Journal of Machine Learning Research 26 (2025) 1-16 Submitted 3/25; Revised 8/25; Published 8/25

Talent: A Tabular Analytics and Learning Toolbox

Si-Yang Liu liusy@lamda.nju.edu.cn
Hao-Run Cai caihr@lamda.nju.edu.cn
Qi-Le Zhou zhouql@lamda.nju.edu.cn
Huai-Hong Yin yinhh@lamda.nju.edu.cn
Tao Zhou zhout@lamda.nju.edu.cn
Jun-Peng Jiang jiangjp@lamda.nju.edu.cn
Han-Jia Ye yehj@lamda.nju.edu.cn
School of Artificial Intelligence, Nanjing University, China
National Key Laboratory for Novel Software Technology, Nanjing University, 210023, China

Editor: Zeyi Wen

Abstract

Tabular data is a prevalent source in machine learning. While classical methods have proven effective, deep learning methods for tabular data are emerging as flexible alternatives due to their capacity to uncover hidden patterns and capture complex interactions. Considering that deep tabular methods exhibit diverse design philosophies, including the ways they handle features, design learning objectives, and construct model architectures, we introduce Talent (**T**abular **A**nalytics and **LE**ar**N**ing **T**oolbox), a versatile toolbox for utilizing, analyzing, and comparing these methods. Talent includes over 35 deep tabular prediction methods, offering various encoding and normalization modules, all within a unified, easily extensible interface. We demonstrate its design, application, and performance evaluation in case studies. The code is available at https://github.com/LAMDA-Tabular/TALENT.

Keywords: Tabular Data, Deep Learning, Deep Tabular Prediction, Machine Learning

1 Introduction

Machine learning has achieved significant success across a wide range of domains. Tabular data, characterized by datasets arranged in a table format, represents one of the most prevalent types of data applied in machine learning applications such as click-through rate (CTR) prediction (Guo et al., 2017), cybersecurity (Buczak and Guven, 2016), medical analysis (Schwartz et al., 2007), and identity protection (Liu et al., 2022). In these datasets, each row typically represents an individual instance, while each column corresponds to a specific attribute or feature. In the context of supervised learning, each training instance is paired with a label, which can be discrete for classification tasks and continuous for regression tasks. Machine learning models are designed to learn a mapping from the input instances to their corresponding labels using the training data, with the goal of generalizing this mapping to unseen data from the same distribution.

Methodologies for analyzing tabular datasets have evolved substantially. Classical techniques such as Logistic Regression, Support Vector Machine, Multi-Layer Perceptron, and decision trees have long served as the foundation for numerous algorithms (Bishop, 2006).

In practical applications, tree-based ensemble methods—including XGBoost (Chen and Guestrin, 2016), LightGBM (Ke et al., 2017), and CatBoost (Prokhorenkova et al., 2018)—have demonstrated substantial performance improvements. Inspired by the success of Deep Neural Networks in visual and linguistic tasks (Simonyan and Zisserman, 2015; Vaswani et al., 2017; Devlin et al., 2019), researchers have developed deep learning models tailored for tabular data (Zhang et al., 2016; Borisov et al., 2022). The advanced deep tabular models excel at uncovering hidden patterns and improving predictive performance (Chen et al., 2023b; Gorishniy et al., 2024; Ye et al., 2024b). These methods offer several key advantages, such as seamless integration into multi-modal pipelines (Gorishniy et al., 2021; Jiang et al., 2024a), efficient modeling of complex feature interactions (Wang et al., 2017), and flexibility in gradient-based optimization (Borisov et al., 2022). These strengths make them especially well-suited for tasks involving diverse data types or multi-task learning, where shared representations across related tasks can further enhance both performance and efficiency (Zhou et al., 2023; Yan et al., 2024; Ye et al., 2023).

Although deep learning offers significant benefits for analyzing tabular data, its practical application is often hindered by the lack of consistent interfaces and varying preprocessing requirements among different methods. We introduce a versatile and comprehensive toolbox, Talent (**T**abular **A**nalytics and **LE**ar**N**ing **T**oolbox), designed for tabular data prediction. Talent integrates both classical and advanced deep methods into a unified architecture, standardizing interfaces, streamlining preprocessing steps, and enabling fair, consistent evaluation across diverse scenarios. Additionally, the toolkit supports the composition of effective deep learning modules for tabular data, offering scalable solutions tailored to various complexities and data-specific needs.

2 Advantages and Comparison to Existing Toolkits

To highlight the advantages of Talent, we compare it against several widely used toolkits, including RTDL (Gorishniy et al., 2021), Pytorch_tabular (Joseph, 2021), DeepTables (Jian Yang, 2022), Pytorch_widedeep (Zaurin and Mulinka, 2023), and Pytorch_frame(Hu et al., 2024), as shown in Table 1.

Model Diversity. As outlined in Appendix A.1, Talent offers over 35 deep tabular methods, significantly exceeding the model diversity of other toolkits, allowing users to select the best-fit model based on the complexity and specifics of their tasks. More than

Table 1: The main differences between Talent and other tabular deep learning toolkits.

Toolkit	Deep Methods	Methods in Three Years	Datasets	Encoding Technologies	General Models
RTDL	9	0	11	✓	✗
Pytorch_tabular	11	1	0	✗	✗
DeepTables	15	0	9	✗	✗
Pytorch_widedeep	12	0	8	✗	✗
Pytorch_frame	8	2	55	✗	✗
Talent	**35+**	**25+**	**300**	✓	✓

60% of these methods have been introduced within the last three years, ensuring they are cutting-edge and relevant to modern research challenges. Moreover, Talent uniquely includes general tabular models—pretrained models ready for direct use on downstream tasks—which are absent in other toolkits. These general models enhance the toolbox's versatility, supporting a broader range of tasks and offering greater flexibility across various data types and prediction scenarios.

