Data Science - Coursework 2

Comparison of two methods for calculating SHAP values for categorical variables.

02344391

May 2024

Introduction

It is often necessary to explain the predictions made by a machine learning model in order to understand the model's decisions. The simplest models, such as linear regression models, can be explained by their coefficients. On the other hand, more complex models sometimes require the use of an explanatory model to locally explain the predictions associated with each input.

The SHAP *Python* module uses Shapley values from game theory to explain the predictions of a wide range of models. According to Lundberg et al. [2019], the module calculates the contribution ϕ_i for each feature i given an input u on a model f by:

$$\phi_i = \sum_{S \subseteq N \setminus \{i\}} \frac{|S|!(M - |S| - 1)!}{M!} [f_u(S \cup \{i\}) - f_u(S)] \tag{1}$$

where N is the feature set, $M = |N|$ and:

$$f_u(S) = \mathbb{E}[f(u)|u_S]$$

is the expected value of the function given a subset S of the features of u.

In the case of tree-based or ensemble models such as random forest, the `TreeExpalainer` class of SHAP calculates the Shapley values of each feature in an input from the fitted trees. The algorithm calculates $f_u(S)$ in (1) by traversing the branches of the decision tree according to the feature values of S. When a node of the tree corresponds to a feature not included in S, the algorithm descends both branches under this node. Each of these descents leads to a value. The contribution of the subgroup S is then the weighted sum of the values obtained by traversing the tree. The weighting corresponds to the proportion of samples that passed through the branches of the tree during training.

So we have a method to calculate the contribution of each variable in this kind of models. In this report, we will not investigate the consistency of this method for explaining predictions, but we will study the particular case where a model has categorical variables that have been encoded by One-Hot Encoding. In this case, Shapley values are associated with each of the encoded variables. One method to calculate the contribution of categorical variables as a whole is to sum the Shapley values of the encoded variables corresponding to the categorical variable. This method for calculating the Shapley values of a decision tree assumes that the sub-variables produced by encoding a categorical variable are independent. This assumption is inherently false: the encoded sub-variables should be considered as a whole when calculating the Shap values using the fitted trees.

The goal of this project is to modify the algorithm calculating the SHAP values of `TreeExplainer` to consider the entire set of sub-variables of a categorical variable as known when it is part of S. Moreover, this project heavily relies on functions from Scikit-learn (Pedregosa et al. [2011]), some of which are modified (e.g. `sklearn.tree.plot_tree`) or not, for visualizations, dataset construction, or machine learning model construction. this project

A simple example of a regression tree

We illustrate the calculation of SHAP values for a short regression tree with a dataset containing a numerical variable x and a categorical variable that has been encoded with one-hot encoding cat_1 and cat_2. We plot in figure (1) the fitted tree. For the following, we want to analyse the prediction of the input $u = (u_x, u_{\text{cat}_1}, u_{\text{cat}_2}) = (60, 0, 1)$.

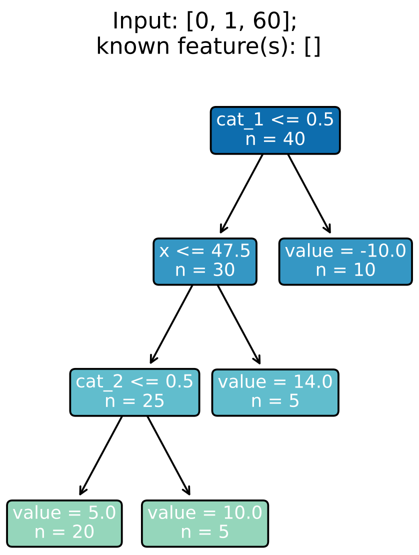

Figure 1: Plot of the fitted tree. The colored nodes correspond to the paths taken by the algorithm when certain features of an input are known. Here, we consider that no feature is known.

In figure (1), we consider that no feature is known, which corresponds to

$$f_u(S) = f_u(\{\}) = \mathbb{E}[f(X)] = \frac{1}{40}\left(20 \times 5 + 5 \times 10 + 5 \times 14 + 10 \times (-10)\right)$$

the mean value of the prediction on the training set.

