Acta Scientific MEDICAL SCIENCES (ISSN: 2582-0931)

Volume 4 Issue 11 November 2020

Short Communication

BactClass: Simplifying the Use of Machine Learning in Biology and Medicine

Tian Tong Liu[1] and Maurice HT Ling[2]*

[1]Department of Information Systems and Operation Management, Warrington College of Business, University of Florida, USA

[2]HOHY PTE LTD, Singapore

***Corresponding Author:** Maurice HT Ling, HOHY PTE LTD, Singapore.

Received: September 02, 2020

Published: October 17, 2020

Abstract

Machine learning has many applications in biology and medicine. However, most existing tools require substantial programming skills, which can be a challenge to many biologists. Here, we present BactClass as a command-line tool for machine learning algorithms on formatted data, aiming to reduce the challenges faced by biologists who are interested to use machine learning approaches. BactClass is part of the Bactome project (https://github.com/mauriceling/bactome) and is licensed under GNU General Public Licence version 3 for academic and non-commercial purposes only.

Keywords: BactClass; Biology; Medicine

Introduction

Machine learning opens up many opportunities in biology [1,2] and medicine [3,4]. For example, predicting stages of cancer [5] and prognosis [6], as well as predicting disease progression from gene expression [7] and using clinical parameters to predict the presence of particular disease [8]. However, these approaches require computational tools, which have been the cause of headache for many biologists, even for those who grew up in the computer era. Hence, making such tools accessible is one of the main challenges in contemporary biology [9].

These tools can come in three forms. The first is in the form of codes, libraries, and modules; such as, Biopython [10], Biotite [11], and pySB [12]. This requires substantial knowledge in the corresponding programming language - a skill that can be challenging for traditionally laboratory trained or field trained biologists [13,14]. Yet, this provides most flexibility in use. The second form is command-line tools; such as, BLAST+ [15], PyBDA [16] and RadAA [17]. This requires a familiarity with command-line interface (CUI or CLI), which can be considered as a mini-language, and is generally less daunting full programming languages. The third form is tools with graphical user interface (GUI), which can be as standalone software or web applications. Examples are InterProScan [18], and Galaxy [19]; which are most user friendly to many biologists. Hence, there is an attempt to move CLI-based tools to GUI [20]. However, GUI-based tools can be the most limiting; for example, InterProScan [18] is limited to 40 thousand amino acids. Any larger input will require the CLI-based InterPro [21] as many GUI-based tools are powered by CLI-based tools. Indeed, GUIs such as Galaxy [19] can be built on top of many CLI-based tools [22]. Hence, majority of bioinformatics tools are command-line tools. Seemann [23] suggests 10 recommendations for developing command-line tools, highlighting its pivotal role in bioinformatics.

Currently, machine learning tools; such as, Scikit-learn [24], PyTorch [25], MXNet [26], and MLlib [27]; are mostly in the first form (codes, libraries, and modules). Here, we present BactClass as a CLI-based tool for machine learning algorithms on formatted data, bringing machine learning tools from the first form (codes, libraries, and modules) to the second form (command-line tools). BactClass is part of the Bactome project (https://github.com/mauriceling/bactome) and is licensed under GNU General Public Licence version 3 for academic and non-commercial purposes only.

Features and use cases

In this first version, BactClass is built on top of Scikit-learn [24] due to its consistency in API design [28] and is implemented as a command-line tool using Python 3 and Python-Fire module (https://github.com/google/python-fire), which aims to simplify the implementation of command-line interface in Python 3. This has been exemplified in previous tools [29-33]. The following selected supervised machine learning classifiers in were implemented in BactClass; namely, (a) Support Vector Machine (SVM; SVC class) [34], (b) Multilayer Perceptron (ANN/MLP; MLPClassifier class) [35], (c) Gaussian Naïve Bayes (GNB; GaussianNB class) [36], (d) Naïve Bayes for multivariate Bernoulli models (BNB; BernoulliNB class) [37], (e) Multinomial Naïve Bayes (MNB; MultinomialNB class) [38], (f) Complement Naïve Bayes (CNB; ComplementNB class) [38], and (g) Decision Tree (DT; DecisionTreeClassifier class) [39].

