Magic Gamma Telescope

Chetali Jain

25/01/2022

—

Machine Learning

Investigation Aim

This data analysis report is to provide the best machine learning algorithm used to predict whether the showers arise from gamma particles or hadron.

Background

The data used for this analysis is assigned by Professor Jarvis and can also be extracted from the Magic Gamma Telescope dataset,
https://archive.ics.uci.edu/ml/datasets/magic+gamma+telescope

The data is used to simulate the registration of high-energy gamma particles in a ground-based atmospheric Cherenkov gamma telescope. Telescope observes the gamma rays and the radiations emitted by charged particles inside the electromagnetic showers allow reconstruction of the shower parameters. The energy produced by these gamma particles results in a few hundred to some 10000 Cherenkov photons in patterns (shower image). The goal is to distinguish statistically the showers which arise from gamma particles from those which come from hadron.

Magic Gamma Telescope dataset contains 11 variables. The first 10 variables are continuous independent variables and the last one is the dependent variable i.e., class label, g (for gamma), and h (for hadron). The variables and their units are as follows:

1. Length: the major axis length of the ellipse (mm)
2. Width: the minor axis length of the ellipse (mm)
3. Size: the (log of the) total brightness of the ellipse (photons)
4. Conc: a measure of the concentration of the brightness
5. Conc1: a measure of the maximum brightness to the size
6. Asym: a measure of how far the brightest pixel is from the center (mm)
7. M3long: a measure of the concentration along the major axis (mm)
8. M3Trans: a measure of the concentration along the minor axis (mm)
9. Alpha: the angle of the major axis to the axis of the telescope (degree)
10. Dist: the distance from the central point of the telescope to the ellipse (mm)
11. class: either g (for a gamma particle) or h (hadron).

The data contains 4000 values in which 1999 are gamma particles, 2000 are hadrons particles, and 1 missing value which was omitted during analysis. Feature importance for the prediction of particles is shown in Figure I which exhibits that the angle of the major axis to the axis of the telescope to the ellipse (Alpha) is an important variable in predicting the type of particles and the least important feature is the measure of the concentration along the minor axis (M3Trans).

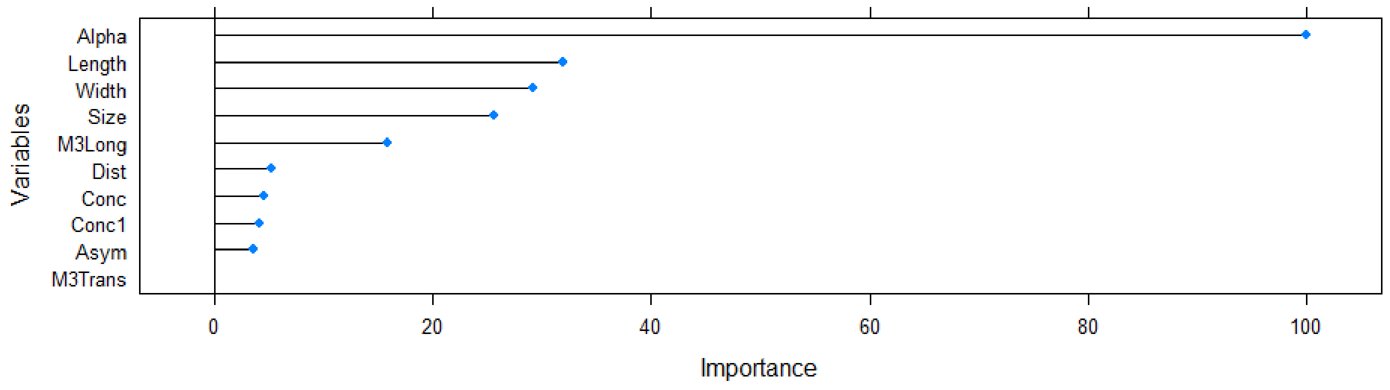

Figure 1: Variable Importance Plot

Overview

- Seed is set 123 (random number) so that the sample remains the same.
- *createDataPartition()* function in *Caret* library is used to split the data in 8:2 ratio.
- Data is cross-validated (resampling method that uses different portions of the data) 8 times and this process is repeated 5 times to avoid the problem of overfitting. This was achieved by using the *trainControl()* function from the *Caret* library.
- Each model is fit using *the train()* function in the *Caret* library. *Method* parameter was used to define the algorithm used. *preProcess* parameter is set as c("center", "scale") to standardize (i.e., mean = 0, variance = 1) the feature variables. *tuneLength* is set as 5 to optimize the parameters used in the algorithm. Models are compared using Accuracy *metrics* i.e., the greater the accuracy, the better the model.
- Gradient Boosting Model gave the maximum accuracy for both training and testing datasets and therefore it can be stated that it did best among all 12 models trained on the dataset.

