A Additional results and discussions

A.1 RMSE@K discussion on univariate regression problems with mixed-type data

RMSE@K Introduction For point prediction, we introduce a new metric called root mean squared error at K (RMSE@K). It is a version of the Root Mean Squares Error (RMSE) metric that takes into account multiple predictions from the model. The formula for the metrics is as follows:

$$\text{RMSE@K}(\mathbf{y}, \hat{\mathbf{y}}) = \sqrt{\frac{1}{N}\sum_{i=1}^{N} \min_{k=1,\ldots,K} (y_i - \hat{y}_{i,k})^2}$$

It is suited for uni- and multi-variate regression problems where multiple-point predictions are possible. Such a situation origins from probabilistic regression where a model can produce multimodal distributions. In such situations, it's not possible to provide one point estimate, and often for practical reasons, analysis of the whole probabilistic distribution is not feasible. The possible solution is to provide multiple point estimates (usually up to 3 as more modalities rarely occur in real-world settings) for a particular observation.

Detailed results analysis of RMSE@K on univariate regression problems with mixed-type data In the experiments, we analyze three approaches for obtaining point estimates from TreeFlow: Samples averaging (Avg) - the simple average of samples, RMSE@1 - usage of the most probable sample, standard RMSE, RMSE@2 - usage of the two most probable samples

In tab. 2 we can observe that usually the best results are obtained in the two-point prediction approach, then the samples averaging, and at the end selection of the most probable sample. Such results could be easily explained by considering bimodal distribution that has two almost the same probable modalities. In the first scenario, we are almost always wrong as we do not predict the exact modalities but something between them. In the second scenario, we predict only one modality thus in approximately half of the examples we are right, and in the latter half wrong. In the last scenario, we always select both modalities and check which one is the correct one. Such an approach simulates the real-world scenario where in case of two predictions some end user would check the results and select the correct one.

A.2 Statistical significance

Additionally, we have performed the Wilcoxon Signed Rank test with the null hypothesis, that there is no difference between the models' performance, i.e., between TreeFlow and CatBoost, and between TreeFlow and PGBM. We used NLL results from univariate regression problems on mixed-type and numeric data (overall 16 datasets). In the first scenario, we obtained a p-value equal to 0.01, and in the second scenario 0.02. In both cases it is less than our significance level $\alpha = 0.05$, thus we reject the null hypothesis.

B Datasets

B.1 Univariate regression on synthetic data

In order to properly evaluate our method, we need to have a dataset where the true probability distribution is known. It is almost impossible to obtain such a dataset from a real-world scenario; thus, we generated synthetic data. The samples from the dataset were generated using the following probabilistic model:

$$\begin{aligned}
P(Y|X_1 = 0, X_2 = 0) &= \mathcal{N}(Y|\mu = 0, \sigma = 1) \\
P(Y|X_1 = 0, X_2 = 1) &= \mathcal{E}(Y|\lambda = \tfrac{1}{3}) \\
P(Y|X_1 = 1, X_2 = 0) &= \tfrac{1}{2}\mathcal{N}(Y|\mu = -10, \sigma = 1) \\
&\quad + \tfrac{1}{2}\mathcal{N}(Y|\mu = 10, \sigma = 1) \\
P(Y|X_1 = 1, X_2 = 1) &= \Gamma(Y|k = 7.5, \theta = 1.0)
\end{aligned} \tag{6}$$

The justification for the following probability distribution is as follows. We selected normal distribution to validate if the methods can fit the simplest scenario, exponential distribution to check the fit to heavy-tailed distributions, a mixture of Gaussians for multimodality, and Gamma to check behavior for distributions close to Gaussian distribution.

During the experimental phase for training purposes, we sampled 5,000 observations from each distribution, resulting in a 20,000 samples dataset. We also used 1,000 observations from each distribution for the early stopping / best epoch selection process. The final log likelihood was calculated on a dataset constructed from 500,000 samples per distribution.

B.2 Univariate regression on mixed-type data

Extended information about datasets used in univariate regression on mixed-type data experiments is provided in tab. 6. We can observe that these datasets consist of various proportions of categorical to numerical features, and cover a broad range of categorical features cardinality. These properties are easily handled by a Tree-based Feature Extractor component with an underlying CatBoost model. Moreover, for selected datasets, we have used log10 transformation of the target variable, mostly due to high absolute values. The non-linearity of this transformation affects the shape of the distribution but favors the CatBoost and PGBM model. Price distribution is usually heavy-tailed, and log transformation makes it more Gaussian. Even though TreeFlow performed better on these datasets.

