Politecnico di Torino

1859

Mathematics in Machine Learning
MAGIC Gamma Telescope Data Set

Mohamad Mostafa

s291385

2020/2021

Contents

1. Introduction:

Identifying the gamma ray events from background hardon signals in gamma-ray Cherenkov telescope is the one of the important issues to progress detector technology in astronomy. The data set is MAGIC, a Cherenkov telescope experiment taken from an online machine learning site called UCI. The data has got two classes, i.e., the image originating either from incident gamma rays or caused by hadronic showers. The task is to classify gamma signals from hadron signals representing a rare class classification problem. In this contribution, we present a classification technique that classifies the data and gives the result as good as other classification techniques with great ease and fast.

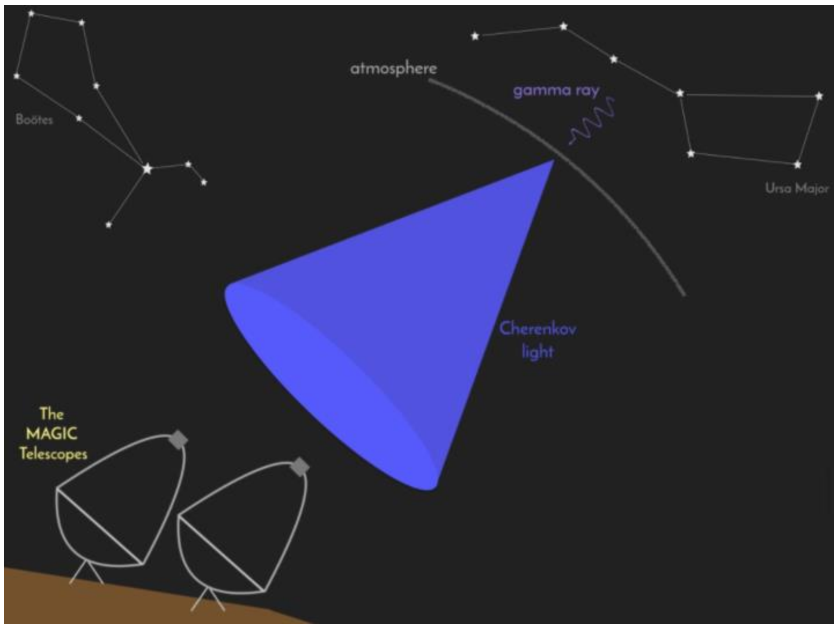


Figure 1. Illustration of gamma rays detection by Magic Telescope.

2. Data Exploration:

The dataset is composed of 19020 instances each of which is associated with its corresponding characteristics where all our attributes are numerical, and the target variable is binary:

Predictors:

1. fLength: continuous # major axis of ellipse [mm].
2. fWidth: continuous # minor axis of ellipse [mm].
3. fSize: continuous # 10-log of sum of content of all pixels [in #phot].
4. fConc: continuous # ratio of sum of two highest pixels over fSize [ratio].
5. fConc1: continuous # ratio of highest pixel over fSize [ratio].
6. fAsym: continuous # distance from highest pixel to center, projected onto major axis [mm].
7. fM3Long: continuous # 3rd root of third moment along major axis [mm].
8. fM3Trans: continuous # 3rd root of third moment along minor axis [mm].
9. fAlpha: continuous # angle of major axis with vector to origin [deg].
10. fDist: continuous # distance from origin to center of ellipse [mm].

Target:

1. class: g, h # gamma (signal), hadron (background).

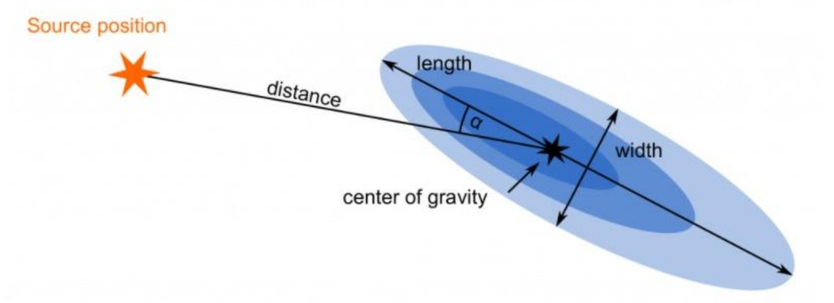
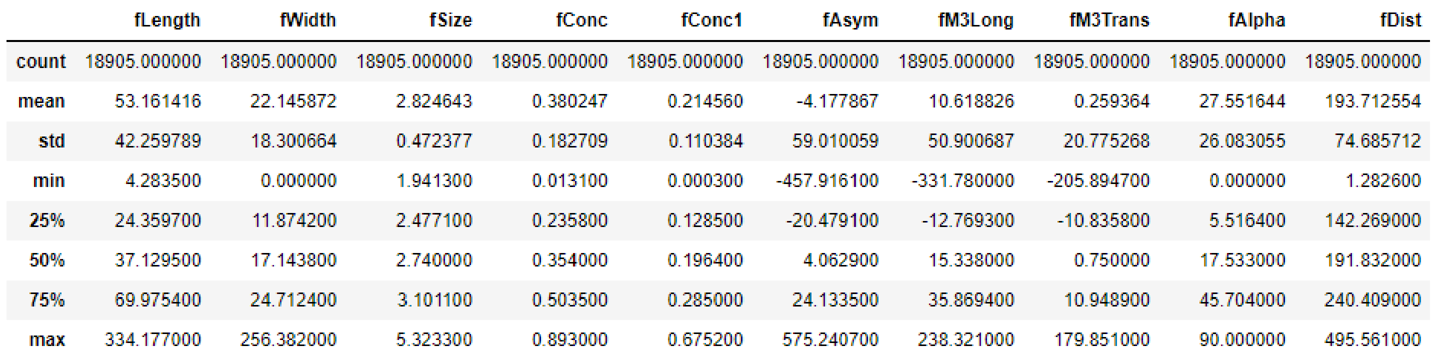


Figure 2. Visual understanding of the variables

2.1. Statistical Overview:

For each attribute we report the count, mean, standard deviation, minimum and quartiles.



	fLength	fWidth	fSize	fConc	fConc1	fAsym	fM3Long	fM3Trans	fAlpha	fDist
count	18905.000000	18905.000000	18905.000000	18905.000000	18905.000000	18905.000000	18905.000000	18905.000000	18905.000000	18905.000000
mean	53.161416	22.145872	2.824643	0.380247	0.214560	-4.177867	10.618826	0.259364	27.551644	193.712554
std	42.259789	18.300664	0.472377	0.182709	0.110384	59.010059	50.900687	20.775268	26.083055	74.685712
min	4.283500	0.000000	1.941300	0.013100	0.000300	-457.916100	-331.780000	-205.894700	0.000000	1.282600
25%	24.359700	11.874200	2.477100	0.235800	0.128500	-20.479100	-12.769300	-10.835800	5.516400	142.269000
50%	37.129500	17.143800	2.740000	0.354000	0.196400	4.062900	15.338000	0.750000	17.533000	191.832000
75%	69.975400	24.712400	3.101100	0.503500	0.285000	24.133500	35.869400	10.948900	45.704000	240.409000
max	334.177000	256.382000	5.323300	0.893000	0.675200	575.240700	238.321000	179.851000	90.000000	495.561000

2.2. Visualization:

Before any further analysis and visualization of the data we check that no missing data exist, but there exist 115 duplicate record which is probably a consequent to the way the data was generated, rather than an indication of frequently occurring patterns. Duplicates may cause a bias in our model. For this reason, we decide to discard them from our analysis keeping only the first occurrence of each object leading to 18905 distinct observations. Moving on to the visualization of the target variable as we see in Fig. 3, we notice that the data is distributed having 12332(65.23%) of total data belonging to the gamma class and the remaining 6573(34.76%) records belonging to the hadron class showing some imbalance of the data which will be discussed in later section.

