Communications for Statistical Applications and Methods
2023, Vol. 30, No. 2, 215–226

https://doi.org/10.29220/CSAM.2023.30.2.215
Print ISSN 2287-7843 / Online ISSN 2383-4757

Recent deep learning methods for tabular data

Yejin Hwang[a], Jongwoo Song[1,a]

[a]Department of Statistics, Ewha Womans University, Korea

Abstract

Deep learning has made great strides in the field of unstructured data such as text, images, and audio. However, in the case of tabular data analysis, machine learning algorithms such as ensemble methods are still better than deep learning. To keep up with the performance of machine learning algorithms with good predictive power, several deep learning methods for tabular data have been proposed recently. In this paper, we review the latest deep learning models for tabular data and compare the performances of these models using several datasets. In addition, we also compare the latest boosting methods to these deep learning methods and suggest the guidelines to the users, who analyze tabular datasets. In regression, machine learning methods are better than deep learning methods. But for the classification problems, deep learning methods perform better than the machine learning methods in some cases.

Keywords: deep learning, tabular data, regression, classification

1. Introduction

A great advance in deep learning has been successfully made with good performance in problems dealing with unstructured data such as text, image, and audio data. However, in the case of predicting tabular data, deep learning is not yet performing as well as unstructured data.

Although it is not clear why deep learning methods perform not as well as the latest boosting methods, we believe that convolutional neural network (CNN) (Krizhevsky *et al.*, 2012) and recurrent neural network (RNN) (Sherstinsky, 2021) perform well for some specific type of data because of the following reasons. CNN identifies the characteristics of image data through a convolution layer that extracts local features of the image and a pooling layer that reduces the dimension. In RNN, the output of the previous step is used as the input of the current step. So RNN is mainly used for sequential data such as text or audio data. Meanwhile, tabular data has a structure in the form of a table with rows and columns. Each row corresponds to each observation and each column corresponds to a variable or feature. However, there is no deep learning method known to have a structure that can capture the characteristics of the tabular data.

Currently, the State-of-the-art model in predicting tabular data is often the ensemble model based on the gradient-boosted decision tree (GBDT) (Friedman, 2001) such as XGBoost (Chen and Guestrin, 2016), CatBoost (Prokhorenkova *et al.*, 2017), LightGBM (Ke *et al.*, 2017). These models give good performance in both regression and classification problems. Also, tree-based ensemble models give feature importance value so we can identify which variables are important in prediction. In terms of predicting tabular data, these GBDT-based models generally outperform deep learning methods. According to the XGBoost and LightGBM official GitHub page (Chen and Guestrin, 2016; Microsoft,

[1] Corresponding author: Department of Statistics, Ewha University, 52 Ewhayeodae-gil, Seodaemun-gu, Seoul 03760, Korea. E-mail: josong@ewha.ac.kr

Published 31 March 2023 / journal homepage: http://csam.or.kr

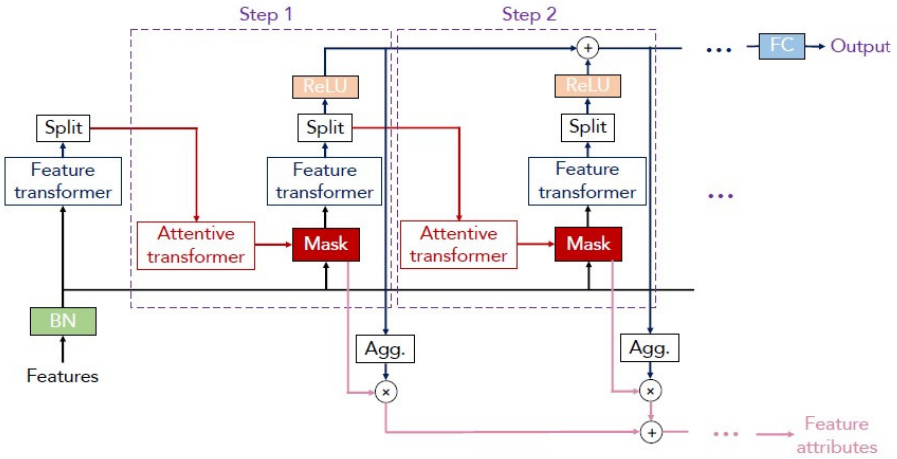

Figure 1: *TabNet encoder architecture.*

2020), XGBoost and LightGBM took a top tier in several Kaggle competitions. Also, XGBoost has been discussed a lot more on Kaggle compared to deep learning (Bansal, 2018).

