What is the best classifying technique in Python's scikit-learn?

1. Introduction

The aim of this project is to undertake various popular classification methods which are used to predict a category of a data point within Python of a dataset, through which a comparison of techniques can be made.

The techniques used in this project are:

- Logistic Regression
- Linear Discriminant Analysis (LDA and QDA)
- Decision Trees
- Random Forests
- Gradient Boosted Decision Trees
- XGBoosted Decision Trees
- Support Vector Machines (SVM)
- Neural Networks

These methods will be compared using different metrics, in particular:

- **Accuracy**: models' ability to correctly predict both classes.
- **Precision**: model's ability to correctly detect positive classes from all predicted positive classes
- **Sensitivity (Recall)**: models' ability to correctly detect positive classes from all actual positive classes
- **F1 Score**: weighted average of precision and recall (used for unbalanced problems).

1.1 Background

We are going to use techniques in the scikit-learn library for python. There is a popular cheat sheet which guides the statistician which technique best suits their dataset in order to maintain the best possible performance. The cheat sheet suggests using 6 of the techniques previously mentioned. It is quite often dataset-specific as to what is the best statistical technique to use and many statisticians use may techniques in order to achieve a best possible model.

1.2 Grid Search

In this report, for each classifying technique a "grid search" is used – also known as parameter tuning i.e. trying many different parameters for each statistical technique and using the parameters for that technique that yield the best accuracy. Parameters selected for the grid search are kept as equal as possible amongst the models in order to produce a fair comparison.

Accuracy is used to select the best parameters in the grid search since this is a balanced problem and there is no justification to favour the detection of positive cases or negative (in some medical studies this can be the case).

2. The Data

The data set consists of simulated data on high energy gamma particles in an atmospheric Cherenkov telescope, extracted from extracted from the MAGIC Gamma Telescope data set (https://archive.ics.uci.edu/ml/datasets/magic+gamma+telescope).

As particles pass through the telescope, a shower of electromagnetic radiation is produced, which are approximated by elliptical shapes. The goal is to distinguish the showers which arise from gamma particles from those which come from hadrons. The data has 11 columns, 10 continuous variables in the first 10 columns, and a class label, g or h, in the final column.

The data is balanced and therefore the Accuracy statistic can be used to judge the performance of the statistical technique.

2.1 Preparation

The data doesn't contain any missing values and is a balance dataset. Each variable is normalised, making it's mean and standard variance equal to 0 and 1 respectively. This is necessary to do for a neural network and a support vector machine but for consistency the standardised variables are used for all techniques.

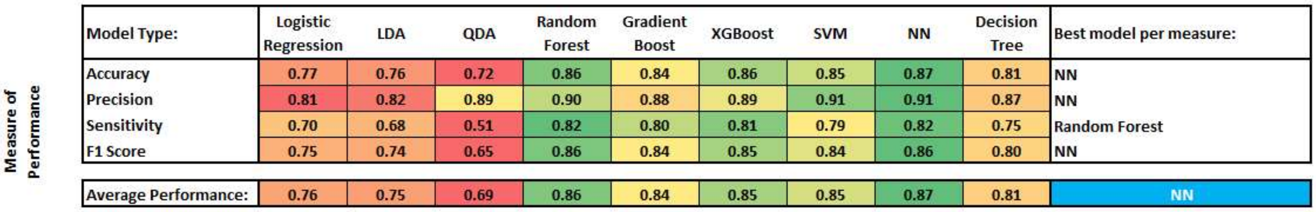
A dataframe is created which only includes the “class” variable – whether it’s a gamma or hadron particle (what we’re trying to classify). Another dataframe is created including all the standardised predictor variables.

Each of the dataframes’ rows are randomly split by a 75:25 ratio, forming train and test datasets respectively. The analysis will be carried out using the train datasets and evaluated using the test datasets.

3. Analysis

3.1 Results

The table below shows how each classifying technique performed across each performance metric. Red cells indicate relatively poor scores and green cells indicate good scores. The last column shows which classifier performed the best and the cell in blue, the bottom-right most cell shows the classifier that has the best score averaged across all the metrics. The Neural Network model is crowned the champion here.



