Writeup - sbv IMPROVER Metagenomics Diagnosis for IBD Challenge

SUBCHALLENGE: 2

Date: 29 February 2020

Table of Contents

1. Data used for training

We haven't used any additional data, only provided ones.

Depending on the 2-class problem we excluded some samples from training sets. In case IBD vs non-IBD we haven't made any changes. In case UC vs non-IBD we excluded samples which had class labels CD both from class labels files and training sets. In case CD vs non-IBD we excluded samples which had class labels UC both from class labels files and training sets. In case UC vs CD, we excluded samples which had class labels non-IBD both from class labels files and training sets. The excluded class label is not significant in that specific case. Also, we had to do that because of the binary conversion.

2. Data processing

From each file Class_labels_He.txt and Class_labels_Schirmer.txt we extracted the last column (group) and merged them. Denote it with class label.

On files:

- TrainingHe_PathwayAbundance_matrix.txt,
- TrainingHe_TaxonomyAbundance_matrix.txt,
- TrainingSchirmer_PathwayAbundance_matrix.txt,
- TrainingSchirmer_TaxonomyAbundance_matrix.txt

We applied:

1) transpose operation in a way that we get samples in rows and features in columns,
2) normalization (every row was normalized with its sum).

After these transformations we wanted to excluded some samples as mentioned above from each matrix and from class labels. To simplify this procedure, we merged the two taxonomy matrices and two pathway matrices with the corresponding class labels, did the exclusion (if necessary; depending on the case) and split them again.

In the end, class labels were encoded in the following way:

Table 1 Class labels coding

Case	Case name	Label	Encoded label
1	IBD vs nonIBD	IBD	1
		nonIBD	0
2	UC vs nonIBD	UC	1
		nonIBD	0
3	CD vs nonIBD	CD	1
		nonIBD	0
4	UC vs CD	UC	0
		CD	1

3. Approach to train classification model

Feature selection and classification models are implemented with Python (version 3.7) and estimators from *Scikit-learn* library (version 0.22.1) [1]. We sequentially execute estimators such that output from feature selection estimator is input for the classification one. Each estimator has its parameters and the most optimal ones was found with grid search [2]. All data was provided to grid search. Since it contains within itself cross-validation technique (10 folds), additional data split was not necessary.

We train and test several methods for feature selection and classification by this procedure and evaluate performance for each. The best one is submitted and reported.

Denote:

- C1 – case IBD vs nonIBD
- C2 – case UC vs nonIBD
- C3 – case CD vs nonIBD
- C4 – case UC vs CD

Code can be found on the following link: https://github.com/SafariChicken72/Challange.

3.1. Feature selection

Tested feature selection methods are based on [1], [3]. Not all tested algorithms and their parameters are reported, just the ones with best performance, which was evaluated and will be presented later.

3.1.1. Taxonomy feature selection

The best method for the first three cases is founded on univariate statistical test which uses ANOVA F-value [4] to rank features and selects the best *k* ones. After grid search we obtained optimal *k* equals 375, 500, 500 for each case respectively.

In case C4 the best method for feature selection is similar to the one above (uses ANOVA F-value [4]) but selects the best *p* percentile features. We had optimized parameter *p* through grid search and got it equals 0.9.

3.1.2. Pathways feature selection

For the all four cases the best method was the one describe in 3.1.1. Grid search yields parameter *k* equals 300, 475, 200, 300 for each case respectively.

3.2. Classification model

Trained estimators for classification are Random Forest Classifier [5] and Support Vector Classifier (SVC) [6] [7]. These estimators were last in our pipeline model and thus after their training we evaluated performance (Table 2, Table 5) and on its base we picked the best models.

3.2.1. Taxonomy classification

In this section we will present performance of chosen models and their best parameters. Our measurements for comparing and choosing the best methods were Accuracy and F1 score [8].

