ML project 2

Amanda Nathali NERIO AGUIRRE
Mojahed Sameer Moh'd ABUSHAWISH
Nikhil MOZUMDAR
Snehankit PATTNAIK
University of Caen-Normandy, France
Erasmus Mundus Joint Master Degree in Nuclear Physics
Machine learning
Prof. Morten Hjorth-Jensen

Radiation beams coming from pulsars can be observed when they point toward Earth. To discover pulsars, the identification of periodic signals is needed. A candidate is a potential signal detection. The High Time Resolution Universe (HTRU) produces some millions of candidates but a great part of them correspond to radio-frequency interference and noise. A data set of candidates produced by the HTRU was studied using two different models, XGBoost and Artificial Neural Networks (ANN). A feature correlation was performed and it was seen that Excess kurtosis and Skewnesscurve were highly correlated, 95% for integrated profile and 92% for the DM-SNR curve. The F1 scores found for XGBoost and ANN classifiers were 0.89 and 0.88, respectively for the data without dropping any features. After dropping Excess kurtosisof the integrated profile and the Skewness of the DM-SNR curve the F1 scores was 0.88 for the XGBoost and 0.87 for the ANN.

1 Introduction

Pulsars are very strongly magnetized stars with very strong dipole field that emits beams of electromagnetic radiation out of their magnetic poles. As pulsars rotate, their emission beam go in many directions, making it observable only when a beam of emission is pointing toward Earth [1]. Each pulsar generates a slightly different emission pattern that varies a bit with each rotation. Discovering pulsars requires the identification of periodic signals to reduce them into a set of candidates. A potential signal detection is known as a candidate. A candidate's signal is averaged over many rotations of the pulsar, as determined by the length of an observation. The High Time Resolution Universe (HTRU) is an all-sky survey for pulsars that produces various millions of candidates which also include radio-frequency interference (RFI) and different forms of noise. The selection of promising candidates is still a very time-consuming process that depends a lot on human inspection. This, and the generation of massive amounts of information are some reasons of why these processes are becoming not viable, thus, making an automated candidate selection a suitable solution. Supervised Machine Learning classifiers are general purpose methods that can be used to classify multiclass data according to different features. Supervised Machine Learning classifiers first build a model of these features for each data class through the training process that is run using a training data set. This process is where the classifier *learns* to label new data into different classes [2].
The data set consisted of 16,259 spurious examples caused by RFI or noise and 1,639 real pulsar examples with 8 continuous features that described those candidates, and a single class variable with two values 0 for the spurious examples and 1 for the real pulsar examples. In this work, we used two different training and evaluation models to classify the data set into pulsars candidates

and spurious candidates. The first model was XGBoost, it uses the decision trees as weak learners to build a strong learner. The second model was an Artificial Neural Network (ANN) based on a collection of connected nodes called artificial neurons, which loosely model the neurons in a biological brain. Both models of Supervised Machine Learning were evaluated.

2 Methods

2.1 ***Decision trees***

Decision trees are supervised learning algorithms used for both, classification and regression tasks. The main principle of a decision tree is to find the features that have the highest correlation with the target feature and then splits the data sets along the values of those features. This process is repeated until the subset has all the same values of the target feature [3].
Figure 1 shows a scheme of a basic decision tree. A decision tree is typically divided into a root node, the interior nodes or decision nodes, and the final leaf nodes or just leaves. These entities are then connected by so-called branches.

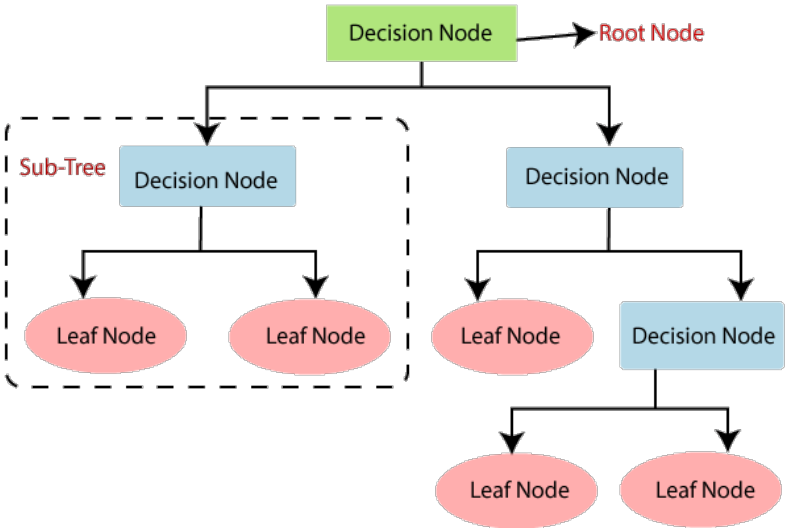


Fig. 1: Basic scheme of a decision tree.

