POLITECNICO DI TORINO

MATHEMATICS IN MACHINE LEARNING

HIGGS BOSON DATA ANALYSIS

CANDIDATE:

MOHAMMED EL DOR s289298

PROFESSORS:

FRANCESCO VACCARINO

MAURO GASPARINI

Abstract

Higgs Boson particles were theorized about sixty years ago and lately were discovered experimentally in the large Hadron collider. For analyzing the data better, a machine learning challenge of simulated data was created. The Higgs boson machine learning challenge (HiggsML or Challenge in short) has been set up to promote collaboration between high energy physicists and data scientists. The ATLAS experiment at CERN provided simulated data used by physicists to optimize the analysis of the Higgs boson. The Challenge is organized by a small group of ATLAS physicists and data scientists. It is hosted by Kaggle at https://www.kaggle.com/c/higgs-boson

The Higgs Boson particle have the role of giving mass to other elementary particles. It is the final ingredient of the Standard Model of particle physics, ruling subatomic particles and forces. The challenge is a task to use artificial intelligence to distinguish between signal data and background data. The classification of data is the first step to start analyzing the whole data.

Contents

1 Introduction

As said, the aim of this challenge is to create a collaboration between data scientists and physicists to cooperate in classifying a fundamental particle through improving the procedure that produces the selection region of signal/background labels. The task is compromised of a training set with signal/background labels and with weights, a test set (without labels and weights), and a formal objective representing an approximation of the median significance (AMS) of the counting test. The objective is a function of the weights of selected events. The online competition aims to correctly classify the data label based on a given metric, but I will do a more general work and leave the choice of continuing my work based on the needed task. Therefor this work will be a full analysis of the dataset depending on numerous metrics.

2 Data Overview and Description

The data are simulated and aimed to find a decision boundary between signal and background vibrations. The competition on Kaggle provides two datasets one for training (training.csv) made of 250000 events, with an ID column, 30 feature columns, a weight column, and a label column. And another testing dataset (testing.csv) made of 550000 events with an ID column and 30 feature columns.

By taking a quick look on the data, there are no missing values filled with "*NAN*", but further investigation shows that there are some outliers filled with "-999", and surprisingly they belong to the same instances in fixed attributes and are of high occurrence. After deeper search it was shown that those points were neither outliers nor missing data, they were values that cannot be filled due to physical reasons and some characteristics of the Higgs Boson particle decay [2]. The missing values were filled with "-999" by default, which is a problem to be dealt with in section 4.3.

Some details to get started:

- all variables are floating point, except PRI_jet_num which is integer, and it can be considered as categorical feature also.
- variables prefixed with PRI (for PRImitives) are "raw" quantities about the bunch collision as measured by the detector.
- variables prefixed with DER (for DERived) are quantities computed from the primitive features, which were selected by the physicists of ATLAS.
- it can happen that for some entries some variables are meaningless or cannot be computed; in this case, their value is −999.0, which is outside the normal range of all variables.

2.1 Data Attributes

The dataset has 30 attributes where each one specifies a special characteristic of each instant.

Here is a description for the columns:

- **EventId** An unique integer identifier of the event. Not to be used as a feature.

- **DER_mass_MMC** The estimated mass m_H of the Higgs boson candidate, obtained through a probabilistic phase space integration (may be undefined if the topology of the event is too far from the expected topology).
- **DER_mass_transverse_met_lep** The transverse mass between the missing transverse energy and lepton.
- **DER_mass_vis** The invariant mass of the hadronic tau and the lepton.
- **DER_pt_h** The modulus of the vector sum of the transverse momentum of the hadronic tau, the lepton, and the missing transverse energy vector.
- **DER_deltaeta_jet_jet** The absolute value of the pseudorapidity separation between the two jets (undefined if *PRI_jet_num* ≤ 1).
- **DER_mass_jet_jet** The invariant mass of the two jets (undefined if *PRI_jet_num* ≤ 1).
- **DER_prodeta_jet_jet** The product of the pseudorapidities of the two jets (undefined if *PRI_jet_num* ≤ 1).
- **DER_deltar_tau_lep** The R separation between the hadronic tau and the lepton.
- **DER_pt_tot** The modulus of the vector sum of the missing transverse momenta and the transverse momenta of the hadronic tau, the lepton, the leading jet (if *PRI_jet_num* ≥ 1) and the subleading jet (if *PRI_jet_num* = 2) (but not of any additional jets).
- **DER_sum_pt** The sum of the moduli of the transverse momenta of the hadronic tau, the lepton, the leading jet (if *PRI_jet_num* ≥ 1) and the subleading jet (if *PRI_jet_num* = 2) and the other jets (if *PRI_jet_num =* 3).
- **DER_pt_ratio_lep_tau** The ratio of the transverse momenta of the lepton and the hadronic tau.
- **DER_met_phi_centrality** The centrality of the azimuthal angle of the missing transverse energy vector w.r.t. the hadronic tau and the lepton.
- **DER_lep_eta_centrality** The centrality of the pseudorapidity of the lepton w.r.t. the two jets (undefined if PRI_jet_num ≤ 1).
- **PRI_tau_pt** The transverse momentum $\sqrt{p_x^2 + p_y^2}$ of the hadronic tau.
- **PRI-tau_eta** The pseudorapidity η of the hadronic tau.
- **PRI_tau_phi** The azimuth angle ϕ of the hadronic tau.
- **PRI_lep_pt** The transverse momentum $\sqrt{p_x^2 + p_y^2}$ of the of the lepton (electron or muon).
- **PRI_lep_eta** The pseudorapidity η of the lepton.
- **PRI_lep_phi** The azimuth angle ϕ of the lepton.
- **PRI_met** The missing transverse energy
- **PRI_met_phi** The azimuth angle ϕ of the missing transverse energy.
- **PRI_met_sumet** The total transverse energy in the detector.
- **PRI_jet_num** The number of jets (integer with value of 0, 1, 2 or 3; possible larger values have been capped at 3).
- **PRI_jet_leading_pt** The transverse momentum $\sqrt{p_x^2 + p_y^2}$ of the leading jet, that is the jet with largest transverse momentum (undefined if *PRI_jet num* = 0).

- **PRI_jet_leading_eta** The pseudorapidity η of the leading jet (undefined if *PRI_jet_num* = 0).
- **PRI_jet_leading_phi** The azimuth angle ϕ of the leading jet (undefined if *PRI_jet_num* = 0).
- **PRI_jet_subleading_pt** The transverse momentum $\sqrt{p_x^2 + p_y^2}$ of the leading jet, that is, the jet with second largest transverse momentum (undefined if *PRI_jet_num* ≤ 1).
- PRI_jet_subleading eta The pseudorapidity η of the subleading jet (undefined if *PR_jet_num* ≤ 1).
- **PRI_jet_subleading_phi** The azimuth angle ϕ of the subleading jet (undefined if *PRI_jet_num* ≤ 1).
- **PRI_jet_all_pt** The scalar sum of the transverse momentum of all the jets of the events.
- **Weight** The event weight w_i, not available in the test sample.
- **Label** The event label (string) $y_i \in$ {s, b} (s for signal, b for background). Not to be used as a feature. Not available in the test sample.

In *Figure 2* are some characteristics of the attributes.

2.2 Datasets Partitions:

I already mentioned that there are two datasets (training and test). The training dataset has the "Label" and "Weight" attributes that are missing in the test. The competition aims to find a model that is trained on training and can run efficiently on test. Labels of test are not available anywhere, but they are the main target variable that are submitted on the KAGGLE platform and then evaluated, whereas the weight attribute is also missing but not needed for the competition. The goal of this project is not just to find an algorithm that can score good on the platform but also to make a model and analyze it. For that the project will be of two parts, first, a study will be done only on the training set and for that it will be partitioned into three parts:

1. **Training** (size = 175000)
2. **Evaluation** (size = 7500)
3. **Holdout** (size = 13500)

For that the study will be on both the Label and Weight where the algorithm is trained over the new training and then evaluated over the holdout set so that the algorithm does not overfit on the training data. After that it is tested again on the evaluation dataset so that hyper parameters are not overfitted on the holdout data.

	0	1	2
name	training	evaluation	holdout
size	175000	7500	13500
num_sig	59863	2539	4714
num_bkg	115137	4961	8786
frac	0.342074	0.338533	0.349185
weight	288562	12491.8	22149.4
weight_sig	483.505	20.8146	38.4035
weight_bkg	288079	12470.9	22111
weight_frac_sig	0.00167557	0.00166627	0.00173384

Figure 1 Statistical Characteristics of the datasets

After the study is done on the whole training set given by the competition sponsors, part two will be over the testing dataset that will be

analyzed and some solutions will be done to run the model on the test dataset and predict the Labels as most efficient as possible. The upcoming studies by now will be over the partitioned training set and the whole project will continue the first part for now.

	mean	std	min	25%	50%	75%	max
DER_mass_MMC	121.858528	52.749898	9.044000	95.665000	119.958000	130.606250	1192.026000
DER_mass_transverse_met_lep	49.239819	35.344886	0.000000	19.241000	46.524000	73.598000	690.075000
DER_mass_vis	81.181982	40.828691	6.329000	59.388750	73.752000	92.259000	1349.351000
DER_pt_h	57.895962	63.655682	0.000000	14.068750	38.467500	79.169000	2834.999000
DER_deltaeta_jet_jet	2.403735	0.938491	0.000000	2.403735	2.403735	2.403735	8.503000
DER_mass_jet_jet	371.783360	214.230095	13.602000	371.783360	371.783360	371.783360	4974.979000
DER_prodeta_jet_jet	-0.821688	1.930801	-18.066000	-0.821688	-0.821688	-0.821688	16.690000
DER_deltar_tau_lep	2.373100	0.782911	0.208000	1.810000	2.491500	2.961000	5.684000
DER_pt_tot	18.917332	22.273494	0.000000	2.841000	12.315500	27.591000	2834.999000
DER_sum_pt	158.432217	115.706115	46.104000	77.550000	120.664500	200.478250	1852.462000
DER_pt_ratio_lep_tau	1.437609	0.844743	0.047000	0.883000	1.280000	1.777000	19.773000
DER_met_phi_centrality	-0.128305	1.193585	-1.414000	-1.371000	-0.356000	1.225000	1.414000
DER_lep_eta_centrality	0.458290	0.214759	0.000000	0.458290	0.458290	0.458290	1.000000
PRI_tau_pt	38.707419	22.412081	20.000000	24.591750	31.804000	45.017000	764.408000
PRI_tau_eta	-0.010973	1.214079	-2.499000	-0.925000	-0.023000	0.898000	2.497000
PRI_tau_phi	-0.008171	1.816763	-3.142000	-1.575000	-0.033000	1.565000	3.142000
PRI_lep_pt	46.660207	22.064922	26.000000	32.375000	40.516000	53.390000	560.271000
PRI_lep_eta	-0.019507	1.264982	-2.505000	-1.014000	-0.045000	0.959000	2.503000
PRI_lep_phi	0.043543	1.816611	-3.142000	-1.522000	0.086000	1.618000	3.142000
PRI_met	41.717235	32.894693	0.109000	21.398000	34.802000	51.895000	2842.617000
PRI_met_phi	-0.010119	1.812223	-3.142000	-1.575000	-0.024000	1.561000	3.142000
PRI_met_sumet	209.797178	126.499506	13.678000	123.017500	179.739000	263.379250	2003.976000
PRI_jet_num	0.979176	0.977426	0.000000	0.000000	1.000000	2.000000	3.000000
PRI_jet_leading_pt	84.822105	47.002359	30.000000	57.439000	84.822105	84.822105	1120.573000
PRI_jet_leading_eta	-0.003275	1.382702	-4.499000	-0.433000	-0.003275	0.433000	4.499000
PRI_jet_leading_phi	-0.012393	1.405048	-3.142000	-0.556000	-0.012393	0.503000	3.141000
PRI_jet_subleading_pt	57.679474	17.229894	30.000000	57.679474	57.679474	57.679474	721.456000
PRI_jet_subleading_eta	-0.011845	1.094446	-4.500000	-0.011845	-0.011845	-0.011845	4.500000
PRI_jet_subleading_phi	-0.001582	0.978743	-3.142000	-0.001582	-0.001582	-0.001582	3.142000
PRI_jet_all_pt	73.064591	98.015662	0.000000	0.000000	40.512500	109.933750	1633.433000
Weight	1.646767	1.875103	0.001502	0.018636	1.156188	2.404128	7.822543

Figure 2 Statistical characteristics of the attributed

3 Data Exploration

In this part I will go deeper into the data to get better insights about the attributes and other characteristics.

3.1 Balance

Figure 3 shows that the label 0, which represents class “b” (background), has much higher instances approximately the double the ones of 1, that represent class “s” (signal), and this might create a bias toward the background and for that we should keep in mind that we need algorithms that are robust for that.

