PRO feature attribution documentation

Author: Jiaxuan Wang
Date: 3/28/2023

Overview

Predictive Response Optimization (PRO) is a machine learning system using reinforcement learning to fight unauthorized scraping. For IG logged in, PRO decides what responses to issue to users who are suspected to be scrapping. For IG logged out, it decides whether to block requests from an ip address. Despite its importance, for long, we treat PRO as a black box, making it hard to understand its logic to debug the system. Feature attribution is one approach to fill this void.

Feature attribution has the following benefits

- It improves model understanding and enhances defensibility of our decisions
  For assessor requests for safeguard evidencing we often get questions around why a particular response was chosen for a user and the most important features which influence PRO's decision. With feature attribution in place, we can answer those questions (see use case 1).
- It helps monitoring distribution shifts in the production traffic
  This will aid debugging when response distribution spikes or when online MSE degrades (focus our attention on the important shifting feature, see use case 2).

The purpose of this wiki is to a) explain how we compute feature attribution so that you have the necessary terminology to understand our feature attribution dashboard (methodology section), b) showcase a few use cases of the dashboard (example use cases section), and c) document all the tools provided by the dashboard (reference section).

Feature attribution dashboard ("go pro_attribution") currently supports

- **A**: Visualizing most important features
- **B**: Tracking feature attribution and value shifts
- **C**: Visualizing decision logic (attribution over value)
- **D**: Monitoring the volume of logged feature attribution

for both IG logged in and out. Furthermore each of the aforementioned features can be filtered by action and context features.

There are additional contexts on design choices and tools built around feature attribution besides the dashboard, which are tracked in T128501575.

Methodology

We use SHAP[1] to obtain local feature attribution, that is attributing PRO's decision for each sample to the features used by PRO. For example, to answer the question why an ip got blocked for IG logged out, local feature attribution could reveal that the ip has too many requests for the day and that's why PRO blocked it.

We implement SHAP online on WWW. That is feature attribution is computed at the same time a live decision is made by PRO. Due to the expensive nature of computing SHAP, we only compute it for a small proportion of input.

Q: Why do we use SHAP for feature attribution?
A: We choose SHAP because it is model agnostic. PRO not only has categorical input, but also the business logic on top of the model's output (e.g., cooldown, allowed_actions) makes PRO's decision surface non smooth. We therefore cannot turn to methods such as integrated gradient or CAM that assumes access to gradient. Furthermore, being model agnostic allows our approach to continue working when the underlying machine learning model is changed. In addition to being model agnostic, SHAP is widely used and has game theoretic interpretation stemming from Shapley Value.

Q: How to compute SHAP?
A: In short, the attribution of a feature in SHAP is the difference in output when adding the feature to a set of features, averaged over all possible feature sets. For example, denote v as the decision function for PRO that takes a set of input containing two features A and B, the SHAP value of B is (v({A, B}) - v({A}) + v({B}) - v({})) / 2. In other words, we try all the ordering in which B is present in the input set and observe the difference in output when excluding B from the set. In reality, we don't work with set functions. We usually have a function f with two inputs A and B. Then, what does it mean to exclude B in the formula above? Well, it means we define a value for A and B representing the case they are "excluded". That is we compute (f(A, B) - f(A, B') + f(A', B) - f(A', B')) / 2 for B's attribution, where A' and B' are called the background/reference/baseline value of A and B, used to contrast with the actual value of A and B. The background values are usually set to some typical values of the feature. In the case of PRO, we set background value to 0 for continuous features and "" for categorical features. In the future, we may support letting developers change the background value. I prefer explicitly setting this value because it is clearer what SHAP is computing. If we define this background as expected value, it shifts through time, which makes it hard to see and interpret the meaning of feature attribution. In our case, the meaning of attribution for a categorical/continuous feature is the difference in PRO's decision (1 if the decision changed, 0 not) when changing its value from

[1] As a technical aside, there are a few variants of SHAP (e.g., conditional SHAP and independent SHAP), we implement independent SHAP as it is simple to implement and has causal interpretation (see Janzing et. al.; it turns out all those variants can be unified with a casual graph, therefore if we want to try other variants later, we can: see Jiaxuan et. al. or ask me for details).

its current value to “”/0 respectively, averaged over the on or off state of other features (i.e., whether other features take the background value).

Q: Why was SHAP implemented online in WWW?
A: Implementing SHAP online guarantees the feature attribution uses the same settings of the PRO (e.g., metric weights, allowed actions, exploration factors, etc.) when a decision is made. If we delay the feature attribution to offline, we need to make more infrastructure changes to ensure the consistency of attribution settings, which is non-ideal for maintenance purpose (say someone added a new logic for PRO in WWW, we need to add the same logic offline to make feature attribution consistent with it). Furthermore, computing SHAP online has a computational benefit as we don’t need to call through the network when querying PRO’s feature stores repeatedly.

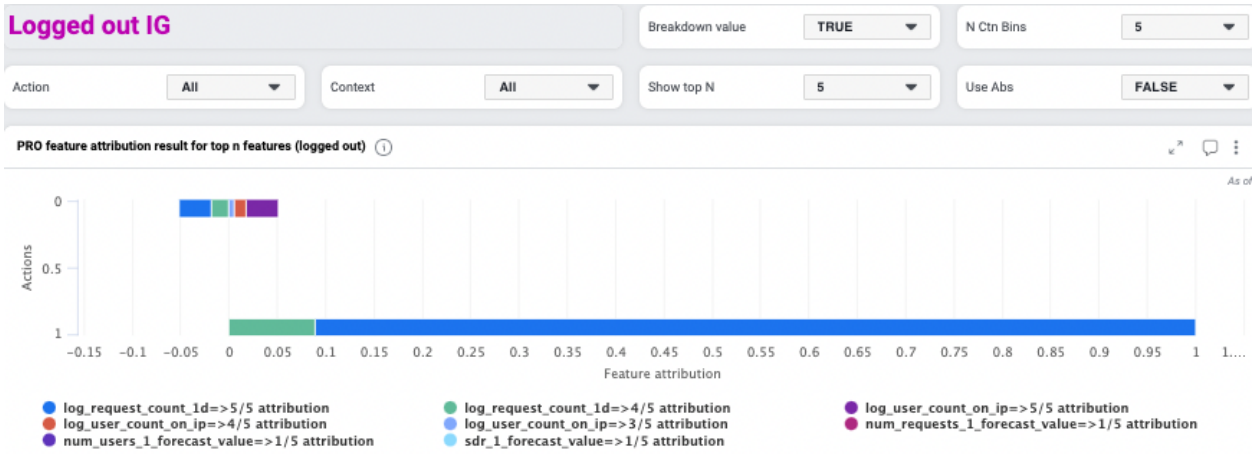
Keep in mind that SHAP as well as any explanation methods are no more than summaries of the decision surface. It won’t test for cases that are far from the training distribution. However, a complete understanding of the decision surface is nearly impossible with complex models that have lots of features. If one aims for a complete understanding, we need to change model structure to something inherently interpretable (e.g., linear model with no input interactions, low depth decision tree, GAM, shape constrained models, etc.), though it is also likely to restrict the expressiveness/accuracy of the model itself.

Example use cases

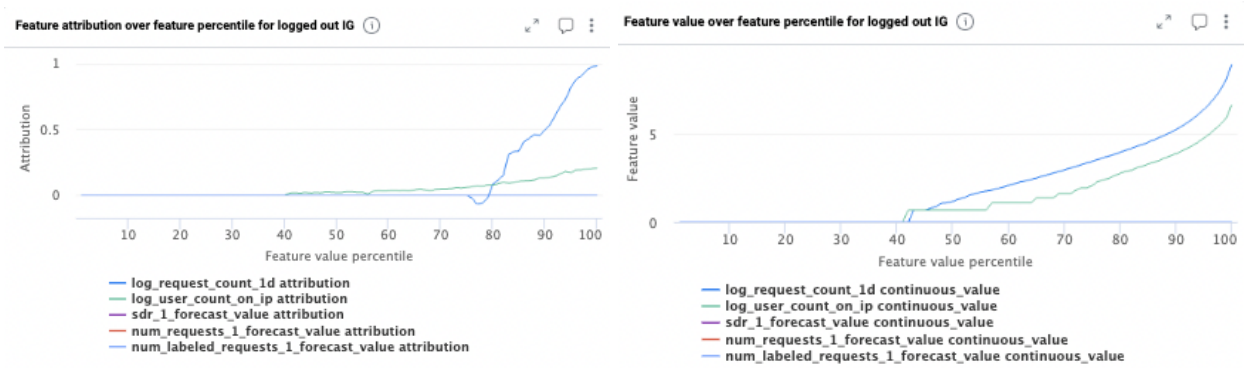
This section gives 2 use cases that I believe can be useful to oncalls or anyone interested in understanding PRO’s behavior.

Understanding PRO’s decision logic

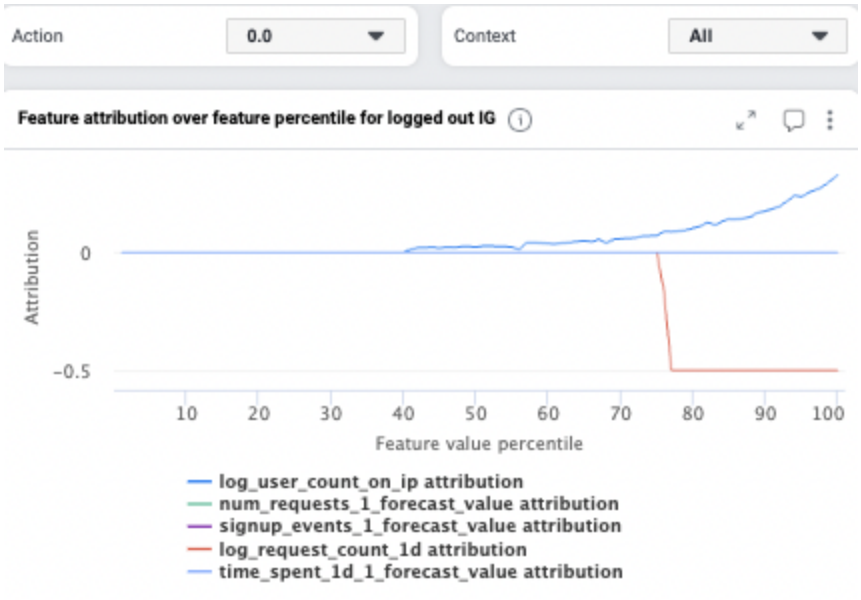
For understanding the decision logic of PRO (e.g., what was the criteria PRO used to block an user?), one can use panels **A** and **C**. **A** shows an overview of important features, while **C** dives into one feature and shows you what values of this feature are important. For example, looking at the logged out feature attribution panel **A** below, we see that “log_request_count_1d” is the only important feature for deciding whether to block an ip (i.e., action 1.0). Furthermore, we see that only high values of request count triggers the blocking action (a note on notation: “log_request_count_1d=>⅘ attribution” means the 4th out of 5 equally spaced bin for the continuous feature log_request_count_1d). On the flip side, when focusing on action 0.0, we see high request count decreases the chance of getting no response.