2.2 ***Boosting***

Boosting method combines weak learners into a single strong learner. The data is first weighted according to the learners accuracy, . Then, each time a weak learner is added, the weights will be readjusted, giving a higher weight to the misclassified data, to let the future weak learners focus more on them. A typical weak learner is a decision tree with only one node [3].

2.3 ***XGBoost***

XGBoost or Extreme Gradient Boosting, is an optimized distributed gradient boosting library designed to be highly efficient, flexible and portable. It implements machine learning algorithms under the Gradient Boosting framework. XGBoost provides a parallel tree boosting that solve many data science problems in a fast and accurate way [3].

2.4 Artificial Neural Networks

Artificial Neural Networks are a computational model inspired by the biological neurons.The basic building block of the network are the neurons, the network consists of minimum a two layers, input layer and output layer. Between these two layers there can be any number of hidden layers, where each of them can contain an arbitrary number of neurons. The connection between the neurons of each layer is represented by a weight variable. In case of classification, the number of neurons in the input layer have to be equal to the number of features and the number of neurons in the output layer have to be equal to the number of categories. For a given feature vector x the network outputs an activation value y given by:

$$y = f\left(\sum_{i=1}^{n} w_i x_i + b\right) = f(u) \tag{1}$$

with b as the bias and f as the activation function [3].

3 Data set

The data set was collected during the High Time Resolution Universe Survey (South), it consists of 16,259 spurious examples caused by RFI or noise and 1,639 real pulsar examples. These examples have all been checked by human annotators.
There are 8 continuous features that describes those candidates, and a single class variable with two values 0 for the spurious examples and 1 for the real pulsar examples. The features are:

- ✓ Mean of the integrated profile.
- ✓ Standard deviation of the integrated profile.
- ✓ Excess kurtosis of the integrated profile.
- ✓ Skewness of the integrated profile.
- ✓ Mean of the DM-SNR curve.
- ✓ Standard deviation of the DM-SNR curve.
- ✓ Excess kurtosis of the DM-SNR curve.
- ✓ Skewness of the DM-SNR curve.

3.1 Features selection

The linear correlation matrix between the features and the target class was plotted to show if there was multicollinearity within the features, and also to show if some features were weakly correlated to the target class. The data set was imported using `Pandas` library and the correlation matrix was calculated using the same library. The code used is shown below.

```
#Plots the correlation matrix using Pandas
correlation_matrix = DataFrame.corr().round(2)
sns.heatmap(data=correlation_matrix, annot=True)
```

Listing 1: Plotting the correlation matrix using Pandas.

Figure 2 shows the correlation matrix, it can be seen that Excess kurtosis and Skewnesscurve were highly correlated, 95% for integrated profile and 92% for the DM-SNR curve. So we chose to drop the Excess kurtosis of the integrated profile and the Skewness of the DM-SNR curve, and then we applied our machine learning models to both data sets to see how it was affected.

