SHAP (SHAPLEY ADDITIVE EXPLANATIONS)

A PREPRINT

Yann Leterrier
Statistical Learning
La Sapienza University

Matteo Broglio
Statistical Learning
La Sapienza University

Pol Ribó León
Statistical Learning
La Sapienza University

June 17, 2020

ABSTRACT

The best explanation of a simple model is the model itself; it perfectly represents itself and is easy to understand. For complex models (black box models), such as ensemble methods or deep networks, we cannot use the original model as its own best explanation because it is too hard for a human mind to comprehend every detail. Instead, a simpler explanation model has to be manufactured, a model which works as any interpretable approximation of the original model.

SHAP (SHapley Additive exPlanations) is a method developed by Lundberg and Lee (2016) in his papers "Consistent individualized feature attribution for tree ensembles" and "A unified approach to interpreting model predictions", in an attempt to explain individual predictions; is a method for evaluating the contribution of an input feature to a model's output.

The authors developed this method because popular feature attribution methods, such as SplitCount or Gain, are inconsistent, meaning they can lower a feature's assigned importance when the true impact of that feature actually increases because they are often based on heuristic rules. In this report we are going to go further in detail about SHAP, both theoretically and practically, in an attempt to present new methods that show improved computational performance and/or better consistency with human intuition than such previous approaches.

1 Background

1.1 LIME (Local Surrogate)

A simple methods to explain a specific output of a black box model is focusing on training local surrogate models to explain this individual predictions. Here is its recipe :

- Select the instance of interest
- Perturb your dataset and get the black box predictions for these new points
- Weight the new sample according to their proximity to the instance of interest
- Train a weighted and interpretable model on the new dataset
- Explain the prediction by interpreting the local model

This steps are illustrated in the below example :

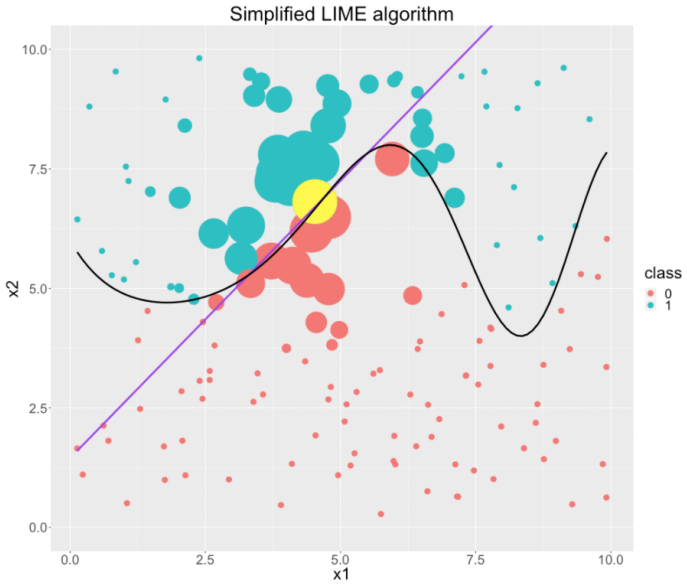


Figure 1: LIME in 2D

More generally, LIME is about resolving an optimization problem :

$$explanation(x) = \arg\min_{g \in G} L\left(f, g, \pi_x\right) + \Omega(g) \tag{1}$$

where L is the loss-like fidelity, f is the black box model, g is the explainer, π is the neighborhood measure and Ω is the complexity term.

1.2 Shapley Values

Shapley values are values computed from concepts of game theory. In the analogy, the features of the model represent the players of the game and the prediction of the model stands for the payout of the game. Shapley values tell us how to distribute the payout among the features. It is the average contribution of a feature value to the prediction in different coalitions, not the difference in prediction when we remove the feature from the model.

To fix the ideas, let's get an example. Let's consider a dataset with four features (cat-banned, park-nearby, floor, aerea) to predict the output, which is the apartment price. Shapley values could help us computing the feature importance of the feature cat-banned, for instance. For that, we compute and average the output with and without this feature, for every possible coalition among the rest of the features (7 possible coalition of 3 features).