To compete with machine learning algorithms, many deep learning methods for tabular data are continuously proposed these days. TabNet (Arik and Pfister, 2020) is a deep neural network model that reflects the feature selection characteristics of decision trees in a neural network. It uses a masking structure within the encoder to increase the influence of meaningful variables and reduce the influence of variables without a significant effect on learning. NODE (Popov *et al.*, 2019) applied the ensemble method of oblivious decision tree to the neural network using differentiable trees with Entmax function (Peters *et al.*, 2019). Grownet (Badirli *et al.*, 2020) uses a structure of gradient boosting by connecting multiple shallow trees for neural networks. AutoInt (Song *et al.*, 2019) transforms high-dimensional data into low-dimensional space by using an embedding layer to reduce data sparsity. SAINT (Somepalli *et al.*, 2021) uses two different attention layers to learn the interaction of variables and the interaction of samples.

These new deep learning methods for tabular data are not as widely known as the deep learning methods of image and text data. We would like to explain the concepts of these new deep learning methods briefly and compare their performances using various datasets. We will also compare the performances of these deep learning methods to the latest boosting algorithms since it is well known that boosting algorithms are known to perform better for tabular data.

This paper is organized as follows. We introduce and explain the deep learning methods for tabular data in Chapter 2. We will compare the performances of these methods with various datasets in Chapter 3. Chapter 4 is for the conclusion.

2. Deep learning methods for tabular data

2.1. TabNet

In the decision tree, the algorithm selects the splitting point in each depth that minimizes the impurity of the model to split the feature space. TabNet (Arik and Pfister, 2020) is a deep neural network model that reflects the feature selection of the decision tree.

In the encoder architecture in TabNet, as shown in Figure 1 (Arik and Pfister, 2020), the model

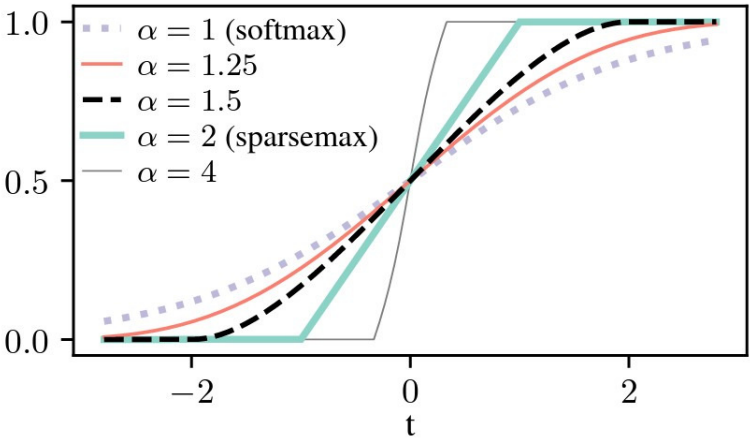

Figure 2: *Entmax function.*

creates a masking layer through a feature transformer and an attentive transformer. The attentive transformer uses the sparsemax function (Martins and Astudillo, 2016) to create a masking layer for feature selection. The shape of the sparsemax function is similar to that of softmax, but both ends of the distribution have higher sparsity than softmax. Therefore, the selected variable is used as it is for learning, and the unselected variable is not reflected in the model. TabNet selects variables for each step in the model, and each selection is made sequentially. Prior scales, the parameter in the attentive transformer, determines the reuse probability of the variable at every step. Also, the model gives the visualization of feature importance for each step of learning. In this way, it is possible to interpret which variables are importantly used in each prediction step. Since the model is a form of instance-wise feature selection that selects meaningful variables for each sample, we can also check the importance of each sample.

TabNet has encoder-decoder architecture. The goal of this structure is to output the same result as the input value, so the input value becomes the correct answer for the model. It is possible to impute missing values in the dataset through the self-supervised learning of the encoder-decoder structure. Therefore, it doesn't need to handle the missing value in the data preprocessing step.

2.2. NODE

NODE (neural oblivious decision ensembles) (Popov *et al.*, 2019) has an ensemble structure of differentiable oblivious decision trees. An oblivious decision tree (Kohavi, 1994) is a constrained tree that uses the same splitting criterion in all nodes of the same depth.

