Accelerate Negotiation Guidance

Version 2.0.1

December 2023

Accelerate Negotiation Guidance

The Negotiation Guidance Accelerator provides a fast implementation of segmentation and computation of price elasticities by segments.

In this section:

- Overview (Optimization - Negotiation Guidance)
- Business User Reference (Optimization - Negotiation Guidance)
- Admin User Reference (Optimization - Negotiation Guidance)
- Technical User Reference (Optimization - Negotiation Guidance)
- Release Notes (Optimization - Negotiation Guidance)
- Archive of Documentation (Optimization - Negotiation Guidance)

To query a negotiation guidance result from outside of the Optimization module, please refer to Query Optimization Engine Results.

Overview (Optimization - Negotiation Guidance)

The idea behind the Negotiation Guidance model is very intuitive. When facing a new pricing event, you need to make a pricing decision. A sensible way to make this decision is to compare the situation with a similar one from the past. If you have sold the same product to the same customer before, you already have a good indication to start with. Or maybe you have never sold that product to the same customer before so you have nothing to compare to. In these cases, you would also want to check if you sold a similar product to that customer, or to a similar customer and consider that pricing too. That is where segmentation takes place because it provides a richer comparison set than if we were just looking for the exact same situation in the past. It allows us to explore more possibilities, align with your pricing strategy and avoid sticking with the same (possibly not ideal) pricing that was used in the past.

Pricefx Key Accelerators

To get started, you can also watch a video introducing Negotiation Guidance and its benefits.

Pricefx Solution

In our Negotiation Guidance model, first we provide some analysis and **guidelines** on the key attributes that drive the prices which we call **Price Drivers**. That way you can better understand if geography, product types or customer segments have a impact on the prices paid. Moreover, knowing the Price Drivers enables us to recommend segmentation levels. Of course, you can always select the segmentation level you wish to use.

Then, a **segmentation tree** is built and provides **price recommendations** and elasticity based on the information from each segment. The segmentation levels are usually a combination of product attributes, customer attributes, and even any selected transaction attributes.

The fact that we build a segmentation ***tree*** instead of considering all the combinations of attributes is crucial. We only go into such depth where there are enough transactions to make an informed decision and also we do not have to consider lots of irrelevant attribute combinations. Ideally, the segments should be consistent in the sense that most of the transactions within a segment should have similar pricing. Indicators are provided, so that you can make further analysis to understand consistency of segments.

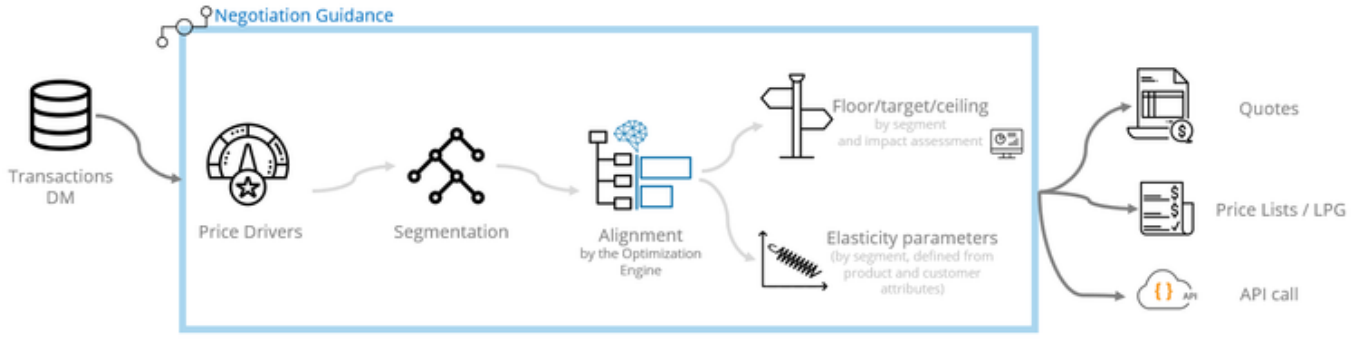
The goal of our price optimization is then to assess the pricing potential of each segment and adjust subsequently. On top of that, **alignments** can be enforced in order to get consistency across segments based on your pricing strategy.

> See also a video explaining how to optimize deals with Negotiation Guidance.

Approach

Negotiation Guidance relies on several steps:

- Data analysis resulting in **Price Drivers**.
- Building a **segmentation tree**.
- Providing recommendations of floor, target, and ceiling as margin % or discount % depending on the selected optimization target.
- Adjusting those recommendations by **aligning** segments based on pricing strategy. These recommendations are then ready to be used in other parts of the solution.
- Each segment also comes with metrics, including elasticity parameters that can be leveraged for some other usage.



Outputs

The outputs of Negotiation Guidance consist of recommendations of floor, target, and ceiling for either the discount rate or the margin rate. Meaning for each segment, defined by a set of attributes and built

within the segmentation tree, Negotiation Guidance recommends guardrails: floor, target, and ceiling for discount rate or margin rate (depending on chosen optimization target). Those values are intended to be used later in the process to compute prices from either list prices for discount or costs for margin rate. That process comes with the benefit of aggregating the recommendations at a segment level and using the recommendations dynamically following potential changes in list prices or costs.

Each segment also comes with elasticity parameters that are computed with segment scope and can also be leveraged for some other usage by other modules or models.

Limitations

- **Attributes/features** – Negotiation Guidance relies on having the right features that impact the price /margin rate or discount rate. If those features are not available or the data are incomplete, output quality will be low.
- **Optimization target** – For now, two optimization targets are supported: discount rate (relying on a list price and differentiating prices through discounts) and margin rate (for a cost-plus strategy). So if you wish to use another metric (win rate, unit price directly...), this needs additional effort.
- **Business alignments** are computed based on price gaps that rely on costs (for margin%) or list prices (for discount%), so if those are not consistent, those inconsistencies will be propagated. So it is better to use discount% as an optimization target, as list prices tend to be more consistent and less volatile over time.
  For now, business alignments cannot be used if a "weight" is defined.
- **Model outputs and recommendations** – Negotiation Guidance is intended to provide recommendations by segment and not at a product and customer level (but most probably at a higher level of granularity). The recommendations will not directly include recommended prices for a specific product and customer. In case a maximum price change is required, this should be enforced later in the process, for example in the quote logic.
- **No predefined extension point** – There is no out-of-the-box extension point defined for now. If you intend to add specific features, custom code should be written. (But then the accelerator becomes specific and it cannot be updated without extra effort to port those modifications.) Do not hesitate to report specific requirements and possible extension points to Pricefx.
- **Data requirements** – See Data Requirements (Optimization - Negotiation Guidance).

Business User Reference (Optimization - Negotiation Guidance)

Data Requirements (Optimization - Negotiation Guidance)

A Negotiation Guidance model needs a Transactions Datamart to run. Here are the prerequisites for the fields.

Field	Required?	Comment
Customer Field	Yes	The customer ID.
Product Field	Yes	The product ID.

Quantity Measure	Yes	It is better to avoid negative or null values (they are filtered out by default by the model itself).
Revenue Measure	Yes	Extended to the quantity. It is better to avoid negative or null values (they are filtered out by default by the model itself).
Margin Measure	Yes	Extended to the quantity. It is better to avoid negative or too large values (they are filtered out by default by the model itself).
Optimization Target	Yes	The margin percentage or the discount percentage, with a value typically set between 0 and 1 (1 meaning 100%).
Weight Measure	No	The numerical value to weight the transactions, typically the quantity field.
List Price	If the Target is of type Discount%.	Extended to the quantity.
Segmentation Dimensions	At least one	Must be dimensions. It is necessary to avoid null values (they are replaced by default by a default string value).

Additionally, for the best user experience, it is recommended to type fields that are of Money type (such as Revenue, Cost, etc.) as Money and not as Numeric. (If such field could be selected in the Analysis and Configuration steps, it could negatively influence the computation of the price drivers importance and their recommendations.)

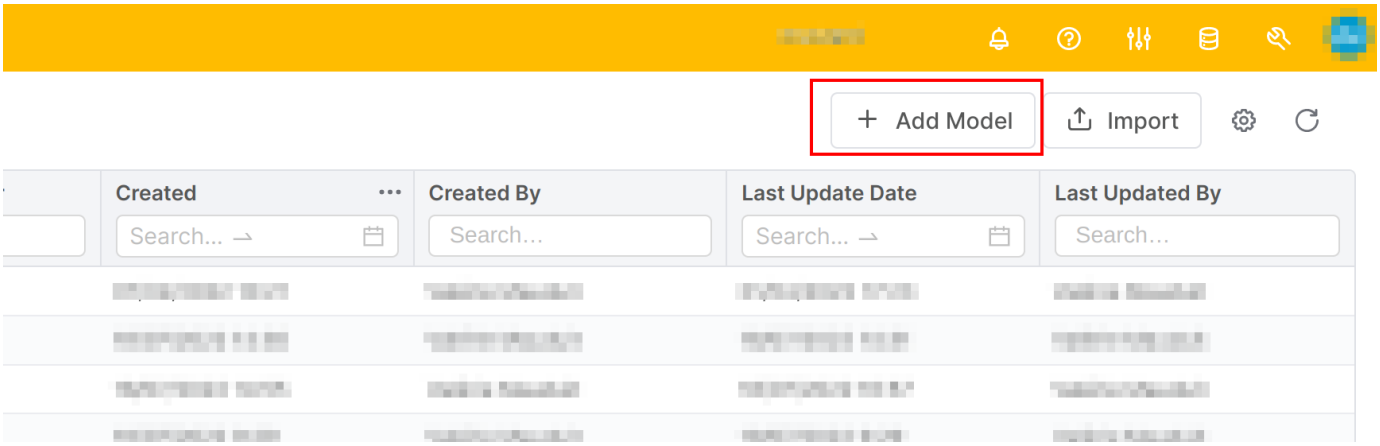
Usage (Optimization - Negotiation Guidance)

Take the following steps to configure a model:

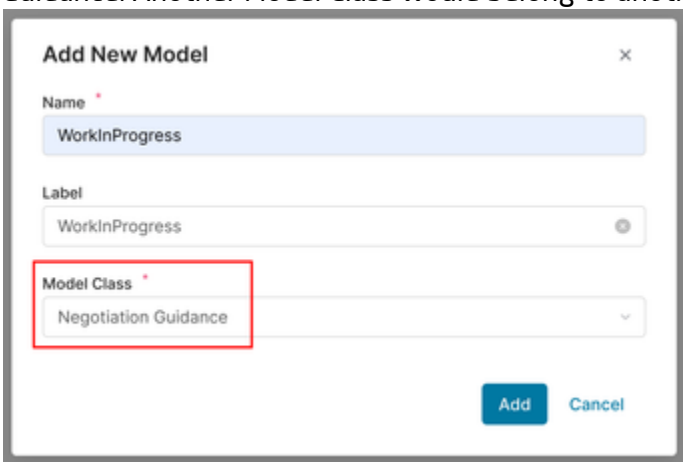
- 1. Create a Model Based on Negotiation Guidance Model Class
- 2. Set the Scope of Transactions (Definition Step)
- 3. Analyze Data in the Scope and Set Price Drivers Parameters (Analysis Step)
- 4. Configure the Segmentation (Configuration Step)
- 5. Set up Optimization Parameters (Segmentation Step)
- 6. Manage the Segmentation Results (Result step)

1. Create a Model Based on Negotiation Guidance Model Class

Go to **Optimization** > **Models** and click the **Add Model** button at the top right.

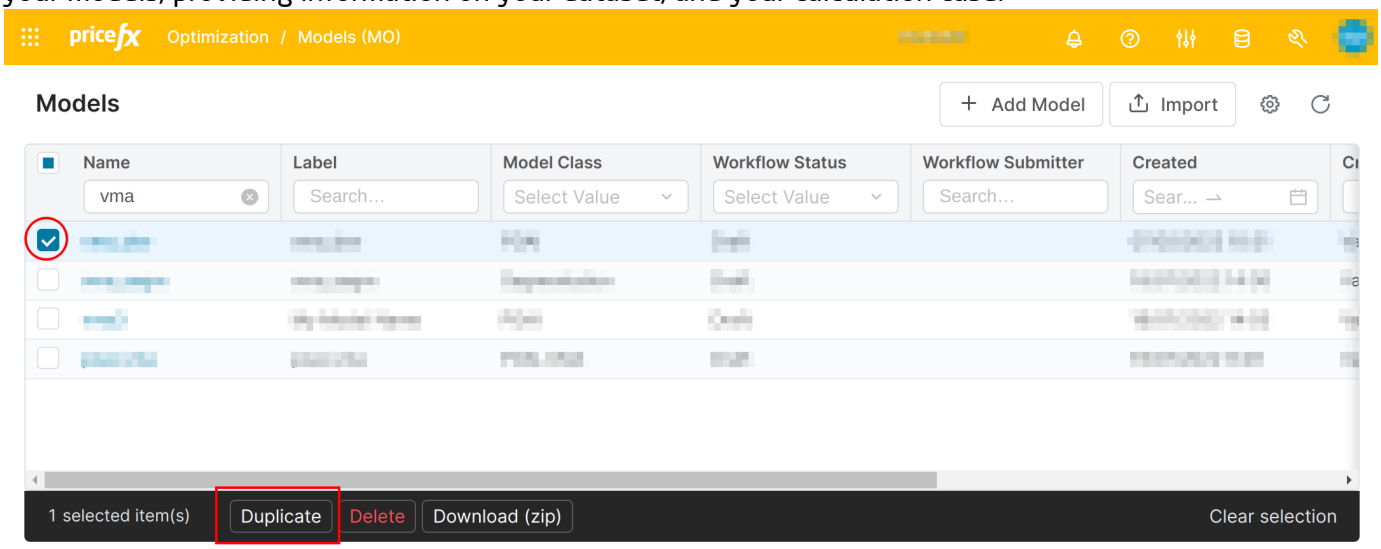


A pop-up is shown where you provide a name for your model, and the Model Class, which is *Negotiation Guidance*. Another Model Class would belong to another kind of optimization model.

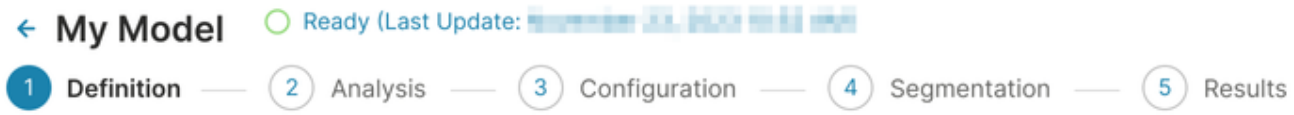


You can also duplicate an existing model. In this case, you will keep all the inputs of the previous model and will have to rerun all the steps to get the outputs. Once you have copied a model, you can change its name by double-clicking the blank side of its name/label.
⚠️ Remember to do it before running the model. You cannot change a model name once it has been computed.

The same model class, *Negotiation Guidance*, can be used by many models. Use informative names for your models, providing information on your dataset, and your calculation case.



When you create a model, it is automatically opened in the browser. Any negotiation model has five steps: Definition, Analysis, Configuration, Segmentation, and Results. Each step has one or more tabs; some tabs are dashboards, and other ones have user entries to interact with them or to set the parameters of the next calculations. A step is available from the previous one after clicking **Continue**, for the first time. When clicking **Continue**, the state of the model is saved, and the calculation that starts the next step begins if it has not run yet with this model state. Otherwise, you simply focus on the next step without any action.



2. Set the Scope of Transactions (Definition Step)

The step is detailed in Definition Step (Optimization - Negotiation Guidance).

You can then click the **Continue** button (top right) which takes you to the Analysis step.

3. Analyze Data in the Scope and Set Price Drivers Parameters (Analysis Step)

The step is detailed in Analysis Step (Optimization - Negotiation Guidance).

Once the price drivers choice is made, click **Continue**. The next step, Configuration, will first run the calculation of price drivers and then provide the next section.

4. Configure the Segmentation (Configuration Step)

The step is detailed in Configuration Step (Optimization - Negotiation Guidance).

Once the segmentation parameters are set, click the **Continue** button (top right) to go to the Segmentation step.

5. Set up Optimization Parameters (Segmentation Step)

The step is detailed in Segmentation Step (Optimization - Negotiation Guidance).

Once the optimization parameters are set, click the **Continue** button to run the optimization itself and go to the Results step.

6. Manage the Segmentation Results (Result step)

The step is detailed in Results Step (Optimization - Negotiation Guidance). This page also explains how to evaluate the model from any module in the partition.

Definition Step (Optimization - Negotiation Guidance)

The aim of this step is to set the scope of the transactions. There is a single tab, composed of a configurator (left frame) and a dashboard (right frame).

Inputs

In the Definition step, you map the inputs and set the scope of the model. The user inputs are always on the left. By default the inputs are filled with values set during the deployment of the accelerator.

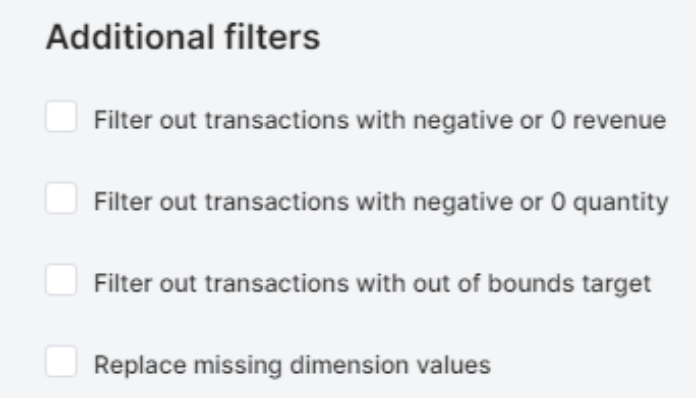
- **Transaction source** - Datamart or Data Source used to calculate the segments. It must fill the requirements listed in Installation (Optimization - Negotiation Guidance). Once provided, some fields based on it appear:

- **Transaction Filter** – Allows you to filter the data. We recommend that you define at least these filters:
  - Positive cost, revenue, and quantity.
  - Optimization target in a realistic range, for example between -0.1 and 1.
- **Customer Field** – Customer ID field. It is used to aggregate data by customer.
- **Product Field** – Product ID/SKU field. It is used to aggregate data by product.
- **Quantity Measure** – Field that indicates the quantity in a transaction.
- **Revenue Measure** – Field that provides the transaction extended price which will be the basis of the analysis. It may be a net price, a gross price, or another, depending on the policy you want to simulate.
- **Margin Measure** – Field providing the extended margin of the transaction. Similarly to the Revenue Measure, it is chosen according to the policy you want to simulate.
- **Optimization Target** – Usually the margin rate or the discount *rate* (i.e. a value of 10% must be represented with the value 0.1). No null values are allowed in the optimization target field.
  If you intend to use alignments later in the process, discount rate is a better choice as alignment will then take the list prices as a reference, which is generally more consistent and less volatile than the costs.
  ⚠️ Your selection here (margin or discount rate) must be accompanied by a corresponding value in the Target Type option (below).
- **Target Type** – Select either *Margin%* or *Discount%* depending on what you have selected in the Optimization Target option above.
- **Weight Measure** (optional) – Field that indicates the weight of a transaction row. If not set, all the Datamart rows are weighted equally. You may use the quantity field, for instance, if you want to weigh ten times more a transaction to sell ten products than one to sell only one product. For now, business alignments cannot be used if a "weight" is used.
- **List Price** – Mandatory field if the metric is of type *Discount %*. It represents the price without any discount applied to it. It must be an extended value.