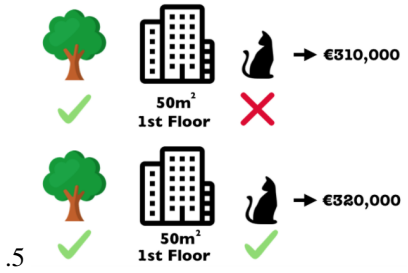


Figure 2: With and without cat-banned for a single coalition

Figure 3: All possible coalitions

More generally, this results to the below formula :

$$\phi_i = \sum_{S \subseteq F \setminus \{i\}} \frac{|S|!(|F| - |S| - 1)!}{|F|!} \left[f_{S \cup \{i\}} \left(x_{S \cup \{i\}} \right) - f_S \left(x_S \right) \right] \tag{2}$$

where ϕ_i is the Shapley value for feature i, F represents the set of features, S represents every coalition among the features and f is the back-box model.

2 SHAP (Shapley Additive exPlanations)

ShAP, is an additive feature attribution method to explain model predictions. It drinks both from LIME methods and Shapley values theory. Let us remark that although it is heavily related to Shapley values and makes use of them, it is not an application. The reason is simple: through KernelSHAP, a kernel-based estimation for Shapley values, inspired by surrogate models (LIME) is a method of its own. We will dive into it in this section. SHAP is also an alternative to popular feature attribution methods, such as SplitCount or Gain, which are often inconsistent, as they can lower a feature's assigned importance when the true impact of that feature actually increases.

As so, the goal of SHAP is to explain the prediction of an instance x by computing the contribution of each feature to the prediction. The SHAP explanation method computes Shapley values from coalitional game theory. The feature values of a data instance act as players in a coalition, and the Shapley values tell us how to fairly distribute the prediction among the features.

2.1 Additive Feature Attribution Methods

Additive feature attribution methods have an explanation model that is a linear function of binary variables.

$$g\left(z'\right) = \phi_0 + \sum_{i=1}^{M} \phi_i z_i' \tag{3}$$

, where g is the explanation model, $z'\{0,1\}$ is the coalition vector (a feature = 1 if taken, 0 otherwise), while M is the maximum coalition size and the ϕj's are the Shapley values.

Methods with explanation models matching the definition below approximate the output $f(x)$ of the original model by attributing an effect ϕi to each feature and then summing the effects of all feature attributions (the Shapley values).

2.2 Properties of SHAP

SHAP and for the class of additive feature attribution methods provide the presence of a single unique solution in this class with three desirable properties.

Local Accuracy

$$f(x) = g\left(x'\right) = \phi_0 + \sum_{i=1}^{M} \phi_i x_i' \tag{4}$$

When approximating the original model f for a specific input x, local accuracy requires the explanation model to at least match the output of the original model for the coalition vector x'. It is similar to the 'Efficiency' property of Shapley values.

Missingness

$$x_i' = 0 \Longrightarrow \phi_i = 0 \tag{5}$$

If coalition vector represents feature presence, then missingness requires that features missing in the original input to have no impact. Although a missing feature could have an arbitrary Shapley value without hurting the local accuracy (since it is multiplied by xj=0), missingness enforces missing features to have a value of 0.

Consistency

If we let

$$f_x\left(z'\right) = f\left(h_x\left(z'\right)\right) \tag{6}$$

and

$$z' \backslash i \tag{7}$$

indicate that

$$z_i' = 0 \tag{8}$$

, for any two models f and f', if:

$$f_x'\left(z'\right) - f_x'\left(z' \backslash i\right) \geq f_x\left(z'\right) - f_x\left(z' \backslash i\right) \tag{9}$$

for all inputs

$$z' \in \{0,1\}^M \tag{10}$$

then

$$\phi_i\left(f', x\right) \geq \phi_i(f, x) \tag{11}$$

The consistency property states that if a model changes so that the marginal contribution of a feature value increases or stays the same (regardless of other features), then the Shapley value also increases or stays the same.

2.3 KernelSHAP

KernelSHAP consists of 5 steps:

- Sample coalition vectors
- Get the prediction for each coalition vector by first converting it to the original feature space and then applying the original model with a mapping function.
- Compute the weight for each coalition vector with the SHAP kernel.
- Fit the weighted linear model.
- Return the Shapley values, which are the coefficients from the linear model.

To apply this step, we can create a random coalition by repeated coin flips until there is a chain of 0's and 1's. The K sampled coalitions become the dataset for the regression model. The target for the regression model is the prediction for a coalition. Then, to get from coalitions of feature values to valid data instances, a mapping function is needed. As so, the function hx maps 1's to the corresponding value from the instance x that we want to explain and for example, for tabular data, it maps 0's to the values of another instance that we sample from the data, so an absent feature value is replaced by a random feature value from the data.