	Model Type:	Logistic Regression	LDA	QDA	Random Forest	Gradient Boost	XGBoost	SVM	NN	Decision Tree	Best model per measure:
Measure of Performance	Accuracy	0.77	0.76	0.72	0.86	0.84	0.86	0.85	0.87	0.81	NN
	Precision	0.81	0.82	0.89	0.90	0.88	0.89	0.91	0.91	0.87	NN
	Sensitivity	0.70	0.68	0.51	0.82	0.80	0.81	0.79	0.82	0.75	Random Forest
	F1 Score	0.75	0.74	0.65	0.86	0.84	0.85	0.84	0.86	0.80	NN
	Average Performance:	0.76	0.75	0.69	0.86	0.84	0.85	0.85	0.87	0.81	NN

Fig 3.1a: Table of performance measures for each classifier, with the best overall classifier in the blue cell.

Neural Networks and SVM scored well in precision. This could be a good classifier to use then if identifying true positives is of particular interest. QDA performs quite badly and LDA along with Logistic Regression perform relatively poorly as expected due to their simple algorithms. The Decision Tree and perform fairly well, Gradient boost and SVM perform better and Random Forests, XGBoosts and Neural Networks perform really well.

Why QDA performs so poorly can be explained by the fact it overfits the data by allowing for a non-linear quadratic decision boundary. However, the results are extremely poor in contrast with the other methods that perhaps isn't fully explained by overfitting, especially since the accuracy on the trained data wasn't good. This is a good example that supports the often used strategy by statisticians of running the data through different algorithms to allow for quirks such as this.

The tree methods perform better than the decision boundary methods (Logistic, LDA, QDA) by bisecting the space of data points into smaller and smaller sections, allowing for complex relationships.

Random Forests (Gradient Boosted and XGBoost) build on decision trees by producing many decision trees thereby limiting overfitting and the error, and we see they perform better.

SVM performs well with it's complex algorithm and non-linear decision boundary allowing for complex relationships.

Neural Networks performs the best, again allowing for complex non-linear relationships and using the extremely effective backpropagation algorithm.

Shown below are the confusion matrices for all the models. Highlighted in red is the Neural Network confusion matrix which performs the best. These show the percentage of the data correctly predicted (top left and bottom right) and the percentage incorrectly predicted (top right and bottom left).

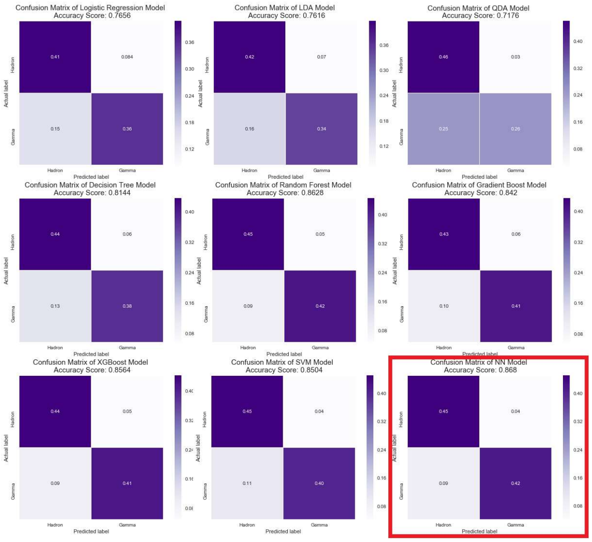


Fig 3.1b: Confusion matrices for each classifier. The red box highlights the best classifier, the Neural Network. Notice the deep purple colours contrasted with the almost white other cuadrants. This contrast demonstrates the accuracy of the Neural Network.

3.2 Learnings

This analysis brought up some interesting traits and goes some way to answer why a random forest might be a good option to use when classifying a data point - and that trait lies in the nature of the grid search.