Encoding Techniques. In addition to supporting a wide variety of encoding strategies for categorical features, Talent provides eight distinct techniques for encoding numerical features, as detailed in Appendix A.2, offering a more comprehensive approach to data preprocessing. This versatility in encoding allows users to customize their data representations to suit specific analytical and modeling requirements, enhancing both the adaptability and performance of their models across diverse tasks.

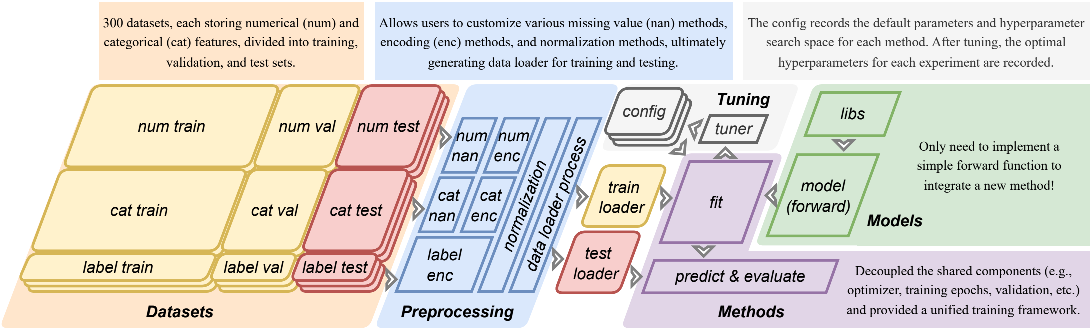
Extensibility. The modular architecture of Talent is specifically designed for both flexibility and scalability. Users can effortlessly incorporate new models and methods following the guidance provided in Appendix C, enabling the toolkit to adapt to evolving research demands and practical applications. Whether enhancing its functionality or introducing innovative approaches, the framework's extensible design ensures that it remains a powerful and up-to-date resource in the rapidly evolving field of tabular deep learning.

3 Toolbox Usage

In this section, we introduce the dependencies and workflow for using Talent.

3.1 Dependencies

Talent leverages open-source libraries to support its advanced data processing and machine learning functionalities, following the organized code structure introduced by RTDL (Gorishniy et al., 2021). For model optimization and hyperparameter tuning, it utilizes Optuna (Akiba et al., 2019). These carefully chosen dependencies offer users a powerful, flexible, and efficient toolkit for addressing various challenges in tabular data analysis.



300 datasets, each storing numerical (num) and categorical (cat) features, divided into training, validation, and test sets.

Allows users to customize various missing value (nan) methods, encoding (enc) methods, and normalization methods, ultimately generating data loader for training and testing.

The config records the default parameters and hyperparameter search space for each method. After tuning, the optimal hyperparameters for each experiment are recorded.

Only need to implement a simple forward function to integrate a new method!

Decoupled the shared components (e.g., optimizer, training epochs, validation, etc.) and provided a unified training framework.

Figure 1: Flowchart depicting the data prediction process with Talent.

3.2 The workflow of Talent

The flowchart in Figure 1 visually illustrates the streamlined workflow enabled by our toolbox. It begins with data loading, followed by preprocessing, hyperparameter tuning, model training, prediction, and ultimately evaluation. This structured workflow ensures a smooth transition from raw data to meaningful results.

The following example demonstrates how to use the toolbox to run experiments across multiple seeds, ensuring a robust evaluation of method performance:

```python
from tqdm import tqdm
from TALENT.model.utils import (
get_deep_args,show_results,tune_hyper_parameters, get_method,set_seeds)
from TALENT.model.lib.data import  get_dataset
args, default_para, opt_space = get_deep_args()
train_val_data, test_data, info = get_dataset(args.dataset, args.dataset_path)
if args.tune:
    args = tune_hyper_parameters(args, opt_space, train_val_data, info)
for seed in tqdm(range(args.seed_num)):
    args.seed = seed
    method = get_method(args.model_type)(args, info["task_type"] == "regression")
    time_cost = method.fit(train_val_data, info, train=True)
    vres, metric_name, predict_logits = method.predict(test_data, info)
```

- The **get_args** function retrieves and parses the arguments, default hyperparameters, and the optimization space for hyperparameter tuning.
- The **get_dataset** function loads the specified dataset from the provided path, splits it into training/validation and test sets, and provides additional information about the dataset.
- If hyperparameter tuning is enabled, the **tune_hyper_parameters** function adjusts the arguments based on the optimization space and the training/validation data.
- The **get_method** function selects the appropriate model class based on the model type specified in **args.model_type**.
- The seed is updated for each iteration, and the performance metrics and predictions are recorded for each seed, enabling a comprehensive evaluation of the model across different initializations. Classification tasks are evaluated using metrics like Accuracy, F1-Score, Log Loss, and AUC, while regression tasks are assessed with MAE, RMSE, and R2 (Lewis-Beck, 2015). As an application of Talent, we conducted fair comparisons of representative methods on 300 benchmark datasets (Ye et al., 2024a), as detailed in Appendix D.

4 Conclusion

We introduce Talent, a machine learning toolbox designed for tabular data prediction tasks. Talent incorporates both classical and deep tabular methods and includes modules for hyperparameter tuning and preprocessing, aiming to improve learning efficiency and performance on tabular datasets. Additionally, we use Talent to conduct fair comparisons of recent deep tabular methods across a wide range of datasets. The toolbox is designed to be user-friendly and accessible to practitioners across diverse fields, providing a unified interface that is flexible and easily adaptable for integration with newly designed methods.

Acknowledgments

This work is partially supported by NSFC (62376118), Key Program of Jiangsu Science Foundation (BK20243012), the Fundamental Research Funds for the Central Universities (14380018), Collaborative Innovation Center of Novel Software Technology and Industrialization. Thanks to Hengzhe Zhang for providing a Scikit-Learn compatible wrapper for Talent, and to Chen-Ming Xu and Han Li for improving the documentation presentation of Talent.