Then, it is time to compute the weight of each instance (that we gathered from the coalition vector and the mapping function), and in this step is where the biggest difference to LIME happens. Meanwhile LIME weights the instances according to how close they are to the original instance, SHAP weights the sampled instances according to the weight the coalition would get in the Shapley value estimation. So, small coalitions (few 1's) and large coalitions (many 1's) get the largest weights. The intuition behind this method is explained in this motto: "We learn most about individual features if we can study their effects in isolation." Given this, if a coalition consists of a single feature or if a coalition consists of all but one feature, this manufactured "isolation effect" provides a better learning process about the feature's effect on the prediction. To achieve Shapley compliant weighting, Lundberg et. al propose the SHAP kernel:

$$\pi_x\left(z'\right) = \frac{\left(M-1\right)}{\left(\begin{array}{c} M \\ z'| \end{array}\right)\left|z'\right|\left(M-\left|z'\right|\right)} \tag{12}$$

Once this is done, we fit the weighted linear regression model:

$$g\left(z'\right) = \phi_0 + \sum_{j=1}^{M} \phi_j z'_j \tag{13}$$

which is trained by optimizing the following loss function L:

$$L\left(f, g, \pi_x\right) = \sum_{z' \in Z} \left[f\left(h_x\left(z'\right)\right) - g\left(z'\right)\right]^2 \pi_x\left(z'\right) \tag{14}$$

2.3.1 Specifications of KernelSHAP

KernelSHAP has the drawback that it is slow, which makes it a bit too unfeasible when computing Shapley values for many instances.

Also as KernelSHAP replaces feature values with values from random instances, it is more practical to randomly sample from the marginal distribution. However, this may cause to putting too much weight on unlikely data points if features are dependent. So, if two features are highly correlated, this ma lead to unreliable results. Despite this issue, sampling from the marginal distribution is necessary, as if we would sample from the conditional distribution, then the resulting values are no longer Shapley values, as they would violate the Shapley axiom of Dummy. Most other permutation based interpretation methods have this problem.

2.4 TreeSHAP

Lundberg et. al (2018) proposed TreeSHAP, a variant of SHAP for tree-based machine learning models such as decision trees, random forests and gradient boosted trees. TreeSHAP was introduced as a fast, model-specific alternative to KernelSHAP.

Speed is the main advantage of Tree SHAP ,as it reduces the computational complexity from

$$O\left(TL2^M\right) \tag{15}$$

to

$$O\left(TLD^2\right) \tag{16}$$

, where T is the number of trees, L is the maximum number of leaves in any tree and D the maximal depth of any tree.

3 The SHAP library

Implementation of SHAP method is provided by the authors in the shap Python package. This implementation works for different models, in particolar for tree-based models in the scikit-learn machine learning library for Python. The datasets used in this chapter are provided by the library itself in the module *shap.datasets*. We used *adult* and *iris* datasets with a Gradient Boost Model (*XgBoost* library).

3.1 Feature Importance Plot

The idea behind SHAP feature importance is quiet simple: Features with large absolute Shapley values are more important. Since we want the global importance, we average the absolute Shapley values for each feature:

$$I_j = \sum_{i=1}^{n} |\phi_j^{(i)}| \tag{17}$$

Then we plot the features sorting by decreasing importance.

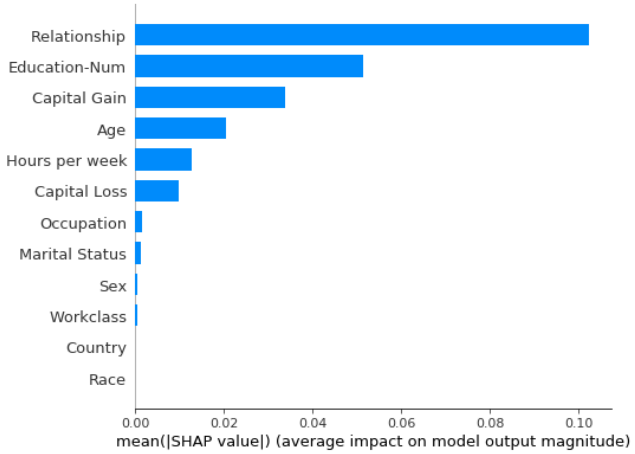


Figure 4: Feature Importance Plot

3.2 Summary Plot

The summary plot combines togheter feature importance(as seen in the past section) with feature effects on the output. Each point on the summary plot is a Shapley value for a feature and an instance. On the y-axis we have the features used in the model sorted by features importance(as seen before) and on the x-axis we have the Shapley value. The color represents the value of the original feature from low to high for eache features' range. Overlapping points are jittered in y-axis direction(like violin plots), in order to have a sot of density of the Shapley values per feature. To get an overview of which features are most important for a model we can plot the SHAP values of every feature for every sample.

