Software Impacts 13 (2022) 100383

Contents lists available at ScienceDirect

Software Impacts

journal homepage: www.journals.elsevier.com/software-impacts

Original software publication

A machine learning software tool for multiclass classification

Shangzhou Wang [a], Haohui Lu [a], Arif Khan [a], Farshid Hajati [b], Matloob Khushi [c,d], Shahadat Uddin [a,*]

[a] School of Project Management, Faculty of Engineering, The University of Sydney, Level 2, 21 Ross Street, Forest Lodge, NSW 2037, Australia
[b] College of Engineering and Science, Victoria University Sydney, 160 Sussex Street, Sydney, NSW 2000, Australia
[c] University of Suffolk, Ipswich, UK
[d] School of Computer Science, The University of Sydney, Australia

ARTICLE INFO

Keywords:
Disease comorbidity
Disease multimorbidity
Machine learning
Multiclass classification

ABSTRACT

This paper describes code for a published article that can assist researchers with multiclass classification problems and analyse the performances of various machine learning models. Further, feature importance, feature correlation, variable clustering, confusion matrix and kernel density estimation were also implemented. The original study was published in Expert Systems with Applications, and this paper explains the code and workflow. Administrative healthcare data has been used as an example to run the code. The results and insights can assist healthcare stakeholders and policymakers reduce the negative impact of illness comorbidity and multimorbidity.

Code metadata

Current code version	V1.0
Permanent link to code/repository used for this code version	https://github.com/SoftwareImpacts/SIMPAC-2022-108
Permanent link to reproducible capsule	https://codeocean.com/capsule/3697326/tree/v1
Legal code license	MIT License
Code versioning system used	
Software code languages, tools and services used	Python
Compilation requirements, operating environments and dependencies	Pandas, Keras, scikit-learn, TensorFlow, NumPy, matplotlib, seaborn, XGboost
If available, link to developer documentation/manual	
Support email for questions	shahadat.uddin@sydney.edu.au

1. Introduction

Data has fundamentally altered how people live and conduct business in the 21st century. Data have been used in many sectors to address practical problems, including predicting disease in healthcare [1–3], assisting policymakers in making decisions [4], and predicting corporate bankruptcy [5]. Chronic diseases are becoming more prevalent throughout the world, various disease burdens are rising, and the social and economic consequences will have an impact on people's quality of life. In many circumstances, the existence of one chronic disease leads to the development of one or more other chronic disorders, which significantly strains global healthcare systems.

This paper describes the models developed by Uddin et al. [6] and how these models can be applied to new datasets related to disease comorbidity or multimorbidity. Disease comorbidity is described as the presence of many diseases simultaneously. An individual who has more than two comorbidities is referred to as multimorbid. Obtaining good quality prediction performance is a key and critical factor when determining whether a patient will be diagnosed with comorbidity or multimorbidity. However, we intend to delve into deeper detail in our

The code (and data) in this article has been certified as Reproducible by Code Ocean: (https://codeocean.com/). More information on the Reproducibility Badge Initiative is available at https://www.elsevier.com/physical-sciences-and-engineering/computer-science/journals.

* Corresponding author.
E-mail addresses: steven.wang1@iqvia.com (S. Wang), haohui.lu@sydney.edu.au (H. Lu), arif.khan@sydney.edu.au (A. Khan), farshid.hajati@vu.edu.au (F. Hajati), matloob.khushi@sydney.edu.au (M. Khushi), shahadat.uddin@sydney.edu.au (S. Uddin).

https://doi.org/10.1016/j.simpa.2022.100383
Received 26 June 2022; Received in revised form 12 July 2022; Accepted 13 July 2022

code on the interpretation of the machine learning (ML) model findings since it gives additional insight into how practitioners may better treat patients and learn about important features.

The code was created in Python and Jupyter notebook [7] and can be used as a template for future ML applications. In the Code Ocean capsule, we included the Jupyter notebook and a Python script so that others could reproduce the same results as in the original research. There are two parts to the software. The first part implements the five ML models (Logistic regression (LR), *k*-nearest neighbours (KNN), Naïve Bayes (NB), Random forest (RF) and eXtreme Gradient Boosting (XGBoost)). Two deep learning models (Multilayer perceptrons (MLP) and Convolutional neural networks (CNN)) were implemented in the second part. The performance of these models are compared with accuracy, precision, recall and F1-score. Afterwards, feature importance, feature correlation, variable clustering, confusion matrix and kernel density estimation (KDE) for the best-performed model are explored.