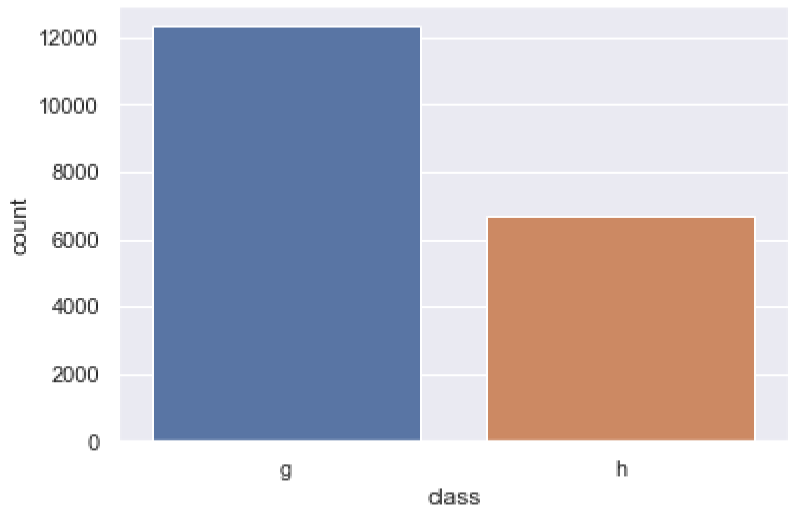


Figure 3. Distribution of the target variable

Regarding the predictors we decide to use the boxplot representation to represent their distribution due to the fact that our predictors are all numerical and it is shown in Fig. 4. We also notice from fAlpha attribute the significant difference in its mean corresponding to each class.

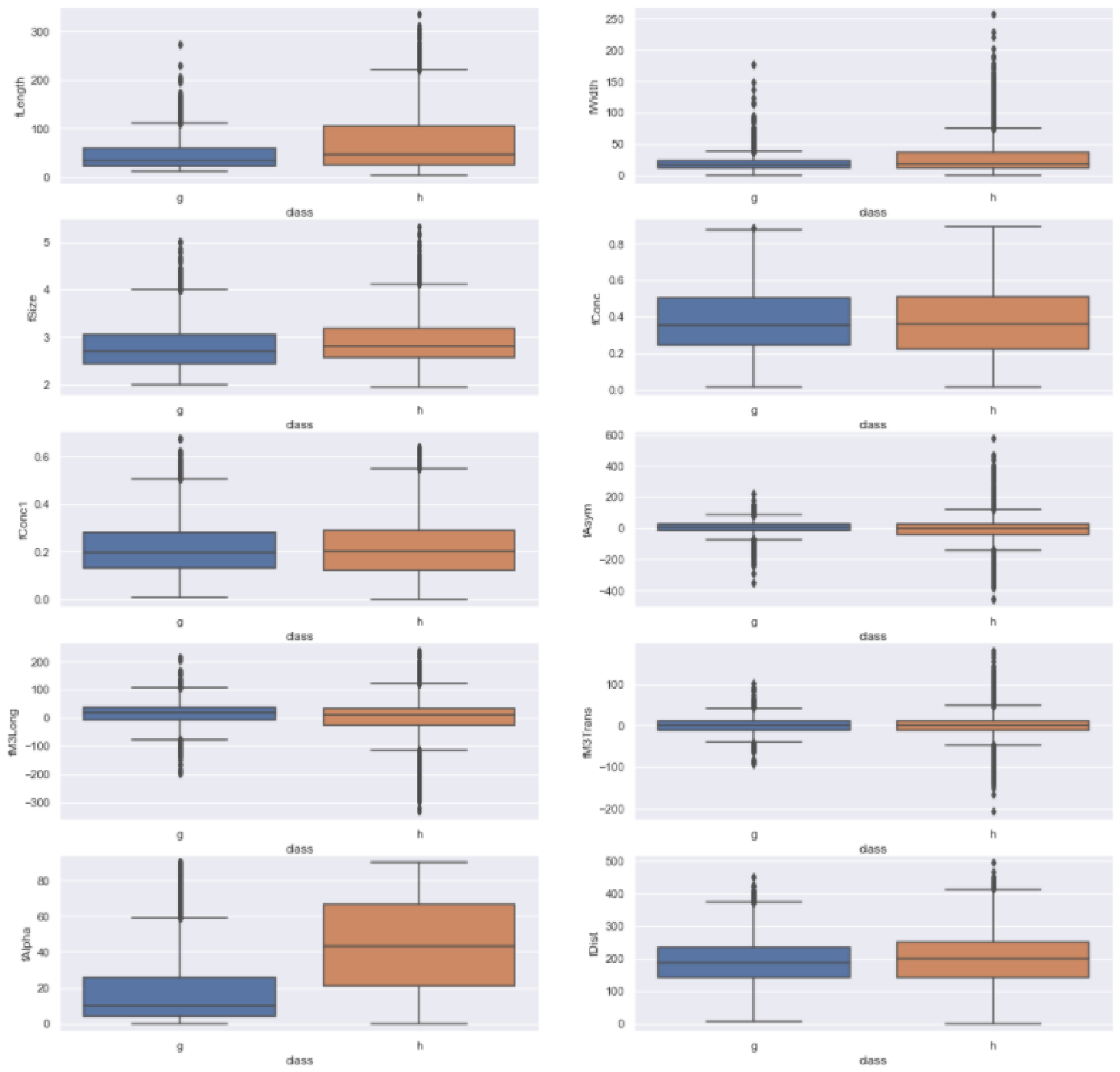


Figure 4. Boxplot representation of the variables

To understand relationship between variables we use the pairplot in Fig. 5 to plot multiple pairwise bivariate distribution on dataset where on the diagonal we have univariate plot.

