Classification of High Energy Gamma Particles from MAGIC Telescopes

Name: Gabriel Linares

1. Introduction

1.1 Problem Statement

Gamma-ray astronomy is a subset of Astronomy in ground-based, and space telescopes which aim to detect Gamma-rays emitted by cosmic bodies. Gamma rays are high energy particles that can originate from many radioactive processes in the universe such as supernova explosions, black holes, and neutron stars. Thus, the separation of Gamma and Hadron events is critical for cosmological research. When a high energy gamma ray interacts with Earth's atmosphere it generates an air shower of particles that emit a feint blue light known as Cherenkov light (*H.E.S.S*). When the telescope detects this light, the gathered data can then be used to extract the showers shape parameters. Consequently, after some preprocessing these parameters can be used to classify a Gamma ray from cosmic rays (*Bock, R*).

1.2 Motivation and Challenges

One of the most prominent challenges in this topic is that these events are severely imbalanced. As Hadron events happen significantly more frequently than Gamma events, it makes the misclassification of Hadron events much more costly than the misclassification of Gamma events.

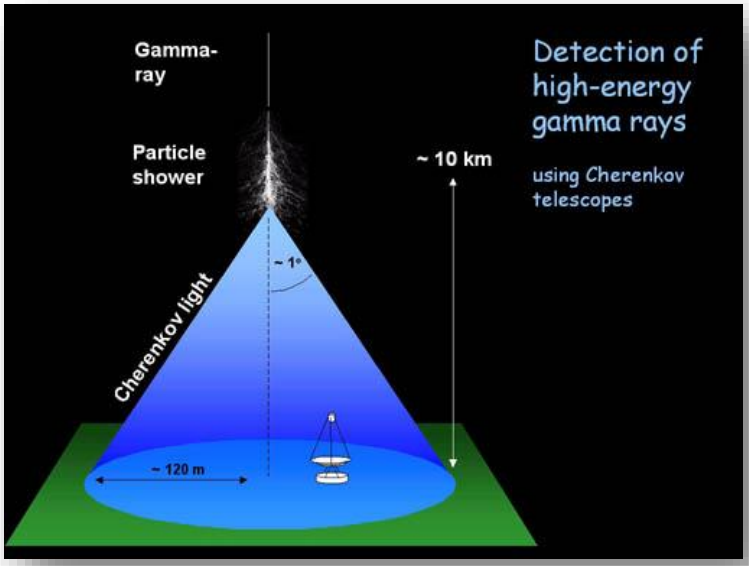
1.3 Summary

I have always been interested in cosmology and scientific computing; this has been one of the main motivating factors for the exploration of this problem. In this project, I aim to gain some insight into how the class imbalance in this dataset plays a role in classification performance.

2. Data

2.1 Introduction

For this project, I will be using a dataset that has been generated by a Monte Carlo program which is designed to simulate imaging as would have been collected from a **Major Atmospheric Gamma Imaging Cherenkov Telescope (MAGIC)**. This dataset simulates a Cherenkov Air Shower and provides over 19,000 samples and 10 features from which I will train several Classification Machine Learning Models. Each sample in this dataset provides the characteristics of the shower image which takes an elliptical shape. We can then use these parameters to statistically discriminate the classes (gamma, hadron). The following diagram shows a representation of a Cherenkov Air Shower.



2.2 *Visualization and Analysis*

To visualize this dataset, I chose to use relatively simple tools provided by Seaborn and Pandas. Firstly, we can visualize the distribution of classes among each feature by using a pair-plot colored accordingly to each class (orange: hadron, blue: gamma). By analyzing the distributions on the main diagonal of this matrix we hypothesize that this data will not be easily separable by using linear classifiers. Thus, we may have to use more complex algorithms or feature engineering techniques to solve this problem. Although, we can see several linear and inverse relationships between features (not classes), there are several others that are randomly distributed. We can see a very high correlation between features (fConc, fConc1), this may cause some issues down the line. We may be able to drop one

