Algorithmic Event Classification for Atmospheric Cherenkov Telescope Image Data

Ryan McNeil

Background

Imaging Atmospheric Cherenkov Telescopes (IACT) are devices designed to indirectly detect high-energy gamma ray sources by observing the Cherenkov radiation (a form of electromagnetic radiation; detectable as photons) emitted when high-energy gamma particles strike the Earth's atmosphere causing an electromagnetic shower. The effect can be somewhat analogized as light's equivalent to a sonic boom and is useful for studying a range of astronomical phenomena. Since 1989, IACTs have been making the highest-energy detections yet possible in astronomy, facilitating the discovery and study of new very-high-energy sources, such as pulsars, high-energy binaries, and active galactic nuclei.

Four IACT systems currently exist, the one most relevant to this paper being the Major Atmospheric Gamma Imaging Cherenkov (MAGIC) Telescopes, a stereoscopic system of powerful ground-based telescopes located on La Palma, in the Canary Islands. The detection of gamma particles requires combing over a large amount of data. The MAGIC system alone collects about 800 GB of raw data per night of operation, which must be stored on a system of RAID 0 HDDs to prioritize efficiency over reliability. Moreover, the Cherenkov Telescope Array (CTA), an international collaboration to build the next generation of IACTs, is scheduled to begin operation later this year.

Techniques are necessary for reducing and analyzing this image data efficiently. One of the oldest and most common techniques used in gamma-ray imagery is Hillas parameterization, which reduces an image to a set of 5 values defining an ellipse. Over time these parameters have been bolstered by an additional 5 parameters derived from advanced analytical techniques, designed to reduce information loss from parameterization, and geared towards use with stereoscopic imagery. Still, further methods are needed to efficiently and effectively keep up with such a flow of data which can only be expected to increase.

The plethora of data and the challenge of analyzing it has led to a growing relationship between the observational sciences and the field of statistics over the last 30 years. This has been marked by efforts like the *Conference on Advanced Statistical Techniques in Particle Physics* and the growing popularity of the triennial STATPHYS conference. With the need for statistical solutions in the observational sciences expected to increase, advanced statistical methods like machine learning present promising solutions for more effective and more efficient data analysis. This paper aims to present one such solution.

Data Set

The "MAGIC" dataset is a *simulated* dataset generated using a Monte Carlo simulation (nicknamed "Corsika" [4]) of high-energy gamma particle detection in an IACT telescope. Monte Carlo simulations are commonly used in statistical physics for modeling complex systems with an element of random noise. The dataset was used for training algorithms in attempts to devise the best possible classification techniques for IACT imagery. The dataset consists of 19,020 observations, 12332 gamma particle events (signal) and 6688 hadron events (background). Each observation consists of 10 parameters, including Hillas parameters, some of which are represented in fig.

1. fLength: major ½ axis (mm)
2. fWidth: minor ½ axis (mm)
3. fSize: Log_{10} image photon count
4. fConc: $\frac{2\ brightest\ pixels}{image\ size}$ (mm)
5. fConc1: $\frac{brightest\ pixel}{image\ size}$ (mm)
6. fAsym: ellipse center – brightest pixel (mm)
7. fM3Long: $\sqrt[3]{Major\ Axis\ Third\ Moment}$ (mm)
8. fM3Trans: $\sqrt[3]{Minor\ Axis\ Third\ Moment}$ (mm)
9. fAlpha: angle of major axis (deg)
10. fDist: ellipse center – origin (mm)

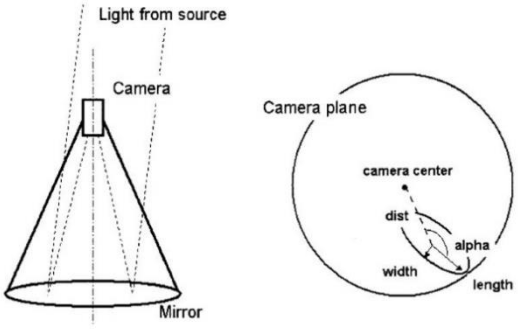

Fig. 1: Cherenkov Telescope Image Parameters [1]

Understanding the exact meaning of each of these parameters is not necessary for readers, or even necessarily for other researchers. These parameters have been taken as given by prior work, as they were in the original study, Bock et al. 2004. Efforts to optimize these values further to minimize information loss and maximize usefulness in modeling are outside the scope of both this paper and that of Bock et al. 2004 on which it builds. As we will see later, further transformation of these values seems to yield mixed results and was disregarded in Bock et al. 2004.

This dataset was chosen here (as it was in the Bock et al. 2004) because it appears to be the most manageable and representative publicly-available dataset for gamma particle classification, which itself represents a significant area of overlap between astrophysics and statistics. The Monte Carlo simulation has been optimized to maximize the representativeness of the simulated images, which have already been cleaned and parameterized for the final dataset.

Data Cleaning

Our dataset was relatively clean to begin with, and only minor changes were needed to prepare it for analysis. Given that this dataset is simulated, there were no missing data points. Even if this had been based on real observational data, there likely be no missing data points unless some feature of the telescope or storage system had malfunctioned. Given the high variation in our data and the amount of noise in real imagery, it would have been best to drop any rows containing these hypothetical missing values, rather than imputing missing values.

The original “MAGIC04.data” file is read into Python as a data frame using the Pandas package. This can be done either from a local system, by updating the hashed-out directory command, or from the online repository as in the default code. All parameters read in correctly as floats, and only column names and our target class need to be defined. Our target class “gamma” is manually coded to an integer variable of 1 for gamma particle events (signal) and 0 for hadron

events (background). This is done only as a matter of preference. LabelEncoder() could have been used to do the same, but would encode 0 for signal and 1 for background.