References

Takuya Akiba, Shotaro Sano, Toshihiko Yanase, Takeru Ohta, and Masanori Koyama. Optuna: A next-generation hyperparameter optimization framework. In *KDD*, pages 2623–2631, 2019.

Sercan Ö. Arik and Tomas Pfister. Tabnet: Attentive interpretable tabular learning. In *AAAI*, pages 6679–6687, 2021.

Sarkhan Badirli, Xuanqing Liu, Zhengming Xing, Avradeep Bhowmik, and Sathiya S. Keerthi. Gradient boosting neural networks: Grownet. *CoRR*, abs/2002.07971, 2020.

Christopher Bishop. *Pattern recognition and machine learning*. Springer, 2006.

David Bonet, Daniel Mas Montserrat, Xavier Giró-i-Nieto, and Alexander G. Ioannidis. Hyperfast: Instant classification for tabular data. In *AAAI*, pages 11114–11123, 2024.

Vadim Borisov, Tobias Leemann, Kathrin Seßler, Johannes Haug, Martin Pawelczyk, and Gjergji Kasneci. Deep neural networks and tabular data: A survey. *CoRR*, abs/2110.01889, 2022.

Anna L. Buczak and Erhan Guven. A survey of data mining and machine learning methods for cyber security intrusion detection. *IEEE Communications surveys & tutorials*, 18(2): 1153–1176, 2016.

Jintai Chen, Kuanlun Liao, Yao Wan, Danny Z. Chen, and Jian Wu. Danets: Deep abstract networks for tabular data classification and regression. In *AAAI*, pages 3930–3938, 2022.

Jintai Chen, KuanLun Liao, Yanwen Fang, Danny Chen, and Jian Wu. Tabcaps: A capsule neural network for tabular data classification with bow routing. In *ICLR*, 2023a.

Jintai Chen, Jiahuan Yan, Danny Ziyi Chen, and Jian Wu. Excelformer: A neural network surpassing gbdts on tabular data. *CoRR*, abs/2301.02819, 2023b.

Kuan-Yu Chen, Ping-Han Chiang, Hsin-Rung Chou, Ting-Wei Chen, and Tien-Hao Chang. Trompt: Towards a better deep neural network for tabular data. In *ICML*, pages 4392–4434, 2023c.

Tianqi Chen and Carlos Guestrin. Xgboost: A scalable tree boosting system. In *KDD*, pages 785–794, 2016.

Yi Cheng, Renjun Hu, Haochao Ying, Xing Shi, Jian Wu, and Wei Lin. Arithmetic feature interaction is necessary for deep tabular learning. In *AAAI*, pages 11516–11524, 2024.

Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. BERT: pre-training of deep bidirectional transformers for language understanding. In *NAACL-HLT*, pages 4171–4186, 2019.

Jacob Goldberger, Sam T. Roweis, Geoffrey E. Hinton, and Ruslan Salakhutdinov. Neighbourhood components analysis. In *NIPS*, pages 513–520, 2004.

Yury Gorishniy, Ivan Rubachev, Valentin Khrulkov, and Artem Babenko. Revisiting deep learning models for tabular data. In *NeurIPS*, pages 18932–18943, 2021.

Yury Gorishniy, Ivan Rubachev, and Artem Babenko. On embeddings for numerical features in tabular deep learning. In *NeurIPS*, pages 24991–25004, 2022.

Yury Gorishniy, Ivan Rubachev, Nikolay Kartashev, Daniil Shlenskii, Akim Kotelnikov, and Artem Babenko. Tabr: Tabular deep learning meets nearest neighbors in 2023. In *ICLR*, 2024.

Yury Gorishniy, Akim Kotelnikov, and Artem Babenko. Tabm: Advancing tabular deep learning with parameter-efficient ensembling. In *ICLR*, 2025.

Huifeng Guo, Ruiming Tang, Yunming Ye, Zhenguo Li, and Xiuqiang He. Deepfm: A factorization-machine based neural network for CTR prediction. In *IJCAI*, pages 1725–1731, 2017.

Noah Hollmann, Samuel Müller, Katharina Eggensperger, and Frank Hutter. Tabpfn: A transformer that solves small tabular classification problems in a second. In *ICLR*, 2023.

Noah Hollmann, Samuel Müller, Lennart Purucker, Arjun Krishnakumar, Max Körfer, Shi Bin Hoo, Robin Tibor Schirrmeister, and Frank Hutter. Accurate predictions on small data with a tabular foundation model. *Nature*, 637(8045):319–326, 2025.

David Holzmüller, Léo Grinsztajn, and Ingo Steinwart. Better by default: Strong pre-tuned mlps and boosted trees on tabular data. *CoRR*, abs/2407.04491, 2024.

Weihua Hu, Yiwen Yuan, Zecheng Zhang, Akihiro Nitta, Kaidi Cao, Vid Kocijan, Jure Leskovec, and Matthias Fey. Pytorch frame: A modular framework for multi-modal tabular learning. *CoRR*, abs/2404.00776, 2024.

Xin Huang, Ashish Khetan, Milan Cvitkovic, and Zohar S. Karnin. Tabtransformer: Tabular data modeling using contextual embeddings. *CoRR*, abs/2012.06678, 2020.

Alan Jeffares, Tennison Liu, Jonathan Crabbé, Fergus Imrie, and Mihaela van der Schaar. Tangos: Regularizing tabular neural networks through gradient orthogonalization and specialization. In *ICLR*, 2023.

Haifeng Wu Jian Yang, Xuefeng Li. DeepTables: A Deep Learning Python Package for Tabular Data. https://github.com/DataCanvasIO/DeepTables, 2022. Version 0.2.x.

Jun-Peng Jiang, Han-Jia Ye, Leye Wang, Yang Yang, Yuan Jiang, and De-Chuan Zhan. Tabular insights, visual impacts: Transferring expertise from tables to images. In *ICML*, pages 21988–22009, 2024a.