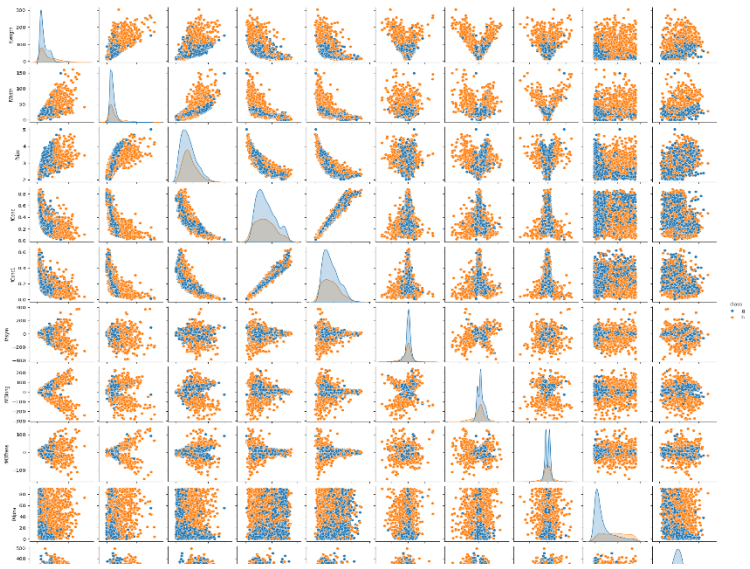


Figure 1: Seaborn pair-plot of feature distribution among classes

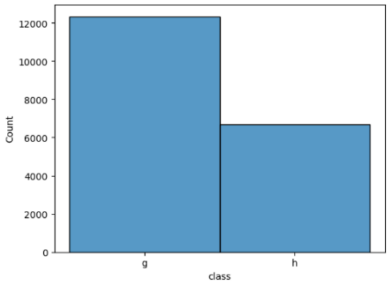


Figure 2: Class counts

of these features to simplify our models without sacrificing performance. Lastly, by visualizing the class counts in Figure 2 we can see that this dataset is severely imbalanced with about 70% of our data belonging to the gamma class.

Furthermore, by displaying a correlation matrix we can confirm that there is a highly positive correlation between fConc and fConc1. We can also see several other highly negative correlations such as fConc/fConc1 and fSize. These correlations only make our models more complex without adding much more performance value.

	fLength	fWidth	fSize	fConc	fConc1	fAsym	fM3long	fM3Trans	fAlpha	fDist
fLength	1.000000	0.770512	0.702454	-0.630999	-0.598145	-0.368556	-0.119747	0.013389	-0.008777	0.418466
fWidth	0.770512	1.000000	0.717517	-0.609779	-0.581141	-0.266961	-0.176234	0.039744	0.066061	0.336816
fSize	0.702454	0.717517	1.000000	-0.850850	-0.808835	-0.159863	0.095157	0.015455	-0.186675	0.437041
fConc	-0.630999	-0.609779	-0.850850	1.000000	0.976412	0.112272	-0.121899	-0.011294	0.235272	-0.328332
fConc1	-0.598145	-0.581141	-0.808835	0.976412	1.000000	0.100159	-0.118769	-0.010966	0.229799	-0.304625
fAsym	-0.368556	-0.266961	-0.159863	0.112272	0.100159	1.000000	0.274045	0.002553	-0.055689	-0.206730
fM3long	-0.119747	-0.176234	0.095157	-0.121899	-0.118769	0.274045	1.000000	-0.017197	-0.186275	0.037025
fM3Trans	0.013389	0.039744	0.015455	-0.011294	-0.010966	0.002553	-0.017197	1.000000	0.004659	0.011427
fAlpha	-0.008777	0.066061	-0.186675	0.235272	0.229799	-0.055689	-0.186275	0.004659	1.000000	-0.220556
fDist	0.418466	0.336816	0.437041	-0.328332	-0.304625	-0.206730	0.037025	0.011427	-0.220556	1.000000

Figure 3: Correlation Matrix of MAGIC features

Lastly, by looking at some basic descriptive statistics we can gather some more information about the central tendencies and dispersion of this dataset. We can see that these features take on a different range of values which may be problematic for our models. Thus, we will have to normalize this dataset to provide a more stable range of values for the models.

	fLength	fWidth	fSize	fConc	fConc1	fAsym	fM3long	fM3Trans	fAlpha	fDist
count	19020.000000	19020.000000	19020.000000	19020.000000	19020.000000	19020.000000	19020.000000	19020.000000	19020.000000	19020.000000
mean	53.250154	22.180966	2.825017	0.380327	0.214657	-4.331745	10.545545	0.249726	27.645707	193.818026
std	42.364855	18.346056	0.472599	0.182813	0.110511	59.206062	51.000118	20.827439	26.103621	74.731787
min	4.283500	0.000000	1.941300	0.013100	0.000300	-457.916100	-331.780000	-205.894700	0.000000	1.282600
25%	24.336000	11.863800	2.477100	0.235800	0.128475	-20.586550	-12.842775	-10.849375	5.547925	142.492250
50%	37.147700	17.139900	2.739600	0.354150	0.196500	4.013050	15.314100	0.666200	17.679500	191.851450
75%	70.122175	24.739475	3.101600	0.503700	0.285225	24.063700	35.837800	10.946425	45.883550	240.563825
max	334.177000	256.382000	5.323300	0.893000	0.675200	575.240700	238.321000	179.851000	90.000000	495.561000

