Searching for the Higgs Boson Particle using Data Analytics

Ananya Choudhury*
Dept. of Computer Science
Virginia Tech
Blacksburg, U.S.A
ananya@vt.edu

Vivek Bharath Akupatni†
Dept. of Computer Science
Virginia Tech
Blacksburg, U.S.A
vivekb88@vt.edu

Vignesh Adhinarayanan‡
Dept. of Computer Science
Virginia Tech
Blacksburg, U.S.A
avignesh@vt.edu

ABSTRACT

Abstract goes here

1. INTRODUCTION

Physicists use high-energy particle colliders such as the Large Hadron Collider (LHC) to produce new particles in order to understand the elementary structure of matter. In these colliders protons collide at high speed to produce several interesting particles which quickly decays into other stable particles. The particles produced in these experiments are identical in many aspects but differ in their kinematic features. Of particular interest in the Higgs-Boson particle which aids in understanding the fundamental structure of matter. Studying this particle is expected to provide insight into the functioning of universe and its possible fates. While the"real world" impact of the particle is not known yet, the scientific impact is so huge that billions of dollars has been invested in constructing colliders that produce this particle. Identifying the Higgs boson particle reliably and efficiently enables solving the bigger problem of studying the particle.

One fundamental limitation of using magnetic detectors is their inability to distinguish signal (Higgs-Boson producing collision) from background noise (other collisions) when the signal-to-background ratio is too low. An alternative approach is to measure the properties of stable particles and infer the decaying particle originally produced by collision. For this purpose, scientists have created models of Higgs-Boson and other particles, simulate their decay using these models, and record the kinematic properties of the stable particles produced at the end. Based on these kinematic properties, they can create models to find the originally produced particles. This model can then be used in real accelerators for real-time detection of Higgs-Boson particle. Based on the above, we formally state our data mining task as follows.

Data Mining Task. The problem of interest is classification. Given the kinematic features of stable particles produced in a collision, we classify it as signal (decays from Higgs-Boson) or background (otherwise).

Given the rarity in occurrence of signal events and their importance in understanding the nature of matter, the primary goal is to identify most signal events correctly. Secondarily, since millions of events occur every second, we are also interested in identifying techniques that maximize the number of classifications per second.

Our major contributions are as follows:

1. Implementation of several classifiers covering the following broad categories of classifiers for Higgs-Boson detection: Bayesian, tree-based, instance-based, deep learning, ensemble methods, meta-classifiers.

2. Comparison of classification accuracy for the different classification schemes.

3. Evaluation of throughput (Number of classifications per second) for the different classification schemes.

Our contributions help in identifying the appropriate classifiers for the given task and provides clarification on the complexity of features necessary for classification.

Our major findings are as follows: <Fill in interesting results here>

The rest of the paper is organized as follows. Section II provides the necessary background in theoretical physics. Section III describes the related work. Section IV describes our approach. We present our results in Section V and conclude in section VI.

2. BACKGROUND

The LHC collides bunches of proton every 50 ns producing a random number of proton-proton collisions called events. These events produce new particles, most of which are very unstable and decays quickly. The ATLAS detector measures three properties of these surviving particles: type, energy and 3D direction of the particle. Based on these properties, the property of the heaviest primary particles is inferred.

*Ananya worked on.
†Vivek worked on.
‡Vignesh worked on.

An online trigger system selects about 400 events per second producing 3 pb of raw data per year.

Each event contains about ten particles of interest in the final state. The particles of interest for this challenge are electrons, muons, hadronic tau, jets and missing transverse energy. Electrons and muons live longer enough to reach the detector, so their properties (direction and energy) can be measured directly. The other particles such as tau, hadron and jets decays into different different particles and hence their properties are inferred using the law of momentum conservation. The measured momenta of all the particles of the event is the primary information provided in the dataset.

Higgs boson is a particle which is responsible for the mass of all other elementary particles. In the original discovery, the Higgs boson was seen decaying into ÎşÎş, WW, and ZZ, which are all boson pairs[] . The signal class is comprised of events in which Higgs boson decays into two taus. Our goal is to identify those signals from significantly higher number of background events.

