The Higgs Boson Machine Learning Challenge

EE 660 Course Project

Project Type-1: Design a system based on real-world data

Shashank Nelamangala Sridhara

nelamang@usc.edu

12/03/2020

1. Abstract

This Machine Learning (ML) project uses the potential of advanced classification methods to classify the events as signal (positive class), in which the Higgs Boson decay into *tau-tau* particles and background (rest) . The dataset released by CERN contains 800,000 simulated data which the physicists had used in their experiments in search of Higgs Boson. Beyond binary classification, the goal of this ML project is to find the region of interest (signal-rich region) in the space of measured features where the events result in the formation of Higgs boson. Different classification methods like Logistic Regression, Decision Trees, Random Forest, Boosted decision trees and stacking machine learning models were used and the performance of each classifier is evaluated based on the given metric called Approximate Median Significance (AMS). Random forest performed better than other models in the constructed hypothesis set with an AMS score of 2.83 on the test set. The winner of the competition used an ensemble of neural networks and has got an AMS score of 3.80.

2. Introduction

2.1. Problem Type, Statement and Goals

Higgs Boson, often referred to as “God Particle” is of paramount importance in particle physics. According to the Standard Model (SM), Higgs Boson is responsible for the mass of other subatomic particles. The discovery of such a particle in 2012 led to the study of characteristics and nature of events which produces Higgs boson. This is a unique Machine Learning problem which aids the study of high energy physics with the help of advanced classification techniques in ML. The simulated dataset of 800,000 proton collision events released by CERN has been extensively used by ML researchers to

effectively develop classification methods which can succinctly separates the events into *signal* (events forming Higgs Boson) and *background* (rest of the events). Details about decay of Higgs boson, and experimental simulation of proton collision can be found in the original paper [1].

In the Machine Learning point of view, this problem can be viewed as a **binary classification problem**. The events data acquired from the collider are represented as a feature vector. The main goal is to classify the events as *signals* (event of interest - Higgs boson to tau tau decay) or *background* (other events which are already known). This can be stated precisely as a ***selection method***, where the classifier should define the signal rich region in the feature space.

The problem in ML perspective is not straightforward and there are many challenges and difficulties. Some of them are listed below.

(1) The biggest challenge is to address the huge proportion of missing data which by nature is inherently different than usual and needs to be handled carefully.
(2) The classes are highly imbalanced in the real world collision, but the simulated dataset has signal rich events. To compensate for this bias, each event is weighted differently based on their probability of occurrence.
(3) There is a significant overlap between the two classes. More precisely, the signal class is a small blob inside a big patch of background class.
(4) Classification accuracy or error rate is a poor measure of success. So, a new performance evaluation metric is derived from the problem - Approximate Median Significance (AMS).

2.2. Our Prior and Related Work (Mandatory)

Prior and Related Work - None

2.3. Overview of Our Approach

The main objective of this project is to classify the events of proton-proton collision into events where Higgs Bosons were formed vs the rest. Exploratory data analysis is done to identify the distribution of each feature and to understand the correlation between them. Missing values were handled in the preprocessing stage and the features were standardized. Dimensionality reduction technique (PCA) is used to handle high dimensionality of the input feature space. Graphical and non linear learning models like classification tree, random forest, AdaBoost and Gradient boost classifiers were used. In addition to these classifiers, “Stacked Machine Learning models” technique is tried to improve the performance. The models were evaluated using a newly derived performance metric called Approximate Median Significance (AMS). The final model was used to get the prediction on test-set and the prediction is submitted to get the final score and ranking.

3. Implementation

3.1. Data Set

The dataset is a list of events (proton-proton collision) and the features are observations that are noted from each event. Presently, the entire dataset consists of 800,000 events (instances in ML terminology), out of which 250,000 instances are allocated as training set, 550,000 instances as test set. The dataset is meticulously divided such that the instances of two sets are identically distributed and about one third of the instances in each set were *signals* (positive class).

Additionally, the training set is provided with the weights (based on the probability of occurrence of the event) to compensate the class imbalance and to help in calculating the performance metric - AMS (explained in later sections).

There are 30 features ($d = 30$) and are divided into PRImitives (prefixed by *PRI*), the values which are directly measured by the detector in hadron collider and DERived (prefixed *DER*), the quantities which are calculated using PRImitives. Detailed information about the features is listed in Appendix 1.

All the features are floating point numbers, except one feature which is categorical. The azimuthal angles are in radians in the range $[-\pi, \pi]$, and quantities such as energy, mass, momentum are in GeV.

3.2. Dataset Methodology

There are a total of 800,000 events (data points) in the entire dataset which are divided into two sets: test set with 550,000 events and training set with 250,000 events. The separation was done by the organizers of the competition. The true labels and weights are not exposed to the public till now and the result on the test set can be evaluated only by submitting the submission file to the kaggle server.

The 250,000 data points which are intended for training are separated[1] into 200,000 points for training the models and 50,000 for validation and model selection. This separation is done before exploratory data analysis so that generalization bound can be obtained using performance on the validation set. Pre-train data is not used because eliminating a portion of a data can shrink the data points for training. Also, a simple validation is employed for model selection instead of cross validation because there is sufficient amount of data available and also cross validation is computationally expensive.

All the classifiers are trained on training data of 200,000 events and the results of each classifier is evaluated on the validation set of 50,000 events. Based on the results obtained on the validation set, the best performing model was chosen and it was evaluated on the test-set consisting of 550,000 events. As there is no access to the true

[1] The data was split into training and validation set using train_test_split from sklearn package

labels and weights to evaluate the final mode, a result file was generated and submitted to the kaggle server. Based on the submission file, kaggle gives the performance score and the ranking in leaderboard. It is important to mention that the test-set was used only once (at the end) to get the final ranking on the leaderboard. Figure 1 shows the entire process.