To find a splitting point, the decision tree uses the greedy method that chooses the best splitting point in each depth of the tree. Since this method is greedy local optimization, end-to-end optimization is not possible in the decision tree. To create a differentiable tree, NODE uses the entmax function (Peters *et al.*, 2019) instead of the greedy method. Entmax is a generalization of softmax and sparsmax functions that has a range from 0 to 1. Depending on the parameter α, the sparsity at both ends of the distribution varies. Figure 2 (Peters *et al.*, 2019) shows the distribution of the entmax function depending on α. This entmax function is differentiable, so NODE can use end-to-end optimization with oblivious decision trees (Lou and Obukhov, 2017).

In the decision tree, the output is one of the response values of 2^d terminal nodes. To make the

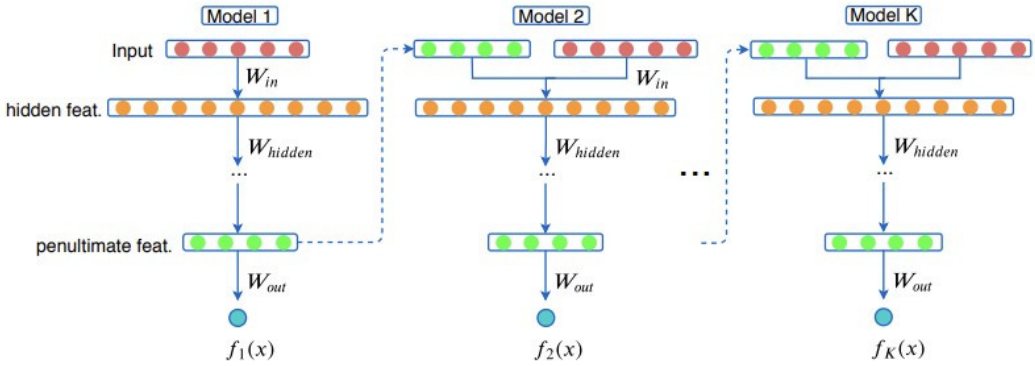

Figure 3: *GrowNet architecture.*

process of calculating the tree output differentiable, NODE determines the tree output through the linear combination of the response tensor and the weights calculated by using the entmax function. Response tensor is a learning parameter of NODE, which is initially set to an initial value that follows a specific distribution and then updated while training the model.

2.3. GrowNet

GBDT trains many shallow trees (weak learners) sequentially and their ensemble result shows good performance in prediction. GrowNet (Badirli *et al.*, 2020) implements this GBDT structure in a neural network.

In GrowNet, the shallow network that has one or two hidden layers acts as a weak learner so the model learns multiple shallow networks. Networks are connected by concatenating the output of the penultimate layer of one network with the input layer of the next network, so the following network uses the output of the penultimate layer as input values. Figure 3 (Badirli *et al.*, 2020) shows this architecture.

The final output of the model is a weighted sum of scores from all shallow neural networks with boosting rate α_k, the weight parameter of each weak learner's output. In GBDT, the boosting rate (learning rate) is fixed for all weak learner outputs. In contrast, GrowNet updates the boosting rate for each network through back-propagation in the corrective step. The back-propagation process also updates the parameters of each shallow network.

2.4. AutoInt

AutoInt (Song *et al.*, 2019) is a model designed to learn feature interactions between explanatory variables. This model is created for click-through rate (CTR) prediction, which predicts the probability of clicking on an item when user information and item information are given. In this case, the dimension of the explanatory variable is usually very high and variables are mostly sparse, so it is easy to be overfitted. Therefore, AutoInt uses an embedding layer to express high-dimensional explanatory variable vectors in low-dimensional continuous space.

Figure 4 (Song *et al.*, 2019) shows the overall architecture of AutoInt. In the embedding layer, all variables including numeric variables are converted into a specific low-dimensional vector. After