Table 6. Extended information about datasets used in univariate regression problems with the mixed-type experiment. Symbols: D_{CAT} - Number of categorical variables, D_{NUM} - number of numerical variables, Max card. - Maximum cardinality number among categorical variables, Log transform - Flag if the logarithm of base 10 was used on the target variable.

Dataset	N	D	D_{CAT}	D_{NUM}	Max card.	Log transform	Link
Avocado	18,249	11	3	8	54	✗	Link[1]
BigMart	8,523	10	6	4	16	✓	Link[2]
Diamonds	53,940	9	3	6	8	✓	Link[3]
Diamonds 2	119,307	7	6	1	10	✓	Link[4]
Laptop	1,303	10	7	3	118	✓	Link[5]
Pak Wheel	76,690	7	4	3	326	✓	Link[6]
Sydney Housing	199,504	6	3	3	685	✓	Link[7]

B.3 Univariate regression on numerical data

Datasets for univariate regression on numerical data experiments were used without any preprocessing. None of the datasets contained missing values. Extended information regarding dataset sizes is provided in tab. 7.

Table 7. Extended information about datasets used in univariate regression problems with numerical data.

Dataset	N	D	CV Splits
Concrete	1030	8	20
Energy	768	8	20
Kin8nm	8192	8	20
Naval	11934	16	20
Power	9568	4	20
Protein	45730	9	5
Wine	1588	11	20
Yacht	308	6	20
Year MSD	515345	90	1

B.4 Multivariate regression

Datasets for multivariate regression experiment were used without any preprocessing except Oceanographic. Here, we multiplied the target value by 100 due to numerical stability. The same operation was used in the reference paper. Moreover, none of the datasets contained missing values. Additional information about dataset sample sizes, dimensionalities of features, and target variables are presented in tab. 8.

Table 8. Extended information about datasets used in multivariate regression problems.

Dataset	N_{train}	N_{test}	D	P
Parkinsons	4,112	1,763	16	2
scm20d	7,173	1,793	61	16
WindTurbine	4,000	1,000	8	6
Energy	57,598	14,400	32	17
usFlight	500,000	200,000	8	2
Oceanographic	414,697	20 CV	9	2

C Implementation details

In this section, we cover the essential information related to implementation details. The code for experiments is available in the Supplementary Materials. After the review process, they will be published publicly on the GitHub repository.

We presented the architecture of TreeFlow model in fig. 1. We have used CatBoost as the Tree-based Feature Extractor, one layer neural network with tanh activation function as Shallow Feature Extractor and Conditional Continuous Normalizing Flow presented in [26] for the Conditional CNF component.

[1] https://www.kaggle.com/neuromusic/avocado-prices
[2] https://www.kaggle.com/yasserh/bigmartsalesdataset
[3] https://www.kaggle.com/shivam2503/diamonds
[4] https://www.kaggle.com/miguelcorraljr/brilliant-diamonds
[5] https://www.kaggle.com/muhammetvarl/laptop-price
[6] https://www.kaggle.com/ebrahimhaquebhatti/75000-used-cars-dataset-with-specifications
[7] https://www.kaggle.com/mihirhalai/sydney-house-prices

In terms of the multipoint estimation, it starts with sampling 1000 observations from the target distribution. In the next step, we use kernel density estimation (KDE) that approximates the probability density function (PDF), and then we run the find_peaks procedure provided by the SciPy Python package. As a side note, TreeFlow provides PDF, however, the function is highly unsmooth and our practical experiments showed that KDE approximation works much better.

In the tab. 9 and tab. 11 we used the following naming convention:

- Depth - maximum depth of the single tree in the CatBoost ensemble;
- N trees - number of trees in the CatBoost ensemble;
- Context dim - dimensionality of the output layer of the Shallow Feature Extractor;
- Hidden dim - dimensionality of the consecutive layers in the dynamic function of the CNF block;
- N blocks - number of CNF blocks;
- N epochs - number of training epochs.

For all experiments purposes, we used a machine with AMD Ryzen 9 5950X 16-Core Processor CPU, 2 NVIDIA GeForce 2080 Ti GPUs, and 64 GB RAM.

C.1 Univariate regression on synthetic data

In this experiment, we were responsible for training both CatBoost and TreeFlow models. CatBoost was trained using default hyperparameters as they are known to work very well out of the box. For TreeFlow we used: Depth: 2, N trees: 100, Context dim: 50, Hidden dims: [50, 10], N blocks: 2, Num epochs: 50.