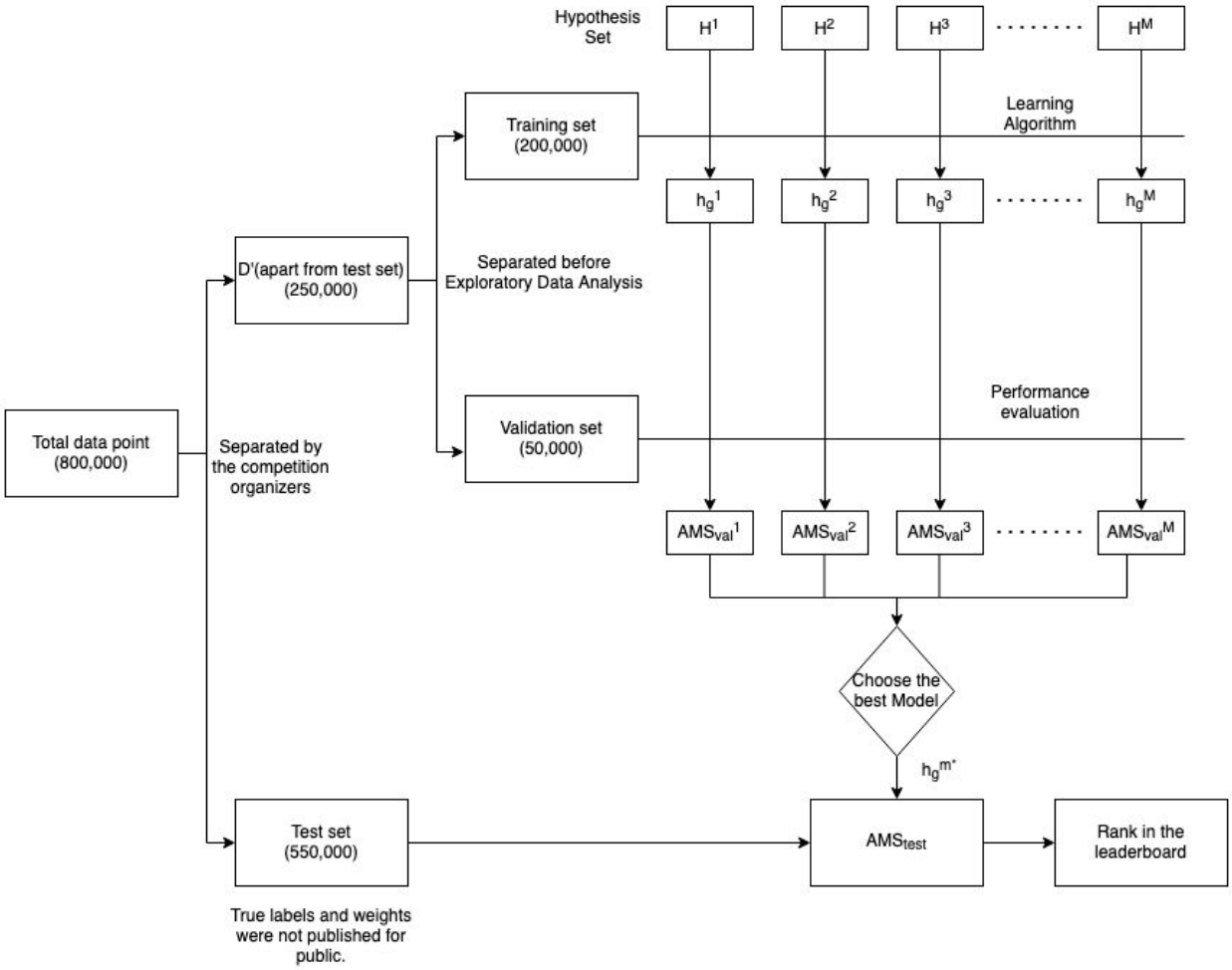

Figure 1: Dataset Methodology

3.3. Exploratory Data Analysis, Preprocessing, Dimensionality Adjustment

3.3.1. Exploratory Data Analysis[2]

In order to study the inherent nature of the data, a comprehensive exploratory data analysis is carried out. Initially, the distribution of each feature is plotted using the training set (after setting the validation set aside). Figure 2 shows the distribution of each feature.

[2] EDA is performed using pandas-profiling function where a report is generated with comprehensive data analysis. HTML file is included in the directory which contains the details of EDA. It can be opened in Google Chrome/Firefox.

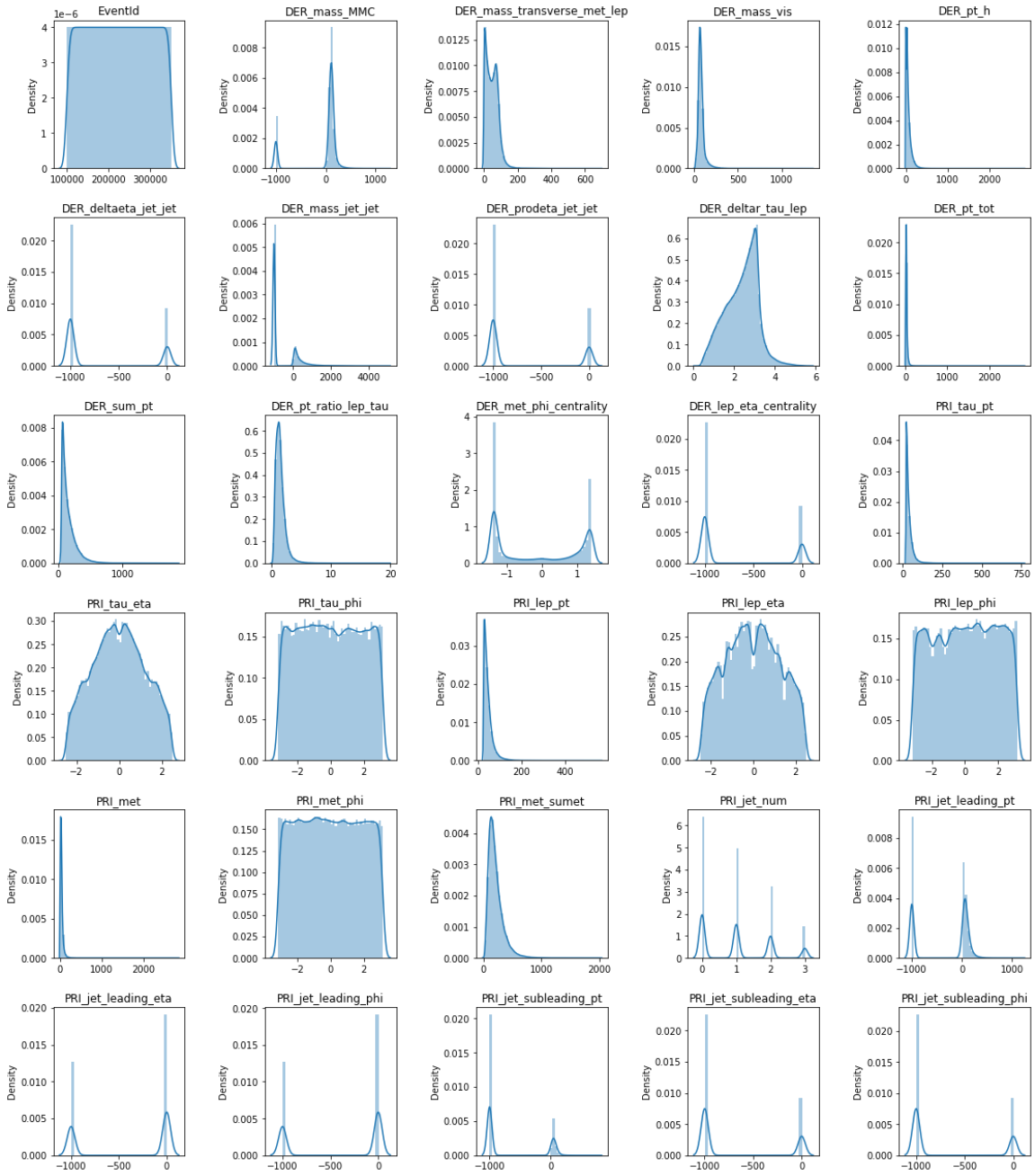

Figure 2: Distribution of each feature.

Based on the distribution of the features, we can observe that PRI_jet_num is a categorical feature and most of the features (around 11) have values near 1000 (-999), which represent the missing data (according to the documentation). The analysis on missing data is explained in the next section. The important observation is that 14 out of 30 features have **long tailed distribution.** These features can be log transformed to minimize positive skew towards smaller values to get uniform distribution.

Using correlation plot the relationship between variables can be inspected. Figure 3 shows the pearson's correlation plot. The correlation matrix reveals that there is huge correlation between some features, especially with the *jet* variables. The PRI_jet variables are highly correlated with DER_deltaeta_jet_jet, DER_mass_jet_jet, and DER_prodeta_jet_jet. This is because the DER quantities are derived from the primary (PRI) quantities.So, there will be some linear relationship between those variables. Inorder to analyse the correlations in detail, scatterplots are observed for selected features. Figure 4 shows the scatterplots.