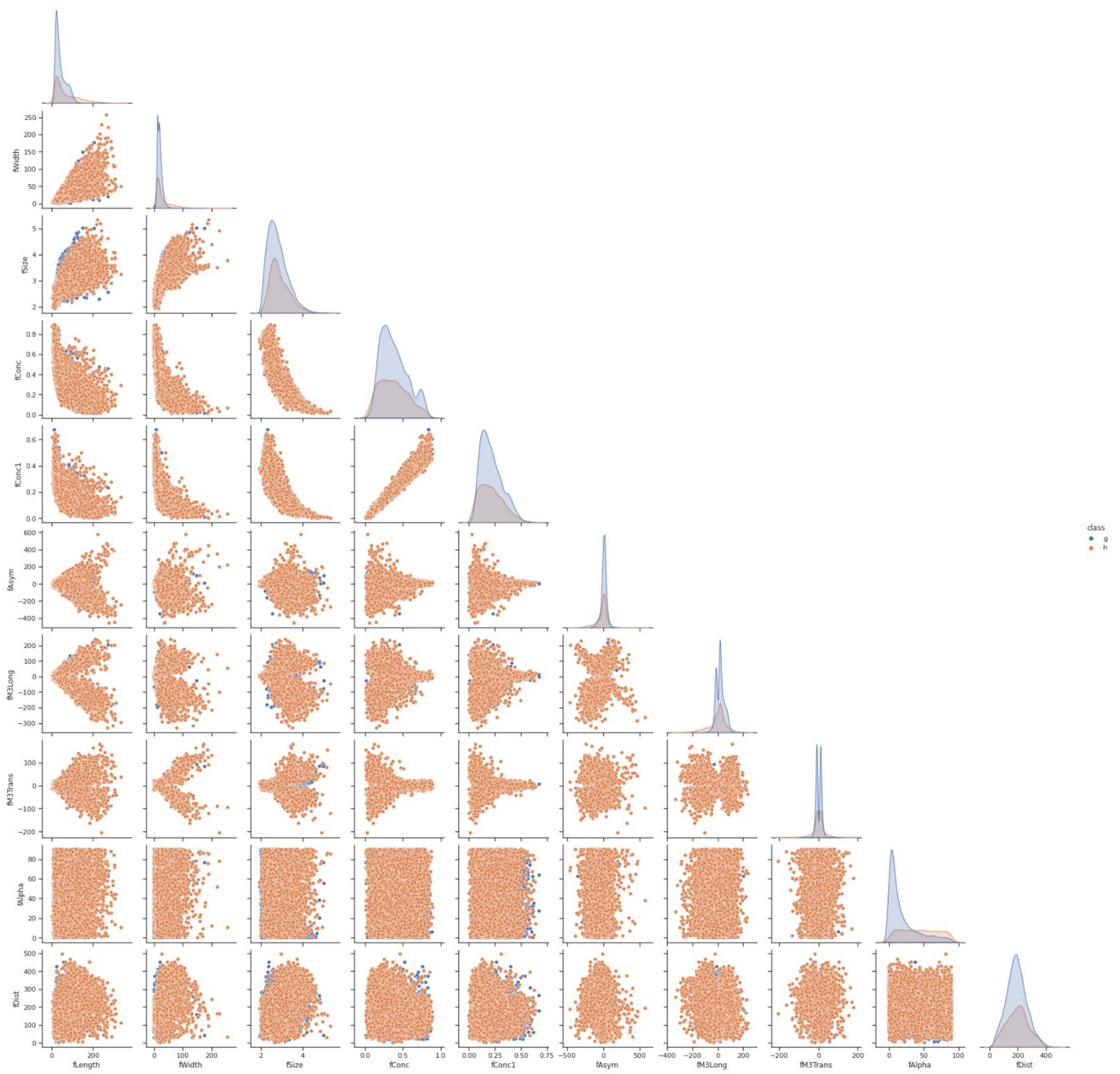


Figure 5. Pairplot of the variables

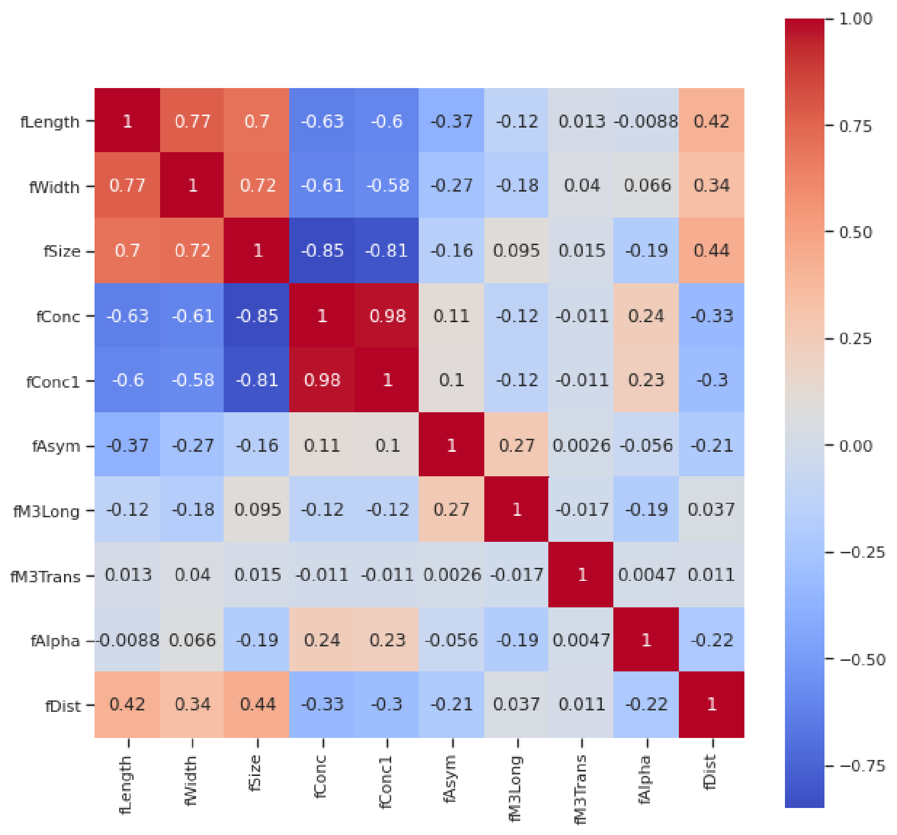
2.3. Feature correlation:

A Correlation is a statistical relationship between two variables, and it commonly refers to the degree to which a pair of variables are linearly related. There are different correlation coefficients that measure the degree of correlation. Among them, the Pearson correlation coefficient is the most common and is sensitive only to a linear relationship between two variables. It returns the values between -1 and 1. Use the below Pearson coefficient correlation calculator to measure the strength of two variables. Pearson correlation coefficient formula:

$$r = \frac{\sum (x_i - \bar{x})(y_i - \bar{y})}{\sqrt{\sum (x_i - \bar{x})^2 \sum (y_i - \bar{y})^2}}$$

r	= correlation coefficient
x_i	= values of the x-variable in a sample
$\bar{x}$	= mean of the values of the x-variable
y_i	= values of the y-variable in a sample
$\bar{y}$	= mean of the values of the y-variable

And we can observe from the heatmap in Fig. 6 the correlation coefficients between each pair of variables and we notice a strong positive correlation between fConc and fConc1, so we decide to select one of them to reduce data redundancy which may have some effects on linear models, also fConc have higher correlations with other variables with respect to fConc1 which is why we decide to drop fConc variable.



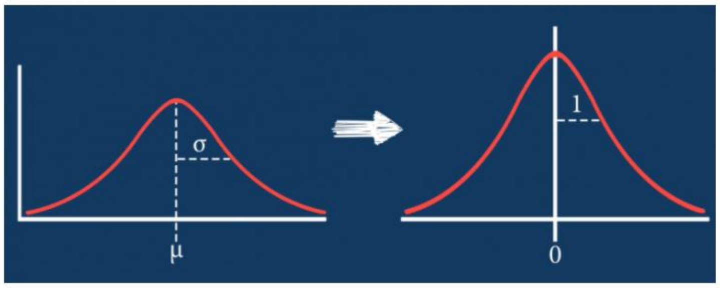
	fLength	fWidth	fSize	fConc	fConc1	fAsym	fM3Long	fM3Trans	fAlpha	fDist
fLength	1	0.77	0.7	-0.63	-0.6	-0.37	-0.12	0.013	-0.0088	0.42
fWidth	0.77	1	0.72	-0.61	-0.58	-0.27	-0.18	0.04	0.066	0.34
fSize	0.7	0.72	1	-0.85	-0.81	-0.16	0.095	0.015	-0.19	0.44
fConc	-0.63	-0.61	-0.85	1	0.98	0.11	-0.12	-0.011	0.24	-0.33
fConc1	-0.6	-0.58	-0.81	0.98	1	0.1	-0.12	-0.011	0.23	-0.3
fAsym	-0.37	-0.27	-0.16	0.11	0.1	1	0.27	0.0026	-0.056	-0.21
fM3Long	-0.12	-0.18	0.095	-0.12	-0.12	0.27	1	-0.017	-0.19	0.037
fM3Trans	0.013	0.04	0.015	-0.011	-0.011	0.0026	-0.017	1	0.0047	0.011
fAlpha	-0.0088	0.066	-0.19	0.24	0.23	-0.056	-0.19	0.0047	1	-0.22
fDist	0.42	0.34	0.44	-0.33	-0.3	-0.21	0.037	0.011	-0.22	1

Figure 6. Pearson Correlation Heatmap

3. Data Preparation:

3.1. Standardization:

It is a scaling technique used when the independent variables are continuous and measured in different scales applying rescaling to ensure the mean and the standard deviation to be 0 and 1, respectively.



$$x_{\text{stand}} = \frac{x - \text{mean}(x)}{\text{standard deviation }(x)}$$

Figure 7. Standardization

3.2. Outlier Detection:

Outliers are extreme values that deviate from other observations on data, they may indicate a variability in a measurement, experimental errors or a novelty. In other words, an outlier is an observation that diverges from an overall pattern on a sample. We already saw in Fig. 4 outliers visually using boxplots, other popular methods exist like z-score which is a metric that indicates how many standard deviations a data point is from a sample's mean.

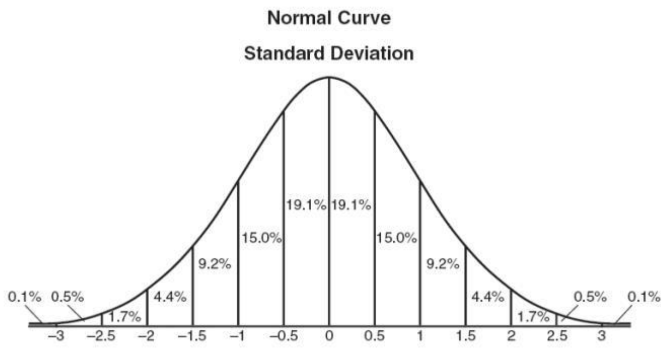


Figure 8. Z-score method

It was applied to our data, and we noticed that 1181 data points lie above the threshold and 1040 from them belong to the hadron class, which is already in the minority class, so we decided to leave them untouched.

3.3. Oversampling:

The dataset at hand is 65:35 which is fairly imbalanced which is why Synthetic Minority Oversampling Technique (SMOTE) was used to rebalance the original training set. Instead of applying a simple replication of the minority class instances, the key idea of SMOTE is to introduce synthetic examples. These new examples are created by interpolation between several minority class instances that lie together. For this reason, the procedure is said to be focused on the "feature space" rather than on the "data space". A simple example of this oversampling process is illustrated in Fig. 9. An x_i of minority class instance is selected as basis to create new synthetic data points. Based on a distance metric, several nearest neighbors of the same class (points x_{i1} to x_{i4}) are chosen from the training set. Finally, a randomized interpolation is carried out in order to obtain new instances r_1 to r_4.

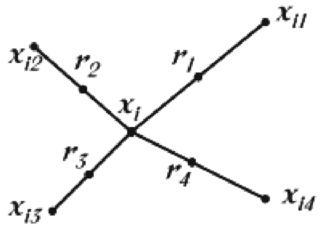


Figure 9. An illustration of how to create synthetic data points using SMOTE

We will see in the tuning and model selection section how SMOTE is fit in the pipeline while doing hyperparameter tuning with cross validation.