Figure 4: Descriptive Statistics of the MAGIC dataset

2.3 Preprocessing

The preprocessing of this dataset is relatively simple as it does not have any inconsistencies or missing values. However, since several of the features have different scales, we will have to normalize this data. I experimented with the StandardScaler and MinMaxScaler and ultimately chose the MinMaxScaler because it provided slightly better performance on this dataset.

3. Method

3.1 Stochastic Gradient Descent with Perceptron Loss

Gradient Descent aims to find the minimum by computing the derivative of the loss function. Since the derivative signifies the direction of steepest ascent. Thus, to minimize the loss this algorithm moves in the direction of the negative of the derivative. Stochastic Gradient Descent employs random sampling to calculate the direction of the steepest descent to minimize the loss. Additionally, the Perceptron loss function only makes necessary weight updates when an instance is misclassified which then moves the decision boundary to minimize the error (closer to the misclassification).

3.2 Logistic Regression

The Logistic Regression model is a probabilistic binary classification algorithm. This model uses the Sigmoid activation function which enables the model to assign a probability for each sample corresponding to a certain class label (0,1). Furthermore, the logistic regression finds the maximum likelihood given by some feature vector and their respective weights.

3.3 Support Vector Classifier

The Support Vector Machine is an algorithm that can be used for either classification or regression problems. The SVM attempts to find a hyperplane that can separate the data into the 2 classes for binary classification. This model can be used with several different Kernels including Radial Basis Function (RBF) which can map features into infinite dimensions, the Sigmoid function and several others. For this experiment, I chose to use a simple polynomial Kernel which maps the features to a specified dimension.

4. Results

4.1 Experimental Setup

For the classification of Gamma/Hadron particles, I have used a dataset which contains over 19,000 samples from a MAGIC telescope. This is a public dataset which has been preprocessed; thus, I had to do very little work to apply it to a machine learning model. For this experiment, I opted to map the categorical values of (g, h) to (0, 1) respectively. Additionally, I split the data by assigning 80% of the total data for training and 20% for testing. Lastly, a 20% sample of the training data was pulled for validation.

I evaluated the performance of several traditional machine learning models which included: Stochastic Gradient Descent with Perceptron loss, Logistic Regression (with Polynomial features) and Support Vector Machine (with Polynomial Kernel). I opted to preset the class weight parameters to achieve a more balanced dataset, since the Hadron class contained only about 6600 samples and the Gamma class contained over 12000 samples.

For this experiment, I will be considering the Receiver Operating Characteristic (ROC) curve for each of these models due to the natural imbalance in which real Gamma and Hadron events occur. In real data, the number of Hadron class represents most of the events according to the dataset documentation.

4.2 Testing Results

Stochastic Gradient Descent with Perceptron loss

I decided to train a simplistic model for comparison with more advanced algorithms. Scikit-learn provides a general-purpose algorithm which uses Stochastic Gradient Descent training and multiple loss functions. I decided to use a "Perceptron" loss function to begin this experiment which is a linear loss function. For this model I used an adaptive learning rate parameter and no regularization.

```
Accuracy: 0.8209779179810726
Classification Report:
              precision    recall  f1-score   support

           0       0.72      0.80      0.76      1344
           1       0.88      0.83      0.86      2460

    accuracy                           0.82      3804
   macro avg       0.80      0.82      0.81      3804
weighted avg       0.83      0.82      0.82      3804
```

Figure 5: SGD Perceptron Accuracy Report

This model's performance was subpar, initially only achieving about 77% accuracy with the original feature set. The performance was only slightly improved to approximately 82% by balancing the dataset to have equal weights for each class along with adding polynomial features of the 4^{th} degree. The highest accuracy for this model was achieved during training and did not increase on the validation or testing data sets.

