Contents lists available at ScienceDirect

Software Impacts

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

Original software publication

Modelly: An open source all in one python package for developing machine learning models

Tushar Sarkar [*], Disha Shah

Analytica, Mumbai, India

ARTICLE INFO

Keywords:
Modelly
XBNet
Neural networks
Classification
Non-linear model

ABSTRACT

Various machine learning algorithms are developed for classification and prediction purposes. These models have been developed to provide solutions and ease our everyday lives in many fields. Neural networks are used extensively in all fields, yet developing them is a difficult and time-consuming process. In this paper, we discuss our package Modelly which provides interactive no-code as well as low code options for developing, testing, and tuning neural networks and their variants like XBNet. Further, we also provide tree-based models in our package that can also be built interactively. Our package aims to facilitate the entire process of developing machine learning and deep learning models to ease the process of developing real-world applications.

Code metadata

Current code version	1.4.6
Permanent link to code/repository used for this code version	https://github.com/SoftwareImpacts/SIMPAC-2022-143
Permanent link to Reproducible Capsule	https://codeocean.com/capsule/0983862/tree/v1
Legal Code License	MIT
Code versioning system used	Git
Software code languages, tools, and services used	Python3
Compilation requirements, operating environments & dependencies	sklearn, pandas, matplotlib, torch, numpy, xgboost, flask, kivy
If available Link to developer documentation/manual	
Support email for questions	tusharsarkar866@gmail.com, dishashah.shah40@gmail.com

Software metadata

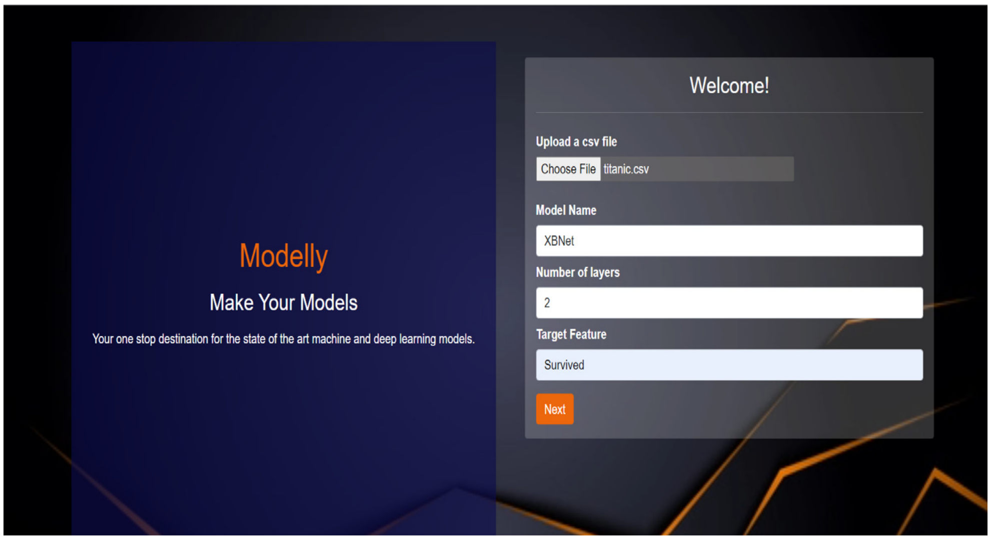
Current software version	1.4.6
Permanent link to executables of this version	*github.com/tusharsarkar3/XBNet*
Permanent link to Reproducible Capsule	*codeocean.com/capsule/ 0983862/tree/v1*
Legal Software License	MIT
Computing platforms/Operating Systems	Linux, OS X, Microsoft Windows, Unix-like
Installation requirements & dependencies	sklearn, pandas, matplotlib, torch, numpy, xgboost, flask, kivy
If available, link to user manual - if formally published include a reference to the publication in the reference list	
Support email for questions	tusharsarkar866@gmail.com, dishashah.shah40@gmail.com

1. Introduction

We use various forms to represent data. Tabular data representation is one of them. Tabulated data is represented as a set of rows and columns where each data sample has a set of properties. When we need to predict the value of a particular column or outcome, we need a predetermined bunch of inputs. This is the classification problem.

DOI of original article: https://doi.org/10.1016/j.iswa.2022.200097.
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 address: tushar.sarkar@somaiya.edu (T. Sarkar).

https://doi.org/10.1016/j.simpa.2022.100407
Received 9 July 2022; Received in revised form 27 July 2022; Accepted 5 August 2022


Modelly

Make Your Models

Your one stop destination for the state of the art machine and deep learning models.

Welcome!

Upload a csv file: Choose File titanic.csv

Model Name: XBNet

Number of layers: 2

Target Feature: Survived

Fig. 1. Landing page of Web interface.

