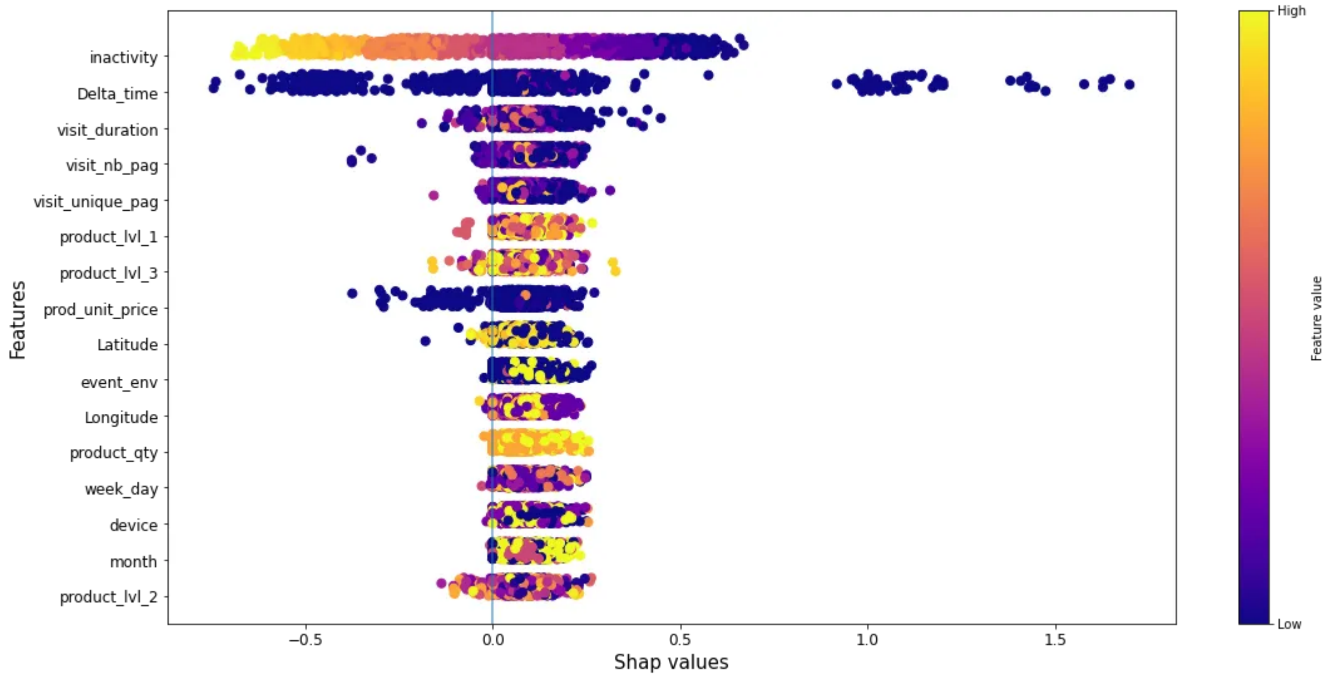
SHAPling around a custom network

Will game theory save the day?

Roberto Martorelli · Follow
Published in Towards Data Science
17 min read · May 20, 2021

Rbm, Thomas G., Benjamin Larrousse



This is the second part of a study in which the application of LIME and SHAP to a neural network for predicting the purchase probability of a person is analyzed. For details on the network architecture, together with the result of the application of LIME we refer to the previous article.

· · ·

The other headliner next to LIME in the field of interpretability is SHAP (acronym for *SHapley Additive exPlanations*).

Such as LIME, SHAP is a model-agnostic explainer whose aim is to provide local insights into the output of a model. The main difference to LIME is the rigorous mathematical foundations of the explainer: it enables the definition of a general framework for interpretability that includes several already existing algorithms (including LIME).

The next sections provide a brief introduction to SHAP and the results of its application to our use case.

· · ·

SHAP(ley) or the art of collaboration

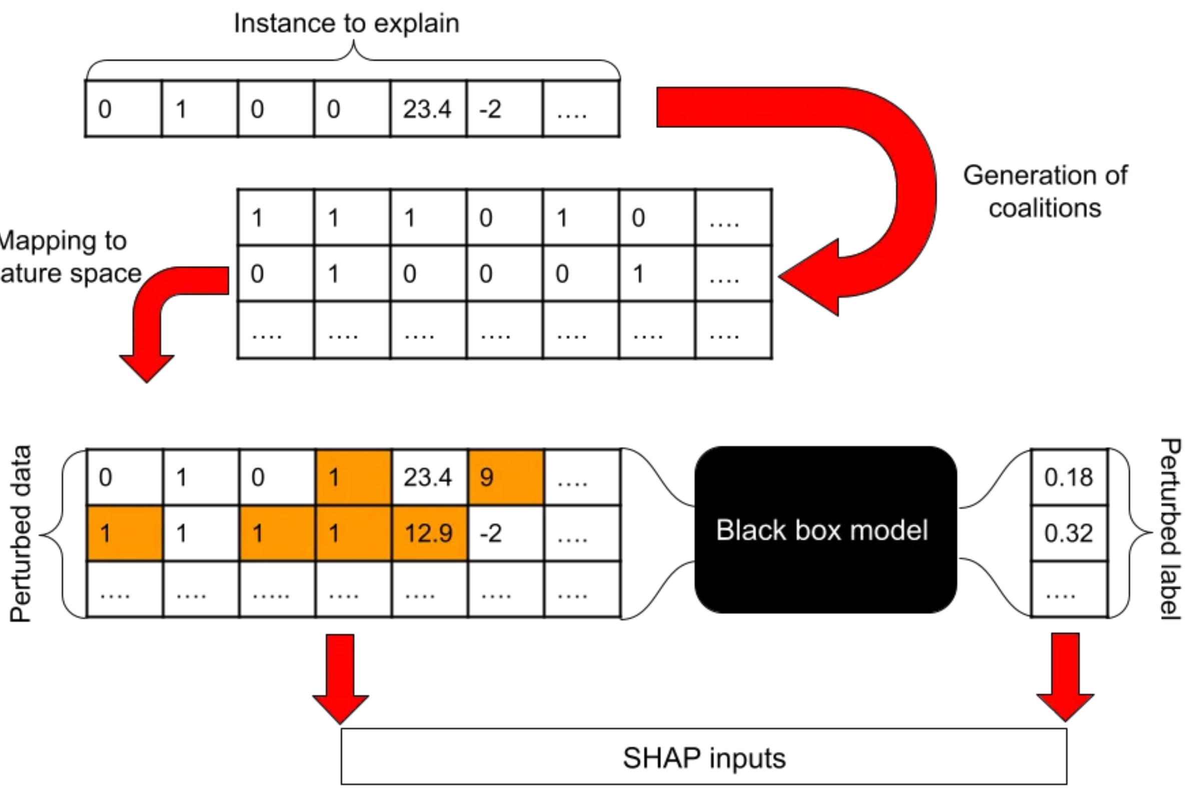
Being far more than a new explainer, SHAP combines different already developed methods, thereby underlining certain common features and providing a shared background that allows to group them under the same umbrella and to ensure a solid mathematical foundation.

In accordance with our use case, the introduction focuses on the method Kernel SHAP, which is a combination of LIME and Shapley values. For more information on the other methods and a deeper understanding of SHAP we refer to the original paper. For simplicity, in the following sections, we write “SHAP” for “Kernel SHAP”.

Such as LIME, SHAP aims to fit a linear model in the vicinity of the instance to explain. The first difference between LIME and SHAP is the way in which the training data for the fit of the linear model are generated.

LIME generates a set of perturbed data based on the characteristics of the training set for the model (or based on the training set itself if available). SHAP generates coalitions of features that are kept or removed from the instance to explain (Imagine it as applying a mask of “0/1” on the instance, with “0 = remove feature” and “1= keep feature”) and replaces the features to discard with some “representatives” values. These values can be either based on the training data or specified ad-hoc.

Such as in LIME, the perturbed set is then passed to the original model in order to generate the corresponding labels.



Schema of the generation of input data and labels for SHAP. (Image by Author)

Once the perturbed data have been generated, the interpretable model is trained by minimizing a weighted RMSE, exactly as in LIME.

So what is the big fuss about? So far, SHAP seems identical to LIME with a different perturbation method. However, SHAP provides specific choices for the weighting function and complexity measure: *these are the only valid choices for the explainer to meet local accuracy, missingness and consistency*.

Why do we even want these properties?

These properties ensure that we obtain the Shapley value for each feature (or an estimation of the Shapley values) when we evaluate the features contributions.

The Shapley value is a possible solution for a cooperative game, i.e., it is the only solution for “fairly” distributing the output of the game to each player. It evaluates the marginal contribution of a player averaged over all possible permutations, which represent coalitions of the player with others.

In our scenario, the game is the output of the model, and the players are the features contributing to the output.