To get these results we had to do one simple cross validation (10 folds) and for each fold predict class labels. Further, we compare ground truth and predicted labels to get Accuracy and F1 score.

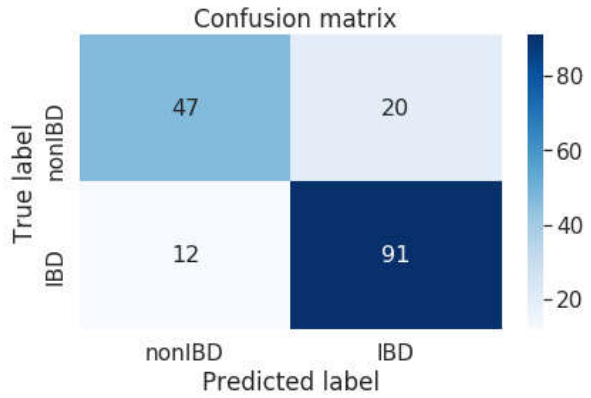
Table 2 Performance of taxonomy classifiers

Case	Models	Performance	
		Accuracy (%)	F1 score (%)
1	Select K Best and Random Forest	81.18	85.04
2	Select K Best and Random Forest	83.33	63.16
3	Select K Best and Random Forest	84.97	86.86
4	Select Percentile and SVC	84.47	91.30

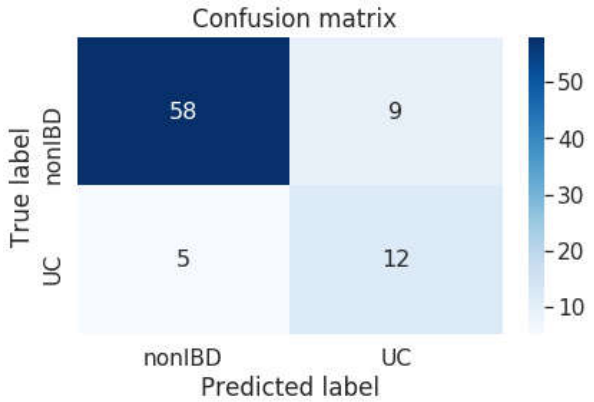
Next figure represents confusion matrices for the obtained methods.

Figure 1 Confusion matrices for taxonomy data

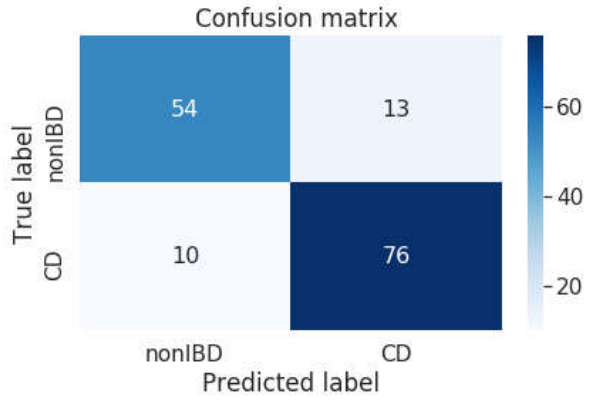
(a) C1: Methods Select K Best and Random Forest Classifier



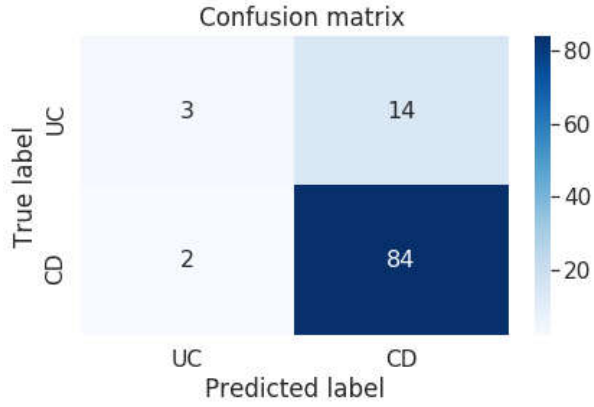
(b) C2: Methods Select K Best and Random Forest Classifier



