Fictional Space Probe Data Classification

Marek Topolewski

Introduction

The aim of the project was to analyse data generated by two UKSA space probes that collected data on extra-terrestrial life forms. Given complete samples gathered by probe A, the task consisted of predicting missing information from probe B. To achieve that, a number of machine learning models has been analysed and tested, the most accurate of which will be discussed in this paper along with feature engineering techniques employed.

Data Pre-processing

The first stage of the project depended on thorough analysis of the data provided by probe A. After plotting the samples in tuples of the four chemical compounds in different resolutions (see Figure 1), it was observed that resolution 1 had values mostly smaller than resolution 2, which in turn had smaller values than resolution 3. Moreover, a significant number of outliers existed for each resolution. Values of these outliers were situated in the regions where other resolutions had the highest distribution. Visual analysis was followed by examining the sample data, which further showed that whenever such outlier exists in one resolution for a given compound, at least one other resolution in that sample diverges significantly from its average value. The conclusion was as follows – within the three resolutions of every compound, **resolution shuffling** occurs in some samples. Hence, to mitigate this noise, the rows for each compound (three columns of corresponding resolutions) were sorted, which greatly reduced the variance for each resolution (see Figure 2) and improved overall performance.

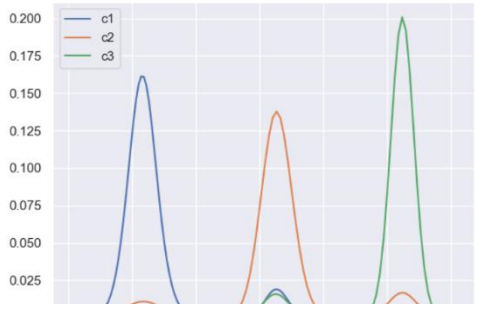

Figure 1: Distribution of the component C plotted for each resolution prior to data pre-processing.

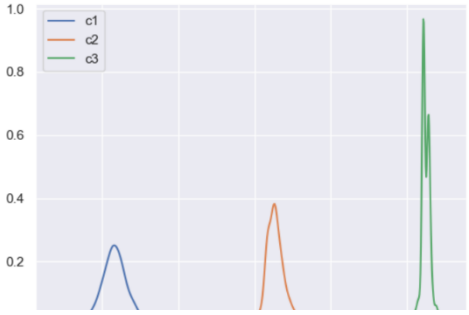

Figure 2: Distribution of the component C plotted for each resolution after data pre-processing.

Finally, features were **standardised** using *StandardScaler* object from *sklearn.preprocessing* [**1**], which centres the data and subtracts the standard deviations of each feature to reduce the variance.

Task 1

Purpose of this task was to predict missing **class information** for probe B using observations from probe A, initially without the TNA feature because it is not present in probe B. Since only two possible values of class exist, a classifier had to be selected and fitted on the pre-processed training data. Two models were considered: k-NN Classifier and Random Forest Classifier (RF). To compare them, **10-fold cross validation** was implemented, which calculated the average **Area Under Curve** (AUC) of the Receiver Operating Characteristic (ROC) of the folds. For each model **hyperparameter tuning**, as described in [**2**], was applied to find the most optimal configuration of a model for the pre-processed data. This way not only were the best settings obtained, but also overfitting was avoided thanks to *RandomisedSearchCV* object from *sklearn.model_selection*. It was deemed beneficial to use the **Random Forest** model, as it returned more accurate predictions in term of AUC and hence was selected as the primary model for this task.

To further improve its performance, **feature importance** in predicting the class were calculated and examined (see Figure 3). Initially omitted TNA proved to be highly influential – over 0.20 while the remaing values were between 0.04 and 0.11. Therefore, regression model from Task 2 was used to estimate and **append TNA** values for probe B samples, especially since the predictions where deemed highly reliable (see Task 2). This operation resulted in a 10% increase of the average AUC (see Figure 4).