We are aware that the concept may not be crystal clear to the reader; it was not meant to be clear after so few lines of explanation. The core idea is that in SHAP, the choice of the weighting function and complexity measure has a solid theoretical background, which is absent in LIME where these terms are customizable parameters.

. . .

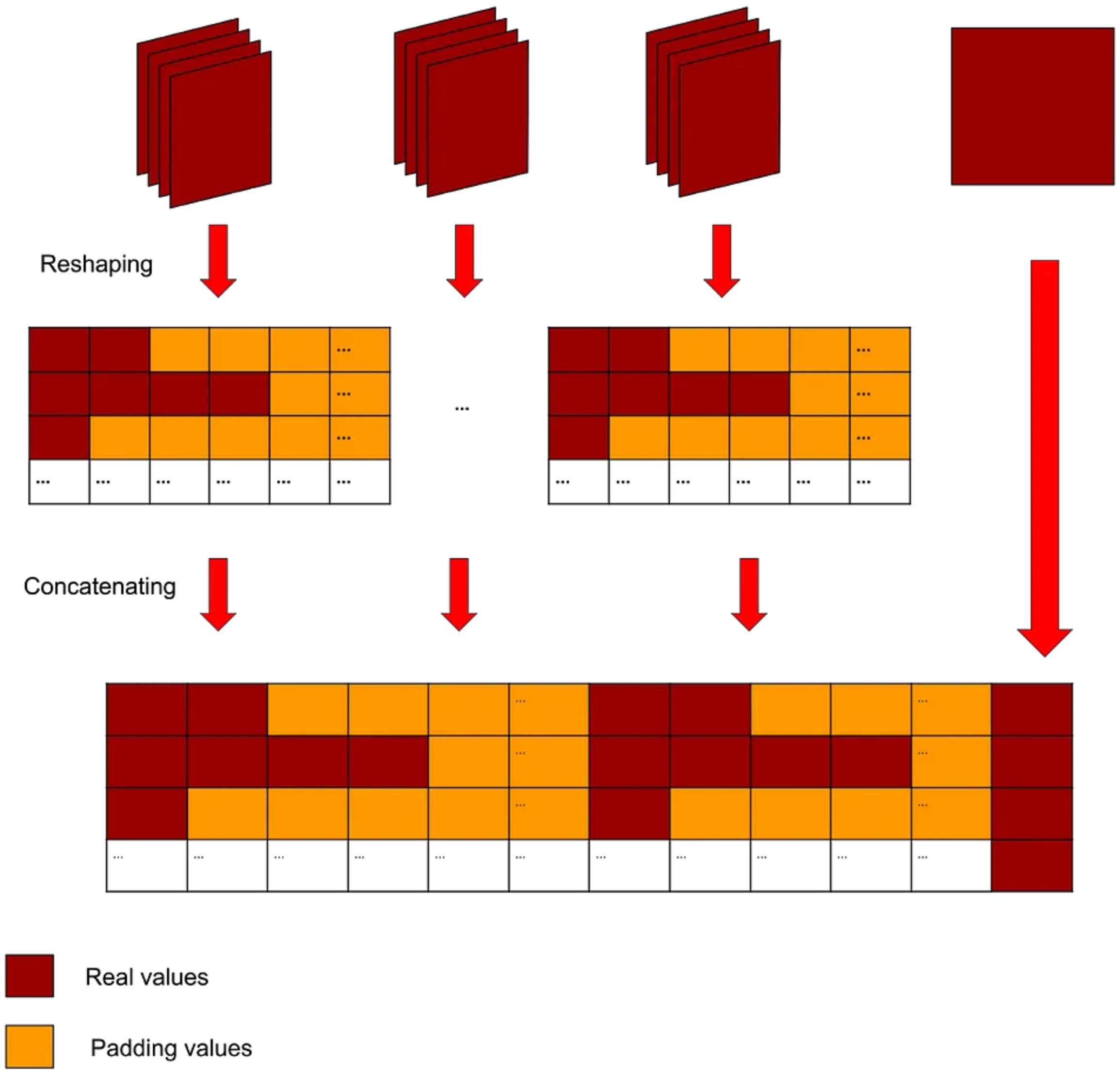
Matching together two pieces of a puzzle

Because SHAP is just a LIME explainer with a different choice of the parameters (Please recall that we are implicitly talking about "KernelSHAP"), we thought we could easily apply it to our custom model (for details on the architecture of the model we refer to the previous article). We were of course wrong 😅.

Some of the problems that occurred with LIME also occur in SHAP, such as dealing with one-hot-encoded or correlated features. These problems can be solved with the same pre-/post-processing steps that we constructed for LIME.

The transformation that we could not perform in SHAP is the reshaping of input data from the shape required in the original model to the 2D shape required in SHAP (and LIME). Of course, the difficulties are not related to the actual reshaping action (we are well aware of the existence of Numpy) but to the consequences that the procedure would involve in SHAP.

In LIME the transformation involved the reshaping of the input of the LSTM cells in a 2D array; the second dimension has the size n_f*n_s, where n_f is the number of features, and n_s is the number of sequences of the instance. Because we wanted to use the same explainer for different instances, the number of sequences must be the maximal number of sequences among all instances.



Transformation from model input to LIME input. (Image by Author)

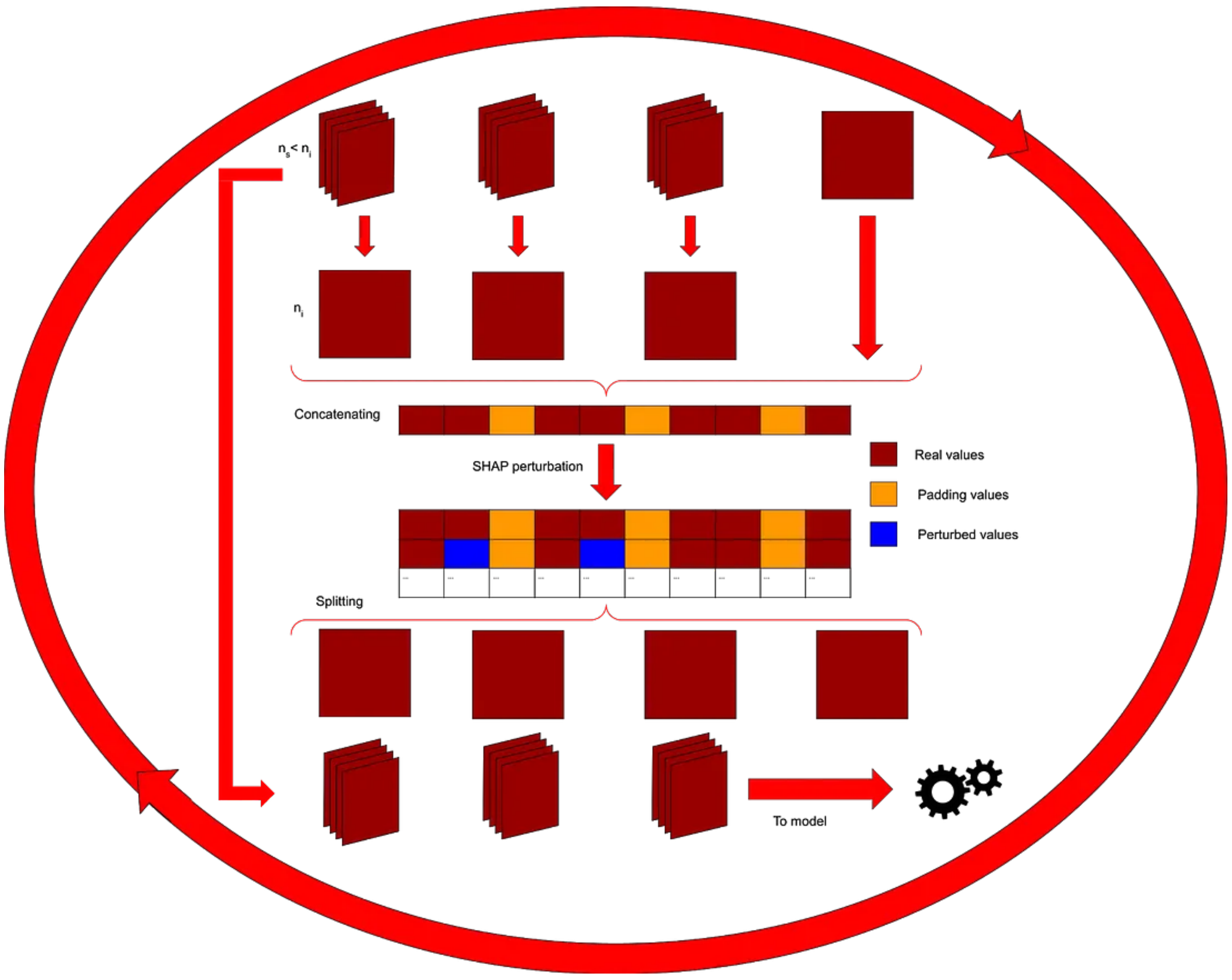
Let us imagine that the same data transformation method is applied in SHAP.

