Exoplanets detection using random forest

Lorenzo Loconte

Abstract

This work consists of identifying exoplanets using random forest which hyperparameters are automatically optimized with techniques that come from the `AutoML` research. The model is trained using the processed data that comes from the `NASA Kepler` project of discovering exoplanets (i.e. planets outside our solar system). The hyperparameters of the model are optimized and cross-validated with `Hyperband`, a simple yet effective and scalable method for hyperparameters optimization. The dataset used in this work can be found at **https://exoplanetarchive.ipac.caltech.edu**.

1 Introduction

Kepler Object of Interest The dataset used in this work is the cumulative list of the `Kepler Object of Interest` (`KOI`) that comes from the `NASA Kepler` project of discovering exoplanets. The dataset is composed by a list of examples regarding exoplanets. Each example have a label indicating if the corresponding exoplanet is candidate, false positive or confirmed. The provided dataset consists of a lot of heterogeneous features. For the classification task the selected features can be found in Appendix B. The candidate exoplanets (i.e. the exoplanets which existence is uncertain) are discarded in order to reduce noise and improve the learning process. For simplicity all the examples having null values are removed from the dataset. Furthermore, all the selected features are numeric and so the preprocessing method applied to the dataset is the standard normalization. The resulting dataset contains `6884` samples, which `~67%` are false positives and the remaining are confirmed.

Random Forests The model used for the classification task is a random forest, an ensemble of random decision trees. The prediction of the random forest is computed as the mode of each tree's classification. Random decision forests correct the habit of decision trees of overfitting. Each tree of the forest is trained on a subset of the features and samples. The hyperparameters of the random forest are optimized and cross-validated using approaches that come from the `AutoML` research. The following hyperparameters are optimized:

- The split criterion (Gini impurity or information gain, see Appendix A for details)
- The maximum depth of each tree
- The minimum number of samples required to split an internal node
- The minimum number of samples to be at a leaf node
- The percentage of features to use for each tree

As the next section will expose, the number of trees in the random forest is not considered an hyperparameter because it represents the budget of the hyperparameters optimization algorithm chosen. In this work we refer with budget as the computational cost of training and cross-validating a certain model.

2 Hyperparameters Tuning

The hyperparameters optimization task consists to find the hyperparameters for a model that maximize a certain score. In this work we refer to hyperparameters as the ones that cannot be trained (e.g. the ones described in the previous section). There are a lot of different techniques for hyperaparameters optimization, some of them are the following:

- Random Search
- Grid Search
- Hyperband
- Bayesian Optimization
- Hybrid Approaches (like `BOHB`)

In this work, given the restricted hyperparameters search space, we used *Hyperband* as the hyperparameters optimization algorithm. It's a simple method that consists of random sampling some points in the hyperparameters space and cross-validating them on the validation set. After that we extract a portion of the best hyperparameters and build more complex random forests. After some iterations we pick the best hyperparameters point corresponding to the best model. Even if it is a simple algorithm, it is well scalable on multiple CPUs because we assume that every point in the hyperparameters space is independent from the others. In this work the hyperparameters search space can be defined formally as:

$$\Omega = \{gini, entropy\} \times \mathbb{N}^3 \times (0, 1]$$

Hyperparameters	Values
Split Criterions	$\{gini, entropy\}$
Maximum Depth	$\{8, 9, ..., 32\}$
Minimum samples to split	$\{2, 3, ..., 16\}$
Minimum samples at a leaf	$\{1, 2, ..., 10\}$
Features percentage	$(0, 1]$

Table 1: The hyperparameters space subset used in this work.

Table 1 shows the subset of the search space used in this work.

The optimization task consists to find a point in the hyperparameters space:

$$\omega = (c, h, k, t, p) \in \Omega$$

where c is the split criterion, h is the maximum depth, k is the minimum number of samples to split, t is the minimum number of samples at a leaf, p is the features percentage for each tree, such that the random forest built from these hyperparameters maximize a certain score. The choice of this score is dependent from the task. For simplicity we use the F_1 score (i.e. we want to maximize both *Precision* and *Recall*).

3 Conclusion and Results

As you can see from Table 2 the false positives count (the number of examples, which are predicted as exoplanets, but that are not) and the false negatives count (the number of actual exoplanets not being discovered), are pretty low.

As you can see from Table 3 the auto-tuned random forest used in this work obtained a way better precision and recall. For this task we want to have a better recall because we want to maximize the number of actual exoplanets discovered. In the end we present Table 4 that shows the importances of the first five out eighteen features used.

		Actual	
		Positive	Negative
Predicted	Positive	**409**	51
	Negative	54	**863**

Table 2: Confusion matrix over the test set.

Model	Precision	Recall	F_1
k-NN	0.922	0.829	0.873
SVC	0.925	0.875	0.899
2-layer NN	0.930	0.917	0.926
Random Forest	0.938	0.932	0.938
Random Forest w/`Hyperband`	**0.944**	**0.941**	**0.943**

Table 3: Precision, recall and F_1 metrics of several models for comparison. The hyperparameters were optimized automatically. In this table it was built with the following hyperparameters found: $\omega = (entropy, 32, 7, 2, 0.5)$

Feature #	Description	Importance
7	Planetary Radius	0.196
5	Planet-Star Radius Ratio	0.118
6	Fitted Stellar Density	0.078
1	Orbital Period	0.075
8	Orbit Semi-Major Axis	0.071

Table 4: Features importances in descending order.

A Split criterions

A.1 Gini impurity

Citing `wikipedia.org`, Gini impurity is a measure of how often a randomly chosen element from the set would be incorrectly labeled if it was randomly labeled according to the distribution of labels in the subset. To compute Gini impurity for a set of items with J classes, suppose $i \in \{1, 2, ..., J\}$, and let p_i be the faction of items labeled with class i in the set.

$$I_G(p) = \sum_{i=1}^{J} p_i \sum_{k \neq i} p_k = \sum_{i=1}^{J} p_i(1 - p_i) = \sum_{i=1}^{J} p_i - \sum_{i=1}^{J} {p_i}^2 = 1 - \sum_{i=1}^{J} {p_i}^2$$

A.2 Information gain

Information gain is based on the concept of entropy and information content from information theory. For each node of the tree, the information value represents the expected amount of information that would be needed to specify whether a new instance should be classified yes or no, given the example reached that node. Given T the set of training examples, suppose $i \in \{1, 2, ..., J\}$, where J is the number of classes, and let p_i be the faction of items labeled with class i in the training set.

$$IG(T, a) = \mathrm{H}(T) - \mathrm{H}(T|a) = -\sum_{i=1}^{J} p_i \log_2 p_i - \sum_{a} p(a) \sum_{i=1}^{J} -p(i|a) \log_2 p(i|a)$$

B Exoplanet Features

1. Orbital Period [days]
2. Impact Parameter
3. Transit Duration [hrs]
4. Transit Depth [ppm]
5. Planet-Star Radius Ratio
6. Fitted Stellar Density [g/cm**3]
7. Planetary Radius [Earth radii]
8. Orbit Semi-Major Axis [AU]
9. Inclination [deg]
10. Equilibrium Temperature [K]

11. Insolation Flux [Earth flux]
12. Stellar Effective Temperature [K]
13. Stellar Surface Gravity [log10(cm/s**2)]
14. Stellar Radius [Solar radii]
15. Stellar Mass [Solar mass]
16. RA [decimal degrees]
17. Dec [decimal degrees]
18. Kepler-band [mag]