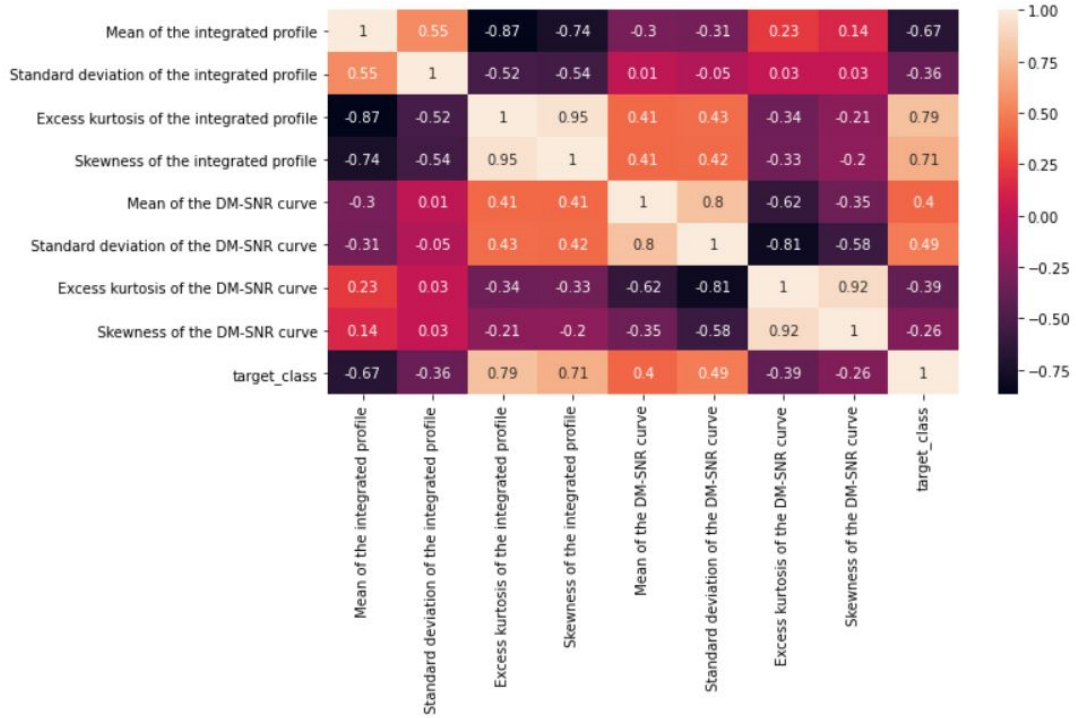


Fig. 2: Correlation matrix of the HTRU features.

3.2 ***Preprocessing of the data***

To avoid having extreme values that can affect the accuracy of the machine learning model, the data was re-scaled using the `StandardScaler` function of the `Scikit-Learn` library. This function re-scales the data to ensure that for each feature the mean is zero and the variance is one. The code used for the re-scaling of the data is shown below.

```
#Re-scale the data using StandardScaler
scaler = StandardScaler()
scaler.fit(data)
data_scaled=scaler.transform(data)

#Splits the data into test and train sets
X_train_scaled,X_test_scaled,y_train,y_test = train_test_split(data,target,
    test_size=0.2,random_state=42)
```

Listing 2: Re-scale the data and split it into train and test data sets.

The data was then split into a training data set and a testing data set, the proportion of the testing data set was 20% of the total data. Consequently, the training data set was used to build the machine learning model and the test data to evaluate the model.

4 Training and evaluation of the models

Both models were evaluated using the F1 score. The F1 score was used instead of the accuracy, due to the imbalance between the two target classes. For example if our model classified all the data as class 0 we will have an accuracy of 91% which will give false indication of the performance of the model.

4.1 ***XGBoost***

XGBoost uses the decision trees as weak learners to build a strong learner or model. To increase the performance of the model we needed to tune a number of hyper parameters. The parameters we tuned for this model were:

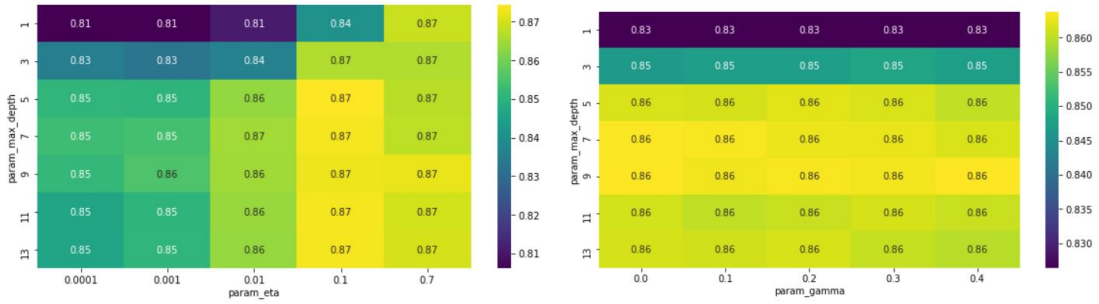
- ✓ The maximum depth of the tree (max_depth),
- ✓ The learning rate (eta) which is the step size shrinkage used in update to prevent overfitting,
- ✓ Gamma which is minimum loss reduction required to make a further partition on a leaf node of the tree. The larger gamma is, the more conservative the algorithm will be.