Because SHAP evaluates the Shapley value, the exact evaluation requires the generation of a number of perturbed data capable of exploring all the possible permutations of all the features contributing to the output of the model.

When applying the same transformation method as in LIME, 2^(n_f*max(n_s)) permutations are obtained, which are intractable for our model. Reducing this high number of perturbed data, will reduce the accuracy of the resulting Shapley values.

Moreover, the distribution of the number of sequences is skewed toward low sequences; therefore most of the data will be composed of padded values, needlessly overcharging the computation.

Based on these considerations, we modified the application of SHAP with respect to that of LIME. Rather than transforming the original inputs into a single 2D array, we recursively apply SHAP to each sequence step of the instance to explain, thereby generating perturbed data only for the last sequence in each iteration. This reduces the number of optimal perturbations to n_s*2^n_f.



Transformation from model input to SHAP input. (Image by Author)

. . .

Feeling the water before jumping in

With all this handiwork done, we can start looking at the interesting matters: the obtained actual explanations.

One great feature of SHAP is the large amount of information that can be gathered by aggregating the explanations obtained for different instances. This can also be done with LIME; however in SHAP, owing to the use of Shapley values rather than the weights as the measure of the impact, the resulting aggregated explanations have a more solid background.

While this is a great property, in the specific case of sequenced inputs (such as in our model), properly assembling the Shapley values to obtain insight into the model can be difficult.

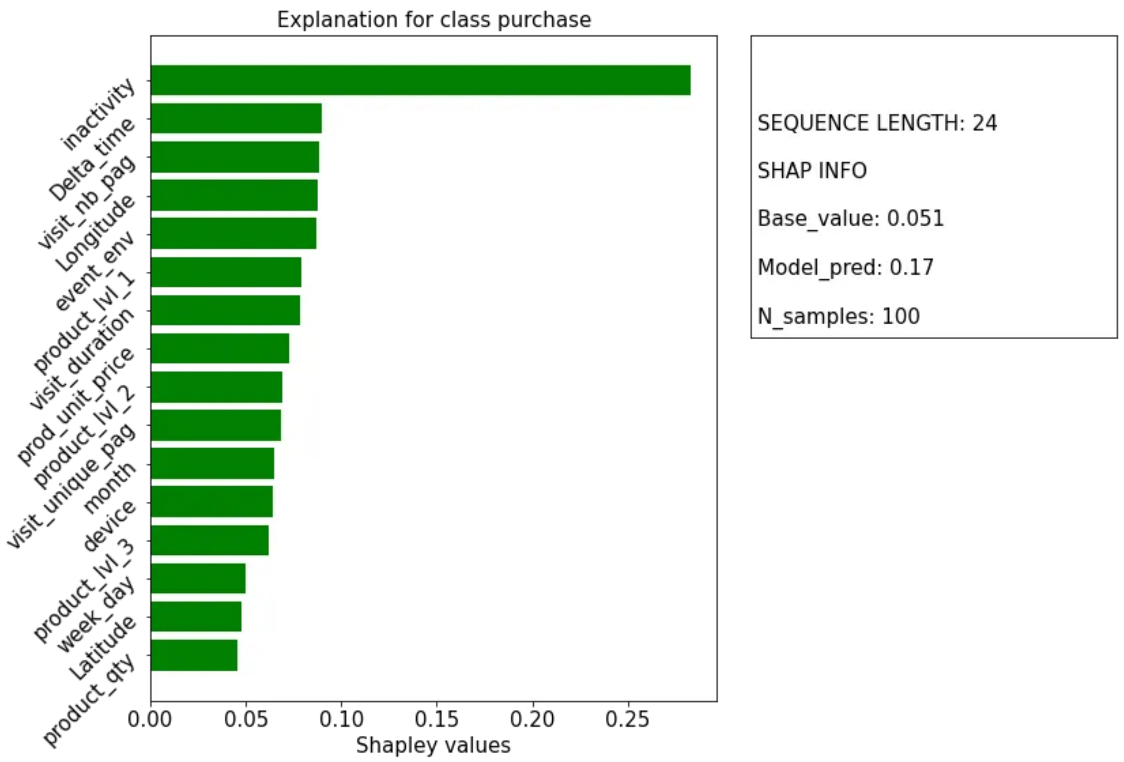
The most common SHAP applications involve two dimensional inputs (e. g., observations and features) such that the explanation presents, in an understandable way, the relationship between the Shapley values and feature values across the different instances (we are basically dealing with a three dimensional set).

In our case, there is also dependence on the position within the sequence of events; this additional dimension must be considered when the explanation is presented. Providing meaningful insight into the relationships between all the different players in the game can be a difficult task.

We will address this problem in the next sections.

Let us start by simply presenting the explanation for a single instance in the final step of its sequence of events.

To establish continuity with respect to the previous work on LIME, the following Figure presents the explanation with a layout similar to that in the previous article (for the same user). We explicitly omitted the two features related to the ratings of the products from the explanation because we saw in the previous study that the data are probably missing.



Explanation for a single instance at its last step using SHAP. (Image by Author)

Two aspects clearly emerge from the explanation: first, the *inactivity_time* (i.e., the time passed between the last action of this user and the prediction date) is the most important feature; second, all the features contribute to increasing the probability of purchase.

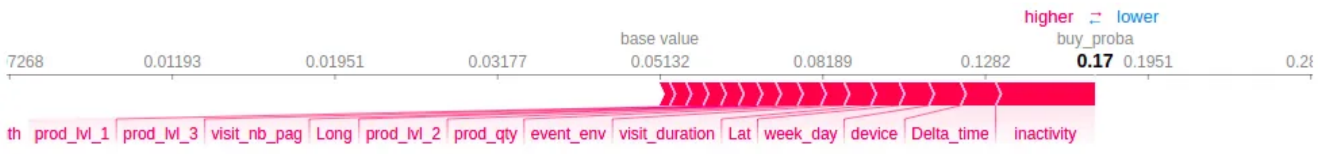
The first insight is identical to that we obtained with LIME. However, while SHAP assigns a positive impact to the *inactivity_time*, LIME assigns a negative impact, i.e., the *inactivity_time* decreases the predicted probability of purchase.

Which of the two results is correct?

In fact, both results are correct. The difference is due to the different interpretations provided by LIME and SHAP.

SHAP defines a feature impact with respect to a base value for the prediction obtained from the training data. An *inactivity_time* with a positive impact means that the value of the *inactivity_time* of this user increases the probability of purchase with respect to the base value of the model.

This interpretation is clearly presented with the *force_plot* visualization method of the SHAP library in the following Figure.



Explanation for a single instance at its last step using base SHAP force_plot function. (Image by Author)

The base value obtained from the training data is lower than the predicted one for this user; in addition, all the features contribute to shifting the purchase probability from the base value to the predicted one (*n. b. having a base value that is lower than the predicted one does not necessarily mean that all the features must have a positive impact; we can have features with a negative impact that are balanced by other features with a stronger positive impact*).

The other features have a much lower impact than the *inactivity_time* and differ only slightly from each other.

Keep in mind that this explanation refers to the last step of the series of events for this user. Because we can obtain, with the current implementation of SHAP, an explanation *in each step* (rather than one explanation for all the steps such as in LIME), the evaluated impact should be regarded as follow: *as the contribution of the feature X to the prediction with respect to the base value in the last step of the sequence*.

So far, the insights provided by SHAP and LIME are similar. They both stress the dominance of the *inactivity_time* over the other features. Nevertheless, one should remain careful while interpreting the results in the two cases.

This section regarded the explanation for a single instance in the last step of its sequence. What about the other steps of this user? Are the conclusions for this instance also globally valid?

Using the Shapley values evaluated in SHAP we can establish an interesting explanation for the impact of the features along the path and their overall effect on a single user. Here we go!

. . .