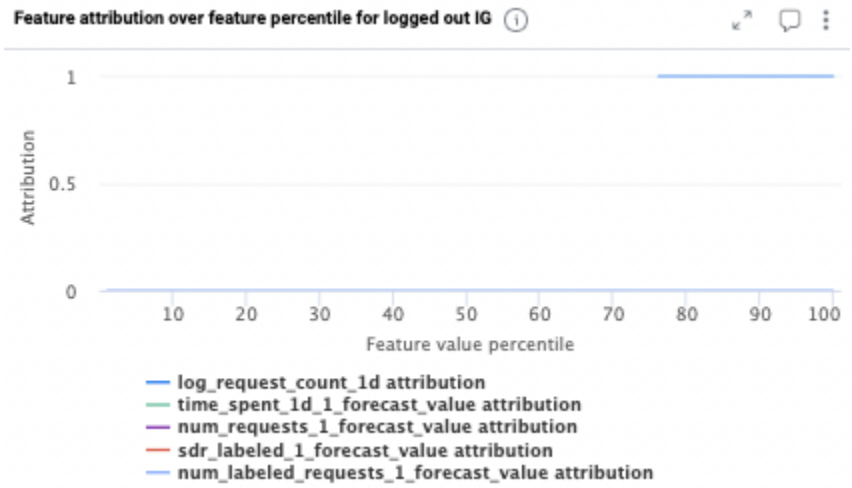
To further understand how the decision was made, panel **C** below shows us that "log_request_count_1d" is only important when it passes 76 percentile (reading from the left plot, 76 percentile along the x axis is when attribution starts to be non zero), which corresponds to 3.77 "log_request_count" (reading from the right plot, which maps from feature percentile to its actual value).



Now if we zoom in to samples with no response (i.e., action 0.0). We see that "log_request_count_1d" decreases the likelihood of issuing no response at the 76th percentile. In other words, it increases the likelihood to block once request count is high (confirming our insight from the overview plot). On the other hand, we see that high user count on ip increases the likelihood of PRO to issue no response, which comes at no surprise as more user count on ip justifies increases in requests coming from the ip.



We can repeat the exercise, filtering for samples that are blocked (i.e., action 1.0). We see again that request count is 100% responsible for choosing the blocked action[2].



In summary, all the above plots show us that PRO blocks ip addresses with high request count while accounting for number of user count on the ip.

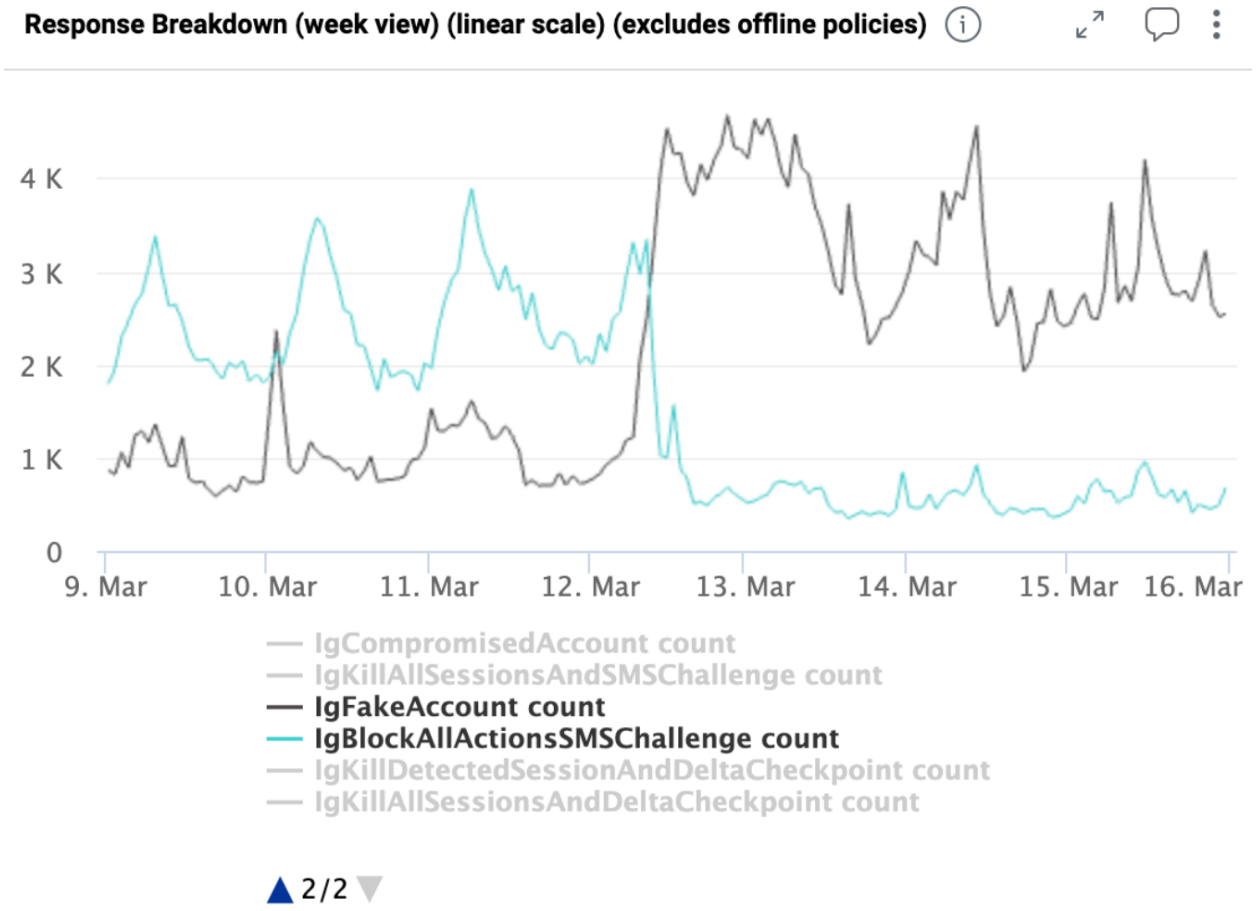
Understanding spikes in responses

Often, we want to understand irregularities in PRO's behavior, such as spikes in some of its recommended actions or degradation in online MSE. Feature attribution can be used to locate

[2] You might have noticed that the attribution plot for the block action has no value from 1 to 76 percentile because no samples that get the blocking decision have value in that range. If we want to see the attribution for all actions regardless of the decision made, we need to repeat doing SHAP for all actions (currently only doing attribution on the action chosen), which will set the computation cost at (number of action - 1) times the current computation.

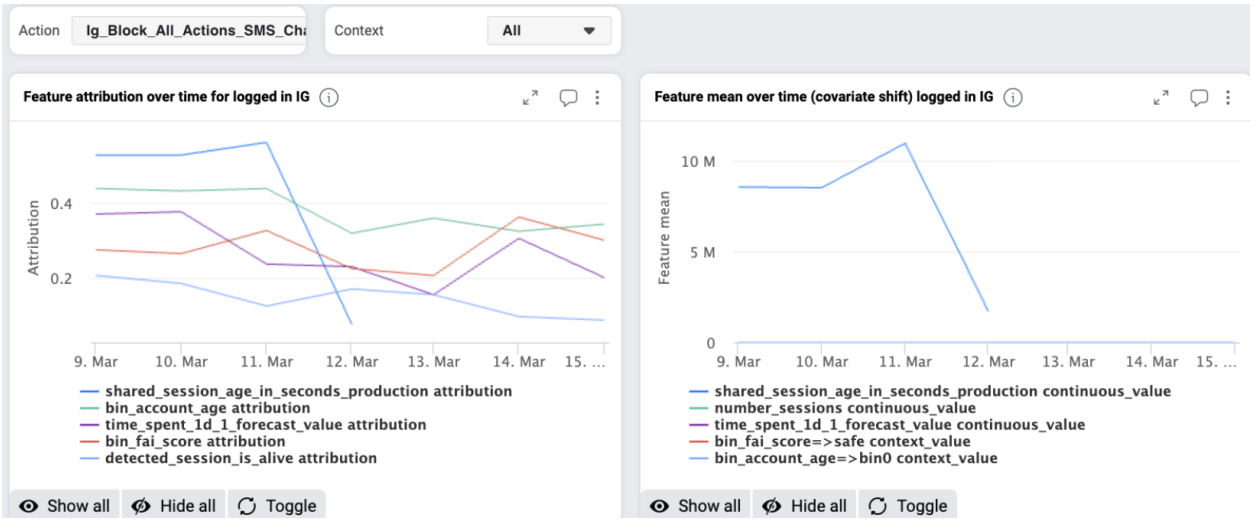
changes in important features that correlate with PRO's irregularities, thus speeding up the investigation of root cause (e.g., shift in input caused by upstream classifier degradation; shift in model logic due to addition or deprecation of features). The cause of irregularities along time are either feature/covariate shift (i.e., input signal changes) or conditional shift (i.e., model logic changes), which are both tracked in panel **B**.

Take understanding spikes in actions for instance (e.g., why we blocked much more people today than yesterday), on 3/12/2023, the intervention team[3] observed a sudden spike in IgFakeAccount and a drop in IgBlockAllActionsSMSChallenge.



Looking at panel **B** shown below for IgBlockAllActionsSmsChallenge, it is evident that the feature "shared_session_age_in_seconds_production" was important before 3/12/2023, but no longer shows up as important later on (left figure tracks attribution mean for top features across time). Not only did its attribution change, looking at the right plot (which tracks the feature mean across time), we can see that the mean shifted for this feature as well.