The revenue, margin, and list price values are used to simulate the transactions when the optimization target value changes. The mapped fields should be consistent: Optimization Target, Revenue Measure, and Margin Measure are checked together in case the Target Type is set to Margin%; Optimization Target, Revenue Measure, and List Price are checked together in case the Target Type is set to Discount%.

If needed, create some calculated fields in the transaction source: revenue, margin, margin rate (i.e., margin /revenue).

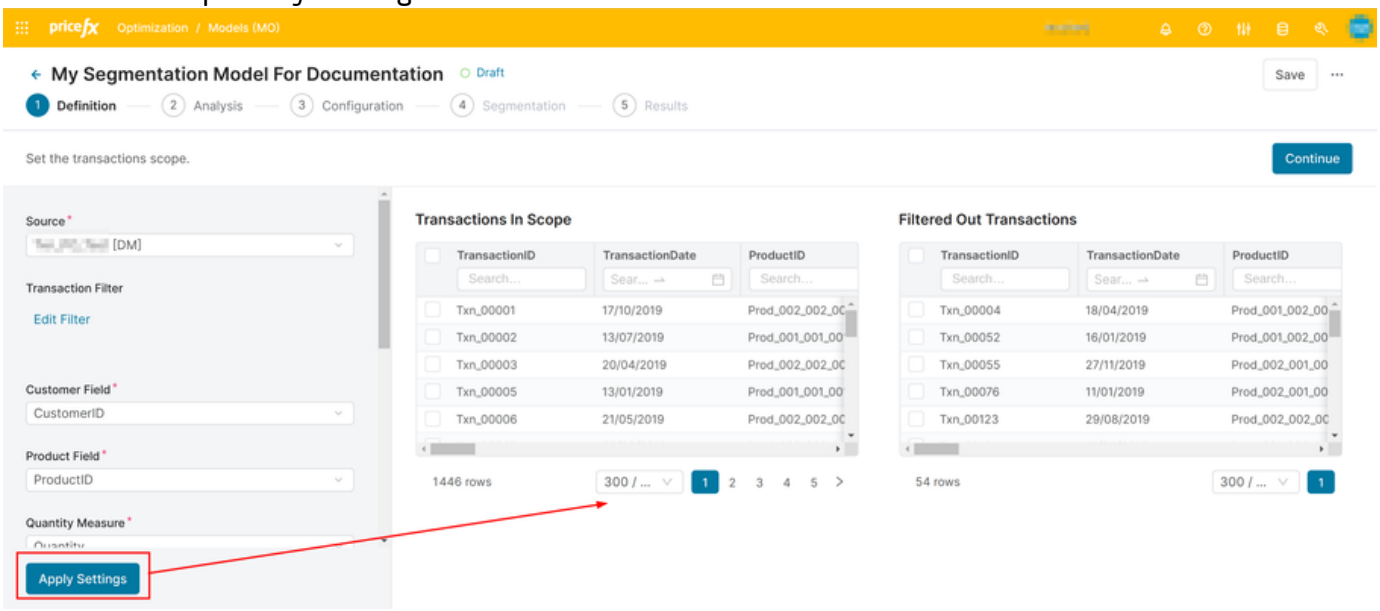
Use the check-boxes for additional filters ensuring calculations without exceptions. These parameters are checked by default. Note that the "Replace missing dimensions" replaces the null values of each dimension with a given string (by default, it is `__missing__`). If you uncheck this checkbox, the dimensions with null values will not be selectable as segmentation levels.



Dashboard

Once you apply the settings, the right panel provides:

- **Transactions in Scope** - Data that are in the scope of the segmentation.
- **Filtered Out Transactions** - Data that are filtered out by the set transactions filter.
  ⚠ Some data rows appear neither in the *Transactions in Scope*, nor in the *Filtered Out Transactions*. It is the case if a value is filtered out by the advanced filter, but its value is null and it is also filtered out with the opposite of the user filter. Please consider the Filtered Out Transactions portlet as source of information possibly missing data.



You can then click the **Continue** button (top right) which takes you to the Analysis step.

Analysis Step (Optimization - Negotiation Guidance)

The aim of this step is to analyze the data in the scope and set the price drivers parameters. It starts with a calculation whose aim is to materialize used data inside the model. Once the calculation is done, there are two tabs: Data Profile, and Price Drivers Setup.

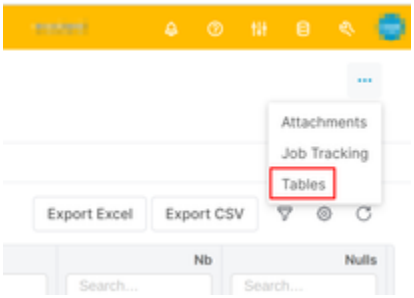
- Data Preparation
- Data Profile
  - Scope Summary
  - Details for all Fields
  - Distinct Values
- Price Drivers Setup

Data Preparation

When you arrive at the Analysis step from the Definition step, the model first runs a calculation to prepare the data. It takes up to some minutes, depending on the size of the transaction source. The goal of this calculation is to format and save the data that will be used in the next steps of the model.

This step also checks the mapping of the fields set by the user. The mapped fields should be consistent. The average optimization target value must be close to the value calculated using the revenue and the margin measure (in the case of a margin % type) or the revenue and the list price (in the case of a discount % type).

Two tables are created: *Transactions* and *Profile*. The tables of a model are always accessible through the menu in the top right corner. But usually, you would not need to access them like this; all needed information is directly provided in the sections of the model.



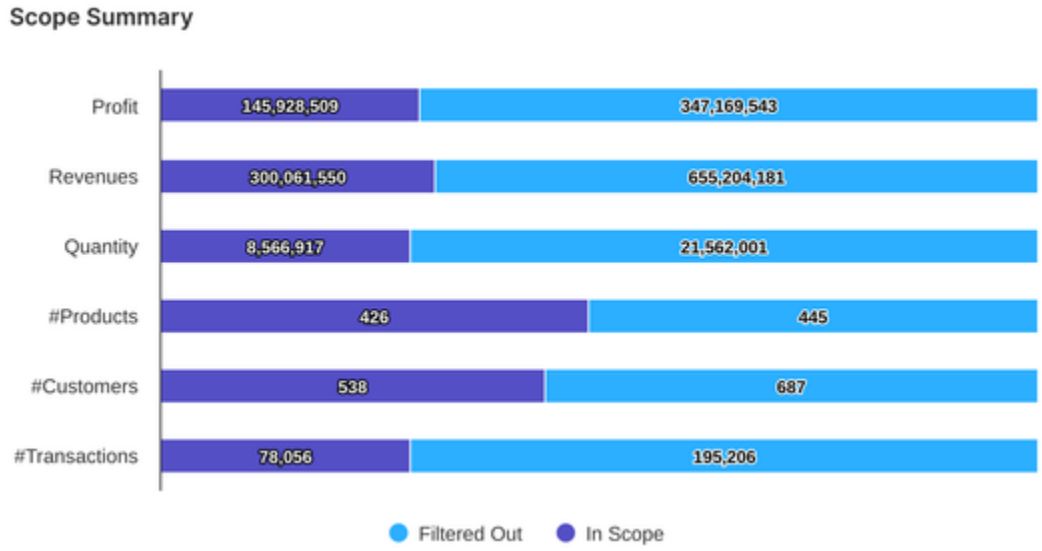
Once the calculation has run, two tabs appear: Data Profile and Price Drivers Setup.

Data Profile

This tab is mainly a dashboard made of three portlets:

Scope Summary

This bar chart shows how much data is in the scope of the segmentation and how much is filtered, among different dimensions, from the total profit to the number of transactions.



Details for all Fields

This table is a summary of all the mapped fields and the dimensions of the transaction source, taking into account the filtered data in the scope. For any data, you get the min and max values, the number of nulls, the number of different values (cardinality), and information about the type of the field and its owner.

It is useful to check if there are some nulls in fields that you would want to use as segmentation levels, or if some cardinalities are too high for a segmentation.

Distinct Values

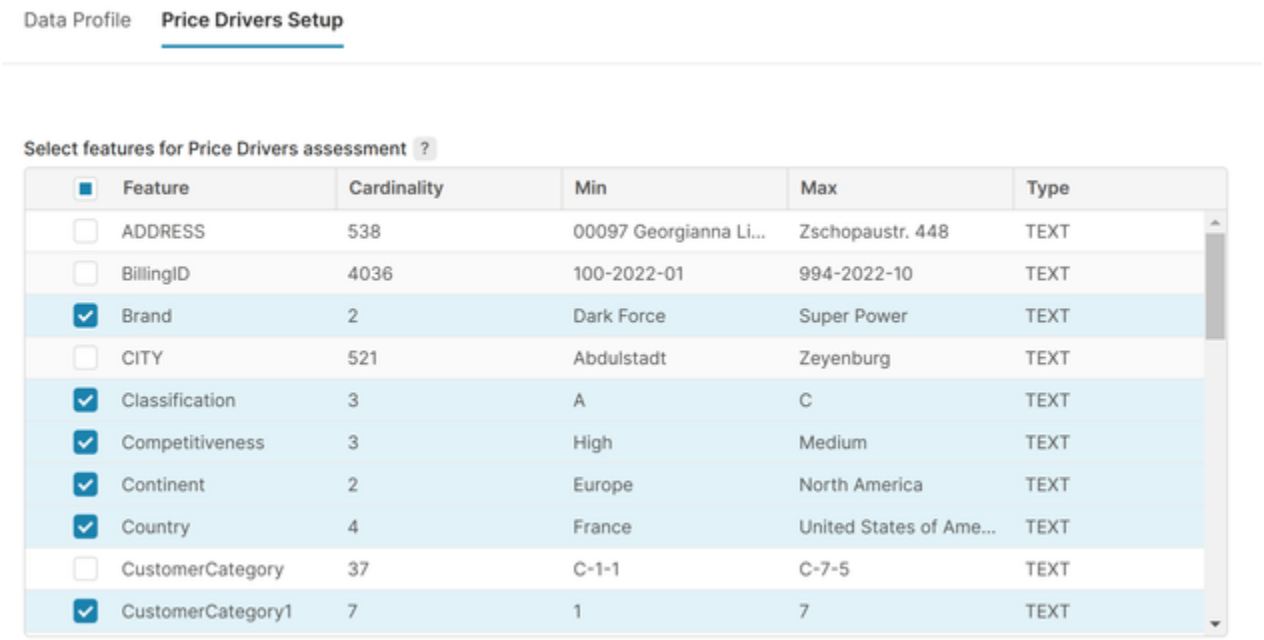
This portlet is optional. It is created if you enter a value in the left user input **Show distinct values for**, and apply the settings. It allows you to deeper check any dimension of the data scope, to validate if you are interested in using it for the segmentation.

Price Drivers Setup

This is the place where you choose the parameters for the next calculation. In this tab, you choose the fields on which the price drivers are calculated.

⚠ *Only the fields that are selected in this price drivers selection are candidates to be the segmentation levels.* Please select all the fields that you want to use in your segmentation tree.

The price drivers evaluate the importance of any feature of the data to forecast the optimization target (i. e. generally the margin percentage), and the interactions among all features, detect hierarchies, and provide dimension recommendations for the segmentation. These values and recommendations will then help you define the dimensions you will take as levels of segmentation and their order. It will only provide help: you can use any dimension in the segmentation even if the related price driver has not been calculated. The more features you choose, and the more cardinality they have, the longer the calculation will take. By default, all the dimensions of cardinality between 2 and 30 are pre-selected (excluding non-dimensional features).



Once the choice is made, click **Continue**. The next step, Configuration, will first run the calculation of price drivers and then provide the next section.

Configuration Step (Optimization - Negotiation Guidance)

The aim of this step is to configure the segmentation. It starts with the calculation of the price drivers. When the calculation is done, the step displays a tab composed of a left panel to set the segmentation parameters, and a right panel to analyze the price drivers.

Price Drivers Calculation

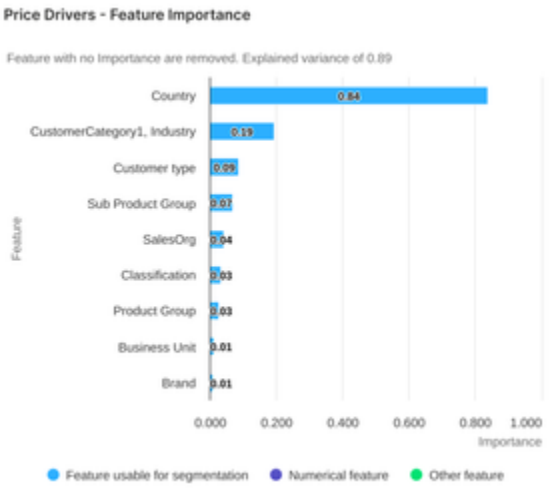
When you arrive at the Configuration step from the Analysis step, the model first runs a calculation to evaluate the relationships between the price drivers and the optimization targets. This calculation is based on a triggered Python job: first a preprocessing work is done, then a Python job is triggered. During the calculation, you will see that two different jobs are run.

The calculation creates five tables, each of them prefixed with “PriceDrivers”. They are used as inputs to the price drivers dashboard.

Price Drivers Dashboard

There are six portlets in the dashboard. *The input values of the left panel are not related to the dashboard itself.* Their goal is to define the calculation for the next step.

Price Drivers - Feature importance



This portlet shows the importance of the selected price drivers, in a decreasing importance order. The importance is a measure of how good the feature is at predicting the optimization target (defined in the Definition step). Simply put, the higher the importance, the more the feature is accurate to provide a good segmentation. The importance is based on feature permutation, and enhanced for the purpose of segmentation:

- by containing its natural randomness (the feature importance will not change if you recalculate with the same features selected),
- by removing natural noise (features that have no real importance are put at 0), and
- by adjusting the values for the purpose of the segmentation (features with lower cardinality are preferred for segmentation).

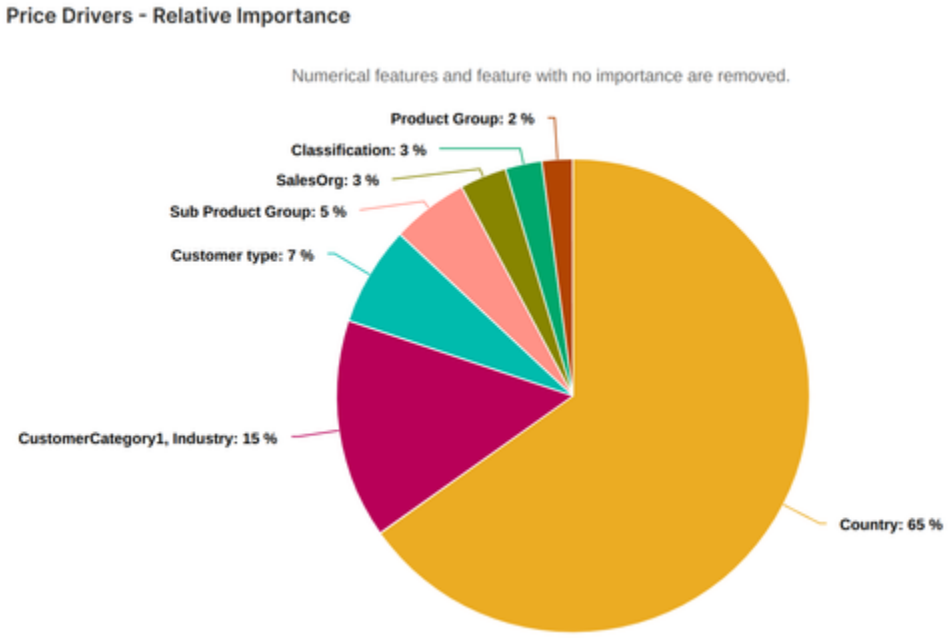
The importance measure is highly dependent on the features selected in the Analysis step. For example, if you selected only two features, you might have high importance for the two features, but the features might not be good candidates for the segmentation. A measure of the goodness of the importance measure called Explained Variance, is available in Price Drivers - Feature Importance portlet (in the figure Price Drivers - Feature Importance portlet, Explained Variance is 0.89). Given it is possible to select Numerical features in the Analysis step, but they cannot be used for the segmentation, they will be displayed differently in the Price Drivers - Feature Importance portlet, and will not be displayed in the

Price Drivers - Relative Importance portlet. The “Numerical feature” and “Other feature” bars are not usable for the segmentation.

Numerical features cannot be directly used as the segmentation is based on categories, so a solution could be to structure a numerical feature into bins in the Data Source and then this new feature can be used. “Other feature” corresponds most of the time to features that are not set as a “dimension” in the Datamart or Data Source, which is required for the next step (for performance reasons).

The subtitle indicates the global explained variance for all the price drivers combined.

Price Drivers - Relative importance



This pie chart displays the sharing of importance between the different price drivers usable for the segmentation.