Wandering along the path

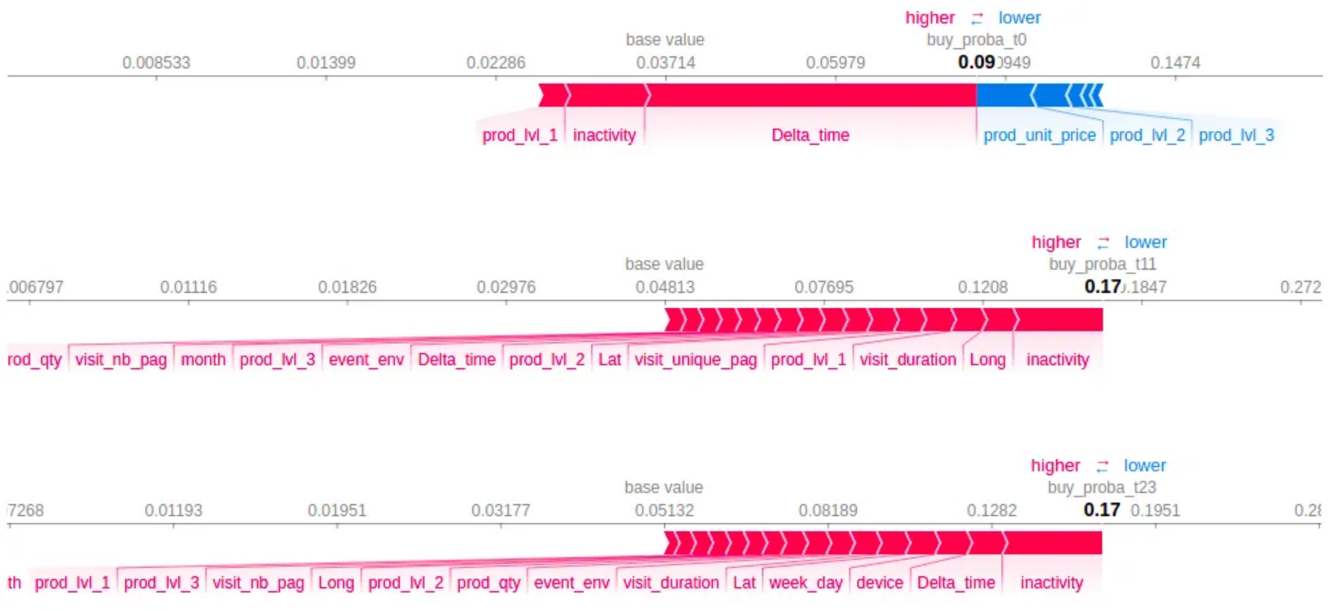
The effectiveness and popularity of SHAP are related to its solid foundation, computational efficiency, and global insight that it can provide by combining the local explanations of different instances.

How about using the same potentialities to generate an explanation for a single user along its whole sequence of events?

We can indeed aggregate the Shapley values for each step into the sequence to obtain insight into the evolution of the impact of a feature, overall feature impact on the sequence, or even the importance of a step with respect to the others. The possibilities are plenty.

First, does something interesting happen during different events in the sequence?

The intuitive answer would be “yes”. For example, some features may have greater impacts at the beginning of the user interaction. The following Figure presents this idea.



Explanation for a single instance in the first, mid and final steps of the sequence using the SHAP force_plot function. (Image by Author)

The impact of the feature can indeed change in different steps, also owing to the different base values and different predicted probabilities in different steps (*Because a SHAP explainer is trained in each step of the sequence, we have a base value and predicted probability in each step).*

The previous Figures seem to confirm our initial assumption (that the position along the sequence also has an effect on the final prediction).

The explanations for the mid- and final steps show some resemblance: the *inactivity_time* has the leading role with a positive impact, followed by the other features.

By contrast, the explanation for the initial step is different.

First, just a few features have an impact on the prediction in this case.

The *inactivity_time* is not the most important feature anymore; it has been replaced with the *delta_time* (i.e., the time passed after the previous interaction of a user).

This observation is curious .The *delta_time* in the initial step is set to zero because there are no previous steps. The importance of this feature at the beginning of the events may suggest that the shorter the *delta_time* is, the more the purchase probability increases (which is a reasonable intuition). As an alternative, it may simply stress the importance of the first step as an initializer of a path to a purchase.

In this first step we can see features with negative impacts on the predicted probability; notably, they are features related to the product.

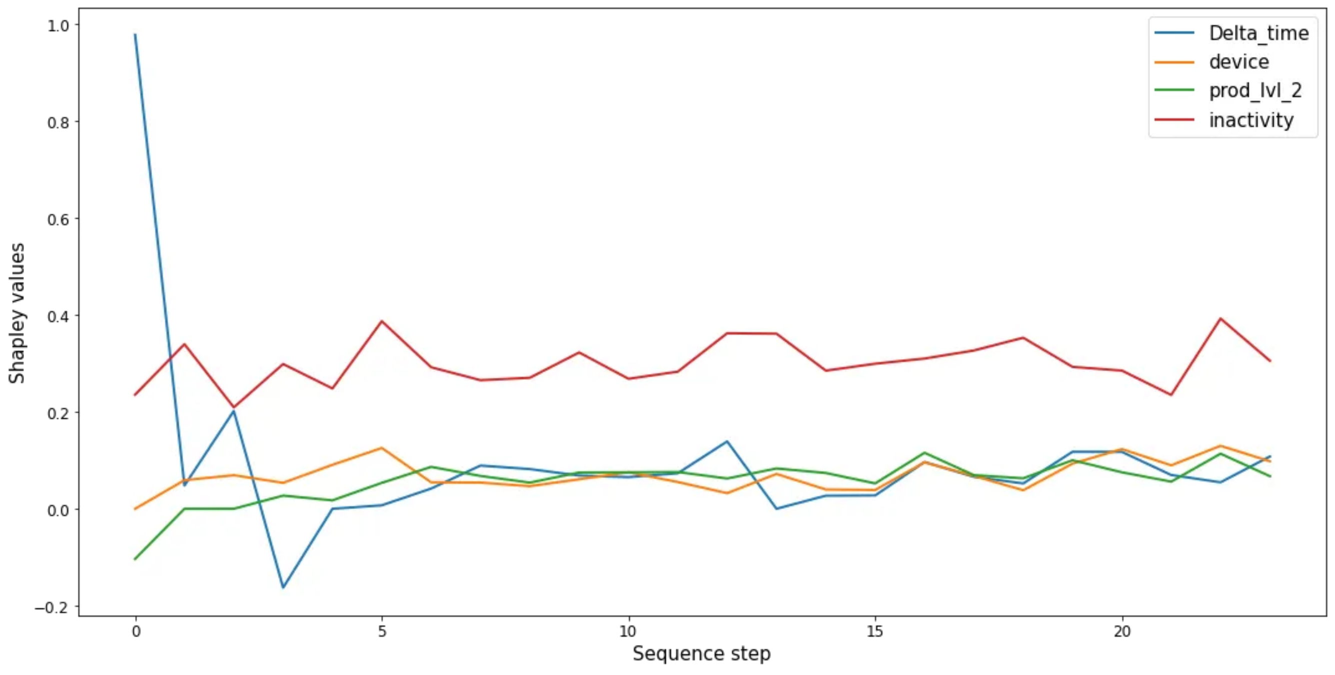
According to only these three figures, some hypotheses on the role of the features in the purchase prediction can be established:

- the initial step is fundamental for predicting the purchase probability of a user based on his/her interactions (according to the impact of the *delta_time* at the beginning of the sequence);
- the first product visualized by the user can have a significant negative impact on the purchase probability;
- the more the user interacts (thereby increasing the sequence length), the less is the impact of the individual features, except that of the *inactivity_time.*

We recall that we are looking at the explanation *for a single user*; thus, we cannot yet extrapolate these hypotheses to a global explanation. We will get there.

These are interesting (but somehow limited) insights; we should try to validate them by looking at the impact of all features on all the sequence steps.

In this direction, the following Figure presents the evolution of the Shapley values of four features.



Evolution of Shapley values of four features as functions of the sequence step. (Image by Author)

The *inactivity_time* is always superior to the other features; it maintains an approximately constant positive impact.

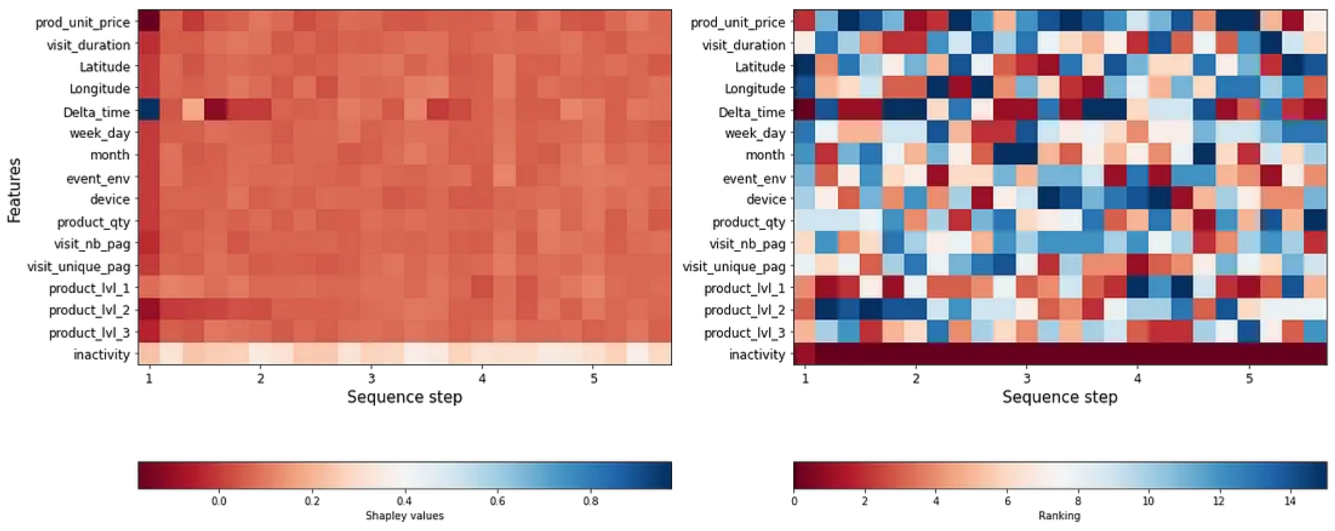
The impacts of the other features are rather weak, except that of the *delta_time*; all the features converge towards an impact of approximately zero toward the end of the sequence. In addition, the features' impacts exhibit a visible difference just until approximately the fifth step.