Applications of tabular data classification include diagnosing patients for diseases, detecting spam emails, image segmentation, fraud detection, customer behavior prediction, and many more [1–5]. Predefined algorithms can train the classification models using a set of known parameters. These trained classification models predict the unknown values of the data. Decision trees, logistic regression, and XGBoost are some of the universally used algorithms for training a classification model [6–8]. Our package introduces an interactive way of building neural networks and similar architectures like XBNet alongside the traditional tree-based and linear models that are used to tackle the above-mentioned classification problems [9].

Coding neural networks is a time-consuming process because of the uncertainty of the number of layers, the number of nodes in each layer, and their corresponding relation to the result which makes it prone to a lot of experiments [10]. These experiments require modifying the code every time and running the entire pipeline which wastes a lot of time. Also, making some modifications like boosting certain layers becomes a cumbersome process when implementing XBNet and similar architectures. Our package makes the entire iterative process of modeling easy to implement with rapid prototyping capabilities. It gives better performance, training stability, and interpretability for tabular data with minimum coding requirements for creating any neural network with or without boosting. Overall it substantially reduces the coding requirements and makes it possible for even non-technical people to use the latest techniques according to their requirements.

2. Methodology

We use different components to provide flexibility to the users of our package so that all the functionalities can be provided synchronously or separately. Our package provides two options to the users:

(1) No code implementation using an interactive interface.
In this alternative, the user is provided with an interactive graphical user interface(web and desktop application) where the dataset has to be uploaded. After uploading the data, the name of the model to be used has to be selected. Subsequently, the information about the model has to be provided which includes parameters corresponding to the layers in the case of neural networks and XBNet, and hyperparameters for other models. Analysis can be done based on the results by utilizing the graphs and the performance on different metrics generated by the system. Figs. 1, 2, 3, and 4 illustrate the process for using our application.

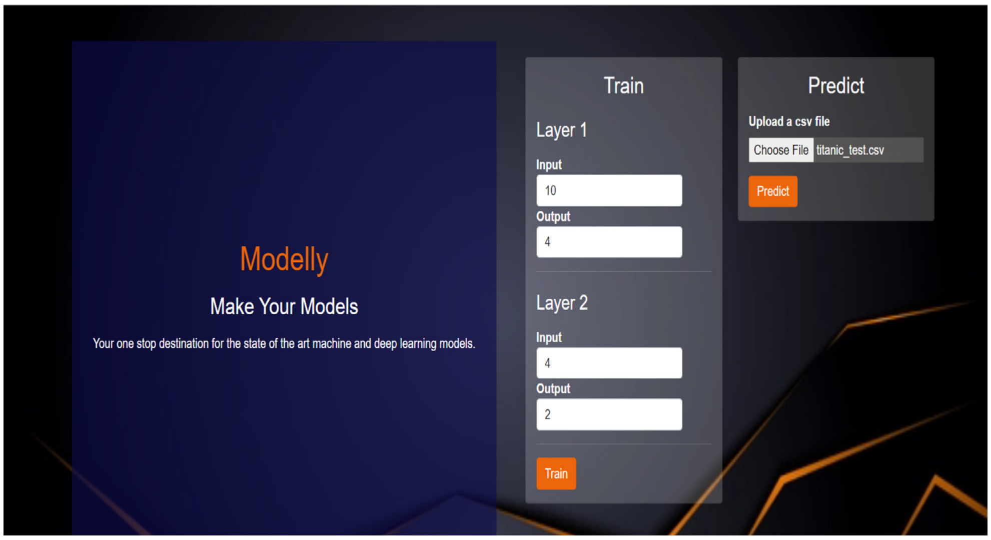
Table 1
Different models available on Modelly.

Name of models
Vanilla neural networks
XBNet
XGBoost
Random forest
LightGBM
Decision tree

(2) Low code implementation for additional flexibility, hyperparameter tuning, and easy integration of XBNet and vanilla neural networks into other software. This part of the package is broken down into several modules like training_utils(train, test, predict, and other utility functions), Seq(overridden base PyTorch Sequential base class), and models. All of the above can be integrated independently or in a sequence depending on the use case of the application. Seq is created for implementing approaches like XBNet where we need to change the sequential flow of data and the pipeline of gradient computation and updation. We have added an XGBoost model but any other model can be added there to create a confluence of two or more techniques. The model defines the class for the architectures that are built on neural networks, whereas the other tree-based models are defined in the apps. The models include various methods that are used to implement the different functionalities of the models and also provide the flexibility to develop new architectures as and when required. Training_utils contains various functions that are used to train, validate, and evaluate our models. It also contains the default metrics on which the results of our model are evaluated. Fig. 5 demonstrates how some of the functions in the low-code implementation can be used and shows how the low-code implementation increases the flexibility of the system by allowing the user to change any component as per their requirement. Using Modelly we can define, train, validate and predict results from vanilla neural networks and XBNet in just one line of code (see Table 1).