C.2 Univariate regression on mixed-type data

For the reason of fair comparisons, in this experiment, we used the set of the same hyperparameters for all datasets. We were responsible for training all methods. The default hyperparameters of CatBoost and PGBM were used, and for TreeFlow we used Depth: 4, N trees: 200, Context dim: 128, Hidden dim: [16, 16], N blocks: 2, Num epochs: 150.

C.3 Univariate regression on numerical data

For this experiment, we performed a hyperparameter search. The process consisted of both grid search and manual trials and errors, i.e., we performed an initial grid search to obtain intuitions on the validation dataset and then ran consecutive grid searches with changed ranges of hyperparameters. The final hyperparameter setting is presented in tab. 9. Results in tab. 3 for the reference methods were taken from the reference papers [5, 12] except PGBM that was trained by us with hyperparameters from [24].

Table 9. Hyperparameters used for TreeFlow method in the univariate regression on numerical data experiment. Outer square brackets are equivalent to a grid search among provided values.

Dataset	Tree parameters		Flow parameters			General
	Depth	N trees	Context dim	Hidden dim	N blocks	N epochs
Concrete	1	[300, 500, 750]	[50, 100, 200]	[[100, 100, 50], [200, 100, 100, 50]]	3	100
Energy	[1, 2]	[100, 300]	[40, 100]	[[80, 40], [80, 80, 40], [80, 80, 80, 40]]	3	200
Kin8nm	[1, 2]	[100, 300]	[40, 100]	[[80, 40], [80, 80, 40], [80, 80, 80, 40]]	3	20
Naval	4	[500, 750]	200	[[100, 100, 50], [200, 100, 100, 50]]	4	25
Power	4	500	[100, 200]	[[100, 50], [100, 100, 50], [200, 100, 100, 50]]	[3, 5]	30
Protein	4	750	100	[100, 100, 50]	3	25
Wine	[1, 2]	[100, 300]	[40, 100]	[[80, 40], [80, 80, 40]]	3	200
Yacht	[1, 2]	[300, 500, 750]	[50, 100, 200]	[[100, 100, 50], [200, 100, 100, 50]]	[1, 2]	100
Year MSD	[1, 2]	[100, 300]	[40, 100]	[[80, 40], [80, 80, 40], [80, 80, 80, 40]]	3	3

C.4 Multivariate regression

In this experiment, we trained models with hyperparameter search for both NGBoost and TreeFlow models. For NGBoost models (Independent and Multivariate Gaussian), we performed a grid search on all datasets except Oceanographic, where results work taken from the reference paper [18]. The range of the hyperparameters was inspired from [5]. The parameters are presented in tab. 10. For the TreeFlow method, the same approach was applied with the final hyperparameter space presented in tab. 11.

D Ablation study

In this section, we perform two experiments to analyze the contribution of specific TreeFlow's components to the overall performance. The first experiment discusses the Tree-based Feature Extractor component and the second Shallow Feature Extractor.

Table 10. Hyperparameters used for NGBoost methods in the multivariate regression on experiment. Square brackets are equivalent to a grid search among provided values.

Dataset	Tree parameters			NGBoost parameters
	Max Depth	Max Leaf Nodes	Min Samples Leaf	Num trees
Parkinsons	[5, 10, 15]	[8, 15, 32, 64]	[1, 15, 32]	[100, 300, 500]
Scm20d	15	[8, 15, 32, 64]	[1, 15, 32]	[100, 300, 500]
WindTurbine	[5, 10, 15]	[8, 15, 32, 64]	[1, 15, 32]	[100, 300, 500]
Energy	15	[8, 15, 32, 64]	[1, 15, 32]	[100, 300, 500]
UsFlight	[5, 10, 15]	[8, 15, 32, 64]	[1, 15, 32]	[100, 300, 500]

Table 11. Hyperparameters used for TreeFlow method in the multivariate regression experiment. Outer square brackets are equivalent to grid search among provided values.