This hypothesis should be confirmed for other features and to obtain a more general overview of the evolutions of all the features of this instance.

To reach this goal, the following two Figures present heatmaps of the Shapley values for each feature and its rank in each sequence step.

The first plot confirms the conclusion obtained previously: the general low impacts of the features decrease further after the fifth step (except that of the *inactivity_time*).

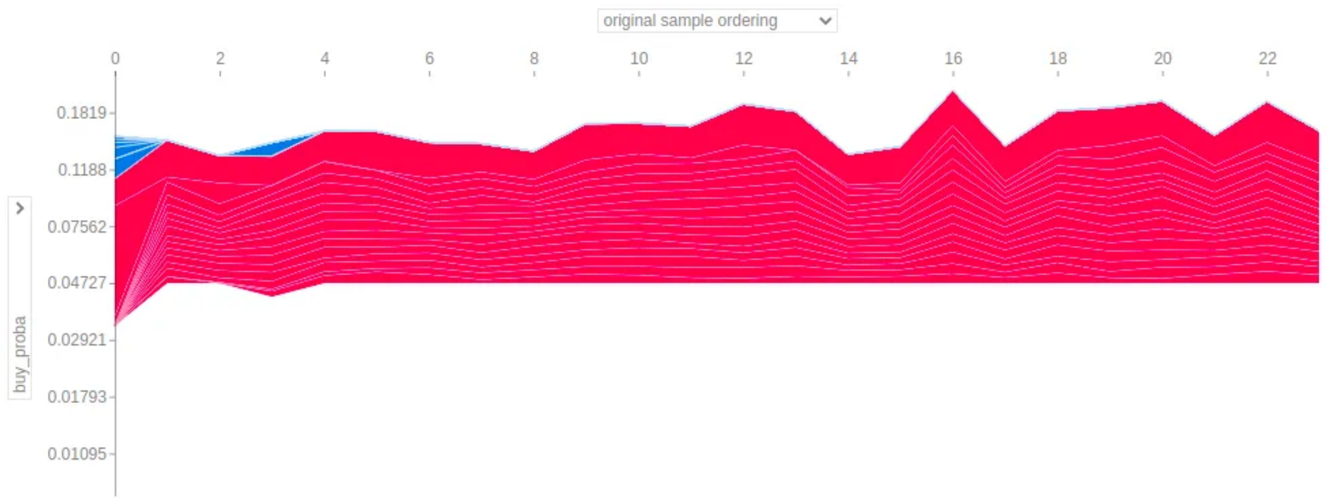
The second figure presents the evolution in terms of the rank of the feature, which is defined according to the absolute value of the Shapley value. While the figure shows a somehow richer behavior, the actual differences between the impacts of all features are rather small.

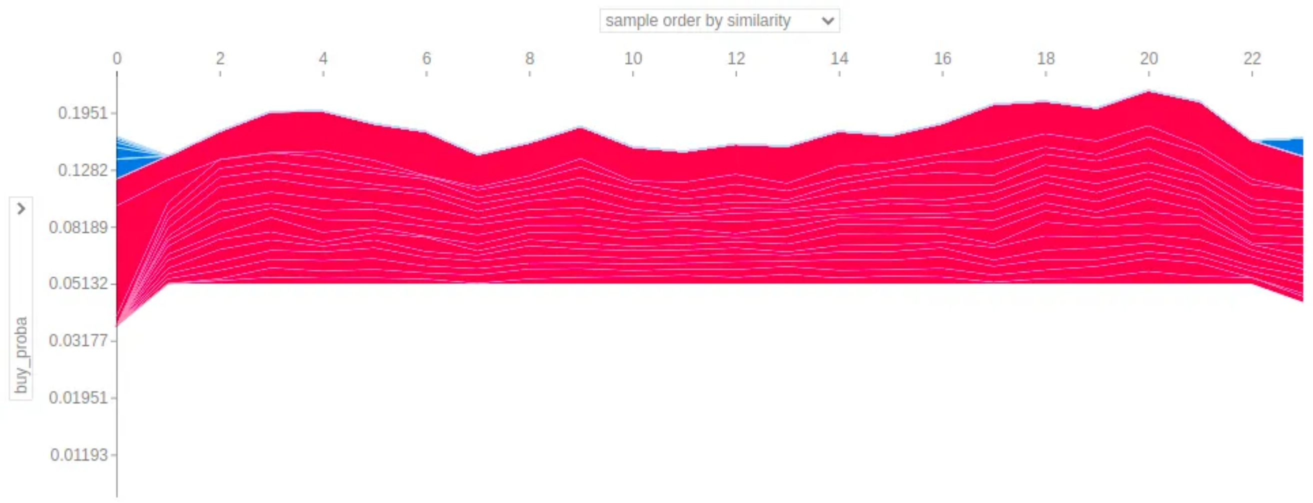


Heatmap showing 1) the Shapley values of each feature in each sequence step; 2) ranking of each feature and in each sequence step. (Image by Author)

By stacking the results of each sequence step, we can visualize them with the *force_plot* function in the package; the case is analogous to that of multiple instances, except that the different instances are the different sequence steps in this case.

The following Figures compare the stacked representations that order the sequences according to the original order and to their similarities based on the Shapley values, respectively.





Stacked explanations for the different sequence steps using the built-in function force_plot: 1) Stacked explanation for the original sequence order; 2) Stacked explanation with sequences ordered by similarities. (Image by Author)

The sequences have been rearranged; thus, some steps that are not necessarily close to each other exhibit similarities. Because the base_value is different for each step, the average value over all the sequences was considered in this case.

So far, we have not considered whether the value of a feature and its impact in terms of the Shapley value could be related. Insight into this correlation can be usually obtained with the built-in summarizing plots available in SHAP. Therefore, the following two Figures present the recreated beeswarm plots; in this case, each point represents a sequence step of the same instance instead of an instance itself.

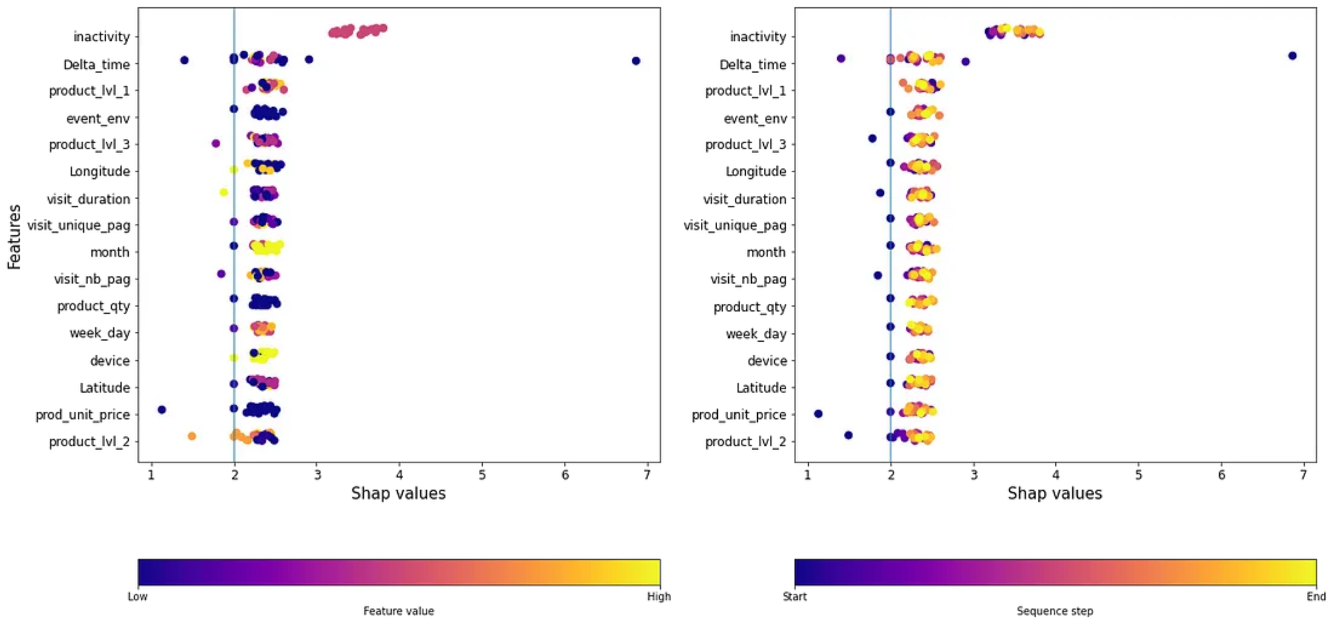
The first one is a replica of the plotting routine in the SHAP package; the Shapley values and features are presented on the x- and y-axes respectively and ordered according to the absolute value of the sum of the Shapley values of all the sequences. The color represents the feature value.