3. Software impact

Our software has been used for applying XBNet to various datasets and the results are documented in the paper [9]. Our experiments have shown that it performs better than the State-of-the-Art models on four


Modelly

Make Your Models

Your one stop destination for the state of the art machine and deep learning models.

Train

Layer 1

Input: 10

Output: 4

Layer 2

Input: 4

Output: 2

Train

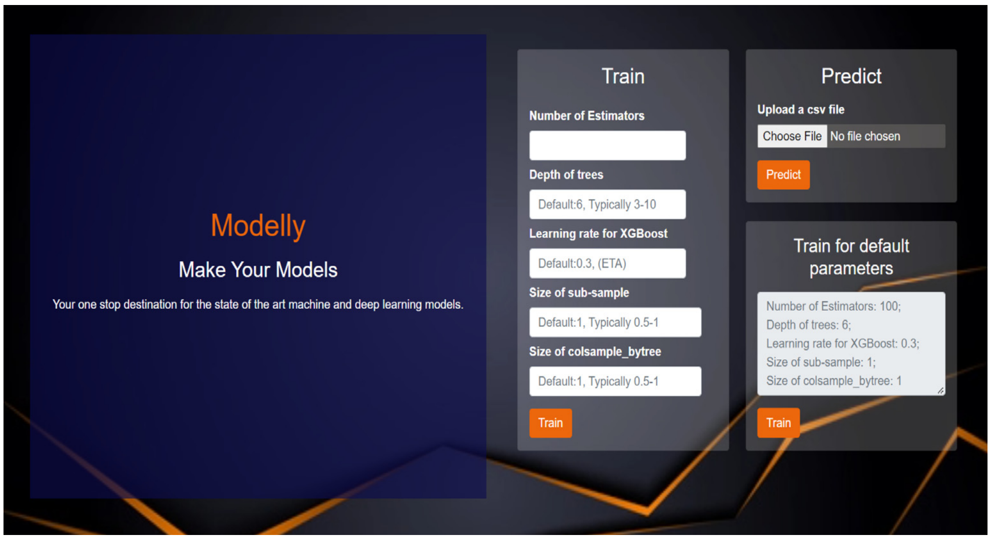
Predict

Upload a csv file

Choose File titanic_test.csv

Predict

Fig. 2. Second page for entering details if neural network is selected.


Modelly

Make Your Models

Your one stop destination for the state of the art machine and deep learning models.

Train

Number of Estimators

Depth of trees: Default:6, Typically 3-10

Learning rate for XGBoost: Default:0.3, (ETA)

Size of sub-sample: Default:1, Typically 0.5-1

Size of colsample_bytree: Default:1, Typically 0.5-1

Train

Predict

Upload a csv file

Choose File No file chosen

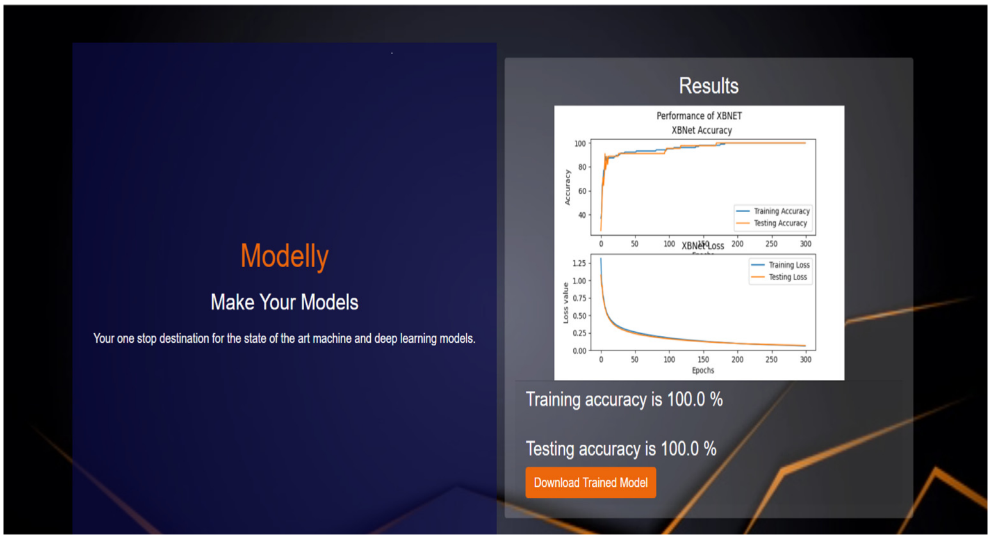
Predict

Train for default parameters

Number of Estimators: 100;
Depth of trees: 6;
Learning rate for XGBoost: 0.3;
Size of sub-sample: 1;
Size of colsample_bytree: 1

