Exoplanets detection using Auto-Tuned Random Forest

Lorenzo Loconte
Knowledge Engineering Project
University of Bari Aldo Moro, Department of Informatics

Abstract

The `NASA Space Kepler` project consists of discovering exoplanets (i.e. planets outside our solar system). The data generated by the `Kepler Telescope` is analyzed by humans and algorithms to discover new exoplanets. The main problem is that a lot of exoplanets are revealed false positives. This work consists of identifying exoplanets using random forest, a *Supervised Machine Learning* model. Furthermore, the fitted model is analyzed in order to determine which features are relevant. The hyperparameters are automatically optimized with techniques that come from the `AutoML` research. In fact, the hyperparameters of the model are optimized and cross-validated with `Hyperband`, a simple yet effective and scalable method for hyperparameters optimization.

1 Introduction

Kepler Object of Interest Exoplanets are particular planets that are situated outside our solar system which orbit around one star (sometimes even multiple stars). The `NASA Kepler` project aims to discover these objects by looking at some stars and observing their luminosity during time. Basically suppose we are observing a star. If a rapid decrease and consequently increase of its luminosity occurs it's most likely that an object passed the way between us and the star. Analyzing the variation of luminosity for a certain amount of time we can extrapolate features like the radius and the orbit eccentricity of the object. Sometimes these object are actually exoplanets but it's common they can be something else like asteroids or even another star. Sometimes they are just false positives due to measurements noise.

The goal of this work is to build a model (using *Supervised Machine Learning*) that is capable to determinate if a hypothetical exoplanet can be confirmed or is just false positive. The dataset used is the cumulative list of `Kepler Object of Interest` (`KOI`) tables that come from the `NASA Kepler` project of discovering exoplanets. The complete dataset can be found at `https://exoplanetarchive.ipac.caltech.edu`. The dataset is composed by a list of hypothetical exoplanets. Each sample 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 A.

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 samples having null values are discarded from the dataset. Furthermore, all the selected features are numeric and the preprocessing method applied to the dataset is the standard normalization. The resulting dataset contains `5860` samples, of which only `~38%` are confirmed and the remaining are false positives.

Random Forests As described before, the classification task consists to determinate if some numerical characteristics can be attributed to an existing exoplanet. The model used for the classification task is a random forest, a bagging ensemble of decision trees. The prediction of the random forest is computed as the mode of the predictions made by the decision trees. Note that each tree of the forest is trained on a subset of the features. Random forests correct the habit of decision trees of overfitting on their training set. In fact, decision trees tend to overfit their training sets because they have a low bias but an high variance.

In *Supervised Machine Learning*, bias consists of erroneous assumptions about the data and variance is a measure of sensitivity to the noise in the training data. So, random forests generally obtain better performance.

The law that regulates the bias and variance is called `Bias-Variance tradeoff` [1]. This law states that models having low bias have an high variance and vice versa. In *Machine Learning* bias and variance corresponds to the models behaviors that we want to regulate in order to minimize the error. An high bias causes *underfitting* while an high variance causes *overfitting*.

$$\text{Error} = \text{Bias}^2 + \text{Variance} + \text{Irreducible Error}$$

Hyperparameters Often it's difficult (or nearly impossible) to select the correct hyperparameters that guarantee good performance on some specific datasets. `AutoML` is a branch of *Machine Learning* that try to simplify the process of model selection and hyperparameters optimization. This work focuses on solving the hyperparameters search problem for random forests. The algorithm used is `Hyperband` that, as it'll be shown in the next section, represents an evolution of random search designed especially for hyperparameters optimization.

For the hyperparameters optimization algorithm chosen, the number of trees in the random forest is not considered an hyperparameter. Furthermore, the following hyperparameters are optimized:

- The split criterion (Gini Impurity or Information Gain, see Appendix B for details)
- The fraction of features to use for each tree
- 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

2 Hyperparameters Optimization

In this section it will be shown a formalization of the hyperparameters optimization task as a search problem. In this work we refer to hyperparameters as the ones that cannot be trained (e.g. the ones described in the previous section). After that we briefly introduce the hyperparameters optimizer used in this work, `Hyperband`, as described in [2].

Given a training set T, a model L and its hyperparameters search space Ω, the hyperparameters optimization task consists to find an hyperparameters configuration $\omega \in \Omega$ such that the resulting model $L\langle\omega\rangle$ maximizes a certain score S using cross-validation on T. Note that most search algorithms don't converge to the global maximum. So, the goodness of the hyperparameters found is dependent of the amount of resources we have.

There are a lot of different techniques for hyperparameters optimization, some of them are the following:

- Random Search
- Grid Search
- Hyperband
- Bayesian Optimization
- Hybrid Approaches (like `BOHB` [3])

Of the techniques cited above only `Bayesian Optimization` guarantees the convergence to the global maximum in the hyperparameters search space (see [4] for details). The simplest algorithm is `Random Search` and it works well if the hyperparameters search space is not too big. `Random Search` is expensive because each evaluation of the score function is done on models completely built from scratch. So, each evaluation of the score function on some randomly selected hyperparameters will use the same amount of computational resources, even if some hyperparameters of the search space obtain a very low score. The idea behind `Hyperband` is that it's better to save up resources on these hyperparameters configurations that doesn't obtain a sufficiently good score by gradually building more complex models over time.