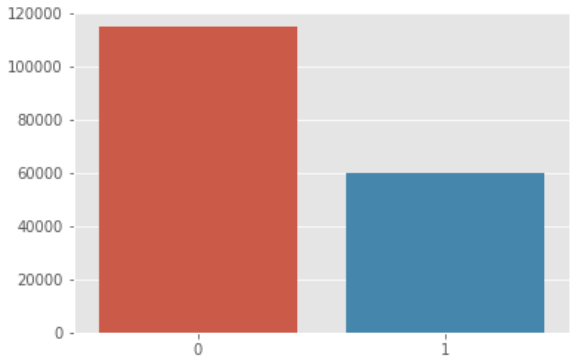


Figure 3 Count of label classes

3.2 Data Correlation

A very useful way to analyze the correlation between numerical features is the correlation matrix. For each pair of attributes, we can compute the covariance and divide it by the product of the standard deviations.

$$\mathbf{cov}(\boldsymbol{x},\boldsymbol{y}) = \frac{\mathbf{1}}{\boldsymbol{n-1}}\sum_{\boldsymbol{i=1}}^{\boldsymbol{n}}(\boldsymbol{x_i} - \bar{\boldsymbol{x}})(\boldsymbol{y_i} - \bar{\boldsymbol{y}}) \qquad \boldsymbol{\rho} = \frac{\boldsymbol{cov(x,y)}}{\boldsymbol{S_x S_y}}$$

Figure 5 show *the correlation matrix of the data attributes and Figure 4 shows the correlation between the attributes and the target variable.*

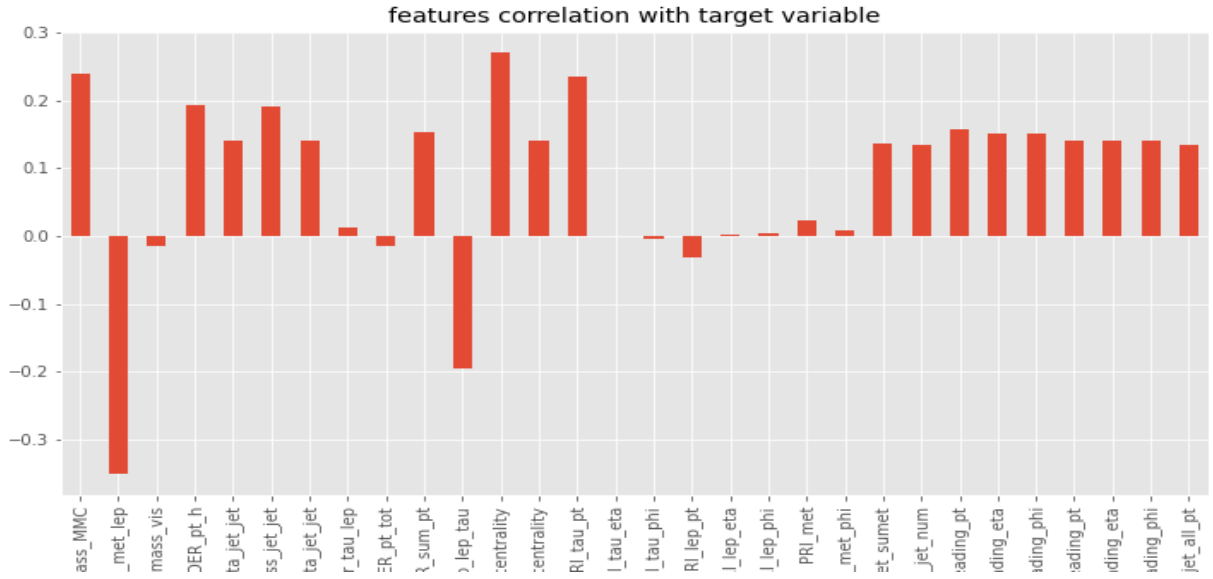


Figure 4 Feature Correlation with Target Variable

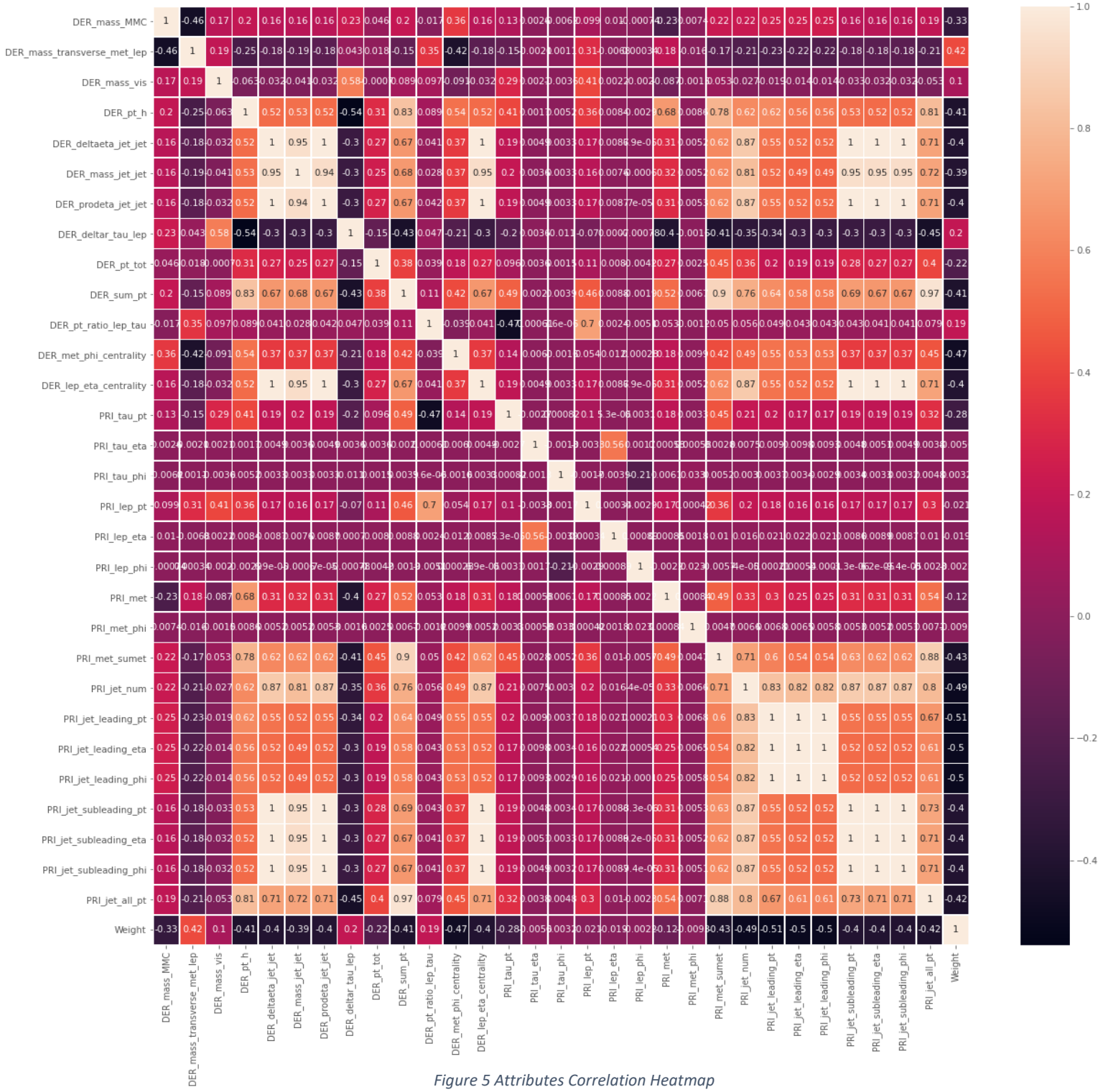


Figure 5 Attributes Correlation Heatmap

3.3 Data Distribution and Domain

In this section of the analysis an in-depth exploration of the features both as occurrence count, to show the exact composition of the dataset in terms of domain and distribution for each feature grouped by class, as well as visualizing the distributions as probability density functions separately for each attribute to be able to compare them.

