ML2 Final Project

EXPLORING THE HIGGS BOSON DATASET

Team Standard Deviation | Arpan Banerjee, Bitan Biswas, Harsh Vardhan Goyal, Sukriti Paul, Surojit Bhattacharya | 13th April,2021

Introduction

The Higgs Boson Dataset is a 30 featured dataset comprising of two events: Signal events and Background Events. The Higgs Boson Machine Learning Challenge was organized to promote collaboration between high energy physicists and data scientists. The ATLAS experiment at CERN provided simulated data that has been used by physicists in a search for the Higgs boson. The Challenge was organized by a small group of ATLAS physicists and data scientists. It was hosted by Kaggle at https://www.kaggle. com/c/higgs-boson; the challenge data is now available on http://opendata.cern.ch/ collection/ATLAS-Higgs-Challenge-2014. This paper provides the physics background and explores the different ensemble models of Machine learning on the dataset and see if the models are comparable amongst each other based on the ROC-AUC curve. We also implement a RNN model and calculate our AMS score based on that to further analyze our results. Keywords: high energy physics, Higgs boson, RNN, Ensemble, Machine learning.

Physics Motivation:

The ATLAS experiment and the CMS experiment, in 2012, claimed the discovery of the Higgs boson (Aad et al., 2012a; Chatrchyan et al., 2012). The discovery was acknowledged by the 2013 Nobel prize in physics given to François Englert and Peter Higgs. This particle is predicted to exist theoretically almost 60 years ago as part of the mechanism by which other elementary particles have mass. Its importance is considerable because it is the final ingredient of the Standard Model of particle physics, ruling subatomic particles and forces. Without confirmation of its existence, the fundamental principle on which our current Standard Model of elementary particles is based would collapse. The discovery relies on experiments being carried out at the Large Hadron Collider (LHC) at CERN (the European Organization for Nuclear Research), Geneva, which began operating in 2009 after about 20 years of design and construction, and which continued to run for at least the next 10 years. The Higgs Boson has many different processes through which it can disintegrate or decay. Beyond the initial discovery, the study of all modes of decay increases confidence in the validity of the theory and helps characterize the new particle. When a particle decays, it produces other particles, and these are classified as being of one of two fundamental types: fermions or bosons, which differ in their amount of intrinsic angular momentum or “spin”. The decay into specific particles is called a channel by physicists. The Higgs boson was first seen in three distinct decay channels which are all boson pairs. One of the next important topics is to seek evidence on the decay into fermion pairs, namely tau-leptons or b-quarks, and to precisely measure their characteristics. The subject of the Challenge was to study the H to tau tau channel. The first evidence of the H to tau tau channel was reported by the ATLAS experiment (The ATLAS Collaboration, 2013).

Dataset and Classification:

From the machine learning point of view the problem can be formally cast into a binary classification problem. Events generated in the collider are pre-processed and represented as a feature vector, out of which not all of them are strongly co-related with the dataset. The problem is to classify events as signal (that is, an event of interest, in our case a H to tau tau decay) or background (an event produced by already known processes). More precisely, the classifier is used as a selection method, which defines a signal-rich region in the feature space.

Terms used in the original ATLAS paper on Kaggle:

- Event: an elementary record. In classification vocabulary, an event is an instance.
- Signal event: an event in which a Higgs boson decays to a pair of tau leptons. In classification vocabulary, a signal event is a member of the positive class.
- Background event: any event other than the signal type. In classification vocabulary, a background event is a member of the negative class.
- Selected event: an event that a selection apparatus deems a candidate for being signal. In classification vocabulary, a selected event is a predicted positive.
- Selected background: a non-signal event that, because of the physical statistical fluctuations, has properties close to those of signal. In classification vocabulary, a selected background event is a false positive.
- Selected signal: In classification vocabulary, a true positive.