Lastly, we had experimented with log-normalizing some of our highest-variance parameters, fLength and fWidth. While Log_{10} normalization *did* reduce variance several orders of magnitude (i.e. fLength's variance was reduced from 1794.68 to 0.46), these normalized values yielded either comparable or *worse* performance from our supervised learning algorithms later on and were dropped in favor of the original un-normalized values. Although normalization tends to be more effective for methods like kNN, (which deal with absolute values, rather than just order), it yielded only comparable performance in our kNN model. In a random forest (do not necessitate normalization), normalization reduced the accuracy of our model by several percent. This could be due to several phenomena. For one, normalizing focal length and width, which here represent real and comparable physical distances could destroy meaningful information and obscure the relationship between them and other physical measurements. In Bock et al. 2004, this data was also not normalized before analysis. The Hillas parameters were considered to have already been given in a particular form for analysis (including one already log-normalized value, fSize, and two root-normalized values, fM3Long and fM3Trans).

Data Visualization

Before testing split our full dataset to simulate not having access to testing data in a real-world scenario. We use test_train_split() with default parameters (75% for training, 25% for testing), stratifying on our target class. Stratification will ensure that the proportion of signal to background remains the same in our training and testing sets as it is in our full data. In fig. 2, we can see that our testing data set contains just under twice as many gamma particle events as hadron events. Hadron

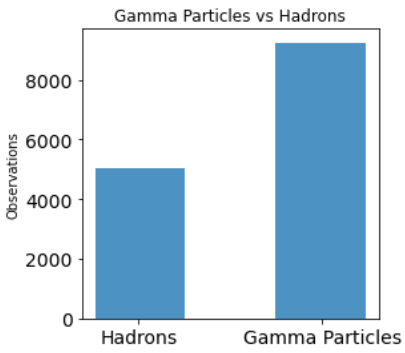

Fig. 2: Target class distribution in y_train

events compose approximately 35% of our total observations, and gamma particle events the remaining 65%.

This means that random guessing would yield an accuracy of 65%, making that our absolute minimum accuracy to beat. Ultimately, we want to meet or exceed the maximum accuracy of 85.2% achieved by the optimal model in Bock et al. 2004.

We also build a scatter matrix of the X variables in our data. A default scatter_matrix() of X_train generates a cluttered and difficult-to-read plot, so we include code to build a custom scatter matrix, the output of which can be seen in fig. 3. Unfortunately, most of the relationships here either offer very little insight, or show no discernable pattern. This does allow us to better choose variables for cluster analysis later. *Some* intuitive patterns appear, such as fLength & fWidth appearing to have something of a linear correlation, fConc & fConc1 having a strong linear correlation. We can see from the density curves of fSize, fConc & fConc1 that observations seem to concentrate at relatively lower values, with this dataset including a relatively small number of particularly high-brightness outliers. From those of fAsym, fM3Long & fM3Trans, we can also see that most of these values tend to be within relatively narrow bands. This likely represents the fact that events (whether signal or background) are more likely to be registered closer to the center of the telescope's center of focus.

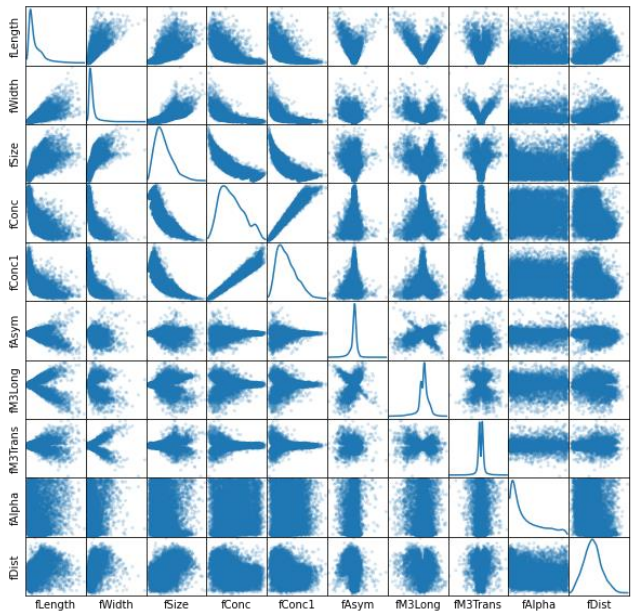

Fig. 3: Scatter matrix of X training data

To further inspect the relationships depicted in fig. 3, we recreated several of the subplots with markers added for target class. We discovered that gamma particles tend to lie within a narrower range of values than hadrons along many of our X variables. This reinforces the accepted norm that hadron and gamma particle events should distribute differently along Hillas parameters. Unfortunately, we also see a lot of overlap in values between either class when plotted on just 2 axes. This suggests that dimensionality reduction may prove challenging, since many dimensions may be necessary for clearly drawing regions within which either class lies. Fig. 4 shows one representative example using fLength and fWidth.

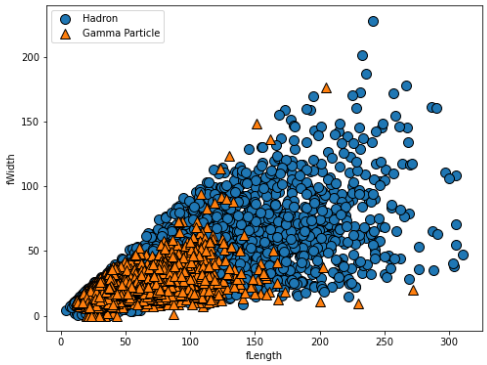

Fig. 4: fLength vs fWidth with markers for target class

As aforementioned, we elected not to normalize values for our analysis. However, normalization *did* give some values a more normal distribution. As part of our experimentation with normalization, we had visualized the distribution of some of our variables. For one example in fig. 5, we can see the way that the distribution of fLength is normalized by Log_{10} normalization.