(c) C3: Methods Select K Best and Random Forest Classifier



(d) C4: Methods Select Percentile and SVC



We got the following parameter for Random Forest Classifier as best ones:

Table 3 The best Random Forest Classifier parameters for taxonomy data

Parameter	C1	C2	C3
# of trees in the forest	200	100	200
Maximum features	*sqrt*	*sqrt*	*sqrt*
Minimum # of samples required to be at a leaf node	3	4	2
Minimum # of samples required to split an internal node	2	2	2
Bootstrap	True	True	True
Random state	42	42	42
Class weight	balanced	balanced	balanced

In C4 the best method was SVC, with used parameters:

Table 4 The best SVC parameters

Parameter	C4
C	50
Kernel	rbf[1]
Gamma	0.2
Degree	3
Class weight	balanced

3.2.2. Pathways classification

In this section we will present results for the pathway classification. Here the measurements are the same: Accuracy and F1 score.

Table 5 Performance of pathways classifiers

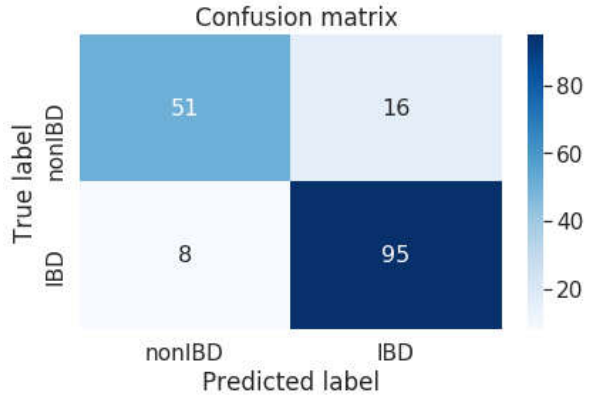
Case	Models	Performance	
		Accuracy (%)	**F1 score (%)**
1	Select K Best and Random Forest	85.88	82.35
2	Select K Best and Random Forest	92.86	63.16
3	Select K Best and Random Forest	86.27	88.00
4	Select K Best and Random Forest	83.50	90.29

Also, on Figure 2 are presented confusion matrices for Random Forest Classifier as the best method.

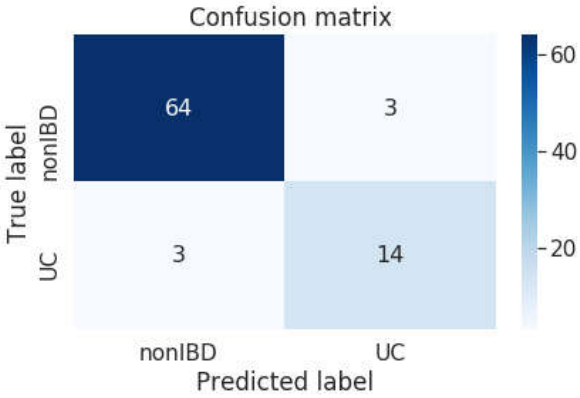
[1] Radial basis function

Figure 2 Confusion matrices for pathway data

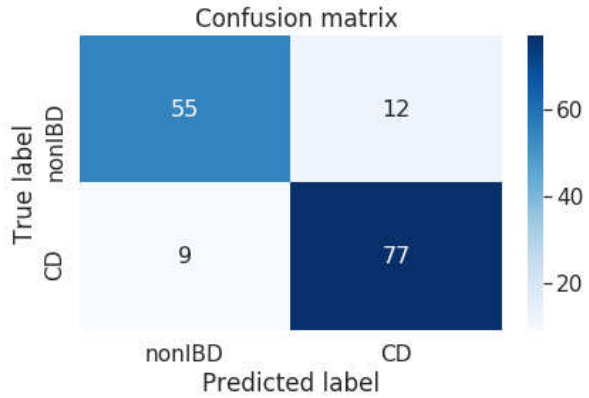
(a) C1: Methods Select K Best and Random Forest Classifier