3. RELATED WORK

Detecting exotic particles in high-energy physics (HEP) using data analytics techniques instead of the traditionally used physical detectors [17] is not new. Cutts et al. are among the first to use neural networks to identify interesting events in HEP experiments [8]. This was quickly followed by several attempts in improving the classification accuracy of neural networks [13, 16]. Apart from the widely popular techniques based on neural networks, only decision trees were explored for a long time. Bowser-Chao and Dzialo used binary decision trees wto detect top quarks and compared their results with neural networks [4]. The conventional wisdom was that neural networks were by far the best when it comes to classification in HEP until Roe et al. came along and projected boosted decision trees as an alternative to artificial neural networks [19]. By combining several *weak* classifiers, Roe et al. showed that it is possible to obtain better accuracy than a neural network. However, more recently, Baldi et al. showed that a deep learning neural network with several hidden layers outperforms the boosted decision tree [2]. While a number of classification techniques have been developed and applied over the last several years, the HEP community has so far explored only neural networks and boosted decision trees in any depth. This lead to the development of statistical packages for the HEP community such as StatPatternRecognition so that several other techniques based on Rotation Forest, Discriminant Analysis etc. could be explored [14]. Despite this effort, no documented work exists in HEP where alternative techniques are explored even though other techniques are thought to be inferior.

Since Baldi et al. [2] work is closest to ours, we describe it in detail here. The authors in this paper used deep learning methods of neural network to find exotic particles in high energy particle colliders. Deep learning models are neural networks with multiple hidden networks. Current technics like shallow models which are single hidden layer feedforward network fail to capture all features. The deep learning model here is used on 2.6 million training examples and 100,000 validation examples. The mode is a five-layer neural network with 300 hidden units in each layer, learning rate of 0.05, and a weight decay coefficient of 0.00001 . Testing is done on 500,000 examples. For Higgs benchmark, Area Under the Curve(AUC) - complete for deep neural network is 0.88 and for shallow neural network = 0.81.

4. METHODOLOGY

4.1 Data Preprocessing

In some situations, it becomes difficult to measure the physical properties of a particle such as momentum and energy accurately. In our original dataset, as many as 177000 out of 250000 instances had missing attributes. We considered three approaches to deal with missing values. First, we tried ignoring instances with missing attributes. But this resulted in very few useful instances. Second, we tried to replace the missing values with the mean/median. However, this biases the experiments. Finally, we decided to adopt a method known as multiple imputations [3] which replaces the missing values with a random number that follows the distribution for that attribute. We use the `Amelia` [10] package to perform this task.

4.2 Feature Engineering and Selection

The raw dataset includes 17 features. From these 17 basic features, 13 additional features were derived. These 13 derived features describe some property of the particle and requires knowledge of physics. The features are provided by physics. Their description is given in the appendix. From these 30 features, a subset is selected based on the following factors. First, we decide the ability of the feature/attribute to distinguish between signal an background. The distribution of different attributes is shown in Fig. 1 for signal and background separately. Second, we try to avoid using features correlating with each other while building the classifier. Fig. 2 shows the correlation matrix of the features. The final set of features selected differs for each classifier. The details are given in their respective subsections.

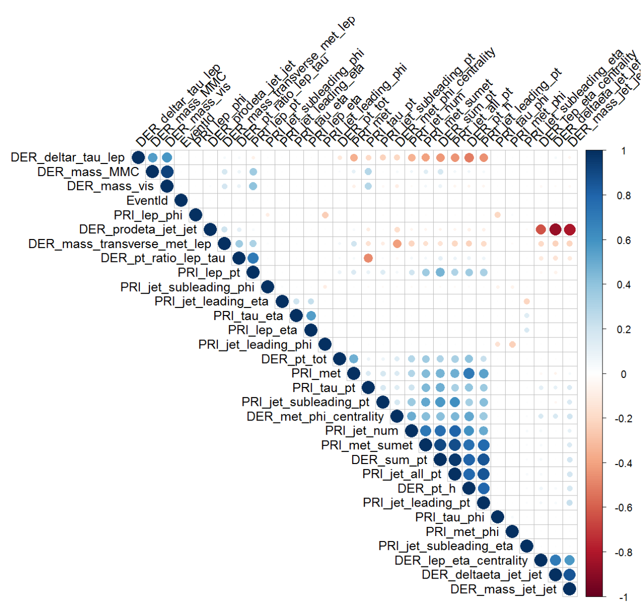

Figure 2: Correlation matrix. Dark blue indicates features that shows strong positive correlation. Dark red indicates features that show strong negative correlation.

