Sofia Georgieva

Report

Model 1

Selection

The first implemented model was a Random Forest. Firstly, such a model is well-suited for datasets where the relationship between the features is complex and non-linear. Considering the dataset at hand has variety of physical features of astronomical objects, it is likely that it is indeed non-linear and complex. This assumption was also backed-up by the box plots, which showed overlapping distributions (Fig.1.) for many of the given features, which suggests that those features alone cannot linearly separate the classes.

Secondly, the random forest supports multi-class classification, which is needed in this case because in the dataset there are 4 labels (4 classes) and thus the model would be able to effectively distinguish between the classes. Additionally, single decision trees are prone to overfit, especially when they are deep and if noise is present, however random forest addresses this by aggregating the multiple generated trees. What's more, because each tree is trained on a different subset of the data and features, the model will have higher generalizability. Another advantage is that random forests deal effectively with outliers, which as presented in the boxplots (Fig.1.) were quite many, by training each tree on a random subset of the data, thus making the overall model less sensitive to outliers.

Input to classifier

After conducting the initial data exploration, the dataset consisted of 5645 samples, each representing a celestial object with 12 numeric features which describe various characteristic of each of the observations. The target variable is a categorical label indicating the type of the celestial object (a planet, a moon, a star or a dwarf planet), which are represented as numeric codes (0,1,2,3). The features and the targets are in a tabular format, where each row represents a sample, and each column represents a feature.

Next, it was important to check for missing values. They were imputed by replacing them with the mean of each feature. This would help retain the existing data distribution without introducing additional variance. Additionally, a standardization of the data was performed. This guaranteed that the features would be represented on the same scale and thus possibly increasing the generalizability of the model. The dataset was also imbalanced, with some classes having much more samples than the rest. Considering that each model has a parameter handling imbalanced datasets, balancing of the data was not performed beforehand.

Hyperparameter tunning

The hyperparameters that were set at the beginning and were not changed were the `random_state=42` and the `class_weight = "balanced"` in order to balance the dataset. The other parameters were tuned with various values. `max_depth` had the following values [1,2,4,6,8]. The idea was to test different depths of each tree to find balance between model complexity and generalizability with lower values prone to underfit and higher values to overfit.
`min_samples_split` had the following values [2,3,4]. With these values the idea was to explore

different levels of splitting control, with low number indicating less frequent splits, in order to find a balance between accuracy and generalization. `min_samples_leaf` had the values [1,2,4,6]. This examines how much details are allowed in the terminal nodes, as there are datasets that benefit from leaves with more samples in them. `max_features` had the values ['sqrt', 'log2', 0.5, 0.75]. This helps make each tree more diverse by making it consider only a subset of features at each split. The first two values are common options for this parameter, and the last two were set to investigate the impact of larger subsets of features. The hyperparameters were tuned using random search (`RandomizedSearchCV`). This was chosen because the number of combinations to be looked at was large, thus this is the more efficient way of finding the best combination of parameters.

Model training

To train the random forest model, I used 5-fold cross-validation with `RandomizedSearchCV` to select the most optimal parameters. To the final model, the best values of the parameters that were applied included `max_depth = 8`, `min_samples_split = 4`, `min_samples_leaf = 1`, `max_features = ”log2”`

Model evaluation

After performing the evaluation, the final test accuracy is 0.721 (72%) and the best cross-validation is 75%.

Model 2

Selection

The second implemented model is Neural Network (Multi-Layer Perceptrons). This model is especially useful for capturing non-linear and complex data which is the case with the present dataset. It additionally supports multi-class classification which makes it appropriate for the task at hand. Since the 12 features in the dataset represent different physical attributes, it is likely that they interact between each other, but the multiple layers in the neural network can model said interactions. Neural networks can also handle large datasets with multiple features and their flexibility in tunning the hyperparameters allows to build a model that has optimized performance. The ability to learn intricate patterns and adapt to diverse data structures makes them a powerful choice for the current multi-class classification problem.

Input to classifier

The same steps as for model 1 were performed for model 2.

Hyperparameter tunning

The hyperparameters that were set at the beginning and were not changed were the number and size of the hidden layers (`hidden_layer_sizes=(12,12)`), the `random_state=42` and the number of iterations (`max_iter=10000`). There were three parameters thar were tuned. `alpha = [0.001,1,10]` the range of the values is such that regularization can be applied with a different strength to the dataset, starting from very low regularization and ending at high regularization which would prevent overfitting if data had high variability. `activation =[‘relu’, ‘tanh’]` both activation functions deal with non-linear data, so both of the options are valid to apply for this specific dataset. `solver=[‘sgd’, ‘adam’]` both are effective optimizers for neural networks and thus both would be a good choice for this case in hand.

Model training

To train the neural network (MLP), I used 5-fold cross-validation with `RandomizedSearchCV` to select the most optimal parameters. To the final model, the best values of the parameters that were applied included `solver = “adam”`, `alpha = 1`, `activation = “tanh”`.

Model evaluation

After performing the evaluation, the final test accuracy is 0.723 (72%) and the best cross-validation is 75%.

Model comparison

Based on the results, both models perform equally well in terms of accuracy (for both the accuracy is 72%). On the same note, their cross-validation values were also the same (0.75). Based on their average training times, the random forest took 2.09 seconds to train, while the neural network took 9.64 seconds to train. Thus model 1 was faster to train. For this dataset, I would choose model 1 (Random Forest) as the better model firstly, because it is easier to interpret a random forest than a neural network. Secondly, the current configuration of the neural network did not outperform significantly the random forest, thus I would prefer model 1 because I know that it would generalize to unseen data especially with the values that were used here to tune the parameters.