Dataset	Tree parameters		Flow parameters			General
	Depth	N trees	Context Dim	Hidden Dim	N blocks	N epochs
Parkinsons	[1, 2]	[100, 300]	[40, 100]	[[80, 40], [80, 40, 40], [80, 80, 80, 40]]	3	500
Scm20d	[1, 2]	[100, 300]	[40, 100]	[[80, 40], [80, 40, 40], [80, 80, 80, 40]]	3	200
WindTurbine	1	[500, 750]	[50, 100]	[[100, 50], [100, 100, 50], [200, 100, 100, 50]]	[3, 5]	150
Energy	[1, 2]	[100, 300]	[40, 100]	[[80, 40], [80, 40, 40], [80, 80, 80, 40]]	3	30
UsFlight	[1, 2]	[100, 300, 500]	[40, 80, 120]	[[80, 40, 40], [80, 80, 80, 40], [200, 100, 100, 50]]	[1, 3, 5]	5
Oceanographic	2	[750, 1000]	[100]	[[50, 50]]	1	30

D.1 Tree-based Feature Extractor

We introduced the Tree-based Feature Extractor component as the tree-based model, and in the experiments, we specifically focused on the CatBoost implementation. Our motivation was to enable our method to deal with categorical variables efficiently. The most common alternative is to use One Hot Encoder, which encodes each possible category as a binary vector of category occurrence.

In this ablation study, we performed an experiment where we replaced CatBoost with One Hot Encoder as a Feature Extractor. In practice, it reduces the model to CNF with an additional MLP layer for the conditioning factor encoding (in this work called Shallow Feature Extractor). The experiment methodology and hyperparameters were the same as in the Univariate regression on mixed-type data experiment. The results of the experiments are presented in tab. 12. We can observe that for almost all of the datasets TreeFlow obtains better or comparable results to CNF and thus, we conclude that the Tree-based Feature Extractor is a crucial component of TreeFlow.

Table 12. Results of the ablation study regarding Tree-based Feature Extractor. In this scenario Tree-based Feature Extractor (CatBoost backbone) in TreeFlow was replaced by the One Hot Encoder (OHE) which results in CNF model. D_{OHE} represents the number of features in the dataset after One Hot Encoding categorical variables.

Dataset	D	D_{OHE}	TreeFlow	CNF (TreeFlow with OHE)
Avocado	11	65	**-0.47 ± 0.03**	-0.27 ± 0.02
BigMart	10	46	-0.08 ± 0.02	**-0.12 ± 0.01**
Diamonds	9	26	**-1.94 ± 0.03**	-1.78 ± 0.03
Diamonds 2	7	37	**-2.14 ± 0.05**	-1.53 ± 0.13
Laptop	10	344	**-0.74 ± 0.13**	-0.70 ± 0.27
Pak Wheel	7	402	**-1.60 ± 0.03**	-1.26 ± 0.02
Sydney Housing	6	697	**-0.66 ± 0.01**	-0.60 ± 0.06

D.2 Shallow Feature Extractor

The next component which we introduced was the Shallow Feature Extractor. Its task was to map high-dimensional binary vectors extracted from forest structures to low-dimensional feature representation. The main goal of that operation was to reduce computational overhead. We present an ablation study where we exclude the Shallow Feature Extractor component and pass the output of the Tree-based Feature Extractor directly to the Conditional CNF component.

We used the same methodology as in the previous ablation study, but we only calculate the training time of one epoch of the model. The experiment was run on a CPU, so the relative speed up is the crucial factor in the comparison. The results are presented in tab. 13. We can observe that TreeFlow was on average 11 times faster than a model without Shallow Feature Extractor.

Concluding, our ablation study showed that Shallow Feature Extractor is a crucial component of the method in terms of computational time performance.

Table 13. Results of the ablation study regarding Shallow Feature Extractor. In this scenario Shallow Feature Extractor (SFE) in TreeFlow was not present. D_{SFE} represented the number of features extracted from the Tree-based Feature Extractor and passed directly to the Conditional CNF component. The calculation time was the one epoch training time and the experiments were conducted on CPU time. TreeFlow on average was 11 times faster than the model without SFE.

DATASET	D	D_{SFE}	TREEFLOW	TREEFLOW WITHOUT SFE	SPEED UP
AVOCADO	11	1600	8.87 ± 0.19 S	113.25 ± 2.01 S	12.7 X
BIGMART	10	1600	4.67 ± 0.13 S	41.37 ± 0.79 S	8.9 X
DIAMONDS	9	1600	25.26 ± 0.33 S	345.28 ± 5.03 S	13.7 X
DIAMONDS 2	7	1600	55.56 ± 0.28 S	756.62 ± 17.91 S	13.6 X
LAPTOP	10	1600	1.90 ± 0.08 S	5.84 ± 0.07 S	3.1 X
PAK WHEEL	7	1600	36.20 ± 0.89 S	493.18 ± 7.82 S	13.6 X
SYDNEY HOUSING	6	1600	91.21 ± 1.69 S	1192.03 ± 30.09 S	13.1 X