3.4. Metrics:

Different performance metrics are used to evaluate classification algorithms. A model may give a very satisfying results when evaluated using a metric like accuracy but poor result on precision for a certain minority which may happen in highly imbalanced data. The metrics used in our case are:

- Accuracy: It is the ration of number of correct predictions to the total number of input samples.

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

- Precision: It is the number of correct positive results divided by the number of positive results predicted by the classifier.

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

- Recall: It is the number of correct positive results divided by the number of positive results predicted by the classifier.

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

- F1_score: It is the Harmonic Mean between precision and recall.

$$\text{F1\_score} = 2 * \frac{Precision * Recall}{Precision + Recall}$$

- Area Under Curve (AUC): It is one of the most widely used metrics for evaluation. It is used for binary classification problem. AUC of a classifier is equal to the probability that the classifier will rank a randomly chosen positive example higher than a randomly chosen negative example. AUC is the area under the curve of plot False Positive Rate vs True Positive Rate at different points in [0, 1], where:

$$\text{TruePositiveRate} = \frac{\text{TP}}{\text{TP+FN}}$$

$$\text{FalsePositiveRate} = \frac{\text{FP}}{\text{FP+TN}}$$

3.5. Tuning and Model Selection:

To tune the hyperparameters for each model and selecting the best model we can observe from Fig. 10 the approach used, where the original dataset is divided into original training and test sets, where the original training is used for tuning hyperparameters using cross-validation with 10 folds where in each iteration after taking a fold out for validation we oversample using SMOTE the n-th training set to give the best configurations of each model and then these models are retrained using the best hyperparameters on the original training set to be tested on the test set and be compared where the model with the best F1_score is selected.

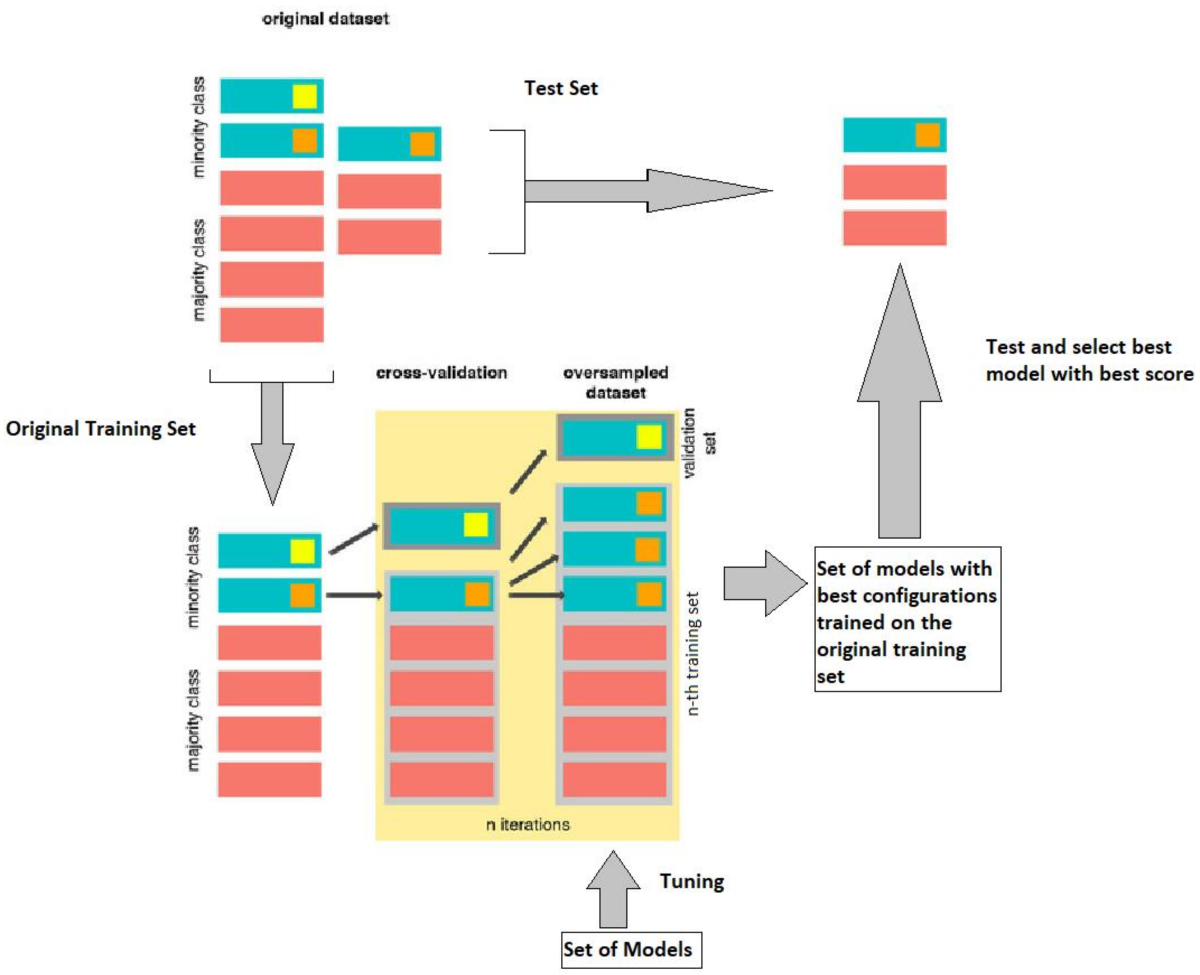


Figure 10. Hyperparameters tuning and model selection.

4. Model Selection:

4.1. Decision Trees:

The decision tree is defined as a supervised learning model that hierarchically maps a data domain onto a response set. It divides a data domain (node) recursively into two subdomains such that the subdomains have a higher information gain than the node that was split, the information gain means the ease of classification in the subdomains created by a split. Finding the best split that gives the maximum information gain (i.e., the ease of classification) is the goal of the optimization algorithm in the decision tree-based supervised learning. The classification tree helps assign a label to a new data set. For example, it can help us decide if a new observation belongs to a class 1 or a class 0. The left diagram in

Fig. 11 shows the propagation of tree split that divides the data domain and creates subdomains for appropriate classification based on the given class labels. It can be seen as the generation of simple multiple thin layers of data domains with split-regions. The first thin layer shows the given data domain and a split condition which is applied to the first feature. The domain is divided into two subdomains, and it is shown in the second thin layer. These subdomains are further divided into four subdomains based on the second feature, and they are shown in the third thin layer. Finally, the class labels are highlighted, and we can see they are classified into different disjointed subdomains. The diagram on the right side shows the given data domain and the final classified data domain. Different impurity measures can be used to decide at each split the best feature and feature value:

- Gini index at a given node t: GINI(t) = 1 - $\sum_j [P\ (j \mid t)]^2$, where $P\ (j \mid t)$ is the relative frequency of class j at node t.
- Entropy at a given node t: Entropy(t) = -$\sum_j P\ (j \mid t)\ log_2 P\ (j \mid t)$, where $P\ (j \mid t)$ is the relative frequency of class j at node t.

For each feature, the best split location is selected based on the split that gives the highest information gain, and the feature values at that location will be recorded.

The data domain is then divided into two subdomains (SD1 and SD2) at the split locations. And this will be repeated for the subdomains SD1 and SD2 to get the best feature, best split location, and the best split feature value for the new nodes.

The decision tree will be built following these processes until the subdomain that does not need a split because of the subdomain that has a significantly large number of observations from a single class.