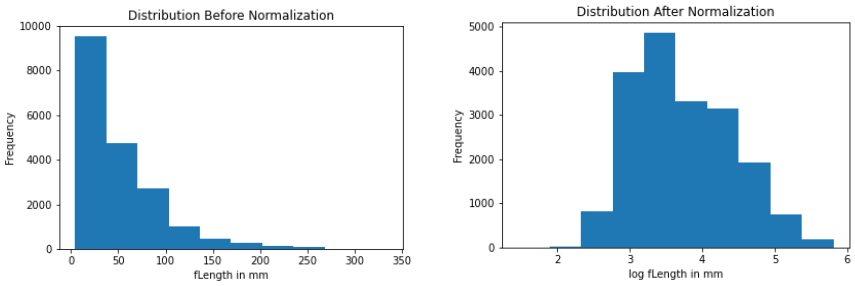

Fig. 5 Distribution before and after normalization

Supervised Learning - kNN

We began testing supervised learning models with kNN, a method which yielded sub-optimal results in Bock et al. 2004. Although we eventually elected to use non-normalized data, we had initially attempted to run kNN on a log-normalized transformation of our data. This produced results only comparable or slightly worse than running kNN on our non-normalized data. This dataset initially seemed suited for kNN, as it is relatively small, and has already been reduced to a relatively small number of dimensions.

Our first model began with default hyperparameters and 4 neighbors. Without any additional tuning, kNN achieves a training accuracy of about 85% (and a testing accuracy of about 79%). This is sub-par performance for our purposes so we begin optimizing by testing a range of neighbors values and plot their training and testing accuracy. We first investigated up to 50 neighbors, but this visualization showed little variation past about 10, so we repeat with a range of 12. From fig. 6, we can see that our testing accuracy tapers (and even decreases slightly) after about 7. Increasing the number of neighbors past this value may only lead to underfitting. We repeated the process with n_neighbors = 7. We evaluate this new model using 5-fold cross-validation on our training data, which finds an average accuracy of about 81%. Our CV scores all fall between 79% and 81.5%, suggesting that no one cut of our data contained any values which significantly effected the performance of our kNN model.

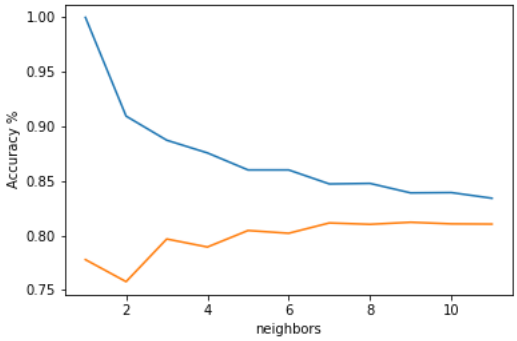

Fig. 6: kNN accuracy by number neighbors

We automate the optimization of our model by implementing grid search. Our grid consists of a range of neighbors values, weighting methods, and kNN algorithms. Unfortunately, our optimal model still only achieves about 81.6%

accuracy, identical to the highest accuracy achieved by the optimal kNN model in Bock et al. 2004.

	Pred. Neg.	Pred. Pos.
Act. Neg.	982	690
Act. Pos.	186	2897

Table 1: Optimal kNN confusion matrix

	Precision	Recall	F1
Hadron	0.84	0.59	0.69
G.P.	0.81	0.94	0.87

Table 2: Optimal kNN classification report

From both a confusion matrix and classification report of our optimal kNN, we see that our model is more likely to register false positives than false negatives. For our context, this is actually desirable. Since it is more costly to miss gamma particle events than it is to flag hadron events unnecessarily, the more informative metric may be *recall*. We want to make sure we are capturing as many actual positives as possible. A recall of 94% for gamma particle events may speak well for this model overall.

Lastly, we evaluated our model by with a *Receiver Operator Characteristic* (ROC) curve of our optimal model. The angle of our curve (AUC = 0.86) reflects that our model *does* have predictive value when scored on our testing set. Judging from this curve, the optimal sensitivity and specificity (closest to the top left corner) seems to fall between a TPR of around 80%, and an FPR of around 25%.

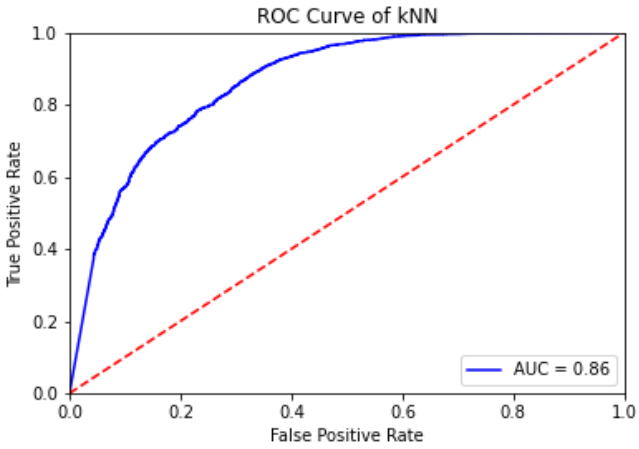

Fig. 7: kNN ROC curve

Neither *our* transformations, the more advanced transformations applied in Bock et al. 2004, nor any method of optimization in either were able to produce a kNN model with an accuracy exceeding 81.6%. We are left to suspect that kNN simply cannot capture the full complexity of phenomena in our data beyond a certain threshold. The cause could be the high variability in this data, since kNN tends to be sensitive to outliers. Alternatively, any complex multidimensional patterns in this data may be better handled by another classification method,

since kNN tends to perform best on narrower (and shorter) data. kNN would also be of limited use for real-world application on this type of data because we would ultimately need our algorithm to run on *much* larger bodies of data, and kNN is very costly in terms of time and memory for very large datasets.