There is a pair of functions for each classifier – one for generating the classifier while the other for using the classifier. To generate a classifier, a comma-separated file of labelled data is given. A header row is required and one of the columns (denoted by column name in the header row) represents the label (the dependent variable) while the rest of the columns will be used as independent variables. The generated classifier will be saved into a file for persistence, using either joblib protocol (https://github.com/joblib/joblib) or pickle protocol (https://docs.python.org/3/library/pickle.html), which can be used for future classification tasks. A recycle function is provided to read in and write out a classifier; which can be used to update previously saved classifier to the latest serialization protocol or to change the change the type of serialization; such as, from pickle to joblib. To use a previously generated and saved classifier, a comma-separated file of independent variables is given, which is the same file format used to generate the original classifier without the label column. The predicted classification is given in the result file as "Predicted" column, together with all independent variables.

To demonstrate the use of BactClass, we attempt to generate classifiers for Wisconsin Breast Cancer data set [40]. As the data file can only contain numbers, two modifications were made to the data set – (a) removal of ID column, and (b) re-coding malignant (denoted as "M") and benign (denoted as "B") in diagnosis column as 1 and 0 respectively. The modified data was then used to generate classifiers on default parameters (see Appendix for actual command) and evaluated using 10-fold cross validation [41] on 10 evaluation metrics [42] for binary classification (Table 1); namely, (a) precision, (b) recall, (c) F1 score, (d) accuracy, (e) area under receiver operating characteristic curve (AUC), (f) Jaccard similarity, (g) normalized mutual information score (NMI), (h) V-measure [43], (i) mean square error (MSE), and (j) coefficient of determination (R-square).

Metric	ANN/MLP	BNB	CNB	DT	GNB	MNB	SVM
Precision	0.940 (0.0589)	0.000 (0.0000)	0.935 (0.0445)	0.865 (0.0451)	0.942 (0.0401)	0.940 (0.0487)	0.958 (0.0380)
Recall	0.873 (0.0517)	0.000 (0.0000)	0.770 (0.0646)	0.896 (0.0697)	0.887 (0.0602)	0.765 (0.0632)	0.920 (0.0590)
F1 Score	0.902 (0.0245)	0.000 (0.0000)	0.842 (0.0362)	0.880 (0.0409)	0.912 (0.0404)	0.841 (0.0356)	0.937 (0.0286)
Accuracy	0.932 (0.0165)	0.627 (0.0070)	0.893 (0.0228)	0.912 (0.0313)	0.937 (0.0284)	0.893 (0.0228)	0.954 (0.0195)
AUC	0.982 (0.0113)	0.518 (0.0221)	0.946 (0.0258)	0.915 (0.0339)	0.988 (0.0131)	0.946 (0.0258)	0.990 (0.0114)
Jaccard	0.828 (0.0363)	0.000 (0.0000)	0.728 (0.0548)	0.812 (0.0525)	0.841 (0.0662)	0.727 (0.0540)	0.883 (0.0506)
NMI	0.653 (0.0767)	0.000 (0.0000)	0.522 (0.0713)	0.550 (0.1002)	0.668 (0.1120)	0.527 (0.0768)	0.586 (0.0213)
V-measure	0.653 (0.0767)	0.000 (0.0000)	0.522 (0.0713)	0.574 (0.1245)	0.668 (0.1120)	0.527 (0.0768)	0.749 (0.0860)
MSE	0.069 (0.0165)	0.373 (0.0070)	0.107 (0.0228)	0.093 (0.0359)	0.063 (0.0284)	0.107 (0.0228)	0.046 (0.0195)
R-square	0.707 (0.0713)	0.594 (0.0180)	0.542 (0.0958)	0.632 (0.1316)	0.730 (0.1205)	0.542 (0.0958)	0.805 (0.0834)

Table 1: 10-fold cross validation results. The values in backets are standard deviation.

Conclusion and Future Work

We present BactClass as a command-line tool for machine learning algorithms on formatted data, reducing the need for biologists who are interested to use machine learning approaches to learn programming. As the first version, only selected supervised machine learning algorithms from Scikit-learn [24] are incorporated. Future version may add unsupervised algorithms and data preprocessing algorithms from Scikit-learn [24], and algorithms from other libraries; such as, PyTorch [25], MXNet [26], and MLlib [27].

Conflict of Interest

The authors declare no conflict of interest.

Appendix

BactClass is recommended to run on Anaconda (https://www.anaconda.com), which contains most of the required data analysis libraries on top of Python Standard Library. The following full and abbreviated (only mandatory parameters / options) commands are executed on Anaconda Prompt, after installing Anaconda. The abbreviated commands will save the generated classifier in pickle format with a file name in the following format – classifier_<type of classifier>.pickle.