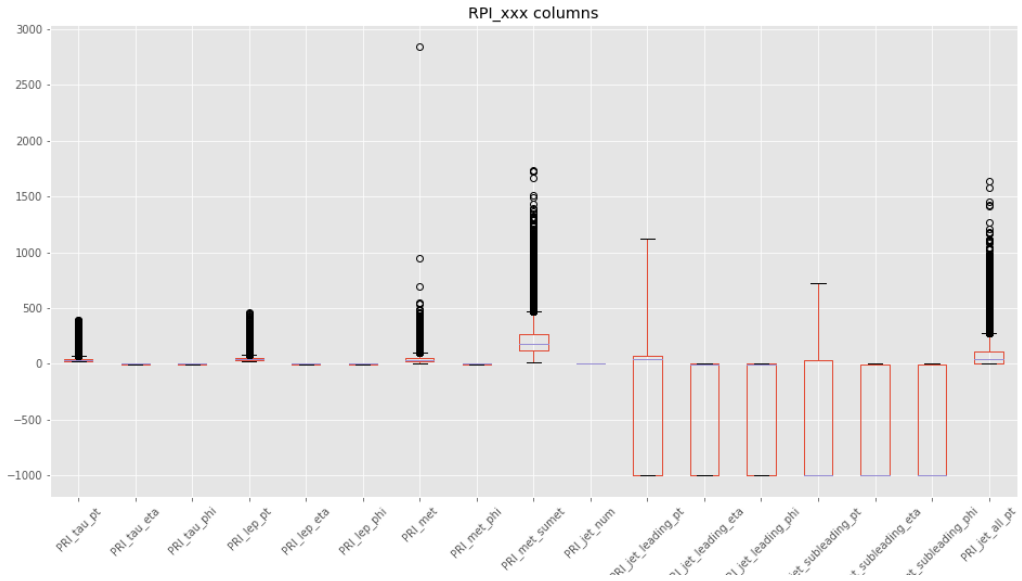


*Figure 6 Box Plot od columns starting with PRI_*

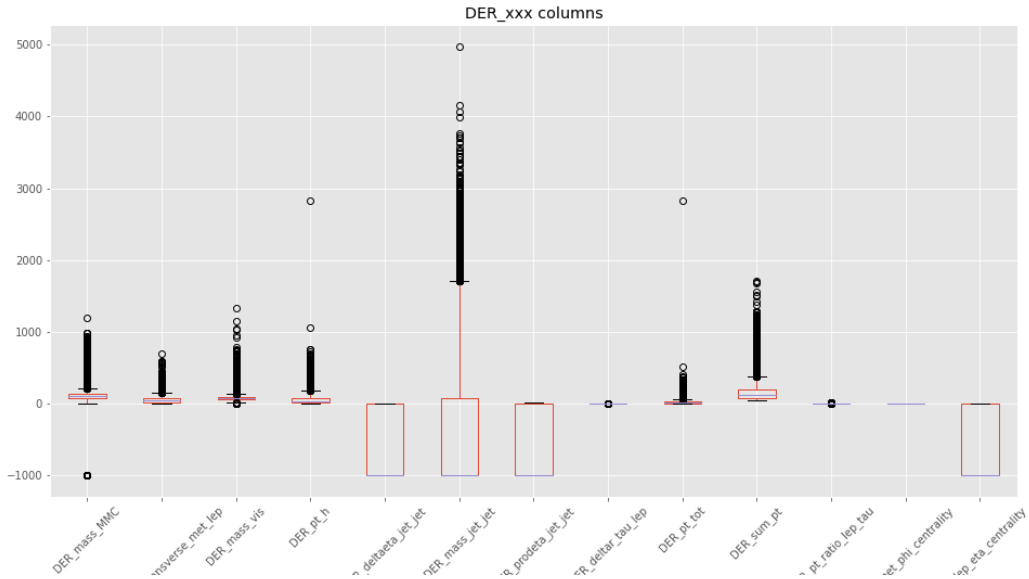


*Figure 7 Box Plot of columns starting with DER_*

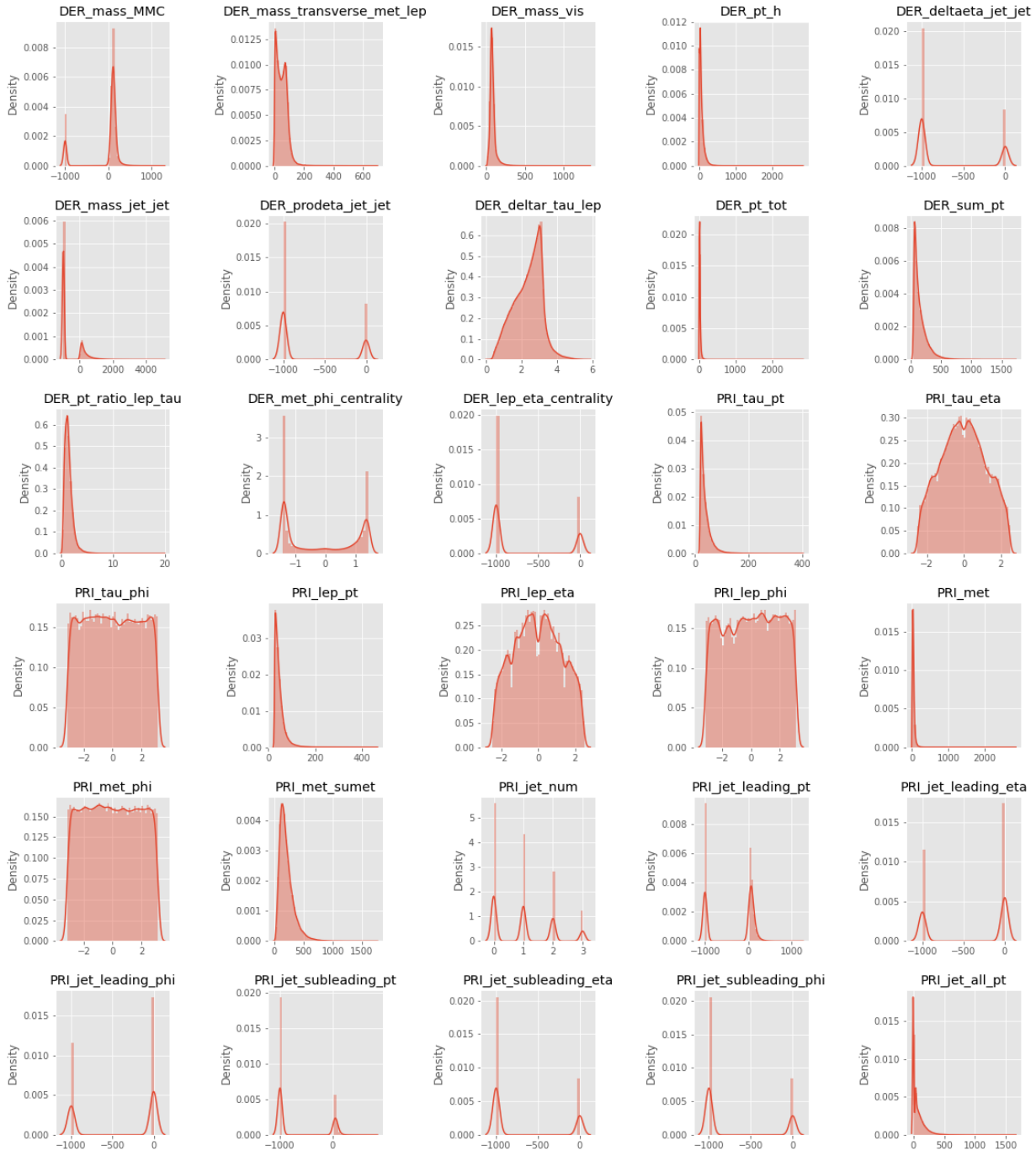


Figure 8 Probability density function of data columns

It can be noticed that some data are affected by the outliers (undefined data) in a way that its distribution is shifted in a high bias toward the value "-999" and that should be solved later in the preprocessing step.

4 Data Processing and Feature Selection

In this section the data will be processed so that it is ready to be tested in various algorithms. For that I need a simple algorithm to help me compare performances of different settings. Normally the first algorithm that pops up the head is the logistic regression. We generally like to begin with logistic regression, as it is extremely common and useful, however, on a first examination of the features, few of them are monotonic in the ratio of signal to background, and therefore the core assumption of logistic regression (that the log odds are linear in the features) appears to be violated.[3]

I could do some simple feature transformations to recover this assumption, but I will just switch to another simple model that can handle these feature distributions: *Naive Bayes*.

Before normalizing the data, outliers (-999) should be dealt with, and features must be selected. For that the following plan will be followed in order:

- First the data will be trained using Naïve Bias and metric visualizations will be done
- Machine learning algorithms will be run over the data to predict alternative values for the outliers and then visualizing the results
- The outliers will be filled with the mean of the column and then visualizing the results

After running the three steps the best method will be chosen from the above 3 steps.

Then comes the step of adding artificial data calculated by Physical experts according to an open source [4]. The credibility of the method cannot be proven but if results improved it can be a boost for the outcome. After that I choose the best configuration from the above three and use it. But because it will add many columns feature selection will be needed. For that, features will be chosen using two ways:

- Visualizing columns density functions for both distinct classes and choose the columns that show most discrimination between the two classes distribution.
- Run principal component analysis (PCA) and tune it.

After all these steps the best configuration of the first and second step will be chosen, and other powerful algorithms will be tested on the processed data in the following section.

For choosing best configuration and visualizing results three metrics will be used:

- ROC curve (Receiver Operating Characteristic)
- AMS loss
- Accuracy

4.1 Metrics

4.1.1 ROC Curve

The ROC curve, or Receiver Operating Characteristic curve, is a metric used to evaluate a classification algorithm's performance. It plots the TPR (true positive rate) as function of FPR

(false positive rate). The area underneath the curve is the true value that specifies the quality of the performance (AUC).

First to put some formulas:

$$\boldsymbol{TPR = \frac{TP}{P}} \qquad \boldsymbol{FPR = \frac{FP}{N}}$$

Where TP stands for the number of truly predicted positives, P for the count of all original positives, FP for the count of the labels that were predicted positive but were negative, and N for the count of all negative labels.

The positive labels are the ones with the label of signal since they were mapped into 1. And the negative labels are the background, labeled to 0.

A theoretically fair ROC curve is over a straight line that has a slope of 1 and thus an area of 0.5, since it has a domain [0,1] and has a maximum value of 1 along the y-axis (TPR-axis). But in practice we aim to achieve much higher values trying to approach the ideal ROC curve that is a vertical line. *Figure 9* demonstrates an example of a ROC curve.

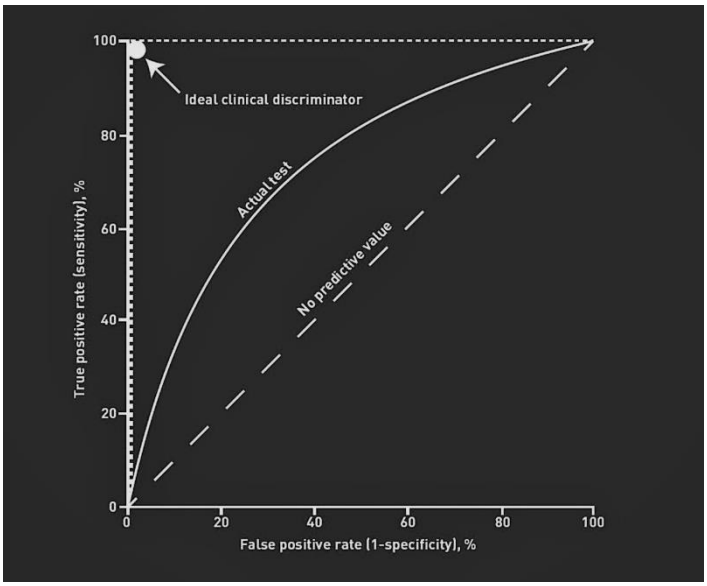


Figure 9 ROC curve demonstration

The ROC curve focuses on the positive (signal) label prediction and thus does not give the equal attention to the negative (background) labels. [1]

There will be a second ROC curve that will consider the weights of the above variables and sum them up. This method was used since the weight plays a significance in terms of labels. It is not present in the test set that its labels should be predicted and submitted on the KAGGLE platform, but still plays an important role in terms of training the classifier and evaluating it, knowing that it cannot be a training attribute as mentioned before.

4.1.2 AMS Loss

It is a loss function that is given by the competition, it can be used as an evaluation metric and can be used in algorithms were the loss function can be specified.

It is given as follows:

$$\boldsymbol{AMS = \sqrt{2(s + b + b_r)\, log\left(1 + \frac{s}{b+b_r}\right) - s}}$$

This loss function is not differentiable, this project will not alter the loss functions of the algorithms but still will study this metric. Therefore, a derived form will be used that is differentiable is differentiable.

$$AMS = \frac{s}{\sqrt{b+b_r}}$$

The first will be ams2 and the second will be ams3.

s: number of correctly assigned signal labels (True Positives)

b: number of true background events classified as signal (False Positives)

There are additional metrics that will be used after selecting the final training algorithm and tuning it. They will be as a final evaluation of the performance. The metrics used now are used due to their significance in specified tasks and have more scientific significance than others.

4.1.3 Accuracy

It is the number of labels predicted correctly over the total number of data instances

$$Accuracy = \frac{TP+TN}{n}$$

TP: True positive, TN: Trues negative, n: Length of the tested dataset

4.2 Naïve Bayes

Bayesian classifiers assign the most likely class to a given example described by its feature vector. Learning such classifiers can be greatly simplified by assuming that features are independent given class, that is:

$$P(X|C) = \prod_{i=1}^{n_i} P(X_i|C)$$

Where:

$$X = (X_1,\ldots, X_i)$$

X is a feature vector and C is a class. Despite this unrealistic assumption, the resulting classifier known as *naive Bayes* is remarkably successful in practice, often competing with much more sophisticated techniques. It is not robust to outliers, but it runs fast and is one of the simplest algorithms for classification.

From the above equation we can derive:

$$P(C|X) = p(X|C)\frac{P(C)}{P(X)}$$

The highest probability of a class knowing X will be the assigned class.

4.3 Dealing with Outliers

First, I will run the algorithm on the data without preprocessing or changing anything, so outliers are kept with the value of -999. The algorithm was set to output probabilities and not labels, by that the threshold can be chosen manually and visualizations are done in a better way. The results are in *Figure 10.*

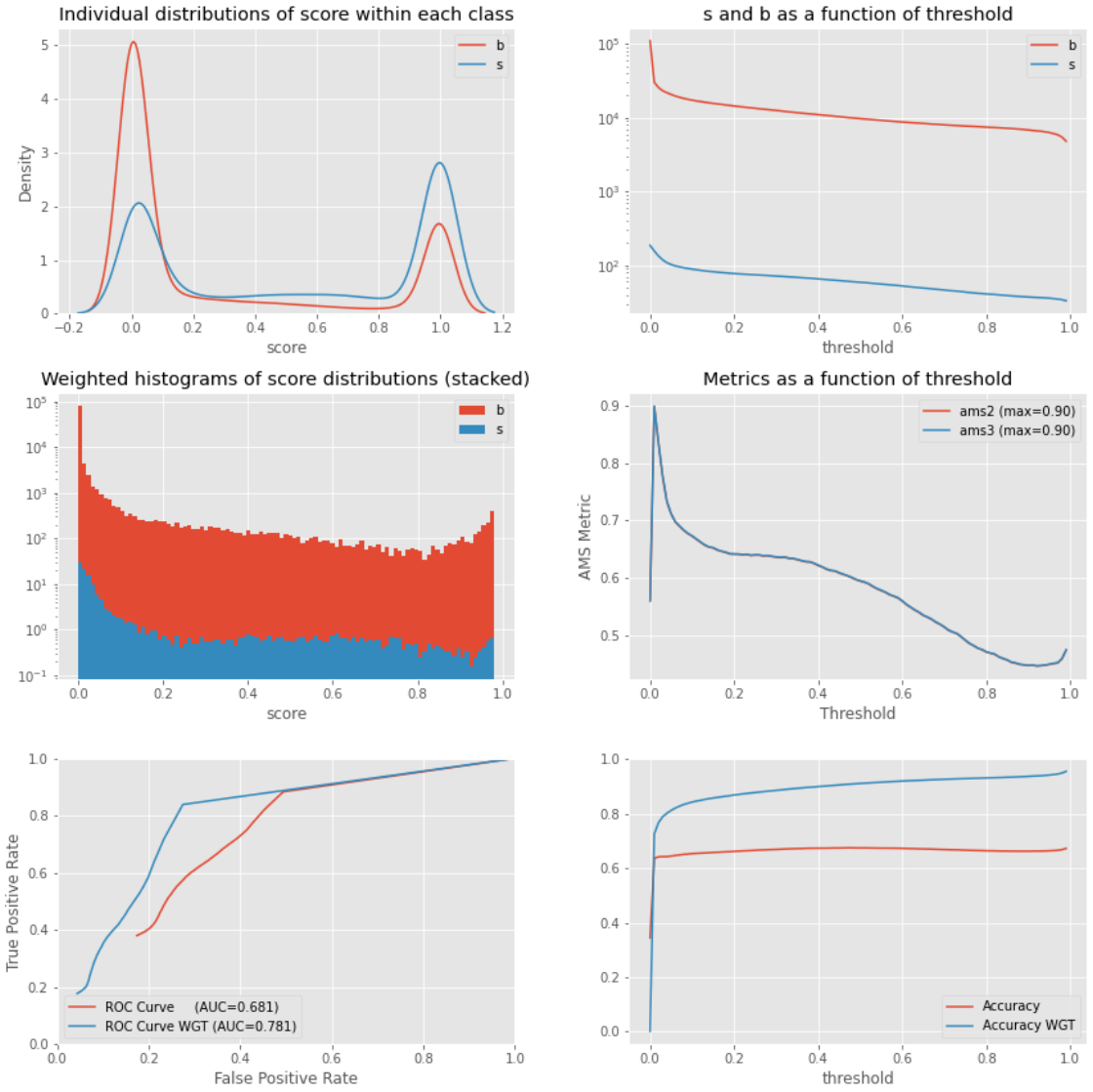


Figure 10 Results of Raw data under Naive Bayes

These are the results that other will be compared to. The first figure on the left and the one underneath it will not be of any great decision impact they just visualize the distribution of the two classes and give an insight on how the output probabilities of the classifier are. The one on the top right is showing the number label count drop with the chosen threshold, also no decision impact but helps in analyzing the classifier's behavior. Well in the *Metrics as function*

of threshold we will see the two AMS errors, the given one that is not differentiable and the approximated one the is differentiable, as function of the threshold showing the maximum value. I should note that a convex loss is not necessary because it is as function of a threshold and not of data columns. For ROC curve and accuracy, the decision impact will be on the non-weighted graph in red, whereas the weighted one in blue will be only for further physical analysis.

4.3.1 Filling Outliers with Smart Algorithms

In this section I will use machine learning algorithms to predict values to fill the outliers. For that task, the three partitioned datasets will be divided as follows:

The 3 sets will be divided into columns that has no missing values to fit the learning algorithm, and into target columns with missing values. The chosen algorithm is the Random Forest Regressor, the algorithm will be explained down in section 5.2. The reason behind choosing Random Forest Regressor is that it is robust to outliers and is one of the best algorithms in regression field.