One of the problems with grid search is the time it takes to carry out the amount of models which correspond to the combinations of parameters entered in by the user to the grid search. In this particular report, the initial grid search for the Gradient Boost model failed to complete after being ran for 18 hours. That meant reducive the number of paramters selected for the more complicated, computationally expensive classifiers so that the grid search wouldn't take as long to get results.

This reduction of the amount of parameters was carried out with Gradient Boost, XGBoost, SVM and Neural Network classifiers. However it was not carried out with the rest.

The relatively moderately complicated Random Forest was allowed to run with a good amount of parameters - and this took 1 hour to run.

The accuracy of Neural Network is highly dependent on the number of epochs (cycle through the training set) selected. Changing the number of epochs in the grid search increases the time required to run, but in what was observed to be a predictable fashion (unlike changing the number of parameters of Gradient Boost which would increase the running time exponentially in the literary sense). For example, changing the epochs from 5 to 500 increased the running time from 2 minutes to 90 minutes; increasing the accurady from 0.82 to 0.87. That increase is worth waiting for.

For that reason Neural Networks offers a way to fine tune the complexity without the fear of the grid search taking a lot of time.

The conclusions here would be, if a statistician is concerned with running time of the grid search, the Random Forest and Neural Network offers a good balance between complexity of the algorithm and speed. In particular the Random Forest classifier is allowed to experiment with many more parameters in a short space of time.

All these higher scoring classifiers however come at a price - interpretability. Logistic Regression is a highly interpretable model whereas the other models are a bit of a "black box", especially SVM and Neural Network. A Random Forest can be explained with difficulity to the layman, but a Neural Network and SVM are nigh on impossible. If interpretability is paramount, a Random Forest might serve the user better in this case.

3.3 Weaknesses

The main weakness of this report as mentioned before is time. Some of the parameters of the grid search had to be reduced in order to produce the results in a timely fashion. This has real world applications as often results are needed quite quickly.

An improvement here could be running the mode with many parameters, or perhaps using a random parameters search, not on a local computer but on a server. This means the code would run on powerful hardware without interruption allowing for more parameters to be used without demanding as much time.

4. Conclusion

To conclude this report, a PCA of the different models is carried out - allowing us to see how the models perform differently across the metrics.

4.1 PCA results tables

Below is a table of the loading scores for the metrics:

	Accuracy	Precision	Recall	F1 Score
PC1	-0.555847	-0.337699	-0.52722	-0.546838
PC2	-0.007286	0.90922	-0.362511	-0.204576

Fig 4.1a: PC Loadings for PC1 and PC2 for each performance measure

Below is a table of the loading scores for each of the models:

	Logistic Regression	LDA	QDA	Random Forest	Gradient Boost	XGBoost	SVM	NN	Decision Tree
PC1	1.765105	1.915738	3.420752	-1.651047	-0.924015	-1.430164	-1.26186	-1.849758	0.015247
PC2	-1.445549	-1.024678	1.766692	0.145345	-0.302925	0.049933	0.500408	0.466846	-0.156073

Fig 4.1b: PC Loadings for PC1 and PC2 for each classifier

4.2 PC1 vs PC2 plot

Below is a graph of PC1 vs PC2. PC1 acconts for ~ 80% of the variance, and PC2 ~ 19%.

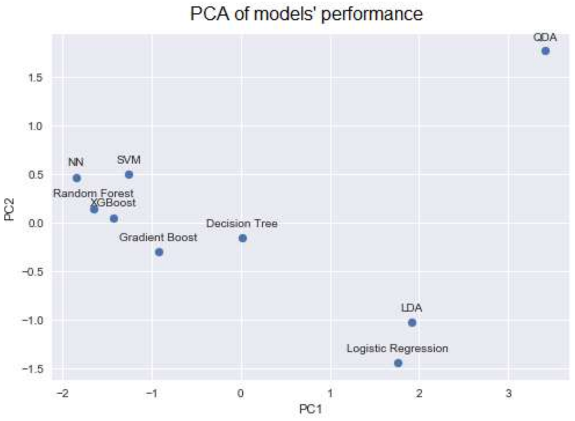


Fig 4.2: PC1 vs PC2 of classifier performance

4.3 PCA takeaways

PC1 - overall performance : High score = Poor overall performance, Low Score = Good overall performance

PC2 - precision : High score = High precision, Low score = low precision

QDA performs poorly so it has a high PC1 but with a relatively good precision so has a high PC2. QDA was a quirk in this report and for that reason it is an outlier.