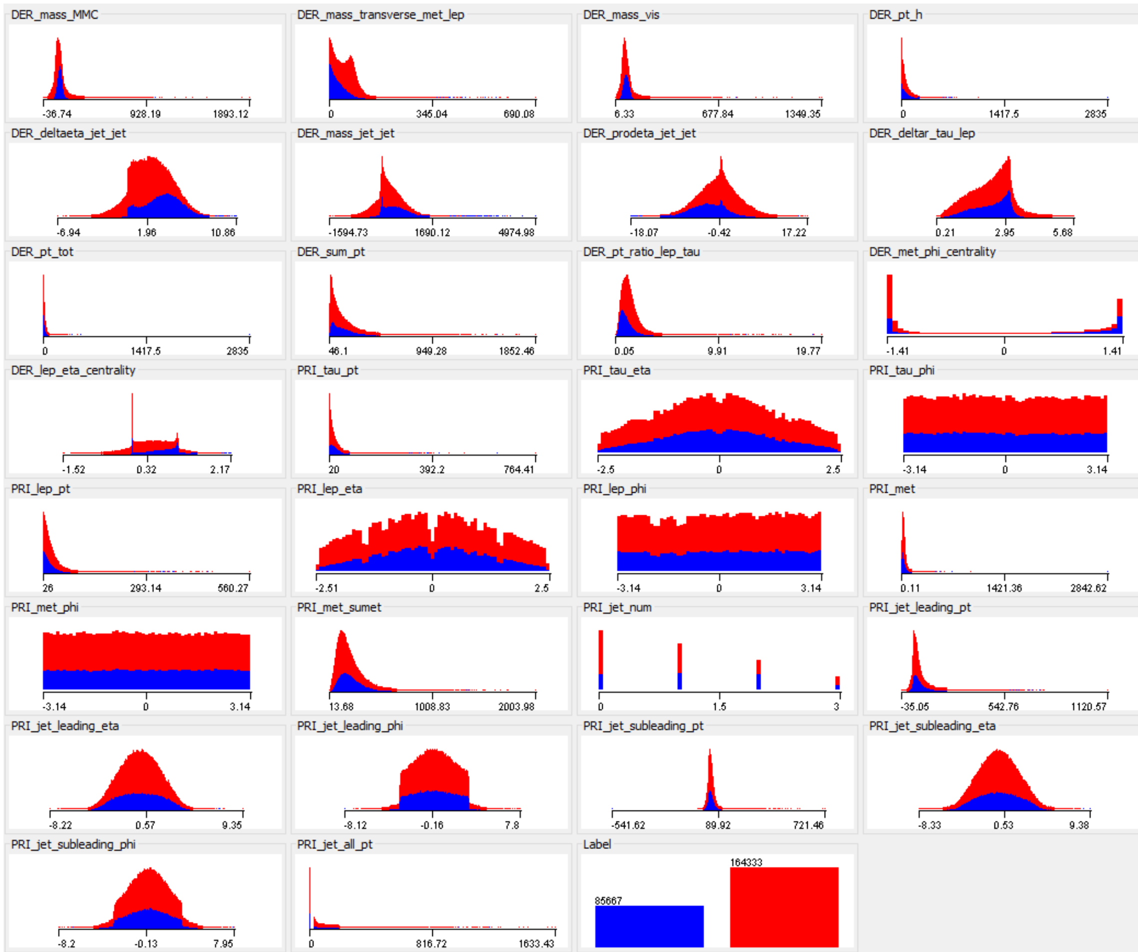

Figure 1: Distribution of different attributes for signal. Blue represents signal and red represents background.

4.3 Data Analytics Techniques

In this section, we describe the various classification schemes we explored. For each technique, we also describe the parameter settings explored in brief.