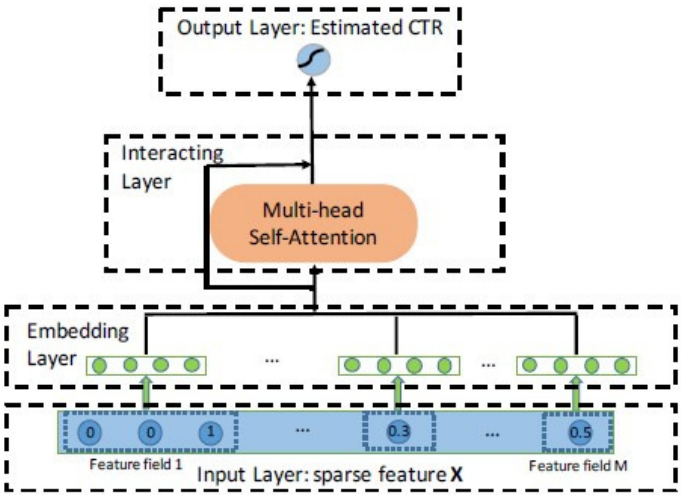

Figure 4: *AutoInt architecture.*

that, using a multi-head self-attention mechanism (Vaswani *et al.*, 2017) in the interacting layer, it finds automatically a combination of variables that are helpful for prediction. At each interacting layer, the attention mechanism combines the variables, and then a multi-head mechanism evaluates the variable combinations created. The multi-head self-attention mechanism shows good performance in modeling complex relationships.

2.5. SAINT

SAINT (Somepalli *et al.*, 2021) is a method to learn both the interaction between columns and samples. SAINT includes an embedding layer and an attention layer like AutoInt. In AutoInt, the multi-head self-attention mechanism (Vaswani *et al.*, 2017) calculates attention about variables to learn the relationship between variables. SAINT also calculates the interaction between variables in the same way as AutoInt. The next step is multi-head intersample attention which calculates the attention between samples to express the relationship between samples. Figure 5 (Somepalli *et al.*, 2021) shows the saint architecture used in the model.

SAINT also has self-supervised contrastive pre-training to improve model performance. Contrastive learning is a pretext task that uses data augmentation. This method learns to minimize the distance between two augmented data from the same sample, and maximize the distance of augmented data from different samples. This method is mainly used to analyze image or text data, but SAINT can be used for tabular data because it uses cutmix (Yun *et al.*, 2019) and mixup (Zhang *et al.*, 2017) as augmentation methods. When there is one data point consisting of n variables, cutmix erases some values of n variables and fills in the blanks with values from another sample. Mixup creates new data by mixing two data points in a certain ratio.

3. Performance comparison

In this chapter, we compare the deep learning methods for tabular data introduced in Chapter 2 using various datasets. For all deep learning methods, we use TensorFlow or PyTorch module with Nvidia

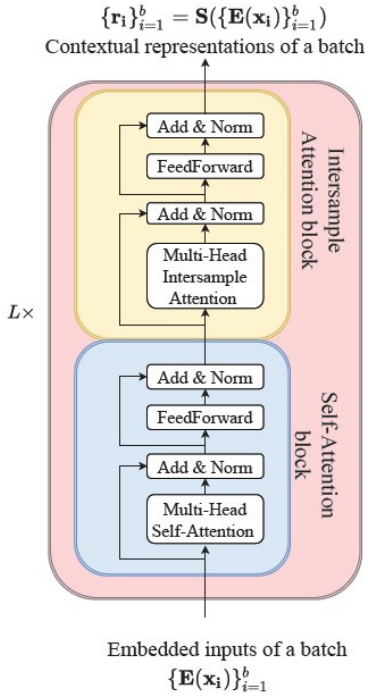

Figure 5: *Two attention blocks in SAINT architecture.*

GTX 2080Ti GPU.

In addition to the deep learning algorithms, we compare the performance of three GBDT-based machine learning algorithms. These three GBDT-based models are as follows. XGBoost (Chen and Guestrin, 2016) is a gradient boosting method with several factors added for effectiveness and scalability. CatBoost (Prokhorenkova *et al.*, 2017) is a model with a method of handling categorical explanatory variables based on GBDT. LightGBM (Ke *et al.*, 2017) is a fast and efficient algorithm with elements to reduce computation time.

3.1. Datasets

In this paper, we compare the performances of models using 10 datasets: California housing prices (Pace *et al.*, 1997, **California**), New York City airbnb open data (Dgomonov, 2019, **airbnb**), bike sharing dataset (Fanaee *et al.*, 2013, **bike**), Beijing PM2.5 data (Liang *et al.*, 2015, **Beijing**), Belarus used cars prices (Pasedko, 2019, **Belarus**), airlines customer satisfaction (Jana, 2020, **airlines**), German credit data (Hofmann, 1994, **German**), Online shoppers purchasing intention (Sakar *et al.*, 2018, **online**), mobile price classification (Sharma, 2017, **mobile**), dry bean dataset (Koklu and Ozkan, 2020, **dry bean**). The first five datasets are regression problems and the other five datasets are classification problems. Table 1 contains the information of each dataset.