The dataset has 30 features, one weights column and one 'label' column which is the output. Due to the complexity of the simulation process, each simulated event has a weight that is proportional to the conditional density divided by the instrumental density used by the simulator (an importance-sampling flavor), and normalized for integrated luminosity such that, in any region, the sum of the weights of events falling in the region is an unbiased estimate of the expected number of events falling in the same region during a given fixed time interval. In our case, the weights correspond to the quantity of real data taken during the year 2012. The weights are an artifact of the way the simulation works and so they are not part of the input to the classifier. For the Kaggle Challenge, weights had been provided in the training set so the AMS can be properly evaluated. Weights were not provided in the qualifying set since the weight distribution of the signal and background sets are very different and so they would give away the label immediately.

The signal sample contains events in which Higgs bosons (with a fixed mass of 125 GeV) were produced. The background sample was generated by other known processes that can produce events with at least one electron or muon and a hadronic tau, mimicking the signal. For the sake of simplicity, only three background processes were retained for the Challenge. The first comes from the decay of the Z boson (with a mass of 91.2 GeV) into two taus. This decay produces events with a topology very similar to that produced by the decay of a Higgs. The second set contains events with a pair of top quarks, which can have a lepton and a hadronic tau among their decay. The third set involves the decay of the W boson, where one electron or muon and a hadronic tau can appear simultaneously only through imperfections of the particle identification procedure.

The Evaluation metric provided in Kaggle is Approximated Median Significance.

$$\text{AMS} = \sqrt{2\left((s+b+b_{\text{reg}})\ln\left(1+\frac{s}{b+b_{\text{reg}}}\right)-s\right)}$$

Where,

s, bs, b: un-normalised true positive and false positive rates, respectively,

br =10 br=10 is the constant regularisation term,

log is the natural log.

Method:

We implement multiple Ensemble models of Classification to compare them based on the ROC_AUC score of the models as well the AMS score, as provided by Kaggle. The dataset was, initially highly imbalanced with larger number of background events than signal events and hence is treated using SMOTE and Random Under Sampling. We, initially take a Baseline Model of Logistic Regression with cross-validation on the training set and use it as the minimum threshold ROC_AUC score that we expect from the ensemble models. The model does a decent job with a certain number of parameters. DecisionTreeClassifier is then implemented on the entire dataset to get hold of the feature importance, so that we can input the features selectively, based on their collinearity and model-based importance. We tune the hyperparameters and get a decision tree which does a better job than the Logistic regression. Feature importance plot shows that the top ten features that are highly correlated to the output and also are comparatively less collinear to each other are:

```
'DER_mass_MMC','DER_mass_transverse_met_lep','DER_mass_vis','DER_deltar
_tau_lep','PRI_tau_pt','DER_met_phi_centrality','DER_pt_h','PRI_met'
```

These features show a strong relation with the outputs of our model and hence are specifically used to train our models to improve the classification rate.