4.3.1 *Bayesian Classifiers*

Naive Bayes.

This classifier is based on the Naive Bayes technique developed by John et al. [11]. The modeling and prediction overhead were negligible as this method is known to be highly scalable. This method could classify background noise better when compared to signal. During data preprocessing step, we included only those features whose Pearson correlation coefficient was below a particular threshold. Our experiments suggested that out by increasing the threshold value, which in turn increases of number of correlated features, resulted in decrease in the performance of this method.

4.3.2 *Functions-based Classifiers*

Logistic Regression.

This classifier is based on the Ridge estimation technique developed by Cessie et al. [7]. Being a linear method, the modeling and prediction overhead were small. In this technique, we used L1 regularized logistic regression. We explored the performance of this method by varying the number of features in the dataset. The results on our dataset suggest that performance seems to be more or less the same with varying number of features.

Linear Discriminant Analysis.

Quadratic Discriminant Analysis.

4.3.3 *Tree-based Classifiers*

Simple CART.

This classifier is based on the classification and regression trees (CART) technique developed by Brieman et al. [6]. In this technique <describe technique here>. <Describe parameter tuning here>.

4.3.4 *Instance-based Classifiers*

k-Nearest Neighbor.

For the instance-based classifier we explore the k-nearest neighbor (kNN) technique [1]. This is a lazy classification technique in which the k nearest neighbors are looked at to decide the class of a new instance. This technique has practically no training overhead. The classification overhead is very high though, particularly if the distance for all previously seen instances are computed exhaustively. Since this technique is known to perform bad with noisy attributes and irrelevant attributes, we explore removing such attributes from our dataset. We also tried to construct the classifier using top few principal components in order to keep only relevant features. In addition, we also experiment with the number of neighbors considered.

4.3.5 Neural Networks

For Higgs Boson project, we used both deep and shallow neural networks. We analyzed our data using multiple networks each of which are discussed below.

feedforwardnet.

Feed Forward networks consist of a series of layers. The first layer has a connection from the network input. Each subsequent layer has a connection from the previous layer. The final layer is the output layer which produces the network output. The hidden layer is of size 60. The transfer function used in this layer is 'tansig' (tan-Sigmoid Transfer Function). Tansig is a neural network transfer function which calculates a layer's output from its net input. The output layer produces only 1 output (0 or 1). The transfer function used in this layer is 'purelin'. It is a linear transfer function which calculates the final output.

patternet.

It is a Pattern Recognition Network which is trained to classify inputs from target classes. The target data should consist of a vector of 0 or 1. It contains one hidden layer of size 40. The output layer is of size 1.

cascadeforwardnet.

Cascade Forward Network is very similar to feed forward networks but includes a connection from the network input and from every previous layer to following layers. The output layer has two inputs : one from the previous hidden layer and the other from the input. The hidden layer is of size 80. The rest of the network is same as that of FeedForwardNetwork.

Custom Neural Network.

We designed a deep neural network with 4 layers: three hidden layers, one output layer. All the hidden layers are connected to the network input. There is no direct connection between any of the three hidden layers. The output layer takes 4 inputs, three of the inputs are outputs of each hidden layer. The output layer is also recursive layer. So it's own output is fed back as its fourth input. The network model is trained using 'trainrp' network training function which updates weight and bias according to resilient back propagation algorithm (Rprop). The 1st hidden layer contains 20 neurons. The transfer function used in this layer is 'tansig' (tan-Sigmoid Transfer Function). The second hidden layer has 10 neurons. The transfer function used here is 'logsig'. It is log-sigmoid transfer function. The third hidden layer has 20 neurons. The transfer function used here is again 'tansig'. The output layer produces only 1 output (0 or 1). The transfer function used in this layer is 'purelin'.

The number of layers in a network, number of neurons in each layer, transfer function to be used to calculate output of each layer, etc are based on heuristics. We have used biases in the network as networks with biases are more powerful. Each layer's weights and biases are initialized with the Nguyen-Widrow layer initialization method [15].

4.3.6 Meta Classifiers and Ensemble Methods

Classification via Clustering and Regression.