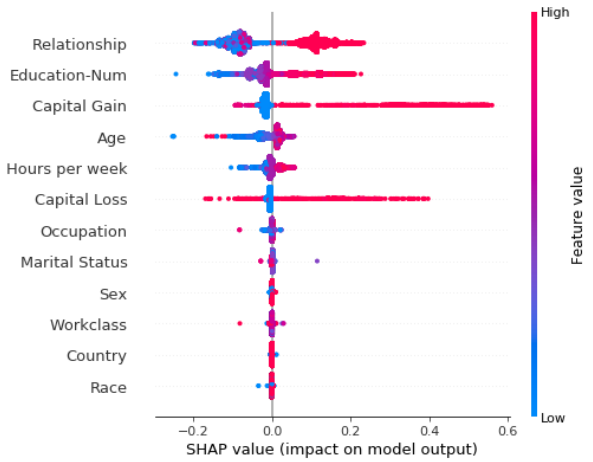


Figure 5: Summary Plot

3.3 Force Plot and Waterfall Plot

A Force plot like a Waterfall plot are both effective way to show how the model arrived at its decision. They show how the different values move from the base value, 0.336 in the figure 7), to the final value of 0.244

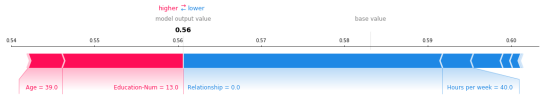


Figure 6: Force Plot Probs

In the Waterfall plot we have a clear vision of how each feature moves the starting value (the expected value of the entire dataset) down(to the left) or up(to the right) for each feature till the final value shown above. In the Force plot at figure 6, instead, we have a global vision of both neagtive and positive values and which of them is more important(which of the moves the original value to a better or worst probability).

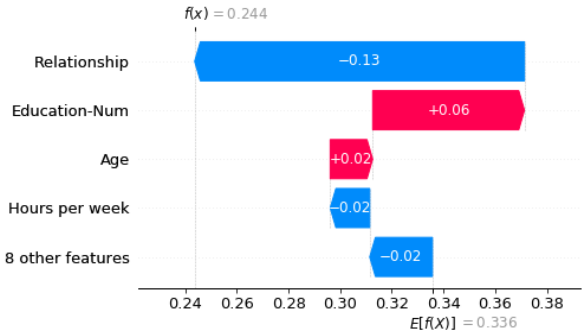


Figure 7: Waterfall Plot

3.4 Dependence Plot

A dependence plot is a scatter plot that shows the effect a single feature has on the predictions made by the model. In this example the log-odds of making over 50k increases significantly between age 20 and 40. Here, we focus on a specific feature. On the x-axis we have the different values of our target-feature. On the y-axis we have got the SHAP

value of that specific feature. Unlike the Summary plot, the different color shows the value of a related feature. Here, we want to see how the SHAP value changes when we move on the features' range and how the dependence of another features influence the Shapley values.

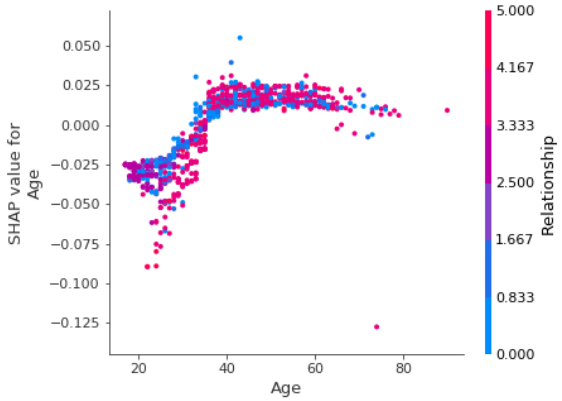


Figure 8: Dependance Plot

3.5 Cluster Plot

The Cluster plot is basically a cumulative plot of Force plots, each of which explains the prediction of an instance. We can cluster our data with a any kind of clustering (hierarchical or not-hierarchical clustering) and order them by similarity. The goal is to find groups of similar instances. Normally, clustering is based on features. Features are often on different scales. The SHAP values are normalized values and they have the same unit, so we can use them for a right and consistent clustering.