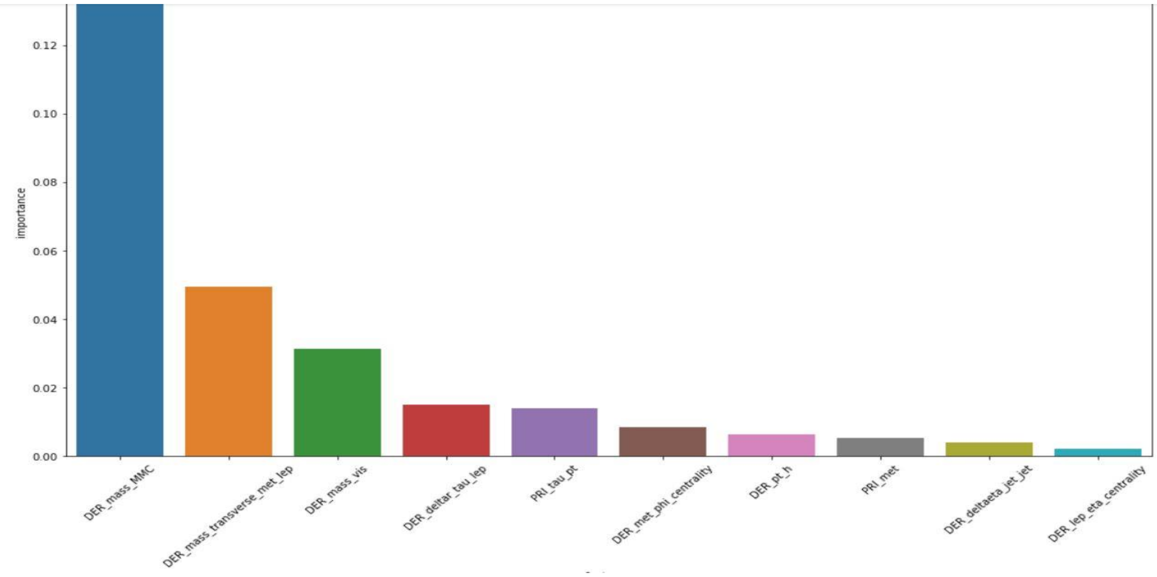


Figure 1: Feature importance using permutation importance for Decision Tree.

Next, we implement Random Forest Classifier with hyper parameter tuning. Implementation is done on both, imbalanced and balanced Dataset, to see if the random forest is doing any better on the balanced dataset or not. The ROC_AUC curve remains the same for both, although the precision-recall curve changes as the number of signal datasets are balanced with the background instances and the prediction accuracy increases accordingly.

Further, we implement XGB Classifier to boost the decision tree models and enhance the predictability of our model. The XGB Classifier does a better job by a very minimal amount from the Random Forest Classifier. Stacking Classifier is also experimented with. The Gradient Boosting , Random Forest and AdaBoost Classifier is implemented and predictions of the same is passed through a Logistic Regression meta classifier.

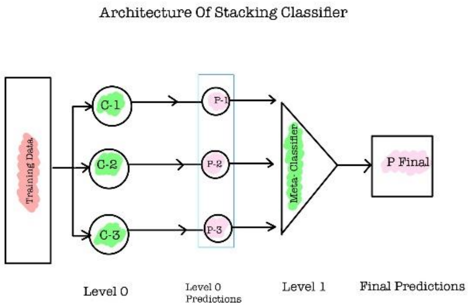


Figure 2: Architecture of the Stacking Classifier

When the ensemble without stacking models were put to test in AMS metric file of Kaggle to check the predictive capacity of the model, we saw it did a poor job and provided a AMS value of 0.2583 on the testing dataset. Thus, it became important to try and input the same feature vector into a Stacking Classifier and a Recurring Neural Network and see if it does a better job than the ensemble models. Further, we had also noticed that most best submissions on Kaggle were based on RNN LSTM and hence, we implemented the same with the help of a few resources as mentioned in the references.

Data

Kaggle provided for a training set and a testing set. The testing dataset was devoid of Weights and label columns whereas the training set had the labels and the weights for our definitive purposes. We take the training set and for each of the models, we train_test_split (Stratified) in 80:20 ratio to obtain the validation and training datasets. The dataset previously was highly imbalanced with a smaller number of signal events and more number of background events. That issue is recovered by using SMOTE and Random Under Sampling. The imbalanced dataset had ***164333 number of background events and 85667 number of signal events.*** The balanced one now has ***164333 number of samples for each of the classes and hence is balanced.*** We train our models on the 80% of our training set and validate or predict on 20% of it. The dataset has 30 columns of features used for predictions and the labels are 0 and 1 where 0 is signal event and b is background.

The features obtained from the permutation importance of the model and the heatmap get us an idea of the important features which are also used to further shorten our feature importance to reduce the cost complexity.

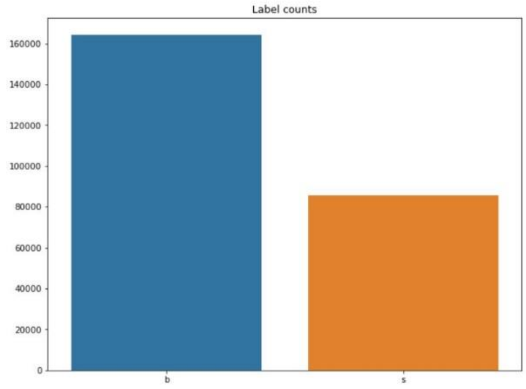


Figure 3: Imbalanced Training Dataset.