Xiangjian Jiang, Andrei Margeloiu, Nikola Simidjievski, and Mateja Jamnik. Protogate: Prototype-based neural networks with global-to-local feature selection for tabular biomedical data. In *ICML*, pages 21844–21878, 2024b.

Manu Joseph. Pytorch tabular: A framework for deep learning with tabular data. *CoRR*, abs/2104.13638, 2021.

Guolin Ke, Qi Meng, Thomas Finley, Taifeng Wang, Wei Chen, Weidong Ma, Qiwei Ye, and Tie-Yan Liu. Lightgbm: A highly efficient gradient boosting decision tree. In *NIPS*, pages 3146–3154, 2017.

Günter Klambauer, Thomas Unterthiner, Andreas Mayr, and Sepp Hochreiter. Self-normalizing neural networks. In *NIPS*, pages 971–980, 2017.

Ron Kohavi and Mehran Sahami. Error-based and entropy-based discretization of continuous features. In *KDD*, pages 114–119, 1996.

Michael S. Lewis-Beck. *Applied regression: An introduction*, volume 22. Sage publications, 2015.

Ling Li and Hsuan-Tien Lin. Ordinal regression by extended binary classification. In *NIPS*, pages 865–872, 2006.

William H. Libaw and Leonard J. Craig. A photoelectric decimal-coded shaft digitizer. *Transactions of the IRE Professional Group on Electronic Computers*, 2(3):1–4, 1953.

Bo Liu, Ming Ding, Sina Shaham, Wenny Rahayu, Farhad Farokhi, and Zihuai Lin. When machine learning meets privacy: A survey and outlook. *ACM Computing Surveys*, 54(2): 31:1–31:36, 2022.

Sascha Marton, Stefan Lüdtke, Christian Bartelt, and Heiner Stuckenschmidt. GRANDE: gradient-based decision tree ensembles for tabular data. In *ICLR*, 2024.

Duncan C. McElfresh, Sujay Khandagale, Jonathan Valverde, Vishak Prasad C., Ganesh Ramakrishnan, Micah Goldblum, and Colin White. When do neural nets outperform boosted trees on tabular data? In *NeurIPS*, pages 76336–76369, 2023.

Daniele Micci-Barreca. A preprocessing scheme for high-cardinality categorical attributes in classification and prediction problems. *ACM SIGKDD explorations newsletter*, 3(1): 27–32, 2001.

Youssef Nader, Leon Sixt, and Tim Landgraf. DNNR: differential nearest neighbors regression. In *ICML*, pages 16296–16317, 2022.

Sergei Popov, Stanislav Morozov, and Artem Babenko. Neural oblivious decision ensembles for deep learning on tabular data. In *ICLR*, 2020.

Liudmila Ostroumova Prokhorenkova, Gleb Gusev, Aleksandr Vorobev, Anna Veronika Dorogush, and Andrey Gulin. Catboost: unbiased boosting with categorical features. In *NeurIPS*, pages 6639–6649, 2018.

Jingang Qu, David Holzmüller, Gaël Varoquaux, and Marine Le Morvan. TabICL: A tabular foundation model for in-context learning on large data. In *ICML*, 2025.

Matteo Rizzo, Ebru Ayyurek, Andrea Albarelli, and Andrea Gasparetto. Leveraging periodicity for tabular deep learning. *Electronics*, 14(6), 2025.

Lisa M Schwartz, Steven Woloshin, and H Gilbert Welch. The drug facts box: providing consumers with simple tabular data on drug benefit and harm. *Medical Decision Making*, 27(5):655–662, 2007.

Deval Shah, Zi Yu Xue, and Tor M. Aamodt. Label encoding for regression networks. In *ICLR*, 2022.

David J Sheskin. *Handbook of parametric and nonparametric statistical procedures*. Chapman and hall/CRC, 2003.

Karen Simonyan and Andrew Zisserman. Very deep convolutional networks for large-scale image recognition. In *ICLR*, 2015.

Gowthami Somepalli, Avi Schwarzschild, Micah Goldblum, C. Bayan Bruss, and Tom Goldstein. SAINT: Improved neural networks for tabular data via row attention and contrastive pre-training. In *NeurIPS Workshop*, 2022.

Weiping Song, Chence Shi, Zhiping Xiao, Zhijian Duan, Yewen Xu, Ming Zhang, and Jian Tang. Autoint: Automatic feature interaction learning via self-attentive neural networks. In *CIKM*, pages 1161–1170, 2019.

Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Łukasz Kaiser, and Illia Polosukhin. Attention is all you need. In *NIPS*, 2017.

Ruoxi Wang, Bin Fu, Gang Fu, and Mingliang Wang. Deep & cross network for ad click predictions. In *ADKDD*, 2017.

Ruoxi Wang, Rakesh Shivanna, Derek Zhiyuan Cheng, Sagar Jain, Dong Lin, Lichan Hong, and Ed H. Chi. DCN V2: improved deep & cross network and practical lessons for web-scale learning to rank systems. In *WWW*, pages 1785–1797, 2021.

Kilian Q. Weinberger, Anirban Dasgupta, John Langford, Alexander J. Smola, and Josh Attenberg. Feature hashing for large scale multitask learning. In *ICML*, pages 1113–1120, 2009.

Yeming Wen, Dustin Tran, and Jimmy Ba. Batchensemble: an alternative approach to efficient ensemble and lifelong learning. In *ICLR*, 2020.

Jing Wu, Suiyao Chen, Qi Zhao, Renat Sergazinov, Chen Li, Shengjie Liu, Chongchao Zhao, Tianpei Xie, Hanqing Guo, Cheng Ji, Daniel Cociorva, and Hakan Brunzell. Switchtab: Switched autoencoders are effective tabular learners. In *AAAI*, pages 15924–15933, 2024.