`GridSearchCV` function from `Scikit-learn` library was used to tune those parameters. This function fits the data using the XGBoost model with a grid of the parameters given. Then it does cross-validation resampling technique with the indicated K-fold to each point of the grid.
The code used is shown below. The parameters that were used are shown in the variable "parameters".

```
#Finds the best model with GridSearchCV
parameters = {'max_depth':[1, 3, 5, 7, 9, 11, 13], 'eta':[1e-4, 1e-3, 1e-2, 1e-1,
    0.7], 'random_state':[42], "gamma": [0.0, 0.1, 0.2, 0.3, 0.4]}
XGB=XGBClassifier()
clf = GridSearchCV(XGB, parameters,scoring='f1',verbose=4,n_jobs=-1)
clf.fit(X_train_scaled, y_train)
```

Listing 3: Search the grid of parameters to find the best model.

Figure 3a shows the F1 scores obtained from the GridSearchCV function for the parameters max_depth and eta. It can be seen that F1 scores are very similar, but there is a slight increase in F1 scores with the max_depth and eta. The F1 scores for the parameter gamma and max_depth are shown in Figure 3b, increasing the parameter gamma didn't make a significant change in the performance of the model. The best model obtained from `GridSearchCV` was with the parameters eta: 0.1, gamma: 0.1 and max_depth: 11, with a F1 score of 0.89.



(a) Heat map of the F1 score for the max depth and eta parameters.

(b) Heat map of the F1 score for the max depth and gamma parameters.

Fig. 3: Heat maps of XGBoost parameters.

After tuning the parameters, the best model obtained was evaluated using the test data set. The recall of the pulsars was 0.85 and for the spurious candidates 0.99, and the F1 score was found to be 0.89. The score obtained without dropping any features were slightly better with the recall of the pulsars 0.86 and for the spurious candidates 0.99, and the F1 score of 0.89.

4.2 ***ANN***

We built the artificial neural network using the `tensorflow` library. And as for the XGBoost we did a grid search to find the parameters that produced the best model. The main drawback of ANN is the high number of hyper parameters that needs to be tuned. The grid search was performed for four parameters: the number of neurons in each layer, the number of layers, the learning rate (eta) and lambda. An increase in the number of neurons or in the number of layers rises the computational time, so we performed the search grid for up to 6 layers and for 10, 50 and 100 neurons. And to reduce the size of the grid of parameters we did a search grid to find the values of lambda and eta that give the highest scores.
The code below shows the neural network structure, we used `relu` as the activation function and `ADAM` as the optimizer.

```
def NN_model(n_layers, n_neuron, eta, lambda, optimizer, activation):
    model=Sequential()
    for i in range(n_layers):       #Run loop to add hidden layers to the model
        if (i==0):                  #First layer requires input dimensions
            model.add(Dense(n_neuron, activation=activation, kernel_regularizer=
    regularizers.l2(lambda), input_dim=data.shape[1]))
        else:                       #Subsequent layers are capable of automatic
    shape inferencing
            model.add(Dense(n_neuron, activation=activation, kernel_regularizer=
    regularizers.l2(lambda)))
    model.add(Dense(2, activation='softmax'))  #2 outputs - ordered and disordered
    (softmax for prob)
    sgd=optimizers.SGD(lr=eta)
    model.compile(loss='categorical_crossentropy', optimizer=optimizer, metrics=['
    accuracy'])
    return model
epochs = 100
batch_size = 100

#Allows for scikit-learn to use the neural network for GridSearchCV
DNN =  KerasClassifier(build_fn=NN_model, epochs=100, batch_size=100, verbose=0)
```

Listing 4: Neural network structure.

The search grid is done as in the code below.

```
#Finds the best model with GridSearchCV
param_grid={'n_layers':[1, 2, 3, 4, 5, 6], 'n_neuron':[10, 50, 100], 'eta':[0.001,
    0.01, 0.1], 'lambda':[1e-4, 1e-3], 'optimizer': ['adam'], 'activation':['relu'
    ]}
grid = GridSearchCV(estimator=DNN, param_grid=param_grid, cv=5,scoring='f1',n_jobs
    =-1,verbose=5)
grid.fit(X_train_scaled,y_train)
```

Listing 5: Grid search of the parameters of ANN.

The F1 scores found using the GridSearchCV function are shown in Fig4. For the number of layers and number neurons, as in XGBoost the obtained F1 scores were very similar, and there was no clear dependency on the number of neurons or on the number of layers. The best model was found with the parameters eta: 0.1, lambda: 10^{-4}, number of layers: 3 and number of neurons 50 with a F1 score of 0.86.