[3] Thanks Priya Krishnamurthy for flagging this spike



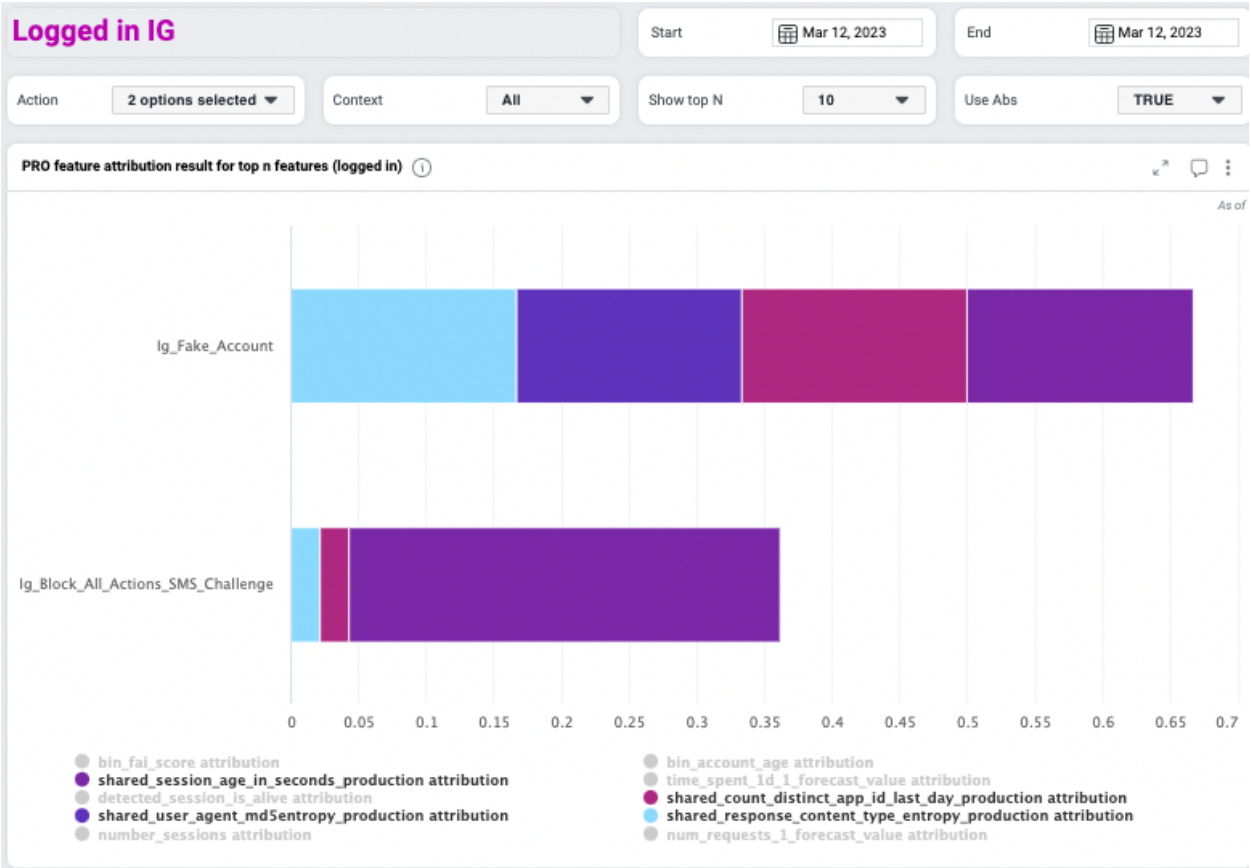
It turns out the feature was removed from the model due to capacity issues in training (D43843906) along with all other features starting with “shared”. The removal of the feature was made on 3/6/2023 but apparently the model was still using it until 3/12/2023. Without panel **B**, we would have taken longer to locate this change.

A few points to observe

- We would’ve located the change entirely based on the drop in attribution, regardless of shift in feature mean (right graph).
- Note that the right graph tracks the feature mean for top features sorted by absolute value in attribution. This complements previous dashboards (real time feature monitoring dashboard and end to end interpretation dashboard) to track feature mean. Filtering by feature importance makes users easier to consume the result. Only important features affect PRO by definition.

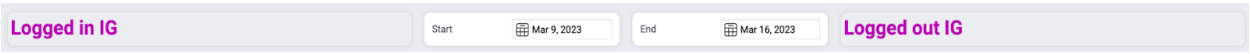
We can confirm the importance of the deleted features by focusing on the two actions Ig_Fake_Account and Ig_Block_All_SMS_Challenge on 3/12/2023. 4 out of the top 10 features for those two actions have been deprecated, therefore it is not surprising that the two actions’ response count changed the following day. However, our tool doesn’t tell you whether the

response count will go up or down.



Dashboard panels reference

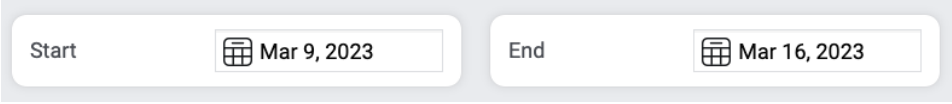
This section describes in detail what each plot in the dashboard does and motivates their usage. Currently, the feature attribution dashboard supports visualization of attributions for IG logged in and out. Panels related to IG logged in/out are listed on the left/right side of the dashboard respectively.



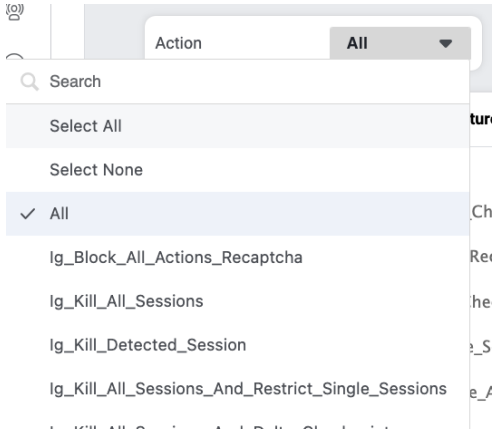
Knobs: Adjusting the aggregation criterion for panels

For analysis, we are often interested in only a subset of users/ip addresses that were actioned upon by PRO (e.g., only aggregate feature attribution across users given lg_Fake_Account who are detected by a particular policy). Tools in this section enable various ways to slice the data.

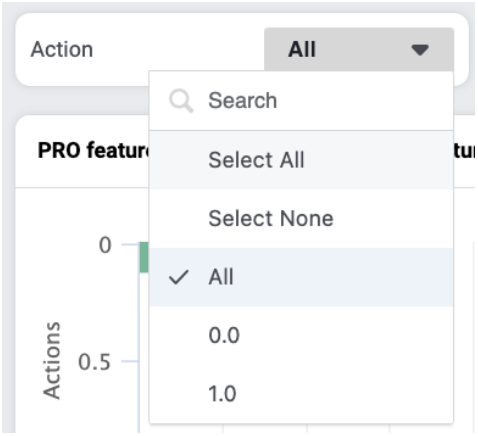
Dates: Start and end dates adjust the time range to pull data from



Action: Recommended action by PRO (left: IG logged in, right: IG logged out). For IG logged out, 1.0 means block and 0.0 means no response.

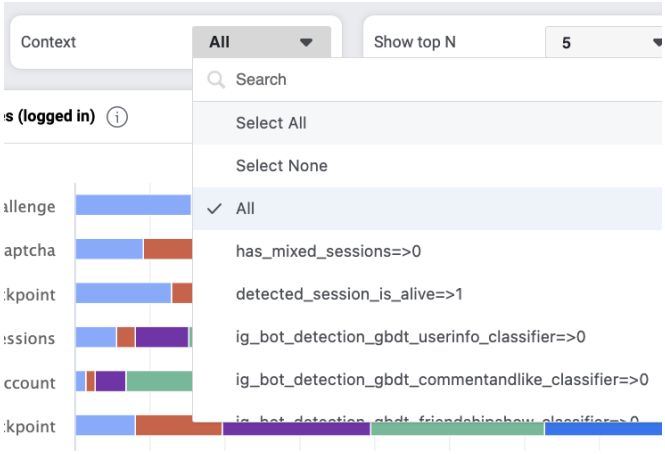


IG logged in

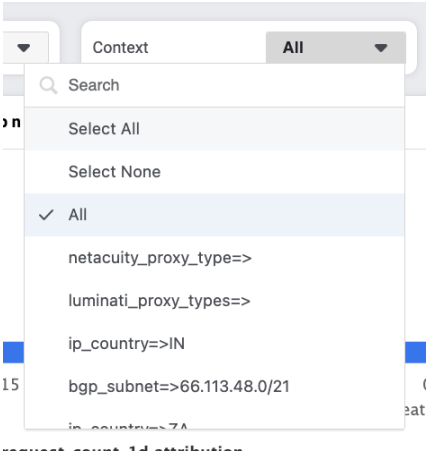


IG logged out

Context: Filtering based on contextual/categorical features. If selected, only aggregate for samples with the selected feature value. The dropdown menu has items in the format of “featureName=>value”. For example selecting “ig_bot_detection_gbdt_userinfo_classifier=>0” means aggregating over samples where ig_bot_detection_gbdt_userinfo_classifier did not fire.

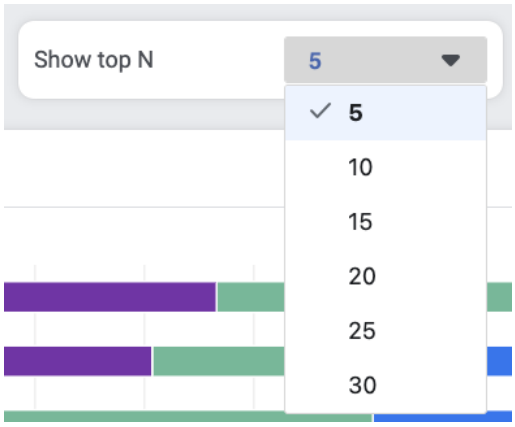


IG logged in

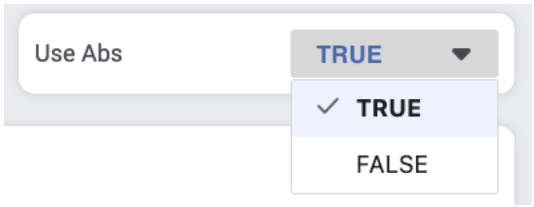


IG logged out

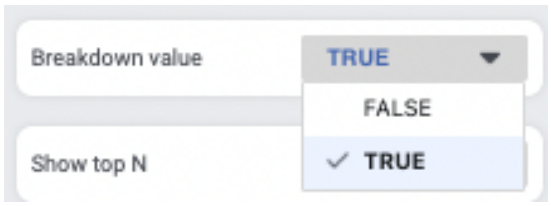
Top N: Adjust the number of features to show (sorted by aggregated attribution). Useful to declutter your graph.