Chenwei Xu, Yu-Chao Huang, Jerry Yao-Chieh Hu, Weijian Li, Ammar Gilani, Hsi-Sheng Goan, and Han Liu. Bishop: Bi-directional cellular learning for tabular data with generalized sparse modern hopfield model. In *ICML*, pages 55048–55075, 2024.

Jiahuan Yan, Jintai Chen, Yixuan Wu, Danny Z. Chen, and Jian Wu. T2G-FORMER: organizing tabular features into relation graphs promotes heterogeneous feature interaction. In *AAAI*, pages 10720–10728, 2023.

Jiahuan Yan, Bo Zheng, Hongxia Xu, Yiheng Zhu, Danny Z. Chen, Jimeng Sun, Jian Wu, and Jintai Chen. Making pre-trained language models great on tabular prediction. In *ICLR*, 2024.

Han-Jia Ye, Qi-Le Zhou, and De-Chuan Zhan. Training-free generalization on heterogeneous tabular data via meta-representation. *CoRR*, abs/2311.00055, 2023.

Han-Jia Ye, Si-Yang Liu, Hao-Run Cai, Qi-Le Zhou, and De-Chuan Zhan. A closer look at deep learning on tabular data. *CoRR*, abs/2407.00956, 2024a.

Han-Jia Ye, Huai-Hong Yin, and De-Chuan Zhan. Modern neighborhood components analysis: A deep tabular baseline two decades later. *CoRR*, abs/2407.03257, 2024b.

Hangting Ye, Wei Fan, Xiaozhuang Song, Shun Zheng, He Zhao, Dan dan Guo, and Yi Chang. Ptarl: Prototype-based tabular representation learning via space calibration. In *ICLR*, 2024c.

Javier Rodriguez Zaurin and Pavol Mulinka. pytorch-widedeep: A flexible package for multimodal deep learning. *Journal of Open Source Software*, 8(86):5027, June 2023.

Weinan Zhang, Tianming Du, and Jun Wang. Deep learning over multi-field categorical data - - A case study on user response prediction. In *ECIR*, pages 45–57, 2016.

Qi-Le Zhou, Han-Jia Ye, Leye Wang, and De-Chuan Zhan. Unlocking the transferability of tokens in deep models for tabular data. *CoRR*, abs/2310.15149, 2023.

Appendix A. Supported Methods and Encoding Techniques

We provide an overview of the tabular prediction methods and encoding techniques supported by Talent.

A.1 Supported Methods

In Talent, we implement a comprehensive range of models that implement the mapping f from features to outputs, all with a unified interface. These models encompass classical methods, tree-based methods, and advanced deep learning techniques for tabular data.

Classical models in Talent include K-Nearest Neighbors (KNN) and Support Vector Machines (SVM) for various tasks, as well as Linear Regression (LR) for regression tasks. For classification tasks, it supports Logistic Regression (LogReg), Naive Bayes, and Nearest Class Mean (NCM).

Tree-based methods in Talent incorporate powerful algorithms such as Random Forest, XGBoost (Chen and Guestrin, 2016), CatBoost (Prokhorenkova et al., 2018), and LightGBM (Ke et al., 2017). These algorithms are well-known for their high efficiency and strong predictive performance across diverse datasets.

Our toolbox provides a comprehensive selection of state-of-the-art deep tabular prediction methods, each meticulously designed to address specific challenges in tabular data analysis.

- **MLP**: A multi-layer neural network, implemented based on Gorishniy et al. (2021).
- **ResNet**: A deep neural network that employs skip connections across multiple layers, which is implemented as described in Gorishniy et al. (2021).
- **MLP_PLR** (Gorishniy et al., 2022): An improved multilayer perceptron (MLP), which utilizes periodic activations.
- **RealMLP** (Holzmüller et al., 2024): An improved version of multilayer perceptron.
- **SNN** (Klambauer et al., 2017): An MLP-like architecture utilizing the SELU activation, which facilitates the training of deeper neural networks.
- **TabM** (Gorishniy et al., 2025): A model based on MLP and variations of BatchEnsemble (Wen et al., 2020).
- **DANets** (Chen et al., 2022): A neural network designed to enhance tabular data processing by grouping correlated features and reducing computational complexity.
- **TabCaps** (Chen et al., 2023a): A capsule network that encapsulates all feature values of a record into vectorial representations.
- **BiSHop** (Xu et al., 2024): An end-to-end framework for deep tabular learning which leverages a sparse Hopfield model with adaptable sparsity, enhanced by column-wise and row-wise modules.
- **DCNv2** (Wang et al., 2021): A method that combines an MLP-like module with a feature crossing module, incorporating both linear layers and multiplicative interactions.
- **NODE** (Popov et al., 2020): A tree-mimic method that generalizes oblivious decision trees, combining gradient-based optimization with hierarchical representation learning.
- **GrowNet** (Badirli et al., 2020): A gradient boosting framework utilizing shallow neural networks as weak learners.
- **TabNet** (Arik and Pfister, 2021): A tree-mimic method employing sequential attention for feature selection, with interpretability and self-supervised learning capabilities.
- **GRANDE** (Marton et al., 2024): A tree-mimic method for learning hard, axis-aligned decision tree ensembles using end-to-end gradient descent.
- **ProtoGate** (Jiang et al., 2024b): A prototype-based model for high-dimensional, low-sample-size biomedical data, which adapts global and local feature selection for improved prediction accuracy and interpretability.
- **TabR** (Gorishniy et al., 2024): A deep learning model that integrates a KNN component to enhance tabular data predictions via an efficient attention-like mechanism.
- **ModernNCA** (Ye et al., 2024b): A deep tabular model inspired by traditional Neighbor Component Analysis (Goldberger et al., 2004), which makes predictions based on the relationships with neighbors in a learned embedding space.
- **DNNR** (Nader et al., 2022) enhances KNN predictions by using local gradients and Taylor approximations for more accurate and interpretable predictions.