- General command to generate (and save) a classifier from data: `python bactclass.py [operation] {optional parameters} --datafile=<data file name> --label=<header name of classification label>` where `[operation]` can be (a) genANN (Multilayer Perceptron), (b) genBNB (Naïve Bayes for multivariate Bernoulli models), (c) genCNB (Complement Naïve Bayes), (d) genDT (Decision Tree), (e) genGNB (Gaussian Naïve Bayes), (f) genMNB (Multinomial Naïve Bayes), or (g) genSVM (Support Vector Machine).
- Full command to generate Artificial Neural Network / Multilayer Perceptron (ANN): `python bactclass.py genANN --hidden_layer_sizes=100 --activation=relu --solver=adam --learning_rate=constant --learning_rate_init=0.001 --power_t=0.5 --max_iteration=200 --shuffle=True --tolerance=0.001 --momentum=0.9 --nesterovs_momentum=True --beta_1=0.9 --beta_2=0.999 --epsilon=1e-8 --n_iter_no_change=10 --verbose=False --classparam=True --confusion=True --classreport=True --cross_validation=10 --oclass=classifier_ANN.pickle --otype=pickle --datafile=WBC.csv --label=diagnosis`
- Abbreviated command to generate Artificial Neural Network / Multilayer Perceptron (ANN): `python bactclass.py genANN --datafile=WBC.csv --label=diagnosis`
- Full command to generate Naïve Bayes for multivariate Bernoulli models (BNB): `python bactclass.py genBNB --oclass=classifier_BNB.pickle --otype=pickle --alpha=1.0 --binarize=0.0 --fit_prior=True --classparam=True --confusion=True --classreport=True --cross_validation=10 --datafile=WBC.csv --label=diagnosis`
- Abbreviated command to generate Naïve Bayes for multivariate Bernoulli models (BNB): `python bactclass.py genBNB --datafile=WBC.csv --label=diagnosis`
- Full command to generate Complement Naïve Bayes (CNB): `python bactclass.py genCNB --oclass=classifier_BNB.pickle --otype=pickle --alpha=1.0 --fit_prior=True --classparam=True --confusion=True --classreport=True --cross_validation=10 --datafile=WBC.csv --label=diagnosis`
- Abbreviated command to generate Complement Naïve Bayes (CNB): `python bactclass.py genCNB --datafile=WBC.csv --label=diagnosis`
- Full command to generate Decision Tree: `python bactclass.py genDT --criterion=gini --splitter=best --max_depth=0 --min_samples_split=2 --min_samples_leaf=1 --min_weight_fraction_leaf=0.0 --max_leaf_nodes=0 --ccp_alpha=0.0 --classparam=True --confusion=True --classreport=True --cross_validation=10 --oclass=classifier_DT.pickle --otype=pickle --datafile=WBC.csv --label=diagnosis`
- Abbreviated command to generate Decision Tree: `python bactclass.py genDT --datafile=WBC.csv --label=diagnosis`
- Full command to generate Gaussian Naïve Bayes (GNB): `python bactclass.py genGNB --oclass=classifier_GNB.pickle --otype=pickle --var_smoothing=1e-9 --classparam=True --confusion=True --classreport=True --cross_validation=10 --datafile=WBC.csv --label=diagnosis`
- Abbreviated command to generate Gaussian Naïve Bayes (GNB): `python bactclass.py genGNB --oclass=classifier_GNB.pickle --otype=pickle --var_smoothing=1e-9 --classparam=True --confusion=True --classreport=True --cross_validation=10 --datafile=WBC.csv --label=diagnosis`
- Full command to generate Multinomial Naïve Bayes (MNB): `python bactclass.py genMNB --oclass=classifier_BNB.pickle --otype=pickle --alpha=1.0 --fit_prior=True --classparam=True --confusion=True --classreport=True --cross_validation=10 --datafile=WBC.csv --label=diagnosis`
- Abbreviated command to generate Multinomial Naïve Bayes (MNB): `python bactclass.py genMNB --datafile=WBC.csv --label=diagnosis`
- Full command to generate Support Vector Machine (SVM): `python bactclass.py genSVM --kernel=linear --degree=3 --gamma=scale --coef0=0.0 --decision_function_shape=ovr --tolerance=0.001`

```
--max_iteration=-1         --break_ties=False
--classparam=True            --confusion=True
--classreport=True     --cross_validation=10
--oclass=classifier_SVM.pickle --otype=pickle
--datafile=WBC.csv --label=diagnosis
```