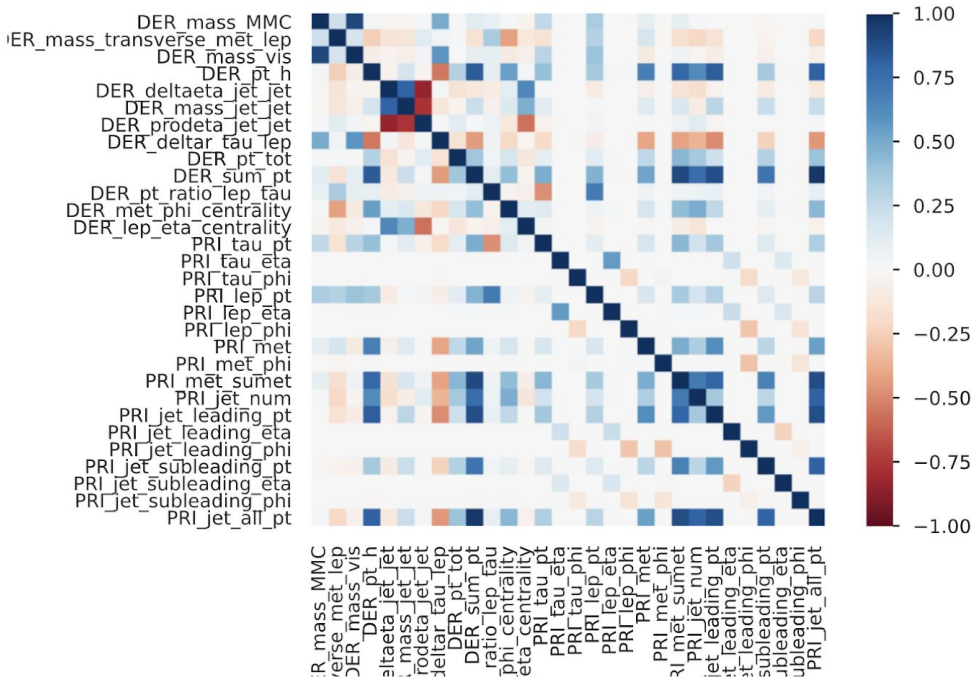

Figure 3: Correlation between features

Looking at the scatterplots, we can see that there exists a linear relationship between DER_deltaeta_jet_jet and DER_prodeta_jet_jet. Also, we can observe that the scatterplot between DER_deltaeta_jet_jet and PRI_jet_subleading_eta has a V shaped correlation because it is derived from the absolute difference between two primitive variables (As explained in the documentation).

3.3.2. Analysis on missing data:

Handling the missing data is one of the most challenging aspects in this project. Totally, more than 70% of the events don't have at least 1 feature. The missing values did not result from unobserved measurements or flaws in the simulation process, instead they are structurally absent i.e, the missingness in the data has some vital information as there is a

pattern in the missing data. For example, the instances where there is no leading jet[3]($PRI\_jet\_num = 0$), the associated PRI quantities are structurally not defined (PRI jet leading pt, PRI jet leading eta, PRI jet leading eta) and the DER features which are derived from these primitive quantities. Figure 5 shows the statistics of the missing data. Statistically, around 7 columns have upto 70% of the missing data, 3 columns have upto 40% of missing data and 1 column (DER_mass_MMC) have 15% of missing data. The missing data has a strong connection to jet number (PRI_jet_num), which is a categorical feature and the observations are listed below,

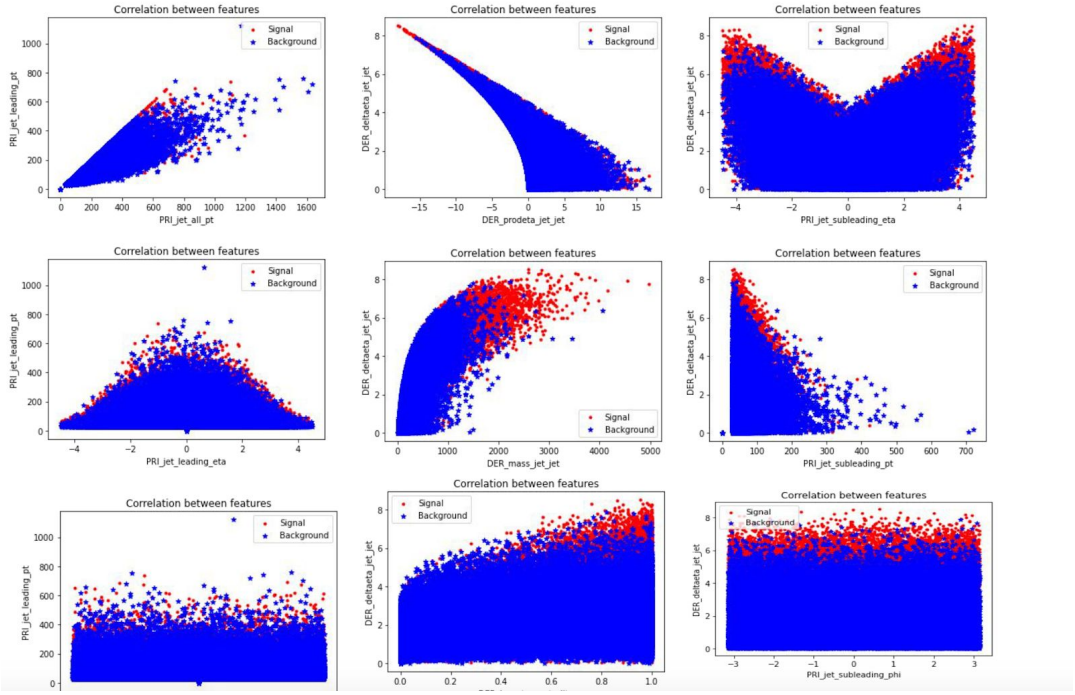

Figure 4: Scatterplots amongst different features

a. When $PRI\_jet\_num = 0$, all the features (both DER and PRI features) related to jets were missing.

b. When $PRI\_jet\_num = 1$ features related to primary jets are no longer missing.

c. When $PRI\_jet\_num = 2\ or\ 3$: all the features related jets are present and there is no missing value. But DER_mass_MMC is still missing.

[3] Jets are the Pseudoparticles which result from quarks decay. 12 out of 30 features are derived from these particles.

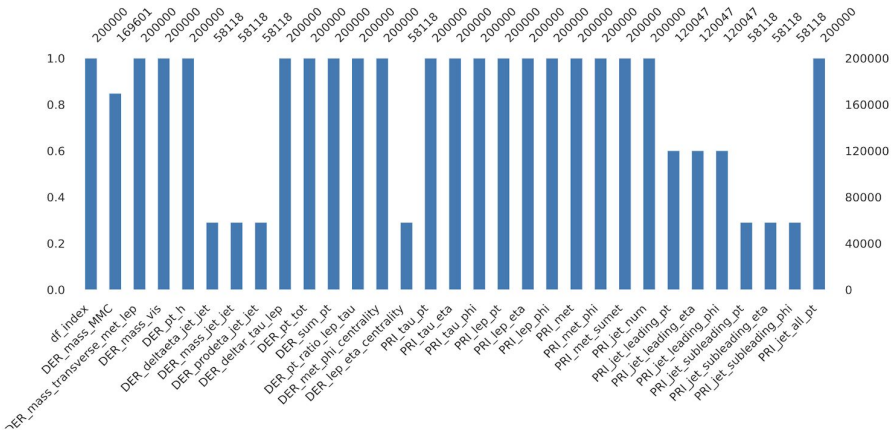

Figure 5: Statistics of the missing data in the training set.

3.3.3. Preprocessing:

The preprocessing part is a bit tricky because of the large number of missing data and categorical features which should be handled with utmost case. Imputing the missing values with mean or median is not a feasible option here because missing values doesn't mean that the values are actually missing, it means nothing is meant to be there. So, preprocessing is done separately for different features because the missingness should be handled separately.

Missing values are handled in the following steps.

a. **Step-1:** When $PRI\_jet\_num = 0$, the missing features actually don't have any significance i.e, if the pseudoparticles were absent then the quantities related to them just don't exist. So, 0 is assigned to those missing values and they are not utilized to calculate parameters during standardization.
b. **Step-2:** Similar strategy is followed when $PRI\_jet\_num = 1$, we assign 0 to all non primary quantities whose values are missing.
c. **Step-3:** Since DER_mass_MMC is missing randomly, we did simple mean imputation[4] for missing values before standardizing the features.

The categorical feature PRI_jet_num (integer values 0,1,2,3) is one-hot-encoded[5] and 3 separate features are created to represent PRI_jet_num (Total features are now 33). The labels were represented as $\{'s', 'b'\}$, where s represents the signal category (positive class) and b represents the background category (negative class). The labels are encoded[6] to integer values $\{1, 0\}$, where 1 represents positive class and 0 represents negative class.

[4] Mean imputation is performed using the function *SimpleImputer* from sklearn package.
[5] One hot encoding is performed using the function *OneHotEncoder* from sklearn package.
[6] Labels are encoded using the function *LabelEncoder* from sklearn package.