Logistic Regression

Simple Logistic regression was fitted using *glm*() function by giving *family* parameter "binomial". This gave an accuracy of 77.4% for the training set and 78.6% for the testing set. Regularized Logistic regression model was also fitted using the *train*() function and the *method* was set as *"glmnet"*. The optimized accuracy was achieved when *lambda* is 0.0046 (approx.). It gave an accuracy of 78.34% for the testing dataset which doesn't vary much from simple logistic regression. An additional R package required for this model is *glmnet*.

Discriminant Analysis

Linear discriminant analysis model and Quadratic linear analysis model were trained on the data by assigning *method* "*lda*" and "*qda*" value respectively in *train*() function and also require *MASS* library. LDA gave training accuracy of 77.12% and 77.97% testing accuracy whereas QDA gave training accuracy of 74% and 75.09% testing accuracy. There were no parameters optimized in discriminant analysis.

Random Forest

Random Forest was built by typing "*rf*" in the argument, *method* of the *train*() in Caret. In a random forest algorithm, *train*() optimizes the value of the number of variables tried on each split (*mtry*). The number of trees used in the model (*ntree*) was optimized manually using *for loop*. The final model was built on 250 trees and 4 variables were tried on each split. Training set accuracy obtained from the final model is 84.79% and the testing accuracy is 84.48%. An additional R package that was required for this model is *randomForest*.

Bagging

The Bagged AdaBoost model was used which is a combination of adaptive boosting and bagging algorithms. This model was achieved by assigning the "*AdaBag*" value to the *method* parameter in the *train*() function. The ideal model used 100 trees to train the data with a maximum depth of 3 and a predicted test set with an accuracy of 78.22%. To use this method *adabag* and *plyr* libraries were also required.

Boosting

Xtreme Gradient boosting was fitted on the dataset by allocating "*xgbtree*" to *method* argument in *train*() function. While fitting the model, parameters like *learning rate (eta)*, the length of the longest path from the tree root to a leaf (*max_depth*), and *gamma* (specifies the minimum loss reduction required to make a split) were enhanced by trying different values using *expand.grid*() function. The number of trees required was finalized using *for loop*. The final model used 200 trees with a maximum depth of 3. Gamma's optimum value was achieved at 0.5 and the learning rate was set at 0.1. This model gave an accuracy of 90.94% for the training set and 86.86% for the testing set which is maximum in comparison to other models making it the best model for predicting the class of particles.
The *Confusion Matrix* for a testing set prediction made by Gradient Boosting is shown in Table 1. From the table, it can be interpreted that 357 gamma particles are correctly predicted whereas 42 particles are misinterpreted as hadron particles. Similarly, 330 hadron particles are correctly predicted and 70 are misclassified to gamma particles.

		PREDICTED	
		g	h
ACTUAL	g	357	42
	h	70	330

Table 1: Confusion Matrix: Gradient Boosting

Also, to use this method *xgboost* and *plyr* libraries were required.

Support Vector Machine

Linear, Polynomial, and Radial support vector machine algorithms were applied to the training dataset. LSVM and RSVM used *train*() function by typing "*svmLinear*" and "*svmRadial*" in *method* argument respectively. *Cost* parameter is tuned in LSVM model and for RSVM, *sigma* and *cost* parameters were optimized. The final LSVM model was built on cost = 1 giving 77.97% testing set accuracy whereas for RSVM, the final model was fitted for sigma = 0.1535 and cost = 4 giving one of the best testing sets accuracy i.e., 84.86%. The *kernlab* library was also required to fit these models.
For PSVM, the *svm*() function is used and the *kernel* is given the value "*polynomial*". Parameters like *degree*, *coefficient*, and *cost* were optimized manually using *for loop*. Final model used degree = 2, coefficient = 0.75 and cost = 1 giving 67.96% testing set accuracy.