In the case of the classification problem, airlines customer satisfaction, German credit data, and online shoppers purchasing intention have binary target variables. Mobile price classification has multiclass target variables with 4 types, and also dry bean dataset has multiclass target variables with 7 types. Figure 6 shows the distribution of the target variable in each dataset. Among the deep learning

Table 1: Description of experiment datasets

Dataset	# of rows	# of columns	Problem	Source
California	19475	10	Regression	Kaggle
Airbnb	48895	16	Regression	Kaggle
Bike	17389	16	Regression	UCI repository
Beijing	43824	13	Regression	UCI repository
Belarus	56244	12	Regression	Kaggle
Airlines	129881	23	Classification	Kaggle
German	1000	20	Classification	UCI repository
Online	12330	18	Classification	UCI repository
Mobile	2000	21	Classification	Kaggle
Dry bean	13611	17	Classification	UCI repository

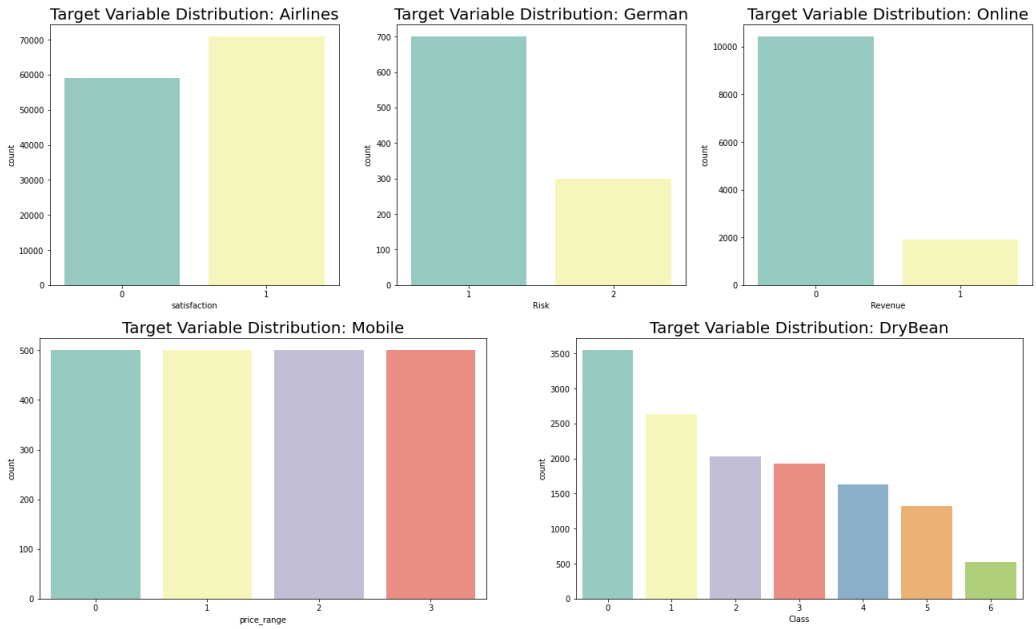

Figure 6: *Target variable distribution for classification problems.*

methods we use, Grownet and AutoInt do not provide a code for predicting a multiclass classification. Therefore, the mobile price classification and dry bean dataset compare the results of the models except for Grownet and AutoInt.

3.2. Preprocessing

Before training the model, datasets have to be transformed into a suitable form for analysis. Therefore, preprocessing was performed for all datasets. We excluded rows with NA values from the analysis, and categorical variables were converted into numeric values using one hot encoding except for AutoInt. In the case of AutoInt, label encoding must be applied to categorical variables to handle categorical variables within the model. So we only used label encoding for AutoInt.

In addition to the basic preprocessing step, datasets that require additional preprocessing steps were transformed as follows.

Table 2: Result of modeling - RMSE

	RMSE				
	California	Airbnb	Bike	Beijing	Belarus
TabNet	48179.356	67.817	77.073	49.620	2900.251
NODE	54502.934	71.125	77.091	52.736	4853.268
GrowNet	49668.250	67.286	35.979	41.683	2862.707
AutoInt	50409.035	67.506	39.688	41.757	2849.807
SAINT	48790.454	67.821	36.146	36.176	2852.220
XGBoost	43207.742	65.387	35.615	35.050	2740.192
CatBoost	41875.720	65.621	**33.977**	38.169	2710.506
LightGBM	**41448.663**	**65.283**	35.695	**34.744**	**2708.251**