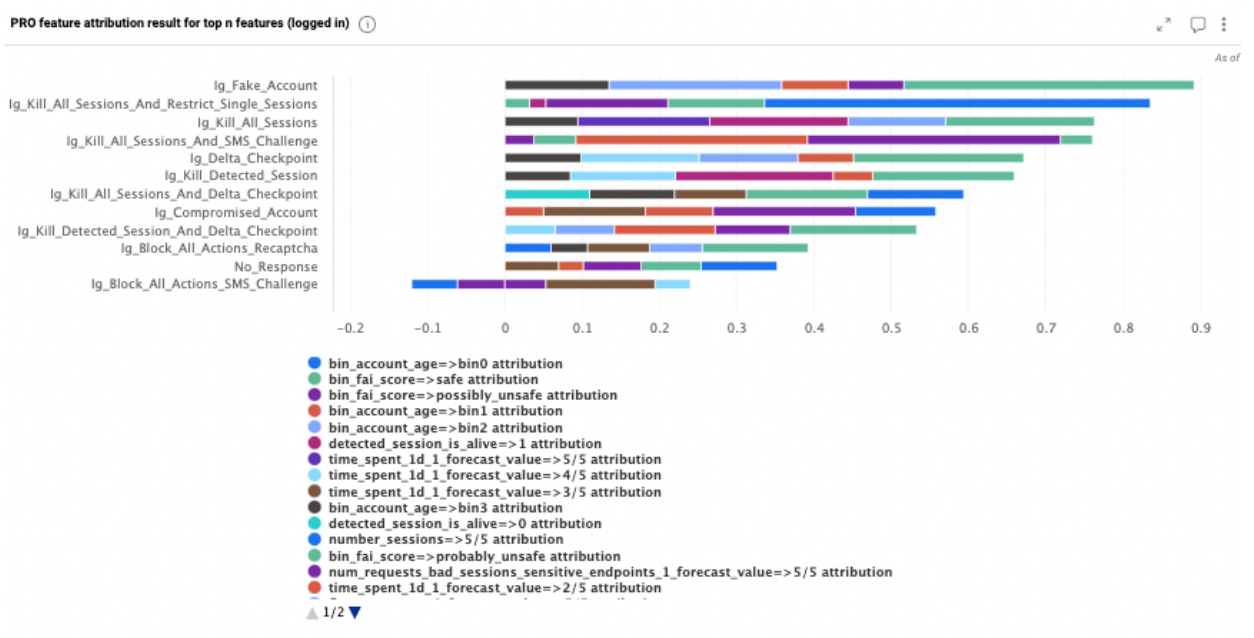
Use abs: Whether to aggregate on the absolute value of the feature attribution. Setting it to true to show feature importance. If false, use the original feature attribution (useful to observe the direction of impact in panel **D**: e.g. does setting bin_fai_score to “safe” increase or decrease the chance of the user getting UFAC).

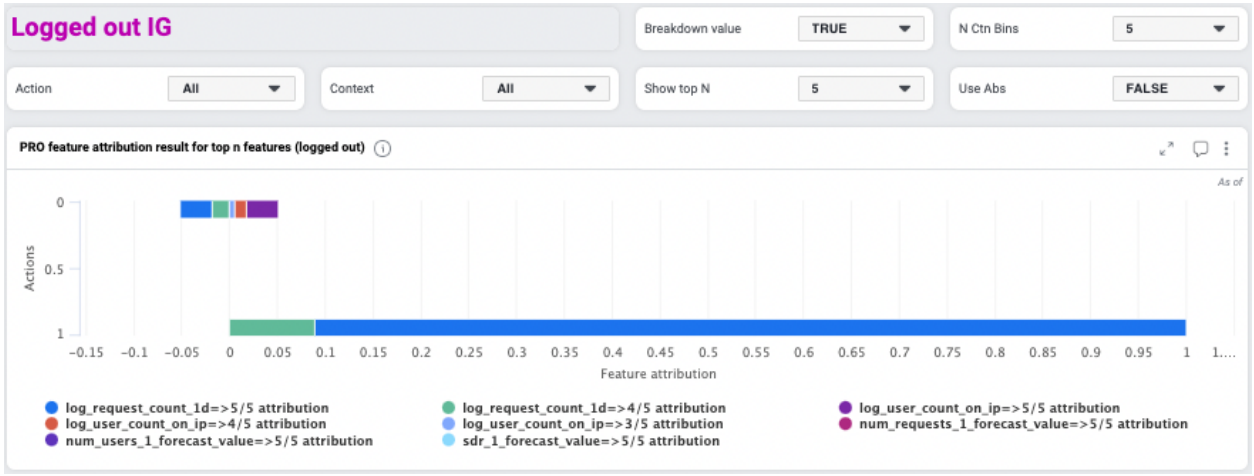


Breakdown value: Adjust whether to breakdown panel A by value

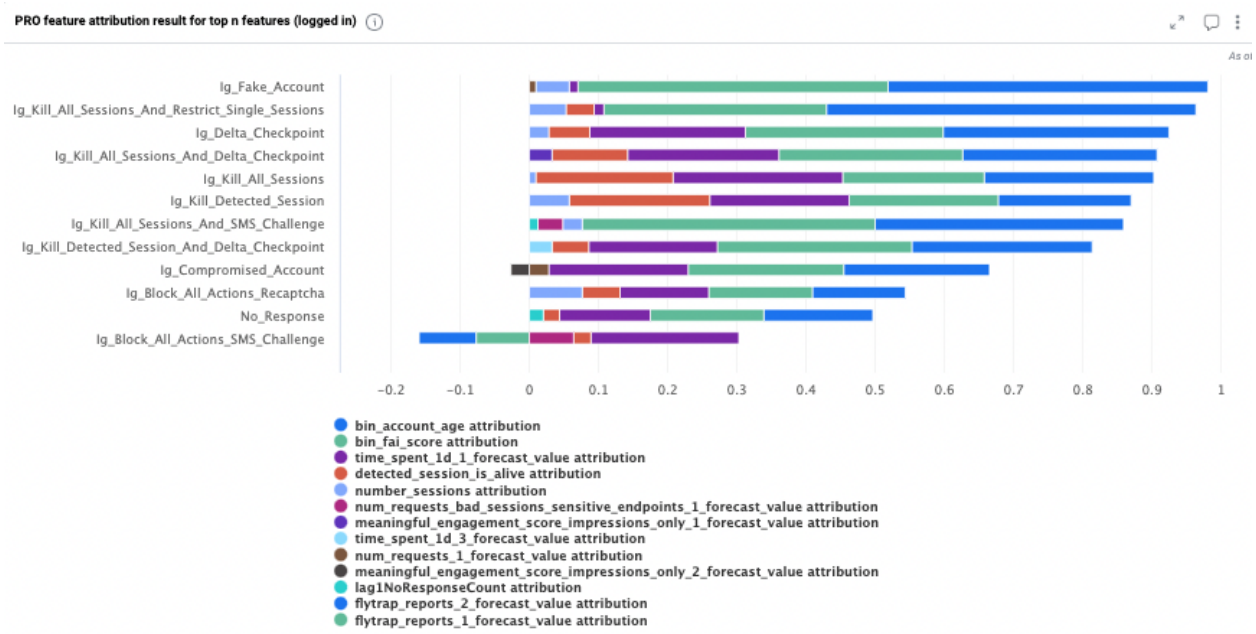


If set to true, we see that panel A shows attribution broken down into values of features. For example, for categorical features like “bin_account_age”, the attribution is on each value of the feature. For continuous features like “time_spent_1d_1_forecast_value” (i.e., the forecast value for the time_spent_1d metric with prediction horizon of 1 day), the attribution is on each binned value. As shown below, the continuous values are broken into 5 bins (e.g., “time_spent_1d_1_forecast_value=>3/5” means it is the 3rd equally splitted bin out of 5 bins). The total bin value can be adjusted as well.