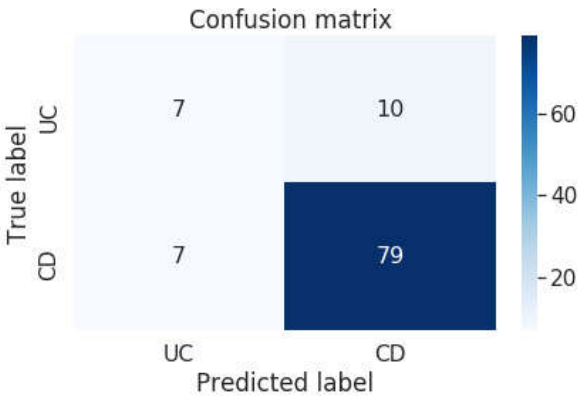
(b) C2: Methods Select K Best and Random Forest Classifier



(c) C3: Methods Select K Best and Random Forest Classifier



(d) C4: Methods Select K Best and Random Forest Classifier



In Table 6 are displayed optimal parameters which is yielded by grid search.

Table 6 The best Random Forest Classifier parameters for pathway data

Parameter	C1	C2	C3	C4
# of trees in the forest	100	100	200	150
Maximum features	*sqrt*	*sqrt*	*sqrt*	*sqrt*
Minimum # of samples required to be at a leaf node	3	2	2	2
Minimum # of samples required to split an internal node	2	2	2	2
Bootstrap	False	False	False	False
Random state	42	42	42	42
Class weight	balanced	balanced	balanced	balanced

4. Application to the testing dataset

We used in-built method provided by *Scikit-learn* library in order to predict probabilities for testing data. Methods were not trained on testing data to prevent overfitting.

5. Additional information

- The main reason that we used *Google Colaboratory* for writing the code was that we can execute our computations on GPU much faster.
- To deduce some meaningful information from data it is beneficial to have a large dataset. Initially we wanted to use Autoencoder for dimensionality reduction, but a lack of provided data forced us to choose another method described in this template.
- As a side note method SVC from *Scikit-learn* library does not have feature importance, so we submit only subset of selected features and not their importance.
- We created a new random GitHub account to provide the code in order to preserve anonymity.

6. References

[1] F. a. V. G. a. G. A. a. M. V. Pedregosa, «Scikit-learn: Machine Learning in Python,» *Journal of Machine Learning Research,* vol. 12, pp. 2825-2830, 2011.

[2] B. D. M. Marc Claesen, «Hyperparameter Search in Machine Learning,» *ArXiv,* vol. 1502.02127v2 , 2015.

[3] I. W. J. B. S. e. a. Guyon, «Gene Selection for Cancer Classification using Support Vector Machines.,» *Machine Learning,* vol. 46, pp. 389-422, 2002.

[4] Y. I. I. L. P. Saeys, «A review of feature selection techniques in bioinformatics,» *Bioinformatics,* vol. 23, n° %119, pp. 2507-2517, 2007.

[5] L. Breiman, «Random Forests,» *Machine Learning,* vol. 45, p. 5–32, 2001.

[6] J. C. Platt, «Probabilistic Outputs for Support Vector Machines and Comparisons to Regularized Likelihood Methods,» chez *ADVANCES IN LARGE MARGIN CLASSIFIERS*, MIT Press, 1999, pp. 61-74.

[7] C.-J. L. Chih-Chung Chang, «LIBSVM: A Library for Support Vector Machines,» *ACM Transactions on Intelligent Systems and Technology,* vol. 27, 2019.

[8] P. R. H. S. Christopher D. Manning, Introduction to Information Retrieval, Cambridge, England: Cambridge University Press, 2008.