Final test accuracy

The random forest has a final test accuracy of 0.721 (72.1%) which is both above chance level and it also performs better than the expert astronomer.

Final results

Based on the classification report, the easiest class to classify was class 1, which was “Moon” (F1-score = 0.91). On the other hand, the most difficult was class 0, “Planet” (F1-score = 0.69) (Fig.2.). In terms of measuring precision, class 1 (Moon) has the highest value, indicating high rate of correct positive predictions. On the other hand, class 0 (Planet) has the lowest precision, suggesting that there are more instances in which another class is being misclassified as a class 0. There was indeed some degree of misclassification of the classes based on the confusion matrix (Fig.2.). More specifically, there are 17 instances of class 0 (Planet) being misclassified with class 2 (Star), while again from class 0 there are 46 instances that are misclassified as class 3 (Dwarf Planet).

Appendix

Fig. 1. Boxplots for visualizing the dataset.

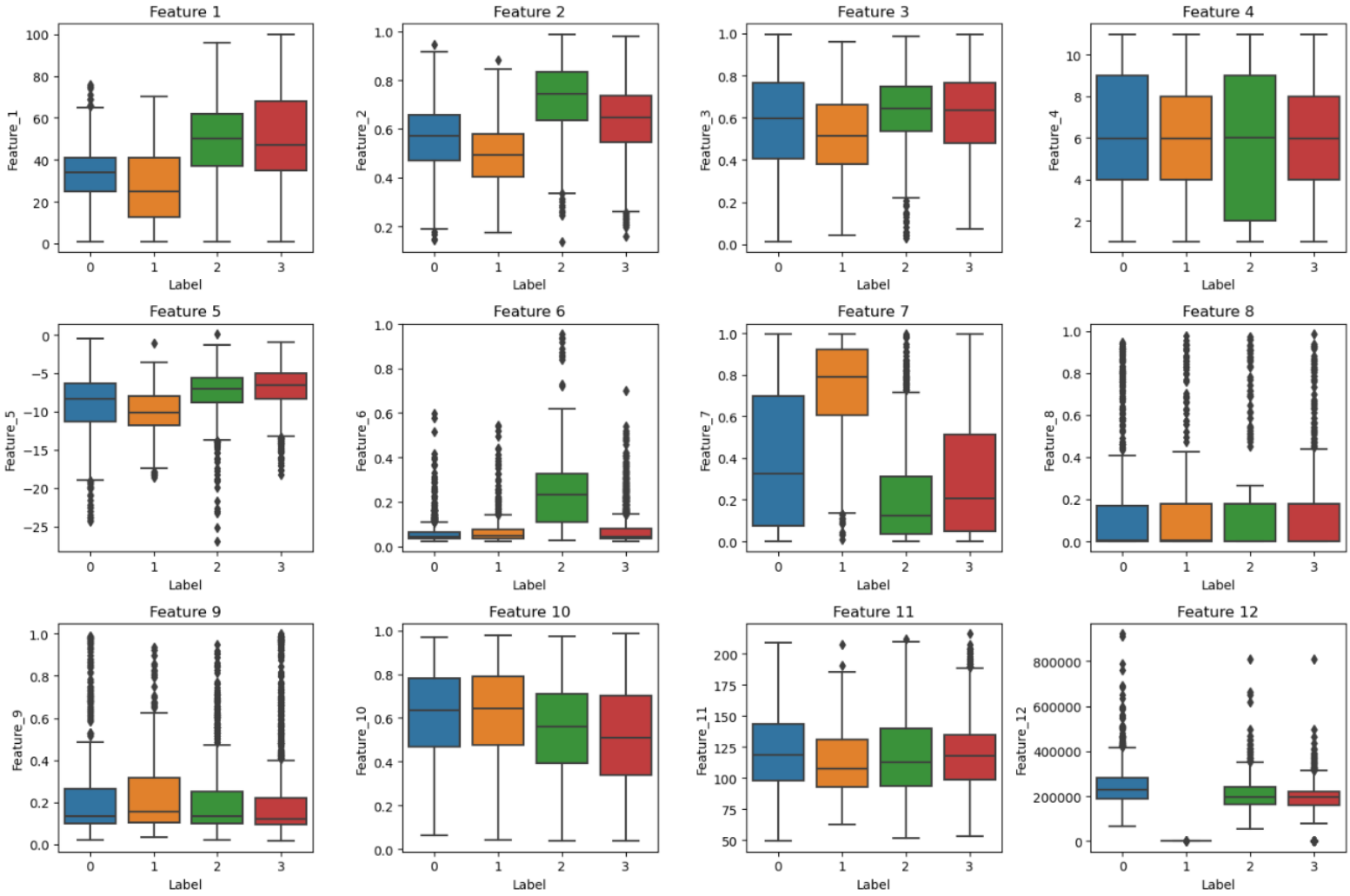

Fig. 2. Classification report and confusion matrix for random forest model.

Classification Report for Random Forest Model:

	precision	recall	f1-score	support
Class 0	0.60	0.80	0.69	319
Class 1	0.88	0.95	0.91	97
Class 2	0.73	0.68	0.71	365
Class 3	0.77	0.67	0.72	631
accuracy			0.72	1412
macro avg	0.75	0.78	0.76	1412
weighted avg	0.73	0.72	0.72	1412