Experiment with Models

Baseline Logistic Regression

A baseline logistic regression model is first implemented on the entire dataset, once on the balanced and once on the imbalanced. Hyperparameters used are:

- Max_iter of 1000
- Penalty of l2
- Standard scaler for continuous variables
- One hot encoder for categorical variables
- Column transformer to transform the columns and pipeline to pass all the metrics

The logistic regression gives an f1-score of 0.6(s class) and 0.82(b class) on imbalanced dataset and 0.82 for both on the balanced dataset. The ROC_AUC curve score for balanced dataset it 0.82. This model acts as our baseline. We must outdo this what-so-ever!

Decision Tree Classifier

The decision tree is fit on the imbalanced as well as the balanced data, without the scaling as Decision Trees do not require scaling. The hyperparameters used for the tree are:

```
'max_depth’:range(1,9),'min_samples_leaf':range(3,5),'criterion’:'gini'
```

The f1 score for the imbalanced dataset is 0.73(s class) and 0.86 for the b class. The same for the balanced dataset is 0.81 for both the classes. The ROC_AUC score is 0.88 and 0.89 respectively for both the datasets.

Random Tree Classifier

The Random Forest Classifier from *sklearn.ensemble* package is a ensemble based on decision trees where the data passed is bootstrapped with replacement along with certain number of features taken into consideration, for each of the stumps. It is a meta estimator that fits a number of decision tree classifiers on various sub-samples of the dataset and uses averaging to improve the predictive accuracy and control over-fitting. The sub-sample size is controlled with the max_samples parameter if bootstrap=True (default), otherwise the whole dataset is used to build each tree. The parameters hyper tuned for the random forest are:

```
"n_estimators":np.arange(200,1200,250),'max_features':np.arange(2,8,
2),'max_depth':np.arange(2,8,2)
```

- Estimators within range (200,1200,250)
- Max_features withing 2-8 range with a skip of 2
- Max_depth of 2 to 8 range with skip of 2

The F1 scores are 0.73(s class) and 0.87 for the imbalanced dataset and 0.81 for both for the balanced dataset. The ROC_AUC score remains the same for both i.e. 0.89.

```
<matplotlib.axes._subplots.AxesSubplot at 0x7f0a84fcee10>
```

