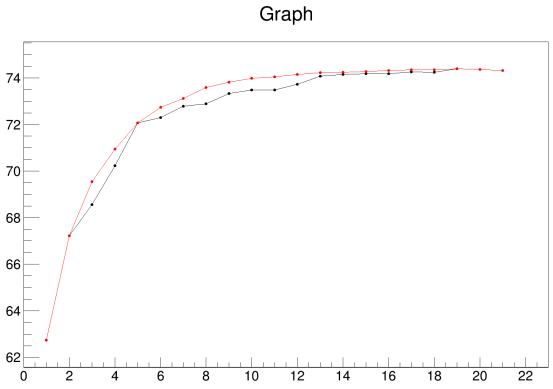
Figure 21: Performance of Random Walk algorithm at different generations 1-6



(g) Generation 7



(h) Generation 8



(i) Generation 9

Figure 21: Performance of Random Walk algorithm at different generations 7-9