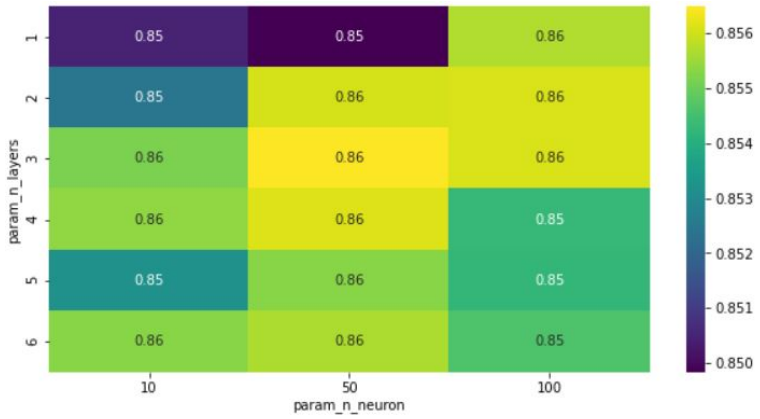


Fig. 4: Heat map of the F1 score for the number of layers and number of neurons parameters of the neural network.

The best model obtained was also evaluated using the test data set. The recall of the pulsars was found to be 0.81 and for the spurious candidates 1.0, and the F1 score 0.87. The F1 score obtained without dropping any features were slightly better with the recall of the pulsars 0.83 and for the spurious candidates 1.0, and the F1 score of 0.88.

5 Summary and discussion

We built two models of supervised machine learning to classify the pulsars from data taken during High Time Resolution Universe Survey (HTRU). The two classifiers were XGBoost which uses decision trees boosting and Artificial neural networks. The hyper parameters of the two classifiers were optimized using `GridSearchCV` function from the `Scikit-learn` library. The best models were then evaluated with the test data, the F1 scores found for XGBoost and ANN classifiers were 0.89 and 0.88 respectively for the data without dropping any features. After dropping Excess kurtosis of the integrated profile and the Skewness of the DM-SNR curve the F1 scores of the XGBoost was 0.88 and 0.87 for the ANN. In both cases the XGBoost performed better than the ANN.
The parameters of the ANN can be optimized further but the computational time it takes to fit the ANN model is significantly larger than the XGBoost, for example the execution time of the parameters grid search of the ANN was 18 minutes to do 540 fits, on average 2 seconds per fit. On the other hand, the execution time of the parameters grid search of XGBoost was 1.8 minutes to do 850 fits, 0.13 seconds on average per fit. Training and optimizing XGBoost was 15 times faster than ANN, for larger data sets the computational time becomes an issue with ANN.

References

[1] Lyne, A. G. and Graham-Smith, F., Pulsar astronomy, Cambridge University Press, Cambridge, 2012.

[2] Morello, V., Barr, E. D., Bailes, M., Flynn, C. M., Keane, E. F. and van Straten, W., Monthly Notices of the Royal Astronomical Society, **443** 1651-1662 (2014).

[3] Hjorth-Jensen, M., Overview of course material: Data Analysis and Machine Learning, (2021), GitHub repository, `https://compphysics.github.io/MLErasmus/doc/web/course.html`

Appendices

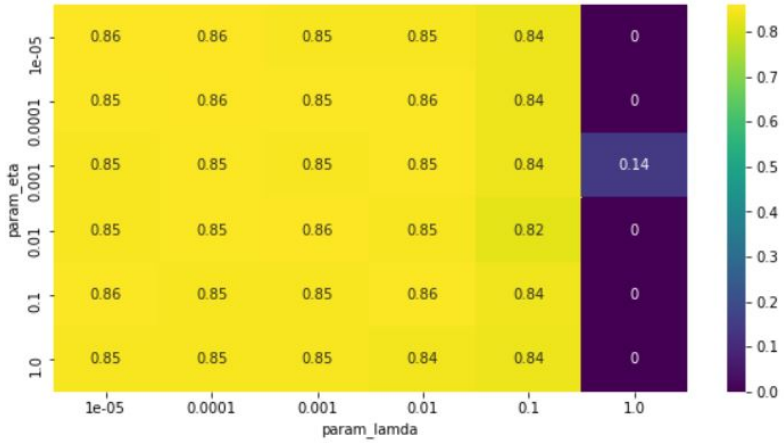


Fig. 5: Heat map of the F1 score for eta and lambda parameters for ANN to find the values with the highest scores.