Standardization[7] is done on all the features except one-hot-encoded categorical features. Special care has been taken to exclude the replaced 0's in place of missing values while calculating sample mean or standard deviation. The reason for choosing standardization over simple normalization is because centering the columns to 0 mean makes the missing values insignificant i.e, it doesn't contribute to the feature in any way. Also the test set was not used in any way while preprocessing to avoid data snooping. After model selection we applied the same preprocessing techniques on testset using the statistics (sample mean and standard deviation) calculated from the training data.

3.3.4. Dimensionality Adjustment

Principal component analysis[8] is performed on the preprocessed data to analyse the importance and relationship between the features. The features are plotted along the first principal component vs the second principal component[9]. Figure 6 shows the plot.

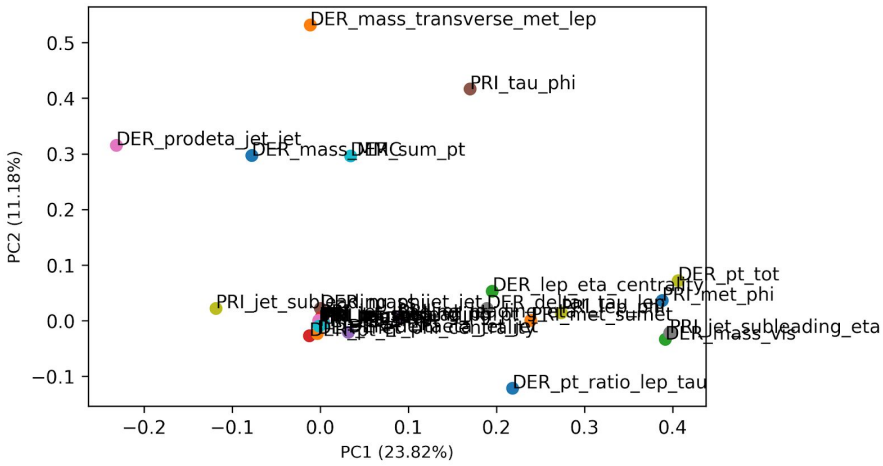

Figure 6: Importance of features in the space of PC1 and PC2

From the figure, some clusters are observed which indicates the presence of collinearity among many of the features. Among all the features, features with mass looks to have higher discriminant power. This can be justified in the physics point of view as well. The derived mass of the Higgs Boson is significantly different from other subatomic particles. CERN scientists used derived mass as a differentiating feature to determine whether a particle was a Higgs Boson or not.

A plot of proportion of variance of each principal component (PCs) is plotted to determine how many principal components should be retained. Figure 7 shows the plot of variance

[7] Standardization is performed on the data-frame using StandardScalar library from sklearn
[8] Principal component analysis is performed using PCA library from sklearn
[9] bioinfokit package is used to plot all the PCA results.

vs PCs.[10] Based on the result, 15 principal components were retained because it captured more than 80% of the variance.

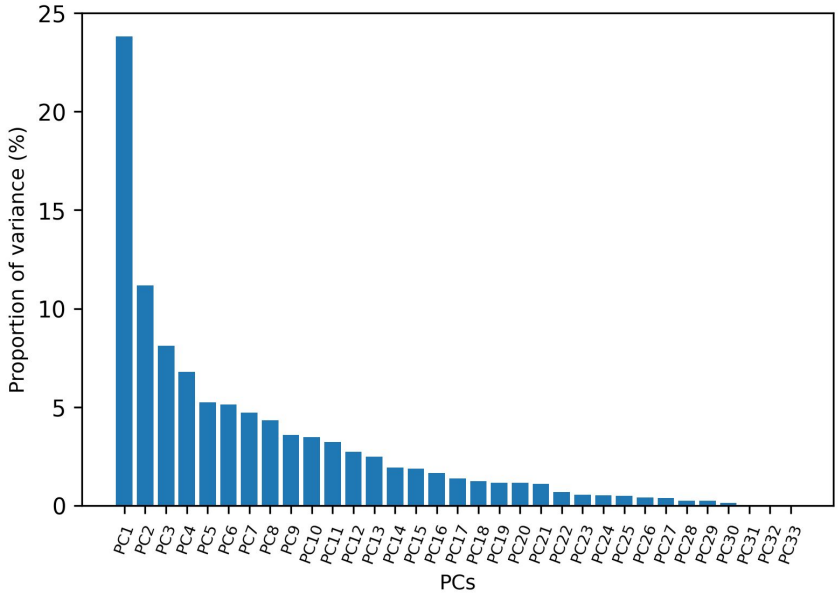

Figure 7: Proportion of Variance vs PCs

3.4. Training Process

Let the dataset be defined as, $D = \{(x_1, y_1, w_1).....(x_n, y_n, w_n)\}$ where $x_i \in R^d$, $y_i \in \{s, b\}$ and $w_i \in R^+$. As mentioned before, in order to compensate for the bias in the number of events between the classes, each event is weighted based on the probability of its occurrence.

In this case, accuracy will be a very poor measure of success. So, a different metric has been given to evaluate the model which is called **Approximate Median Significance (AMS)** [1]. AMS is an objective function used to determine a region in feature space where an enhanced number of signal events are found. AMS is defined as,

$$AMS_C = \sqrt{2(s + b + b_{reg})ln\,(1 + \frac{s}{b+b_{reg}}) - s}$$

where, $s = \sum_{i \in S \bigcap G} w_i$, $b = \sum_{i \in B \bigcap G} w_i$

$$S = \{i : y_i = s\}\,,\ B = \{i : y_i = b\}\,,\ G = \{i : g(x_i) = s\} : Prediction$$

[10] bioinfokit package is used to plot all the PCA results.

s and b are the unbiased estimators of expected number of signal and background events selected by the classifier g. In short, s and b are true positive and true negative rates respectively.

b_{reg} is a regularization term which is set to 10 in the challenge.

Although AMS is a better evaluation metric to test the final model on test-data, it is not suitable for optimization or model selection. AMS is highly unstable and it leads to overfitting, so AUC is used as a metric for model selection on training and validation sets because it is closely related to AMS [2]. Finally, AMS is calculated on the validation set using the selected best model.

First, a hypothesis set is constructed using the initial analysis on training data. So, the results on the training set cannot be used to obtain a generalization bound. Based on the initial analysis, a hypothesis set has been constructed and these reasons to choose each of the models is explained in the later sections. The Hypothesis set if given below,

H = {Naive Bayes, Logistic Regression, Classification trees, Random Forest, AdaBoost, XGBoost, Stacked decision trees models}

3.4.1. Naive Bayes classifier[11]:

Naive Bayes classifier is used as a non-trivial baseline model because it is a simple parametric classifier which estimates the likelihood of an event where Higgs bosons are formed. Equation below explains the Gaussian Naive Bayes classifier concisely.

$$P(x_i|y) = \frac{1}{\sqrt{2\pi\,\sigma^2}} exp(-\frac{(x_i-\mu_i)^2}{2\sigma_y{}^2})$$

To analyse the data, we plotted the data points in 2D feature space along first and second principal components. Figure 8 shows the plot.

As explained in the official documentation from CERN [1], the signal and background events overlap significantly. In-fact signal events (positive class) form a small blob inside a big patch of background events (negative class). Hence a simple classifier like Naive Bayes cannot predict the outcome of signal events accurately.

Result: The AUC score of Naive Bayes classifier on training and validation set is 0.67 and 0.66 respectively. The AMS score on validation set is 1.4047 which is equivalent to the Kaggle rank of 1621[12] (out of 1784 competitors). Hence, Naive Bayes is chosen as a Baseline classifier.

[11] GaussianNB function is used to implement Naive Bayes classifier from sklearn package.

[12] The ranks reported in the intermediate sections are not the final obtained rank from kaggle. They are the approximations provided using the final leaderboard. Final rank on test-set is reported in the Results section.