The second figure is almost identical to the first one; however, the colors of the points represent the positions in the sequence.

According to both figures, all the features have mostly a positive impact by pushing the output from the base value to the predicted one.

The first figure does not exhibit any striking relationship between the Shapley values and value of the feature.

There seems to be a more interesting relationship between the Shapley values and sequence step: the greatest the negative impact occurs in the first steps of the sequence, and the extreme values of the *delta_time* are always obtained at the very beginning of the path.



Beeswarm plot with features ranked according to the sum of the absolute value of the Shapley value along the sequence. 1) The color of the points represents the feature value; 2) The color of the points represents the sequence step. (Image by Author)

We can fairly state that for this user, the position in the sequence of events has a major effect on the predicted purchase probability and that the initial step is crucial.

. . .

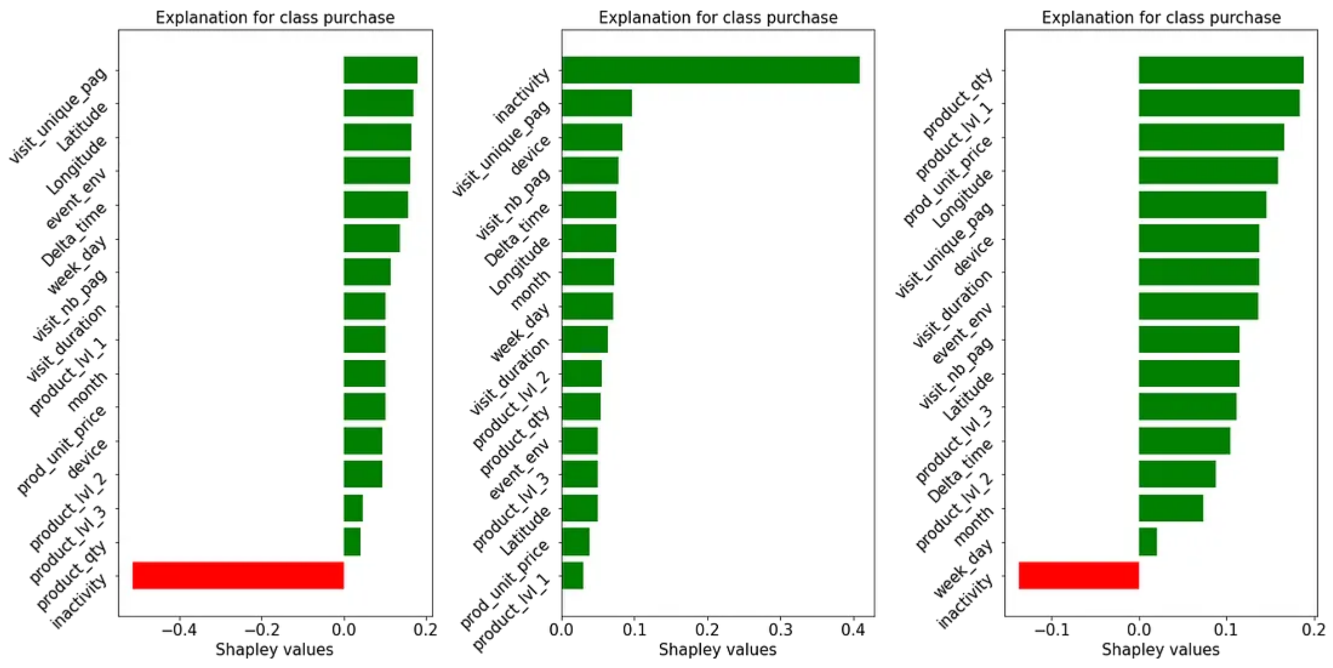
From local to global

A single instance already provided us with a lot of information on the impact of the features on the predictions. In addition, we obtained interesting insights into the role of the sequence step.

What happens if we apply SHAP to several instances?

Thus, we try to obtain insight into the global impact of the features on the model's results by combining the explanations for several users.

Such as before, to obtain information on the impacts of the features we prepared the following Figure for a single explanation for three different users in the final step of their path.



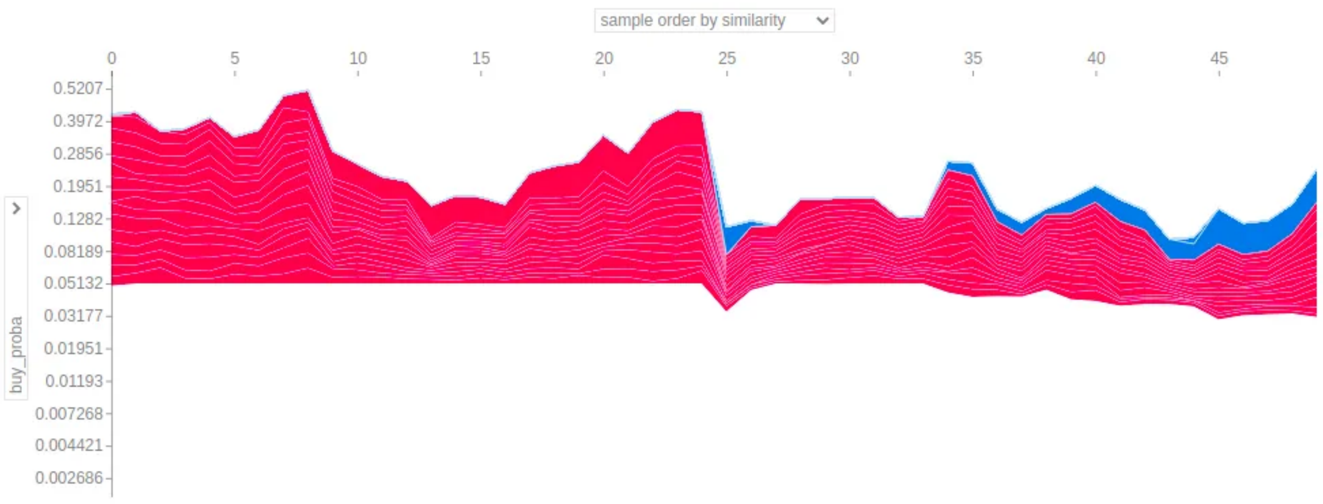
Explanation for three different instances in their last step using SHAP. (Image by Author)

The three explanations differ from each other. In two of them, the *inactivity_time* has a negative impact, which is in contrast to the previous case and recovers the explanation we found using LIME (Please recall the different meanings of the two explanations).

In the last one the *inactivity_time* is not the most impacting feature anymore.

This observation must be more thoroughly investigated if we want to obtain some global insight.

We can start by looking once again at the built-in force_plot, which stacks the different instances in their respective last sequence step. The following Figure presents the presence of two families of instances: in one family, the *inactivity_time* has a positive impact on the final prediction (left), and in the other family, it exhibits the opposite trend (right).