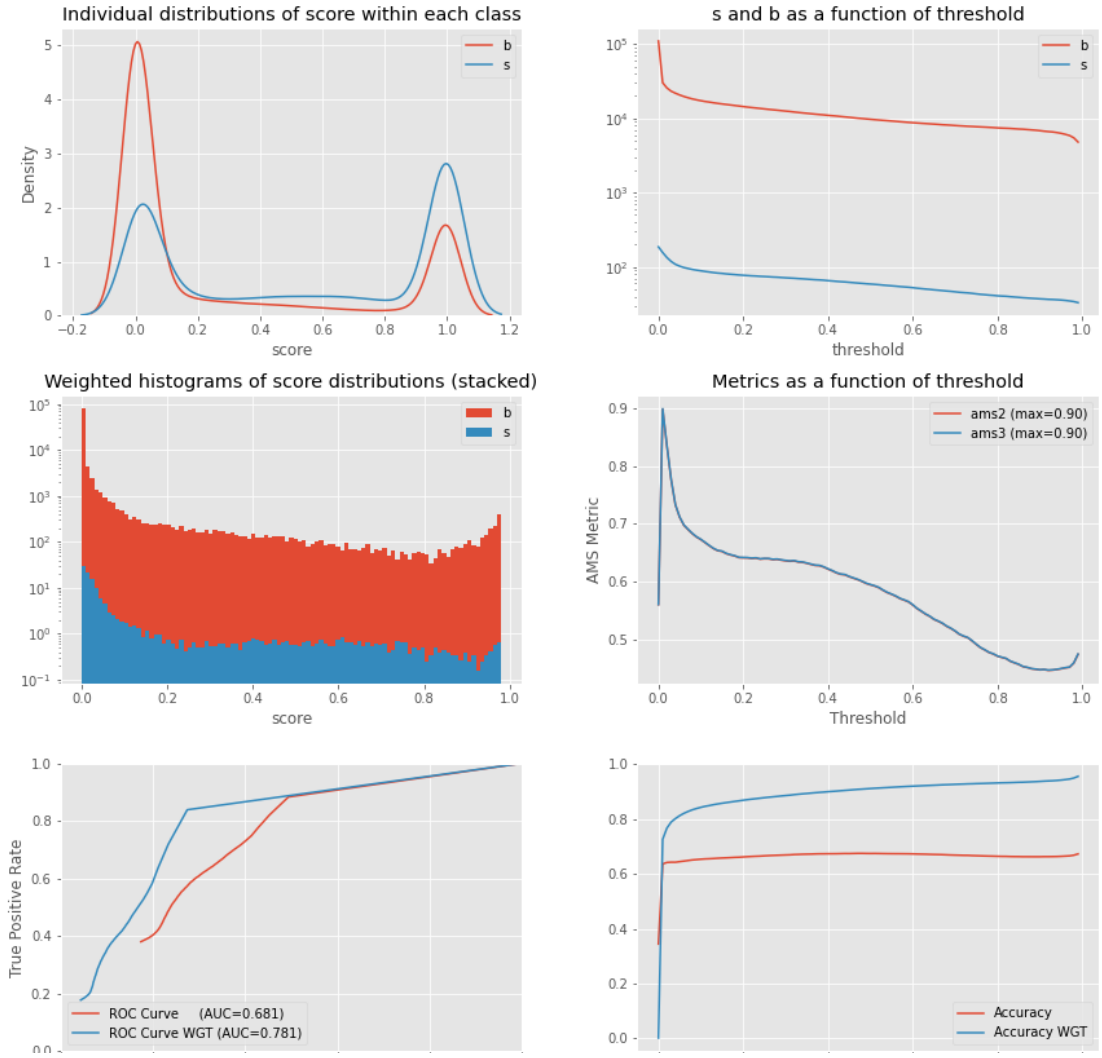


Figure 11 Filling Outliers using Random Forest Regressor

Figure 11 shows the results after applying the process and as we can see there is no much difference in terms of the chosen metrics.

Since this method is computationally expensive it will be dropped and not considered. The next step will be replacing outliers with the mean.

4.3.2 Replacing Outliers with the Mean

For this step, a simple algorithm was fitted. It calculates the mean of each data column dropping the outlier instances and then filling the outliers with the specified mean. This algorithm is not expensive in the computational perspective.

Figure 12 shows that filling with the mean affects the maximum values of ams2 and ams3 and it converges to little higher value from previous results. But in terms of ROC curve the AUC increased from 0.681 to 0.721, this is a significant improvement.

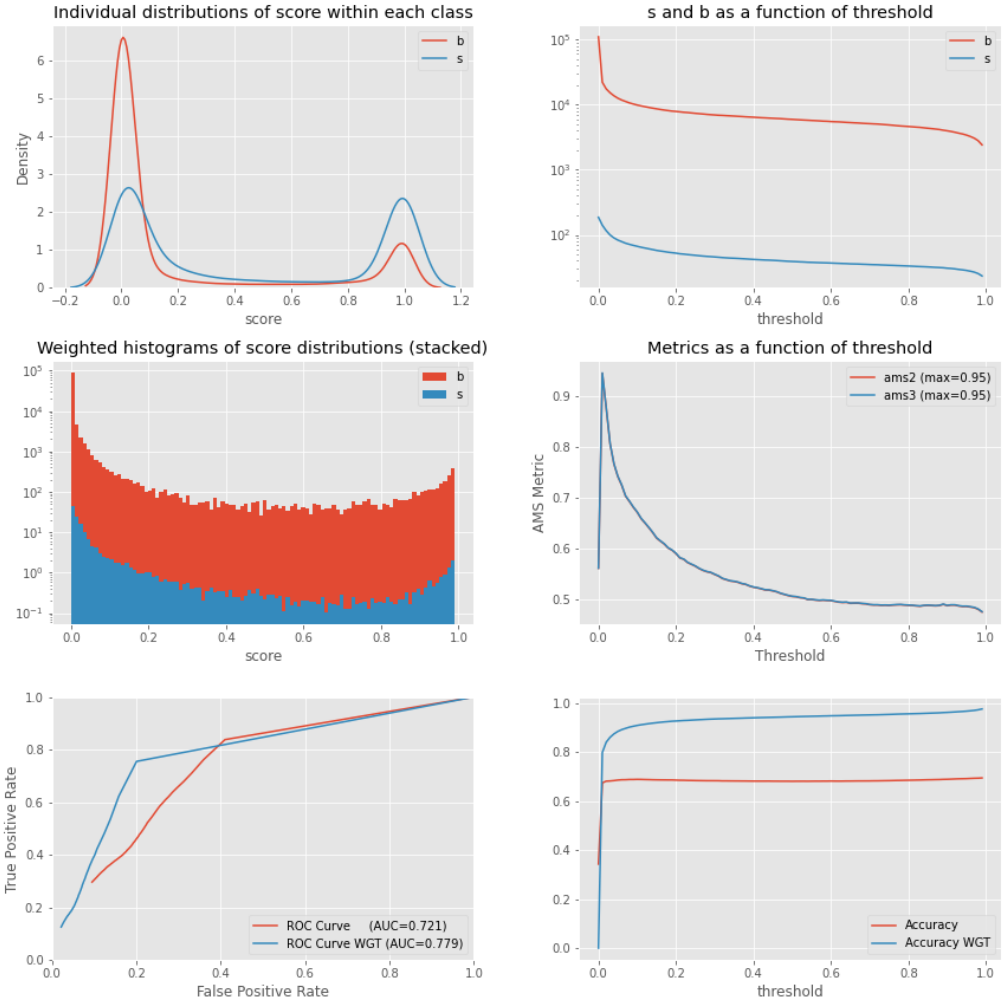


Figure 12 Filling outliers with the Mean

Therefore, Outliers are going to be replaced by the mean in any upcoming algorithm. Now I will move to the next step where I create artificial data.

4.4 Creating Artificial Data and Feature Selection

Our original dataset has 30 feature columns, but according to an open source I was able to find a code that generates artificial data based on domain experts [4]. The credibility cannot be certain but running tests proved that the algorithm works better when using these artificial data.

The new processed dataset will be made of 84 feature column which is a high number of features for that some features should be selected, and others eliminated. The selection of the main features will be done using two approaches:

- Feature selection based on visualizations and a hand-picking approach
- Feature selection based on PCA (principal component analysis)

name	training	evaluation	holdout
size	175000	7500	13500
num_sig	59863	2539	4714
num_bkg	115137	4961	8786
frac	0.342074	0.338533	0.349185
weight	288562	12491.8	22149.4
weight_sig	483.505	20.8146	38.4035
weight_bkg	288079	12470.9	22111
weight_frac_sig	0.00167557	0.00166627	0.00173384

Figure 13 Artificial Data Characteristics

Both approaches will be tested using Naïve Bayes as done previously and same visualizations will be made. The best approach will also follow the best approach in the previous sections (filling outliers with the mean). Then the outcome will be compared with the same settings but without artificial data (section 4.3.2).

At last, we will be ready to choose the very best approach for other algorithms to be tested on.

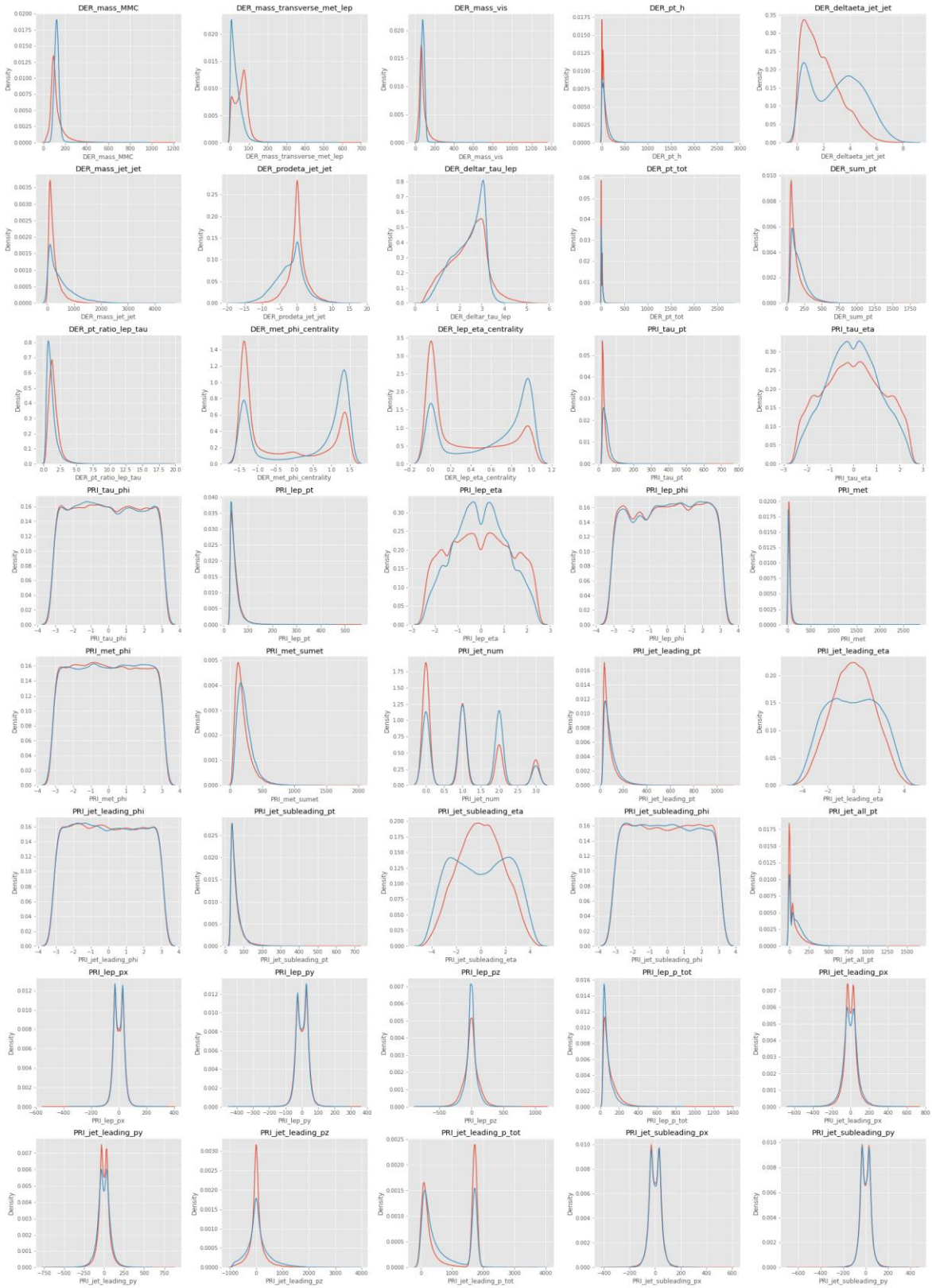
The table in *Figure 13* shows some characteristics of the artificial datasets.

4.4.1 Feature Selection:

In this section I will make probabilistic visualizations of the new data columns that will help me pick the best columns that have discriminative properties between background and signal. It will be based on a naive approach that will consider probability density functions showing the distributions of both classes in each attribute domain, and another one considering the correlation of features with the target variable.