We simply cluster the raw dataset and mark certain clusters as signal and others as noise. Prediction based on the distance of the new data point to the centroid of the two cluster groups.

Bagging.

Bagging technique developed by Brieman involves creating several models using different subsets of the training dataset [5]. Each model does its own classification and they all vote with equal weight to decide on the class. We explored the number of iterations it takes to form the solution. We also experimented with two types of tree based classifiers - REP Tree and Decision Stump Classifier.

Boosting.

Here we explore ADA Boosting developed by Freund and Schapire [9] and MultiBoosting technique developed by Webb [20]. We use these techniques in conjuction with REP Tree and Decision Stump Tree classifiers.

Decision Stump tree with Multi-Boosting.

Based on the

Rotation Forest.

Rotation Forest is an ensemble method developed by Rodriguez et al. [18] where several classifiers are combined to produce a highly accurate classifier.

5. RESULTS

In this section, we describe the dataset used, hardware details, accuracy, and throughput results for the different classifiers.

5.1 Dataset

We make use of the dataset provided by Kaggle [**?**]. This is the cleaned up data from the original dataset provided by physicists at the UC Irvine Machine Learning Repository. This data set contains *250000* instances. Each instance (row) in the dataset describes a collision event detected by the collider. Events are described by the kinematic properties (such as direction and momentum) of the particles produced in a collision. A set of 17 features describe these kinematic properties. In addition, 13 derived features that the physicists deemed important are also included in the dataset. *200000* instances from the original dataset is used for *training* and the remaining *50000* instances is used for *testing.*

5.2 Hardware details

All experiments were run on a MacBook Pro using a *4-core* Intel Core i7 processor running at 2.5 GHz. This machine has *256 KB* of L2 cache per core, *6 MB* of L3 cache, and *16 GB* of DDR3-1600 MHz memory. Modeling and prediction overheads of all techniques were measured and this machine and the corresponding throughput results presented in subsequent sections.

5.3 Accuracy Results

5.3.1 *Bayesian Classifiers*

5.3.2 *Function-based Classifiers*

5.3.3 *Tree-based Classifiers*

5.3.4 *Instance-based Classifiers*

While using the raw and derived features, we did not see any notable impact of the number of neighbors on the accuracy. Also, removing irrelevant and noisy features did not help (F1-score was always around 0.70). However, simply by using principal components improved the accuracy to over 74%. Reducing the number of principal components to four increased the accuracy to over 77.5%. Also, increasing the number of neighbors improved the accuracy further to 80%. The best results were obtained for number of principal components = 4 and number of neighbors = 20. The corresponding results are tabulated in the table.

5.3.5 *Neural Network*

As mentioned before we used multiple NN technics to classify the data between a signal and a background, our results were almost similar. Among the 4 neural networks used here, feedforward and cascade forward network has the best ROC curve (AUC) of 0.91, signal precision and recall as 0.79 and 0.74 respectively.

We noticed that for each network model, the testing performance stops increasing with the increase in the number of neurons in the hidden layers after a certain point . Liu et al. [12] call this as the stop criterion. Beyond this point the neural network overestimates the complexity of the target problem which causes overfitting.

Recently there has been substantial interest in feed forward network with many layers. However, we restricted ourselves to only two layers (one hidden and one output layer) as we noticed an increase in overfitting when the number of layers in a feed forward network is greater than 2. Similarly in the custom network that we designed, the neural network's performance improves when we increase the number of hidden layers from 2 to 3, the AUC is 0.89 but it is still less than other ANN model with 2 layers.

5.3.6 *Ensemble Methods and Meta-classifiers*

Bagging.

Our evaluation indicates that the choice of the decision tree algorithm matters the most when applying the bagging technique. REP tree showed a 20% better accuracy than a Decision Stump Tree. The effect of increasing the number of iterations was low, changing the bag size negligible for reasonable sizes. The best AUC value obtained was 0.91 (Corresponding f1-score was 0.84) when using REP tree with 50% bag size, and 20 iterations. The details of the result is presented in the table.

Boosting.

5.4 Throughput Results

5.5 Discussion

6. CONCLUSION

Conclusion goes here

7. REFERENCES

[1] D. W. Aha, D. Kibler, and M. K. Albert. Instance-based learning algorithms. *Mach. Learn.*, 6(1):37–66, Jan. 1991.