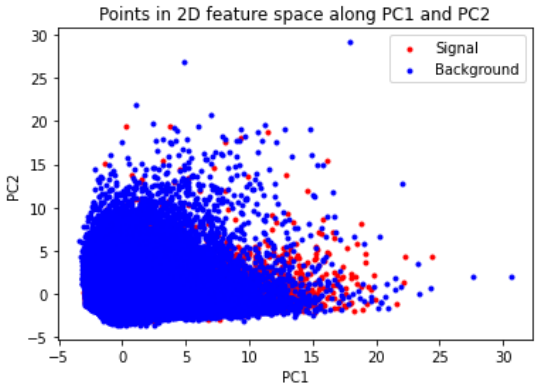

Figure 8: Plot of data point in 2D feature space

3.4.2. Logistic Regression[13]:

Logistic regression was included in the hypothesis set because this is a binary classification problem and logistic regression outputs probabilities using sigmoid function ($\sigma(x)$).

$$\sigma(x) = \frac{1}{1+e^{-x}}$$

To avoid overfitting we introduced l2 regularization term in the objective function and the regularization parameter was chosen based on the performance on the validation set. For model selection, we define the hypothesis set as,

$$H = \{h(0.001),\ h(0.005),\ h(0.008),\ h(0.01),\ h(0.03),\ h(0.05),\ h(0.08),\ h(0.1)\}$$

Therefore, the Number of hypothesis is $M = |H| = 8$. The model parameters in the hypothesis set represent the inverse regularization term ($\frac{1}{\lambda}$). Figure 9 shows the plot of AUC scores vs model parameters on training and validation sets. The VC dimension of the training hypothesis set for each parameter is given by, [3]

$$d_{vc} \geq d + 1 \ and \ d_{vc} < d + 2 .$$

$$So,\ d_{vc} = d + 1 ,$$

where d is the number of features after dimensionality reduction. Therefore, $d_{vc} = 16$.

Result: From the model selection process, the inverse regularization parameter of 0.01 has been chosen to keep the complexity of the model low. Results of Logistic regression were similar to that of Naive Bayes (NB) classifier. AUC scores of 0.6515 and 0.6506 were recorded on training and validation sets respectively. The AMS score on the validation set was 1.34 which is lower than NB. Logistic regression did not work well because the problem requires much more calculations to classify the events than just thresholding the probabilities using sigmoid function.

[13] LogisticRegression function from sklearn package is used to implement the Logistic Regression classifier.

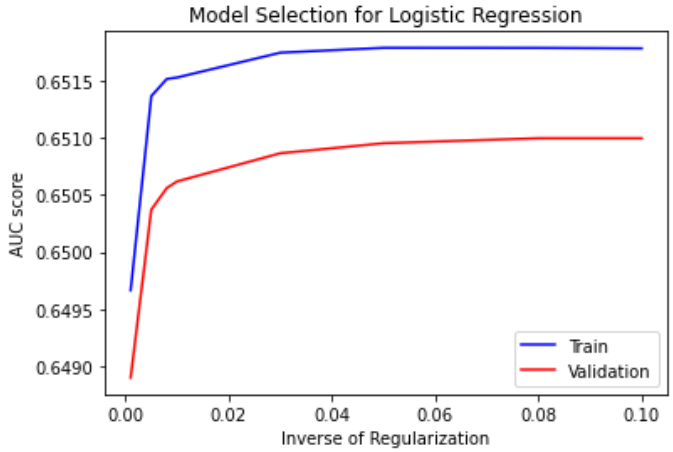

Figure 9: Model section for Logistic regression

3.4.3. Classification tree[14]:

Since the problem is defined as finding a signal rich region in the feature space where the probability of higgs boson formation is higher, simple binary classifiers will fail because the actual decision boundary will be highly complicated. Graphical and non linear learning methods are added as a part of the hypothesis set because they incrementally divide the feature space into 2 separate regions to fit the training data.

Classification tree is a greedy algorithm, so it doesn't not optimize globally at every iteration. There is a high chance for the algorithm to create complex trees which lacks generalizability. Inorder to avoid this we found optimal max_depth parameter using model selection by iteratively evaluating the performance on validation set. Figure 10 shows the model selection result to obtain the optimal depth.

From the below figure, it is evident that decision trees overfit when the number of terminal nodes are more. So, a simplest model which can give reasonable AUC on both training and validation set is picked (occam's razor). Therefore, maximum depth to which the tree is grown is fixed to 10.

Also, the effectiveness of PCA is evaluated on the validation set and it turns out that reducing the dimension of the input feature space using PCA did not help to increase the AMS score on the validation set. This is expected because for graphical and non-linear learning methods features are important to achieve the required discriminant power. Therefore, PCA is not applied for the rest of the classifiers.

[14] DecisionTreeClassifier function from sklearn package is used to implement the classification trees.

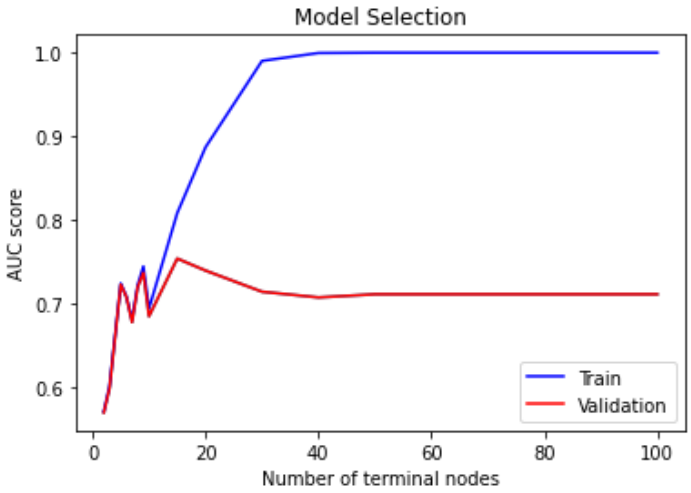

Figure 10: Model selection for decision tree classifiers

Model Complexity: Analysing the VC dimension (model complexity) of a decision tree is complicated because it is hard to define the hypothesis space. Based on the work of Aslan et.al [5], the upper bound can be derived to get the VC dimension of the decision tree. The logical explanation would be that every input will be mapped to a terminal leaf node through a path. If the input feature space is d , then the output of the tree hypothesis is represented by a vector of leaf whose size is 2^d . This shows that the model complexity of decision trees are significantly high and there is a high chance of overfitting.

$$\text{Therefore, } d_{VC} \leq 2^d = 2^{33}$$

Result: There is a significant improvement in the performance compared to the baseline classifier. AUC scores of 0.7835 and 0.7728 were recorded on the training and validation sets respectively. The AMS score of 2.42 was recorded on a validation set which is equivalent to the Kaggle rank of 1357 (out of 1784 competitors). The reason for the improvement is the piecewise complex decision boundary obtained by classification trees.

3.4.4. Random Forest classifier[15]:

It is a well known fact that the ensemble models perform better than a single decision tree because taking average (h_g) over many samples of the dataset will help to reduce the variance which avoids the problem of overfitting. For model selection, there are mainly 4 parameters to tune: number of trees in the forest, maximum depth to which each tree should be expanded, number of features to consider for the best split and maximum number of data to be sampled each time. Number of trees is fixed to 250 and the maximum number of data samples is fixed to 150000 (out of 200000). These values are chosen heuristically based on intuition. The other two parameters are tuned using

[15] RandomForestClassifier function from sklearn package is used to implement the ensemble model.

model selection by evaluating the results on the validation set. Initially the number of features is set to 5 (square root of number of features) and the maximum depth parameter is tuned. Then using the tuned parameter for maximum depth, the number of features parameter is tuned. Figure 11 shows the results of model selection on training and validation sets. Finally, the maximum depth of each tree is set to 12 and the number of features is set to 6 based on the plots.