2. Functionalities

Firstly, this software tool loads all the Python libraries and modules used by this ML application. Three functions (i.e., index_to_label, label_to_index and plot_confusion_matrix) were defined at the beginning of the notebook to accomplish frequently repeated tasks. These functions can be easily modified or reused for any other multiple classification problems. While this study aimed to solve a multiclass classification problem, the code can be generalised to any multiclass classification process, including binary classification. Any dataset can be used in this software if it is conformed to these three requirements for each row: (a) has an identifier (or otherwise use the row index number); (b) has a number of non-nullable features; and (c) has a label.

For any considered ML model, this software tool utilises the Scikit-learn pipeline constructor to encapsulate a sequence transformation and ML steps. Then, Scikit-learn's GridSearchCV function is used to find the best hyperparameter value combination from a predefined parameter dictionary for each model. Each model will then employ the best hyperparameter value combination to perform the model evaluation for training and testing the dataset.

After the best model is selected, in our case, it is XGBoost, a confusion matrix is generated to visually evaluate the model performance. The further feature selection process is conducted based on feature importance score and feature clustering. With the help of a dendrogram hierarchical cluster chart and pairwise feature correlation map, a subset of the most important features is selected for the final model training. Our study shows that, after eliminating those collineated features, the model performance has slightly improved. Finally, a new feature importance score is evaluated on the selected feature subset, and kernel density estimation (KDE) plots are generated for the top 4 features to demonstrate the distribution dissimilarities across the labelled classes. The software flow chart for the current version is presented in Fig. 1.

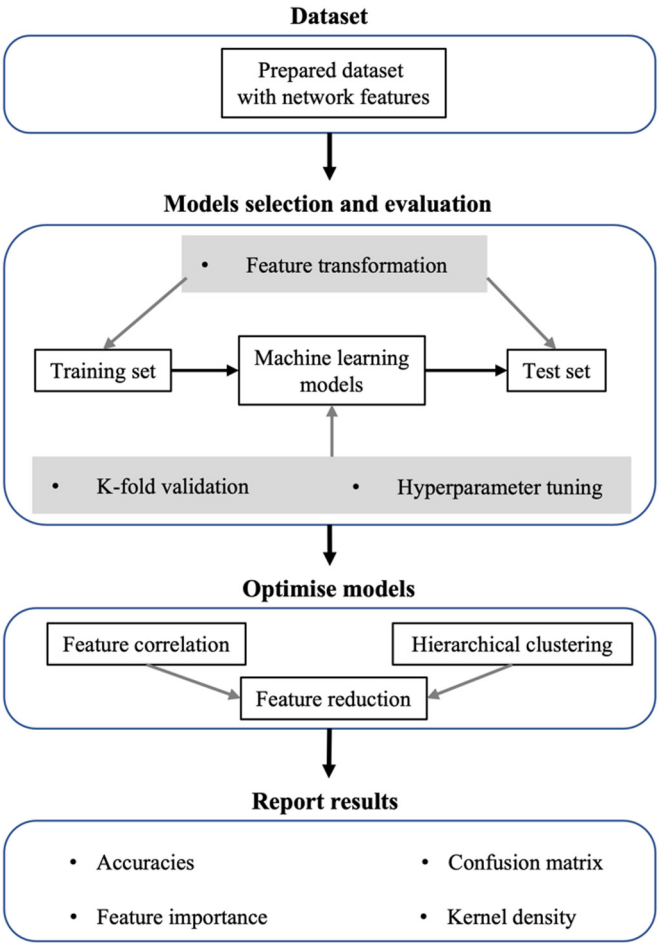

Fig. 1. Software workflow.

2.1. Data processing

Since the dataset used in this software has been shaped into a tidy format, any data tidying, reshaping and/or missing data treatment etc., are not part of this application. The main purpose of data processing is to split the dataset into training and testing sets. The Scikit-learn's train_test_split function is used for data splitting. Before we utilise this function, it is necessary to make sure that the unique row identifier should not be included in the feature or label dataset. The proportion of test data size was set at 20 percent, which can be altered to a different fraction if needed.

2.2. Feature transformation

A *StandardScaler* transformation is applied to all features for all ML models. ML algorithms do not perform well when numeric features have very different scales [8]. Standardisation subtracts off the mean value of a feature and then divides it by the standard deviation. Therefore, a standardised feature always has 0 as mean and 1 as standard deviation.