In figure (2), we plot the tree paths when one feature is known: $S = \{x\}$ or $S = \{\text{cat}_1\}$ or $S = \{\text{cat}_2\}$. We find that

$$f_u(\{\text{cat}_1\}) = \frac{22}{3}$$
$$f_u(\{\text{cat}_2\}) = \frac{11}{2}$$
$$f_u(\{x\}) = 8$$

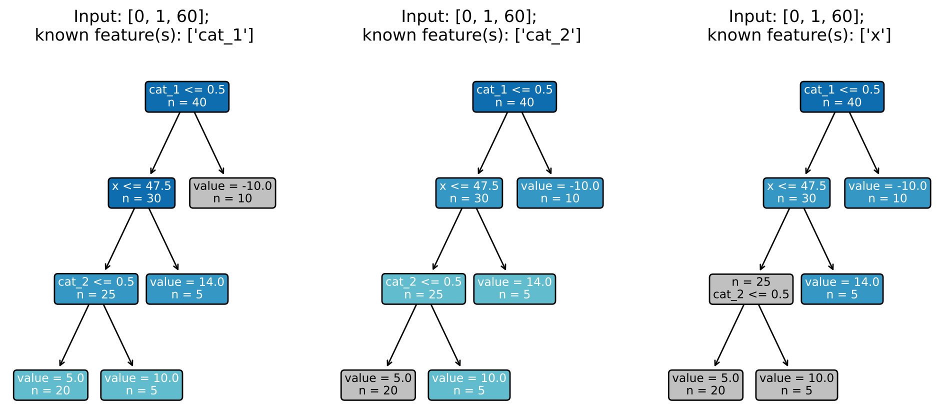
Input: [0, 1, 60]; known feature(s): ['cat_1']
Input: [0, 1, 60]; known feature(s): ['cat_2']
Input: [0, 1, 60]; known feature(s): ['x']
cat_1 <= 0.5 n = 40
x <= 47.5 n = 30
value = -10.0 n = 10
cat_2 <= 0.5 n = 25
value = 14.0 n = 5
value = 5.0 n = 20
value = 10.0 n = 5

Figure 2: Plot of the fitted tree. The colored nodes correspond to the paths taken by the algorithm when certain features of an input are known. Here, we consider that one feature is known.

In figure (3), we plot the tree paths when two features are known: $S = \{\text{cat}_1, \text{cat}_2\}$ or $S = \{\text{cat}_2, x\}$ or $S = \{\text{cat}_1, x\}$. We find that

$$f_u(\{\text{cat}_1, \text{cat}_2\}) = \frac{32}{3}$$
$$f_u(\{\text{cat}_2, x\}) = 8$$
$$f_u(\{\text{cat}_1, x\}) = 14$$

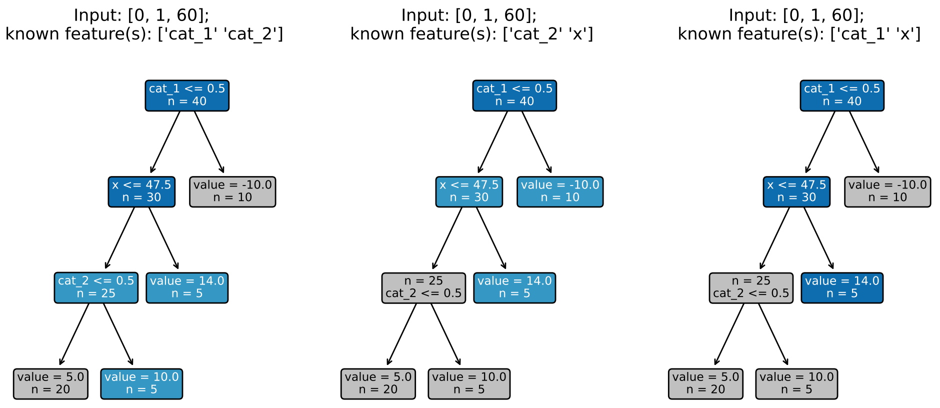
Input: [0, 1, 60]; known feature(s): ['cat_1' 'cat_2']
Input: [0, 1, 60]; known feature(s): ['cat_2' 'x']
Input: [0, 1, 60]; known feature(s): ['cat_1' 'x']
cat_1 <= 0.5 n = 40
x <= 47.5 n = 30
value = -10.0 n = 10
cat_2 <= 0.5 n = 25
value = 14.0 n = 5
value = 5.0 n = 20
value = 10.0 n = 5

Figure 3: Plot of the fitted tree. The colored nodes correspond to the paths taken by the algorithm when certain features of an input are known. Here, we consider that two features are known.