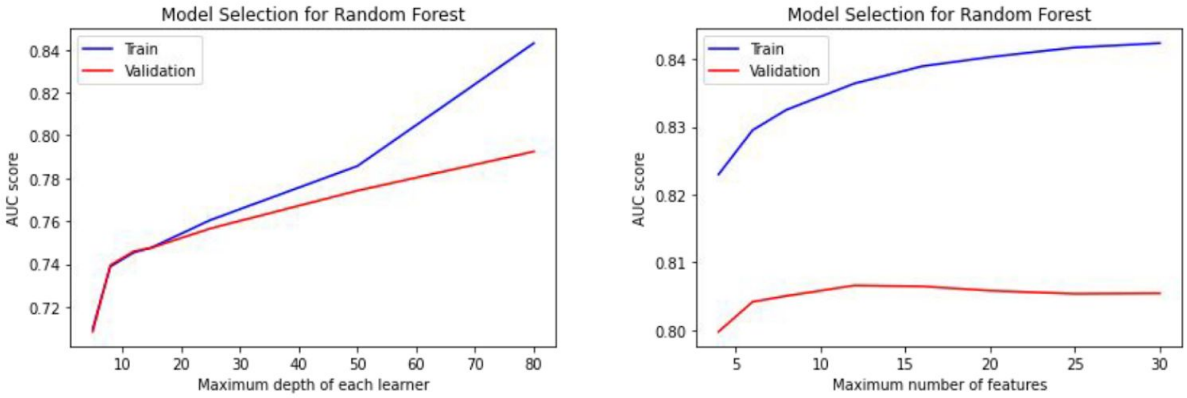

Figure 11: Model Selection for Random Forest

One interesting observation obtained from the Random Forest classifier was the feature ranking[16]. Figure 12 shows the top 10 features and it is interesting to know that the top 3 features are related to mass. A similar argument was made while interpreting the PCA results and also the physics of Higgs boson formation also explains the same. This ensures that we are on the right track in interpreting the results.

Model complexity: Analysing the model complexity of random forest through VC dimension is very hard because there is an inherent randomness in the entire process: sampling of data and features in each iteration. However, an upper bound can be intuitively derived.

Let, n_features = [4,6,8,12,16,20,25,30], depth =[5,8,12,15,25,50,80] and based on the model selection process explained previously, the hypothesis set be defined as,

$$H = \{h(4, depth),\ h(6, depth),\ h(8,\ depth).....,\ h(5,\ nfeature),\ h(8,\ nfeature)\}.....h(80,\ nfeature)\}$$

Therefore, $|H| = 8 + 7 = 15$ .So finding an optimal threshold value for each feature, the algorithm goes through all d dimensions for each data point in the region R_m. Considering the worst case scenario, each data point in the region will have a unique feature and threshold value. In such a case, the VC dimension is the number of points in each region and the number of regions depends on the depth to which we expand the tree.

[16] RandomForestClassifier object have an attribute to obtain the feature importance.

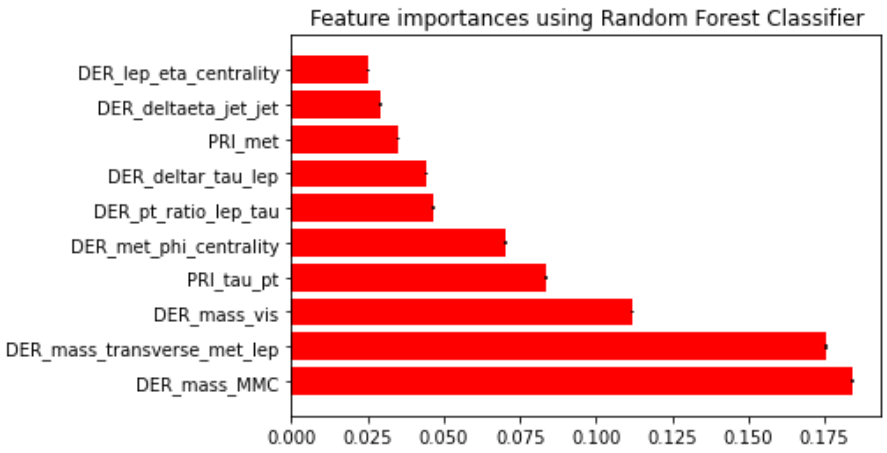

Figure 12: Feature Importance from Random Forest classifier (bottom to top)

Therefore an upper bound for the VC-dimension of each hypothesis in the hypothesis set is,

$$d_{vc} \leq (number\ of\ trees) * (number\ of\ points\ per\ tree) * (depth)$$

Result: The results of Random Forest are quite impressive. AUC score of 0.8295 and 0.8041 is achieved on training and validation sets respectively. Evaluating the final model on the validation set resulted in an AMS score of 2.8294 which is a significant improvement compared to decision trees and the score is equivalent to a kaggle rank of 1190 out of 1784 competitors.

3.4.5. AdaBoost classifier[17]:

Adaptive Boosting (AdaBoost) is one of the first boosting algorithms which is an Adaptive Basis function model (ABM). AdaBoost employs weak learners also called decision stumps, that is only required to do better than chance. It applies the decision stumps sequentially to the weighted data based on the misclassification rate. Also, AdaBoost significantly avoids overfitting because each decision stump is a decision tree of depth =1 and optimizing the sequence of decision stumps actually yields stable performance compared to Bagging methods.

There are two parameters to tune the AdaBoost classifier: learning rate and number of decision stumps. There is a trade-off between the two parameters and they are tuned separately. Initially, learning rate is set to its default value 1 and the optimal number of decision trees is tuned. Later, using the tuned value, optimal learning rate is found. Figure 13 shows the result of model selection.

The figure shows that AdaBoost classifiers don't overfit to the data and it provides a simple decision boundary. Based on the results obtained, the optimal number of weak

[17] AdaBoostClassifier function from sklearn package is used to implement the ensemble model.

learning is set to 250 and the learning rate is set to 0.5, which provides proper optimization.

Model complexity: Analysing the model complexity of a set of decision stumps is straightforward. For a set of m points in a d dimensional feature space, the stumps can classify m point in $2md$ different ways. Using the above bound, it can shown that the VC dimension of the decision stumps can be at most $2(log_2 d + 1)$ [4].

Therefore, $d_{vc} \leq 2(log_2 d + 1)$

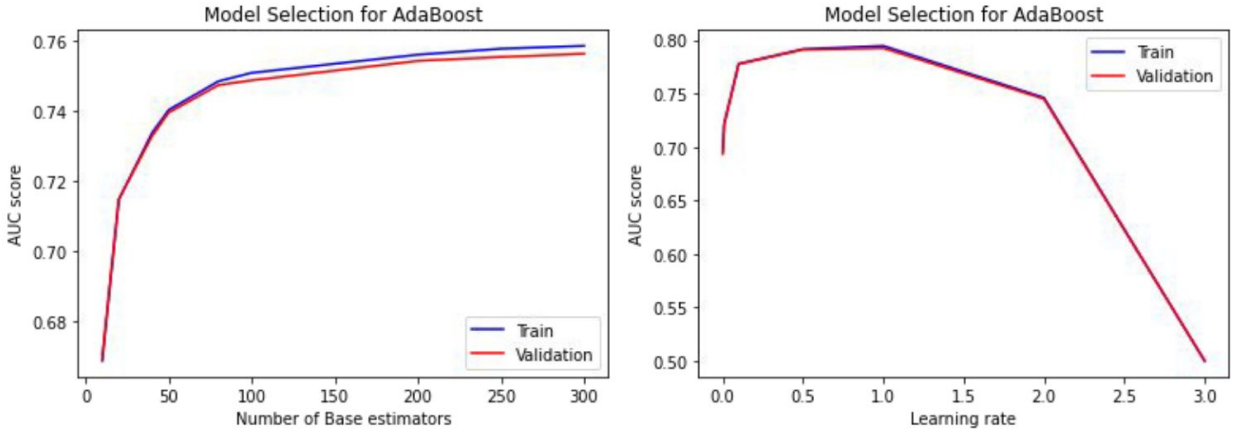

Figure 13: Model selection for AdaBoost Classifier

The simplicity of the model is evident by it's low VC dimension compared to that of decision trees (2^d) and random forest.