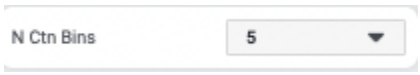




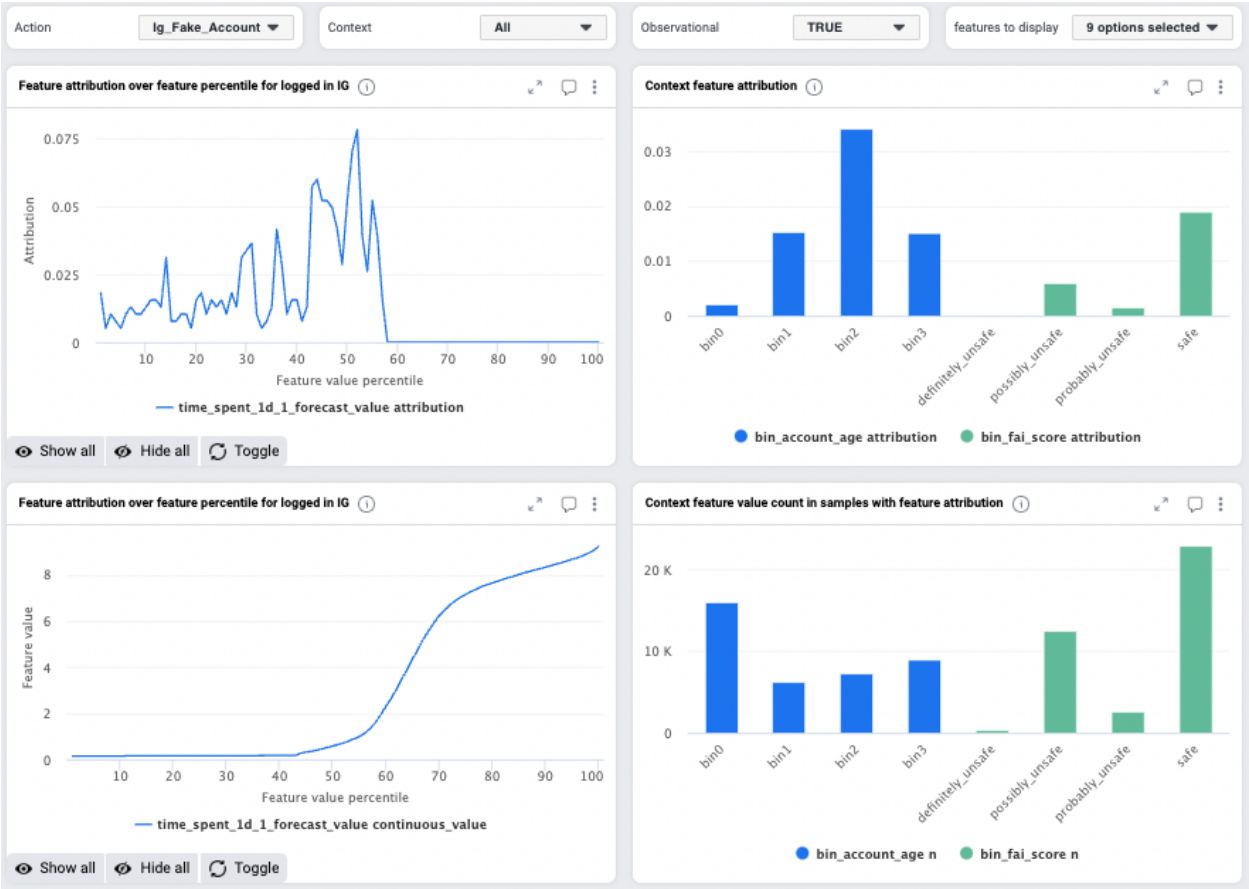
If set to false, we only show the feature names but not the values.



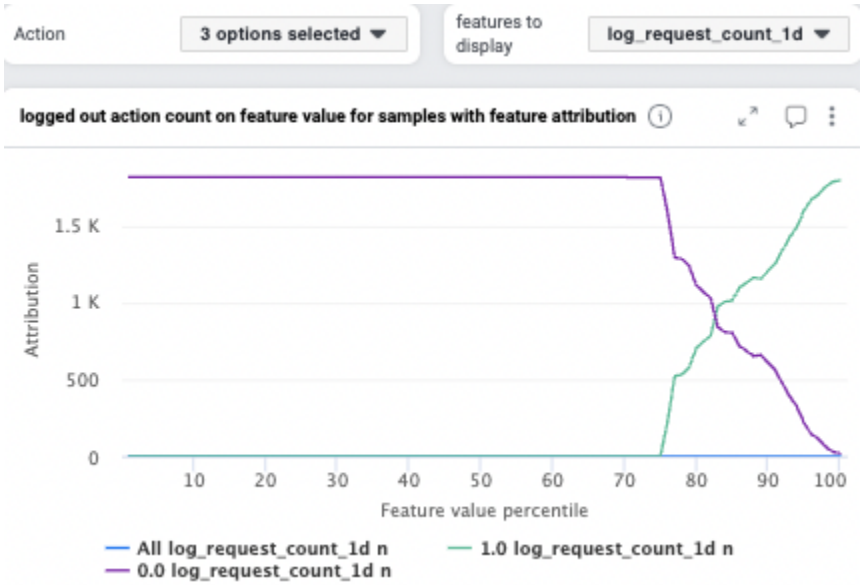
N continuous bins: Adjust the number of bins to split continuous features when breakdown value is set to true in panel A.



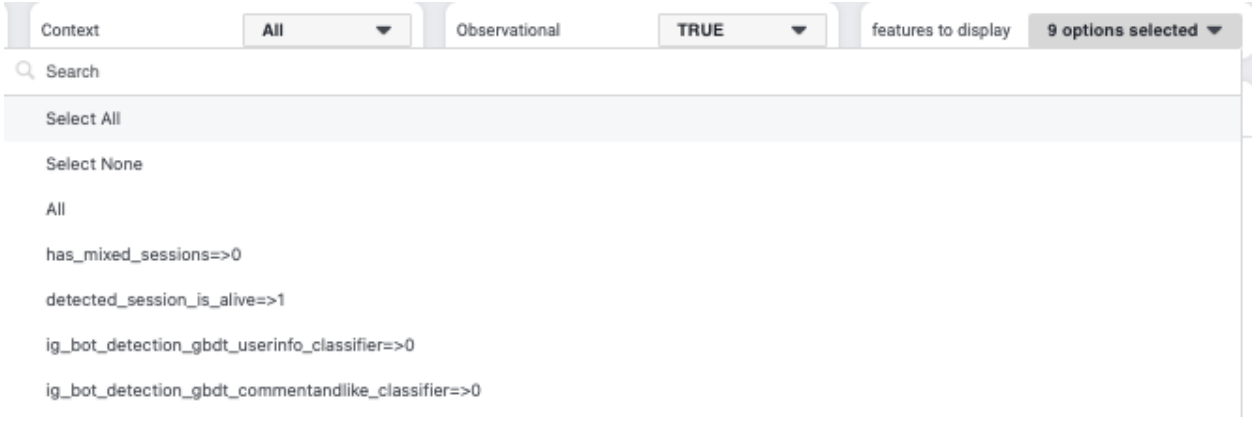
Observational: In panel C, if set to true, we will ignore the attribution based on SHAP computation but show empirical response distribution in panel C. This function is useful to visualize the conditional distribution of Y|X. In contrast, when set to the default of false, we are visualizing the interventional distribution Y|do(X). The difference is that the former is correlational while the later is causal. Note that we cannot average the observational attribution to gauge the importance of the feature as we do for interventional attribution b/c it will just result in the probability a particular response is given (same for all features therefore useless).



Here it shows that low time spent and account age in bin2 are likely to get a high percentage of UFAC responses. To further help one get insight using the observational distribution, we also have a plot of empirical count of action applied through PRO through the angle of feature values shown below.



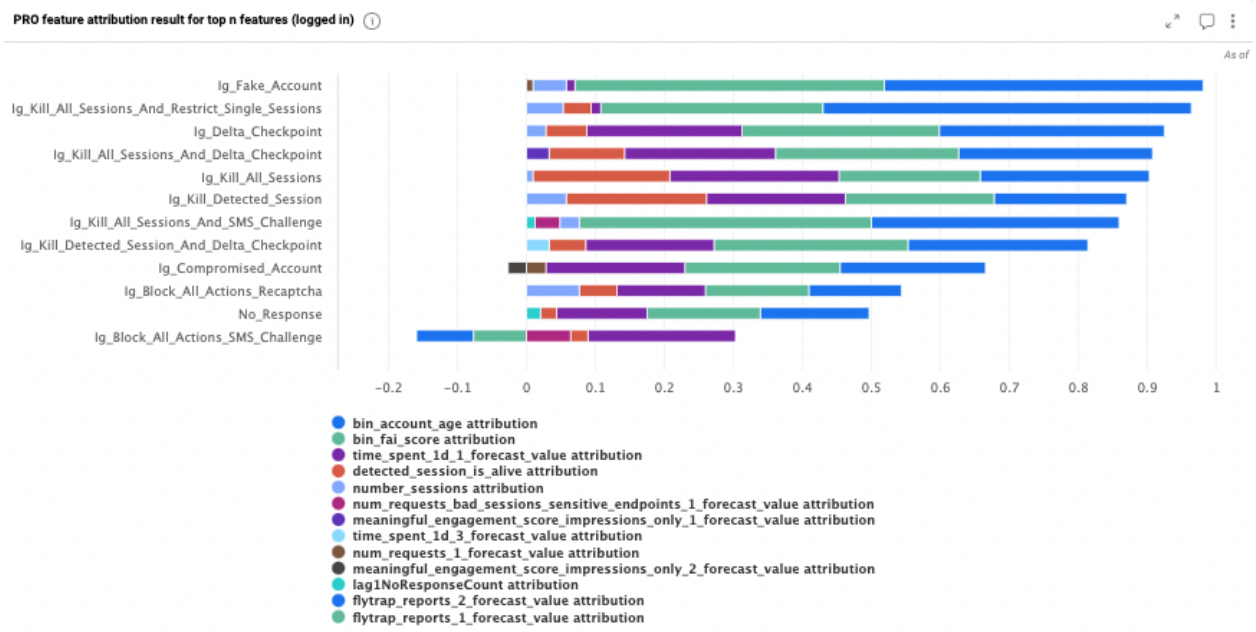
Features to display: Only show the selected features in panel C.



Panel A: Visualizing important features

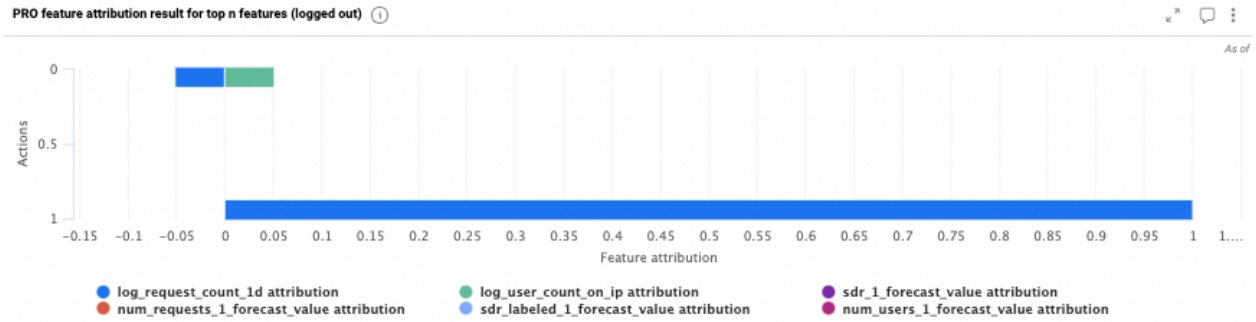
This panel shows an overview of feature importance (i.e., mean value of feature attribution aggregated for all samples with non null feature attribution). The Y axis shows the responses/actions given and the X axis shows the aggregated feature attribution for the top N features with the highest feature attribution stacked. This panel is useful to get a high level understanding of what PRO relies on. One can use this information to consult with domain experts whether the important features are robust to distribution shift. One can adjust the number of top features to show per response, whether to break down attribution by feature value, use abs transformation on the attribution, number of bins for continuous features as explained in the knobs section.