2.3. Principal component analysis

Principal component analysis (PCA) is the best-known unsupervised dimensionality reduction technique [9]. PCA uses an orthogonal transformation to convert a set of possible multicollinear features into a set of linearly uncorrelated variables and reduces dimensionality while retaining most of the information [10].

2.4. Model process pipeline

The pipeline is to assemble several steps that can be cross-validated together while setting different parameters [11]. Our code utilised Scikit-learn's pipeline to streamline a number of routine and repeated processes by encapsulating a sequence of steps involved in the feature transformation and model training. The pipeline is a logical, convenient and efficient way to manage and automate a ML workflow [12].

2.5. Hyperparameter tuning and model performance evaluation

Hyperparameters are specific to a ML algorithm, and they are not from the training data. Hyperparameter tuning is an essential part of any ML process, and it is acknowledged and utilised to achieve better model performance [13,14]. Two commonly used hyperparameter

tuning techniques are grid search and random search. Grid search is a brute-force tuning method. It evaluates the Cartesian product of a pre-supplied finite set of values for each hyperparameter. In contrast, the random search randomly chooses values or sample value combinations from a predefined value range for each hyperparameter. Grid search will find the best combination of hyperparameters from the supplied value set. The random search seeks the best combination of hyperparameters at random until a certain budget for the search is exhausted [15]. In this study, we used grid search with 5-fold cross-validation to find the best hyperparameter value combinations. A grid search performance evaluation needs a scoring metric. We used the area under the receiver operating characteristic (ROC) curve (AUC) as a performance measure in our research.

After the grid search found the best hyperparameter value combination for each model, the value set is used to fit the training dataset with 5-fold cross-validation. The overall average accuracy value and standard deviation are computed to assess each model's training performance. The performance of the testing dataset was evaluated through accuracy, precision, recall, and F1-score. Accuracy is the total number of correctly predicted instances divided by the total instances of the testing dataset. Precision refers to the number of true-positive divided by the number of all positive predictions for the class. Recall is the ratio of correctly predicted true instances out of the total instances of a label. F1-score is the harmonic mean of precision and recall. A model with the best testing accuracy and F1 score is selected as the best performing model in this study, which is XGBoost. A confusion matrix is a much nicer way to visualise model performance. While it has no single code to plot a confusion matrix chart, a function is created in our code at the beginning for an easier confusion matrix plot. As a rule of thumb, a row in the confusion matrix represents the actual value, and a column represents the predicted value. A binary class confusion matrix can be represented by four values: true-positive, false-positive, false-negative and true-negative. A multiclass confusion matrix is much more complex than binary class one. However, the general principle to interpret a multiple class confusion matrix is the same as the binary one, where the diagonal box in a confusion matrix is always the correctly classified one. For any label, while the diagonal box is true-positive, from the row-wise, all other boxes add up together is the false-negative value. From the column-wise, all other boxes add up together is the false-positive value.

2.6. Feature importance and feature selection

Feature engineering is an important part of the ML process. The more features a model has, the more likely they could be collineated and the more complex the model could be. Eliminating the weak or collineated features could reduce the training overhead and improve the common overfitting issue. A common approach to dropping weak features is to check the feature importance score. Many models provide an intrinsic function to obtain the underlying feature importance value, for example, RF and XGBoost. Since our final best-performing model is XGBoost, we used XGBoost's plot_importance function to visualise the ten top important features. One drawback of a feature importance score is that it does not inform the presence of the correlation between a pair of features. A pairwise feature correlation map is a great way to visualise the feature collinearity and subjectively select one feature from the correlated ones. In our study, we first performed hierarchical clustering on the Spearman correlation to find the feature cluster and then used the distance criteria to keep the most important feature from each cluster.

2.7. Kernel density estimation (KDE)

KDE is a non-parametric way to estimate the probability density function of a random variable. KDE is a technique for enabling a user to better analyse the studied probability distribution than when using a traditional histogram. Unlike the histogram, the kernel technique uses all sample points' locations and more convincingly suggests multimodality [16]. In our study, we presented the corresponding KDE analysis for our model's four most important features.

3. Impact

This study utilised and combined a set of commonly exercised ML process techniques to perform feature scaling, model processing, hyperparameter tuning, model selection, performance evaluation and feature evaluation and reduction. The code presented in this notebook can be reused or served as a template to tackle any classification problems. This could save some researchers a lot of time searching through different code sources to find a similar code. It is noteworthy that the novel approach used in this research to combine the augment network features and patient medical record features has yielded relatively high prediction accuracy. A software tool can be further developed for other purposes based on our study's design, analysis, and findings. Such software tools could have a long-run impact on preventing chronic disease comorbidity progression and save spending on healthcare costs.