Supervised Learning – Forests & GBDTs

Next we tested a combination of random forests and gradient-boosted decision trees in order to find a more optimal model. We believed random forests may be better for interpreting this data since they are robust to outliers, tend to run efficiently on large datasets and the data it would be applied to in the real-world is unlikely to ever have missing or sparse features. As part of this section, we also elected to test GBDTs as a method of comparison for several reasons. For one, it is unclear whether boosting may be more or less effective than bagging for analyzing this data. We wanted to investigate whether GBDTs may be able to uncover phenomena in our dataset not observed by RF models. GBDTs may also work better on datasets with unbalanced class distributions, as is the case with ours. Both methods would be more computationally expensive than some other methods in the training and optimization phases, with the tradeoff of being more efficient and effective on large datasets in final application.

Beginning with default parameters and 100 estimators, forests and GBDTs already achieve testing accuracies of 88.5% and 87% respectively. We investigate the relative feature importances in either model. Fig. 8 shows that the most

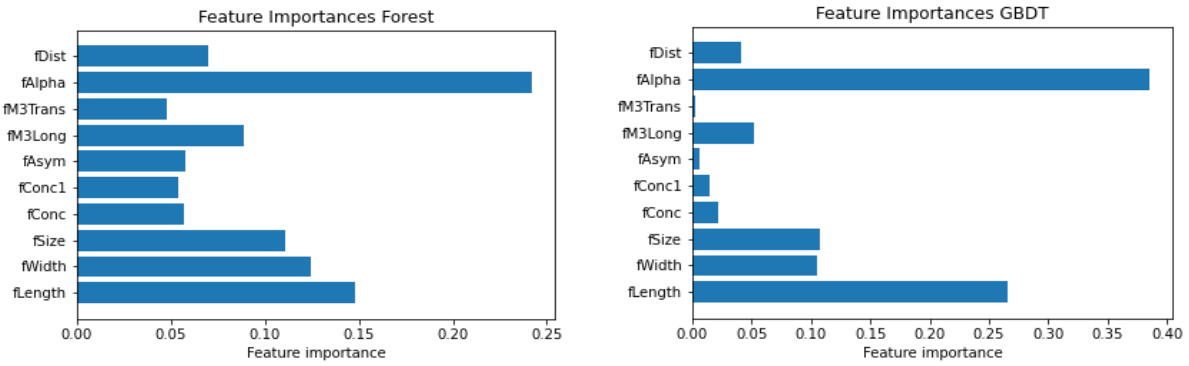

Fig. 8: Feature importances

important feature, by far, appears to be fAlpha, the angle of our major axis with vector to origin. This could mean one of two things: 1) the ability of a detector to identify gamma particle events hinges heavily on the angle at which a shower propagates with respect to the detector, or 2) gamma particle events and hadron events actually tend to propagate at different orientations with respect to the detector. The next most important feature, fLength (and then fWidth), further implies that gamma particle and hadron events tend to be characterized by different shapes in the detector.

Combined with our earlier investigation of the distribution of classes against fLength and fWidth, it would appear that gamma particle events tend to be characterized by tighter distribution in physical space than hadron events. Also of note, the 3rd/4th most important feature, fSize, implies that gamma particle and hadron events are characterized by different average photon counts.

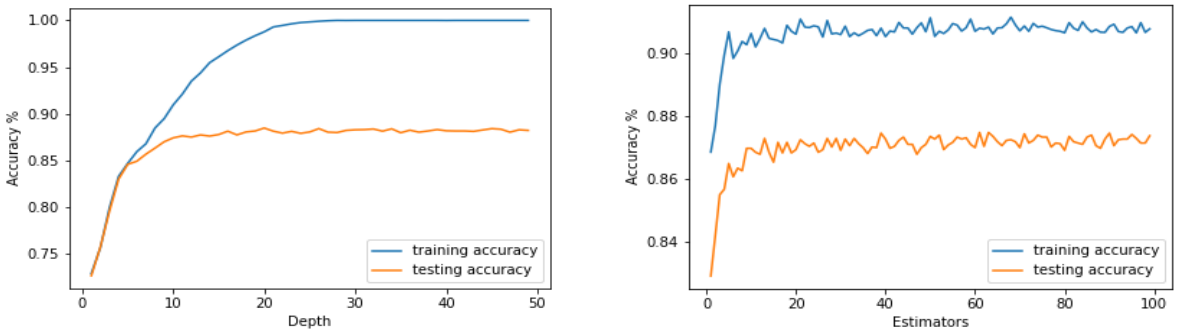

Fig. 9: RF hyperparameter accuracies

For optimization, we begin by investigating the relative predictive power of different values of some of our hyperparameters. For brevity, we will only review plots for our RF model in this paper. Fig. 9 shows that increasing the maximum depth and number of estimators stops improving testing accuracy past a depth of about 10, and n_estimators of about 15. Past this point, (particularly in the case of max_depth), increasing our hyperparameter values may only lead to overfitting. With this in mind, we implement grid search on both RF and GBDT with ranges of estimators, criterion options, max features and max depth for both (plus ranges for loss options and learning rate for GBDT). Note, if you wish to run this code yourself, it may take quite some time

to run on a PC. The optimal random forest model ultimately achieves an accuracy of about 87%, and the optimal GBDT an accuracy of about 85%.

Optimal Random Forest

	Pred. Neg.	Pred. Pos.
Act. Neg.	1197	475
Act. Pos	150	2933

Table 3: Optimal RF confusion matrix

Optimal GBDT

	Pred. Neg.	Pred. Pos.
Act. Neg.	1031	641
Act. Pos	79	3004

Table 4: Optimal GBDT confusion matrix

	Precision	Recall	F1
Hadron	0.89	0.72	0.79
G.P.	0.86	0.95	0.90

Table 5: Optimal RF classification report

	Precision	Recall	F1
Hadron	0.93	0.62	0.74
G.P.	0.82	0.97	0.89

Table 6: Optimal GBDT classification report

Confusion matrices and classification reports show that both models are more likely to return false positives than false negatives. While the optimal forest achieves slightly better accuracy and precision for gamma particle events, our optimal GBDT stands out as achieving the highest recall (97%) of any model without reverting to random guess accuracy. Unfortunately, this seems to come at the cost of many false positives and a low 62% recall for hadron events. Though, both of these models still achieve higher scores across the board than our optimal kNN model.