IG logged in:



Here we show the top 5 features without using the abs transformation. Note that there are more than 5 features in the legend b/c the top 5 features are computed per action.

IG logged out:



In the above IG logged out example, for no response (i.e., action 0.0) , both “log_request_count_1d” and “log_user_count_on_ip” play a role, yet for the blocking decision (i.e., action 1.0) , only “log_request_count_1d” is important.

Panel B: Tracking feature attribution and value shifts

Models trained are often fragile, meaning that they tend to perform well in the training distribution but not in deployment. Ideally, we want to learn a model that is robust to distribution shift to begin with, but this is a lofty goal that often relies on knowing how the distribution is shifting. A more attainable objective is to monitor distribution shifts and take actions to correct model behavior (e.g., revert the model back to an earlier version) or input discrepancy (e.g., understanding why the feature shifts) after we notice a shift in distribution. Furthermore, the predicted metrics themselves aren’t always reliable. Take friction metrics for example, what if in the future flytrap report can be automated and no longer reflect user friction? This will result in a shift in the model decision logic that we want to know, again requiring tracking of model logic across time.

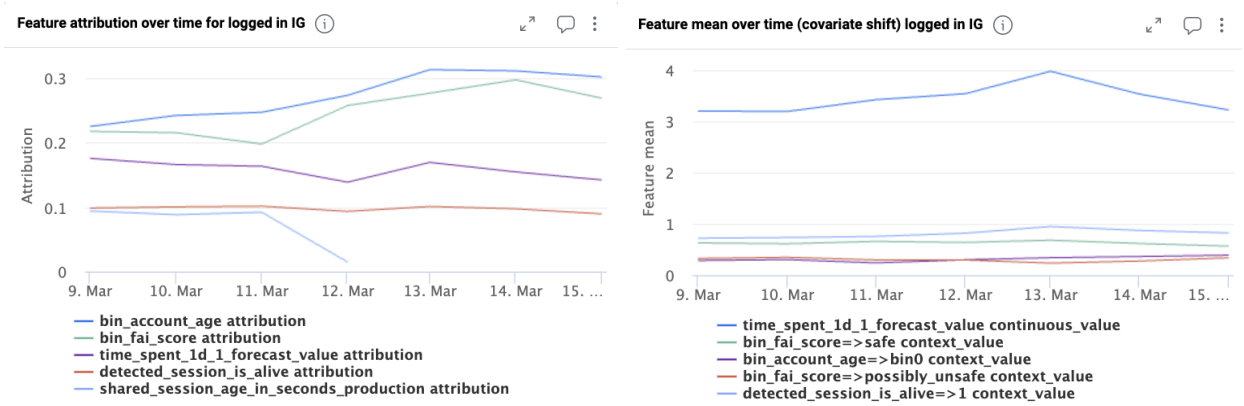
The tools built in this panel helps track distribution shift, both in the input distribution (i.e., P(X) where X is the input) and the conditional distribution (i.e., P(Y|X) where Y is the predicted metrics). **All panels here are good candidates for setting up alerts**.

Feature attribution and value shift:

- Feature attribution may shift due to covariate shift (e.g., more percentage of actioned samples leads to weighting them in aggregation more) or model logic shift (e.g., the model learned to depend on different features from different days). This is shown in the left graph below. Shifts in feature attribution can be used to explain spikes in an action (e.g., if model logic changes, even with the same input distribution, the distribution of model responses will change), facilitating finding the root cause (e.g., did the input distribution shift? Did the data used to train the model get corrupted? Did the metric definition change?).

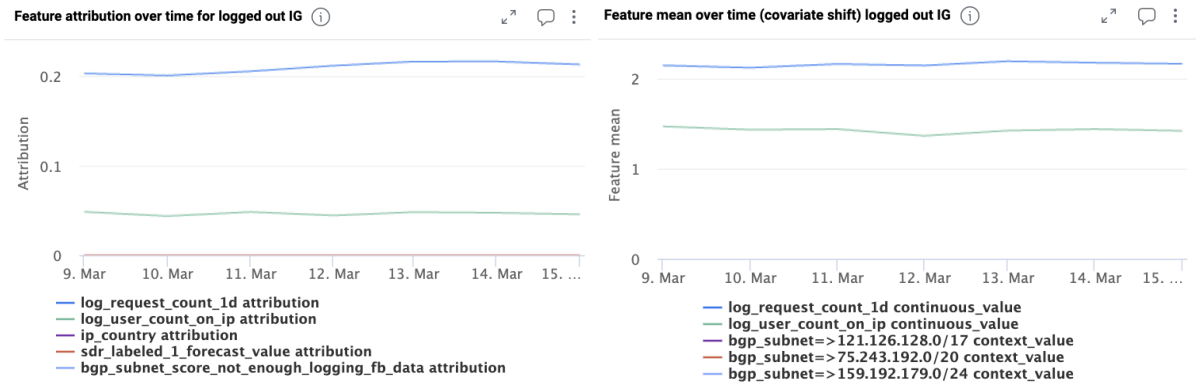
- Feature value/covariate shift for important features are tracked on the right graph. Here we plot the average value for the top N important features across time (for categorical features, this means tracking the percentage of a particular value for a feature, e.g., what proportion of “bin_fai_score” is “safe”). We focus on important features because non important features don't affect PRO’s behavior (e.g., response distribution shift, MSE degradation, and etc.).

IG logged In:



Here we see that “shared_session_age_in_seconds_production” stopped being important starting on 3/12/2023. It turns out the feature was removed from the model due to capacity issues in training (D43843906).

IG logged out:



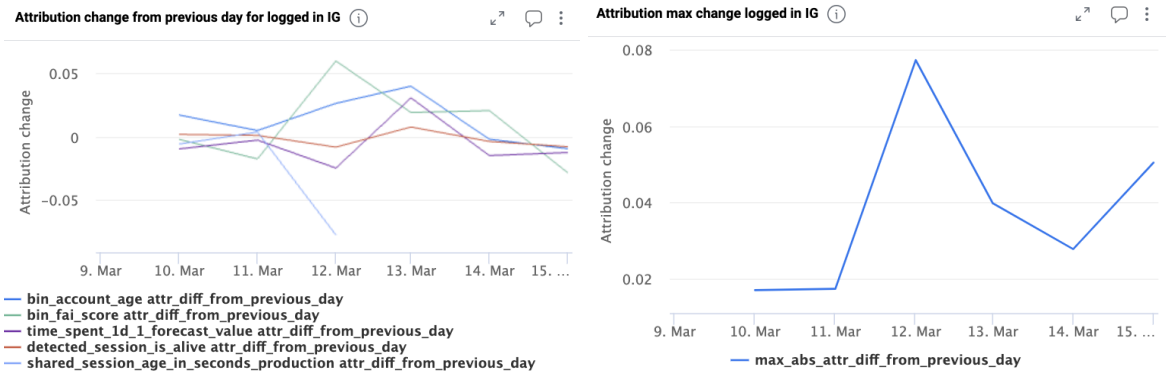
All features appear to have stable attribution and value in the time period chosen.

Caveats: Both attribution and value tracking are not without flaws. For example, they do not detect all cases in which the model or input could be shifting, resulting in false negatives (e.g., the input distribution can appear to be stable in our graph as long as its mean is stable, but other statistics can greatly vary).

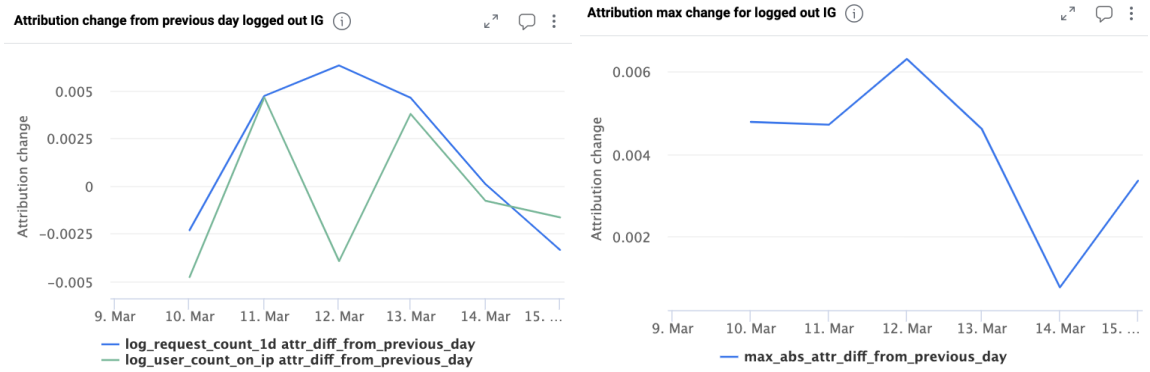
Tracking day to day change in feature attribution:

We also provide a way to track day to day changes of metric attribution. On the left, we plot feature attribution changes for top features and on the right we plot the maximum changes of the left plot. This helps quantify the attribution shift.

IG logged in



IG logged out:



Panel C: Visualizing decision logic (attribution over value)

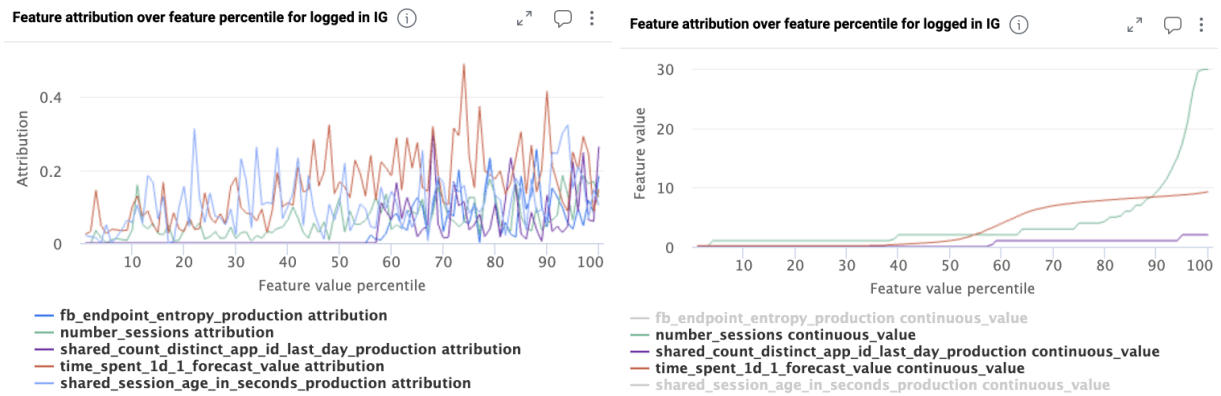
We are often not only interested in finding important features, but also want to know how feature values.affect the decision. For example, it is expected to see that PRO relies on the number of requests to decide blocking an user or not. However, it will be alarming if we see PRO blocks users with few requests but allows users with a high number of requests. In other words, we want to know how PRO is using “important” features. The tools built in this section answers those questions by aggregating feature attribution for each feature value (e.g., what is the mean attribution when the number of requests per day from a user is below 10). For continuous features, we bin the data into 100 equal sized buckets and show the aggregated result for each percentile. For categorical features, we aggregate directly based on feature values.

