Aristotle University of Thessaloniki

Physics Department

Machine Learning Course Project:

Classification of LHC recorded events as signal or background noise using Machine Learning

Anastasia Zachariadou

June 2024

Abstract

The aim of this project is to showcase different classification models utilizing machine learning. More specifically, data from events recorded in the Large Hadron Collider were used in an attempt to classify them as legitimate signal or background noise. The project is split in three parts. The first part is dedicated to the needed preparation and further manipulation of the original data. The second part is dedicated to the use of 3 distinct classifiers and the evaluation of the accuracy and efficiency of their respective prediction models. Finally, in the last part, the use of artificial neural networks is employed and examined in the same way.

Contents

Part 1

Data preparation & Set creation

In this part, the original data is prepared as needed for use in the later creation of the models. The data given to us was in the form of a .csv file of 29 columns. Each event is tied to two sets of quantities; 21 low-level quantities and 7 low-level quantities. Finally, there is a column containing the classification of the event as signal (value of 1) or background noise (value of 0).

The first step is to import the data. This is achieved using the pandas library. A minor manual modification needs to be done in column 17, as it is in string format and causes problems in following computations. It is reformatted as float.

Following that, the data is separated in three categories:

1. **classif**: It contains the values classifying an event as either signal (1) or noise (0). It corresponds to column 0.

2. **low**: It contains the values of the 21 low-level quantities for each event. These correspond to columns 1 through 21.

3. **high**: It contains the values of the 7 high-level quantities for each event. These correspond to columns 22 through 28.

This last step is vital to the training process of each model. The data is further split into 2 sets; the training set (80% of the original data) and the test set (20 % of the original data). The training set is used to train the model and the test set acts as the assessment basis for the models' accuracy. This process is performed twice, once for the low-level quantities and once for the high-level quantities. A random state is selected to ensure repeatability.

Part 2

Classifiers

The first approach taken in building the classification models is using classifiers. Three distinct classifier algorithms are tested:

1. K-nearest neighbors classifier
2. Decision trees classifier
3. Random Forest classifier

For each classifier, two models are built, one based on the low-level quantities and one based on the high-level quantities. In the end their results are evaluated and compared.

2.1 K-nn classifier

This algorithm classifies a sample (or element) based on the classification of its k nearest neighbors.

Scaling

Data scaling is a mandatory procedure before implementing the k-nn algorithm, as it involves the use of the Eucliden distance for the predictions. Appropriate fitting was employed with the help of the StandardScaler method.

Training & Results

The following process is followed for both low-level and high-level quantities. The number of neighbors k is set to 8 for both cases.

The scaled values of the training sets and their corresponding values of the classification training sets are used to train the k-nn model. Once the model produces the predictions, these are compared to the actual values of test sets. To evaluate the quality and accuracy of the model, the accuracy score was calculated along with the confusion matrix for each of the two models. These are the results:

- **Accuracy of the k-nn model based on the low-level quantities: 0.5721424**
- **Accuracy of the k-nn model based on the high-level quantities: 0.6864459**

It is clear that the accuracy of the low-level quantity based k-nn model is underwhelming, with its predictions being correct a little more than half the time (about 57%). The high-level quantity based model is noticeably better at predicting accurately the legitimacy of signals, with a percentage nearing 70%.

2.2 Decision trees classifier

This algorithm operates in a ”divide and conquer” type of logic. No scaling is needed for the use of this method.

Training & Results

Once again the same process is applied twice, once for the low-level quantity data and once for the high-level quantity for the creation of two models. After training the models in a similar way as before, the predictions are compared to the actual data and their accuracy is determined.

- **Accuracy of the decision trees model based on the low-level quantities: 0.5471580**
- **Accuracy of the k-nn model based on the high-level quantities: 0.6283573**

Similarly to the k-nn method the accuracy of the low-level quantity based k-nn model is a bit better than 50%. The high-level quantities based model produces a larger number of accurate predictions, but not to a satisfactory degree.

2.3 Random Forest classifier

The last algorithm employed is the Random Forest algorithm. It’s an ensemble method consisting of a large number of decision tree models trained simultaneously. As a result, scaling is also not needed here.