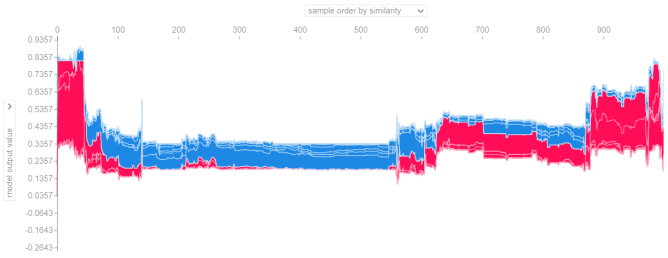


Figure 9: Cluster Plot

3.6 Decision Plot

Similar to Force Plot and Waterfall plot, a Decision plot shows how SHAP values interact with the final result. In figure 10 we have on the x-axis the different log odds values for all the features. In figure 11 we have the same ouput with the probability values. Furthermore, in this figure we have plotted a dashed line for a misclassification. This is useful if we want to investigate how the SHAP values gives the wrong final result.

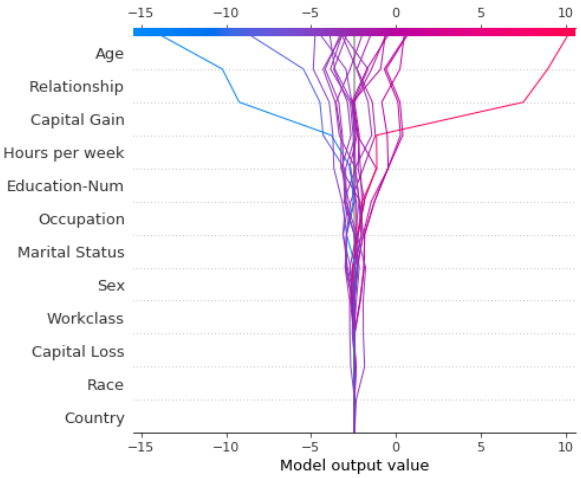
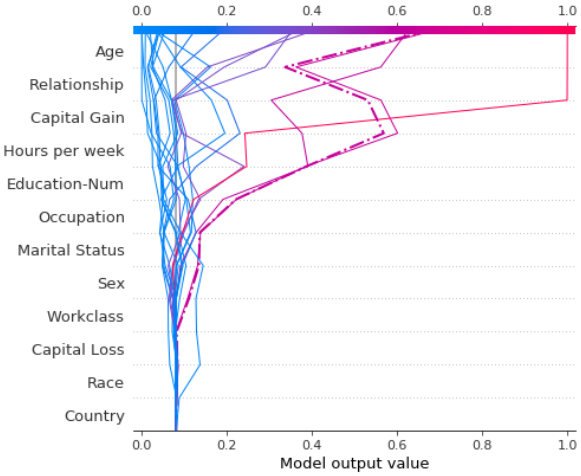


Figure 10: Decision Plot

Decision plots support SHAP interaction values: the first-order interactions estimated from tree-based models. While SHAP dependence plots are the best way to visualize individual interactions, a decision plot can display the cumulative effect of main effects and interactions for one or more observations.



When is a decision plot helpful? There are several use cases for a decision plot. We present several cases here:

- Show a large number of feature effects clearly.
- Visualize multioutput predictions.
- Display the cumulative effect of interactions.
- Explore feature effects for a range of feature values.
- Identify outliers.
- Identify typical prediction paths.
- Compare and contrast predictions for several models.