Segmentation Dimensions Recommendation

Segmentation Dimensions Recommendation

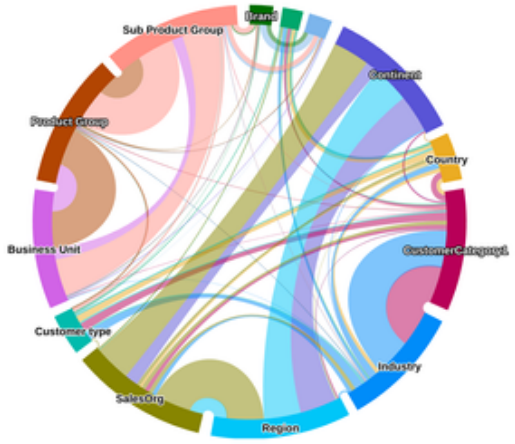
Rank	Feature	Selected	Importance	Percent Importance (%)	Recommendation Reasons	Duplicate With	Highly Correlated With	Cardinality
1	Country	true	0.84	63.77				4
2	CustomerCategory1	true	0.19	14.65		Industry		7
2	Industry	false	0	0	Duplicate with (CustomerCategory1).	CustomerCategory1		7
3	Customer type	true	0.09	6.66				3
4	Business Unit	true	0.01	0.81	Coming from Sub Product Group hierarchy.		Sub Product Group, Product Group	2
5	Product Group	true	0.03	2.06	Coming from Sub Product Group hierarchy.		Sub Product Group	4
6	Sub Product Group	true	0.07	5.13				16
7	Continent	true	0	0.14	Coming from SalesOrg hierarchy.	Region	SalesOrg	2
7	Region	false	0	0	Duplicate with (Continent).	Continent		2
8	SalesOrg	true	0.04	3.13				9
9	Classification	true	0.03	2.45				3
10	Brand	true	0.01	0.71				2
11	Competitiveness	true	0.01	0.5				3

Leveraging feature importance, feature interaction, and hierarchies, the *Segmentation Dimensions Recommendation* portlet recommends an ordering and a selection of the dimensions. The ordering is created leveraging the importance of the features, and the hierarchy. The features are first ordered from

higher importance to lower importance. Then, if a feature (called C) is in a hierarchy and there are features higher in the hierarchy (called A and B), those features are put before the feature (so the ordering would be A, then B, then C) if not already. This calculation also detects duplicated features (shown in the column "Duplicate with") and gives them the same rank.

Feature Interactions

Feature Interactions



Feature Interaction Data

Feature 1	Feature 2	Interaction
Brand	Classification	0
Brand	Competitiveness	0.12
Brand	Continent	0
Brand	Country	0
Brand	CustomerCategory1	0
Brand	Industry	0
Brand	Region	0
Brand	SalesOrg	0
Brand	Customer type	0.01
Brand	Business Unit	0.02
Brand	Product Group	0.01
Brand	Sub Product Group	0.04
Classification	Brand	0
Classification	Competitiveness	0

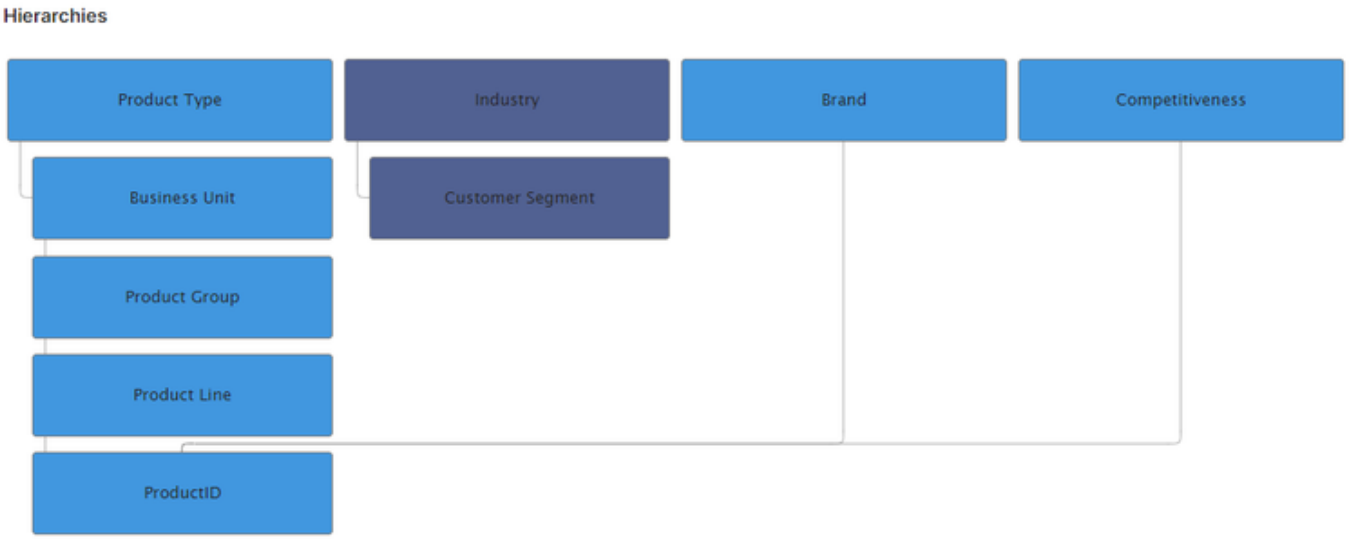
The Feature Interactions section informs the users how much two features are similar. It represents how much you know of feature 2 if you know feature 1. The interaction value is between 0 and 1, where if the interaction between feature 1 and feature 2 is 1, knowing feature 1 you entirely know feature 2. So the higher the value, the more information you know from this feature. Keep in mind this is not symmetric: knowing the customer city gives you the customer country, but not the other way around.

The raw data are displayed in the table **Feature Interaction Data** and the feature interactions are computed from:

- for categorical-categorical feature interactions: Theil's U
- for numerical-numerical feature interactions: Pearson's R
- for categorical-numerical feature interactions: Correlation Ratio

Hierarchies

The interaction value is asymmetrical, so the interaction of feature 1 with feature 2 may not be the same as the interaction of feature 2 with feature 1. This property is used to detect hierarchy structure between the features which is then displayed in the *Hierarchies* portlet:



Segmentation Parameters

In the left panel, you define the parameters to run your segmentation.

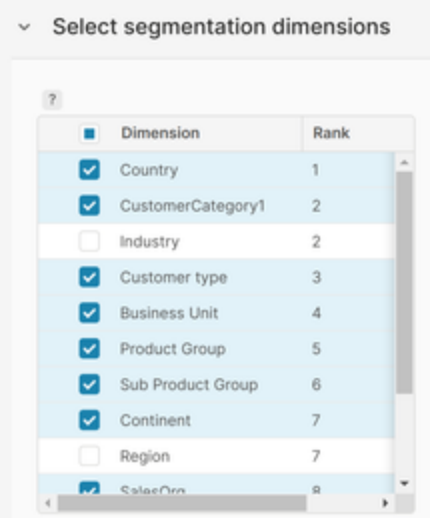
Segmentation Dimensions Selection

The first section, **Select segmentation dimensions**, is a table of the dimensions available for the segmentation. The rank column displays the result of the Segmentation Dimensions Recommendation port let. Only the price drivers set in the Analysis step can be selected as segmentation dimensions.

The dimensions are ordered, and some of them are preselected (for the first run of a new model, otherwise the previous selection is kept), according to the table Segmentation Dimensions Recommendation, in the right panel. Based on your business knowledge, you can reorder them or change the selected ones. Check all the fields that you want to use for the segmentation. You can drag and drop the lines to define the order of the segmentation levels.

You *must* select at least one and up to **twenty levels** of segmentation. Note also that:

- Dimensions with null values will be replaced with the label defined in the *Replacement value* in the Definition step, if used. Otherwise, the segmentation will return an error if a null value is found.
- You should not use a field used to map the transactions source as a segmentation level (product id, customer id). If you need to use one of them, please duplicate the field in the source Datamart and use one field in the Definition step, the other one here.



If you intend to use alignment in the next step, we advice to use the dimensions with alignment part of the first levels of the segmentation for better results, specially due to data sparsity.

Segmentation Thresholds

The second section of this left panel, **Segmentation Thresholds**, contains three minimum values. The segmentation tree will only build nodes that match *all* of these three thresholds.

Elasticity

The third section, **Elasticity**, lets you choose the elasticity model, either Sigmoidal or Exponential. This defines the kind of elasticity functions that will be fitted to each segment's data to get the elasticity parameters. There is also a checkbox **Calculate metrics based on elasticity**. If true, then the next step will not only calculate the elasticity function but also the projected quantity, revenue, and margin if the optimal target metric value is used.

Two parameters allow you to cancel the elasticity calculations for uninteresting or too large segments:

- **Min depth of leaves for elasticity calculation** – If the segment is deeper than this value to the leaf nodes, the elasticity is not calculated in it.
- **Max #Transactions in segment for elasticity calculation** – If the segment represents more than this amount of transactions, the elasticity is not calculated in it.

Elasticity Models

You can choose your elasticity model. The equations behind each elasticity model are in this table where q is the normalized quantity. The output of the elasticity function is a relative quantity: do not use the formula directly, but only in order to compare two different configurations.

Exponential model	Sigmoidal model
$q = \exp\left(\log(q_0) - \frac{A}{1 - M}\right)$	$q = \frac{L}{1 + \exp(-k\,(x - x_0))}$
where M is the optimization target value if this value is higher than the optimization target	where x is the optimization target value and x_0 is its reference value. L, k and x_0 are the elasticity parameters.

average in the segment and the optimization target average of the segment in the other case. *A* and q_0 are the elasticity parameters.	

Click the **Continue** button (top right) to go to the Segmentation step.

Segmentation Step (Optimization - Negotiation Guidance)

This step performs the segmentation itself, meaning that it builds the segmentation tree. It is a sequence of two calculations: first the tree build, then the calculation of the gaps between segments. Once the calculation sequence is done, the step displays five tabs: Tree View, Indicators, Price Gaps, Selection Alignments, and Optimization Setup. The first three tabs are dashboards to navigate into the segmentation results. Selection Alignments and Optimization Setup are where you set the inputs to the optimization itself.

Segmentation and Price Gaps Calculation

Segmentation

First, the calculation creates a segmentation tree, using the segmentation parameters entered in the previous step. Each segment represents a subset of the source data. Each segment contains some metrics, such as the customer and the product counts, revenue, optimization target average (either margin or discount percentage average), etc.

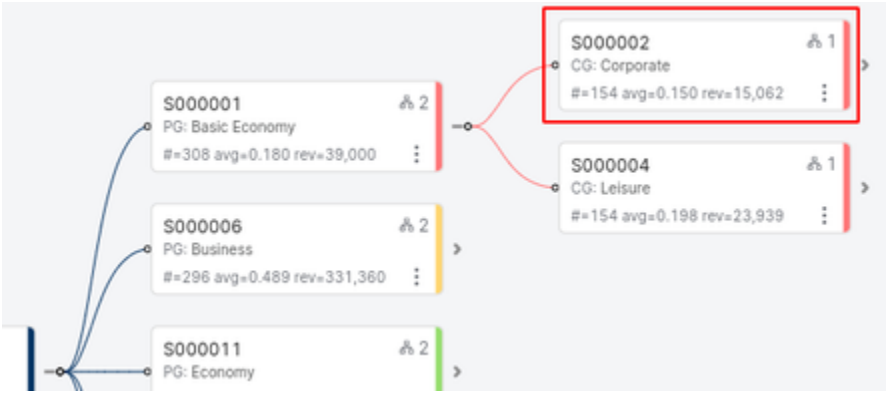
The parameters of the elasticity function for each segment are evaluated. The elasticity function can then be used as the optimum optimization target value to maximize either the profit or the revenue. Those elasticity parameters are provided in the segments' metrics.

Price Gaps Calculation

Then, the prices gaps between each segment and its parent are calculated. It is the reason why, during the calculation run, you see that there are two steps. This price gaps concept is explained below, in the documentation about the place where it is used: https://pricefx.atlassian.net/wiki/spaces/ACCDEV/pages/4804378860/Segmentation+Step+Optimization+-+Negotiation+Guidance#Price-Gaps.

Tree View

This tab is a dynamic view of the segmentation tree. You can expand and collapse it and get information about any segment.

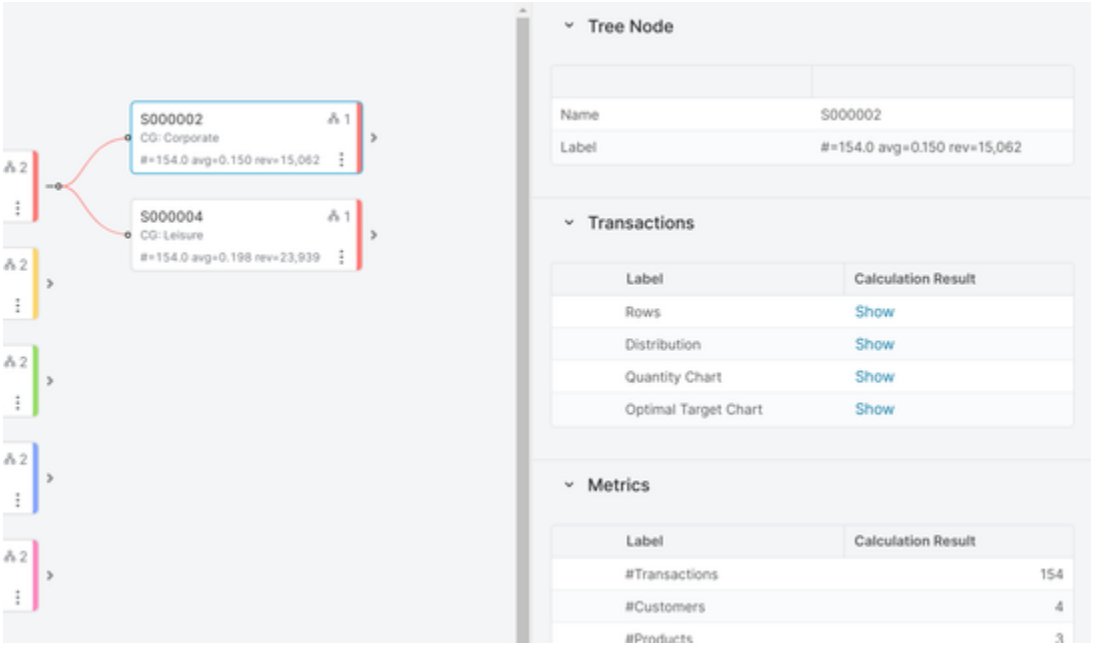


For each segment, i.e. node of the tree, the following information is directly given:

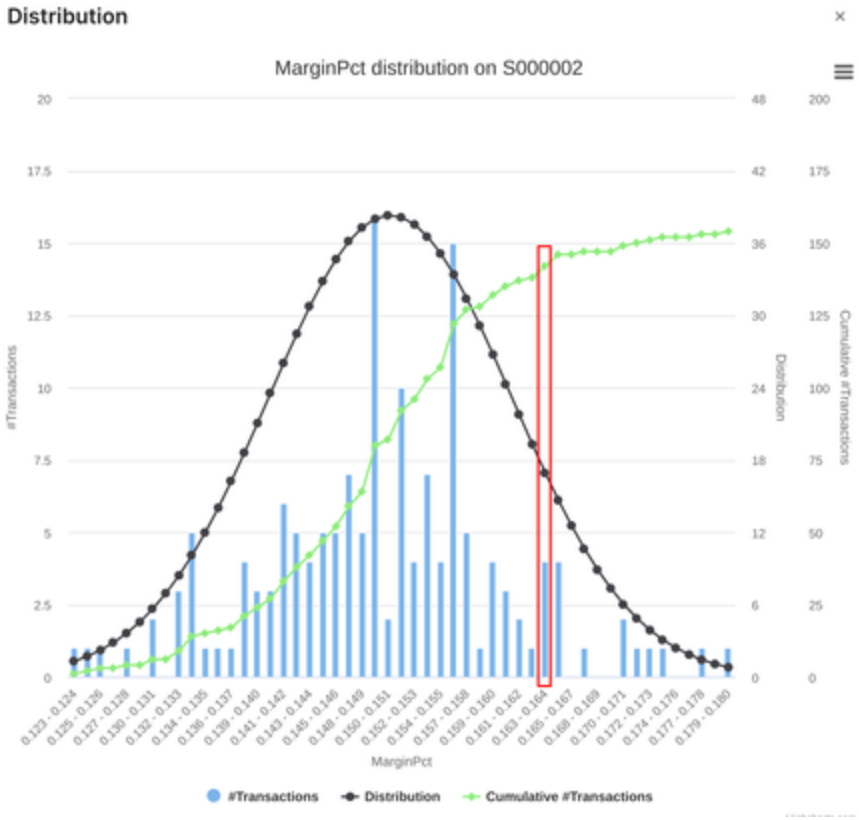
- The name of the segment (here *S000002*).

- The last level of segmentation and the value it belongs to. Here the level is *CG* and its value is *Corporat*
  . Given the previous node in the tree, it means that the segment S000002 corresponds to the transactions where the field *PG* has the value *Basic Economy* and the field *CG* has the value *Corporate*.
- *#* is the number of transactions in the segment (here 154).
- *avg* is the average value of the optimization metric (margin% or discount%) in the segment.
- *rev* is the total revenue represented by the segment.

If you click a segment, more information is provided in the right panel.



In addition to the information you already have in the tree view, you have some other metrics of the segment. In the segment Transactions, you also have access to the sample of input data present in this segment, two histograms, and a *Quantity Chart*. The first histogram is called *Distribution*.



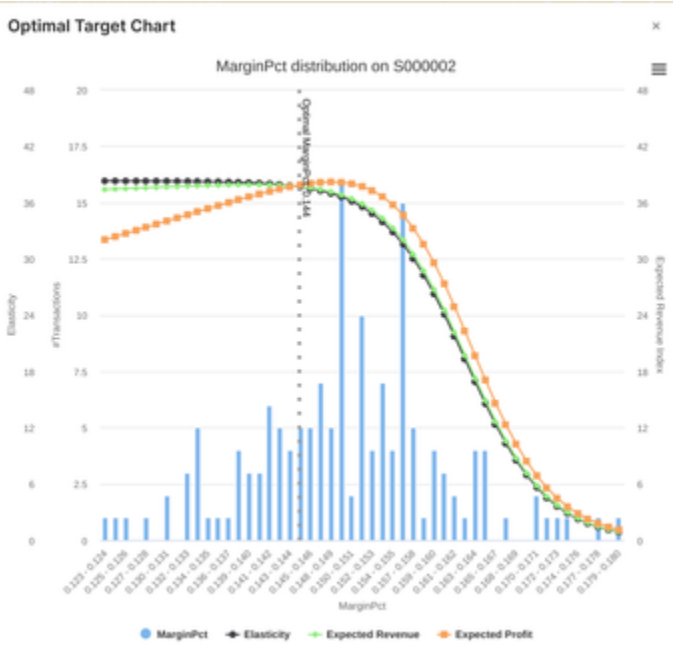
The blue bars represent the number of transactions depending on the optimization metric value. Here, four transactions were done with a margin percentage between 16.3 and 16.4%. The green curve is the cumulative value. Next to it, 142 transactions were done with a margin percentage inferior to 16.4%. The black curve is a fit of a normal law on the data. The average value is 15% and the normal curve does not fit well on the segment.