Table 3: Result of modeling - Pseudo R^2

	Pseudo R^2				
	California	Airbnb	Bike	Beijing	Belarus
TabNet	0.762	0.476	0.821	0.710	0.868
NODE	0.700	0.424	0.820	0.677	0.631
GrowNet	0.740	0.485	0.961	0.798	0.872
AutoInt	0.740	0.481	0.952	0.798	0.873
SAINT	0.756	0.476	0.961	0.848	0.872
XGBoost	0.809	0.514	0.962	0.857	0.882
CatBoost	0.820	0.501	**0.965**	0.831	**0.885**
LightGBM	**0.824**	**0.515**	0.962	**0.860**	**0.885**

- California housing prices: Among the target variable *median_house_value*, values greater than 500,000 are recorded as 500,001. We judged this to be inaccurate data, so the row with a *median_house_value* of 500,001 was excluded from the analysis.
- New York City airbnb open data: *Price*, the target variable of airbnb data, is the price per night of airbnb, and there was data with a price of 0. We considered this as an outlier and excluded data with a price of 0 from the analysis. Also, the *last_review* variable refers to the date on which the most recent review was written, but a date format variable cannot be used for analysis. Therefore, we transform this variable into a variable *last_review_days* that counts the number of days from 2019/12/31, the date of data collection.
- Bike sharing dataset: The following columns that are not used for analysis were dropped - *instant* (record index), *dteday* (date), *casual* (count of casual users: high correlation with target variable), *registered* (count of registered users: high correlation with target variable).
- Belarus used cars prices: The *make* and *model* variables have 51 and 461 levels respectively. When we apply one hot encoding to the data including these variables, the data becomes very sparse. So we excluded *make* and *model* variables from the analysis.

3.3. Results

For every dataset, we randomly selected 80% of the data as a train set and 20% as a test set. We predicted the test set with each trained model, and compared models based on RMSE and Pseudo R^2 for regression problems, accuracy and f1 score for classification problems. We also compared computation times for all problems. The tuning parameters used in each model and be found in https://home.ewha.ac.kr/~josong/DL.TabData

Table 4: Result of modeling - Computation time

	Computation time (sec)				
	California	Airbnb	Bike	Beijing	Belarus
TabNet	926.78	2845.12	1328.78	1777.33	2236.38
NODE	146.00	171.00	376.00	595.00	253.00
GrowNet	285.65	141.01	440.23	843.81	619.46
AutoInt	569.52	67.506	164.28	323.38	467.86
SAINT	265.24	338.28	364.46	855.1	590.17
XGBoost	**1.15**	18.654	8.05	11.01	4.90
CatBoost	3.86	**2.66**	2.72	4.06	**4.08**
LightGBM	3.79	4.81	**1.95**	**2.89**	5.01

3.3.1. Regression problems

Tables 2–4 are prediction results for regression problems. As shown in the tables, one of the three machine learning models had the smallest RMSE and the highest R^2 in all datasets. The results of the machine learning methods did not differ significantly. Computation time was also much faster with machine learning methods. For all datasets except Beijing, the three machine learning methods had lower RMSE and higher R^2 than all the deep learning methods. In the case of Beijing, the RMSE of SAINT was lower than the value of CatBoost.

Among the deep learning methods, the model with the best performance depends on the dataset. For the California dataset, TabNet's RMSE was the lowest among deep learning models. The difference between R^2 of TabNet and the best model (LightGBM) was 0.062. But the computation time was more than 200 times faster in LightGBM than in TabNet. For the airbnb and bike dataset, the model with the lowest RMSE among deep learning methods was GrowNet. In the case of Beijing PM, the RMSE of SAINT was the lowest among the five deep learning methods, and in Belarus, AutoInt was the best. R^2 of NODE was relatively low compared to other methods. For computation time, AutoInt was the fastest on four of five datasets. TabNet took much more computation time than other methods on all datasets.

3.3.2. Classification problems

Tables 5–7 show prediction results for classification problems. For the German and online datasets, machine learning methods had the highest test accuracy. However, in the other three datasets, one of the deep learning methods was the best model. In the case of computation time, machine learning methods were much faster than deep learning methods.