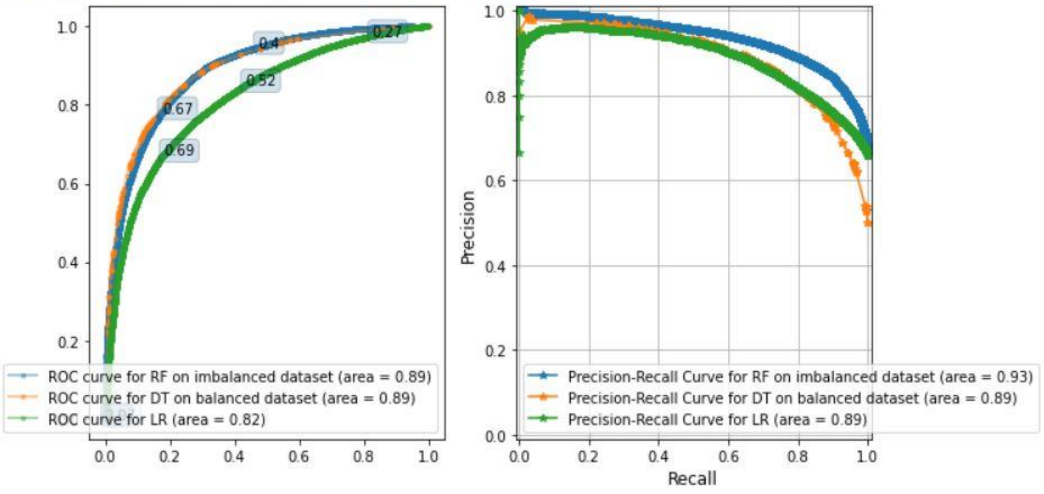


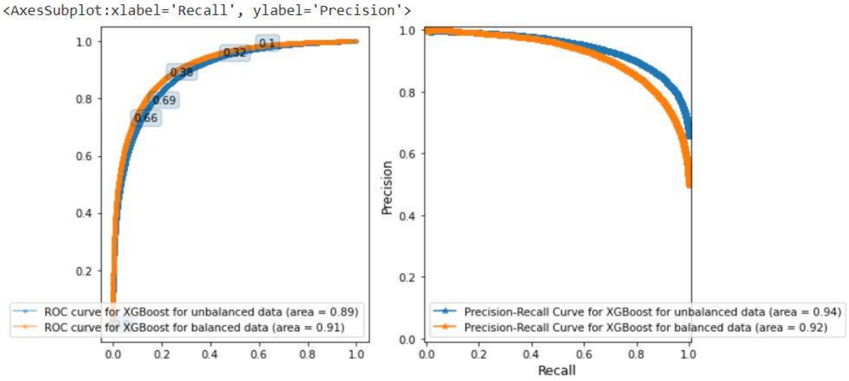
Figure 4: a.ROC_AUC curve for Random Forest, Decision Tree and Logistic Regression
b. Precision Recall curve

Boosting

Boosting is an ensemble modeling technique which attempts to build a strong classifier from the number of weak classifiers. It is done building a model by using weak models in series. In our case, we use a simple XGBClassifier with the following hyperparameters:

```
'n_estimators':np.arange(200,1000,200),'learning_rate':[0.001, 0.01, .1
,.4, .45, .5, .55, .6]
```

Cross validation is carried out on the same to fine tune and obtain the best parameters and that gives us a boosting model with a ROC_AUC score of 0.89 for the imbalanced dataset and 0.92 for the balanced dataset.



```
<AxesSubplot:xlabel='Recall', ylabel='Precision'>
```

Figure 5:Boosting ROC_AUC score and Precision Recall

Stacking Classifier:

Stacking is an ensemble machine learning algorithm that is used to combine well performing models and can be used for both regression and classification problems.

1. We are using the Scikit-learn implementation of stacking.

2. How is stacking different than bagging? Well in stacking the models are different and are fit on the same data set.

3. How is stacking different from boosting? A single model is used to learn how to best combine the predictions from the contributing models.

Approach to get best results with a small model:

1. The first step was to train 3 different classifiers i.e., random forest, gradient boosting classifier and adaboost.

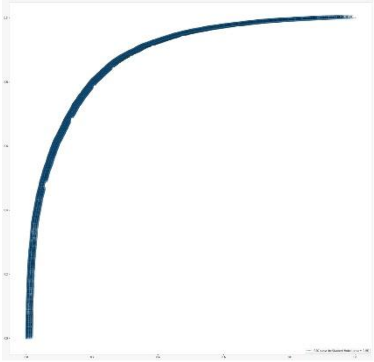


Figure 6: ROC_AUC score of Stacking Classifier 0.88

2. We then take the top 10 features for each of these classifiers using feature importance.

3. A stacking model is prepared with two layers.

4. The first layer has all the models trained earlier.

5. The second layer has a logistic regression model which learns how to best combine the predictions.

6. A cross validated Accuracy, roc_auc score is calculated.

We have used Gradient Boosting , Random Forest , AdaBoost (another one with the earlier 3 and Gaussian Naïve Bayes, K neighbouring Classifiers, Decision tree) at level 0 and meta classifier is Logistic Regression. AMS score obtained is 2.649 which is the best we have had so far!

Comparison of models based on ROC_AUC score

We study the performance of the various models based on the roc_auc score accordingly.