Figure 14 shows the distribution of the two label classes in each attribute domain. We will try to find the columns that show discrimination between the two classes and eliminate attributes that have a close distribution for both classes. And in *Figure 15* we see the

correlation of each attribute with the target variable. I will assist my selection by choosing attributes with a high correlation with the labels.



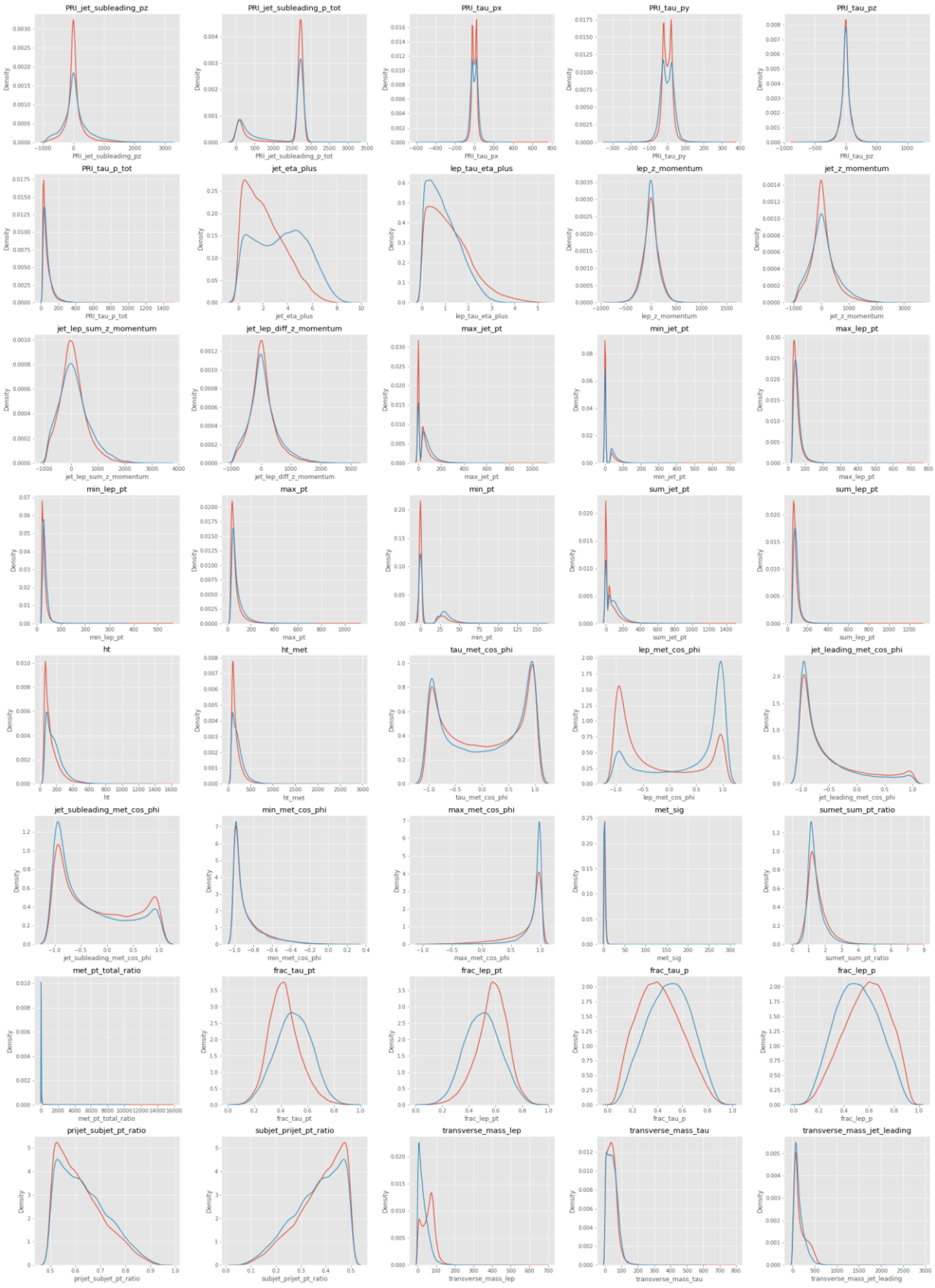


Figure 14 Target Classes Probability Distributions for Each Attribute

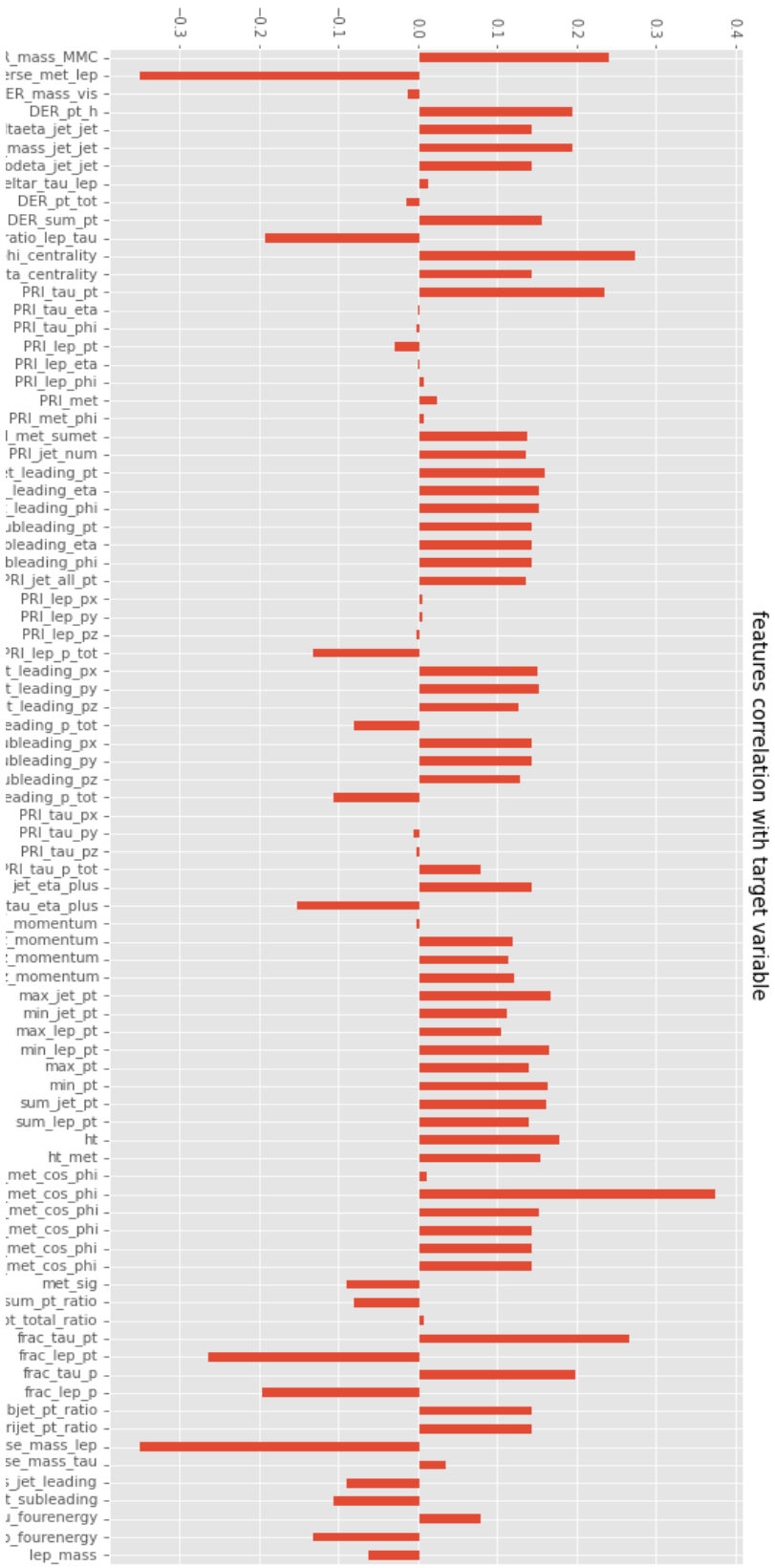


Figure 15 Correlation Between Attributes and Labels

After deep analysis, the following columns were chosen:

```
selected_features = [
    'DER_mass_MMC',
    'DER_mass_transverse_met_lep',
    'DER_mass_vis',
    'DER_pt_h',
    'DER_deltaeta_jet_jet',
    'DER_mass_jet_jet',
    'DER_prodeta_jet_jet',
    'DER_deltar_tau_lep',
    'DER_sum_pt',
    'DER_pt_ratio_lep_tau',
    'DER_met_phi_centrality',
    'DER_lep_eta_centrality',
    'PRI_tau_pt',
    'PRI_tau_eta',
    'PRI_lep_eta',
    'PRI_jet_num',
    'PRI_jet_leading_eta',
    'PRI_jet_subleading_eta',
    'PRI_jet_all_pt',
    'jet_eta_plus',
    'lep_tau_eta_plus',
    'min_pt',
    'lep_met_cos_phi',
    'jet_subleading_met_cos_phi',
    'max_met_cos_phi',
    'frac_tau_pt'
]
```

Running Naïve Bayes gave the results in Figure 16:
We must remember that the algorithm was run with outliers and no action were taken by then to the present outliers. So, comparing it with the initial results we can see a better AUC for the ROC curve, and a very noticeable improvement in the AMS loss. This is enough to ensure that the artificial data are working better than the previous settings.
The feature selection was done so far using no solid mathematical method, for that I will use an algorithm well known for feature reduction in most efficient ways.

4.4.2 Principal Component Analysis (PCA)

"PCA is defined as an orthogonal linear transformation that transforms the data to a new coordinate system such that the greatest variance by some scalar projection of the data comes to lie on the first coordinate (called the first principal component), the second greatest variance on the second coordinate, and so on". Therefore, the goal is to find the best data presentation in the lowest data space without losing the data quality.

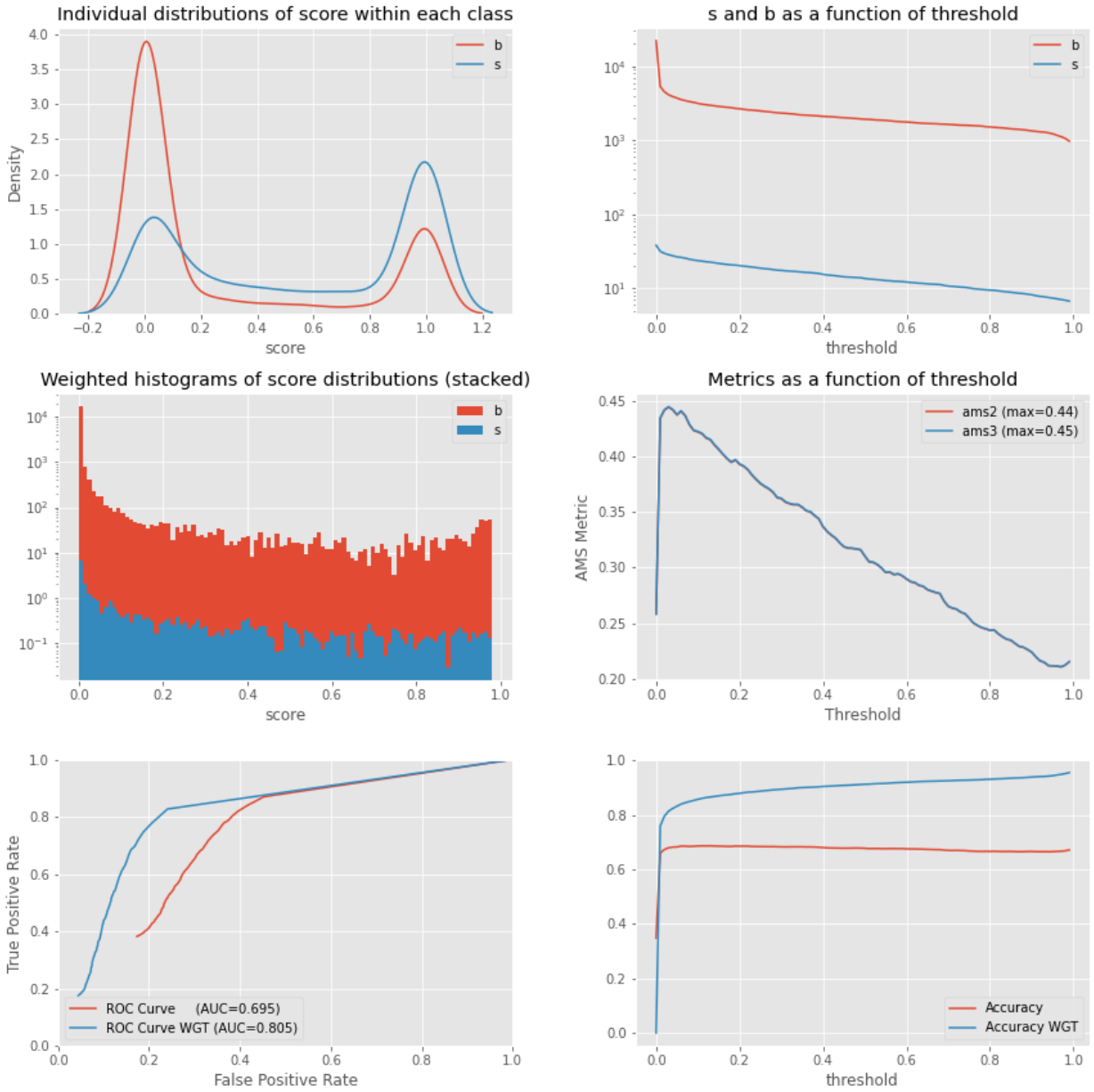


Figure 16 Artificial Data Performance

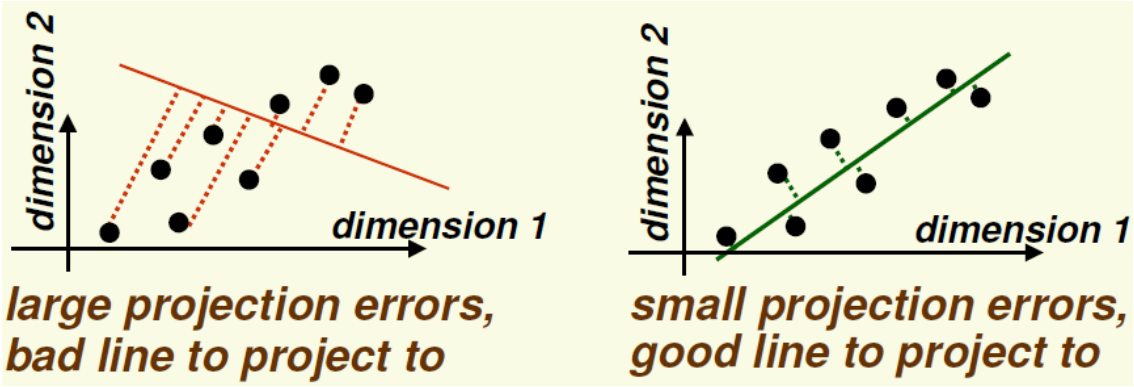


Figure 17 PCA finds the sub-dimension maximizing variance

Figure 17 shows how PCA finds the best plane with lowest projection error.