For the airlines dataset, SAINT obtained the highest test accuracy. There was not much difference in accuracy for the airlines dataset between SAINT, AutoInt, TabNet, and the three machine learning methods. However, the computation time was very different. TabNet took over an hour, while LightGBM trained the model in about 4 seconds. SAINT also obtained the highest accuracy in the dry bean dataset. In the German dataset, XGBoost and CatBoost had the highest accuracy. The difference between the accuracy of the best model and SAINT, the lowest accuracy among deep learning models, was 0.075. For the online dataset, XGBoost and LightGBM obtained the highest accuracy. The test accuracy of GrowNet and SAINT was higher than that of CatBoost. For NODE, the accuracy was 0.842 but the F1 score was 0 because NODE predicted all test data to the same class. In the case of the mobile dataset, NODE obtained the highest test accuracy. In addition, the accuracy of TabNet was higher than that of XGBoost and LightGBM. But the computation time of machine learning was still much faster than deep learning methods.

Table 5: Result of modeling - Accuracy

	Accuracy				
	Airlines	German	Online	Mobile	Dry bean
TabNet	0.957	0.740	0.862	0.930	0.928
NODE	0.951	0.745	0.842	**0.943**	0.890
GrowNet	0.943	0.750	0.900	-	-
AutoInt	0.959	0.745	0.895	-	-
SAINT	**0.961**	0.720	0.903	0.908	**0.933**
XGBoost	0.959	**0.795**	**0.908**	0.928	0.925
CatBoost	0.960	**0.795**	0.896	0.935	0.926
LightGBM	0.960	0.780	**0.908**	0.922	0.929

Table 6: Result of modeling - F1 score

	F1 score				
	Airlines	German	Online	Mobile	Dry bean
TabNet	0.962	0.823	0.614	0.926	0.939
NODE	0.955	0.825	0.000	**0.940**	0.896
GrowNet	0.947	0.808	0.659	-	-
AutoInt	0.962	0.831	0.644	-	-
SAINT	**0.964**	0.806	0.670	0.904	**0.944**
XGBoost	0.962	**0.866**	**0.679**	0.924	0.936
CatBoost	0.963	0.861	0.644	0.931	0.937
LightGBM	0.963	0.856	0.678	0.919	0.939

Table 7: Result of modeling - Computation time

	Computation time (sec)				
	Airlines	German	Online	Mobile	Dry bean
TabNet	4999.62	167.84	645.81	467.12	649.64
NODE	487	98	12	32	347
GrowNet	311.42	90.91	68.88	-	-
AutoInt	436.73	34.96	31.71	-	-
SAINT	243.403	46.22	26.99	48.69	258.21
XGBoost	13.55	1.7	2.79	2.88	17.31
CatBoost	12.17	3.21	1.32	1.332	3.81
LightGBM	**3.99**	**0.6**	**0.54**	**0.94**	**3.69**

4. Conclusion

Deep learning is well known for its good performance in the field of image and text data. However, many applications still have tabular data for analysis. So, we liked to know the performance of deep learning-based methods for tabular data. We introduced deep learning methods designed for analyzing tabular data and compared the prediction performances of these models using multiple datasets. In regression problems, machine learning methods are still superior to deep learning methods. However, in the case of classification problems, deep learning methods achieved higher accuracy than machine learning methods in three out of five datasets. In the case of computation time, deep learning methods take much longer time than machine learning methods because deep learning methods have more complex structures and more parameters in the model.

References

Arik SO and Pfister T (2020). Tabnet: Attentive interpretable tabular learning, *Proceedings of the AAAI Conference on Artificial Intelligence*, **35**, 6679–6687.

Bansal S (2018). Historical data science trends on kaggle, Available from: https://www.kaggle.com/code/shivamb/data-science-trends-on-kaggle/notebook

Badirli S, Liu X, Xing Z, Bhowmik A, Doan K, and Keerthi S (2020). Gradient Boosting Neural Networks: GrowNet, Available from: arXiv:2002.07971v2

Chen T and Guestrin C (2016). XGBoost: A scalable tree boosting system, Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, **2016**, 785–794.

Chen T and Guestrin C (2016). dmlc/xgboost/demo, Available from: https://github.com/dmlc/ xgboost/tree/master/demo

Dgomonov (2019). New York City Airbnb Open Data, Available from: https://www.kaggle.com/datasets/dgomonov/new-york-city-airbnb-open-data

Friedman JH (2001). Greedy function approximation: A gradient boosting machine, *The Annals of Statistics*, **29**, 1189–1232.

Hadi Fanaee-T and Gama J (2013). Event labeling combining ensemble detectors and background knowledge, *Progress in Artificial Intelligence*, **2**, 1–15.

Hofmann H (1994). Statlog (German Credit Data) Data Set [https://archive.ics.uci.edu/ml/datasets/Statlog+%28German+Credit+Data%29]. Irvine, CA: University of California, School of Information and Computer Science.