Stacked explanation for the different instances in the last sequence step using the SHAP built-in function force_plot. (Image by Author)

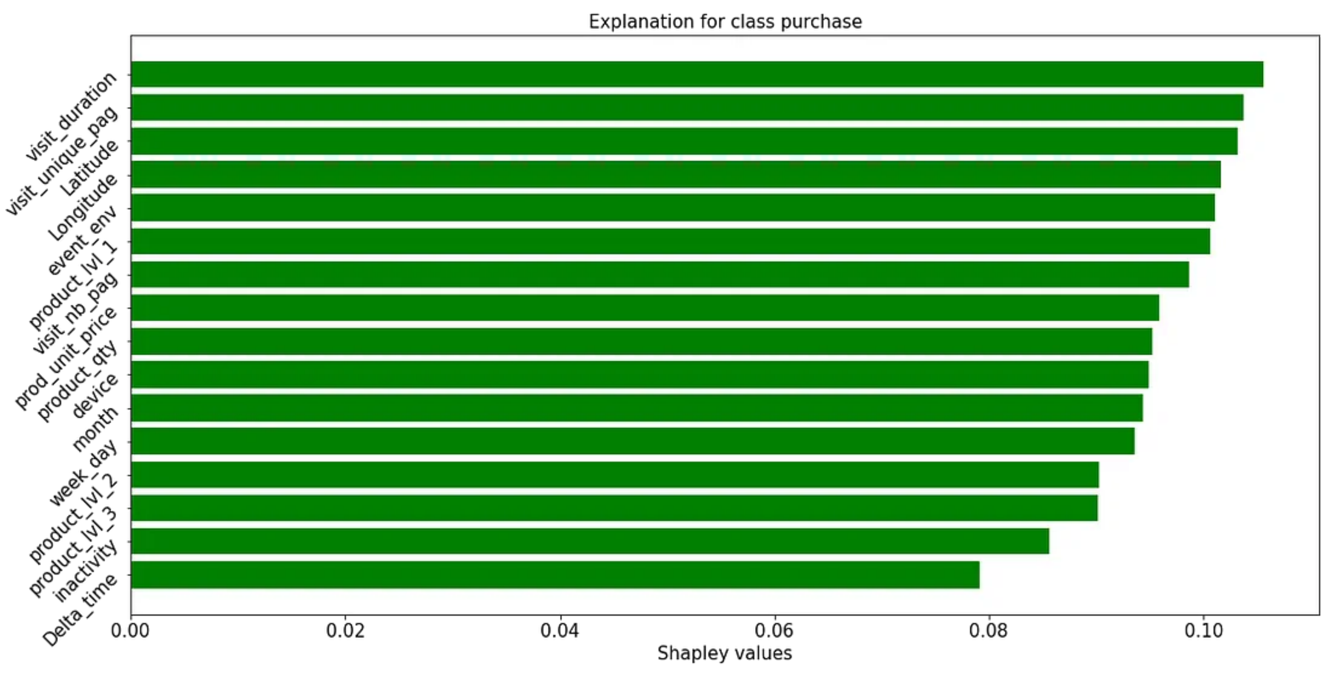
Can we obtain some more quantitative information? What happens in the other steps of the sequence?

Is the *inactivity_time* really the only interesting feature?

These are all interesting questions. In this case, the difficulty is delivering a proper representation of the results owing to the additional dimension (we should present the impact for each feature, at each step and for each instance, including the effects of the feature value on its impact; quite a lot of information to compress in a single figure).

A simple way for obtaining additional insight is averaging the Shapley values over the instances; the result is a representation similar to those presented in the previous section; however this representation refers to the average impact among all the users.

The next Figure presents the impact of the features (averaged over the instances) in the last step of the sequence.

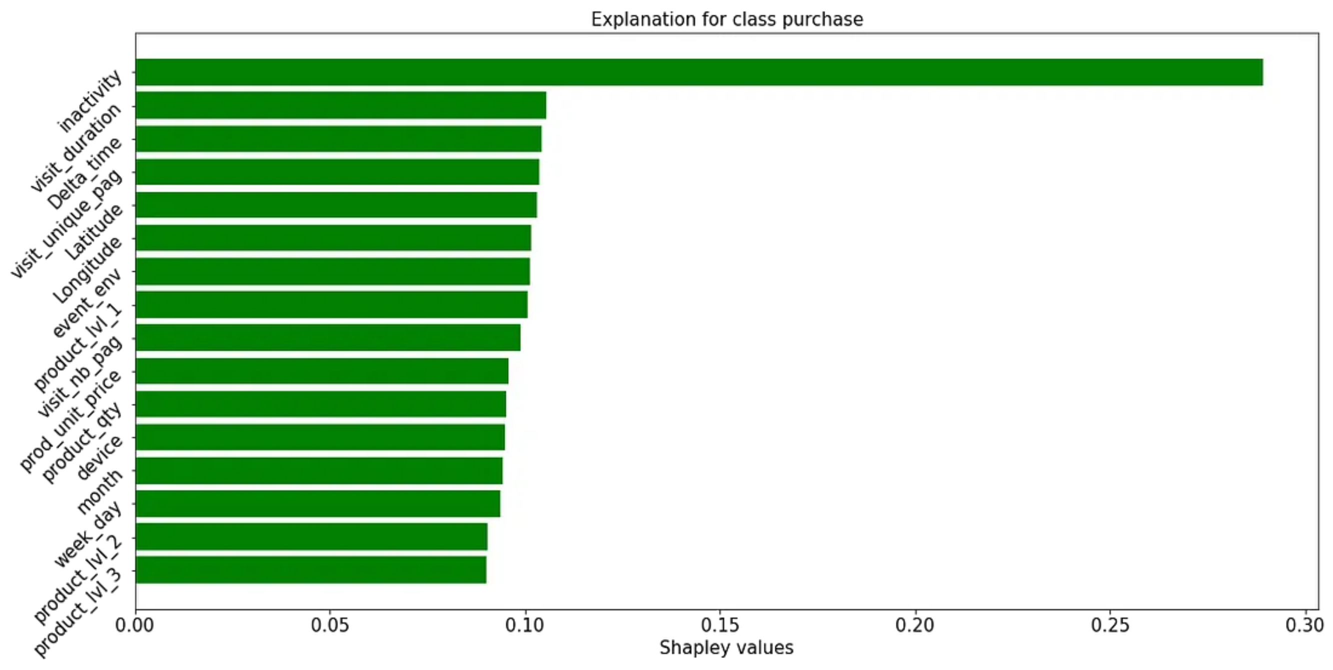


Averaged impact of the features in the last step of the sequence. (Image by Author)

Evidently, the average behavior is significantly different with respect to that of the single instance presented in the previous section; the *inactivity_time* and *delta_time* are the features with the least impacts.

Of course, being an average value, features with negative and positive impacts on different instances will have a net effect that can be lower than that of features with small impacts but identical signs on each user.

More insight into the feature importance can be obtained by looking at the average absolute value.

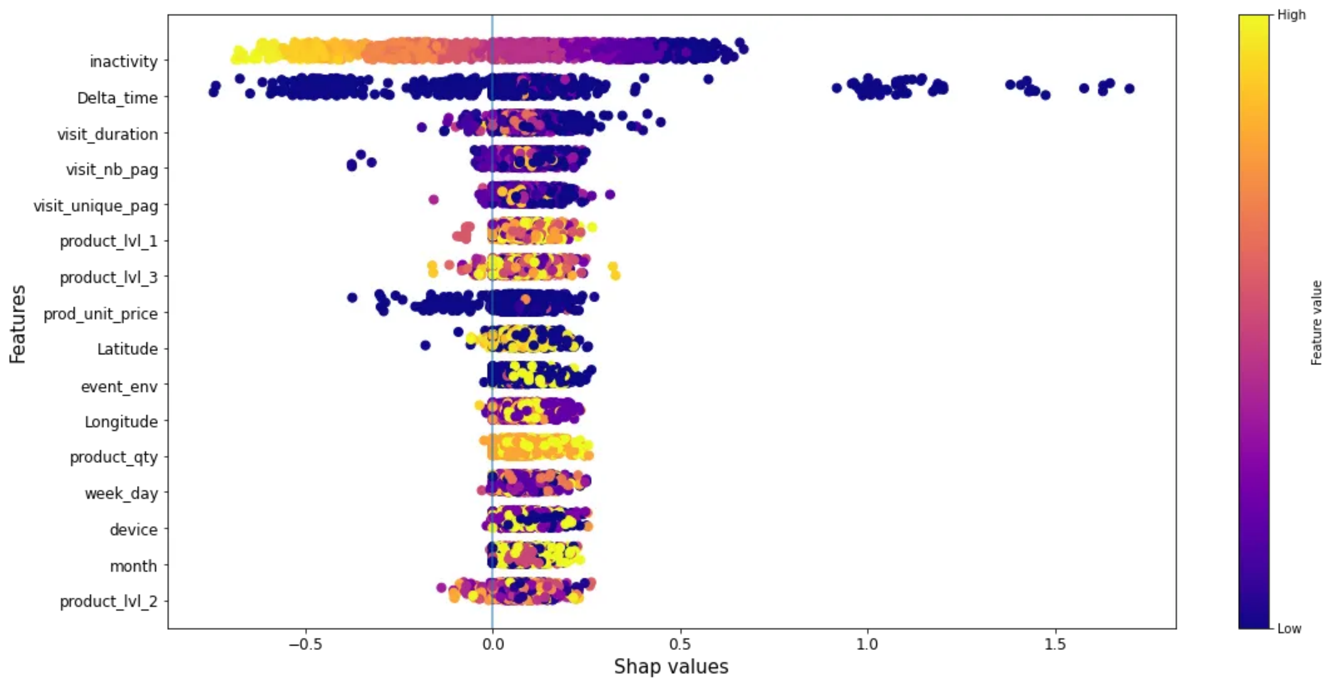


Averaged absolute value of the impact of the features in the last step of the sequence. (Image by Author)

According to the previous Figure, the *inactivity_time* is still the most important feature, and the others have similar impacts in the final step of the sequence of events.