Note that we refer to budget as the computational cost of cross-validating a certain model. For example, for neural networks the budget can be the number of training epochs, while for random forests it can be the number of trees.

Algorithm 1 Hyperband algorithm for hyperparameters optimization

procedure HYPERBAND(T, L, S, Ω, R, η)
  $C \leftarrow \emptyset$
  $s_{max} \leftarrow \lfloor \log_\eta R \rfloor, \quad B \leftarrow (s_{max}+1)R$
  for $s \in \{s_{max}, s_{max}-1, ..., 0\}$ **do**
    $n \leftarrow \lceil \frac{B}{R} \frac{\eta^s}{(s+1)} \rceil, \quad r \leftarrow R\eta^{-s}$
    $H \leftarrow getRandomHyperparameters(n, \Omega)$
    for $i \in \{0, ..., s\}$ **do**
      $n_i \leftarrow \lfloor n\eta^{-i} \rfloor, \quad r_i \leftarrow r\eta^i$
      $H \leftarrow top_{\lfloor n_i/\eta \rfloor} \arg sort_\omega \{S(L\langle \omega \cup \{r_i\} \rangle, T) \mid \omega \in H\}$
    end for
    $C \leftarrow C \cup H$
  end for
  return $\arg max_\omega \{S(L\langle \omega \cup \{R\} \rangle, T) \mid \omega \in C\}$
end procedure

Algorithm 1 shows an implementation of `Hyperband` as described now. Given a budget R, `Hyperband` consists of random sampling some points (for example using the uniform distribution) in the hyperparameters space and cross-validating them using only a fraction of the budget R. After that we pick a fraction $1/\eta$ of the best models. In this way we discard the models that didn't obtain a sufficiently good score saving up computational resources.

After that, we strengthen the previous models using a larger portion of R and iterate in this way until we use the complete budget R. The procedure of consequently eliminating hyperparameters samples that don't obtain a good score is called `Successive Halving`. The fraction of the budget to use for each iteration grows geometrically in respect of η. Furthermore, we execute the discussed algorithm multiple times using less and less initial random points and using an higher starting budget. So, initially *exploration* is preferred and *exploitation* is performed later.

Even if `Hyperband` 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 our case the model L is a random forest. As said before, the budget R is the maximum number of trees that the final random forest will have. The choice of the score S is dependent from the task. For simplicity the F_1 score is used (i.e. we want to maximize both *Precision* and *Recall*). In this work the hyperparameters search space can be defined formally as:

$$\Omega = \overbrace{\{gini, entropy\}}^{\text{Split criterion}} \times \overbrace{(0, 1]}^{\text{Features fraction}} \times \overbrace{\mathbb{N}^*}^{\text{Max depth}} \times \overbrace{\mathbb{N}^*}^{\text{Min samples to split}} \times \overbrace{\mathbb{N}^*}^{\text{Min samples at leaf}}$$

For obvious reasons and for the hyperparameters optimization algorithm chosen, only a subset of this space can be explored. Table 1 shows the subset of the hyperparameters search space used in this work.

Hyperparameters	Values
Split criterions	$\{gini, entropy\}$
Features fraction	$[0.2, 0.8]$
Maximum depth	$\{8, 9, ..., 32\}$
Minimum samples to split	$\{2, 3, ..., 16\}$
Minimum samples at a leaf	$\{1, 2, ..., 8\}$

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

Model	Precision	Recall	$F_1 \downarrow$
k-NN	0.781	<u>0.948</u>	0.855
SVC	0.877	0.946	0.909
2-layer NN	0.917	0.932	0.924
Random Forest	<u>0.938</u>	0.916	<u>0.926</u>
Random Forest w/HB	**0.937**	**0.943**	**0.940**

Table 2: *Precision*, *Recall* and F_1 metrics of several models for comparison. The models hyperparameters are the default ones implemented in `sklearn`. For `Hyperband` $R = 216$ and $\eta = 3$ are used.

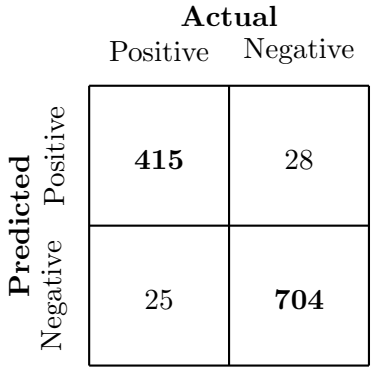

Figure 1: Confusion matrix over the test set.

3 Conclusion

In this section we analyze the results of the model chosen comparing it to other models. As you can see from Table 2 the auto-tuned random forest used in this work obtained a way better *Precision* and *Recall* respect to other models.