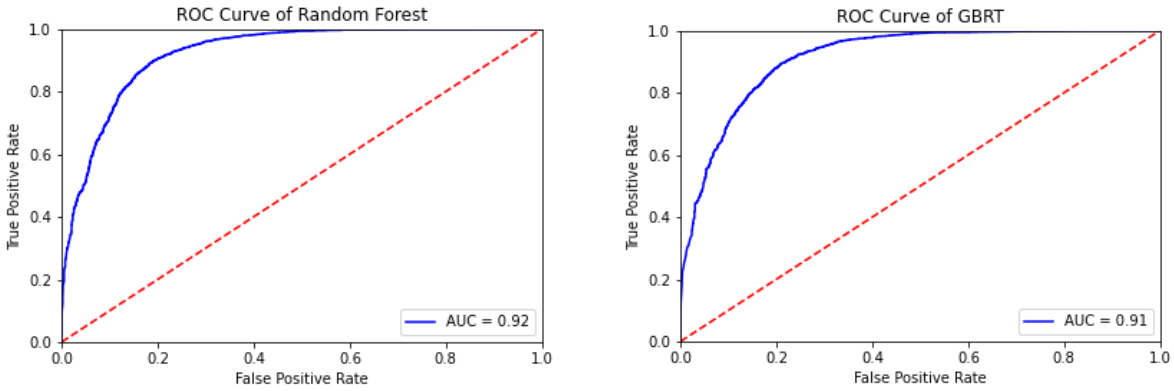

Figure 10: ROC curves of optimal RF and GBRT models

Fig. 10 shows that RF achieves a slightly higher AUC, and both models show relatively high and tight curves, reflecting significant predictive value (higher than our optimal kNN). Both curves appear to show optimal TPRs between 0.8 and 0.9 at FPRs between 0.1 and 0.2. Ultimately, we take the optimal RF as our

"best" model. This best model has at this point been able to beat the highest accuracy achieved by the best model in Bock et al. 2004 (85%), and appears to perform well overall. These results suggest that neither bagging nor boosting produce significantly different results, by extension suggesting that outliers (since they would particularly be a detriment to boosting methods) may not be affecting performance as much as the overall complexity of the data.

Unsupervised Learning - PCA

In order to both investigate our dataset further and assess the possibility of dimentionality reduction, we implement PCA. Our data was prepared for PCA implementation with standard scaler, then a test PCA was fitted to our scaled training data set to 2 components.

Fig. 11 shows the scatter plot of these two components with markers for class. The distribution clearly shows that gamma particle and hadron events do not cluster differently along these two dimensions. While gamma particle events tend to be found within a narrower band of values than hadron events, both still seem to largely overlap. The composition of our two components in Fig. 12 shows that our first component is mostly composed of fLength, fWidth, and fSize (three of our most significant factors in our RF and GBDT models) [correlating in the positive], and fConc and fConc1 (two of our

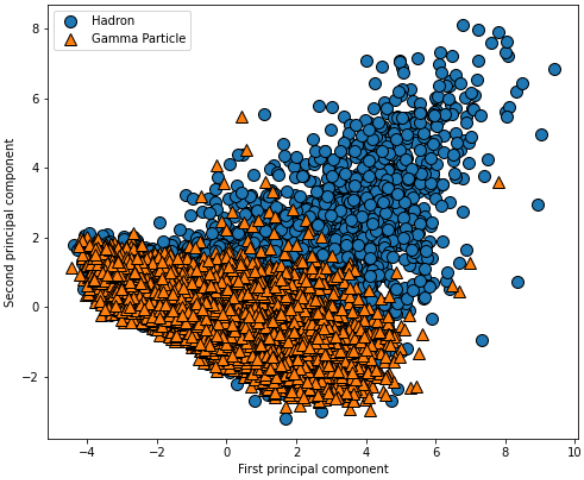

Fig. 11: Class distribution across first 2 principal components

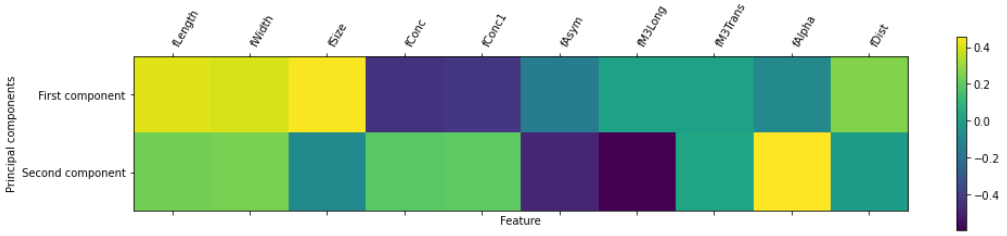

Fig. 12: First two principal components composition

least significant factors in our RF and GBDT models) [correlating in the negative]. Component 2 is mostly formed from fAlpha [correlating in the positive] and fAsym and fM3Long [correlating in the negative]. The remainder, fM3Trans and fDist do not appear to correlate very strongly with either, appearing to be split between the two principal components.