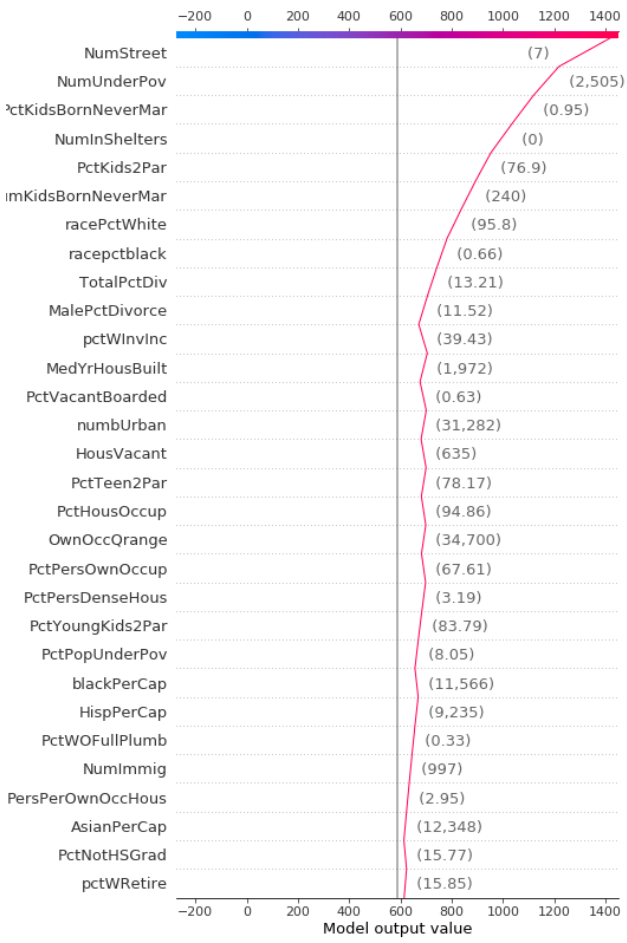


Figure 12: Decision Plot

3.7 Plots for multi-output models

The Summary plot for multi-output models is quiet similar to the traditional Summary plot. Here, we can see the global importance as the average of the absolute Shapley values for each feature for each label.

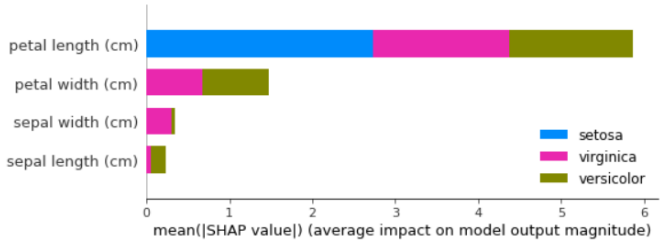


Figure 13: SummaryPlot for Multiple Output

A decision plot is very helpful for multioutput models. In fact, we can show the final result for each output label of every istance. So we can see togheter how SHAP values for each feature change the final result.

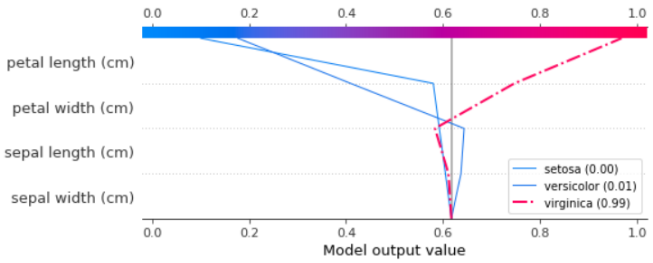


Figure 14: DecisionPlot for Multiple Output

4 CONCLUSION

In conclusion, SHAP (SHapley Additive exPlanations) is a game theoretic approach to explain the output of any machine learning model.

It connects optimal credit allocation with local explanations using the classic Shapley values from game theory and also takes an inspirational breeze from LIME methods in order to build through a linear model. The astonishing close relationship that bound game theory and model interpretability methods is brought to light here in the SHAP method. The SHAP framework identifies the class of additive feature importance methods and shows that there is a unique solution in this class that adheres to the desirable properties listed above.

The success in merging this apparently distinct fields reveals an encouraging sign that common principles about model interpretation following this mindset can promote the development of future methods.

In the aim of showing that these methods are useful and desirable, in this paper, it is presented the theory behind SHAP method, first diving into its background,inspirations and ideas and explained different estimation methods for SHAP values, along with experiments with SHAP Python Library.

References

[1] Scott M. Lundberg, Gabriel G. Erion and Su-In Lee. Consistent Individualized Feature Attribution for Tree Ensembles. February 2018

[2] Scott M. Lundberg and Su-in Lee. A Unified Approach to Interpreting Model Predictions In *Advances in Neural Information Processing Systems 30 (NIPS 2017)*. .

[3] Christoph Molnar. Interpretable Machine Learning A Guide for Making Black Box Models Explainable.(2018). Chapter 5 5.7: Local Surrogate (LIME) 5.9: Shapley Values 5.10: SHAP (Shapley Additive exPlanations

[4] Scott M. Lundberg. Tree ensemble example with TreeExplainer (XGBoost/LightGBM/CatBoost/scikit-learn/pyspark models) https://github.com/slundberg/shap