I ran PCA and tuned it to fit the best lowest number of principal components (35 principal components) without affecting the performance quite much. The results were the best so far, *Figure 18* shows the results.

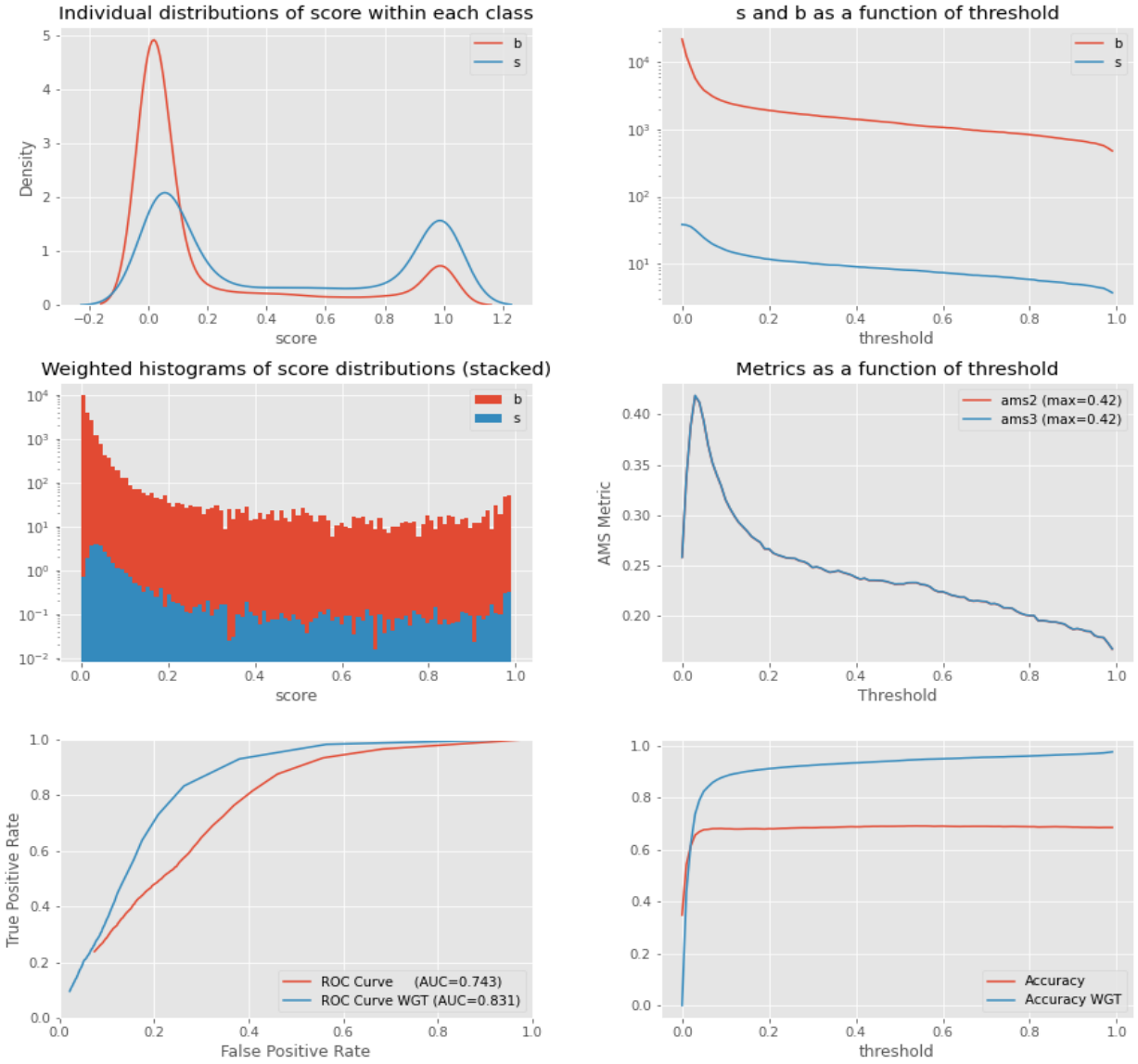


Figure 18 PCA on artificial data results

So, now I have the whole preprocessing approach. And now that I have the preprocessing approach, I should mention that before testing any algorithm I have to normalize the data. The pipeline steps will be as follows:

- Generate artificial data
- Replace outliers by their attribute's mean
- Normalize the data
- Apply dimensionality reduction (PCA)
- Test the data using many algorithms and choose the best performing one.

4.5 Normalizing Data

Data normalization is a preprocessing step that improves the performance of many algorithms. There are many normalization methods, I used the standardization technique.

Standardization aims to centralize the data at the center of the multidimensional space by subtracting the mean of each attribute column from each of its attributes. Then it rescales the data so that the final dataset has a standard deviation of 1.

$$\boldsymbol{Attribute = \frac{Attribute - \mu}{std}}$$

Standardization can drastically improve the performance of models. For instance, many elements used in the objective function of a learning algorithm (such as the RBF kernel of Support Vector Machines or the l1 and l2 regularizers of linear models).

5 Algorithm Selection

In this section I will fit several algorithms and compare results to find the best one that fits the data. The selection of the algorithm will be based on some algorithms covered in the course. After that in the next section I will use other metrics to evaluate the results. The algorithm should be ready by then to be applied on the unlabeled dataset and submitted at the KAGGLE platform.

The tested algorithms will be:

- Logistic Regression
- Random Forest Classifier
- Support Vector Machines (SVMs)

I should note that all the algorithms were set to output a probability and not a label, for the seek of visualizing the results.

5.1 Logistic Regression

Logistic regression is a statistical model that in its basic form uses a logistic function to model a binary dependent variable, although many more complex extensions exist.

In regression analysis, **logistic regression** (or **logit regression**) is estimating the parameters of a logistic model (a form of binary regression).

Mathematically, a binary logistic model has a dependent variable with two possible values, such as pass/fail which is represented by an indicator variable, where the two values are labeled "0" and "1" [3]. As mentioned above it will not be able to perform on our data efficiently, and the data need to

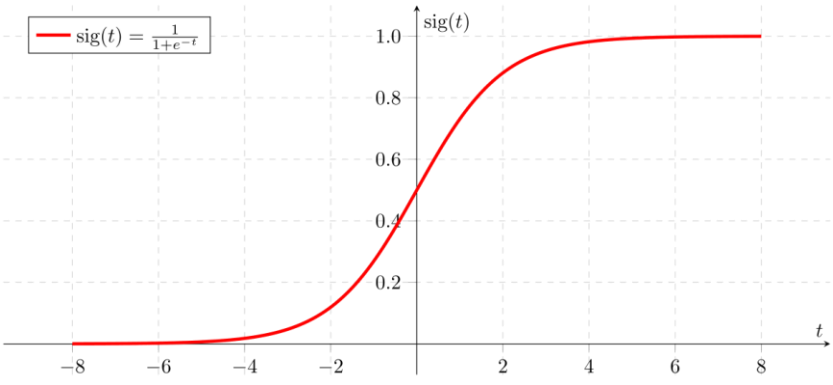


Figure 19 Logistic Regression

be transformed. But it is not a promising algorithm where upcoming ones are way more powerful. But still giving it a try will not affect the process.

Here are some mathematics behind the algorithm:

The logistic regression model transforms a linear regression model into a binary regression model by mapping the linear regression domain $\mathbb{R}$ into a smaller domain [0,1]. This output presents the probability of the labels to belong to class "1".

First, logistic regression uses a sigmoid function that can be seen in *Figure 19*.

$$\sigma(z) = \frac{1}{1+e^{-z}} \qquad z = \sum_{i=1}^{n} \omega_i x_i + b = w.x + b$$

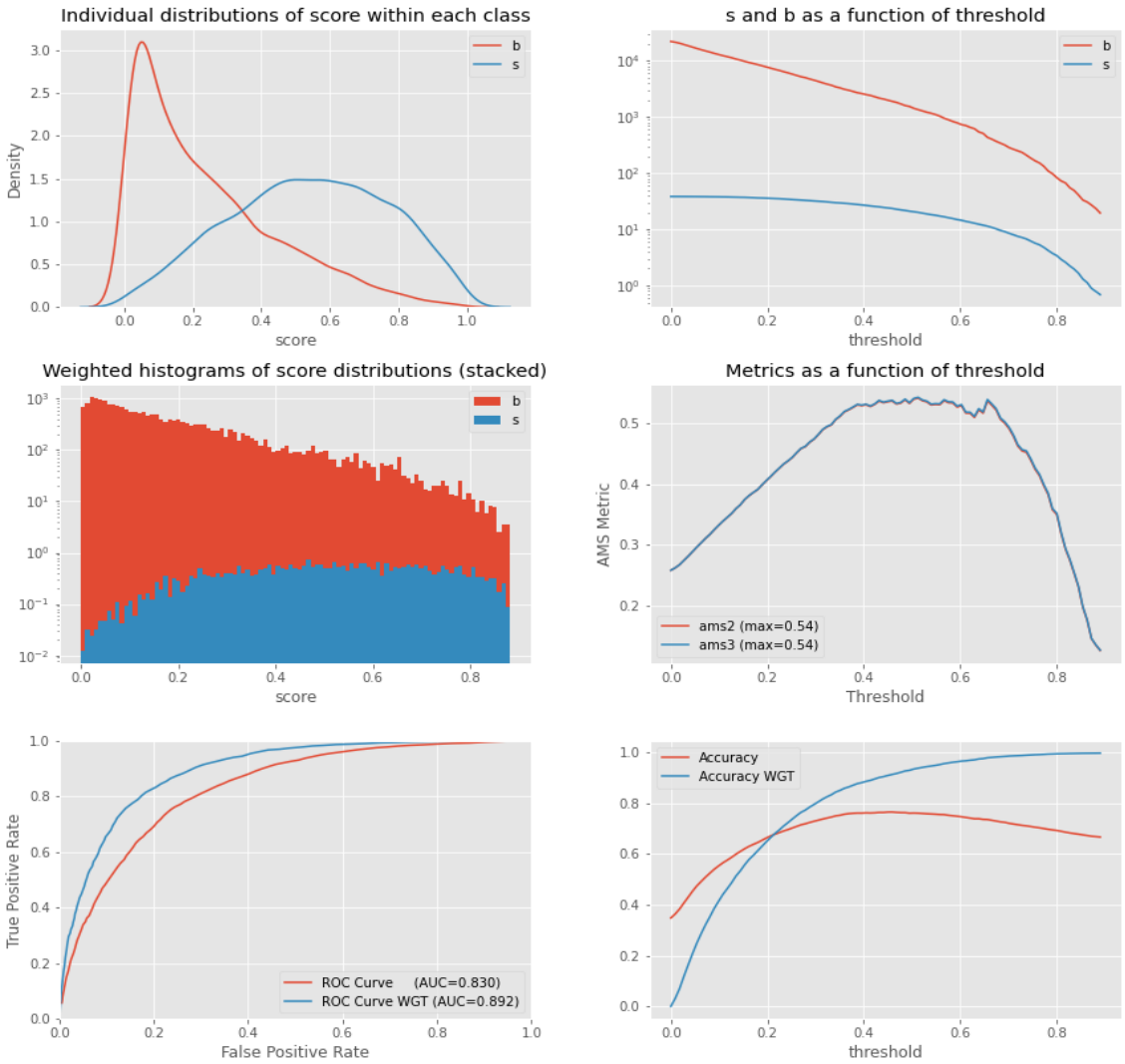


Figure 20 Logistic Regression Results

Where x is the input (dim = #features of the data), w is the weight vector (dim = #features of the data) and z is the typical liner regression model. It fits a mapping:

$F(x) = z$, from $D \rightarrow \mathbb{R}$ (D: domain of the input data)

Then sigmoid maps the output from $\mathbb{R} \rightarrow [0,1]$ as seen in the figure below.