- Abbreviated command to generate Support Vector Machine (SVM): `python bactclass.py genSVM --datafile=WBC.csv --label=diagnosis`
- General command to use a previously generated and saved classifier to classify new data: `python bactclass.py [operation] --classfile=<name of saved classifier> --classtype=<type of classifier file, which is either pickle or joblib> --datafile=<name of file containing data to be classified> --resultfile=<output result file name>` where `[operation]` can be (a) useANN (Multilayer Perceptron), (b) useBNB (Naïve Bayes for multivariate Bernoulli models), (c) useCNB (Complement Naïve Bayes), (d) useDT (Decision Tree), (e) useGNB (Gaussian Naïve Bayes), (f) useMNB (Multinomial Naïve Bayes), or (g) useSVM (Support Vector Machine).

Bibliography

1. Libbrecht MW and Noble WS. "Machine Learning Applications in Genetics and Genomics". *Nature Reviews Genetics* 16.6 (2015): 321-332.
2. Mahood EH., *et al*. "Machine Learning: A Powerful Tool for Gene Function Prediction in Plants". *Applied Plant Science* 8.7 (2020): e11376.
3. Ching T., *et al*. "Opportunities and Obstacles for Deep Learning in Biology and Medicine". *Journal of the Royal Society Interface* 15.141 (2018): 20170387.
4. Colmenarejo G. "Machine Learning Models to Predict Childhood and Adolescent Obesity: A Review". *Nutrients* 12.8 (2020): 2466.
5. Xie T., *et al*. "Machine Learning-Based Analysis of MR Multiparametric Radiomics for the Subtype Classification of Breast Cancer". *Frontiers in Oncology* 9 (2019): 505.
6. Lynch CM., *et al*. "Prediction of Lung Cancer Patient Survival via Supervised Machine Learning Classification Techniques". *International Journal of Medical Informatics* 108 (2017): 1-8.
7. Kegerreis B., *et al*. "Machine Learning Approaches to Predict Lupus Disease Activity from Gene Expression Data". *Scientific Report* 9 (2019): 9617.
8. Patrício M., *et al*. "Using Resistin, Glucose, Age and BMI to Predict the Presence of Breast Cancer". *BMC Cancer* 18.1 (2018): 29.
9. Morais DK., *et al*. "BTW - Bioinformatics Through Windows: An Easy-to-Install Package to Analyze Marker Gene Data". *Peer Journal* 6 (2018): e5299.
10. Cock PJA., *et al*. "Biopython: Freely Available Python Tools for Computational Molecular Biology and Bioinformatics". *Bioinformatics* 25.11 (2009): 1422-1423.
11. Kunzmann P., *et al*. "Biotite: A Unifying Open Source Computational Biology Framework in Python". *BMC Bioinformatics* 19.1 (2018): 346.
12. Lopez CF., *et al*. "Programming Biological Models in Python using PySB". *Molecular Systems Biology* 9.1 (2013): 646.
13. Carey Ma and Papin JA. "Ten Simple Rules for Biologists Learning to Program". *PLOS Computational Biology* 14.1 (2018): e1005871.
14. Auker LA and Barthelmess EL. "Teaching R in the Undergraduate Ecology Classroom: Approaches, Lessons Learned, and Recommendations". *Ecosphere* 11.4 (2020).
15. Camacho C., *et al*. "BLAST+: Architecture and Applications". *BMC Bioinformatics* 10 (2009): 421.
16. Dirmeier S., *et al*. "PyBDA: A Command Line Tool for Automated Analysis of Big Biological Data Sets". *BMC Bioinformatics* 20.1 (2019): 564.
17. Seim I., *et al*. "RadAA: A Command-line Tool for Identification of Radical Amino Acid Changes in Multiple Sequence Alignments". *Molecular Informatics* 2019 38 (1-2): e1800057.
18. Jones P., *et al*. "InterProScan 5: Genome-Scale Protein Function Classification". *Bioinformatics* 30.9 (2014): 1236-1240.
19. Afgan E., *et al*. "The Galaxy Platform for Accessible, Reproducible and Collaborative Biomedical Analyses: 2018 Update". *Nucleic Acids Research* 46 (2018): W537-544.
20. Joppich M and Zimmer R. "From Command-Line Bioinformatics to BioGUI". *Peer Journal* 7 (2019): e8111.
21. Mitchell AL., *et al*. "InterPro in 2019: Improving Coverage, Classification and Access to Protein Sequence Annotations". *Nucleic Acids Research* 47.D1 (2019): D351-360.
22. Afgan E., *et al*. "Galaxy: A Gateway to Tools in e-Science". In: Yang X, Wang L, Jie W, editors. Guide to e-Science. London: Springer London (2011): 145-77.