Decision logic on continuous features

Continuous features need to be bucketed because each sample likely has distinct feature values. Since SHAP is computed with monte carlo simulation, we need enough samples

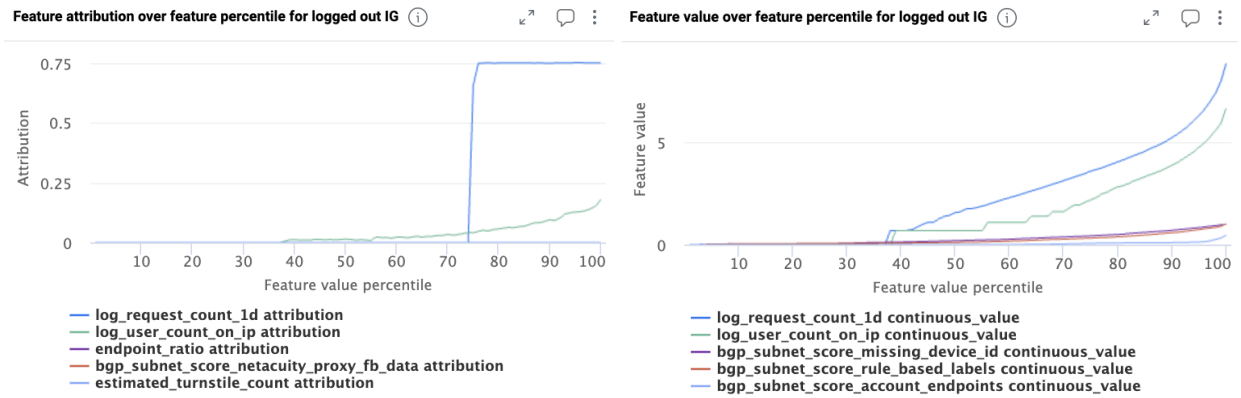
aggregated to show a feature's actual impact. Furthermore, bucking the feature values enables us to visualize all features on the same x-axis. In the left graph, we show attribution versus value percentile, and in the right graph, we show the mapping from value percentile to the minimum feature value that reaches the percentile (e.g., if the feature value for the 50th percentile ranges from 5 to 10, the y value for the 50th percentile is 5). The percentile to value mapping is useful to gauge the actual distribution of features.

IG logged in:



We see that high values of "time_spent_1d_1_forecast_value" is important for PRO's decision. Forecast_value variables are named in the convention of "<metric>_<horizon>_forecast_value". In this case, the metric is "time_spent_1d" and the prediction horizon is 1 (meaning the next day).

IG logged out:

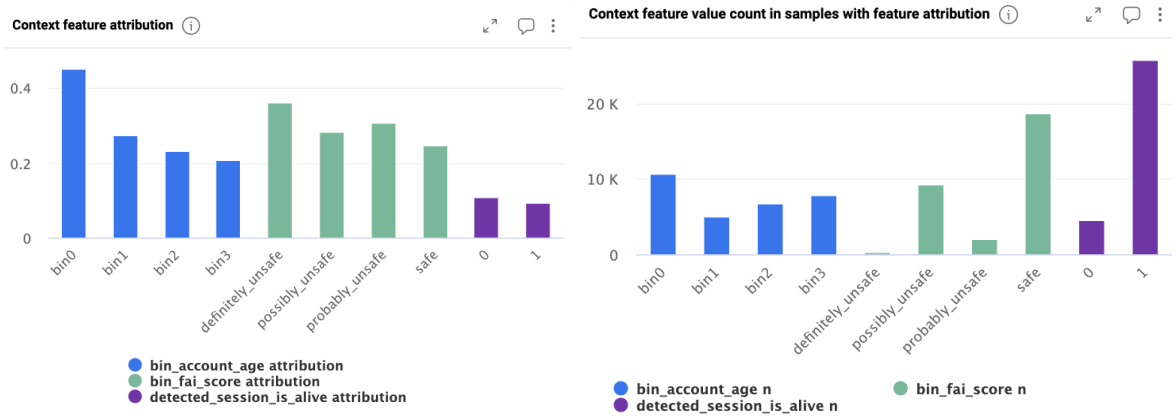


Here we see that "log_request_count_1d" is important for the decision of blocking (set action filter to 1.0). In particular, the model increased its likelihood to block when its value is above 76 percentile, which corresponds to 3.77 "log_request_count" (reading from the right plot).

Decision logic on context/categorical features

Similar to continuous features, we show both feature attribution and feature distribution.

IG logged in:



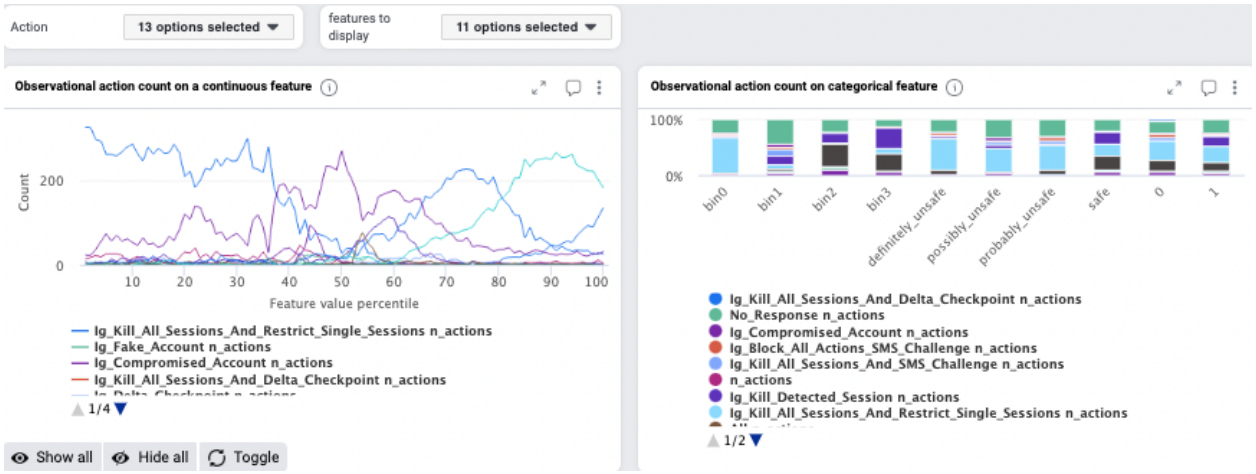
Here we see that bin_account_age of “bin_0” is most important for PRO’s decision making.

IG logged out:
Contextual features are not used in PRO IG logged out as of 3/17/2022, therefore no figures.

Observational action count on feature values:

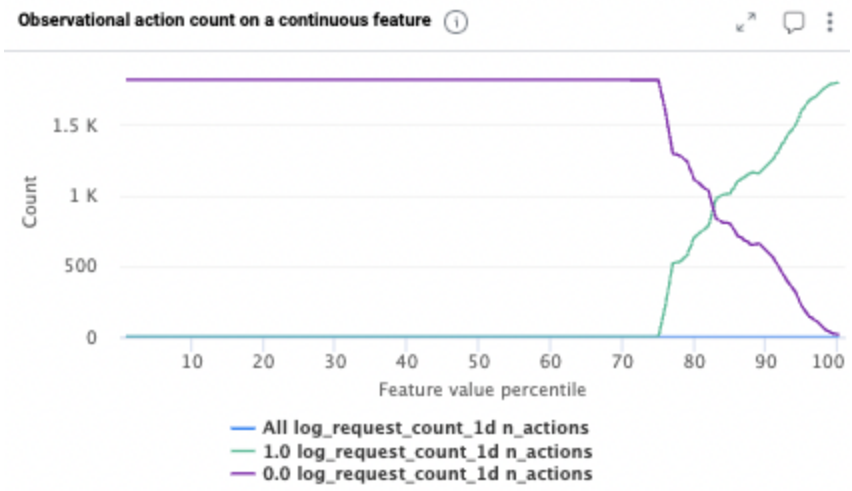
These plots are the empirical count of how many responses are given based on feature value. I recommend filtering by important features to get insight on what the model is doing.

IG logged in:



Here we filter for “time_spent_1d_1_forecast_value” for continuous features, and “bin_account_age”, “bin_fai_score”, and “detected_session_is_alive” for categorical features. We see that no response (purple line) dominates in the middle section of the curve.

IG logged out:



Here we see that when request count is less than 82 percentile, no response dominates. Ignore the “All” action as it is a placeholder with always 0 count.

Panel D: Monitoring the volume of logged feature attribution

Due to the high cost of SHAP computation (i.e., perturbing every feature for different orderings in features), we can only afford to sample a few data points for feature attribution. The plots below show the percentage of features logged out of all traffic sent to PRO (currently set around 0.1% as shown in the left figure) and the total volume sent to PRO (figure on the right).

In the past, we noticed that samples was silently dropped when there were too many calls to model inference due to SHAP (D43668629, D43402921). To overcome this, we optimized feature attribution to only attribute to features stored in the online model (D43848199). While this solves the problem for now, in the future, as the number of features increase, we may again run into the same problem (i.e., no feature attributions are logged due to high number of inference calls). The solution is to randomly sample features that are attributed to and aggregate the result (D44234422, see the next section).