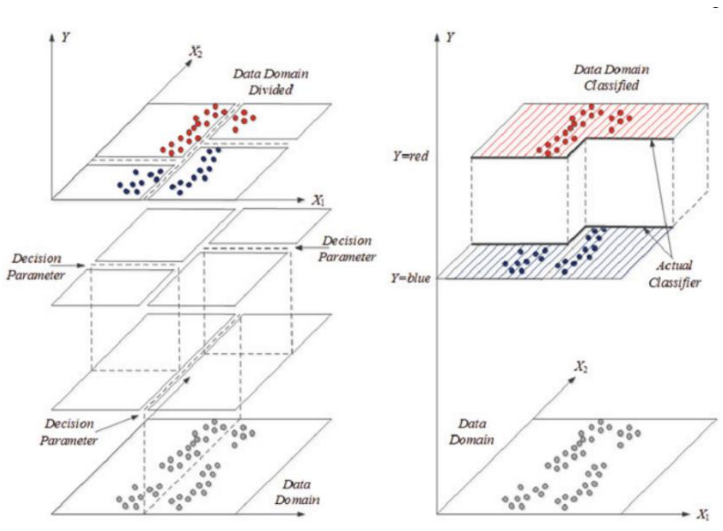


Figure 11. Illustration of classification tree using two classes with domain division properties

Tuning Phase:

Hyperparameters	Values	Best parameter
criterion	gini, entropy	gini
splitter	best, random	best
max_features	auto, sqrt	auto
max_depth	None, 5, 10, 20	10

Model Evaluation:

AUC= 0.812

```
              precision    recall  f1-score   support

           0       0.87      0.87      0.87      2466
           1       0.76      0.75      0.76      1315

    accuracy                           0.83      3781
   macro avg       0.81      0.81      0.81      3781
weighted avg       0.83      0.83      0.83      3781
```

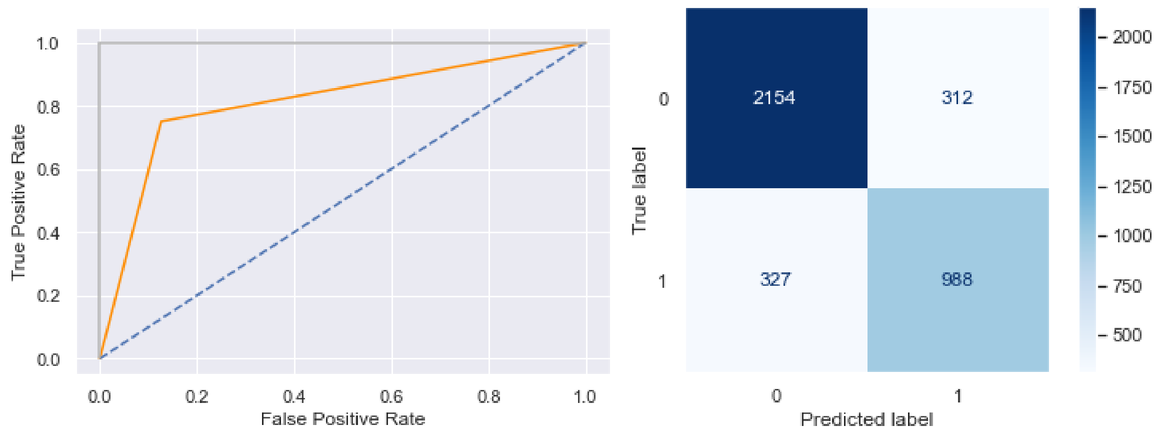


Figure 12. ROC- curve (left) and confusion matrix (right) for Decision Tree Classifier

4.2. Random Forest:

The random forest is a supervised learning technique and, as the name suggests, it forms forest-like structures with decision trees that are generated using the Bootstrapping. The advantage of the random forest is that it provides multiple trained decision tree classifiers for the testing phase. This property of the random forest supervised learning technique makes the random forest a preferred technique over regular decision tree learning. Bootstrapping is a simple randomization technique, but its effect on the supervised learning algorithms, especially on the random forest, is magnificent. It helps generate several subsets from a set of data by randomly selecting the same number of observations as the original data set, but with the replacement. This allows some observations from the original data to be repeated in a subset of the data set. As we see from Fig. 13, we produce multiple bootstrapped samples from the original dataset and apply decision tree classifier but instead of using all features we use a different subset at each level usually if p is number of predictors we use a subset of $\sqrt{p}$ features and choose the best one according to a certain impurity measure similar to decision tree and finally for classifying a certain point we aggregate and apply majority voting to predict the final class.

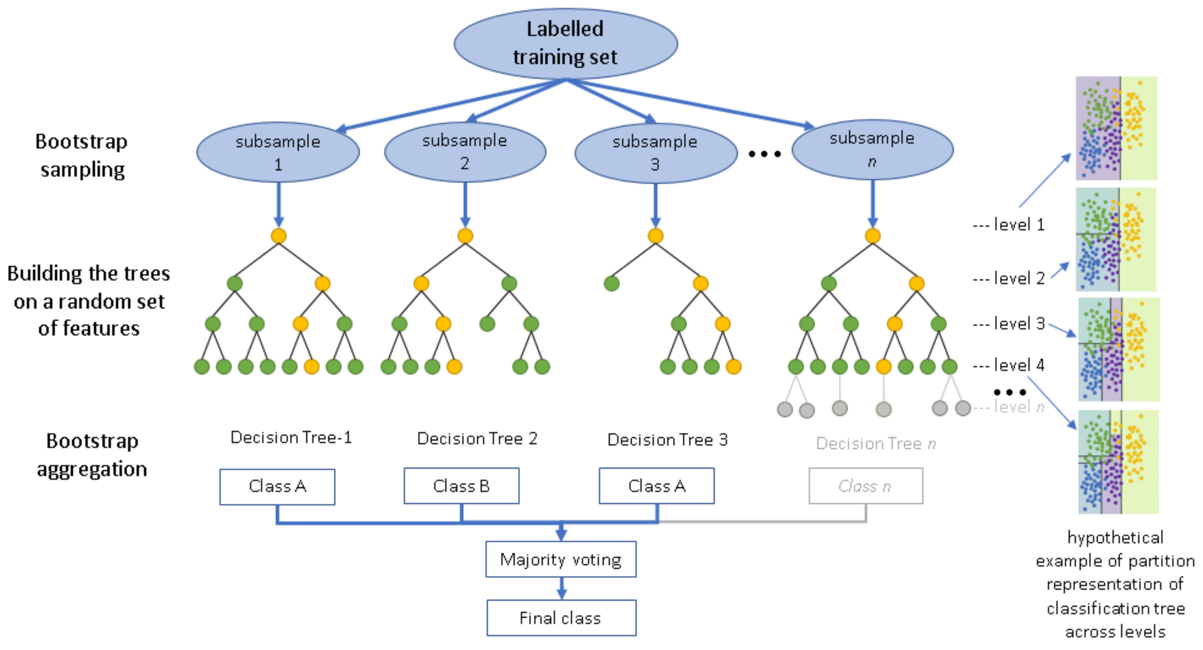


Figure 13. Illustration of Random Forest Classifier

Tuning Phase:

Hyperparameters	**Values**	**Best parameter**
n_estimators	100, 150, 200	150
criterion	gini, entropy	entropy
max_features	auto, sqrt	sqrt
max_depth	None, 5, 10, 20	None

Model Evaluation:

AUC= 0.853

```
              precision    recall  f1-score   support

           0       0.89      0.92      0.90      2466
           1       0.84      0.79      0.81      1315

    accuracy                           0.87      3781
   macro avg       0.87      0.85      0.86      3781
weighted avg       0.87      0.87      0.87      3781
```

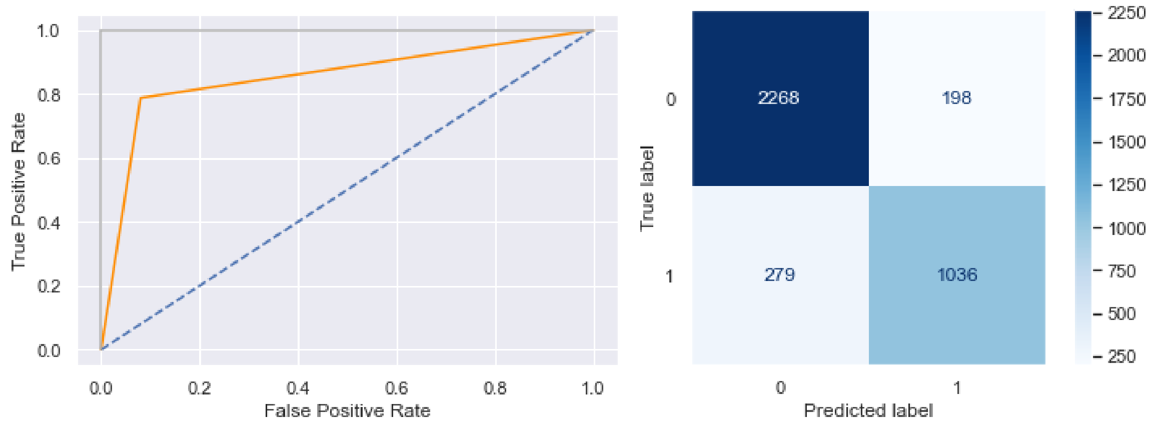


Figure 14. ROC- curve (left) and confusion matrix (right) for Random Forest Classifier

4.3. Logistic Regression:

Logistic regression is a hypothesis parameterized by a d-dimensional vector β and a scalar β_0, but instead of making predictions in {0, 1}, they generate real-valued outputs in the interval (0, 1). A logistic regression has the form:

$$h(X; \beta, \beta_0) = \sigma(\beta^T X + \beta_0).$$

The logistic function, also known as the sigmoid function, is defined as

$\sigma(z) = \frac{1}{1+e^{-z}}$, and plotted below, as a function of its input z.