Finally, we plot in figure (4) the tree paths when all the features are known:

$$S = \{\text{cat}_1, \text{cat}_2, x\}.$$

We find that

$$f_u(\{\text{cat}_1, \text{cat}_2, x\}) = 14$$

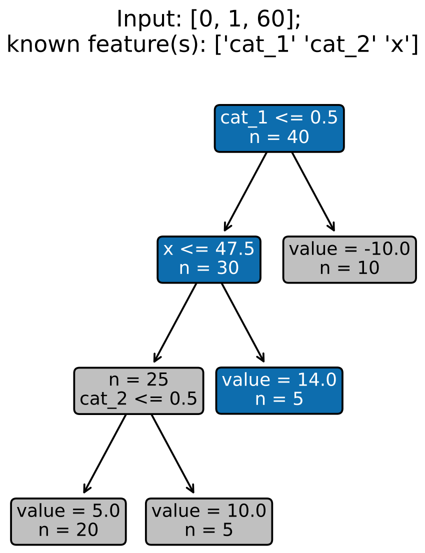

Figure 4: Plot of the fitted tree. The colored nodes correspond to the paths taken by the algorithm when certain features of an input are known. Here, we consider that three features are known.

By applying formula 1 to each of the features, we obtain:

$$\phi_x = \frac{155}{36} \approx 4.3056$$
$$\phi_{\text{cat}_1} = \frac{191}{36} \approx 5.3056$$
$$\phi_{\text{cat}_2} = \frac{25}{18} \approx 1.3889$$

And on the other hand, if we redo the calculation considering that cat_1 and cat_2 form a set cat of cardinality 1 and are inseparable, the SHAP values become:

$$\phi'_x = \frac{25}{6} \approx 4.1667$$
$$\phi_{\text{cat}} = \frac{41}{6} \approx 6.8334$$

We observe that $\phi_{\text{cat}} \neq \phi_{\text{cat}_1} + \phi_{\text{cat}_2}$ and $\phi_x \neq \phi'_x$. However, the values are quite close. Next, we will compare these two methods for different datasets to see if they lead to similar results.

Comparison of the two methods with generated datasets

In order to compare the two methods, we modify the algorithm 2 Tree SHAP on page 4 of Lundberg et al. [2019]. This is a recursive algorithm that performs the calculations we have previously shown, in an optimal way without repeating values already calculated. In particular, we modify in the `RECURSE` procedure, the `FINDFIRST` function, which becomes a function searching for the first occurrence of any member of the feature group of the considered feature. We were inspired by the code in _tree.py to code our own version.

We also need to generate new datasets with categorical variables. Since all the algorithms are coded in *Python*, the computation times are much longer than the original *C++* algorithm.

Therefore, we need to limit the size of the datasets and the number of trees in our predictive models.

To generate datasets, we use the functions `make_classification` or `make_regression` from `sklearn.datasets`. These functions only provide numerical features. Therefore, we need to artificially transform numerical features into categorical variables by associating each value with a category based on its quantile. A random component is added to introduce errors in the category assignment.

In total, 120 datasets are generated (60 for binary classification and 60 for regression). They consist of 7 features with 5 informative features. Among the datasets, we construct 1, 3, or 5 categorical variables in a balanced manner. All categorical variables contain 4 categories. The models used are random forest models with 20 decision trees. The train/test ratio is 0.2.

The performance of the classification models are all acceptable, as reported in table 1. On the other hand, the regression models underperform. Therefore, we decide not to study the SHAP values of these models because it is not relevant to explain predictions of a model that does not perform well.

	1 categorical feature	3 categorical features	5 categorical features
Accuracy	0.82	0.81	0.75
Precision	0.86	0.84	0.80
Recall	0.81	0.79	0.71

Table 1: Average performance of the classification models for datasets containing 1, 3, 5 categorical features.

Since transforming into categories causes a loss of precision in numerical variables, it is normal for performance to decrease as the number of categorical variables increases for the same total number of features.

For each of the datasets, we normalise the absolute values of the SHAP values so that the absolute values of the SHAP values for each prediction sum up to 1.

Then, we construct boxplots in figure 5 to show the distribution of the mean differences of the normalized SHAP values calculated by the two methods for each prediction. We observe that the mean differences are very small for the most part. Indeed, in the most extreme cases, they barely exceed 3%. This shows that the two methods seem to produce similar results.