Models	Imbalanced Data	Balanced Data
Logistic Regression	0.76	0.81
Decision Tree	0.88	0.89
Random Forest	0.89	0.89
Boosting	0.89	0.91
Stacking Classifier	0.88	0.88

The ROC_AUC score of the decision tree and the random forest is almost the same. So, we plot the Precision and recall curve to get a better look into it. **Random Forest does a better job in the precision Recall curve with respect to the decision Tree.(Fig 4)However the Stacking Classifier is the best with respect to AMS score (2.649)**

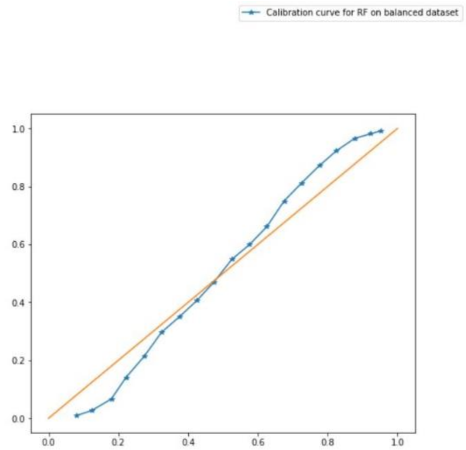


Fig 7: Calibration plot for Random Forest

Neural Network

Long short-term memory (LSTM) units (or blocks) are a building unit for layers of a recurrent neural network (RNN). A RNN composed of LSTM units is often called an LSTM network. A common LSTM unit is composed of a cell, an input gate, an output gate and a forget gate. The cell is responsible for "remembering" values over arbitrary time intervals; hence the word "memory" in LSTM .The expression long short-term refers to the fact that LSTM is a model for the short-term memory which can last for a long period of time. An LSTM is well-suited to classify, process and predict time series given time lags of unknown size and duration between important events. LSTMs were developed to deal with the exploding and vanishing gradient problem when training traditional RNNs.

First, LSTM cell takes the previous memory state Ct-1 and does element wise multiplication with forget gate (f) to decide if present memory state Ct.

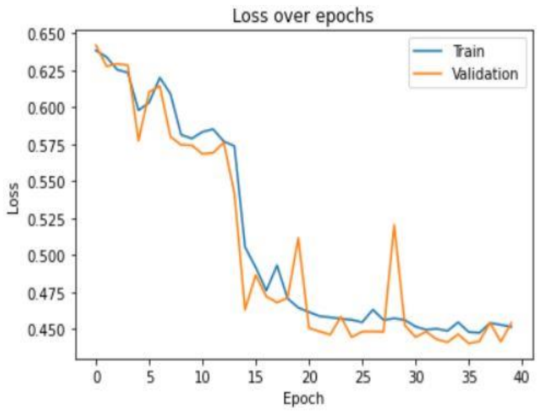


Figure 8: Loss over epoch for RNN model

If forget gate value is 0 then previous memory state is completely forgotten else f forget gate value is 1 then previous memory state is completely passed to the cell.

Ct = Ct-1 * ft

Calculating the new memory state:

Ct = Ct + (It * C`t)

Now, we calculate the output:

Ht = tanh(Ct)

Our neural network model is based on one of the best submissions of the Kaggle Higgs Boson Competition. The model comprises of:

- A LSTM layer of 60 cells with input, forget and output gates and a dropout of 0.1
- 2nd layer of LSTM with recurrent dropout true and 60-unit cells.
- A dense layer of 128 neurons
- A dropout layer of 0.2
- Output layer with the LeakyRelu(alpha=0.1) as the activation function
- Model is compiled with Binary Cross Entropy as loss and Adam as optimizer
- The model is trained with a batch size of 100 and 40 epochs.

The predictions were saved in a csv and submitted in Kaggle to obtain an AMS score of 2.34845.

Inference and Conclusion

The ensemble models seem to do a good job when it comes to ROC_AUC curves but fail to produce an impactful AMS score. Calibrated Models are also tried out, which too do not do a good classification, which is expected. Stacking Classifier surprisingly does better than the RNN model in terms of time and prediction calculation complexity. The models, in general were well calibrated. The Recurring Neural Network model did a good job with an AMS score of 2.35 approximately and the Stacking Classifier did even better with 2.649 AMS score. The models in general showed that the data was dependent on only 10 features.The Boosting model had the highest area under the curve and precision recall curve and hence predicted the signal and background events with 90% accuracy on average.