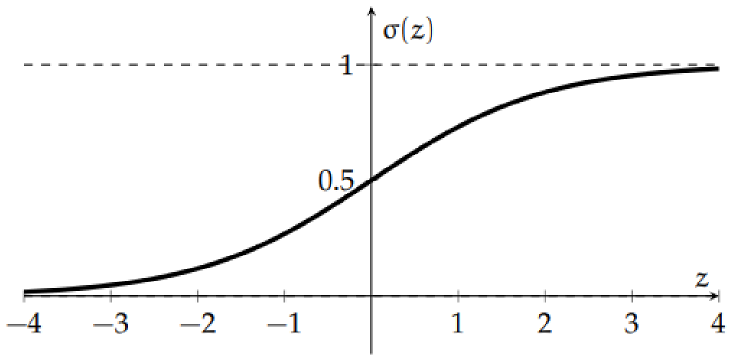


Figure 15. Sigmoid function

Its output can be interpreted as a probability, because for any value of z the output is in (0, 1):

$$P(Y = 1|X) = \frac{1}{1+e^{\beta_0+\beta X}} = h(X; \beta, \beta_0)$$

$$P(Y = 0|X) = \frac{e^{\beta_0+\beta X}}{1+e^{\beta_0+\beta X}} = 1 - h(X; \beta, \beta_0)$$

, but we have training data with target values in {0, 1}. We would like to pick the parameters of our classifier to maximize the probability assigned by the classifier to the correct target values, as specified in the training set. Letting guess $g^{(i)} = \sigma(\beta^T x^{(i)} + \beta_0)$, that probability is:

$$\prod_{i=1}^{n} \begin{cases} g^{(i)} & if\ y^{(i)} = 1 \\ 1 - g^{(i)} & otherwise \end{cases}$$

under the assumption that our predictions are independent. This can be cleverly rewritten when y (i) ∈ {0, 1}, as:

$$\prod_{i=1}^{n} {g^{(i)}}^{y^{(i)}} (1 - g^{(i)})^{1-y^{(i)}}$$

Now, because products are kind of hard to deal with, and because the log function is monotonic, the β, β_0 that maximize the log of this quantity will be the same as the β, β_0 that maximize the original, so we can try to maximize:

$$\sum_{i=1}^{n} \left(y^{(i)} log(g^{(i)}) + (1 - y^{(i)}) \log(1 - g^{(i)})\right)$$

We can turn the maximization problem above into a minimization problem by taking the negative of the above expression, and write in terms of minimizing a loss:

$$\sum_{i=1}^{n} L_{nll}(g^{(i)}, y^{(i)})$$

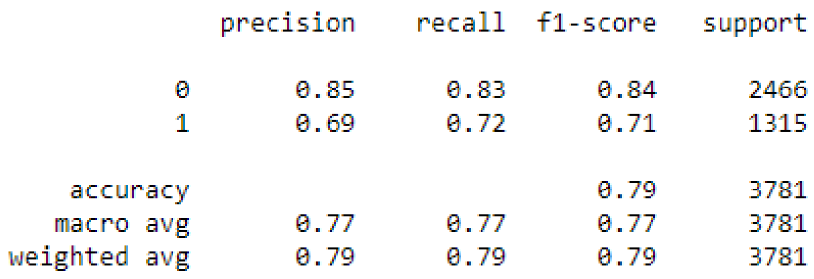
where L_{nll} is the negative log-likelihood loss function: L_{nll} (guess, actual) = − (actual · log(guess) + (1 − actual) · log (1 − guess)). This loss function is also sometimes referred to as the log loss or cross entropy.

Tuning Phase:

Hyperparameters	Values	Best parameter
C	0.001,0.01,0.1,1,10,100,1000	0.01
penalty	l1, l2	l1
solver	lbfgs, liblinear	liblinear

Model Evaluation:

AUC= 0.774



```
              precision    recall  f1-score   support

           0       0.85      0.83      0.84      2466
           1       0.69      0.72      0.71      1315

    accuracy                           0.79      3781
   macro avg       0.77      0.77      0.77      3781
weighted avg       0.79      0.79      0.79      3781
```

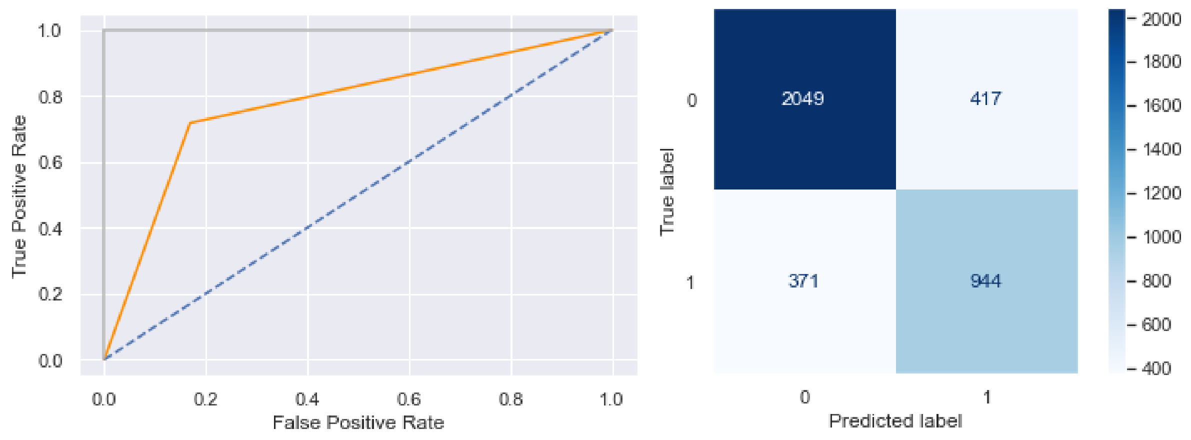
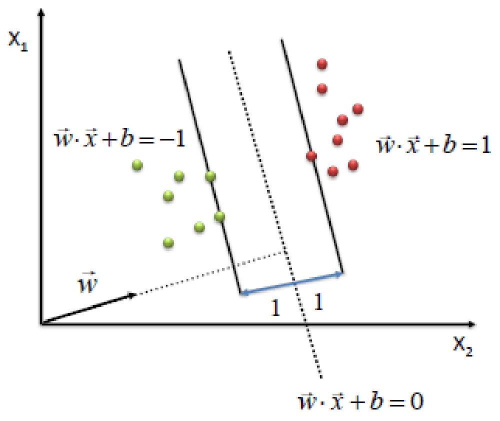


Figure 16. ROC- curve (left) and confusion matrix (right) for Logistic Regression

4.4. Linear SVM:

The Support Vector Machine (SVM) is a linear classifier that can be viewed as an extension of the Perceptron. The Perceptron guaranteed that you find a hyperplane if it exists. The SVM finds the maximum margin separating hyperplane, where the margin is defined as the distance from the hyperplane to the closest point across both classes.



X₁
$\vec{w}\cdot\vec{x}+b=-1$
$\vec{w}\cdot\vec{x}+b=1$
$\vec{w}$
1
1
X₂
$\vec{w}\cdot\vec{x}+b=0$

Figure 17. Hard margin case of SVC

Hard Margin:

So, our objective function is:

$$\underset{w,b}{\text{maximize}} \frac{1}{\|w\|} \text{ subject to } y_i \left[\langle x_i, w\rangle + b\right] \geq 1$$

Which is equivalent to:

$$\underset{w,b}{\text{minimize}} \frac{1}{2}\|w\|^2 \text{ subject to } y_i \left[\langle x_i, w\rangle + b\right] \geq 1$$

Which is solved by taking the Lagrangian dual and finding values where the derivative vanishes and substituting in the objective function:

$$L(w,b,\alpha) = \frac{1}{2}\|w\|^2 - \sum \alpha_i \left[y_i\left[\langle x_i, w\rangle + b\right] - 1\right]$$

$$w = \sum_i y_i \alpha_i x_i$$

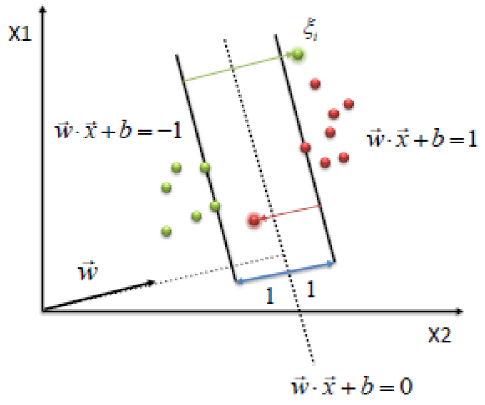
$$\underset{\alpha}{\text{maximize}} - \frac{1}{2}\sum_{i,j} \alpha_i \alpha_j y_i y_j \langle x_i, x_j\rangle + \sum_i \alpha_i$$

$$s.t. \ \sum_i \alpha_i y_i = 0$$

$$\alpha_i \geq 0 \quad \forall i = 1, \ldots, N$$

The beauty of SVM is that if the data is linearly separable, there is a unique global minimum value. An ideal SVM analysis should produce a hyperplane that completely separates the vectors (cases) into two non-overlapping classes. However, perfect separation may not be possible, or it may result in a model with so many cases that the model does not classify correctly. In this situation SVM finds the hyperplane that maximizes the margin and minimizes the misclassifications.