Training & Results

The number of trees that consist the forest is set as 50 for both models. Additionally, a 'criterion' needs to be selected, which is essentially a function that evaluates the quality of each split. Both for the low-level and high-level quantities based models, the function selected is 'entropy'. After the models are trained and their respective predictions are produced, the following results regarding their accuracy are yielded:

- **Accuracy of the Random Forest model based on the low-level quantities: 0.5971269**
- **Accuracy of the Random Forest model based on the high-level quantities: 0.7001874**

Once again, the high-level quantities based model is more accurate than its low-level counterpart, with the first matching the test-set predictions 70% of the time and the second having a 60% success rate.

Part 3

Artificial Neural Networks

This part is concerned with the creation of classification models using artificial neural networks. As done with each classifier in the previous part, two models are created, one based on the low-level quantities and on based on the high-level quantities. The training and test sets have already been created and scaled in previous parts of the project, so there is no need for further manipulation on that part.

ANN Building & Training

The process described in this section is applied to both low and high-level quantities.

Artificial neural networks are created layer by layer. The number of nodes in the first layer is equal to the number of quantities. The in-between number of layers are selected in arbitrarily and the output layer has a single node. As this is a binary classification problem, the activation function selected are 'relu' for the 1st and 2nd hidden layer and 'sigmoid' for the output layer. The final build is:

- Low-level quantities based ANN model:

  input layer: 21 nodes

  1st hidden layer: 25 nodes

  2nd hidden layer: 21 nodes

  output layer: 1 node

- High-level quantities based ANN model:

  input layer: 7 nodes

  1st hidden layer: 25 nodes

  2nd hidden layer: 21 nodes

  output layer: 1 node

For the compiling part, taking into account the binary classification nature of the problem, 'adam' was used as an optimizer choice, 'binary_crossentropy' as a loss function and 'binary_accuracy' metric.

The training part follows the same philosophy as the one applied in the classifiers section. The scaled values of the training set and their corresponding classification values are used to train the ANN (this is done both for the low and high-level quantities). Then, the network produces predictions regarding the classification of the signal. These predictions however here have values ranging from 0 to 1. To make a decision on whether the signal is legitimate or noise, it is assumed that values smaller than 0.5 correspond to predictions of 0 (noise) and values larger than 0.5 correspond to 1 (signal).

Results

The predictions are now ready for comparison to the actual classification data. The evaluation of the results is once again achieved by the creation of the confusion matrix and the calculation of the accuracy score. Test loss is also evaluated.

- **Accuracy of the ANN model based on the low-level quantities: 0.5915053**

  Test loss: 0.8185934

- **Accuracy of the ANN model based on the high-level quantities: 0.7001874**

  Test loss: 0.5653121

The pattern of high-level quantities based models being more accurate continues with the case of the ANN models. It is important to not here however, that in the case of the artificial neural networks running the code again would possibly yield different values of accuracy, as the model is not deterministic.

Part 4

Conclusions & Discussion

- The first conclusion that can be made based on our study is that **the models trained based on the high-level quantities are always more efficient in classifying the events**. This is true **both when using Classifiers and when training Artificial Neural networks**. At an average, **the high-level quantities based models are about** 10% **more accurate** with their predictions than their low-level counterparts.
- Out of all the **classifiers** that were tested, the **Random Forest Classifier model gives the largest number of accurate predictions**. This is the case for both low and high-level quantities based models, with the first having a $\sim 59.7\%$ accuracy and the second having a $\sim 70\%$ accuracy.
- **When comparing all the methods, the Random Forest Classifier model and the ANN are tied!** They produce the same percentage of accurate predictions with their high-level quantities based model reaching $\sim 70\%$ accuracy.

These conclusions are further backed up by the ROC graph plots and AUC (Area under curve) scores that were added to the end of our code to compare all models.

Some notes have to be made however regarding the last point. Firstly, as mentioned before, the ANN models generate different levels of accuracy every time the code is ran, even with the same parameters used as it's not a deterministic model. This doesn't affect the final result much however, as in average the percentages stay near the 60% and 70% mark respectively for the two models. Secondly, the results yielded for the two models with the highest accuracy (Random Forest and ANN) could be influenced by the way we built them. In the case of the Random Forest algorithm, if we opted for a larger number of trees it is possible that a higher accuracy could be achieved. Similarly, if a different number of nodes in the layers of the ANN was selected, or something else was changed structurally the percentage of correct predictions could be different.