Re-running PCA with hyperparameters set to capture 95% variance determines that our data can only be reduced to 7 components without dropping below this threshold. Interestingly, this is also the optimal number of maximum features selected by gridsearch for both our kNN and RF models. This serves as evidence that our predictive power is spread out between most of our featues, and verly little can be removed without losing meaningful information. Further, much of our predictive power *may* lie in the 7 or 8 most important features in our earlier feature importances (Fig. 8). This may also stand to reason since at least two of our low-relative-importance features, fConc1 and fM3Trans, are closely related to other values (fConc and fM3Long, respectively) and may not capture much information that is not already captured better by those other features. We next assess whether scaling and PCA improve the performance of our best model.

	Precision	Recall	F1
Hadron	0.89	0.72	0.80
G.P.	0.86	0.95	0.91

Table 7: No scaling or PCA

	Precision	Recall	F1
Hadron	0.90	0.73	0.80
G.P.	0.87	0.95	0.91

Table 8: Scaled but without PCA

	Precision	Recall	F1
Hadron	0.82	0.62	0.71
G.P.	0.82	0.92	0.87

Table 9: With scaling and PCA

Tables 7-9 show that scaling does slightly improve the performance of our optimal random forest, but the addition of PCA considerably reduces performance (even with 7 features). Accuracy scores also find scaling produces our best model, with an accuracy of 90%. Our optimal random forest classifier fit to scaled training data composes the final “Gamma_Algorithm” provided in the algorithm repository.

Clustering – K-Means

Though our data does not appear to lend itself well to cluster analysis in any 2 dimensions, we proceed with methods of cluster analysis in an attempt to find

any hidden patterns not immediately apparent. We apply cluster analysis methods to both our 7-component PCA-transformed data and to our original data, looking specifically at the variables fAsym and fM3Long.

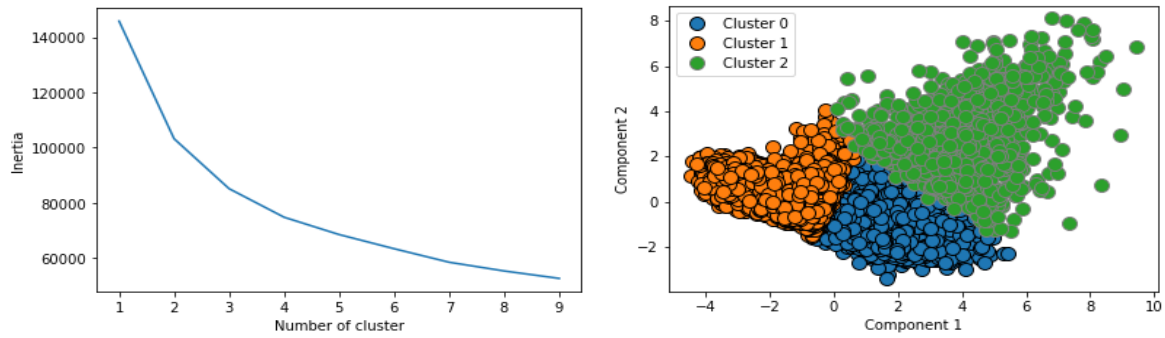

Fig 13: K-Means with PCA

Fig. 13 Shows that overall inertia continues to fall towards high numbers of clusters (with a very-high-value y-axis overall). It does not appear possible to meaningfully minimize the Euclidean distance between points without devolving to infinitely smaller and less meaningful groups. We choose to visualize 3 clusters with Euclidean distance because it is the only value tested which appears to output anything potentially meaningful. Neither other cluster values, nor attempts at using Manhattan distance yielded anything meaningful. The pattern of possible note here is the border between component 2 and components 0 and 1. This pattern appears to vaguely mirror the distribution of gamma particles and hadrons in fig. 11. By definition however, KMeans cannot distinguish the mix of classes within those region, and increasing the number of clusters does not produce a clearer distinction.

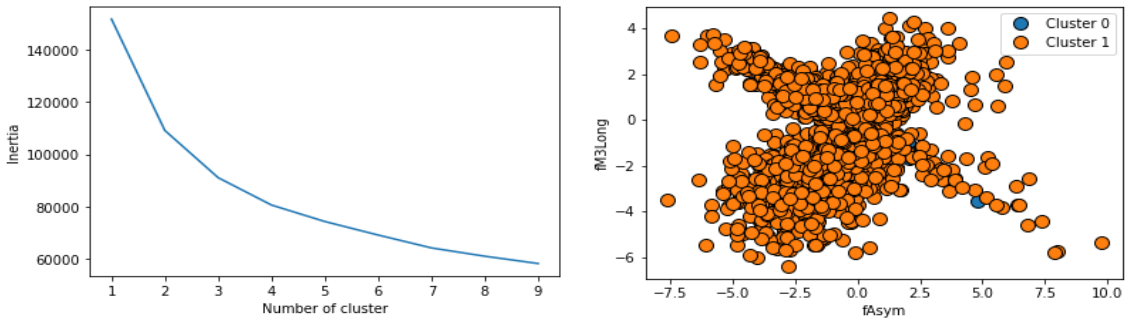

Fig 14: K-Means without PCA

Fig. 14 shows the same process attempted for 2 clusters and visualized along the variables fAlpha and fM3Long. These variables were chosen simply because they form the most irregular shape of any of our variables plotted against one another (see fig. 3). We did attempt all of these methods of cluster analysis on other combinations of factors, but none produced insightful visualizations. Fig. 14 shows that as with PCA, inertia remains high even at high cluster counts and KMeans does not appear to be capable of identifying meaningful clusters. KMeans with 2 clusters and Euclidean distance is visualized here as higher cluster counts and Manhattan distance only yielded more and more cluttered and meaningless visualizations. We can safely say that even taking into account additional un-visualized dimensions, very little meaningful clustering appears to be occurring in our data both with and without PCA.