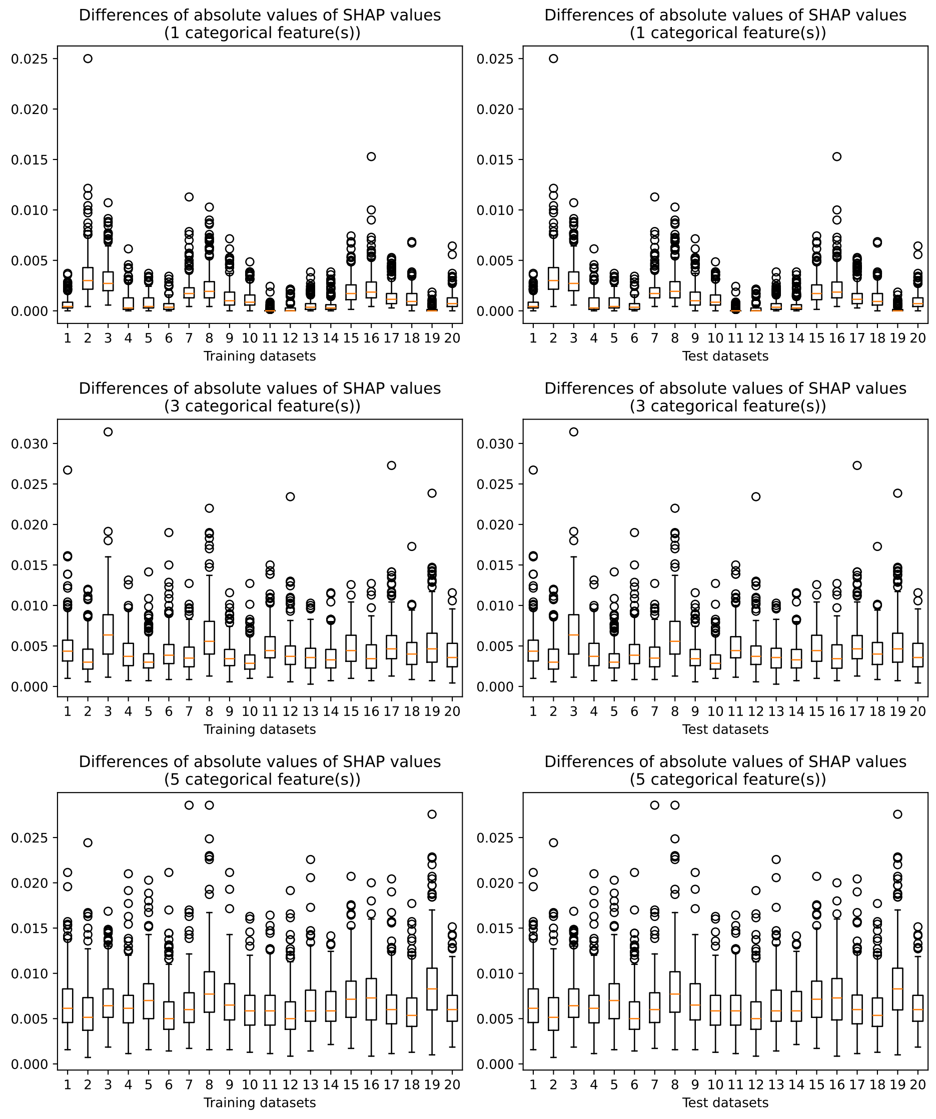

Figure 5: Boxplots of the absolute difference average of the absolute normalised SHAP values of both methods over the features for each prediction on the training and test set

The SHAP values are also used to perform rankings from the most significant variable to the least significant one. It is therefore relevant to count, across the entire prediction set, the number of times the rankings differ. More precisely, for each dataset configuration, we rank the absolute values of the normalized SHAP values that are greater than 10%. We set this threshold to eliminate variables deemed insignificant that may disturb the end of the ranking. Then, we calculate the number of times the rankings differ and divide by the total number of predictions for the training and test sets. We report the error rates in table (2). We observe that as the number of categorical features increases, the rankings differ more. In the case of a dataset with many categorical variables, the differences become significant.

	1 categorical feature	3 categorical features	5 categorical features
Training set	0.05	0.15	0.25
Test set	0.05	0.17	0.25

Table 2: Number of times the rankings differ divided by the total number of predictions for the training and test sets and given the number of categorical data.

References

S. M. Lundberg, G. G. Erion, and S.-I. Lee. Consistent individualized feature attribution for tree ensembles, 2019.

F. Pedregosa, G. Varoquaux, A. Gramfort, V. Michel, B. Thirion, O. Grisel, M. Blondel, P. Prettenhofer, R. Weiss, V. Dubourg, J. Vanderplas, A. Passos, D. Cournapeau, M. Brucher, M. Perrot, and E. Duchesnay. Scikit-learn: Machine learning in Python. *Journal of Machine Learning Research*, 12:2825–2830, 2011.