Result: Both AUC score and AMS score has reduced for AdaBoost classifier, but certainly the AdaBoost classifier gives more reliable estimate for out-of-sample performance because it does not overfit. AUC scores of 0.7914 and 0.7910 were obtained on training and validation sets respectively. An AMS score of 2.53 has been obtained on the validation set.

3.4.6. Gradient Boosting[18]:

Gradient Boosting is a very popular tree-based ensemble method which is a generalized version of AdaBoost classifier. Gradient Boost classifiers can use other loss functions apart from exponential loss. Model parameters are the same as AdaBoost, but there are more degrees of freedom to choose the values. Additionally, we can tune the parameter which represents the maximum depth of the tree. However, to avoid overfitting it is set to 3. The

[18] GradientBoostingClassifier function from sklearn package is used to implement gradient boost ensemble model.

remaining parameters, learning rate and number of estimators are chosen from model selection Figure 14 shows the results of model selection.

Chosen model parameters: {num_estimators = 150, learning_rate = 0.5, depth = 3 }

The model complexity of Gradient Boost classifier is similar to that of AdaBoost classifier [6]. The VC dimension is given by,

$$d_{vc} \leq (2n+1)\, log2(d+2)$$ where n = number of internal nodes.

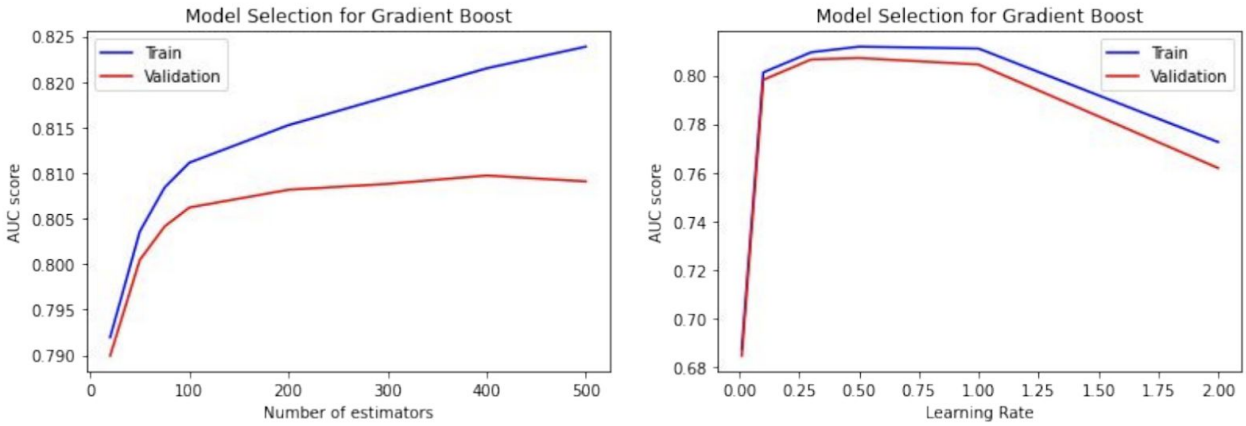

Figure 14: Model Selection for Gradient Boost classifier

Results: The results of Gradient Boost are similar to Random Forest. AUC score of 0.8065 and 0.8131 is achieved on training and validation sets respectively. Evaluating the final model on the validation set resulted in an AMS score of 2.7269 which is a significant improvement compared to decision trees and the score is equivalent to a kaggle rank of 1257 out of 1784 competitors.

3.4.7. Stacking Machine Learning Model[19]:

Stacked generalization is one of the ensemble machine learning techniques which takes the advantages of different well performing models and stack them together to get better performance. The main goal is to learn a meta-learning algorithm to combine the prediction from best-performing models.

Random Forest, AdaBoost and Gradient Boost classifiers were stacked together and used as base models. Finally, a logistic regression model is used as a meta learner to combine the base models. This technique was tried to aggregate the results of previous models since all the graphical and non-linear learning models performed similarly. Figure 15 shows the stacked model of decision trees. The logistic regression was used as the

[19] StackingClassifier function from sklearn package is used to implement stacked ML models

meta-learner because a simple classier is sufficient to combine the results of the decision trees.

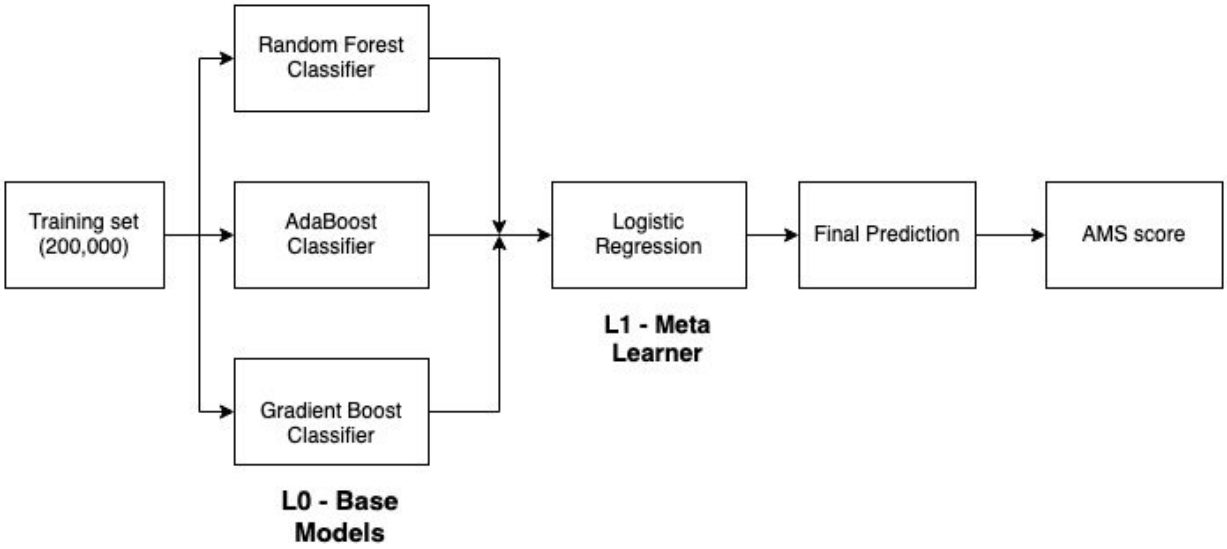

Figure 15: Stacked model of decision trees

Results: Expectation was high on the performance of stacked ML models, but sadly the performance did not increase much. AUC scores of 0.8029 and 0.8023 were recorded on training and validation sets respectively. An AMS score of 2.7310 was observed on the validation set.

3.5. Model Selection and Comparison of Results

As explained in the previous sections, model selection was done with the help of a separate validation set. Cross validation was not employed because of the sufficient availability of data. The results of model selection have already been explained in the previous subsection. After finding the optimal hyperparameters for each model, the trained model has been evaluated on the validation set. The results are compared in Table 1.

Based on the results obtained on the validation set, it is very ambiguous to select the final model to evaluate on the test-set because the AMS scores of Random Forest classifier, Gradient Boost classifier and Stacked models are very close. Classification trees are risky because they may overfit. Random forest is a bootstrap aggregation method used on decision trees to reduce the variance. Boosting techniques on the other hand were expected to perform well, but the performance was close to that of random forest. Instead of interpreting the result as “failure” of boosting techniques, it is optimistic to say that Random Forest has performed exceedingly well because bagging methods along with reducing overfitting handles high dimensionality of the input feature space very well. This is the reason why random forest performed well compared to other models.

Table 1: Performance of different models on training and validation sets.