Let's try to move further from the last sequence step and to gain some overall insight into the whole path. We can do this once again with the summary plot.

The next Figure presents all features for all instances in all sequence steps; the color represents the normalized feature value.

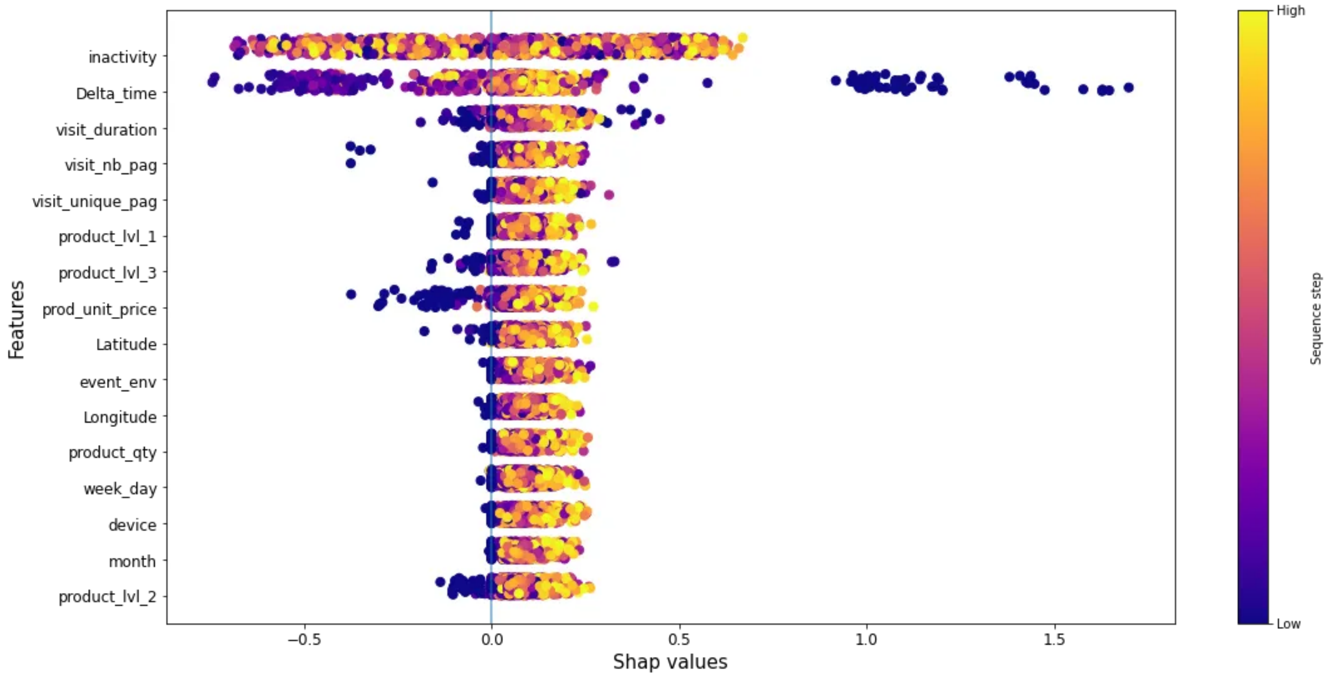


Beeswarm plot with features ranked according to the sum of the absolute values of the Shapley values over the users and sequences. The points represent each instance in each sequence step. The color of the points represents the normalized feature value. (Image by Author)

While most features do not exhibit an evident relationship between the Shapley values and feature values, this is not the case for the *inactivity_time*: evidently, the higher the value is, the more negative the impact is.

We discover the intuitive relationship between the *inactivity_time* and predicted purchase probability: the older the last interaction of a user is, the more negative the impact on the prediction of a purchase is.

As in the previous section, we can present the results by explicitly considering the positions along the sequence of events.

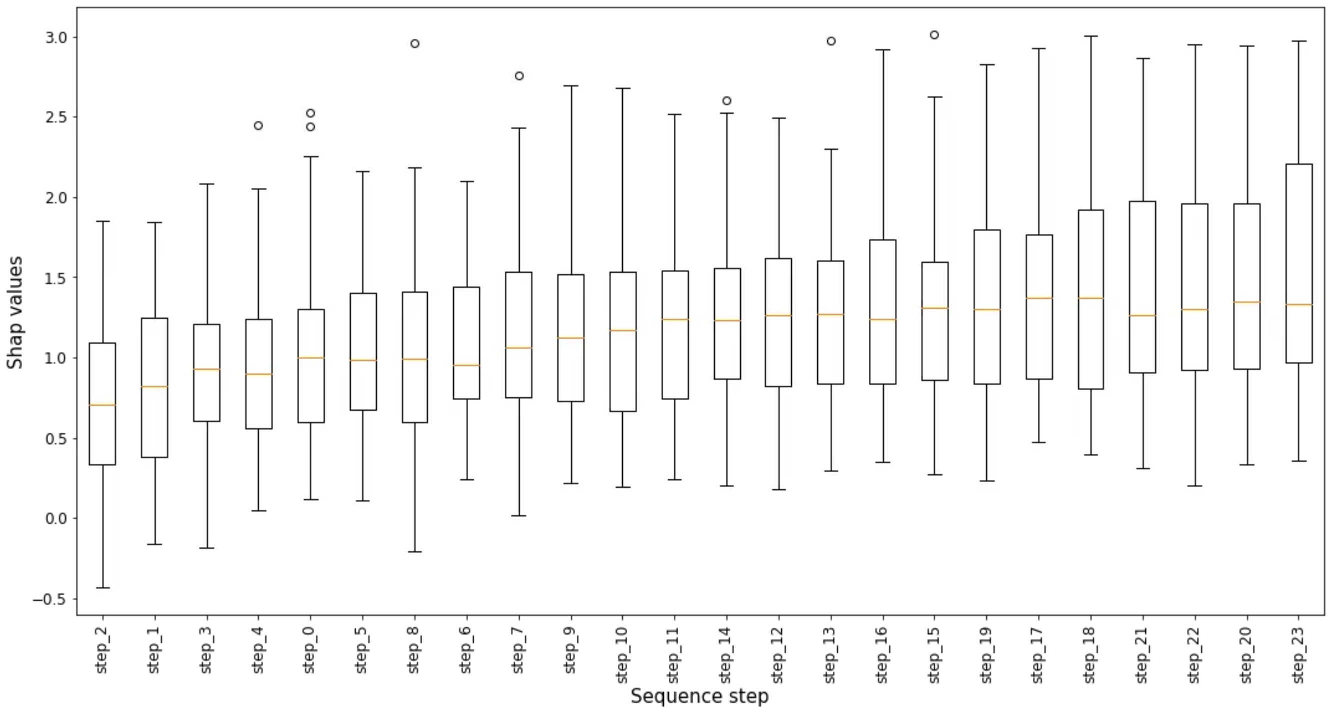


Beeswarm plot with features ranked according to the sum of the absolute value of the Shapley values over users and sequences. The points represent each instance in each sequence step. The color of the points represents the position along the sequence. (Image by Author)

In this case, most features exhibit a clear pattern, except the *inactivity_time*. These results seem to confirm those of the previous section: the features have negative Shapley values at the beginning of the sequence and then shift toward a positive impact during the development of the sequence.

So far, we have focused on the features and the impact in different steps and for different input values. Let us consider the sequence itself. Does it have an impact? (The answer is “Yes”: we have seen a significant difference between the initial and last steps in the sequence).

To gain additional insight into the impact of the individual position in the sequence we prepared the following Figure. We summed the Shapley values of the different features in each step. The resulting figure presents the distribution among the different instances; the sequences are ordered according to their importance.



Box_plot with sequences ranked according to the sum of the absolute values of the Shapley values over the features. (Image by Author)

Again, the farther we go along the sequence, the greater is the positive impact; the most significant impact in terms of the absolute value of the Shapley values is in the last steps.

. . .

We have dive a bit in the potentiality of SHAP in a real-world application involving the use of recurrent neural networks.

We have found similar difficulties as in the application of LIME when applying the explainer to this kind of neural network. In addition, we find the need to modify the preprocessing pipeline in order to better fit to SHAP.

This was far from straightforward but the effort was worthwhile.

> The explanatory capabilities of SHAP are impressive.

It might be less intuitive than LIME, and still computationally expensive for what concerns the KernelExplainer; however the compositional properties of the Shapley values allow establishing rich and variegated types of explanations.

Establishing insightful explanations for the three-dimensional case of a recurrent neural network can be challenging; however, Shapley values provide the fundamental bricks for reaching this goal. The remaining aspects depend on the requirements of the final user and his/her creativity.

. . .