The *Quantity Chart* displays one dot for each transaction of the segment

The dots are placed on X axis according to the optimization target for the transaction – here the margin percentage – and on Y axis according to the quantity of the transaction. Note that for large segments only 500 transactions are displayed on the chart.

The second histogram is called *Optimal Target Chart*.

⚠ This chart is created only if the elasticity is calculated for the given segment. It depends on the parameters you chose to limit some elasticity calculations in the *Configuration* step.



The histogram is the same as the previous chart. On top of it, the fitted elasticity curve is displayed in black. Expected revenue and expected profit curves are also displayed. They are shown as non-dimensional values. That means that they refer to an index and not an absolute value. It is the reason why the profit curve can be higher than the revenue one. Their shapes are important: they show the maximum of each curve (what is the optimal target metric value, in terms of profit or revenue) and how much the metric decreases if the target metric is not at its optimum. The vertical line shows the optimal target metric value based on a mix of profit and revenue: profit optimum represents 2/3 of the weight and revenue one represents 1/3.

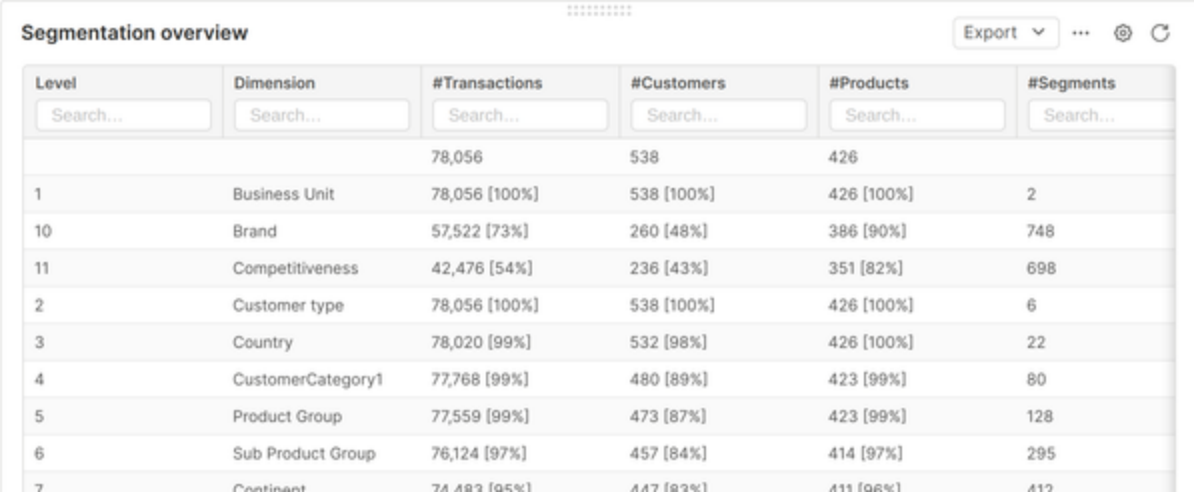
⚠ If the next step has run, its outputs are also displayed in this tree view. See Results Step (Optimization - Negotiation Guidance) for details.

Indicators

This tab provides metrics and data that allow evaluating the pertinence of the segmentation results. There are five portlets.

Segmentation Overview

This table shows what amount of data is kept by each level of segmentation and how much this level of segmentation describes the target metric variations. The count of segments existing at each level is also displayed.



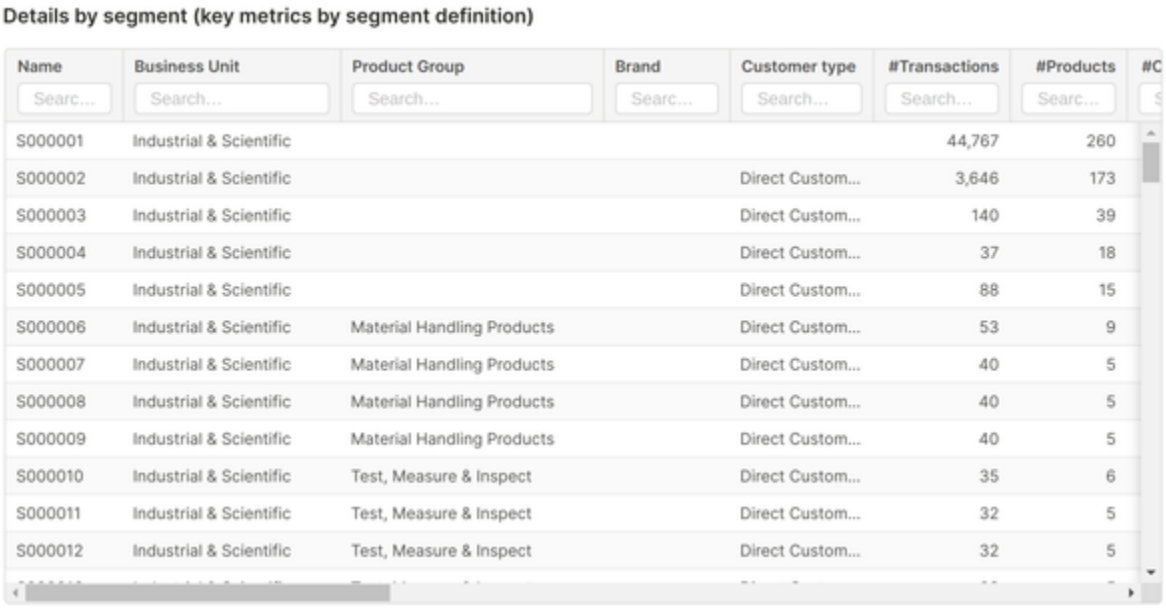
Segmentation overview

Export

Level	Dimension	#Transactions	#Customers	#Products	#Segments
		78,056	538	426	
1	Business Unit	78,056 [100%]	538 [100%]	426 [100%]	2
10	Brand	57,522 [73%]	260 [48%]	386 [90%]	748
11	Competitiveness	42,476 [54%]	236 [43%]	351 [82%]	698
2	Customer type	78,056 [100%]	538 [100%]	426 [100%]	6
3	Country	78,020 [99%]	532 [98%]	426 [100%]	22
4	CustomerCategory1	77,768 [99%]	480 [89%]	423 [99%]	80
5	Product Group	77,559 [99%]	473 [87%]	423 [99%]	128
6	Sub Product Group	76,124 [97%]	457 [84%]	414 [97%]	295

Details by Segment

This table summarizes the key metrics of all the segments in one place.

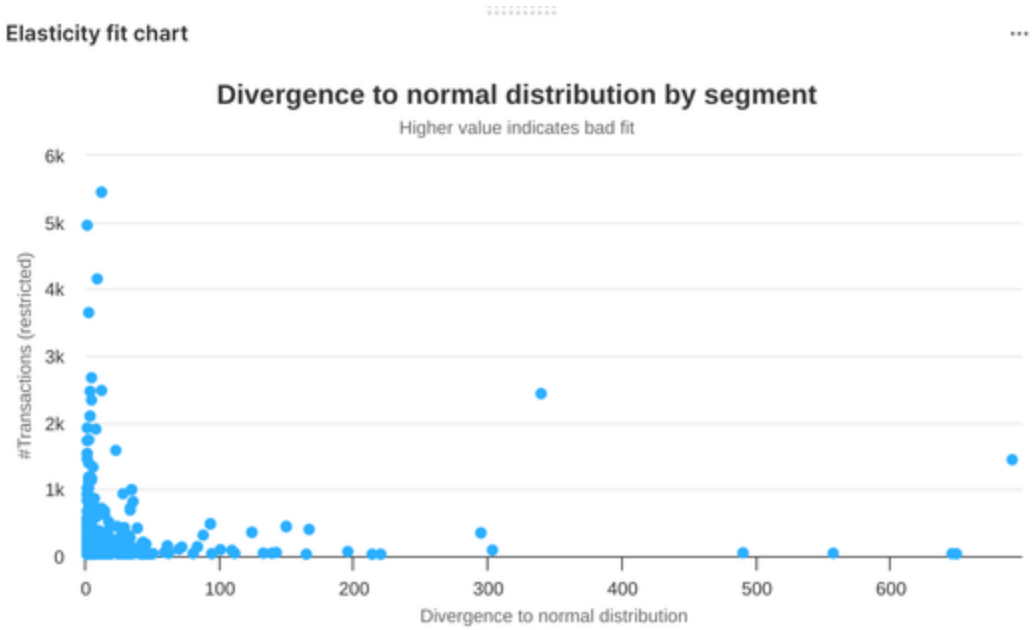


Details by segment (key metrics by segment definition)

Name	Business Unit	Product Group	Brand	Customer type	#Transactions	#Products
S000001	Industrial & Scientific				44,767	260
S000002	Industrial & Scientific			Direct Custom...	3,646	173
S000003	Industrial & Scientific			Direct Custom...	140	39
S000004	Industrial & Scientific			Direct Custom...	37	18
S000005	Industrial & Scientific			Direct Custom...	88	15
S000006	Industrial & Scientific	Material Handling Products		Direct Custom...	53	9
S000007	Industrial & Scientific	Material Handling Products		Direct Custom...	40	5
S000008	Industrial & Scientific	Material Handling Products		Direct Custom...	40	5
S000009	Industrial & Scientific	Material Handling Products		Direct Custom...	40	5
S000010	Industrial & Scientific	Test, Measure & Inspect		Direct Custom...	35	6
S000011	Industrial & Scientific	Test, Measure & Inspect		Direct Custom...	32	5
S000012	Industrial & Scientific	Test, Measure & Inspect		Direct Custom...	32	5

Elasticity Fit Chart

In this scatter chart, each dot represents a segment. The X axis is the divergence to a normal distribution (see above), so the segments on the right are the ones that less fit a normal distribution. The Y axis is the number of transactions in the segment. The segments which are far from the leaves in the segmentation should have more transactions in them and can be farther from a normal distribution (top right of the chart).



Elasticity Fit by Segment

This table provides the divergence to normal distribution (numerical value) for all the segments. The higher the value, the less the segment fits a normal distribution. The parenting level indicates how much the segment is far from the last segmentation level.

Elasticity fit by segment

Segment	Divergence to normal distribution	Parenting Level
Search...	Search...	Search...
S000001	0.321	10
S000002	2.197	9
S000003	1.46	8
S000004	1.312	7
S000005	4.76	7
S000006	4.118	6

Fitting of Elasticity

This portlet provides the average divergence to a normal distribution of all the segments (see paragraph above). The higher this value, the less the segments fit normal distributions.

Price Gaps

Price Gaps across Segments

This table provides for each value of the selected segmentation level:

- the global revenue of the segments corresponding to each value,
- the average optimization target metric through the involved segments,
- the average price gap to the overall mean. So this is computed for each segment related to the parent segment and then aggregated.

Price gaps across segments

Customer type	Revenue	Average Margin rate	Average price gap to overall mean
Industry	365,108,223.49	48.37%	2.17%
Direct Customer	1,236,413.75	23.46%	-19.07%
Distributor	7,077,108.76	24.25%	-18.32%

The price gaps are a way to evaluate how much the segments differ from each other at each level of segmentation. The idea is to evaluate how much a price depends on an attribute by checking the impact on the margin rate or the discount rate. The following formulas are used, with the reference being the one level higher in the segmentation tree:

Margin as an optimization target:

$$Price\ gap = \frac{1 - M\%_{ref}}{1 - M\%} - 1$$

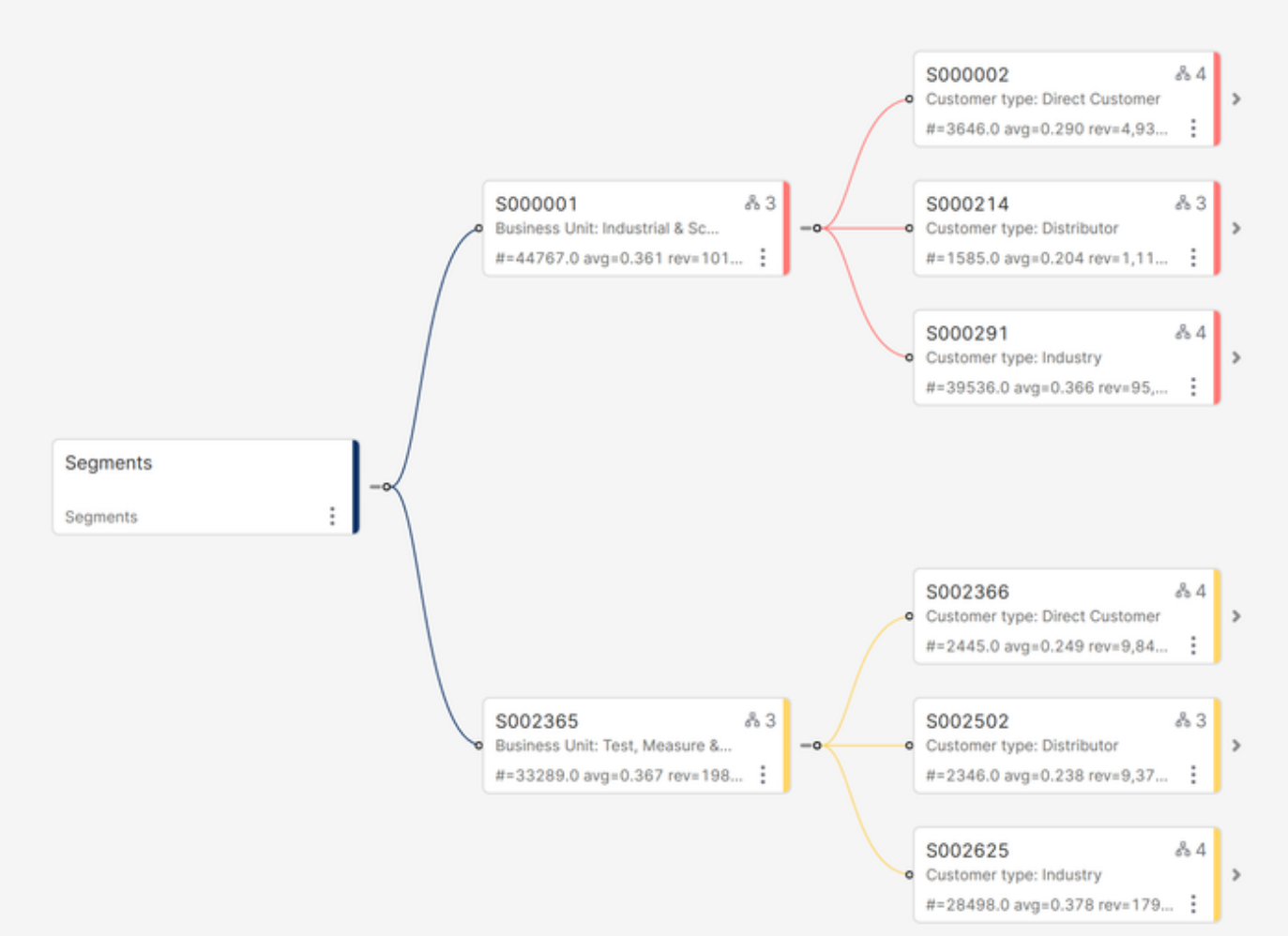
Discount as an optimization target:

$$Price\ gap = \frac{1 - D\%}{1 - D\%_{ref}} - 1$$

For each segment, the gap between its average optimization target and its parent segment one is calculated. Then, at each segmentation level, those values are aggregated by segmentation level value. Moreover, the average optimization target and the total revenue for the segments of this level are calculated too.

As it is complicated, let's propose an example.

Consider this segmentation tree, where the first level is *Business Unit*, and the second one is *Customer type*.

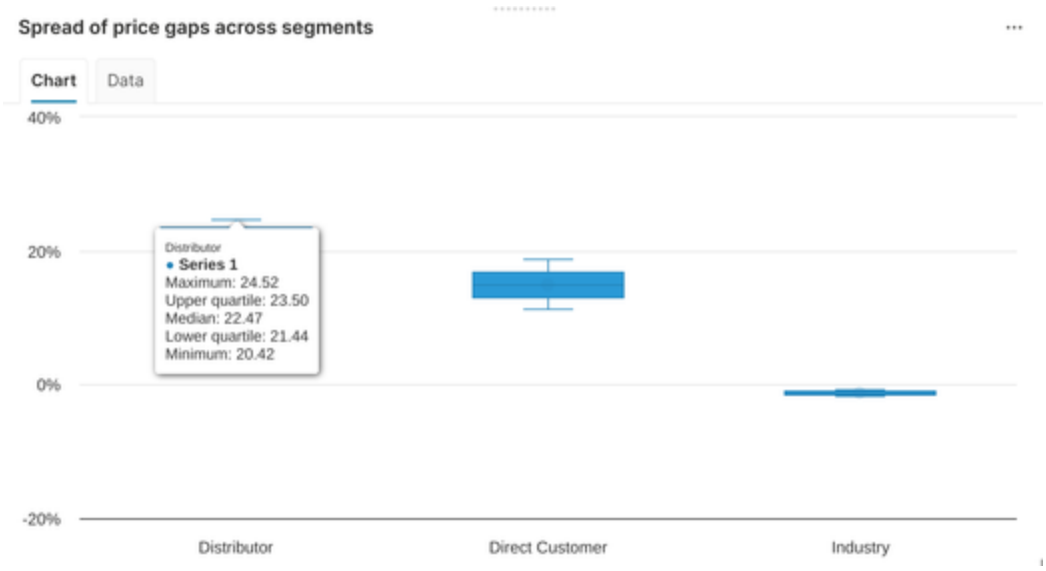


If you consider the *Customer type* level, for the value *Distributor* then:

- The total revenue calculated here is the sum of the revenues of the segments S000214 and S002502.
- The average optimization targets are the average values of the segments S000214 and S002502 weighted with their revenues (in case of margin percentage) or total list prices (in case of discount percentage).
- The gaps between S000214 and S000001 on one hand and S002502 and S002365 on the other hand are calculated. Then, the average of these two gaps is the price gap of the *Distributor* value, for the *Customer type* level. It is implicit that the reference is the *Business Unit* level.

Spread of Price Gaps across Segments

This box plot chart displays the spread of the price gaps of the segments with their parent, for each value of the selected segmentation level.



Selection Alignments

From v2, it is possible to align the segments between each other. The alignments are optional. By default, the option is unchecked. You will have access to the user inputs if you click the checkbox **Enable Price Alignment Criteria**. This option is available only if no **Weight Measure** has been selected at the Definition step .