Now the final output:

$$\hat{y} = P(y = 1|x)$$

A threshold ω_0 (0.5 by default) will be used as a decision threshold to assign the class for the input instance.

$$\hat{y} = \begin{cases} 1 & if\ P(y = 1|x) > \omega_0 \\ 0 & if\ P(y = 1|x) > \omega_0 \end{cases}$$

In my model I made the function output probabilities and not labels for better visualizations as function of the threshold, which were the same as previous visualizations. *Figure 20* visualizes the results.

First, we can see that the best threshold for the accuracy was on 0.4, and the distribution of probabilities outputted by the model and their distribution density functions validate that threshold. The accuracy is a little less that 0.8 which is acceptable. Second the AMS was optimized and minimized to a very low value but when testing it on the threshold it was about 0.53. At last, the ROC curve has AUC = 0.83 that is a very good value, and the curve takes a smooth shape. In general, logistic regression was able to perform better than expected.

5.2 Random Forest

Random Forests provide a huge improvement of the simple decision tree by bagging (building several decision trees on bootstrapped training samples) and by choosing a random subset of all predictors as split candidates from the full set of predictors. This decorrelates the trees from one another and reduces the variance when computing averages, resulting in more accurate and robust models. The final prediction is then reached by majority voting among all trees.

All the combinations of the following hyperparameters have been evaluated in hyperparameter tuning:

- "criterion": ["gini", "entropy"]: it is the criterion used to decide a specified split within each step in each tree.

  Gini = 1 - $\sum_{j=1}^{C} p_j^2$ ***Entropy = -*** $\sum_{j=1}^{C} p_j log(p_j)$

- "n_estimators": [25, 50, 100]: it presents the number of the decision trees inside the random forest.
- "max_depth": [None, 10, 15]: presents the maximum depth of each decision tree in the random forest.

Best performing combination: criterion="gini", max_depth=None, n_estimators=100

The results are in *Figure 21*:

First, we can see that the best threshold for the accuracy was on 0.45, and the distribution of probabilities outputted by the model and their distribution density functions validate that threshold. The accuracy is about 0.82 which is better than before. Second the AMS was optimized and minimized to a very low value but when testing it on the threshold it was about

0.61. At last, the ROC curve has AUC = 0.876 that is a very good value, and the curve takes a smooth shape. In general, Random Forest Regressor performed quite well.

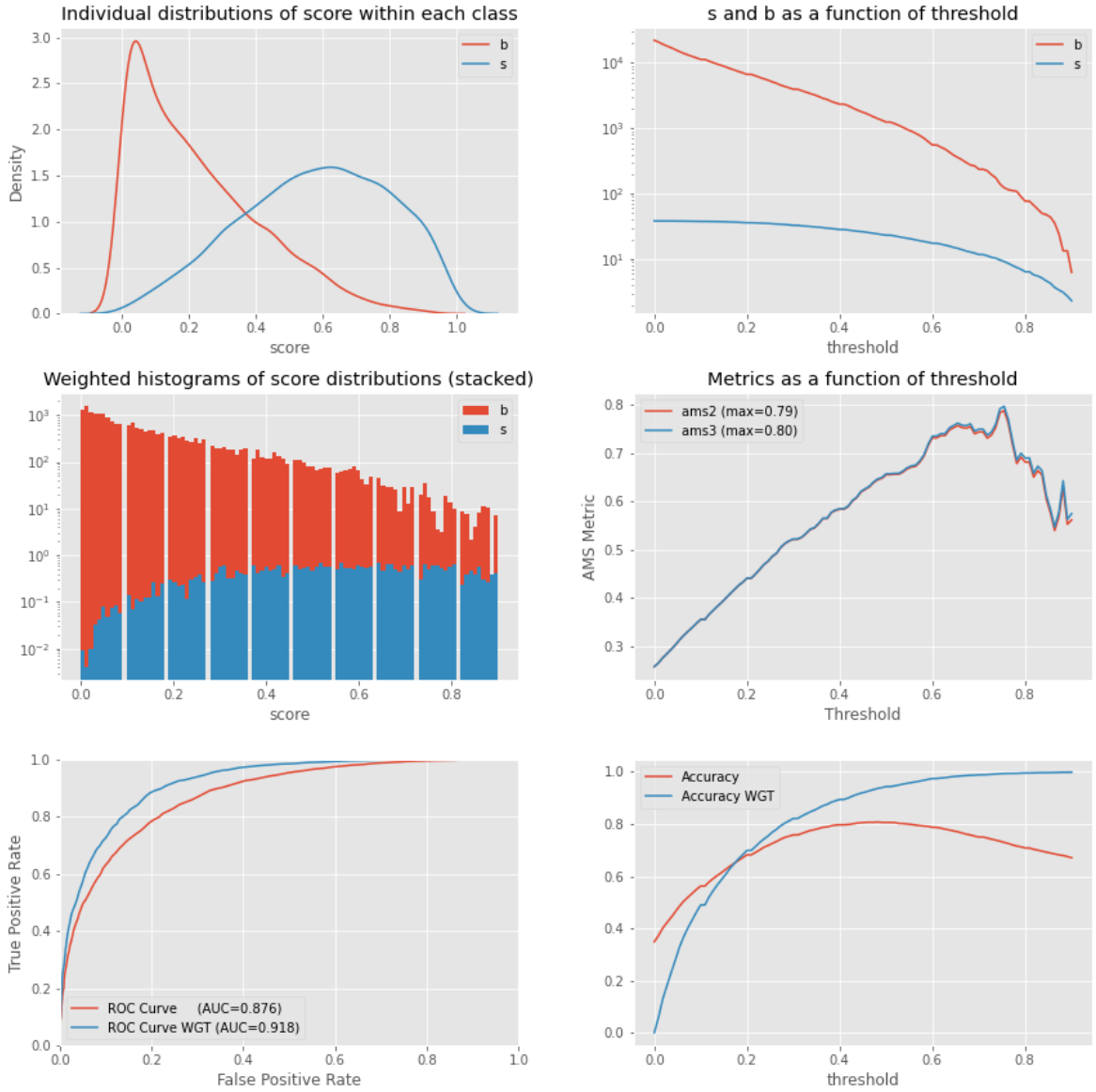


Figure 21 Random Forest Results

5.3 Support Vector Machine (SVM)

SVM is a supervised method that solves an optimization problem: Find the hyperplane that maximizes the margin between two classes in the feature space. More specially, the distance is computed between the margin and the closest data point of each class, that are called support vectors.

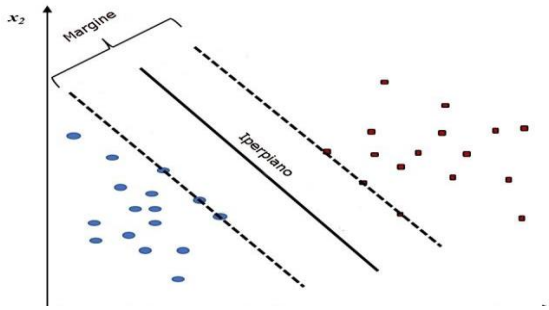


Figure 22 Linear Support Vector Machine

Given a hyperplane:

$$f(x) = \beta_0 + \sum_{i=1} \beta_i X_i$$

We want to find the β that maximizes the margin M such that $y_i f(x_i) > 0 \; \forall i$.
It often happens that data are not linearly separable solving the hard margin problem. One of the alternative is the soft margin problem: We introduce slack variables and allow for some mistakes. The hyperparameter C is the cost of misclassification, when C is large, the algorithm tries to maximize the number of points correctly classified while a smaller C will encourage a larger margin at the cost of training accuracy. *Figure 23* gives a brief example.

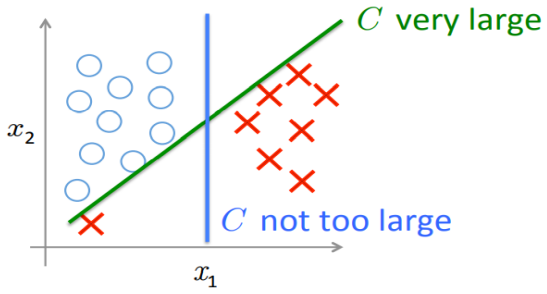


Figure 23 Value of C influence

$$\min_{\omega} \frac{1}{2}\omega^T\omega + C\sum_i max\,(0, 1 - y_i\omega^T x_i)$$

One of the strongest points of support vector machines is that they can be used with kernels. Kernels are symmetric functions that correspond to a scalar product in a higher dimensional features space. This solution is much more efficient and easier than computing the mapping directly. Hence, the hyperplane is computed in the new space where data could become linearly separable. [5]

$$K(x'_1{}^x) = \langle \phi(x), \phi(x') \rangle$$

According to the Mercer Theorem a kernel is valid if it is symmetric and the corresponding gram matrix $G_{ij} = K(x_i, x_j)$ is positive semidefinite. Some of the most popular kernels are rbf (radial basis function) that compute the distance using a gaussian curve, polynomial and sigmoid.

Gaussian: $exp\left(-\frac{\|x-x'\|^2}{2\sigma}\right)$
Linear: $(1 + \langle x, x' \rangle)^2$
RBF: $tanh(\gamma x^T x' + r)$

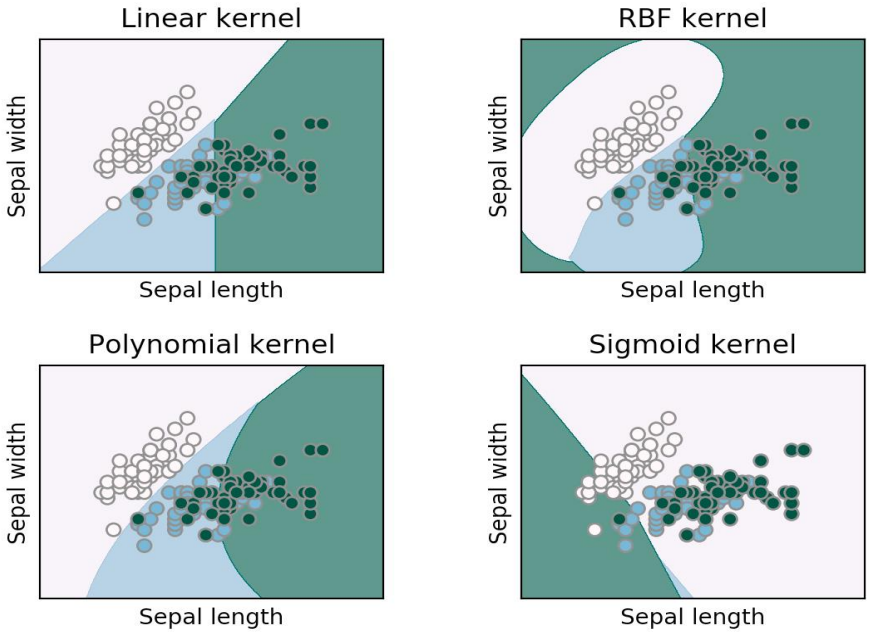


Figure 24 Examples of Kernel Functions

Figure 24 is an example of behavior of the different kernels on the IRIS dataset, visualizing the decision boundaries.
All the combinations of the following hyperparameters have been evaluated in hyperparameter tuning:

- “C”
- “kernel”
- “gamma”

Best performing combination: C=10, gamma="0.01", kernel="rbf". Running over the whole dataset with 35 feature columns after PCA was computationally unfeasible to run locally or on google Collaboratory (predicting probabilities is very expensive but if the algorithm shows promising results it will be trained to find labels, by this it should converge without decreasing the dataset). Therefore, features were decreased, but still it was not converging, thus I left 35 columns and trained on a portion of the data (#instances = 90000). Results are in *Figure 25*.