- *train*() from *Caret* library can be used for the polynomial model as well by providing method value as "*svmpoly*", but here it wasn't used as it is time taking.

Neural Network

Multi-Layer Perceptron (MLP) model was used to train the dataset by providing method = "mlp" in train() function. MLP model was optimized with 5 nodes in the hidden layer giving 87.22% accuracy for the training set and 83.73%accuarcy for the testing set.
The Neural Network model was also trained using *Keras* which used 50 hidden layers (different values for *units* parameter were tested by using *for loop*) and the activation function used was *Rectified Linear Unit (relu)*. Output layer had 2 units and the activation function used here was "*sigmoid*".
The learning rate used in the model is 0.1 and the model is compiled to optimize *root mean square propogation (rmsprop)* and *binary_crossentropy* loss function was used for compilation. Thus, the architecture of the model used is 10 – 50 – 2 (this model is selected as it produced the highest accuracy among different architectures tried) and was fitted using 150 *epochs* giving 72.84% test accuracy.

Summary

Xtreme Gradient boosting is giving the highest accuracy for both training and testing set making it the best among all the models. It will predict the class of particles from an unknown dataset with an accuracy of 86.86 Table 2 represents the cross-tabulation of the fitted model with their associated training and testing accuracy and Figure 2 is the visual representation of the same.

MODEL	ACCURACY	
	TRAINING SET	TESTING SET
Boosting	90.94	86.86
Support Vector Machine - Radial	87.22	84.86
Neural Network (Caret)	85.88	83.73
Random Forest	84.79	84.48
Bagging	78.50	78.22
Logistic Regression (Regularization: glmnet)	77.71	78.34
Support Vector Machine - Linear	77.56	79.22
Logistic Regression	77.47	78.60
Linear Discriminant Analysis	77.12	77.97
Neural Network (Keras)	74.91	72.84
Quadratic Discriminant Analysis	74.00	75.09
Support Vector Machine - Polynomial	65.81	67.96

Table 2: Model Comparision based on Accuracy

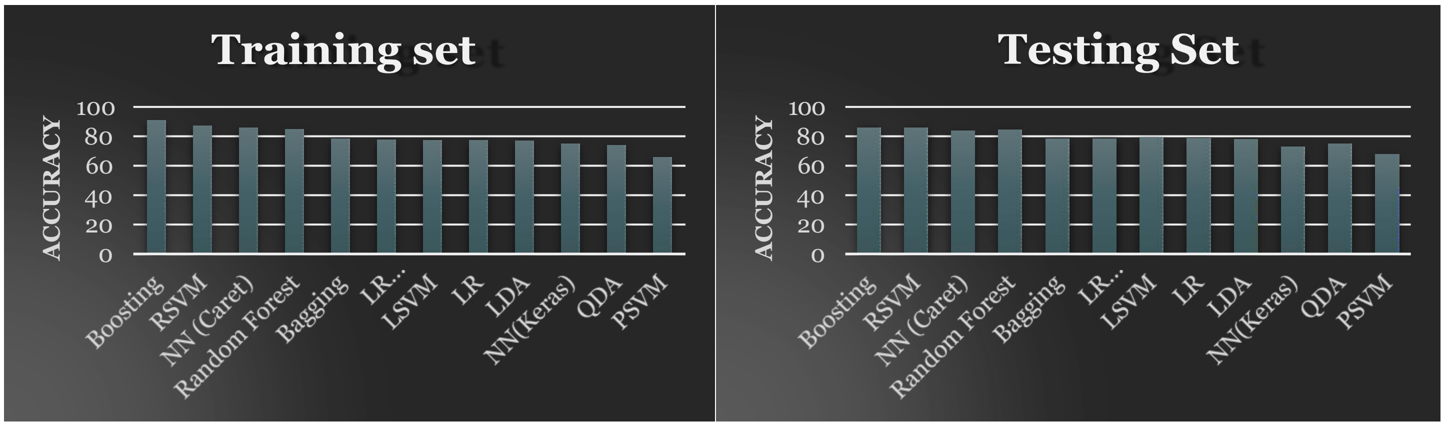

Figure 2: Training and Testing set Accuracy

References

- Jarvis F. (2021), 'MAS369/61007: Machine Learning Lecture Notes.'
- Max Kuhn (2019), 'The Caret Package'
  URL: 7 train Models By Tag | The caret Package (topepo.github.io)