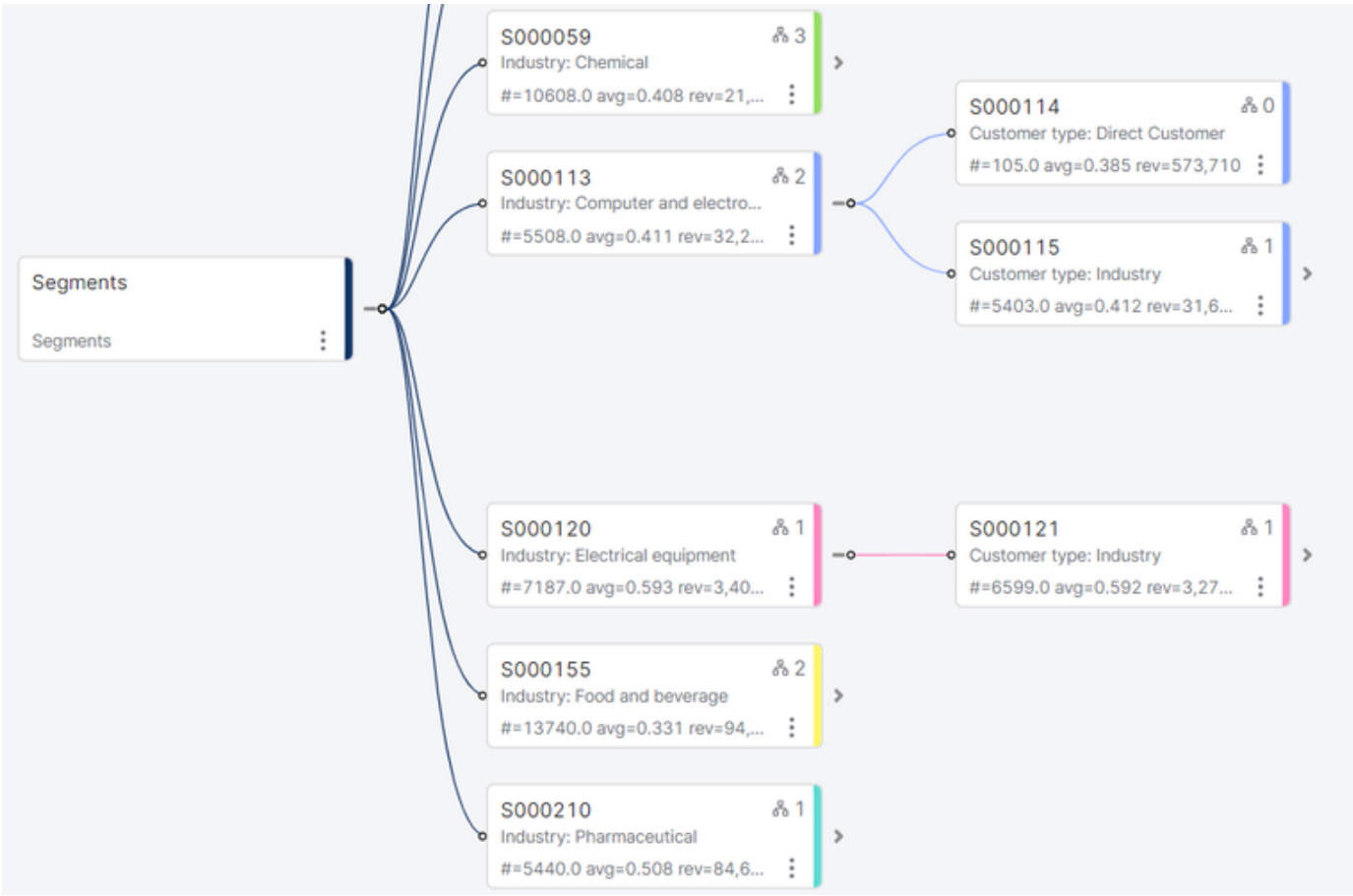
Let's start with few recommendations about the alignment. Multiple alignment might be challenging to enforce as it might get into different directions. So we recommend:

- Get the dimensions with the alignment as the first ones in the segmentation tree. This helps to avoid having a low number of comparison points due to data sparsity.
- Better set the **alignments on customer or transactions** related attributes and **not on product attributes**. Indeed, list prices (or costs for margin%) should already reflect the consistency of prices related to product attributes. Saying there is 5% price gap between two brands would mean there is a difference of discounts that results in a 5% price difference, but that difference should be better directly embedded in the list price than adjusting the discount here.
- Limit to a few alignments in a few dimensions that are the most important ones from a business point of view. It is also difficult to enforce many changes at the same time from a business perspective as it will impact customers more.
- Avoid setting a specific percentile value in the parameters table. Indeed, this can shifted differently segments and alignments might not be satisfied anymore.

What are the alignments?

You may consider that some segments should have, on average, prices that are tied to similar segments on all segmentation levels except one. Let's take an example.

Here is a segmentation tree:



The pricing manager considers that the prices of *Direct Customer* should be 25% lower, on average to *Indu stry* customers. For this, you set an alignment at the *Brand* level. The optimization target values of all the product and customer pairs in the scope of the model will be moved in order for the average price gaps between the segments S000114 and S000115, among the other segments too, to be as close as possible to this gap of -25%. The segment S000121 is not aligned with any segment because there is no similar segment to align with.

For the definition of a price gap, refer to https://pricefx.atlassian.net/wiki/spaces/ACCDEV/pages/4804378860/Segmentation+Step+Optimization+-+Negotiation+Guidance#Price-Gaps. Here, we are speaking about price gaps between two segments in the same level of segmentation.

To do this, an Optimization Engine is automatically configured and executed behind the scene. For each alignment rule, all the relevant pairs of segments are aligned together.

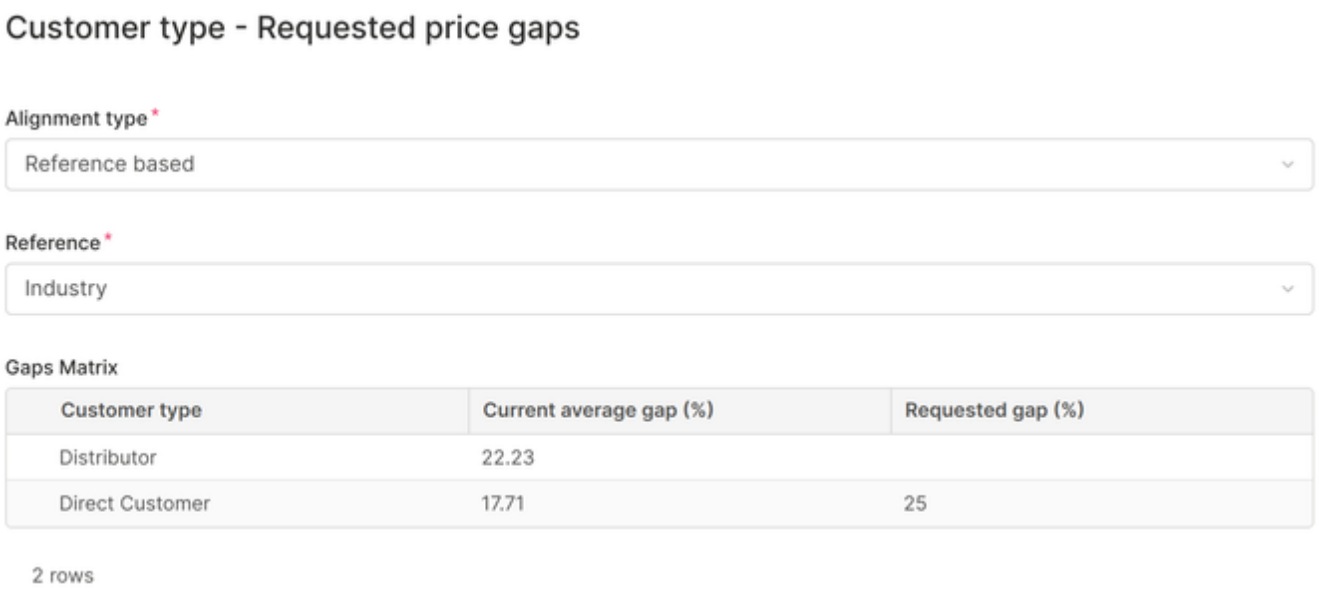
How to configure the alignments?

If the button "Enable Price Alignment" is checked, the tab proposes the segmentation levels to set up alignments. To help the choice, the cardinality (count of different values) is displayed for each segmentation dimension. At most one dimension can be marked as recommended: it is the segmentation level coming from customer master data that has the maximum price driver importance, with a cardinality of less than thirty. You can select the alignments as you want. For each selected row, a new section is displayed, called *"name of the dimension" - Requested Price Gaps*.

First, choose the alignment type. For the moment, there is only one option: *Reference based*. Other type of alignment will be implemented in the future. *Reference based* means that you choose a reference value and you define the price gaps that the similar segments corresponding to another dimension value should reach in average. To help you decide the gap to apply, the average current gap value is displayed. If you

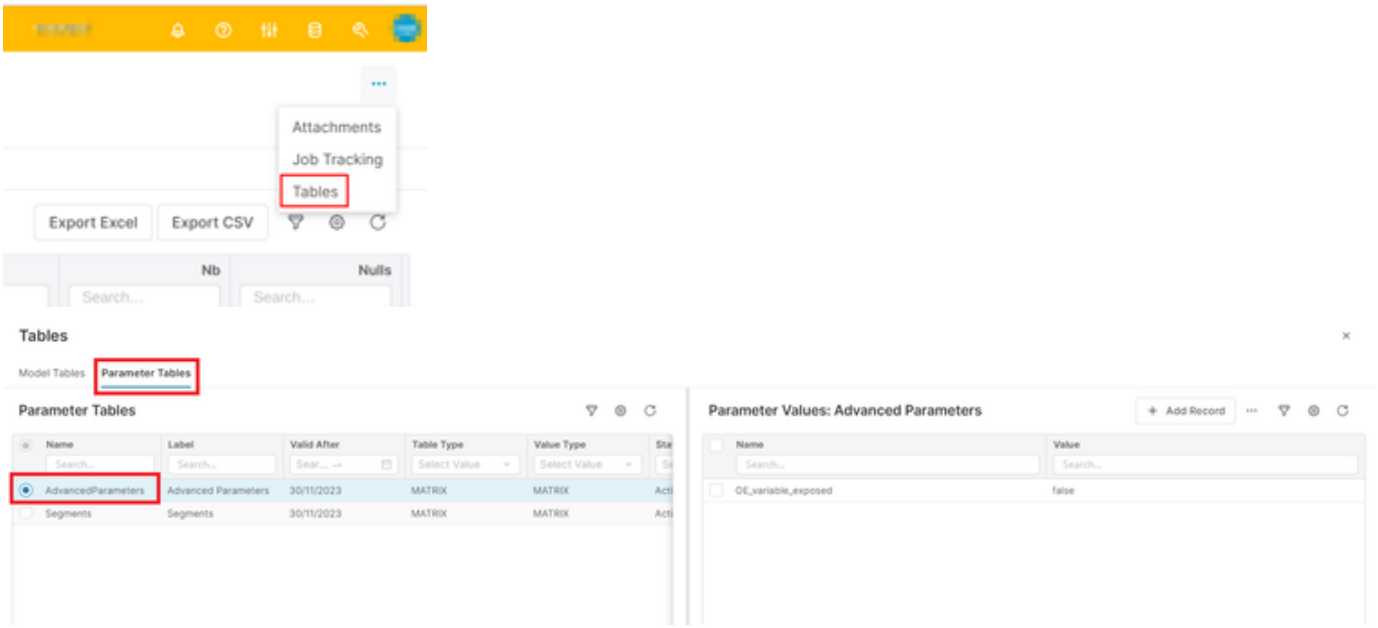
go back to the example below: the current gap between *Direct Customer* and *Industry* customer is the average of the gaps between the segments S000114 and S000115, but also all the other pairs of segments at the same level of the segmentation tree, that are not displayed in the screenshot.

If you select an alignment, at least one requested gap must be set. You do not need to create gaps for each pair of values. In the example below, there is no requested gap for the customer type *Distributor*, compared to *Industry*. This means that no alignment constraint will be set on the segments corresponding to the *Distributor* customer type.



Debug the Alignments

When using the alignment, if the Optimization Engine fails to achieve the requested alignment, it is possible for advanced users to trigger some additional process to analyze the OE run. To do so, go to the Tables menu, to the Parameter Table called *AdvancedParameters* and change the value of OE_variable_exposed from 'false' to 'true'.



If set to true, this variable will expose additional information related to the OE internal process, and create additional table available in Model Tables. It is recommend to use this option only if the OE is not functioning well, as setting this variable to 'true' can make the process considerably slower.

Optimization Setup

The Optimization Setup tab allows you to define the way to provide an optimum for each segment. You define three global percentiles. It is possible to override these global values for some segments (see below).

How the Percentile Values Are Used?

Three percentiles and four strategy coefficients define the optimization strategy.

The percentile values are used to split each segment among the optimization target values. For instance, a floor percentile of 30 means that 30% of the transactions in each segment will be considered "below the floor". The floor and the ceiling percentiles are required, the target percentile is optional: if not provided, a specific value will be calculated for each segment, based on the score segment (see below).

The strategy coefficients are required. They indicate how much the optimization target values will move toward the next threshold in the optimization calculation. The optimization based on the percentiles is performed like this:

- If a transaction was done with an optimization metric *lesser* than the floor percentile value, it is increased toward the floor percentile value, by the amount defined with the *first* strategy coefficient.
- If a transaction was done with an optimization metric *between* the *floor* percentile value and the *target* percentile one, it is increased toward the target percentile value, by the amount defined with the *secon* strategy coefficient.
- If a transaction was done with an optimization metric *between* the *target* percentile value and the *ceili ng* percentile one, it is decreased toward the target percentile value, by the amount defined with the *th ird* strategy coefficient.
- If a transaction was done with an optimization metric *upper* than the ceiling percentile value, it is decreased toward the ceiling percentile value, by the amount defined with the *fourth* strategy coefficient.

Score Calculation

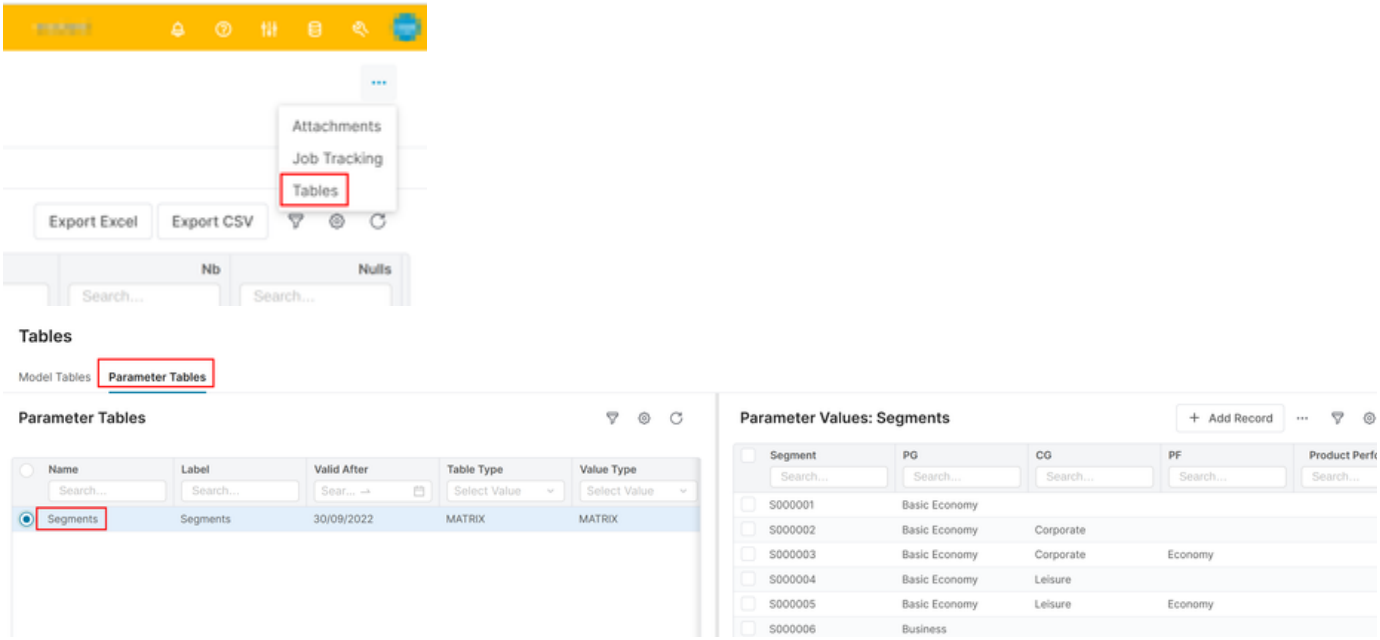
It is mandatory that the user defines the floor and the ceiling percentiles. The target one is optional. If it is not defined, it is calculated, based on the segment score. The score can vary from 0 to 100.

- For "Margin%" Target Type: if score = 0, the target percentile will be the max(floor, 50) percentile; if 100, it will be the ceiling percentile; intermediary values of percentile apply to intermediary scores, using a linear interpolation between floor and ceiling percentiles.
- For "Discount%" Target Type: if score = 0, the target percentile will be the min(ceiling, 50) percentile; if 100, it will be the floor percentile; intermediary values of percentile apply to intermediary scores, using a linear interpolation between floor and ceiling percentiles.

The score can be set by two methods. The default one is the CoV method. The higher the coefficient of variation of the optimization metric in the segment (CoV) is, the higher the score is. You can also define a value of Competitive Intensity (between 1 and 5) and a value of Product Performance (between 1 and 5) and then the score is calculated on top of them.

Override Percentiles for Some Segments

If the global percentiles are not to be used on some specific segment, you can override them by checking the box **Use percentile values from the parameters table when present**. Once it is checked, the values used will be the ones provided in the Parameter Table called *Segments*. Use the three dots at the top right of the model and then go to the Parameter Tables tab.



Any field of this table is editable if there is no calculation processing. Be careful not to change the values defining the segments' names and levels, but only the next fields.

Results Step (Optimization - Negotiation Guidance)

This step first optimizes the segments and the transactions in the scope of the model. The optimization is composed of two parts: the first one, optional, is the alignments of the segments; in the second part, the floor, target, and ceiling percentile values are applied to the source data to simulate the optimization strategy on the historical data. Once the calculations have run, six tabs are displayed. They provide different views on the optimization results. Last, but not least, two evaluations are available: they are ways to use the optimization results from any other part of the partition, such as Quotes, Price Lists etc.