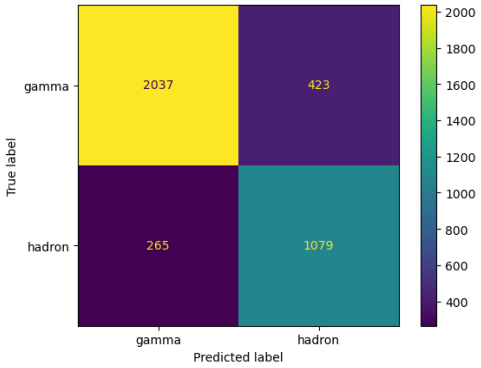


Figure 6: SGD Perceptron Confusion Matrix

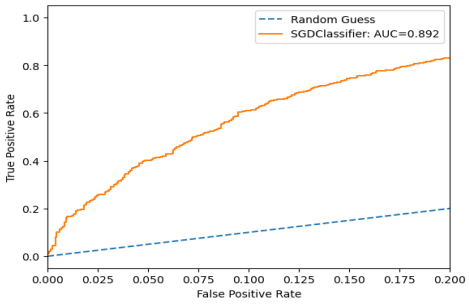


Figure 7: SGD Perceptron ROC curve

As can be seen from the confusion matrix, this model misclassifies 265 Hadrons and 423 Gammas on the testing set. This measure was improved slightly by assigning a higher

weight to the Hadron class; however, this greatly increased the misclassification of Gammas.

Logistic Regression (with Polynomial Features)

For a second model I trained Scikit-learns Logistic Regression for this binary classification problem. For this model the 'lbfgs' solver was used for over 100,000 iterations with no regularization as it provided the best performance.

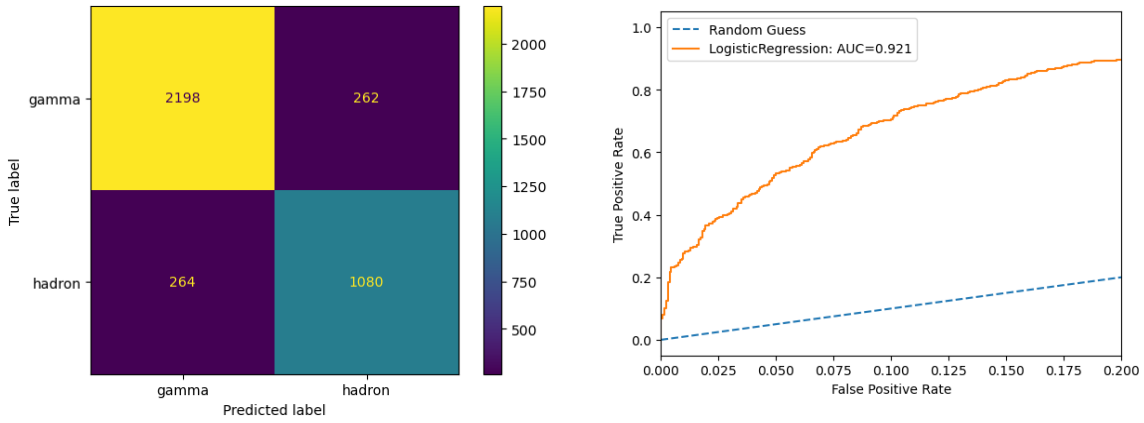
```
Accuracy: 0.8617245005257623
Classification Report:
              precision    recall  f1-score   support

           0       0.80      0.80      0.80      1344
           1       0.89      0.89      0.89      2460

    accuracy                           0.86      3804
   macro avg       0.85      0.85      0.85      3804
weighted avg       0.86      0.86      0.86      3804
```

Figure 8: Logistic Regression Accuracy Report

As can be seen from the classification report on the testing set, the logistic regression performs better than the SGD Perceptron. Additionally, this model's accuracy drops by a few percentage points through the training, validation and testing sets.



Lastly, we see from the confusion matrix that this model reduces the incorrect classification of Gamma by almost half that of the Perceptron's. Again, the false positives could have been minimized at the cost of more false negatives by assigning higher weights to the Hadron class. Additionally, we can see this model provides slightly better classification performance at the left side of the ROC curve than the Perceptron.

Support Vector Machine (with Polynomial Kernel)

I chose to also analyze a Support Vector Machine model as this dataset was not linearly separable. I chose to use a Polynomial Kernel to experiment with higher degree features. This model was trained with features of degree 4, for 3 million iterations of the solver.