Partial dependence plots and ICE plots show the variation of different features with the predictions and all the features seem to have a consistent value all throughout except occasional distortion in the graph due to certain underlying physical reason.

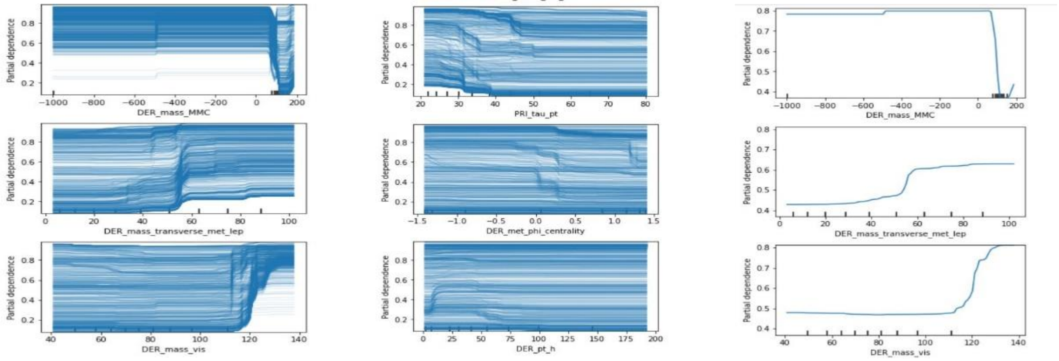
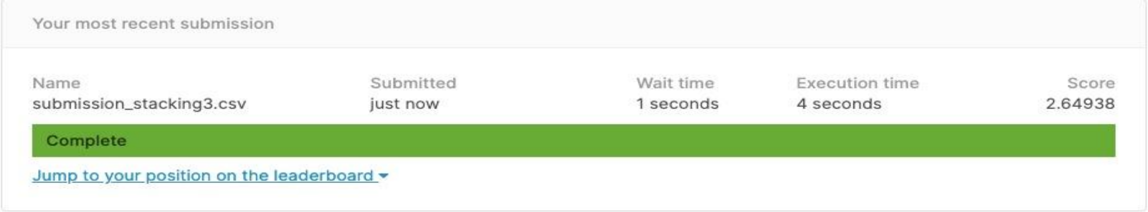


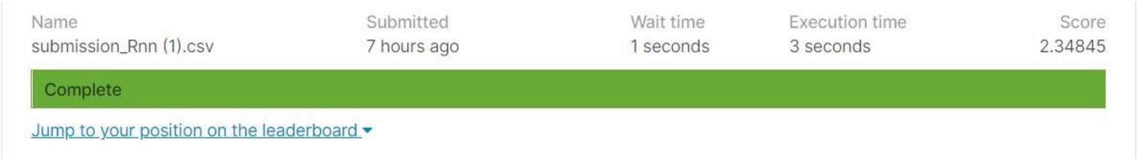
Figure 8: ICE plots Partial dependence plots for the Random Forest Model



Your most recent submission

Name	Submitted	Wait time	Execution time	Score
submission_stacking3.csv	just now	1 seconds	4 seconds	2.64938

Figure 9: AMS score of Stacking Classifier model



Name	Submitted	Wait time	Execution time	Score
submission_Rnn (1).csv	7 hours ago	1 seconds	3 seconds	2.34845

Figure 10: AMS score of RNN model

Further Scope

1. Training Linear SVM ,Isolation Forests, Multilayer Perceptron models on the data and calculating their roc_auc scores.
2. Stacking model using Deep Learning network models.
3. Fine tuning the features to which the model predictions are related, further, through real time analysis and experiments.

References

1. http://proceedings.mlr.press/v42/cowa14.pdf The Higgs Boson Dataset Challenge
2. https://www.kaggle.com/c/higgs-boson
3. http://opendata.cern.ch/search?collections=ATLAS-Higgs-Challenge-2014
4. https://www.kaggle.com/anuragbagadi/detecting-the-higgs-boson-using-rnn
5. https://www.analyticsvidhya.com/blog/2020/07/10-techniques-to-deal-with-class-imbalance-in-machine-learning/