Train

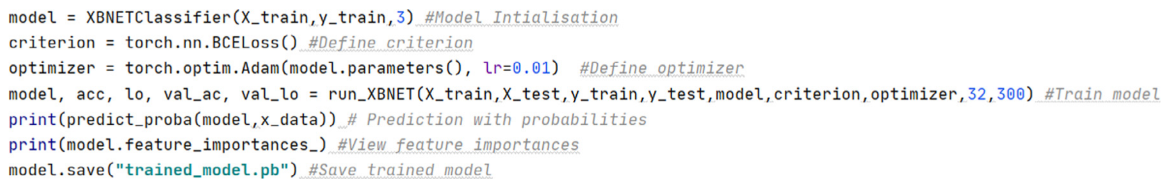
Fig. 3. Second page for hyperparameter tuning if XGBoost is selected.



Training accuracy is 100.0 %

Testing accuracy is 100.0 %

Fig. 4. Results after training.



```python
model = XBNETClassifier(X_train,y_train,3) #Model Intialisation
criterion = torch.nn.BCELoss() #Define criterion
optimizer = torch.optim.Adam(model.parameters(), lr=0.01)  #Define optimizer
model, acc, lo, val_ac, val_lo = run_XBNET(X_train,X_test,y_train,y_test,model,criterion,optimizer,32,300) #Train model
print(predict_proba(model,x_data)) # Prediction with probabilities
print(model.feature_importances_) #View feature importances
model.save("trained_model.pb") #Save trained model
```

Fig. 5. Example of low-code implementation.

out of the seven datasets. It exhibits robust performance and generalizes well on unseen data. As our package provides the performance on various metrics after training and also plots the graphs for training and testing, we can easily analyze the training process and make any changes necessary without writing a single line of code.

Myriad of notebooks on Kaggle have used our package in competitions and have achieved excellent results. Modelly's low code requirements facilitate even non-technical professionals to use such cutting-edge technology for solving business problems easily and efficiently promptly. The metrics provided by XBNet after training a model make the evaluation process easier which helps in the process of hyperparameter tuning. Our package was downloaded 14,285 times after its release which epitomizes its superior usability, popularity, and performance in the research community.

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.

Acknowledgments

I would like to thank Chandan Sarkar, Mallika Sarkar, Pinki Shah, and Nehal Shah for their constant guidance and valuable feedback. I am also grateful to Aparna Sarkar, Sneha Kothi, and the entire community for their priceless suggestions which went a long way toward improving our package.

References

[1] M.A. Myszczynska, P.N. Ojamies, A. Lacoste, D. Neil, A. Saffari, R. Mead, G.M. Hautbergue, J.D. Holbrook, L. Ferraiuolo, Applications of machine learning to diagnosis and treatment of neurodegenerative diseases, Nat. Rev. Neurol. 16 (8) (2020) 440–456.

[2] S. Nandhini, J.M. KS, Performance evaluation of machine learning algorithms for email spam detection, in: 2020 International Conference on Emerging Trends in Information Technology and Engineering, Ic-ETITE, IEEE, 2020, pp. 1–4.

[3] A.D. Kulkarni, Deep Convolution Neural Networks for Image Classification, The Science and Information (SAI) Organization, 2022.

[4] K. Ramani, I. Suneetha, N. Pushpalatha, P. Harish, Gradient boosting techniques for credit card fraud detection, J. Algebr. Stat. 13 (3) (2022) 553–558.

[5] T. Zatonatska, O. Dluhopolskyi, T. Artyukh, K. Tymchenko, Forecasting the behavior of target segments to activate advertising tools: Case of mobile operator Vodafone Ukraine, Econ.-Innov. Res. J. 10 (1) (2022) 87–104.

[6] Y.-Y. Song, L. Ying, Decision tree methods: applications for classification and prediction, Shanghai Arch. Psychiatry 27 (2) (2015) 130.

[7] R.E. Wright, Logistic regression, American Psychological Association, 1995.

[8] T. Chen, C. Guestrin, Xgboost: A scalable tree boosting system, in: Proceedings of the 22nd Acm Sigkdd International Conference on Knowledge Discovery and Data Mining, 2016, pp. 785–794.

[9] T. Sarkar, XBNet: An extremely boosted neural network, Intell. Syst. Appl. (2022) 200097, http://dx.doi.org/10.1016/j.iswa.2022.200097, URL https://www.sciencedirect.com/science/article/pii/S2667305322000370.

[10] S.-C. Wang, Artificial neural network, in: Interdisciplinary Computing in Java Programming, Springer, 2003, pp. 81–100.