[2] P. Baldi, P. Sadowski, and D. Whiteson. Searching for exotic particles in high-energy physics with deep learning. *Nature Communications*, 5:4308, July 2014.

[3] M. Blackwell, J. Honaker, and G. King. <p>a unified approach to measurement error and missing data: Overview</p>. *Sociological Methods and Research*, In Press.

[4] D. Bowser-Chao and D. L. Dzialo. Comparison of the use of binary decision trees and neural networks in top-quark detection. *Physics Rev D.*, 47:1900–1905, Mar. 1993.

[5] L. Breiman. Bagging predictors. *Machine Learning*, 24(2):123–140, 1996.

[6] L. Breiman, J. Friedman, R. Olshen, and C. Stone. *Classification and Regression Trees*. Wadsworth and Brooks, Monterey, CA, 1984. new edition [**?**]?

[7] S. L. Cessie and J. C. V. Houwelingen. Ridge estimators in logistic regression. *Journal of the Royal Statistical Society. Series C (Applied Statistics)*, 41(1):pp. 191–201, 1992.

[8] D. Cutts, J. Hoftun, A. Sornborger, R. Astur, C. R. Johnson, and R. T. Zeller. The use of neural networks in the d0 data acquisition system. *Nuclear Science, IEEE Transactions on*, 36(5):1490–1493, Oct 1989.

[9] Y. Freund and R. E. Schapire. Experiments with a new boosting algorithm. In *Thirteenth International Conference on Machine Learning*, pages 148–156, San Francisco, 1996. Morgan Kaufmann.

[10] J. Honaker, G. King, and M. Blackwell. Amelia II: A program for missing data. *Journal of Statistical Software*, 45(7):1–47, 2011.

[11] G. H. John and P. Langley. Estimating continuous distributions in bayesian classifiers. In *Proceedings of the Eleventh Conference on Uncertainty in Artificial Intelligence*, UAI'95, pages 338–345, San Francisco, CA, USA, 1995. Morgan Kaufmann Publishers Inc.

[12] Y. Liu, J. A. Starzyk, and Z. Zhu. Optimizing number of hidden neurons in neural networks. In *Proceedings of the 25th Conference on Proceedings of the 25th IASTED International Multi-Conference: Artificial Intelligence and Applications*, AIAP'07, pages 121–126, Anaheim, CA, USA, 2007. ACTA Press.

[13] L. Lönnblad, C. Peterson, and T. Rögnvaldsson. Finding gluon jets with a neural trigger. *Phys. Rev. Lett.*, 65:1321–1324, Sep 1990.

[14] I. Narsky. StatPatternRecognition: A C++ Package for Statistical Analysis of High Energy Physics Data. *ArXiv Physics e-prints*, July 2005.

[15] D. Nguyen and B. Widrow. Improving the learning speed of 2-layer neural networks by choosing initial values of the adaptive weights. In *Neural Networks,*

1990., 1990 IJCNN International Joint Conference on, pages 21–26 vol.3, June 1990.

[16] C. Peterson. Track finding with neural networks. *Nuclear Instruments and Methods in Physics Research Section A: Accelerators, Spectrometers, Detectors and Associated Equipment*, 279(3):537 – 545, 1989.

[17] J. Pinfold. Searching for exotic particles at the {LHC} with dedicated detectors. *Nuclear Physics B - Proceedings Supplements*, 78(1âĂŞ3):52 – 57, 1999. Advanced Technology and Particle Physics.

[18] J. J. Rodriguez, L. I. Kuncheva, and C. J. Alonso. Rotation forest: A new classifier ensemble method. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 28(10):1619–1630, 2006.

[19] B. P. Roe, H.-J. Yang, J. Zhu, Y. Liu, I. Stancu, and G. McGregor. Boosted decision trees as an alternative to artificial neural networks for particle identification. *Nuclear Instruments and Methods in Physics Research A*, 543:577–584, May 2005.

[20] G. I. Webb. Multiboosting: A technique for combining boosting and wagging. *Machine Learning*, Vol.40(No.2), 2000.