4. Future improvements

While this application focuses on the multiclass classification to predict if a patient currently has 0, 1 or 2 and more comorbidity and multimorbidity based on their existing medical records, it will be very interesting if we can introduce the time dimension to turn this classification into a temporal classification. We will also continue to streamline our code to make it much easier to reuse.

CRediT authorship contribution statement

Shangzhou Wang: Data analysis, Data interpretation, Writing. **Haohui Lu:** Data analysis, Critical review, Writing. **Arif Khan:** Writing, Critical review. **Farshid Hajati:** Writing, Critical review. **Matloob Khushi:** Writing, Critical review. **Shahadat Uddin:** Conception, study design, Data analysis, Writing.

Declaration of competing interest

The authors declare that they have no known competing financial interests or personal relationships that could have appeared to influence the work reported in this paper.

References

[1] M.E. Hossain, S. Uddin, A. Khan, Network analytics and machine learning for predictive risk modelling of cardiovascular disease in patients with type 2 diabetes, Expert Syst. Appl. 164 (2021) 113918.
[2] A. Khan, S. Uddin, U. Srinivasan, Chronic disease prediction using administrative data and graph theory: The case of type 2 diabetes, Expert Syst. Appl. 136 (2019) 230–241.
[3] H. Lu, S. Uddin, A weighted patient network-based framework for predicting chronic diseases using graph neural networks, Sci. Rep. 11 (1) (2021) 1–12.
[4] N. Elgendy, A. Elragal, Big data analytics in support of the decision making process, Procedia Comput. Sci. 100 (2016) 1071–1084.
[5] T.M. Alam, K. Shaukat, M. Mushtaq, Y. Ali, M. Khushi, S. Luo, A. Wahab, Corporate bankruptcy prediction: An approach towards better corporate world, Comput. J. 64 (11) (2021) 1731–1746.
[6] S. Uddin, S. Wang, H. Lu, A. Khan, F. Hajati, M. Khushi, Comorbidity and multimorbidity prediction of major chronic diseases using machine learning and network analytics, Expert Syst. Appl. (2022) 117761.
[7] T. Kluyver, B. Ragan-Kelley, F. Pérez, B.E. Granger, M. Bussonnier, J. Frederic, K. Kelley, J.B. Hamrick, J. Grout, S. Corlay, Jupyter Notebooks-a Publishing Format for Reproducible Computational Workflows, Vol. 2016, 2016.
[8] A. Géron, Hands-on Machine Learning with Scikit-Learn, Keras, and TensorFlow: Concepts, Tools, and Techniques To Build Intelligent Systems, O'Reilly Media, Inc, 2019.
[9] K. Mao, Identifying critical variables of principal components for unsupervised feature selection, IEEE Trans. Syst. Man Cybern. B 35 (2) (2005) 339–344.
[10] S. Karamizadeh, S.M. Abdullah, A.A. Manaf, M. Zamani, A. Hooman, An overview of principal component analysis, J. Signal Inf. Process. 4 (2020).

[11] F. Pedregosa, G. Varoquaux, A. Gramfort, V. Michel, B. Thirion, O. Grisel, M. Blondel, P. Prettenhofer, R. Weiss, V. Dubourg, Scikit-learn: Machine learning in Python, J. Mach. Learn. Res. 12 (2011) 2825–2830.

[12] L. Buitinck, G. Louppe, M. Blondel, F. Pedregosa, A. Mueller, O. Grisel, V. Niculae, P. Prettenhofer, A. Gramfort, J. Grobler, API design for machine learning software: experiences from the scikit-learn project, 2013, arXiv preprint arXiv:1309.0238.

[13] R.G. Mantovani, T. Horváth, R. Cerri, J. Vanschoren, A.C.P.L.F. de Carvalho, Hyper-parameter tuning of a decision tree induction algorithm, in: 2016 5th Brazilian Conference on Intelligent Systems, BRACIS, 2016.

[14] L. Yang, A. Shami, On hyperparameter optimization of machine learning algorithms: Theory and practice, Neurocomputing 415 (2020) 295–316.

[15] F. Hutter, L. Kotthoff, J. Vanschoren, Automated Machine Learning: Methods, Systems, Challenges, Springer Nature, 2019.

[16] S. Węglarczyk, Kernel density estimation and its application, in: ITM Web of Conferences, 2018.