Clustering – Agglomerate

Next, we tested agglomerate (or hierarchical) clustering. This was chosen as it would allow us to visualize our clustering differently than either of the other methods attempted. Given the structure of our data, it was not possible to properly visualize the sample index here. However, our analysis found no discernable meaningful clustering in this data. Fig. 15 presents the results of agglomerative cluster analysis on our PCA data with 3 clusters for the sake of a clear branching pattern. Cluster analysis was initially attempted here with only 2 clusters, in an attempt to hierarchically distinguish between hadron and gamma particle events. This did not seem to uncover anything informative about our data, nor did it appear to distinguish well between classes. The spatial distribution of these points along the first two components (visualized later in fig. 19) appears similar to our graph of KMeans.

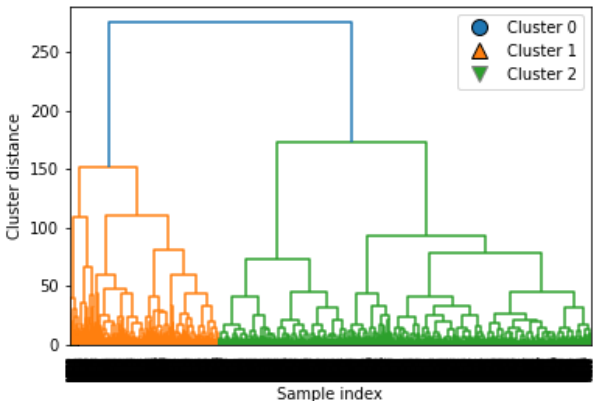

Fig. 15: Agglomerative clustering with PCA

We then attempted the same process on our non-PCA data. Unfortunately, this proved even more difficult to interpret than agglomerative analysis of our PCA data. Little insight could be gleaned. One interesting pattern is that our PCA *and* non-pca dendrograms seem to show unbalanced cluster sizes. More advanced analysis may be able to determine whether these clusters mirror our class distribution (which is also unbalanced). Having uncovered little with this method, we move on to the method we anticipated being most informative of those applied.

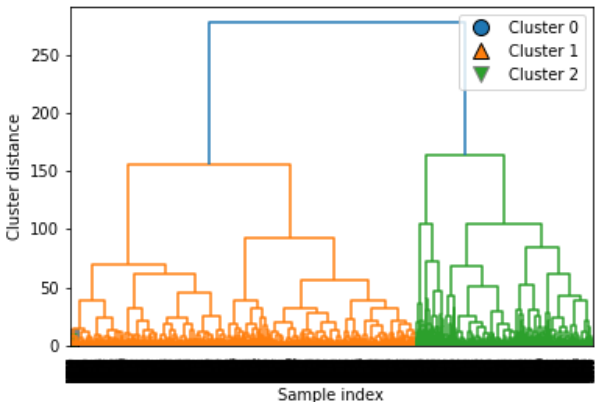

Fig. 16: Agglomerative clustering without PCA

Clustering – DBSCAN

As with previous methods, we began by fitting DBSCAN to our PCA-transformed data. This method seems to have produced the most interesting results of any method attempted. Particularly because the clustering pattern found in fig. 17 appears to mimic somewhat the distribution of our target class in fig. 11 (as KMeans *somewhat* did as well). It seems possible that DBSCAN is picking up on some potentially meaningful clustering across our 7 principal components that is not immediately apparent in 2 dimensions. This is further reinforced by the fact that there appears to be overlap between these clusters when represented in only 2 dimensions. A complication to this theory is that DBSCAN decided on 3 clusters, rather than 2. Alpha has been reduced in fig. 17 to more clearly display the points in the smallest cluster (red circles on the middle-left side of our graph). This smallest cluster does not seem to match any

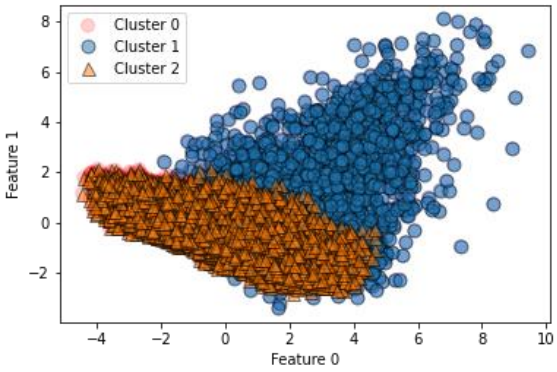

Fig. 17: DBSCAN with PCA

patterns observed elsewhere in our exploration. Future research may find further exploration of clustering in this data to be fruitful after all.

Lastly, we ran DBSCAN on our non-PCA data. As we suspected, it could not find any meaningful phenomena. Note that while this graph has legend items for 2 clusters, one of those clusters consists of a single point, meaning DBSCAN essentially found no clustering. This may still be informative. The fact that DBSCAN could find clustering which may be meaningful in the PCA-transformed dataset could suggest that PCA is useful for identifying clustering in this type of data, even if it is not useful for improving algorithm performance.