We see LDA and Logistic Regressions together, scoring high on PC1 (poor performance) and negative on PC2 (poor precision). These algorithms produce a linear decision boundary and for that reason they are together in the plot.

Decision Tree performs fairly well and the best models clump around the left of the plot, with Neural Network being the left most point (Negative PC1 and neutral PC2 - good precision with good performance).

4.4 Final Conclusion

As seen from the table, Neural Network performs the best here, but SVM, Random Forest and XGBoost offer powerful alternatives.

5. Ammendum: MDS Analysis

5.1 Plot Components

Below is a pairplot of the first four components of an MDS analysis on a random sample of 100 particles (50 of each particle).

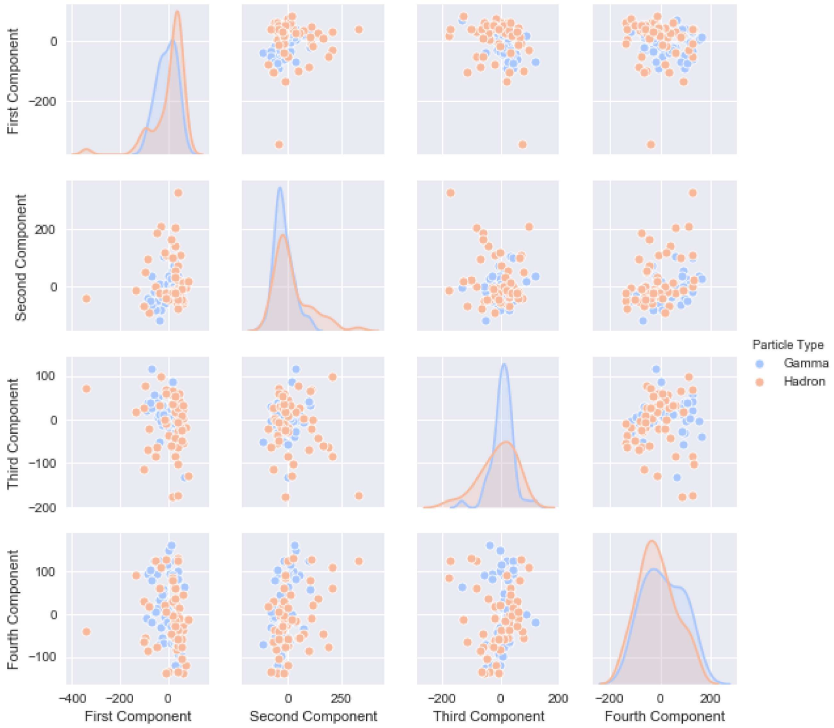


Fig 5.1a: Pair plot of four components of the MDS Analysis

5.2 The means

Below is a table of the means of each component of each particle type - this can be seen as the difference between the centroids of each particle for the component.

	Particle Type	First Component	Second Component	Third Component	Fourth Component
0	Gamma	-2.273044	-4.962544	-11.067118	-21.298796
1	Hadron	2.273044	4.962544	11.067118	21.298796

Fig 5.2b: Means of the components of each type of particle.

5.3 How many components should be chosen?

Below is a list of the explained variances of each component of the MDS analysis. There is a big drop off from the second component to the third - suggesting that only the first two components should be used.

	PC1	PC2	PC3	PC4
Explained Variance	0.411648697	0.275290011	0.138894085	0.107121026

Fig 5.3: Table of Eigenvalues/Explained Variance.

5.4 Decision

Although 2 components seems to be sufficient, eyeballing the pairplot 5.2b, each of the components plotting against the third component produces a good separation between the points. This is backed up with the large distance between the means of the