Model	AUC: Training	AUC:Validation	AMS: Training	AMS: Validation	Rank[20] (out of 1784)
Naive bayes (Baseline)	0.6695	0.6741	1.3929	1.4048	1621
Logistic Regression	0.6515	0.6506	1.3913	1.3408	1625
Classification tree	0.7835	0.7728	2.5831	2.4272	1357
Random Forest	0.8271	0.8028	3.2361	2.8285	1190
AdaBoost	0.7914	0.7910	2.5938	2.5371	1327
Gradient Boost	0.8221	0.8087	2.9700	2.7188	1257
Stacked Models	0.8083	0.8030	2.8659	2.7311	1255

4. Final Results and Interpretation

After scrutinizing the results, Random Forest is selected as the final model because the results were promising on the validation set. Also choosing complicated models like stacked decision trees in spite of similar performance would not be pragmatic based on occam's razor. So, a simple bagging method has been chosen to evaluate on the test-set.

It is important to mention that the test-set was never used during training and model selection. After the final model (Random Forest) was selected, it has been used to get the predictions. Since the test set doesn't have true labels and weights, which are required to calculate AMS score, a submission file has to be created and the predictions should be uploaded to kaggle server. The kaggle server gives the final output on the test-set (only predictions on the test-set will be accepted by kaggle). Figure 16 shows the final score obtained on kaggle.

[20] This rank is not the rank obtained by submitting the validation set result. This is just an equivalent rank based on the current leaderboard rank and AMS score

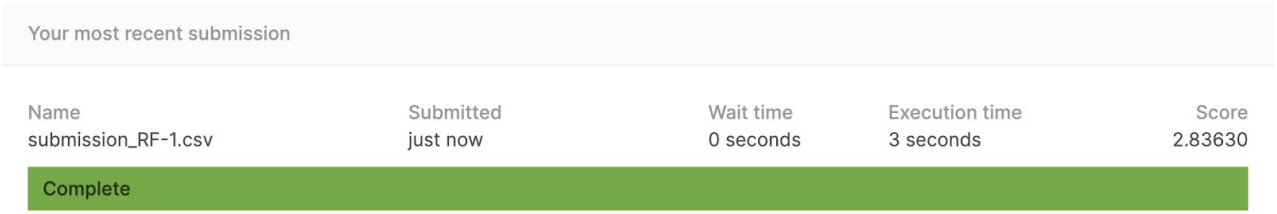

Figure 16: Submitted result on kaggle

All the results of the final model is summarized in Table 2.

Model	AUC: Training	AUC:Validation	AMS: Training	AMS: Validation	Rank[21] (out of 1784)	AMS: Test
Random Forest	0.8271	0.8028	3.2361	2.8285	1190	2.8363

The success of the entire training and model selection process can be explained by demonstrating the generalizability of the chosen machine learning model. Based on the performance on the validation set, we chose Random Forest classifier and it performed equally well on the test-set. This clearly shows that the model is not overfitting the training data.

Performance on out-of-sample events:

Test-set is used to obtain the generalization bound for out-of-sample events because it gives tighter bound compared to validation-set. $Let,\ H = \{h_g^{RF}\}$ be the hypothesis set. The cardinality of the hypothesis set is 1 because we used only one model to evaluate on the test-set. Therefore,

$$Generalization\ error = \sqrt{\frac{1}{2N_{test}} ln(\frac{2M}{\delta})}$$ where, N = 550,000 and M = 1

Assuming $\delta = 0.05$, to predict with probability $\geq 95\%$

$$Generalization\ error = \sqrt{\frac{1}{2*550000} ln(\frac{2}{0.05})} = 1.831 \times 10^{-3}$$

$$\text{So, } E_{out} \leq E_{test} + (1.831 \times 10^{-3}) \text{ with probability } \geq 95\%$$

Interpretation: While Random Forest might not be the best model for this dataset, it is surely better than other models in the constructed hypothesis set. According to the experiments performed, random forest yielded a higher AMS score than the stacked boosted decision trees. This does not mean that stacked models are not suitable for this dataset, in fact the 4th

[21] This rank is not the rank obtained by submitting the validation set result. This is just an equivalent rank based on the current leaderboard rank and AMS score

top competitor in the leaderboard used a meta-ensemble approach with XGboost, Deep Neural Network and Random Forest. However, what separated top performers from others was their feature engineering methods. Indeed it is very hard to come-up with new features without a strong hold on high energy physics and it mattered a lot in the competition. Deriving new features with the help of strong understanding of the given features has given exemplary results for some teams. The winner of the competition has used an ensemble **of deep neural networks (DNN)** and obtained an AMS score of **3.80**. Also, different ways of handling the missing data has influenced the performance in an unusual way.

5. Contributions of each team member

Individual Project.

6. Summary and conclusions

This machine learning project was extremely challenging and at the same time highly rewarding. Understanding the data was the toughest phase in the project because analysing the pattern in missing data required some background in particle physics. Also, the performance metric was new and it was hard to write the code for calculating AMS score on training and validation data. An AMS score of 2.83 has been achieved using random forest classifier with an equivalent kaggle rank of 1190. There is lots of room for improvement. Firstly, deriving new features was very important to obtain good performance. Secondly, ensemble of deep neural networks was a popular model for this data according to the leaderboard submissions, so employing deep learning methods would yield better results. Finally, using popular XGBoost has also proved to be efficient for this dataset.

7. References

[1] Adam-Bourdarios, Claire, et al. "The Higgs boson machine learning challenge." *NIPS 2014 Workshop on High-energy Physics and Machine Learning*. 2015. [Online]. Available: http://proceedings.mlr.press/v42/cowa14.pdf.

[2]Tianqi Chen and Tong He. 2014. Higgs boson discovery with boosted trees. In Proceedings of the 2014 International Conference on High-Energy Physics and Machine Learning - Volume 42 (HEPML'14). JMLR.org, 69–80. [Online]. Available: https://dl.acm.org/doi/pdf/10.5555/2996850.2996854.

[3] http://www.cs.cornell.edu/courses/cs6783/2015fa/lec8.pdf.

[4] http://www.ai.mit.edu/courses/6.867-f03/hw/hw4.pdf .

[5] Aslan, Özlem & Yildiz, Olcay & Alpaydın, Ethem. (2009). Calculating the VC-dimension of decision trees. 193-198. 10.1109/ISCIS.2009.5291847.

[6] https://cs.nyu.edu/~mohri/pub/rgb.pdf

Appendix

7.1. **Appendix I. Detailed Information about the features**

PRI tau pt: Transverse momentum of hadronic tau.	**DER mass MMC** : Estimated mass of the higgs boson candidate
PRI tau eta: Pseudorapidity of hadronic tau	**DER mass transverse met lep:** The transverse mass
PRI tau phi: Azimuthal angle of hadronic tau	**DER mass vis**: Invariant mass
PRI lep pt: The transverse momentum of the lepton	**DER pt h**: Vector sum of transverse momentum and transverse energy
PRI lep eta: Pseudorapidity of the lepton.	**DER deltaeta jet jet**: Pseudorapidity separation
PRI lep phi: Azimuthal angle of the lepton	**DER mass jet jet**: Invariant mass of two jets.
PRI met: Missing transverse energy	**DER prodeta jet jet**: Product of pseudorapidity of two jets.
PRI met phi: Azimuth angle of missing transverse energy	**DER deltar tau lep**: R separation between tau and lepton.
PRI met sumet: Total transverse energy in the detector.	**DER pt tot**: Vector sum of transverse momentum of lepton, leading jet and tau
PRI jet num: Number of jets	**DER sum pt:** Sum of moduli of transverse momenta of particles.
PRI jet leading pt: Transverse momentum of the leading jet	**DER pt ratio lep tau:** Ratio of transverse momenta of lepton and tau
PRI jet leading eta: Pseudorapidity of the leading jet	**DER met phi centrality:** Centrality of azimuthal angle of tau and lepton.
PRI jet leading phi: Azimuth angle of the leading jet	**DER lep eta centrality:** Centrality of pseudorapidity of lepton w.r.t two jets
PRI jet subleading pt: Second largest transverse momentum of the leading jet	**PRI jet subleading phi:** Azimuthal angle of the subleading jet
PRI jet subleading eta: Pseudorapidity of the subleading jet	**PRI jet all pt:** Scalar sum of the transverse momentum of all the events.