Accuracy: 0.8596214511041009
Classification Report:

	precision	recall	f1-score	support
0	0.82	0.78	0.80	1344
1	0.88	0.91	0.89	2460
accuracy			0.86	3804
macro avg	0.85	0.84	0.84	3804
weighted avg	0.86	0.86	0.86	3804

Figure 9: SVM Accuracy Report

This model achieved 86% accuracy on the training set, 86% on validation and 86% on testing.

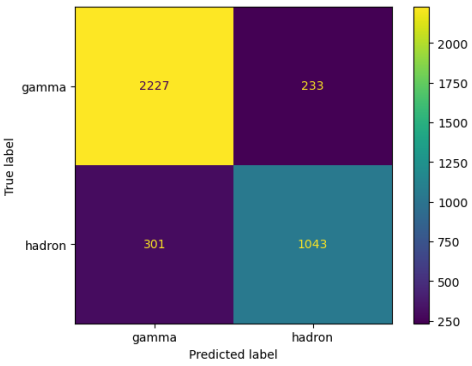


Figure 10:SVM Confusion Matrix

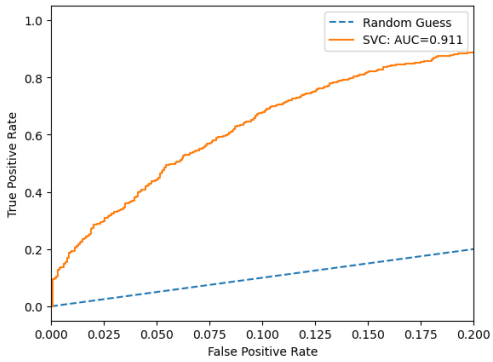


Figure 11: SVM ROC Curve

Analyzing the confusion matrix for this classifier, we can see that even though it achieved a higher accuracy it also misclassifies a higher number of Hadrons than both Logistic Regression and the SGD Perceptron. However, it performs better at diminishing the misclassification of Gammas.

4.3 Analysis

By analyzing the ROC curve, we can easily see that none of these models perform well enough to be used in practice. The reason being that in real world data, the number of Hadron events severely outnumbers the Gamma events by the thousands. Since in practice we care about detecting Gamma signals. This makes our models unscalable in practice as we would be letting in too much contamination into our data by the misclassification of Hadrons as Gamma.

To further visualize how this would create a problem we can think of having a real-world dataset with 1000 Hadron events and 200 Gamma events (As stated in the dataset documentation the Hadron events happen much more frequently than Gamma events). The Logistic Regression model at the most lenient threshold (far right of the ROC curve)

would have 180 (200 * 0.9) True Positives (Actual Gammas) and 200 (1000 * 0.2) False Positives (Hadrons classified as Gammas). We can easily see how this would create useless information as over 50% of the data that we care about is incorrect. This greatly emphasizes the importance of not just considering accuracy as a performance indicator but also individual metrics such as False Positives, False Negatives and the ROC curve.

5. Conclusion

I would like to mention that I consider this project a success even though the results gathered from these experiments are not promising. This made me realize the importance of analyzing features that have a high potential of explaining the end goal. I believe this dataset does not contain enough information to accurately classify Gamma and Hadron events in the real world.

Additionally, the imbalance in this data set created problems with bias towards the majority which is not desirable in this case. I attempted to remedy this issue by balancing the weights for each class, which improved performance slightly for all models. I learned that it is not easy to extract meaningful performance by simply feeding the data to a machine learning model. Careful consideration of different hyperparameters is needed in addition to choosing the correct model for the data.

References

Bock, R.. "MAGIC Gamma Telescope." UCI Machine Learning Repository, 2004, https://doi.org/10.24432/C52C8B.

"The HESS Telescopes." *H.E.S.S. - The High Energy Stereoscopic System*, www.mpi-hd.mpg.de/HESS/pages/about/telescopes/. Accessed 4 May 2025.

Acknowledgements

Documentation: For this project I relied on official documentation provided by Scikit-Learn, Matplotlib, Pandas, Numpy.

Geeksforgeeks: Used Geeksforgeeks website to gain more information about the "class_weight" parameter.

Geeksforgeeks: Used Geeksforgeeks website to learn different methods of plotting a ROC curve.

StatQuest: Utilized StatQuest YouTube channel to gain basic knowledge of the Support Vector Machine algorithm.

NOTE: Most of the information I used for this project came directly from the MAGIC dataset documentation.

Source Code

https://github.com/gabeLin300/MAGIC-Gamma-Classification.git