Soft Margin:



X1
ξ_i
$\vec{w}\cdot\vec{x}+b=-1$
$\vec{w}\cdot\vec{x}+b=1$
$\vec{w}$
1
1
X2
$\vec{w}\cdot\vec{x}+b=0$

Figure 18. Soft margin case of SVC

The objective functions:

$$\underset{w,b}{\text{minimize}} \ \frac{1}{2}\|w\|^2 + C\sum_i \xi_i$$
$$\text{subject to } y_i\left[\langle w, x_i\rangle + b\right] \geq 1 - \xi_i \text{ and } \xi_i \geq 0$$

Using Lagrangian dual and finding vanishing derivative values and substituting:

$$L(w,b,\alpha) = \frac{1}{2}\|w\|^2 + C\sum_i \xi_i - \sum_i \alpha_i\left[y_i\left[\langle x_i, w\rangle + b\right] + \xi_i - 1\right] - \sum_i \eta_i \xi_i$$

$$w = \sum_i y_i \alpha_i x_i$$

$$\underset{\alpha}{\text{maximize}} - \frac{1}{2}\sum_{i,j} \alpha_i \alpha_j y_i y_j \langle x_i, x_j \rangle + \sum_i \alpha_i$$
$$\text{subject to } \sum_i \alpha_i y_i = 0 \text{ and } \alpha_i \in [0, C]$$

Tuning Phase:

Hyperparameters	Values	Best parameter
C	0.1,1,10	0.1
penalty	l1, l2	l2
loss	hinge, squared hinge	hinge

Model Evaluation:

AUC= 0.775

```
              precision    recall  f1-score   support

           0       0.85      0.83      0.84      2466
           1       0.70      0.72      0.71      1315

    accuracy                           0.79      3781
   macro avg       0.77      0.78      0.77      3781
weighted avg       0.80      0.79      0.79      3781
```

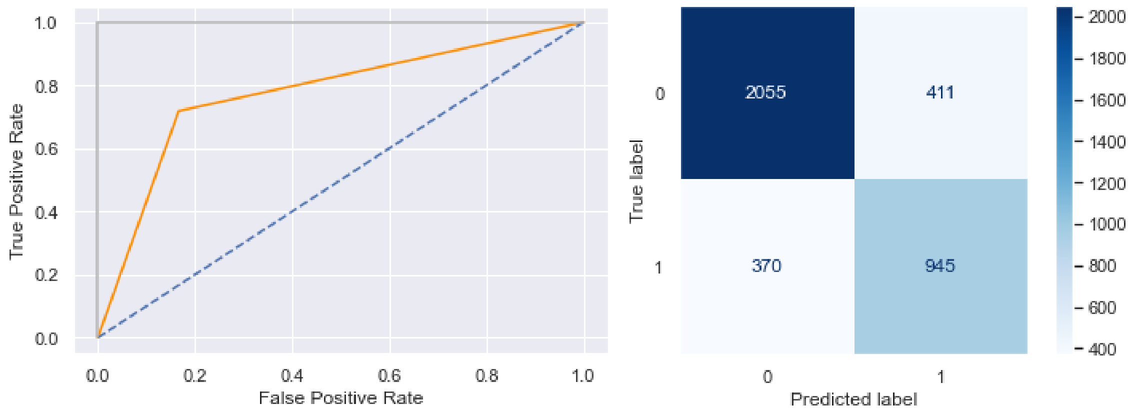
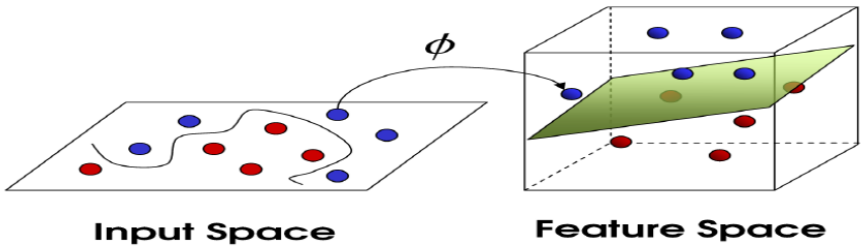


Figure 19. ROC- curve (left) and confusion matrix (right) for Linear SVC

4.5. Non-Linear SVM:

There are situations where the data cannot be separated linearly, this is where we use the kernel trick. The idea is to gain linear separation by mapping the data to a higher dimensional space.



$$\Phi : \mathcal{X} \mapsto \hat{\mathcal{X}} = \Phi(\mathbf{x})$$

Figure 20. Kernel trick

Where Φ is a function mapping data from input space to a higher dimensional feature space.

And the kernel:

$$K(\mathbf{x}_i, \mathbf{x}_j) = \langle \Phi(\mathbf{x}_i), \Phi(\mathbf{x}_j) \rangle$$

Which will be used rather than $\langle \mathbf{x}_i, \mathbf{x}_j \rangle$ in the objective function we reached in the Linear SVM formulation previously getting:

$$\max_{\alpha} -\frac{1}{2} \sum_i \alpha_i \alpha_j y_i y_j \, k(x_i, x_j) + \sum_i \alpha_i$$

$$s.t. \; \sum_i \alpha_i y_i = 0$$

$$\alpha_i \in [0, C] \quad \forall i = 1, \ldots, N$$

Example of kernel functions:

RBF:

$$K(\mathbf{x}_i, \mathbf{x}_j) = \exp\left(-\frac{\|\mathbf{x}_i - \mathbf{x}_j\|_2^2}{2\sigma^2}\right)$$

Polynomial:

$$K(\mathbf{x}_i, \mathbf{x}_j) = \langle \mathbf{x}_i, \mathbf{x}_j \rangle^2$$

Sigmoid:

$$K(\mathbf{x}_i, \mathbf{x}_j) = \tanh\left(\alpha \mathbf{x}_i^\intercal \mathbf{x}_j + c\right)$$

Tuning Phase:

Hyperparameters	Values	Best parameter
kernel	rbf, sigmoid, poly	rbf
C	0.1, 1,10	entropy
degree (for poly kernel only)	2, 3, 4	None

Model Evaluation:

AUC= 0.846

```
              precision    recall  f1-score   support

           0       0.89      0.91      0.90      2466
           1       0.83      0.78      0.80      1315

    accuracy                           0.87      3781
   macro avg       0.86      0.85      0.85      3781
weighted avg       0.87      0.87      0.87      3781
```

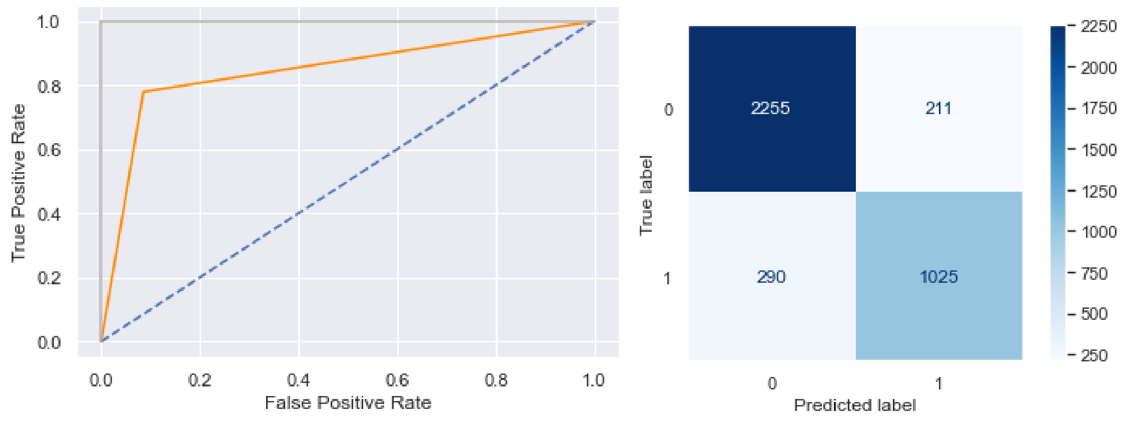


Figure 21. ROC- curve (left) and confusion matrix (right) for SVC

4.6. K-Nearest Neighbors:

K-NN is one of the simplest Machine Learning algorithms based on Supervised Learning technique. K-NN algorithm assumes the similarity between the new case/data and available cases and put the new case into the category that is most similar to the available categories. K-NN is a non-parametric meaning there is no assumption for underlying data distribution. It does not need any training data points for model generation. All training data used in the testing phase. This makes training faster and testing phase slower and costlier. Costly testing phase means time and memory. In the worst case, K-NN needs more time to scan all data points and scanning all data points will require more memory for storing training data. In K-NN, K is the number of nearest neighbors. The number of neighbors is the core deciding factor. K is generally an odd number if the number of classes is 2.