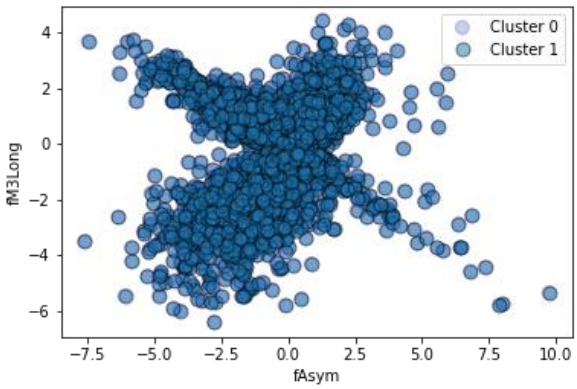

Fig. 18: DBSCAN without PCA

ARI & Silhouette Scores

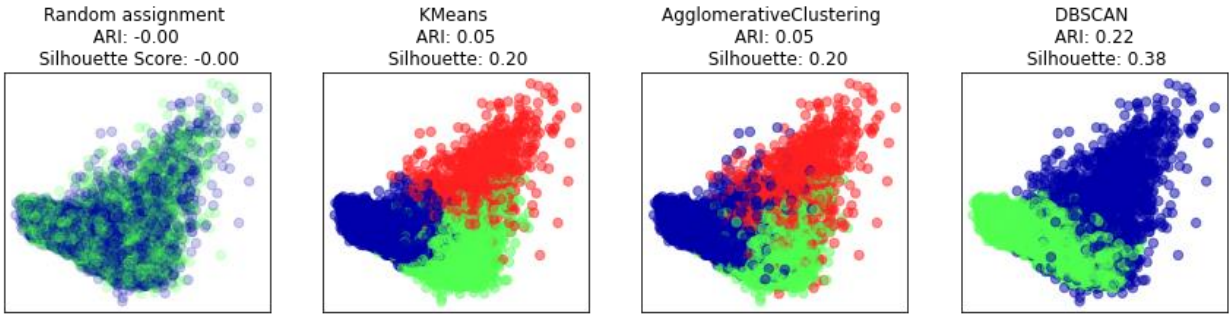

Fig 19: ARI & Silhouette scores with PCA

We further investigate our cluster analysis methods by comparing ARI and Silhouette scores for each method compared to random assignment. Fig 19 shows each method applied to the plot of our first two components. None of our methods have particularly good performance on our PCA-transformed data. DBSCAN perform the best of any method, but still poorly overall. The Adjusted Rand Index (ARI) of all methods would be considered low, meaning that clusters tend to not be very different from one another. Silhouette scores are also low,

meaning that the distance between clusters does not tend to be very significant.

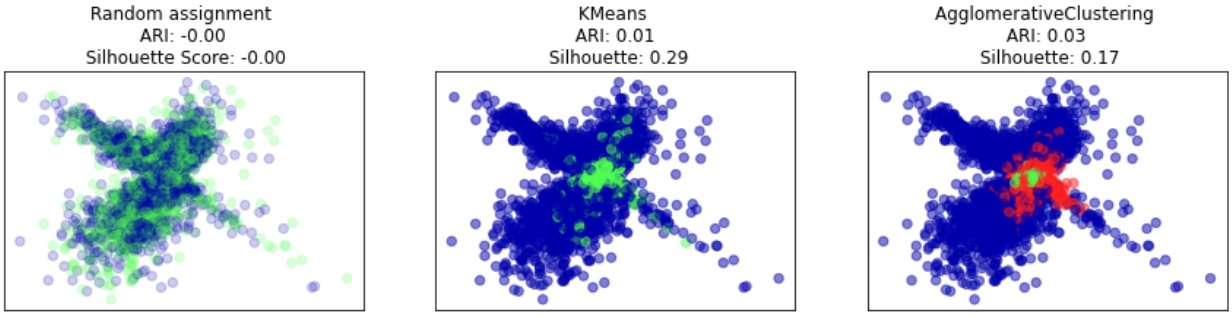

Fig 20: ARI & Silhouette scores without PCA

We perform the same analysis on our non-PCA clustering. DBSCAN method had to be omitted here due to the lack of clusters detected, which would produce an output with undefined ARI and Silhouette scores. Fig. 20 shows each method with alpha reduced to better show overlap in 2 dimensions. Given the overlap in our graphs, we might still suspect that higher-dimensional clustering may be occurring. However, our ARI and Silhouette scores are roughly as poor as they were for our PCA clustering methods. From these graphs in combination with the ARI and Silhouette scores, it is apparent that little meaningful clustering can be detected anywhere in our non-PCA data.

Summary

We have successfully produced an improved algorithm that detects gamma particle events in simulated Hillas-parameterized IACT image data with an accuracy of 90% and a recall of 95%. We observe from the above tested methods that while most algorithms produce comparable or worse results than the optimal model in Bock et al. 2004, our optimized model operating on scaled data can improve detection accuracy by up to 5%. Further gains in recall can be made by other models, but at the cost of their decreased performance on other metrics (primarily high rates of false positives). Additional improvements may be possible if this algorithm were trained on a much larger body of data (approximating those it would be applied to) or if an RF model were optimized on original non-parameterized image data, thus accounting for any information

lost in parameterization. However, this would prove a much more time and hardware intensive process, compounding the relative time and hardware intensiveness of RFs compared to other algorithms. Despite these drawbacks, RF was also chosen as the optimal method in Bock et al. 2004. For the reasons discussed earlier, and given the success of this algorithm here and in Bock et al. 2004, RF is likely the best algorithm for this type of data. Our optimal model has been chosen purely on performance metrics like accuracy, precision, and recall. Further experimentation with other algorithmic methods should take into account the trade-off between accuracy and efficiency, especially in use with real-world datasets. Further research may also utilize additional parameterization techniques or more advanced analytical methods to investigate multi-dimensional clustering observed by DBSCAN in our PCA dataset. Such research could begin by comparing cluster assignments to target class values. If clustering and target class broadly fall along the same lines, this could provide an opportunity for further optimizing classification models.

References

[1] Bock, R.K., A. Chilingarian, M. Gaug, F. Hakl, T. Hengestebeck, M. Jirina, J. Klaschka, E. Kotrc, P. Savický, S. Towers, A. Vaiciulis, W. Wittek. "Methods for multidimensional event classification: a case study using images from a Cherenkov gamma-ray telescope", *Nuclear Instruments and Methods in Physics Research*. August, 2004.

[2] A. Hillas, "Cerenkov light images of EAS produced by primary gamma", Proc. 19nd I.C.R.C., 1985

[3] Mathieu de Naurois. "Analysis methods for Atmospheric Cerenkov Telescopes", *Universités Paris*. June, 2008.

[4] Heck, D. et al., "CORSIKA, a Monte Carlo code to simulate extensive air showers", *Forshungszentrum Karlsruhe*, 1998