Jana S (2020). Airlines customer satisfaction, Available from: https://www.kaggle.com/datasets/sjleshrac/airlines-customer-satisfaction?select=Invistico_Airline.csv

Krizhevsky A, Sutskever I, and Hinton GE (2012). ImageNet classification with deep convolutional neural networks, *Advances in Neural Information Processing Systems*, **2**, 1106–1114.

Ke G, Meng Q, Finley T, Wang T, Chen W, Ma W, Ye Q, and Liu T (2017). LightGBM: A Highly Efficient Gradient Boosting Decision Tree, NIPS.

Kohavi R (1994). Bottom-up induction of oblivious read-once decision graphs: Strengths and limitations, *AAAI*

Koklu M and Ozkan IA (2020). Multiclass classification of dry beans using computer vision and machine learning techniques, *Computers and Electronics in Agriculture*, **174**, 105507.

Liang X, Zou T, Guo B, Li S, Zhang H, Zhang S, Huang H, and Chen SX (2015). Assessing Beijing's PM2.5 pollution: Severity, weather impact, APEC and winter heating, *Proceedings: Mathematical, Physical and Engineering Sciences*, **471**, 1–20.

Lou Y and Obukhov M (2017). BDT: Gradient boosted decision tables for high accuracy and scoring efficiency, *Proceedings of the 23rd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, 1893–1901.

Microsoft (2020). Microsoft/LightGBM/examples, Available from: https://github.com/microsoft/LightGBM/tree/master/examples

Martins A and Astudillo R (2016). From softmax to sparsemax: A sparse model of attention and multi-label classification, *Proceedings of the 33rd International Conference on International Conference on Machine Learning*, **48**, 1614–1623.

Pace RK and Barry R (1997). Sparse spatial autoregressions, *Statistics & Probability Letters*, **33**, 291–297.

Peters B, Niculae V, and Matrins A (2019). Sparse sequence-to-sequence models, In *Proceedings*

of the 57th Annual Meeting of the Association for Computational Linguistics, Florence, Italy, 1504–1519.

Pasedko S (2019). Belarus Used Cars Prices, Available from: https://www.kaggle.com/datasets/slavapasedko/belarus-used-cars-prices

Popov S, Morozov S, and Babenko A (2019). Neural Oblivious Decision Ensembles for Deep Learning on Tabular Data, Available from: arXiv:1909.06312v2

Prokhorenkova L, Gusev G, Vorobev A, Dorogush A, and Gulin A (2017). CatBoost: unbiased boosting with categorical features, NeurIPS.

Sakar CO, Polat SO, Katircioglu M, and Yomi K (2018). Real-time prediction of online shoppers' purchasing intention using multilayer perceptron and LSTM recurrent neural networks, *Neural Computing & Applications*, **31**, 6893–6908.

Sharma A (2017). Mobile price classification, Available from: https://www.kaggle.com/datasets/iabhishekofficial/mobile-price-classification

Sherstinsky A (2021). Fundamentals of recurrent neural network (RNN) and long short-term memory (LSTM) network, *Physica D: Nonlinear Phenomena*, **404**, 132306.

Song T, Shi C, Xiao Z, Duan Z, Xu Y, Zhang M, and Tang J (2019). AutoInt: Automatic feature interaction learning via self-attentive neural networks, Proceedings of the 28th ACM International Conference on Information and Knowledge Management(CIKM), 1161–1170.

Somepalli G, Goldblum M, Schwarzschild A, Bruss C, and Goldstein T (2021). SAINT: Improved Neural Networks for Tabular Data via Row Attention and Contrastive Pre-Training, Available from: arXiv:2106.01342

Vaswani A, Shazzer N, Parmar N, Uszkoreit J, Jones L, Gomez A, Kaiser L, and Polosukhin I (2017). Attention is all you need, *NIPS*, **2017**.

Yun S, Han D, Oh S, Chun S, Choe J, and Yoo Y (2019). Cutmix: Regularization strategy to train strong classifiers with localizable features, In *Proceedings of IEEE/CVF International Conference on Computer Vision (ICCV)*, Seoul, Korea (South), 6022–6031.

Zhang H, Cisse M, Dauphin Y, and Lopez-Paz D (2017). Mixup: Beyond empirical risk minimization, International Conference on Learning Representations(ICLR).

Received October 6, 2022; Revised November 15, 2022; Accepted December 19, 2022