Alignment and Optimization Calculations

When you arrive at the *Results* step from the Segmentation step, the model first runs the alignments of the segments, then the optimization calculation of each segment. It may take some minutes, depending on the size of the model. It is a sequence of six calculations, so the interface may display an advancement of n over six (it may be less, at the beginning of the process, or if no alignments are set).

First, an Optimization Engine is triggered, then change of the optimization target values (margin rate or discount rate) in order for some segments to be aligned with the gap defined previously. Then, the percentile values entered as the floor, target, and ceiling, are applied to the aligned optimization target values in each segment, with the defined strategy coefficients. Refer https://pricefx.atlassian.net/wiki/spaces/ACC/pages/4828954970#Selection-Alignments and https://pricefx.atlassian.net/wiki/spaces/ACC/pages/4828954970#Optimization-Setup for more details.

Impact Tab

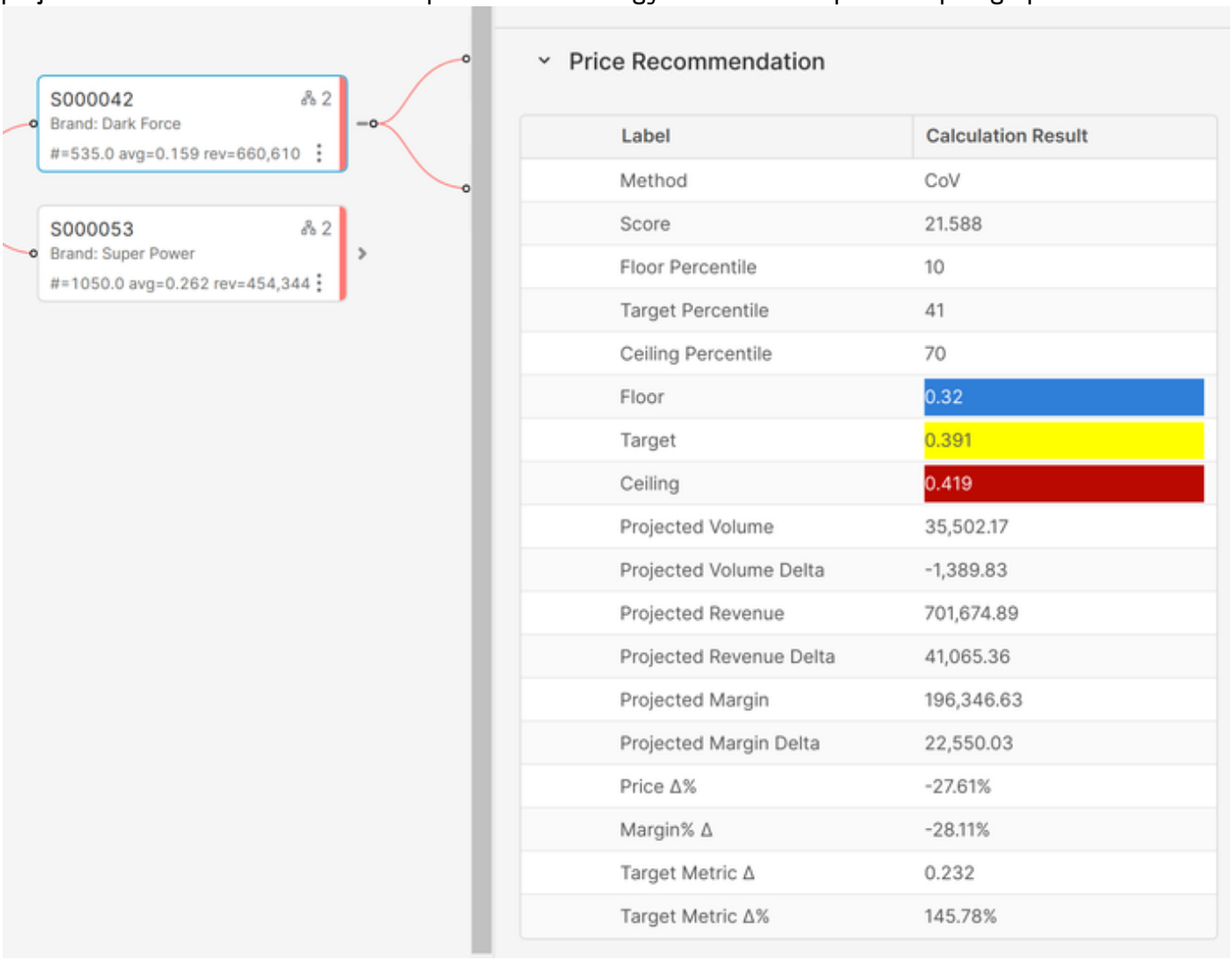
This dashboard displays the optimization result for the strategy floor-target-ceiling. Refer to the paragraph https://pricefx.atlassian.net/wiki/spaces/ACC/pages/4828954970#How-the-Percentile-Values-Are-Used%3F for more explanations.

The left panel allows you to choose the granularity of the bar charts and the overall realization dashboard.

- **Segmentation level** - All the transactions of the scope are aggregated at this level to provide the bar charts of the projected revenue and profit. The global overview and the profit potential matrix are not impacted by this input value.
- **Overall Realization %** - The optimization strategy is the best to have. You can simulate with this input the fact that only a certain percentage of the optimization target is reached. The value must be between 0 and 100 and will impact all the dashboard portlets.

Tree View

This view is the same as https://pricefx.atlassian.net/wiki/spaces/ACC/pages/4828954970#Tree-View but now more results are provided when clicking on a segment. The Price Recommendation section provides the score, target percentile, values of the optimization metric for the given percentiles, and projections of metrics based on the optimization strategy defined in the previous paragraph.



Note: Floor, target and ceiling displayed here use the alignment if any is defined.

Recommendations Tab

Recommendations and Metrics per Segment

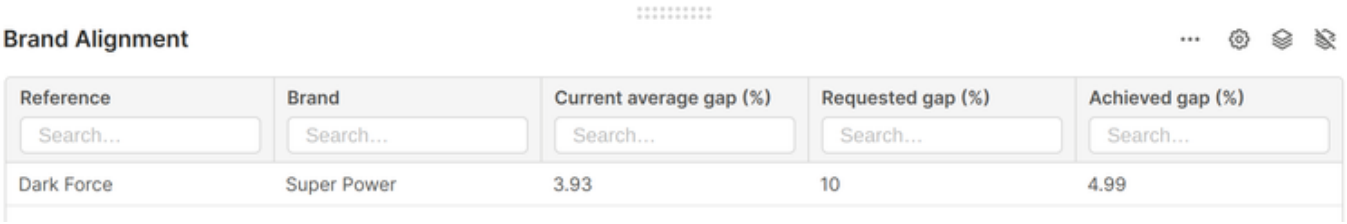
This portlet provides the segments table with all the optimized values for each segment. In particular, the optimization values are provided for the revenue, quantity, and optimization metric.

List of the fields:

- **Name and the levels of segmentation** - Define each segment.
- **#Transactions, #Products, #Customers** - Number of distinct transactions, products, and customers inside the segment.
- **SSE** - Sum of squared errors in the segment, based on a hypothesis of a constant value of the optimization metric.
- **Divergence to normal distribution** - Measures the difference between a perfectly normal distribution and the actual distribution of the target metric in the segment. The smaller the value, the more the segment distribution is close to a normal distribution.
- **CoV** or **Weighted CoV** - Coefficient of variation of the optimization metric, depending on the use or not of a Weight field, i.e. the standard deviation divided by the average.
- **Target std** or **Weighted Target std** - The standard deviation of the target metric.
- **Target avg** or **Weighted Target avg** - The average of the target metric.
- **Revenue** - Sum of the transactions revenues in the segment.
- **List price** - Sum of the extended list prices in the segment (provided if the optimization target is set to *Discount%* only).
- **Margin** - Sum of the transaction margins in the segment.
- **Margin Rate** - Division of Revenue by Margin.
- **Quantity** - Sum of the quantities revenues in the segment.
- **Parenting Level** - Number of levels from the segment to the leaves of the segmentation tree.
- **Scoring Method and Score** - Explained in https://pricefx.atlassian.net/wiki/spaces/ACC/pages/4828954970#Score-Calculation.
- **Floor Percentile**, **Target Percentile**, **Ceiling Percentile** - Explained in https://pricefx.atlassian.net/wiki/spaces/ACC/pages/4828954970#How-the-Percentile-Values-Are-Used%3F.
- **Floor**, **Target**, **Ceiling** - Values of the optimization metric for each of the defined percentiles.
- **Initial Floor, Inital Target, Initial Ceiling** - Values of the optimization metric before alignments
- **Price %**, **Margin%** , **Target Metric** , **Target Metric %** - Deltas between the actual values and the ones that you would have if you used the optimizations rules to the floor/target percentiles.
- **Elasticity Parameters** - Parameters of the elasticity curve fitted on the segment, in a JSON format, to make it usable by another piece of code outside of the model.
- **Optimal Target Metric**, **Optimal Target Metric For Revenue**, and **Optimal Target Metric For Profit** - Values that maximize the profit or revenue curves built using the elasticity formula. The global optimal target value is a pondered average of the profit optimum (2/3) and the revenue one (1/3).
- Then many metrics are projected, based on these optimal target metrics, and the deltas between the actual values and the optimized ones are also presented: quantity, margin, revenue.

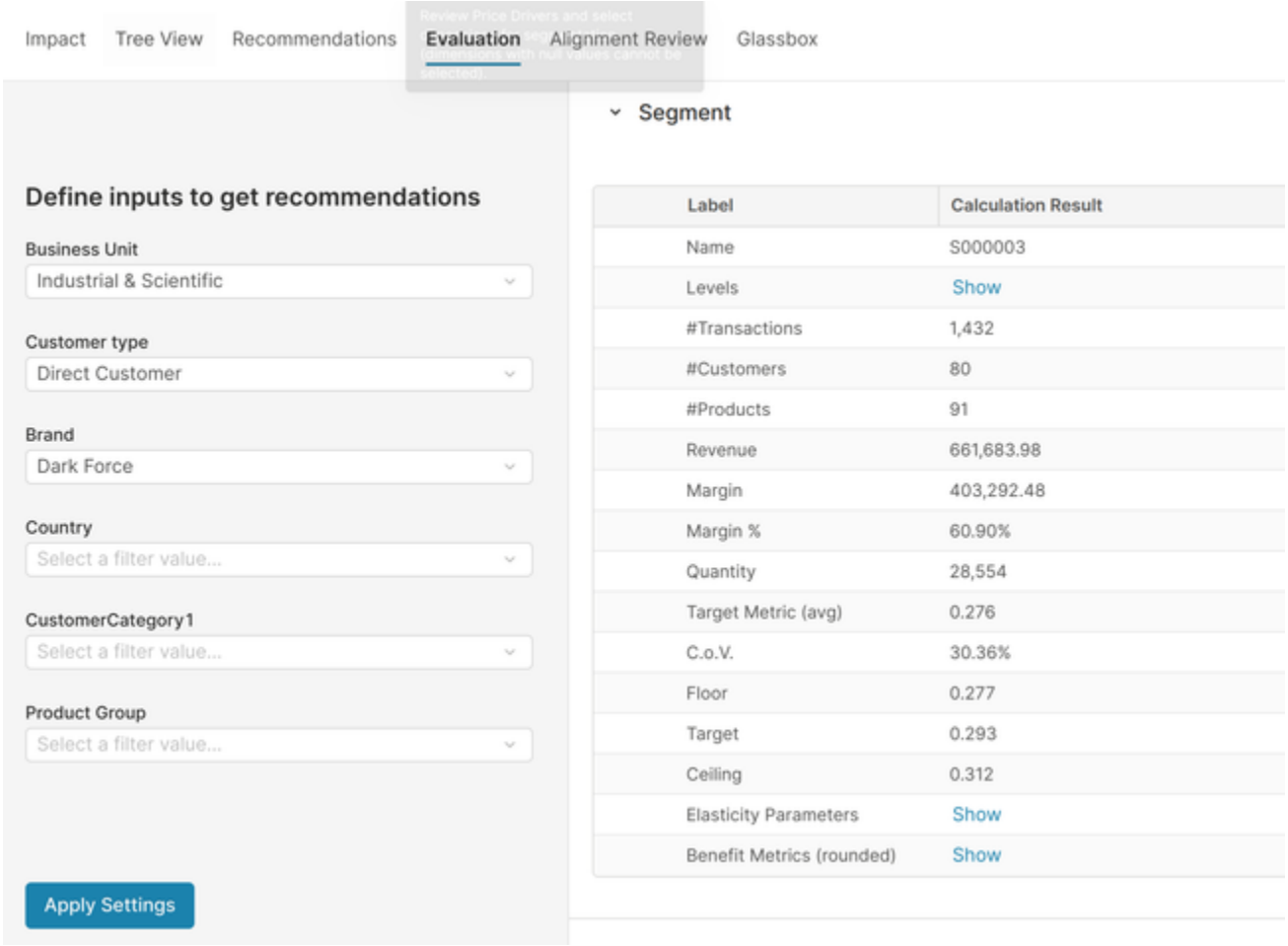
Dimension Alignment Portlets

For each dimension alignment set in the calculation, a corresponding portlet is displayed. They are tables where a row is an alignment rule. The current, requested, and achieved gaps are provided. It is a global view to check if the requested gaps were reached, or how far the optimized results are for the requests.



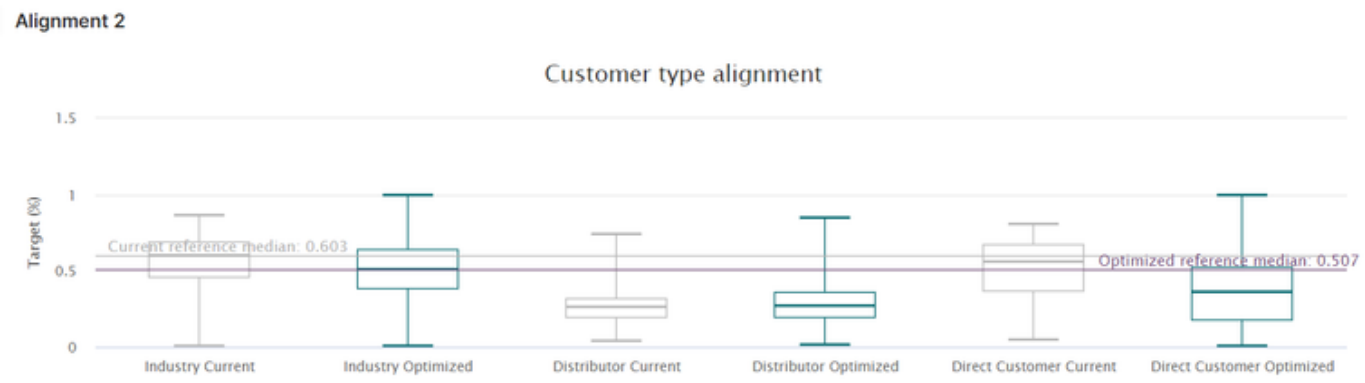
Evaluation Tab

This tab simulates a call of the `query_segment` evaluation of the model, which can be called from any other part of the partition (such as price lists or quotes). The goal is mainly for an advanced user to test the logics, or to see what the inputs and outputs of the model evaluation are.



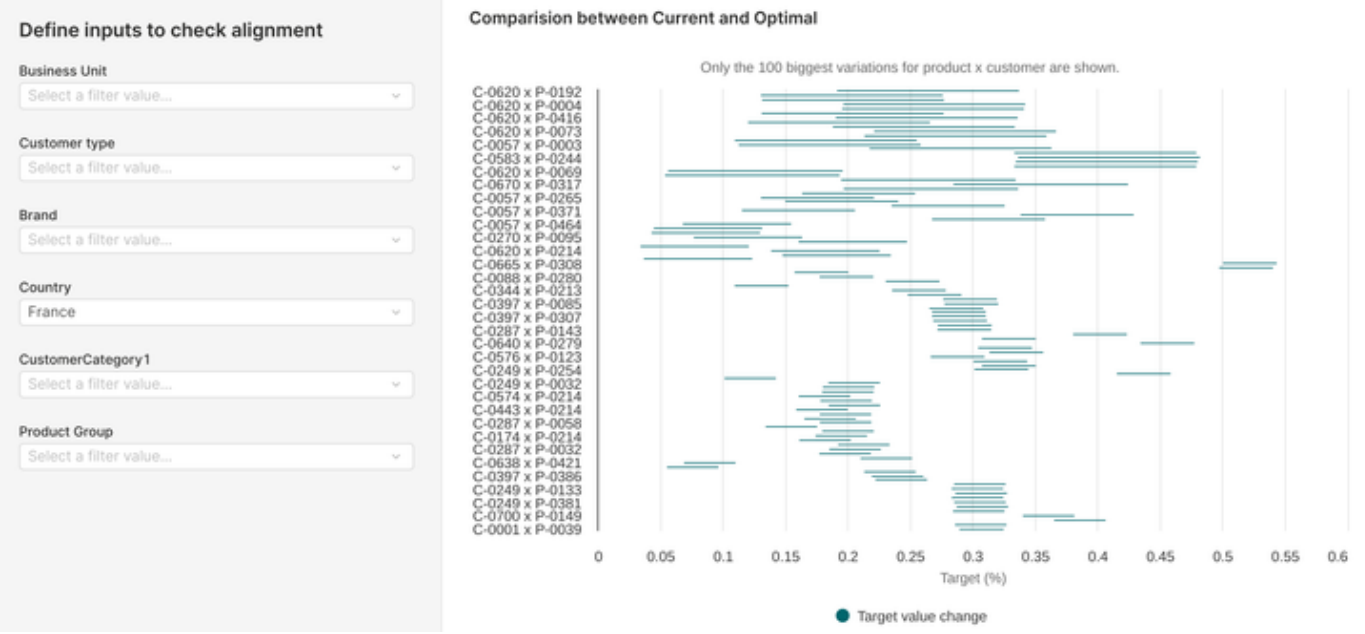
Alignment Review tab

In this dashboard, a boxplot chart is displayed for each alignment set in the optimization.



Each value of the dimension displays two boxes: one for the current values (before alignment, in light grey) and one for the optimized ones (after alignment). Each box represents a summary of the optimization target values corresponding to the given dimension value. On the chart above, for instance: the summary of all the discount rates in the scope of the model, where the *Customer Type* value is, for instance, *Distributor*. We can see in this example that to reach the requested alignments, the *Distributor* di scounts globally increased and the *Industry* discounts globally decreased.