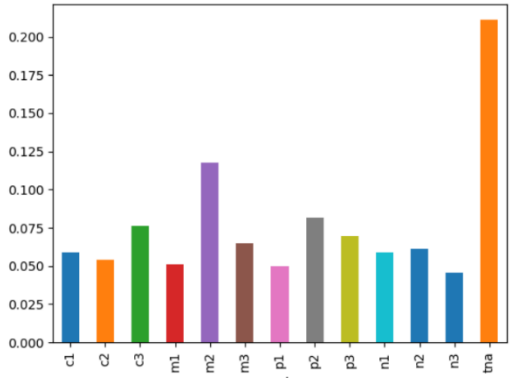

Figure 3: Feature importance from 0 (lowest) to 1 (highest) generated by RandomForestClassifier.

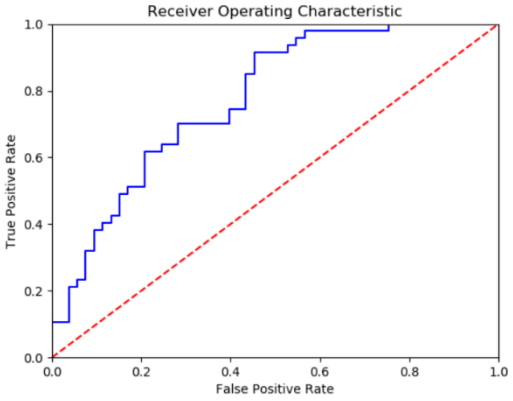

Figure 4: ROC curve representing average AUC of 0.763 obtained through 10-fold CV.

Task 2

Second task was aiming to predict missing TNA values in probe B samples. Because the value of this feature was continuous, a regression model had to be employed. The tested predictors included Ordinary Least Squares (OLS), Lasso and Ridge regressors, all of which were fitted and tested in different configurations. Data was transformed using **polynomial expansion** to allow the models to be more flexible in the sample space but also introduce a danger of overfitting. The degree of this expansion was found using hyperparameter tuning (see Task 1). Parameters, such as the alphas for Ridge and Lasso regression, were determined using *LassoCV* and *RidgeCV* implemented by *sklearn*. These objects enabled to find the most optimal settings for each model and a given degree of polynomial expansion. To avoid overfitting 10-fold CV was used again but the accuracy of the model was measured using **coefficient of determination** (R square or R2).

After testing it has been determined that OLS is the weakest model out of the three, while results Lasso and Ridge were both at least 20% more precise. This observation is also supported by the theory of these models, as both Ridge and Lasso regressors are more robust to outliers thanks to the regularisation parameter (L2 and L1 norm correspondingly) and therefore, are expected to perform better over unseen data. Combination of Lasso and Ridge regression (applying Random Forest idea) were tested, however, a higher score than plain Lasso was not achieved, leading to the selection of **Lasso** as the final model.

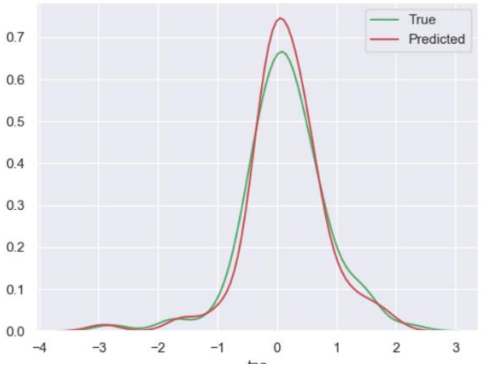

Figure 5: Distribution of true and predicted TNA using Lasso model (alpha=0.012) with R2 score 0.93 (determined by LassoCV). For comparison OLS scored only 0.61 and Ridge 0.81.

It has been observed that predictions for probe B had high variance due to difference in values for columns such as M1 and P1 compared to values in probe A. To account for this, standardisation was performed separately for each probe. This resulted in decreased performance in terms of KFold CV (10%), but hopefully better over true samples in probe B.

References

[1] "Machine Learning in Python", *Sklearn*. [Online]. http://scikit-learn.org/stable/index.html

[2] William Koehrsen, "Hyperparameter Tuning the Random Forest in Python", *Towards Data Science*, 2018. [Online]. https://towardsdatascience.com/hyperparameter-tuning-the-random-forest-in-python-using-scikit-learn-28d2aa77dd74