- **AutoInt** (Song et al., 2019): A token-based method that leverages a multi-head self-attentive neural network to automatically learn high-order feature interactions.
- **TabTransformer** (Huang et al., 2020): A token-based method that transforms categorical features into contextual embeddings to enhance tabular data modeling.
- **FT-Transformer** (Gorishniy et al., 2021): A token-based method which transforms features to embeddings, followed by a series of attention-based transformations.
- **Saint** (Somepalli et al., 2022): A token-based method using row and column attention mechanisms for tabular data.
- **Trompt** (Chen et al., 2023c): A prompt-based deep neural network for tabular data, designed to separate learning into intrinsic column features and sample-specific feature importance.
- **T2G-former** (Yan et al., 2023): A token-based method that processes data guided by relation graphs and uses a Cross-level Readout for global semantics in prediction.
- **ExcelFormer** (Chen et al., 2023b): A token-based model featuring a semi-permeable attention module for tabular data prediction, with tailored data augmentation and an attentive feedforward network.
- **AMFormer** (Cheng et al., 2024): A token-based method which improves the transformer architecture for tabular data by incorporating parallel addition and multiplication attention mechanisms, and uses prompt tokens to constrain feature interactions.
- **TANGOS** (Jeffares et al., 2023): A regularization-based model that uses gradient attributions to promote neuron specialization and orthogonalization for tabular data.
- **SwitchTab** (Wu et al., 2024): A self-supervised method tailored for tabular data that improves representation learning through an asymmetric encoder-decoder framework.
- **PTaRL** (Ye et al., 2024c): A regularization-based framework that enhances prediction by constructing and projecting into a prototype-based space.
- **TabPTM** (Ye et al., 2023): A general method for tabular data that standardizes heterogeneous datasets using meta-representations, allowing a pre-trained model to generalize to unseen datasets without additional training.
- **TabPFN** (Hollmann et al., 2023): A general model which involves the use of pre-trained deep neural networks that can be directly applied to other tabular classification tasks.
- **HyperFast** (Bonet et al., 2024): A meta-trained hypernetwork that generates task-specific neural networks for instant classification of tabular data.
- **TabPFN v2** (Hollmann et al., 2025): An improved version of TabPFN.
- **TabICL** (Qu et al., 2025): A pretrained model similar to TabPFN v2, but with faster processing speed and better performance on large-scale datasets.
- **TabAutoPNPNet** (Rizzo et al., 2025): A tabular model based on periodicity, particularly the Fourier transform and Chebyshev polynomials, with performance on par with or superior to FT-Transformer.

A.2 Encoding Techniques

According to Gorishniy et al. (2022), embeddings for numerical features significantly improve the performance of deep learning models on tabular data by providing more expressive and powerful initial representations. This approach is beneficial for both MLPs and advanced Transformer-like architectures. In Talent, we integrate various numerical encoding techniques, improving the input quality for machine learning models. The diverse selection of encoding methods ensures effective and customized data preprocessing for different analytical needs. Here are the encoding methods included in Talent:

- **Quantile-based Binning (Q_bins)** constructs bins by dividing value ranges according to the quantiles of the individual feature distributions, replacing the original values with their corresponding bin indices.
- **Target-aware Binning (T_bins)** creates bins using training labels to correspond to narrow ranges of possible target values. This approach is similar to the "C4.5 Discretization" algorithm (Kohavi and Sahami, 1996), which splits the value range of each feature using the target as guidance.
- **Quantile-based Unary Encoding (Q_Unary)** (Li and Lin, 2006) converts numerical values into unary binary-encoded bin indices based on quantiles.
- **Target-aware Unary Encoding (T_Unary)** (Li and Lin, 2006) generates unary binary-encoded bin indices using target-aware transformations.
- **Quantile-based Johnson Encoding (Q_Johnson)** (Shah et al., 2022) encodes numerical data based on quantile intervals using Johnson distribution transformations (Libaw and Craig, 1953), replacing original values with Johnson binary-encoded bin indices.
- **Target-aware Johnson Encoding (T_Johnson)** (Shah et al., 2022) applies Johnson transformations with target-aware bins, replacing original values with Johnson binary-encoded bin indices (Libaw and Craig, 1953).
- **Quantile-based Piecewise Linear Encoding (Q_PLE)** (Gorishniy et al., 2022) segments numerical data based on quantiles and applies piecewise linear transformations.
- **Target-aware Piecewise Linear Encoding (T_PLE)** (Gorishniy et al., 2022) builds target-aware bins and applies piecewise linear transformations.

Additionally, Talent incorporates various categorical encoding techniques, including **Ordinal** encoding, **One-Hot** encoding, **Binary** encoding, **Hash** encoding (Weinberger et al., 2009), **Target** encoding (Micci-Barreca, 2001), **Leave-One-Out** encoding, and **CatBoost** encoding (Prokhorenkova et al., 2018).

Appendix B. Usage Example with `scikit_learn`-style Interface

We are pleased to note that the community has contributed an extension named `scikit_talent`[1], which provides a `scikit-learn`-style interface for using models in the Talent framework. This interface supports custom data loading and seamless integration with common Python libraries such as `pandas`, `numpy`, and `scikit-learn`.

Below is a simple usage example of `scikit_talent` that demonstrates how to load a dataset from OpenML, split it into training and testing sets, and evaluate a deep tabular model using `balanced_accuracy_score`:

```python
import numpy as np
import openml
from sklearn.metrics import balanced_accuracy_score
from sklearn.model_selection import train_test_split
from scikit_talent.talent_classifier import DeepClassifier

model = "modernNCA"
```

[1] https://github.com/hengzhe-zhang/scikit-TALENT

```python
M = DeepClassifier(model_type=model)

dataset = openml.datasets.get_dataset(3, download_data=True)
X, y, categorical_indicator, _ = dataset.get_data(target=dataset.default_target_attribute, dataset_format="dataframe"
)
X, y = np.array(X), np.array(y)

dataset_size = 100  # Define training size
X_train_pre, X_test, y_train_pre, y_test = train_test_split(X, y, train_size=dataset_size)

M.fit(X_train_pre, y_train_pre, categorical_indicator)
predictions = M.predict(X_test)
score = balanced_accuracy_score(y_test, predictions)
print(f"{model}: Balanced Accuracy Score = {score}")
```

This example illustrates how users can rapidly prototype and evaluate models using familiar tools and APIs, thereby enhancing the usability and accessibility of the Talent framework for researchers and practitioners alike.