Furthermore, as you can see from Figure 1 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.

Models like neural networks and support vector machines are said *black-box* models. It means that predictions cannot be explained and the features importance cannot be evaluated easily. At contrary, random forests are said *white-box* models. So, predictions can be explained deeply and it's possible to evaluate the importance of each feature used.

In *Artificial Intelligence* the explainability of a model is important because it guarantees transparency. Furthermore, in *Machine Learning*, explainable models can reveal hidden knowledge about the data that isn't known at prior.

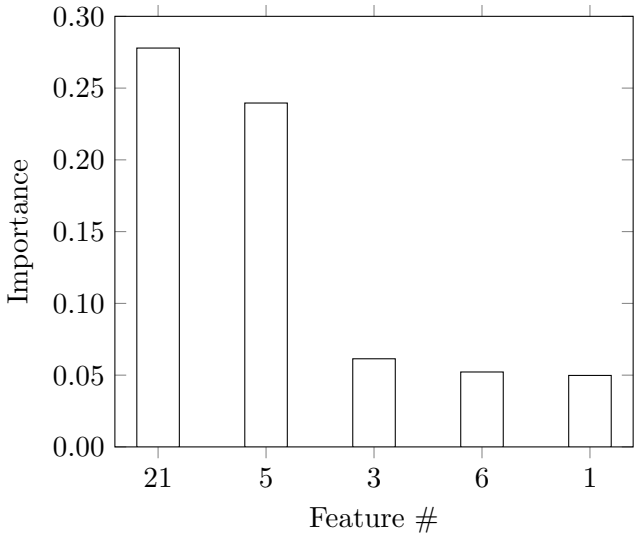

Figure 2: Five most important features importance.

In the end we present Figure 2 that shows the importance of the five most important features. In random forests the features importance are calculated by averaging the features importance in each decision tree. For a decision tree the features importance are calculated during the fit and are computed as the normalized total reduction of the criterion brought by each feature.

The most important feature is PRF $\Delta\theta_{SQ}$ (`Pixel Response Function`), that basically is the angular offset on the plane of the sky between the centroid of the star and the calculated centroid of the star during the passage of the object of interest. It's very important because a lot of false positives are due to the transit of another star in binary systems. So, if this value is big it's most likely that the object of interest is not an exoplanet. For some reason, the planetary radius represents another discriminant feature to determinate if an hypothetical exoplanet is real or a false positive. It is possible that objects that have a very small radius can be attributed to measurements noise.

In conclusion, we can say that most of the false positive occurrences are attributed to binary star systems and measurements noise.

Appendices

A Exoplanet Selected Features

The documentation of the following exoplanet selected features can be found at `https://exoplanetarchive.ipac.caltech.edu`. However most of them are auto-explicative. The features used in this work are regarding the observed star, the processed features of the object of interest and various spectrometric metrics.

1. Orbital Period $[days]$
2. Transit Duration $[hrs]$
3. Transit Depth $[ppm]$
4. Fitted Stellar Density $[g/cm^3]$
5. Planetary Radius $[Earth\,radii]$
6. Inclination $[deg]$
7. Equilibrium Temperature $[K]$
8. Insolation Flux $[Earth\,flux]$
9. Stellar Effective Temperature $[K]$
10. Stellar Surface Gravity $[\log_{10}(cm/s^2)]$
11. Stellar Metallicity $[dex]$
12. Stellar Radius $[Solar\,radii]$
13. Stellar Mass $[Solar\,mass]$
14. RA $[decimal\,degrees]$
15. Dec $[decimal\,degrees]$
16. Kepler-band $[mag]$
17. FW $\Delta\alpha$ $[sec]$
18. FW $\Delta\delta$ $[arcsec]$
19. PRF $\Delta\alpha_{SQ}$ $[arcsec]$
20. PRF $\Delta\delta_{SQ}$ $[arcsec]$
21. PRF $\Delta\theta_{SQ}$ $[arcsec]$

B Decision tree split criterions

B.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.

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

B.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.

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

References

[1] Scott Fortmann-Roe. Understanding the bias-variance tradeoff. 2012.

[2] L. Li, K. Jamieson, Giulia DeSalvo, A. Rostamizadeh, and A. Talwalkar. Hyperband: A novel bandit-based approach to hyperparameter optimization. *Journal of Machine Learning Research*, 18, 2018.

[3] Stefan Falkner, Aaron Klein, and Frank Hutter. Bohb: Robust and efficient hyperparameter optimization at scale. *ArXiv*, abs/1807.01774, 2018.

[4] Jasper Snoek, Hugo Larochelle, and Ryan P Adams. Practical bayesian optimization of machine learning algorithms. In F. Pereira, C. J. C. Burges, L. Bottou, and K. Q. Weinberger, editors, *Advances in Neural Information Processing Systems 25*, pages 2951–2959. Curran Associates, Inc., 2012.