23. Seemann T. "Ten Recommendations for Creating Usable Bioinformatics Command Line Software". *GigaScience* 2.1 (2013): 15.

24. Pedregosa F., *et al*. "Scikit-learn: Machine learning in Python". *Journal of Machine Learning Research* 12 (2011): 2825-30.

25. Paszke A., *et al*. "PyTorch: An Imperative Style, High-Performance Deep Learning Library". In: 33rd Conference on Neural Information Processing Systems (NeurIPS 2019). Vancouver, Canada (2019): 8026-8037.

26. Chen T., *et al*. "MXNet: A Flexible and Efficient Machine Learning Library for Heterogeneous Distributed Systems". In: Neural Information Processing Systems, Workshop on Machine Learning Systems (2015).

27. Meng X., *et al*. "Mllib: Machine Learning in Apache Spark". *Journal of Machine Learning Research* 17.1 (2016): 1235-1241.

28. Buitinck L., *et al*. "API Design for Machine Learning Software: Experiences from the Scikit-learn Project". In: ECML PKDD Workshop: Languages for Data Mining and Machine Learning (2013) 108-122.

29. Ling MH. "Island: A Simple Forward Simulation Tool for Population Genetics". *Acta Scientific Computer Sciences* 1.2 (2019): 20-22.

30. Ling MH. "RANDOMSEQ: Python command–line random sequence generator". *MOJ Proteomics and Bioinformatics* 7.4 (2018): 206-208.

31. Ling MH. "SEcured REcorder BOx (SEREBO) based on blockchain technology for immutable data management and notarization". *MOJ Proteomics and Bioinformatics* 7.6 (2018): 169-174.

32. Ling MH. "Draft Implementation of a Method to Secure Data by File Fragmentation". *Acta Scientific Computer Sciences* 1.2 (2019): 10-13.

33. Ling MHT. "SeqProperties: A Python Command-Line Tool for Basic Sequence Analysis". *Acta Scientific Microbiology* 3.6 (2020): 103-106.

34. Cortes C and Vapnik V. "Support-Vector Networks". *Machine Learning* 20.3 (1995): 273-297.

35. Popescu M-C., *et al*. "Multilayer Perceptron and Neural Networks". *WSEAS Transactions on Circuits and Systems* 8.7 (2009): 579-88.

36. Gayathri B and Sumathi C. "An Automated Technique Using Gaussian Naïve Bayes Classifier to Classify Breast Cancer". *International Journal of Computer Applications* 148.6 (2016): 16-21.

37. McCallum A and Nigam K. "A Comparison of Event Models for Naive Bayes Text Classification". In: AAAI-98 Workshop on Learning for Text Categorization (1998): 41-48.

38. Rennie JD., *et al*. "Tackling the Poor Assumptions of Naive Bayes Text Classifiers". In: 20th international conference on machine learning (ICML-03) (2003): 616-23.

39. Breiman L., *et al*. "Classification and Regression Trees". CRC Press (1984).

40. Wolberg WH., *et al*. "Machine Learning Techniques to Diagnose Breast Cancer from Image-Processed Nuclear Features of Fine Needle Aspirates". *Cancer Letter* 77.2-3 (1994): 163-171.

41. Koul A., *et al*. "Cross-Validation Approaches for Replicability in Psychology". *Frontiers in Psychology* 9 (2018): 1117.

42. Powers DM. "Evaluation: From Precision, Recall and F-measure to ROC, Informedness, Markedness and Correlation". *Journal of Machine Learning Technologies* 2.1 (2011): 37-63.

43. Rosenberg A., *et al*. "A Conditional Entropy-Based External Cluster Evaluation Measure". In: 2007 Joint Conference on Empirical Methods in Natural Language Processing and Computational Natural Language Learning (EMNLP-CoNLL) (2007): 410-420.