The last chart, *Comparison between Current and Optimal*, displays a sample of product-customer pairs and their optimization target variation due to the alignment work. It is the only portlet of the dashboard affected by the set of left panel user entries. At most 100 product-customer pairs are displayed: the most impacted ones. The horizontal bars show the range of the change.



Glassbox Tab

This tab is a set of advanced charts to understand how the optimization engine works. It is for mostly for advanced users. Refer to Glassbox Dashboards for more details.

Still, it is interesting to check quickly how good the alignments are satisfied, from current situation to the recommendations. In this example, all alignments are satisfied or at least acceptable (meaning getting close to the alignment that has been defined)

Model Evaluation from Another Part of the Solution

If you need to call the model evaluation from any Pricefx module, two different evaluations are available: `query_segment` and `batch_query`. The general way to query a model is explained in https://pricefx.atlassian.net/wiki/spaces/KB/pages/3851026646/Query+Optimization+Engine+Results#Using-the-Evaluator.

Query Segment

The user provides the segmentation levels and gets all the outputs corresponding to this specific segment. It is useful and quick to get the optimization values for *one* specific set of data. The command is:

```
def model = api.model("TheModelUniqueName")
def results = model.evaluate(
  “query_segment”,
  [
    segmentationLevel1: "someValue", // keys depend on the segmentation
    //...
  ]).results
def target = results['Target'] // or any other output
```

Batch Evaluation

It is a way to get a *large bunch of optimization results* in one command. The user provides an input query and a fields mapping and the evaluation returns a SQL query. It is an advanced feature, that provides a quick result for a large bunch of data in one command. The usage is:

```
def dmCtx = api.getDatamartContext()
DatamartContext.Query myQuery = dmCtx.newQuery(dmCtx.getFieldCollection
("DM.myDM"))
        .selectAll(true)
        .where(Filter.equal("Country", "France"))
DatamartContext.SqlQuery batchSqlQuery = api.model
("myNegotionGuidanceModel").evaluate(
       "batch_query",
        [
                sourceQuery: myQuery,
```

```
                mapping: [
                        quantity: 'myQuantityField',
                        optimization_target: 'marginPct',
                        segmentationLevels: ['productFamily',
'channel', 'country'],
                        otherFields: ['product', 'customer_id'],
                ]
        ]
)['AsBatchQuery']
api.getDatamartContext().executeSqlQuery(batchSqlQuery)
```

This model evaluation can be called from any place of Pricefx, based on an existing model that has run. The inputs are:

- Query that fetches data to evaluate from a Negotiation Guidance point of view.
- Some mapping information to detect which fields are the quantity, the elasticity based-on variable, the segmentation levels, and the fields to output. *The order of the segmentation levels is important*. The required keys of the mapping map are `quantity`, `optimization_target`, `segmentationLevels`, `otherFields`.

The output is an SQL query that can be executed to get, for each row of the executed input query:

- All the fields defined in the mapping argument.
- `name` - The name of the associated segment.
- Some NG results relative to the segment, corresponding to the elasticity and the optimization thresholds: `ElasticityParameters`, `elasticity`, `normalization_coefficient`, `target_avg`, `weighted_target_avg`, `target_std`, `weighted_target_std`, `floor_value`, `target_value`, `ceiling_value` (if the names have not been changed in NG_Lib.ParametersLib).

Note: this batch evaluation requires a datamart (or data source or a model table) as an input, this is mandatory how the way this function works, in doubts, you should use the datamart that is defined in the first step of this Negotiation Guidance model.

Admin User Reference (Optimization - Negotiation Guidance)

Installation (Optimization - Negotiation Guidance)

The Negotiation Guidance Accelerator is an accelerator package. The accelerator deploys the Negotiation Guidance Model Class and the related logics.

Prerequisites

Before you start the installation of the accelerator, ensure you have the according *Transactions* Datamart or Data Source. Some important points about it:

- It must contain the required fields - create them if needed:

    - Customer field – ID of each customer.
    - Product field – ID of each product/article/SKU.
    - Quantity field – Number of products in each transaction.
    - Revenue field – Extended price of the transaction row.
    - Margin field – Revenue minus cost.
    - Weight field (optional) – Transaction weight.
    - Target field – Generally, it is a margin rate (margin/revenue), or a discount rate (discount/revenue). Be careful, do not use a percentage-formatted equivalent (same formula, multiplied by 100).
    - The segmentation levels must be defined as dimensions in the transaction source. If necessary, mark any field as a dimension in the Datamart.
- Avoid null values in segmentation dimensions. If necessary, create a new field by using an expression that replaces empty values with a text like "Not provided".

Deployment

1. Access PlatformManager at https://platform.pricefx.com/ and log in with your account or using O365. If you do not have an account yet, contact Pricefx Support.
2. Go to **Marketplace** and find the *Optimization - Negotiation Guidance* accelerator.
3. Click the accelerator tile, select the partition where you want to deploy the accelerator package and confirm the deployment dialog to start.
   For detailed description of all deployment options, see PlatformManager documentation.
4. Set up the default mapping.

Data Mapping

Identify the source fields in the transactions data to complete the mapping.

Source Type *

Datamart

Source Name *

NegativeMargin

Customer Field *

CustomerID

Product Field *

ProductID

Quantity Measure *

Quantity

Revenue Measure *

Revenue

Margin Measure *

Margin

Weight Measure

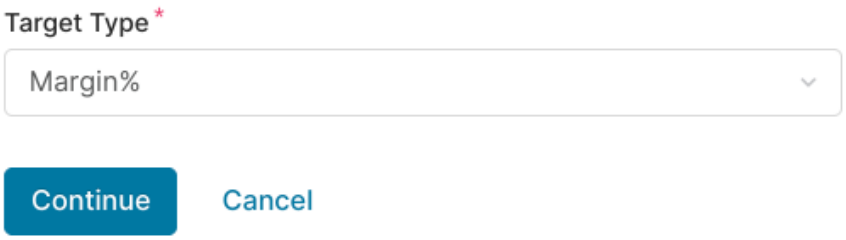
Please select...

List Price

Please select...

Optimization Target *

MarginPct



- Some rules apply in the mapping:
  - The numerical values must be extended.
  - The percentage values must be values between 0 and 1 (i.e. a margin rate is defined by margin/revenue).

5. Click **Continue** and wait until the deployment is complete.

Python Engine and Optimization Engine Setup

Python Engine and Optimization Engine are required to run some steps of the model. They are deployed and configured automatically with the accelerator. You may just need to wait a few minutes to get them enabled on your partition if that was not the case earlier on.

Also note, it is better to deploy the accelerator to a new partition than copy the logic from one partition to another, because then these engines will not be enabled and configured.

Technical User Reference (Optimization - Negotiation Guidance)

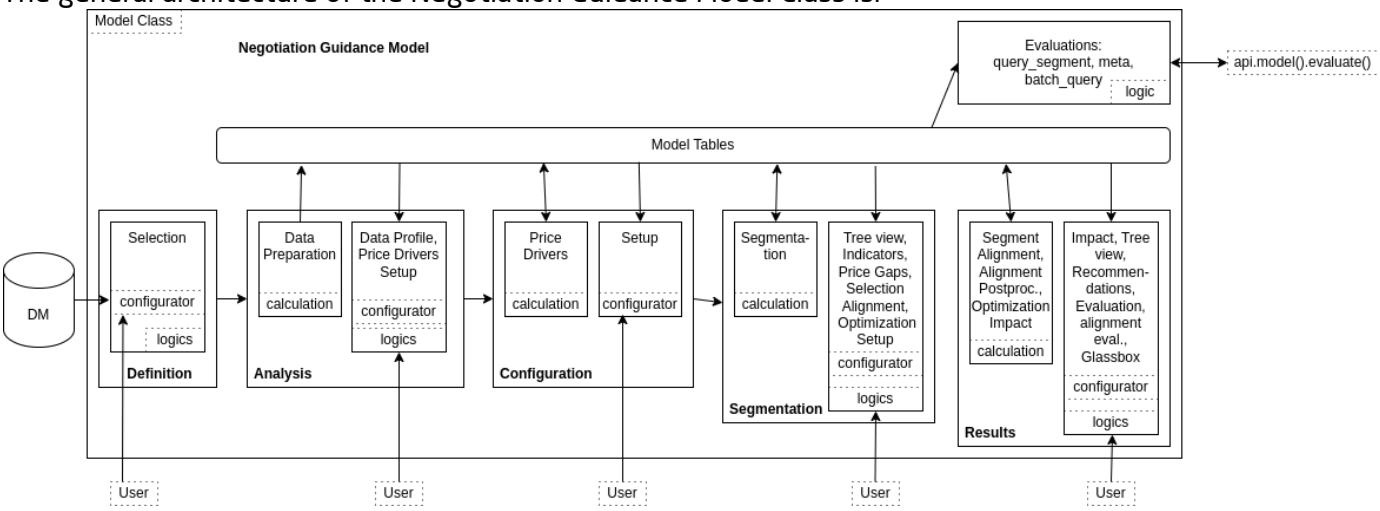
This section details the ModelClass and the logics that the Negotiation Guidance Accelerator deploys. For each step, its aim, outputs, and the main reasons to modify the logics are explained.

In this section:

Negotiation Guidance Model Class

The Negotiation Guidance V2 (NG2) ModelClass organizes a list of logics to create the model architecture. It is a JSON file that refers to some logics and it is transformed into an optimized UI in the Pricefx platform. See Model Classes for more information.

The general architecture of the Negotiation Guidance Model Class is:



It defines five steps:

- **Definition** - Sets the scope of the transactions.
- **Analysis** - Analyzes the data in the scope and set the price drivers parameters.
- **Configuration** - Sets the parameters for the segmentation.
- **Segmentation** - Checks the segmentation and sets the parameters for the optimization.
- **Results** - Looks at the outputs of the negotiation guidance in a user-friendly way.

There are two types of logics: `calculation`, which writes tables in the model, and `evaluation`, whose purpose is only to display some results. The standard Model Class definition is documented in Model Class (MC).

All the logics of the NG2 Accelerator follow a standard naming convention: first *NG2_* prefix, then the order and the first letters of the step name, then *Calc* or *Eval*, depending on the formula nature, then the name of the tab. An evaluation logic may also need a corresponding configurator to take user inputs; it is

defined in a logic of the same name, suffixed with *Configurator*. The logic to render a node from a segmentation tree is slightly different from the others, it is named *NG2_NodeEval_TreeView.* In the end, there is a library logic named *NG2_Lib*.

Library

The logic is **NG2_Lib**.

Aim of the logic

NG2_Lib is used in nearly all the other logics deployed by the Accelerator and defines a set of functions needed specifically for this Accelerator, but also some constants used to easily change the user interface wording. There are the following elements:

- `LabelsUtils`, `TablesUtils`, and `ParametersUtils` - Lists of the labels, names of the tables, and names of the parameters used in all the places in the code.
- `DefinitionUtils`, `AnalysisUtils`, `ConfigurationUtils`, and `SegmentationUtils` - Sets of tools dedicated to each step of the negotiation guidance.
- `AlignmentUtils` - Set of tools dedicated to the segment alignments
- `OptimizationEngineUtils` - Set of tools to describe automatically the optimization engine problem to optimize the alignments.
- `TreeViewUtils` - Tools to display nicely the different charts in the tree view
- `CalculationsUtils` - Groups the tools to deal with the percentiles and all the main formulas that could be useful to move between costs, margins, prices, and discounts.
- `ElasticityUtils` - Groups the methods to deal with different elasticity functions and evaluations.
- `BeneficMetricsUtils` - Set of functions to evaluate the projected values for a given change of the optimization target.
- `CustomUtils` - Defines the formulas to get the most important pricing values changes if the optimization target is of a given type.
- `ImpactUtils` - Set of functions used by the Impact tab of the Results step.
- `ColorUtils` - Groups the tools to generate colors to be used in charts.
- `SettingsUtils` – Function to get fields mapping set during the accelerator deployment.
- `ChartUtils` - Tools to create and display charts in different parts of the model.
- `RecommendationUtils` - Tools to generate automatically the recommendation portlets.
- `BatchQueryUtils` - Tools to build the complex SQL query that can navigate in an optimized way in the segmentation tree.
- `MathUtils` - Simple math functions that are not yet in the shared library.

It is accessed via the calls on `libs.NG2_Lib.XXX` in the code.

Common reasons to modify the logic

Changes of almost any wording in the user interface are done in `LabelsLib`.

Changes of the names of the Model Table or adding a new one are done in `TablesLib`.

You might want to define another optimization target (for instance: markup in place of margin). In this case, the functions to define the revenue depending on the optimization target will change (element `CustomLib`). The `CustomLib` must also be changed if the way to calculate a metric depending on the other ones change, for instance: we cannot consider that the cost is proportional to the quantity because there are some fixed costs.

You might want to define a new elasticity curve, and then the `ElasticityLib` would be modified.

Definition Step

There is no calculation logic in this step, and there is one tab with related evaluation logics: **NG2_1_Def_Eval_Selection** and **NG2_1_Def_Eval_Selection_Configurator**.

- Aim of the logics
  This logic provides the user inputs to define the source data and map it.
- Outputs of the evaluation
  A dashboard shows the transactions in the scope and the filtered-out transactions. It is an evaluation logic, so nothing is written in the model, but the user inputs defining the data mapping are available in the next steps.
- Common reasons to modify the logics
  - Some other mappings are needed or some would be retrieved.
  - Define pre-set filters.
  - Add a chart to better understand the data. (Caution: it could take long, as the data are not yet stored in the model.)

Analysis Step

The calculation logic is **NG2_2_Ana_Calc_DataPrep** and there are two tabs with related evaluation logics: **Data Profile** and **Price Drivers Setup**. In this step, the data are materialized inside the model, an analysis is displayed, and the user is able to select the price drivers to calculate.

Calculation: DataPrep

The logic is **NG2_2_Ana_Calc_DataPrep**.

- Aim of the logic
  This calculation mainly materializes the data in the scope of the negotiation guidance. It also creates a summary table that is used to display some analysis in the Data Profile dashboard.
- Outputs of the calculation
  - Data in the scope: *Transactions* Model Table
  - Summary data: *Profile* Model Table
  - An even more global summary is stored and accessible through `model.outputs('analysis', 'dataPrep')['Metrics']`
- Common reasons to modify the logic
  If there is something else to output when the materialization of data is done.

Data Profile Tab

The logic is **NG2_2_Ana_Eval_Profile**.

- Aim of the logic
  This is mainly a dashboard tab to summarize the data of the scope in a visual way.
- Outputs of the evaluation
  The outputs of the logic are the portlets: a bar chart for the scope summary, a view of the Profile DMT, and the distinct values of the selected field, if there is one.
- Common reasons to modify the logic

You can add or remove some summary portlets that would make sense, for instance, if there is a specific field that you need to visualize in a certain way.

Price Drivers Setup Tab

The logic is **NG2_2_Ana_Eval_Price_Drivers_Setup_Configurator**.

- Aim of the logic
  This logic is a pure configurator. Its aim is to choose the features that will participate in the price drivers' evaluation.
- Outputs of the evaluation
  The configurator defines an input matrix user entry.
- Common reasons to modify the logic
  Some price drivers are pre-selected, you can change the rule to pre-select these ones. You could also provide another field in the input matrix, or sort the attributes in a different way.

Configuration Step

The calculation logic is **NG2_3_Con_Calc_PriceDrivers** and there is one tab with related evaluation logic.

Calculation: PriceDrivers

The logic is **NG2_3_Con_Calc_PriceDrivers**.

- Aim of the logic
  This logic calculates the price drivers, i.e. the importance of each attribute when a boosted tree is built on top of them, using the optimization target as the variable to predict. It also computes the interactions among features, the hierarchies, and the segmentation dimensions recommendations.
- Outputs of the calculation
  The logic writes the tables:
  - *PriceDriversCorrelations*, a table containing the interaction data. It is used in the Feature Interaction Data and Feature Interaction portlets.
  - *PriceDriversHierarchies* and *PriceDriversHierarchiesEdges*; two tables containing the information to rebuild the hierarchy graph contained in the Hierarchy portlet.
  - *PriceDrivers*: The table contains the information required for the Price Drivers - Feature Importance portlet, Price Drivers - Relative Importance portlet, Price Driver Recommendations portlet, and the Select segmentation dimensions section of the configurator
  - *PriceDriverRecommendations*: A slightly more complete table containing information useful for debugging PriceDrivers.
- Common reasons to modify the logic
  The parameters to define the classification and boosting tree can be changed. It is also possible to change the way to calculate the attributes' importance. The correlation computation can also be changed. Finally, the segmentation dimension recommendation could be adjusted to other constraints of the domain.

Setup Tab

The logic are **NG2_3_Con_Eval_Setup** and **NG2_3_Con_Eval_Setup_Configurator**.

- Aim of the logic
  This tab is a configurator to define all the parameters of the segmentation itself: segmentation levels, parameters of the tree nodes, and elasticity parameters, and a dashboard to analyse the price drivers.

Outputs of the evaluation

All the user inputs of this configurator are available in the next steps, using the library element `ConfigurationLib`.

Common reasons to modify the logic

There could be different parameters to build the segmentation tree or to run the elasticity in the dedicated nodes.

Some charts would be removed, or modified.

Segmentation Step

The calculation logics are **NG2_4_Seg_Calc_Segmentation** and **NG2_4_Seg_Calc_PriceGaps**, and there are five tabs with related evaluation logics: **Tree View**, **Indicators**, **Price Gaps**, **Selection Alignments**, and **Optimization Setup**.

Calculation: Segmentation

The logic is **NG2_4_Seg_Calc_Segmentation**.

Aim of the logic

This calculation is a parallel calculation. That means that the first elements, of nature `calculation-init`, are global calculations. Then there are some elements of nature `calculation-item` that are run in parallel for each node of the segmentation tree: they calculate the metrics of the segment and its elasticity curve. Finally, some elements are of type `calculation-summary`. They update the tables of the model.

Outputs of the calculation

- The list of segments with their detailed attributes: *Segments* Model Table. This table contains in particular the elasticity curve parameters and the optimal optimization metric value to maximize the revenue or the profit, based on the elasticity fit. If the user had selected the option to calculate the metrics based on the elasticity, there are also the projected quantity, revenue, and profit based on these optimal values.
- The list of segments with their editable parameters: *Segments* Parameters Table.
- The decomposition of the optimization metric by percentile inside each segment: *Percentiles* Model Table.
- The evaluation of the fit to a normal distribution for each segment: *Fit* Model Table.
- The aggregation of several dimensions by segmentation level: *SegmentationOverview* Model Table.
- The *Segments* segmentation tree.

Common reasons to modify the logic

The segments' attributes could be added or modified. It could also be interesting to define other ways to evaluate if some segments are well-defined or not. For more details on parallel calculations see Build and Use Parallel Calculation.