The K-NN working can be explained on the basis of the below algorithm:

1. Select the number K of the neighbors.
2. Calculate the distance (Euclidean, Manhattan, or Minkowski etc.) of K number of neighbors.
3. Take the K nearest neighbors as per the calculated distance.
4. Among these k neighbors, count the number of the data points in each category.
5. Assign the new data points to that category for which the number of the neighbor is maximum.

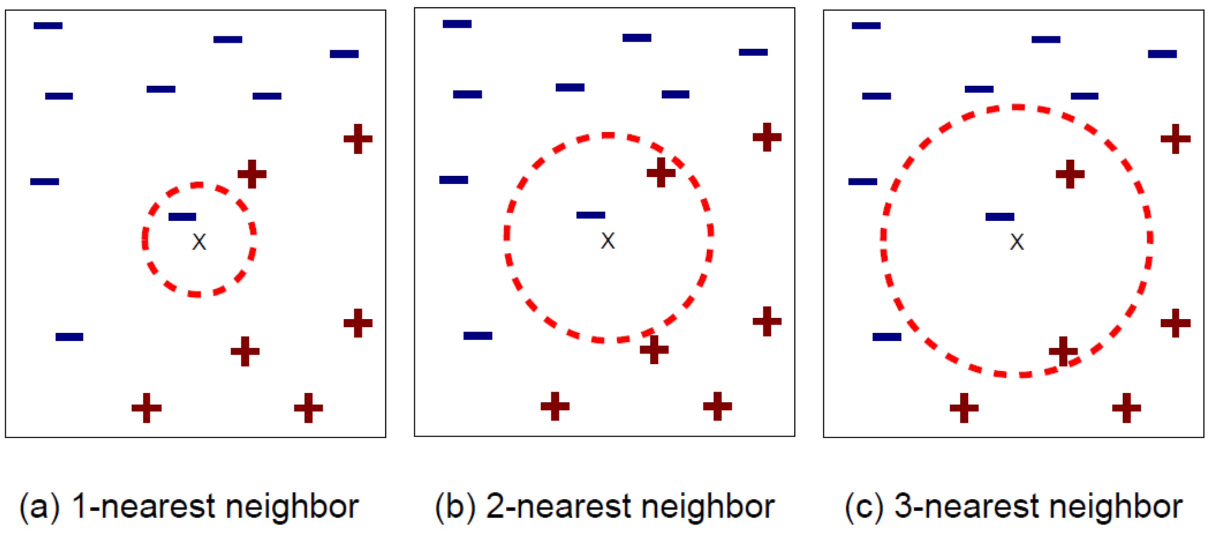


Figure 22. Illustration of KNN

Tuning Phase:

Hyperparameters	Values	Best parameter
n_neighbors	3, 5, 7	7
weights	uniform, distance	distance
p	1, 2	1

Model Evaluation:

AUC = 0.818

```
              precision    recall  f1-score   support

           0       0.87      0.88      0.87      2466
           1       0.77      0.76      0.76      1315

    accuracy                           0.84      3781
   macro avg       0.82      0.82      0.82      3781
weighted avg       0.84      0.84      0.84      3781
```

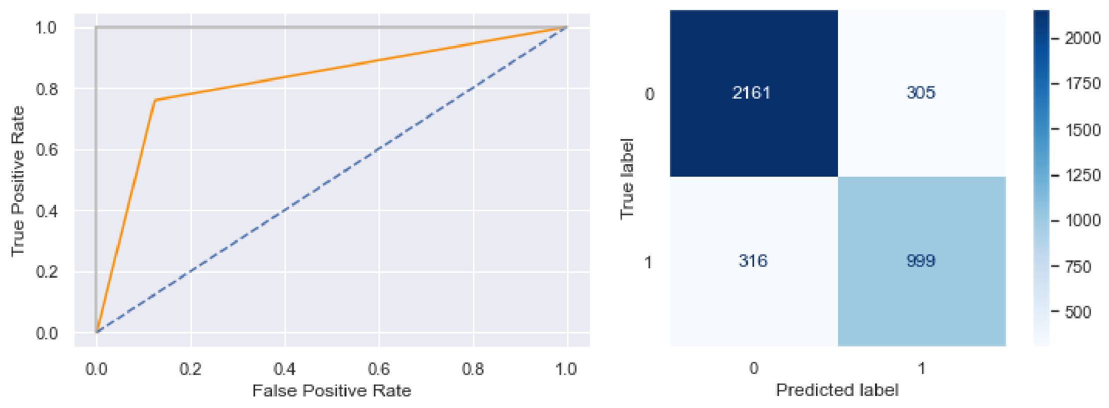


Figure 23. ROC- curve (left) and confusion matrix (right) for KNN

5. Conclusion:

We conclude from the Fig. 24 below that the best model is Random Forest and SVC coming right after it considering the F1_score and AUC_score.

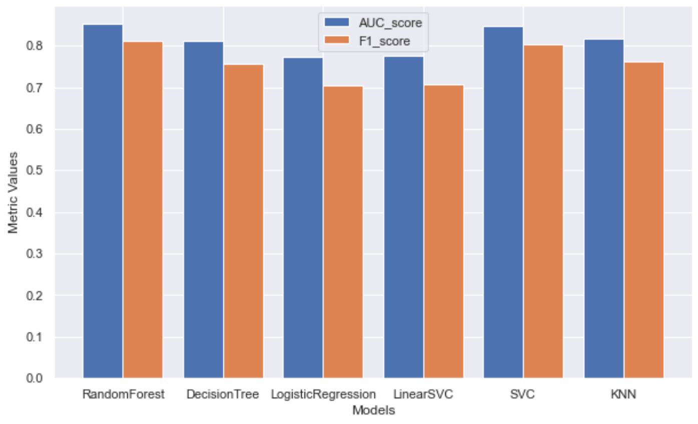


Figure 24. Comparison of F1_score and AUC_score for all models

We can observe the feature importance with respect to the best model which is Random Forest in Fig. 25.

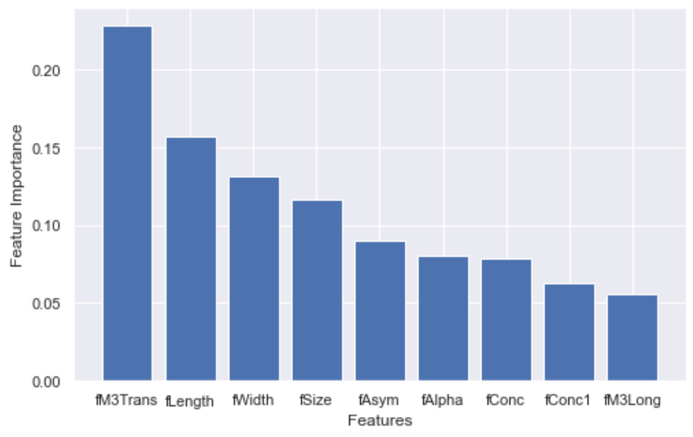


Figure 25. Feature importance by Random Forest Classifier

6. Reference:

- Shai Shalev-Shwartz, Shai Ben-David, 2014. Understanding machine learning theory and algorithms. Cambridge University Press.
- James, Witten, Hastie, Tibshirani, 2013. An introduction to Statistical Learning. Springer.
- Dirk P. Kroese, Zdravko I. Botev, Thomas Taimre, Radislav Vaisman, 2020. Data Science and Machine Learning.
- http://archive.ics.uci.edu/ml/datasets/MAGIC+Gamma+Telescope