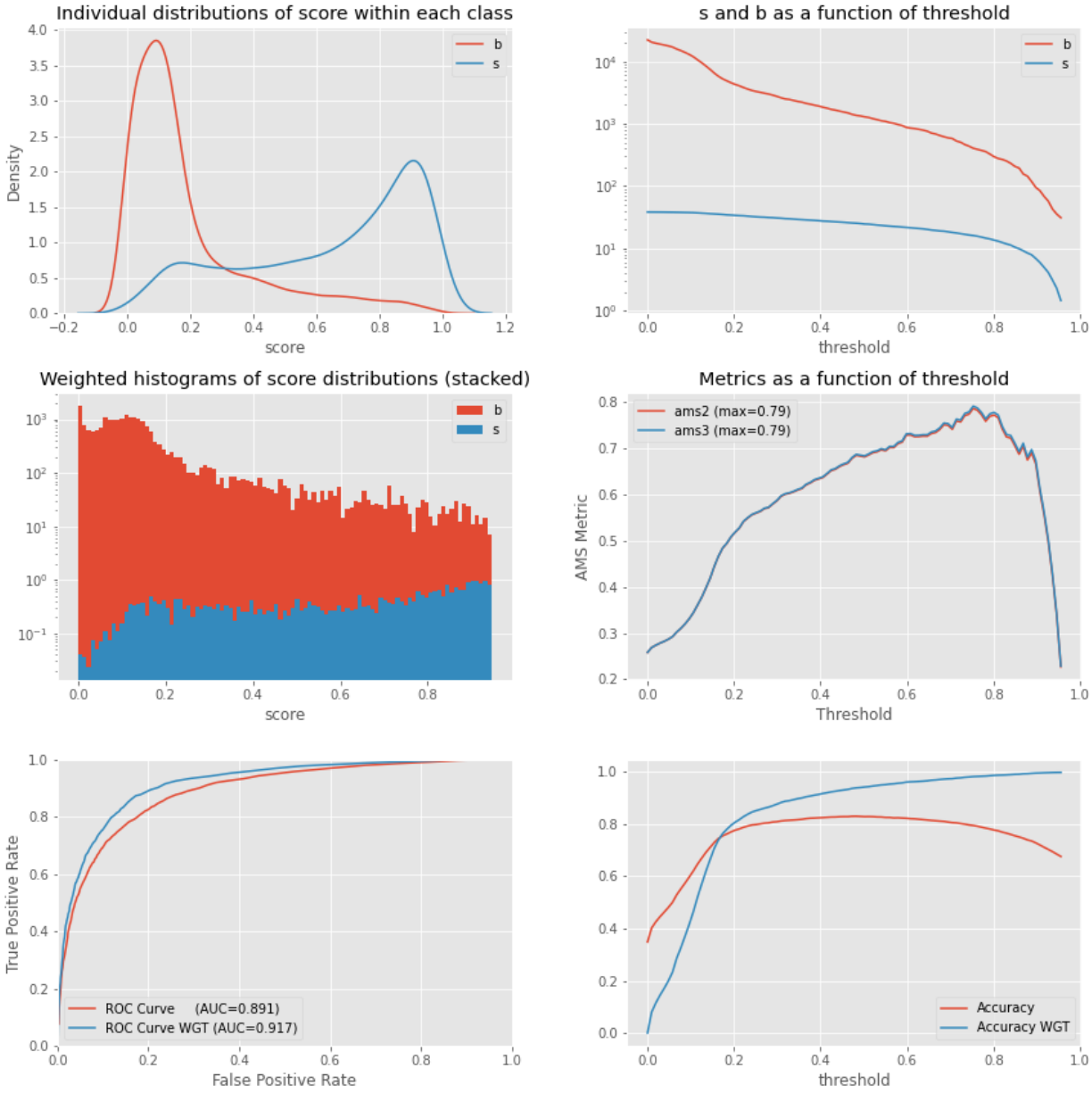


Figure 25 SVM Results

First, we can see that the best threshold for the accuracy was on 0.42, and the distribution of probabilities outputted by the model and their distribution density functions validate that threshold. The accuracy is about 0.83 which is better than before. Second the AMS was optimized and minimized to a very low value but when testing it on the threshold it was about 0.62. At last, the ROC curve has AUC = 0.891 that is a very good value, and the curve takes a smooth shape. In general, Random Forest Regressor performed quite well.

We can now validate that the support vector machine had the best performance.

Now after selecting the best algorithm and best settings after the data has been processed, I will evaluate the model.

6 Results

In this section I will rerun the model but set it to output labels and not probabilities, by that the algorithm should be able to fit the whole data and converge without the need of a hosting server. Then I will implement the confusion matrix and discuss the results quickly before moving to the next step where I discuss how to run the model over the non-labeled data to be submitted on the KAGGLE platform.

6.1 Confusion Matrix

The confusion matrix, also known as an error matrix, is a specific table layout that allows visualization of the performance of an algorithm, typically a supervised learning one (in unsupervised learning it is usually called a matching matrix), *Figure 26*. Each row of the matrix represents the instances in an actual class while each column represents the instances in a predicted class, or vice versa – both variants are found in the literature.

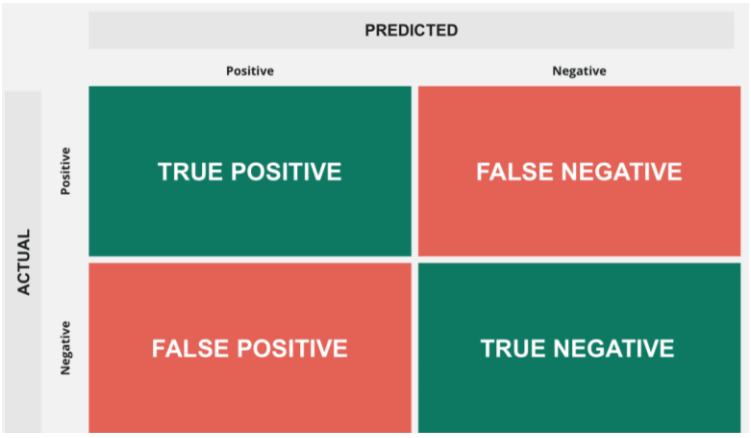


Figure 26 Confusion Matrix Demonstration

From the confusion matrix we can calculate the Precision, Recall, and Accuracy. But before lets mark that the following values are short for:

TP: True Positive, FN: False Negative, FP: False Positive, and TN: True Negative.

$$\textbf{Accuracy} = \frac{TP+TN}{TP+TN+FP+FN}$$

$$\textbf{Precision} = \frac{TP}{TP+FP}$$

$$\textbf{Recall} = \frac{TP}{TP+FN}$$

Precision-Recall score is a useful measure of success of prediction when the classes are very imbalanced. Accuracy score is used to measure the model performance in terms of measuring the ratio of sum of true positive and true negatives out of all the predictions made.

Precision tests how good is the classifier in classifying the positive labels correctly, recall measures how confident the classifier is when classifying positive labels.

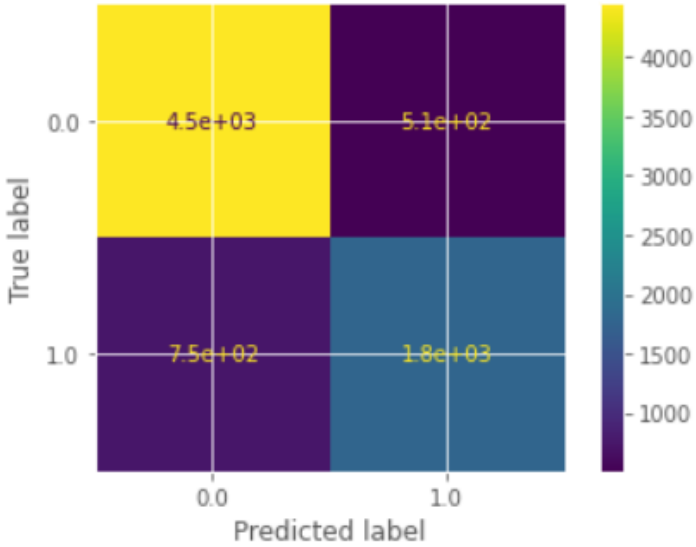


Figure 27 SVM Results Confusion Matrix

Accuracy = 0.833

Precision = 0.856

Recall = 0.898

Results are good in *Figure 27* but could be improved if other algorithms are used like Neural Networks, and Gradient boosted trees.

7 How to use the Model for the Unlabeled Data

After selecting the model and evaluating it I have the best performing model according to my pipeline. But now I will not continue, and I leave the doors open for those who want to run the model over the unlabeled data and submit it on the KAGGLE platform. I still need to add some notes before.

7.1 Domain Shift

The unlabeled dataset seemed to be different in the probabilistic distribution from the training data. In *Figure 28* I used the TSNE model to transform my data into lower dimensions for visualizing it. It was quite noticeable that there is a shift in the probabilistic distribution.

This might affect the model's performance drastically decreasing its efficiency. For that, naively running the model over the data will not help no matter how much the algorithm is robust. There are two solutions:

- Apply data transformations
- Run domain adaptation techniques (Adversarial learning)

I will not go into details here since it is not part of my course, therefore I can leave the doors open for more discussion in this field.

7.2 TSNE

t-distributed stochastic neighbor embedding (t-SNE) is a statistical method for visualizing high-dimensional data by giving each datapoint a location in a two or three-dimensional map. It is based on Stochastic Neighbor Embedding.

The t-SNE algorithm comprises two main stages. First, t-SNE constructs a probability distribution over pairs of high-dimensional objects in such a way that similar objects are assigned a higher probability while dissimilar points are assigned a lower probability. Second, t-SNE defines a similar probability distribution over the points in the low-dimensional map, and it minimizes the Kullback–Leibler divergence (KL divergence) between the two distributions with respect to the locations of the points in the map. While the original algorithm uses the Euclidean distance between objects as the base of its similarity metric, this can be changed as appropriate. [6]

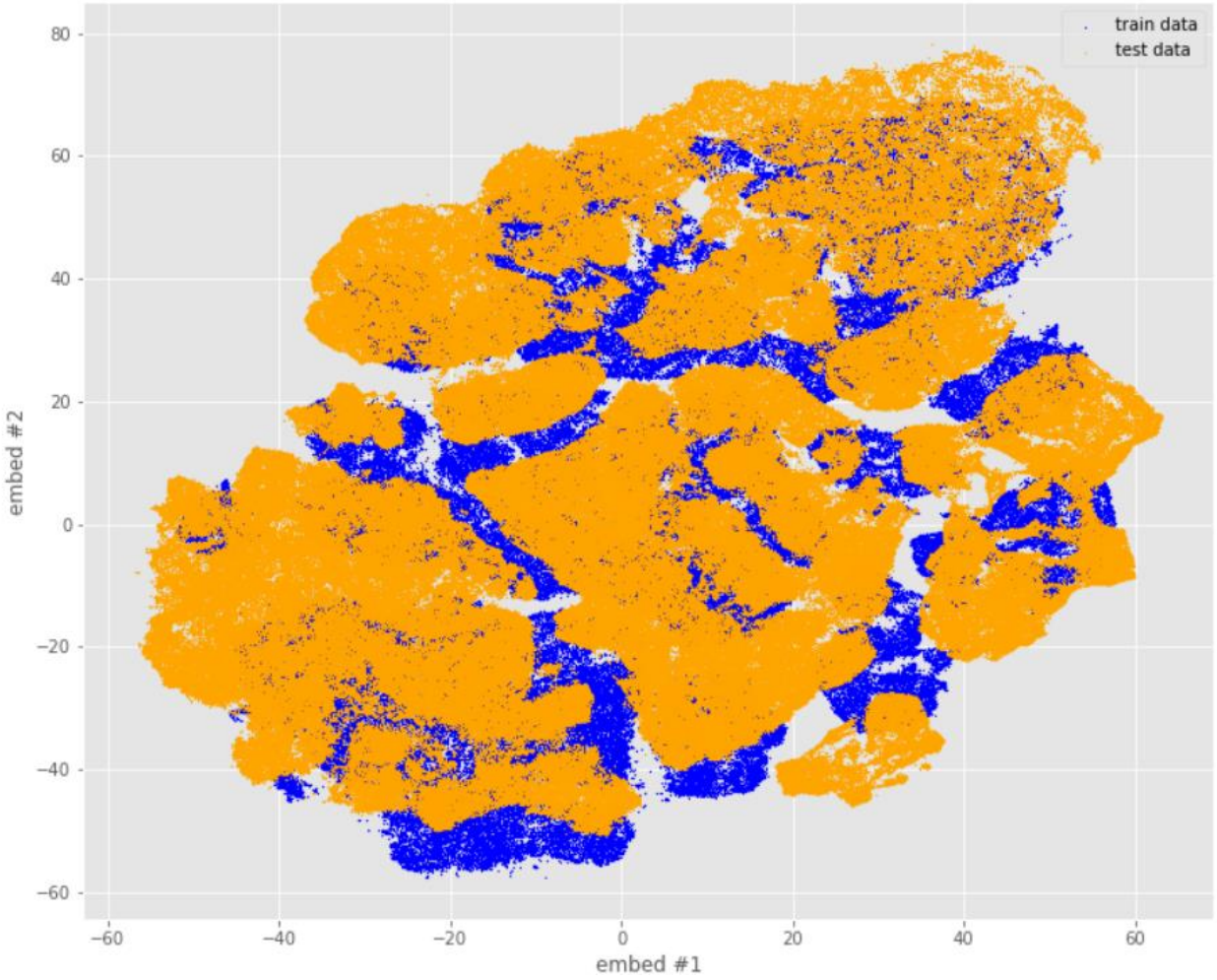


Figure 28 Domain Shift Between Train and Test Sets

8 Conclusion

Machine learning span is getting wider and wider, and we just now saw it is a major part in understanding physical properties of some fundamental particles of physics. The Higgs Boson challenge was a very successful collaboration between physics and data science.

And finally, as we have seen, the performances of all the algorithms were quite close. But support vector machine was the best performing one.

References

[1] Hoo ZH, Candlish J, Teare D What is an ROC curve? *Emergency Medicine Journal* 2017;**34:**357-35

[2] Claire Adam-Bourdariosa, G. C. (2014, July 21). Learning to discover: the Higgs boson machine learning challenge.

[3]

Retrieved from Wikipedia: https://en.wikipedia.org/wiki/Logistic_regression

[4] ghl3. *GitHub*. Retrieved from https://github.com/ghl3/higgs-kaggle/blob/master/notebooks/modeling.ipynb

[5] Retrieved from Wikipedia: Support-vector machine - Wikipedia

[6] Retrieved from Wikipedia: t-distributed stochastic neighbor embedding - Wikipedia