Appendix C. Adding New Methods

Talent is designed to be highly customizable, allowing users to integrate new machine learning methods effortlessly. Whether users are adding a well-known algorithm or experimenting with a novel approach, follow these steps to expand the capabilities of the toolbox, as illustrated in Figure 2:

1. **Register the Model:** Start by registering the new model class in the `model/models` directory. Ensure that this class defines the architecture of the model, specifying how the model will be constructed.
2. **Create the Method Class:** Create a new method class within the `model/methods` directory. This class should inherit from the base class provided in `base.py`. Implement the necessary components of the machine learning method in this class, including the training and prediction processes.
3. **Method Integration:** Integrate the new method into the workflow of Talent by adding its name to the `get_method` function located in `model/utils.py`. This function maps model types to their respective classes, enabling the toolbox to instantiate the correct model.
4. **Configure Parameters:** Update the JSON files in the `configs/default` and `configs/opt_space` directories to include default hyperparameters and hyperparameter search spaces for the new method.
5. **Adjust Training Processes:** If the method requires a unique training procedure, modify the relevant functions in `model/methods/base.py`. Tailor these functions to accommodate any special optimization strategies that the method requires.

By following these steps, researchers can add new algorithms to Talent, adapting it to meet diverse research needs. For detailed examples and additional guidance, refer to the implementation of existing methods in the `model/methods` directory.

For contributors who wish to contribute to Talent, please refer to our contribution guidelines[2] for more information on how to submit your contributions. We welcome and appreciate all forms of contributions to the project.

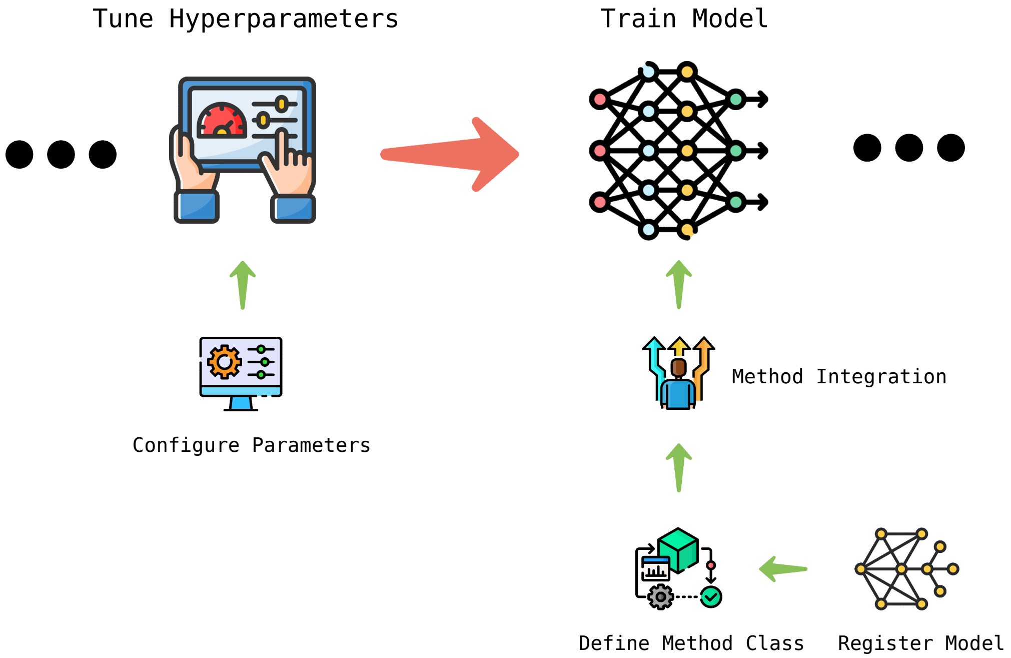


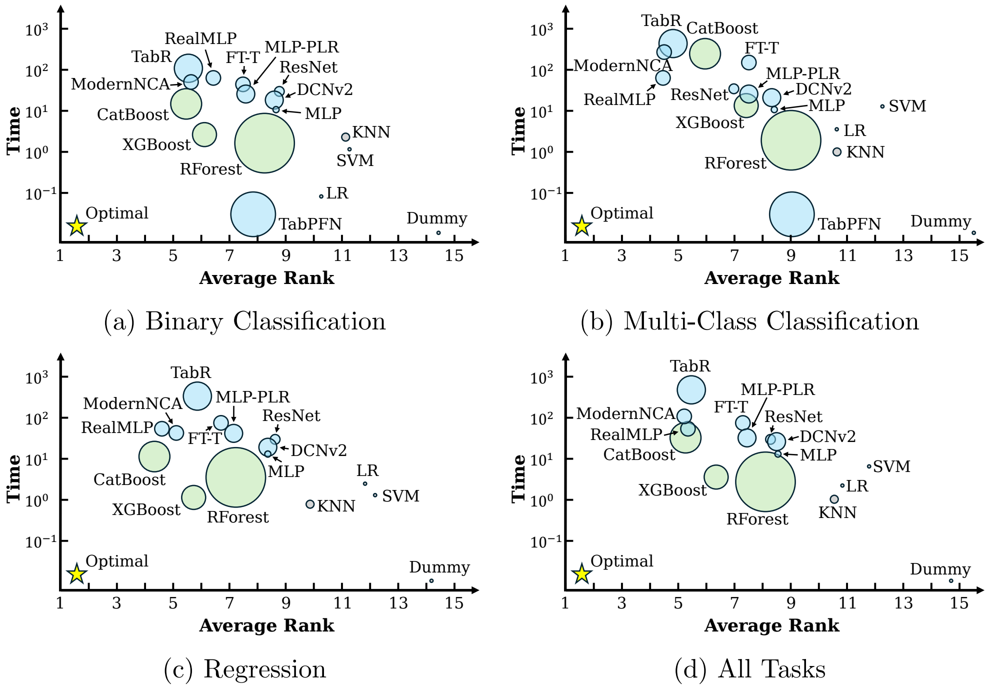
Figure 2: Workflow for Adding a New Method to Talent.