Troubleshooting dropping in feature attribution:

Oncall should monitor the left panel, making sure that the percentage of logged feature attribution does not drop to zero. In case it does, reduce “shap_n_random_features” setting in the config (i.e., reduce number of features to attribute to for each sample) and increase the sampling rate to enlarge coverage[4]. Concretely, if you choose to use less random features, say you change random features from 10 to 1, you should increase the sampling rate by 10 times to make sure that each feature has in expectation the same number of attribution as before.

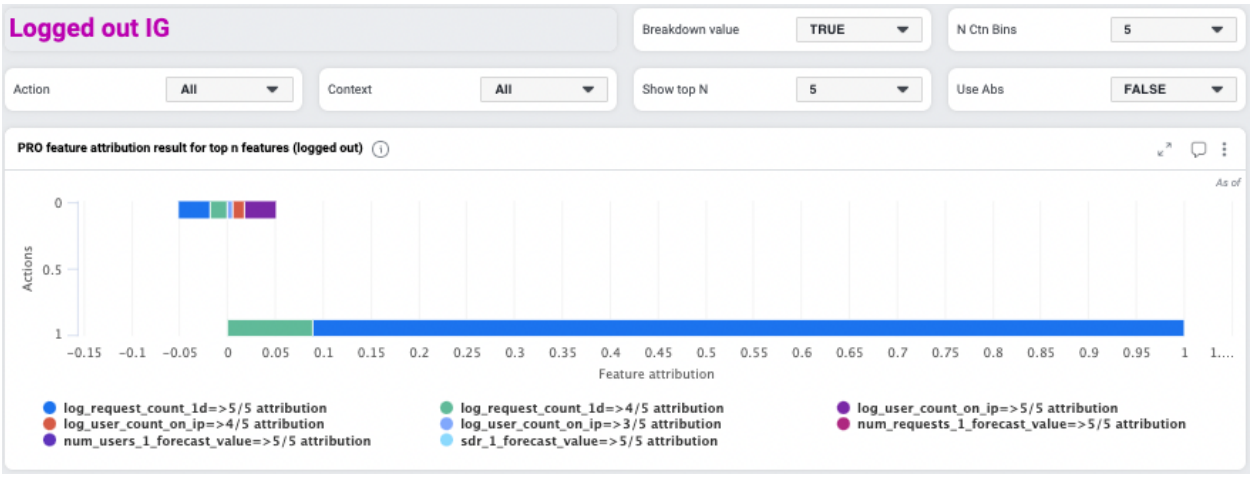
Future good to have

- Refactor Daiqueries used in the dashboard with jinja to make future change easier
- Display confidence interval and number of data points for the aggregated attribution: add a column in daiquery counting var(sum(X)/n) = var(X) / n ~= [sum(X^2) - sum(X)^2] / n^2 where n is number of observations and X is individual attribution
- Provide a general framework to allow explanation at different boundaries (e.g., allowing grouping related features, support on manifold SHAP, and etc); this could improve efficiency and generate different layers of insight
- Feature store allowing live queries on feature attribution
- Offline feature attribution (observational instead of interventional, and running the risk that we are not explaining the model the same time the decision is made) for fast explanation (Example notebook)
- Explain each action regardless if the action is chosen (currently we are explaining the decision, but could be useful to ask what increases the chance of UFAC regardless if UFAC is chosen, this increases computation load to |action|-1 times)
- Allow customized setting of background feature values
- Allow explaining random subset of features in a way that equals in expectation with exact SHAP computation (i.e., accounting for feature ordering), current implementation amounts to always ordering the chosen features at the beginning

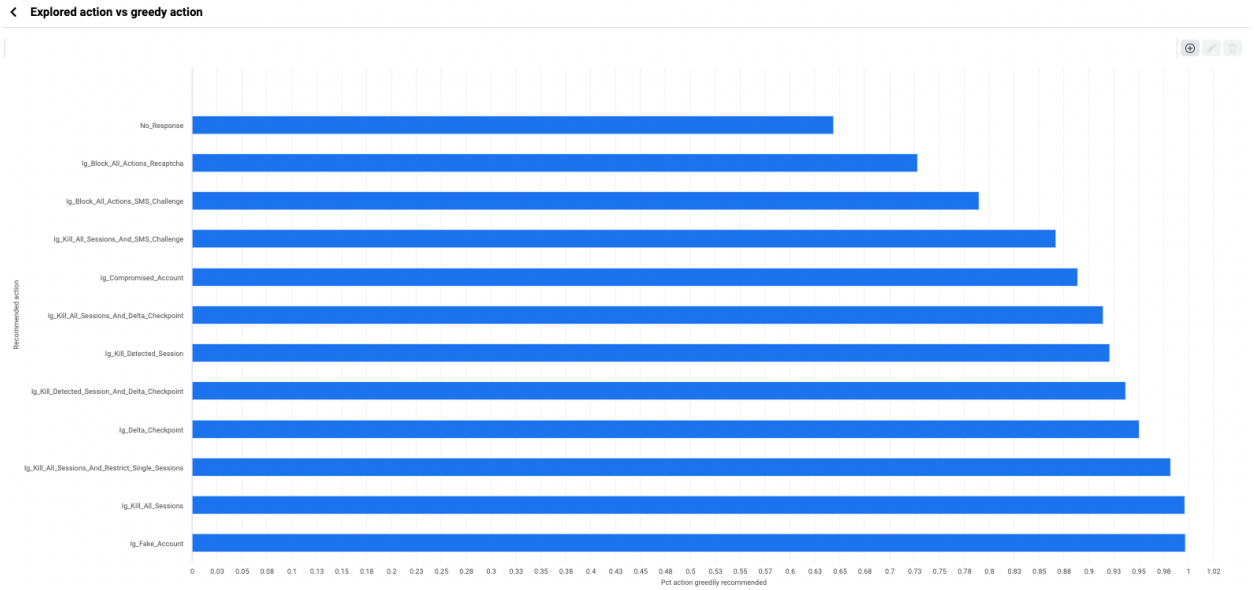
Q&A

Q: For each sample, will the attribution to all features sum to 1 setting use_abs to false?
A: No, it depends on the background value. The only guarantee in our case is the sum should be between 0 and 1. Denote x as the sample's foreground/current value and x’ as a sample with each feature set to the background value. SHAP guarantees that the sum of attribution for all features is f(x)-f(x’) where f is the function to explain. In our case, we are explaining the decision, therefore f(x) is 1 because it equals the decision made, but f(x’) may not be 0. For example if both x and x’ results in no response, then the sum of attribution will be 0. To concretely illustrate this point, we can look at the logged out attribution below as it does not have exploration as of 3/22/2023. For the no response action, the sum is 0 b/c the background response is no response. For the block action, attribution sum to 1 as expected.

[4] To ensure that the sampling will yield the same attribution in expectation with the exact SHAP result, one needs to randomly sample an ordering of features and update feature attribution according to that order; However, as a quick fix, we can ignore this technicality

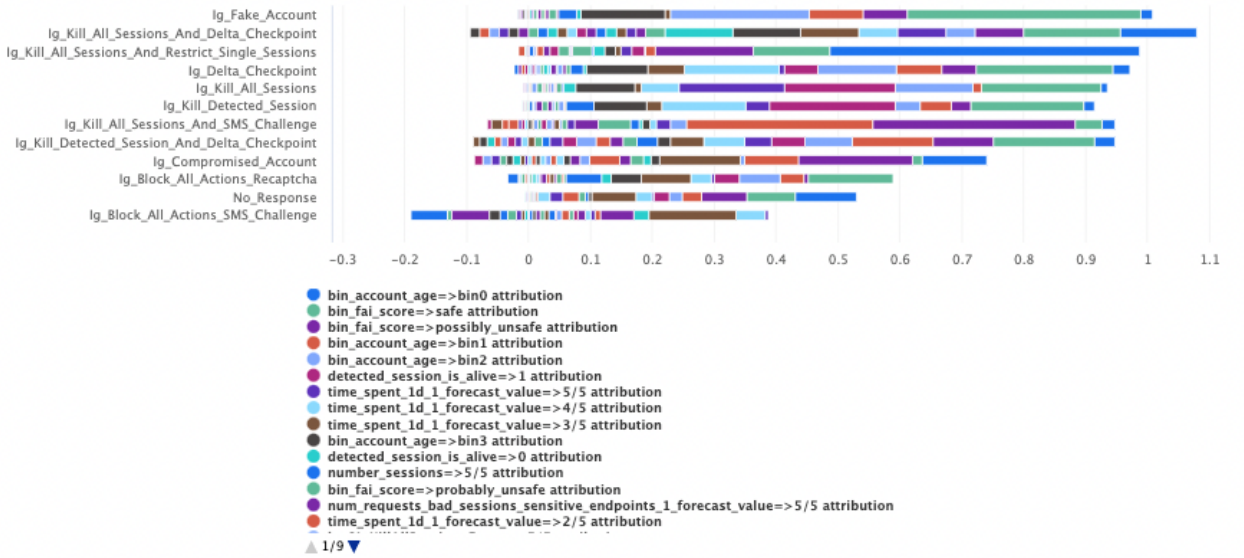


Exploration also affects the sum as well. In our case f is a non-deterministic function due to Thompson sampling, therefore f(x') is a random variable that is either 0 or 1. We therefore expect actions that are more random to deviate more from the sum of 1. Below, we see that no response, Ig_Block_All_Action_Recaptcha, Ig_Block_All_Actions_SMS_Challenge and Ig_Compromised_Account have the most explorations (first graph below), their feature attribution responses are indeed the furthest away from 1 (second graph below). Note that we want to keep exploration as part of the system to explain because certain responses are the results of exploration.



PRO feature attribution result for top n features (logged in) ⓘ

As of



- bin_account_age=>bin0 attribution
- bin_fai_score=>safe attribution
- bin_fai_score=>possibly_unsafe attribution
- bin_account_age=>bin1 attribution
- bin_account_age=>bin2 attribution
- detected_session_is_alive=>1 attribution
- time_spent_1d_1_forecast_value=>5/5 attribution
- time_spent_1d_1_forecast_value=>4/5 attribution
- time_spent_1d_1_forecast_value=>3/5 attribution
- bin_account_age=>bin3 attribution
- detected_session_is_alive=>0 attribution
- number_sessions=>5/5 attribution
- bin_fai_score=>probably_unsafe attribution
- num_requests_bad_sessions_sensitive_endpoints_1_forecast_value=>5/5 attribution
- time_spent_1d_1_forecast_value=>2/5 attribution

1/9