Appendix D. Preliminary Experiments

We provide comprehensive evaluations of classical and deep tabular methods based on our toolbox in a fair manner, as shown in Figure 3. The benchmark covers 300 tabular datasets Ye et al. (2024a) drawn from diverse domains such as finance, education, and physics, encompassing binary classification, multi-class classification, and regression tasks. These datasets exhibit substantial variability in both the number of samples and the number of features, ensuring a broad assessment across different data characteristics. The detailed statistics are provided in Figure 4. The evaluations cover three tabular prediction tasks: binary classification, multi-class classification, and regression, with each subfigure representing a different task type. The datasets are available at Google Drive.

We use accuracy and RMSE as the metrics for classification and regression, respectively. To calibrate the metrics, we choose the average performance rank to compare all methods, where a lower rank indicates better performance, following Sheskin (2003). Efficiency is calculated by the average training time in seconds, with lower values denoting better time efficiency. The model size is visually indicated by the radius of the circles, offering a quick glance at the trade-off between model complexity and performance.

[2] `https://github.com/LAMDA-Tabular/TALENT/blob/main/CONTRIBUTING.md`



(a) Binary Classification

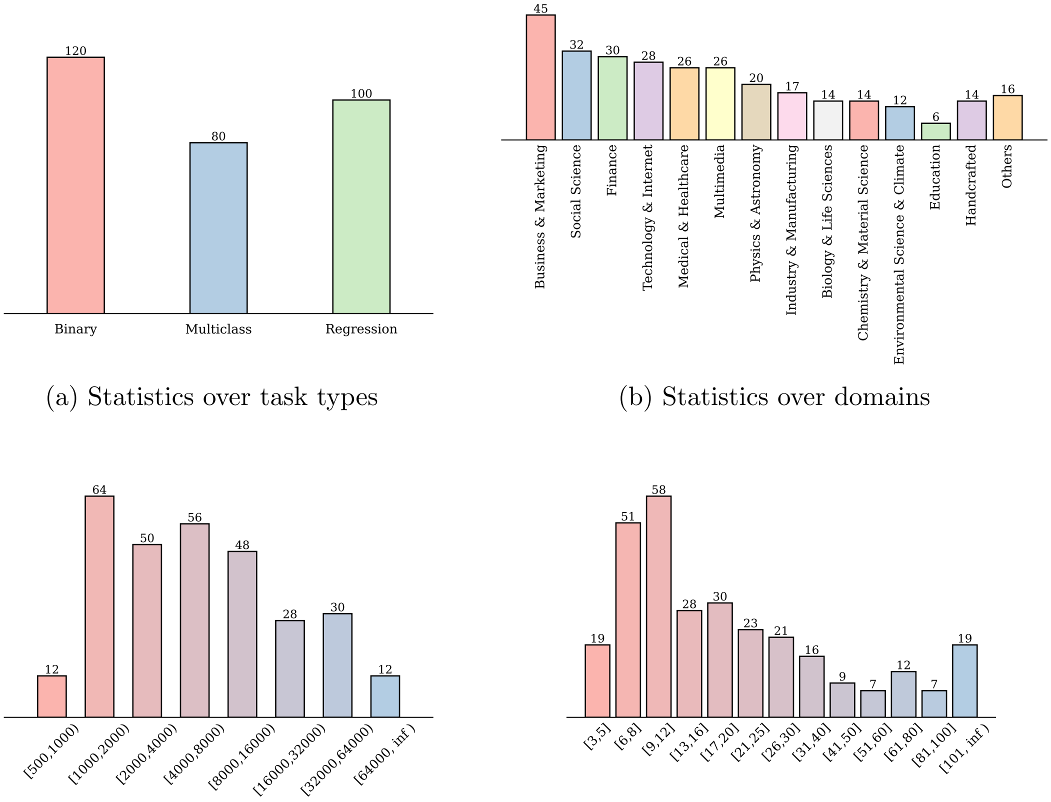
(b) Multi-Class Classification

(c) Regression

(d) All Tasks

Figure 3: Performance-Efficiency-Size comparison of representative tabular methods on our toolbox for (a) binary classification, (b) multi-class classification, (c) regression tasks, and (d) all task types. The performance is measured by the average rank of all methods (lower is better). We also consider the **dummy** baseline, which outputs the label of the major class and the average labels for classification and regression tasks, respectively.

From the comparison, we observe that CatBoost achieves the best average rank in most classification and regression tasks, aligning with findings in McElfresh et al. (2023). Among all deep tabular methods, ModernNCA and RealMLP perform best in most cases while maintaining acceptable training costs. These visualizations serve as an effective tool for quickly and fairly assessing the strengths and weaknesses of various tabular methods across different task types, enabling researchers and practitioners to make informed decisions when selecting suitable modeling techniques for their specific needs.



(a) Statistics over task types

(b) Statistics over domains

(c) Statistics over numbers of samples

(d) Statistics over numbers of features

Figure 4: Statistics of the benchmark datasets. (a) Distribution of datasets across task types: binary classification, multi-class classification, and regression. (b) Distribution across application domains such as finance, education, and physics. (c) Distribution of datasets by the number of samples. (d) Distribution of datasets by the number of features. These statistics highlight the diversity of the benchmark in terms of task type, application domain, and data scale.