It is the place where the elasticity curve is calculated, but the elasticity calculation refers to the library. If you want to change the elasticity function, you would have to update the code in the element `ElasticityLib` of the library NG2_Lib.

The benefit metrics are also calculated here. If there is a change in the way to calculate them, you would have to update the code in the element `BenefitMetricsLib` of the library NG2_Lib.

The `Fit` element calculates how much the segment distribution fits with a normal distribution. One could calculate another fit or another segment's goodness in general. In this case, the main formulas to change are in NG2_Lib, element `CustomLib`.

If there is a need to change the Parameters Table, i.e. the table that can be edited by the end-user to define specific parameters for some segments, then it is important to check also the element `SegmentationLib` of the library NG2_Lib, where some functions have been created to deal with this Parameters Table.

Calculation: Price Gaps

The logic is **NG2_4_Seg_Calc_PriceGaps**.

- Aim of the logic

  In the next step, the segments are aligned with each other. In order to be able to make the choice, the gaps between the different segments are calculated before the alignment setting.

  The logic calculates, for each segment, its price gap with its parent segment. Then, these values are aggregated by the level of segmentation and the values of each level of segmentation (refer https://pricefx.atlassian.net/wiki/spaces/ACC/pages/4828954970#Price-Gaps for more details)

- Outputs of the calculation

  A table called `DimensionValuesMetrics`, with one row per segmentation level value.

- Common reasons to modify the logic

  If there is another segment metric that is interesting to aggregate at each segmentation level.

Tree View Tab

The logic is **NG2_NodeEval_TreeView**.

- Aim of the logic

  This tab is of a special kind, it displays a tree. This tree is a comfortable way to navigate into the segments and get more information about any of them.

- Outputs of the evaluation

  Like any evaluation logic, there is no real output, except the fact that it displays information in the browser.

- Common reasons to modify the logic

  To display such a tree, you must define the tab type as `filtertree`. The model class defines the tree to fetch from the backend. This tree is called `Segments` and is created by the logic NG2_4_Seg_Calc_Segmentation.

  The logic itself refers to the selected segment, to display the right-side dashboard panel. See https://pricefx.atlassian.net/wiki/spaces/KB/pages/3862364240/Model+Class+MC#Filtertree-Tab-Fields.

  The main reason to change this logic is to add or remove an output for the segments.

Indicators Tab

The logic is **NG2_4_Seg_Eval_Indicators**.

- Aim of the logic

  This dashboard displays most of the data written by the logic NG2_4_Seg_Calc_Segmentation.

- Outputs of the evaluation

  Like any evaluation logic, there is no real output, it displays a dashboard.

- Common reasons to modify the logic
  You may want to display the main information about all the segments with different charts or data.

Price Gaps tab

The logics are **NG2_4_Seg_Eval_Price_Gaps_Review** and **NG2_4_Seg_Eval_Price_Gaps_Review_Configurator**.

- Aim of the logics
  The dashboard filters one segmentation level and displays this level's segments metrics.
- Outputs of the evaluation
  Like any evaluation logic, there is no real output, it displays a dashboard.
- Common reasons to modify the logic
  You may want to display other segments metrics.

Selection Alignments tab

The logic is **NG2_4_Seg_Eval_Selection_Alignments_Configurator**.

- Aim of the logic
  This configurator sets all the parameters to describe the structure of the optimization problem. It is a complex set of logics: sections are visible depending on previous checks, and when these features are unselected, the user inputs are kept in the model through hidden user input.
- Outputs of the evaluation
  The selected alignment dimensions and the gaps corresponding to the aligned values.
- Common reasons to modify the logic
  Be careful: if you want to change the inputs of the optimization problem, you should have to change the way the problem itself is described (refer to https://pricefx.atlassian.net/wiki/spaces/ACC/pages/4828955241#Calculation%3A-Segment-Alignment)

Optimization Setup Tab

The logic is **NG2_4_Seg_Eval_Optimization_Setup_Configurator**.

- Aim of the logic
  This configurator aims to define the user parameters for the optimization process inside each segment. Only the global parameters have input here, the ones specific to some segments are changed directly in the Parameters Table called Segments.
- Outputs of the evaluation
  The user inputs are available in the next steps.
- Common reasons to modify the logic
  If there are some other parameters to define for the optimization inside each segment.

Results Step

The calculation is the sequence composed of the **Segment Alignment**, the **Alignment Postprocessing**, the **Optimization,** and the **Impact** calculations logic, and there are four tabs with related evaluation logics: **Impact**, **Tree View**, **Recommendations**, and **Evaluation**.

Calculation: Segment Alignment

The logic is **NG2_5_Res_Calc_Alignment**.

Aim of the logic

The goal of this calculation is to create a simulation and an optimization run and retrieve their results. To do so, we need to create a Problem Description that details the structure of the problem to be solved by the Optimization Engine and gives endpoints for the OE to get the data of the problem. The inputs coming from the *Selection Alignment* tab will change the problem by altering its scope and changing the objectives, and the data will be fed directly to the OE thanks to the model tables.

This step consists of:

- **Data manipulation** to prepare the last tables needed by the OE. These logics are prefixed by *Store_* and create the model tables prefixed by *Problem_* that act as an endpoint for the OE. Be careful, *their names follow a strict format*. These endpoints must be named according to the `Problem_nameOfTheSpace_nameOfTheScope` present in the *ProblemDescription.groovy* and return the corresponding data. The behavior of the OE and its way of reading data from endpoints highlight the need for a well-thought-out Scope step. Creating tables of the needed data, already computed and aggregated, implies being well aware of "where is the data I need" and "how do I need to transform it". Due to computations depending on user inputs from business rules and objectives, some tables have to be created in later steps, explaining why even the RunOptimization calculation can have some *store* logics. That is why it is normal to refactor and improve scope logics and the other store logics during the development of *ProblemDescription.groovy*.
- **ProblemDescription.groovy** element - Returns the problem description in a map. The content of the problem description is detailed in Problem Description.
- **GlassboxConfig.groovy** element - Parses the problem description to automate the post-processing.
- **Run.groovy** element - Contains the code that handles the problem description. It takes the description of the problem and the advanced parameters user inputs, and triggers the two optimization jobs thanks to `model.startJobTriggerCalculation`. Each run will return prefixed tables of similar structures. The jobs will run in parallel. The first one is the optimization itself and its outputs are prefixed by "*Optimized*". The second one is a simulation: it simulates the first state of the optimization and will be a reference to compare before/after values in the results dashboards. Its outputs are prefixed by "*Current*". The job type (optimization vs. simulation) is indicated by the input parameters of the `model.startJobTriggerCalculation` function.

Once the problem description is created, it is sent via a Kafka message to trigger the instantiation of a job running an OE configured by this file. The OE has to have access to the correct endpoints to get the data and to know where to write back the results when the computation is finished.

Outputs of the calculation

The Groovy code does some preparation work. It creates `Problem_nameOfTheSpace_nameOfTheScope` tables -- data manipulation to prepare the last tables needed by the OE. These logics are prefixed by "Store_" and create the model tables that act as an endpoint for the OE. Be careful, *their names follow a strict format*.

A Groovy element also reads the problem description to retrieve a list of parameters used during the postprocessing step to reformat the Glassbox data.

At the end of its run, the OE will write a set of model tables containing its results and the Glassbox information needed to understand why this solution was used. This writing is done directly by the OE and is *not related to a Groovy logic*. This job is done by the two OE jobs, the simulation one and the optimization one.

- The Glassbox table provides optimization indicators for each pair of instantiated value finder - criterion. The simulation job does not create any Glassbox table.
- The tables prefixed by *Results_* present the state of the objectives and constraints at the end of the optimization.

- The tables prefixed by *Solution_* present the raw values that the system was *meant to find* (declared as Value_Finder in the Problem Description). The simulation job does not create any Solution table.
- The tables prefixed by *Simulation_* present the value of computed variables marked as exposed in the description, typically including *values of interest* such as forecasted quantities.

Common reasons to modify the logic

Any modification of the problem modeling and type of constraints to apply or objectives to reach implies a modification of the Problem Description, thus *ProblemDescription.groovy* should change accordingly.

⚠ If the problem description changes, do not forget to check if it is necessary to change or create some Problem tables, in the elements prefixed with *Store_*.

In some cases, it could be useful to change the OE image and/or the OE tag that the job trigger refers to. Their values are in the element *Run.groovy*.

Calculation: Alignment Postprocessing

The logic is **NG2_Res_Calc_Alignment_Postprocessing**.

Aim of the logic

The logics that triggers an Optimization Engine ends with this trigger. The postprocessing work must take place in the next logic. The postprocessing logic is in the same sequence and starts when the *Segment Alignment* one finishes.

This calculation retrieves the outputs of the RunOptimization logic and reformats them to provide tables that can be used to show user-friendly optimization results. Each element stores one model table or some similar tables.

- *Store_Glassbox_* logics calculate aggregated metrics on the optimization agents' criteria and value finders.
- *CreateAggregatedAndSolutionTable* loads a table of the pairs of product, customer in the scope, with the value of the optimization target before and after the alignment. It does not aim to be read by a human, it is used later to display the outputs of the optimization.
- *Store_SegmentAlignmentImpact* loads a table to help an advanced user to understand what as been done during the alignment process, or to debug in case of change. All the segmentation dimension values are aggregated, at all the levels of segmentation. When there is a corresponding segment, it is provided. When an alignment applies on the set of segmentation levels values, the requested alignment is provided. Finally, the values of the optimization target, before and after the alignment run, are provided.
- *Store_RecommendationTable* manipulates the data in order to prepare the display of the *Recommendation* portlets in the step Results.

Outputs of the calculation

This calculation writes a collection of tables:

- **GlassboxVF_FinestGranularity_*OptimizationTargetType*** table – It stores the overall values of each value finder.
- **GlassboxProbe_FinestGranularity_*OptimizationTargetType*** table – It stores the initial movement of each value finder.
- **GlassboxCriteria_** tables – Their names are built as *GlassboxCriteria_NameOfTheSpace_NameOfTheCriterion*, there is one table by alignment. These tables store the overall values of each criterion. The spaces are called "PairsFor*NameOfTheDimension*Alignment" and correspond to the pairs of attributes where alignment criteria are set.

- **Glassbox_AggregatedMetrics** table – Summarizes the global interaction indicators between each value finder key and each criterion key (alignment).
- **Glassbox_VFs_by_Key** table – Summarizes the global overall indicators of each value finder key.
- **Glassbox_Criteria_by_Key** table – Summarizes the global overall indicators of each criterion key.
- **Glassbox_Spaces** table – Summarizes the total number of criteria and value finders in each space.
- **SegmentAlignmentImpact** table - Summarizes the values before and after the optimization for each pair of product and customer.
- **SegmentAlignmentImpact** table - Summarizes the values by set of segmentation levels' values.
- **Recommendation** table - Summarizes the alignment values by single segmentation level's values.

Common reasons to modify the logic

The most common reason to change the logic is to reformat some data to ease the work of providing charts in the Result step tabs.

Calculation: Optimization

The logic is **NG2_5_Seg_Calc_Optimization**.

Aim of the logic

In this calculation, the parameters are used to calculate for each segment: a score, a target percentile (if not provided by the user). Then the alignment results (if done, otherwise the source data values) are used to define the floor, target, and ceiling thresholds of each segment (more details in Usage (Optimization - Negotiation Guidance)).

Outputs of the calculation

The *Segments* Model Table and the *Segments* Parameters Table are updated with new columns corresponding to the optimization strategy.

Common reasons to modify the logic

You might want to define the optimization process differently than explained in Usage (Optimization - Negotiation Guidance).

Calculation: Impact

The logic is **NG2_5_Res_Calc_Impact**.

Aim of the logic

This calculation uses the segmentation done to calculate the impact values that are used to display the Impact dashboard. These values mock the use of the optimization strategy over each segment on the historical data in the scope.

Outputs of the calculation

The *Impact* Model table is defined. It is an aggregation of the historical transactions in the scope using a combination of all the segmentation levels, with the historical and the projected values using the optimization strategy. This table is used by the *Impact tab* dashboard.

This logic uses the model evaluation `batch_evaluation` to get the strategy coefficient to use for any transaction.

Common reasons to modify the logic

If the Impact tab would refer to different optimization calculations.

Impact Tab

The logics are **NG2_5_Res_Eval_Impact** and **NG2_5_Res_Eval_Impact_Configurator**.

Aim of the logics

This dashboard provides the results of the optimization based on the thresholds in some meaningful portlets. The user inputs allow one to choose the level of aggregation and the realization rate. It uses the Model Table created by the previous calculation, called *Impact*.

- Outputs of the evaluation
  Like any evaluation logic, there is no real output, it displays a dashboard.
- Common reasons to modify the logics
  To display other charts or to provide meaningful information in a different way.

Tree View Tab

The logic is **NG2_NodeEval_TreeView**. It is exactly the one used in https://pricefx.atlassian.net/wiki/spaces/ACC/pages/4828955241#Tree-View-Tab, please refer to this section. It uses the same data but also some new fields, calculated by the NG2_5_Seg_Calc_Optimization calculation.

Recommendations Tab

The logic is **NG2_5_Res_Eval_Recommendations**.

- Aim of the logic
  This dashboard returns exactly the Model Table *Segments*, with all the optimized metrics.

  If some alignments are set, then a portlet is created for each alignment. Their values relie on the *Recommendation* table (refer to https://pricefx.atlassian.net/wiki/spaces/ACC/pages/4828955241#Calculation%3A-Alignment-Postprocessing)
- Outputs of the evaluation
  Like any evaluation logic, there is no real output, it displays a dashboard.
- Common reasons to modify the logic
  It is possible to choose not to display some of the fields of the Model Table, or to add some fields with values calculated from the existing ones. It is also possible to display another table in this dashboard, like the *Impact* one, which contains the optimized values based on the thresholds policy, while the *Segments* one contains the optimized values based on the elasticity curve fit.

Evaluation Tab

The logic is **NG2_5_Res_Eval_Evaluation**.

- Aim of the logic
  This tab simulates the `query_segment` evaluation of the model from any place in the Pricefx partition. It is of the type `simple`. It takes the parameters of the evaluation function as user inputs and displays all the outputs of it in a table.
- Outputs of the evaluation
  The displayed table represents the output map of the evaluation function. This function can be called from anywhere in the partition, using the command:

```
def model = api.model("TheModelUniqueName")
def results = model.evaluate(
  “query_segment”,
  [
    segmentationLevel1: "someValue", // keys depend on the
segmentation
    //...
```

```
    ]).results
  def target = results['Target'] // or any other output
```

For details see https://pricefx.atlassian.net/wiki/spaces/KB/pages/3851026646/Query+Optimization+Engine+Results#Using-the-Evaluator.

- Common reasons to modify the logic
  A model evaluation is the main link between any place in the partition and the model. So any output that should be provided to another module must be in this logic.

Alignment Review Tab

The logics are **NG2_5_Res_Eval_Alignment** and **NG2_5_Res_Eval_Alignment_Configurator**.

- Aim of the logic
  The dashboard displays as many boxplots as requested alignments, plus a chart whose scope is parametrized by the user entries selected through the embedded configurator.
- Outputs of the evaluation
  Like any evaluation logic, there is no real output, it displays a dashboard.
- Common reasons to modify the logic
  It is possible to change the kind of charts that are displayed.

Glassbox Tab

The logic is **NG2_5_Res_Eval_Alg_Glassbox**.

- Aim of the logic
  This tab exposes the technical state of the OE execution at the end of the process. It is a development tool to help tune the model.
- Outputs of the evaluation
  This logic displays charts that show the satisfaction, influences, impacts of the value finders and the criteria, initial movements of the value finders, and evolution of the criticality during the process of optimization. For details see Results Description - Glassbox tab.
- Common reasons to modify the logic
  In general, there is no reason to change this dashboard.

Evaluations

The model has three evaluations: `meta`, `query_segment`, and `batch_query`. For more details about model evaluations see https://pricefx.atlassian.net/wiki/spaces/KB/pages/3851026646/Query+Optimization+Engine+Results#Using-the-Evaluator.

Meta Evaluation

The logic is **NG2_5_Res_Eval_Meta**.

- Aim of the logic
  It is a way for the models to communicate outside of themselves some basic information about their state:

  - Is the segmentation available or not? (Has the model run its segmentation?)
  - Is the elasticity available or not?

- What are the segmentation levels of the model?

Usage and outputs

There is no input for this evaluation. The usage is:

```
api.model("myNegotionGuidanceModel").evaluate("meta", [:])
```

The empty map is necessary, as the `evaluate` method expects a parameters map.

The output is a map of:

- `SegmentationAvailable`: boolean value,
- `SegmentationLevels`: list of maps representing the corresponding table fields,
- `ElasticityAvailable`: boolean value.

Common reasons to modify the logic

Add information that would be used from another part of the Pricefx solution.

Query Segment Evaluation

This is based on the logic **NG2_5_Res_Eval_Evaluation** and is detailed in the paragraph https://pricefx.atlassian.net/wiki/spaces/ACC/pages/4828955241#Evaluation-Tab.

Batch Evaluation

The logic is **NG2_5_Res_Eval_Batch**.

Aim of the logic

It is a way to get a large amount of optimization results in one command. This model evaluation can be called from any place in Pricefx, based on an existing model that has run.

Usage of the evaluation

The inputs are:

- Query that fetches data to evaluate from a Negotiation Guidance point of view.
- Some mapping information to detect which fields are the quantity, the elasticity based-on variable, the segmentation levels, and the fields to output. *The order of the segmentation levels is important*. The required keys of the mapping map are `quantity`, `optimization_target`, `segmentationLevels`, `otherFields`.

The output is an SQL query that can be executed to get, for each row of the executed input query:

- All the fields defined in the mapping argument.
- `name` - The name of the associated segment.
- Some NG results relative to the segment, corresponding to the elasticity and the optimization thresholds: `ElasticityParameters`, `elasticity`, `normalization_coefficient`, `target_avg`, `weighted_target_avg`, `target_std`, `weighted_target_std`, `floor_value`, `target_value`, `ceiling_value` (if the names have not been changed in NG2_Lib.ParametersLib).

The usage is:

```
def dmCtx = api.getDatamartContext()
DatamartContext.Query myQuery = dmCtx.newQuery(dmCtx.
```

```
getFieldCollection("DM.myDM"))
        .selectAll(true)
        .where(Filter.equal("Country", "France"))
DatamartContext.SqlQuery batchSqlQuery = api.model
("myNegotionGuidanceModel").evaluate(
       "batch_query",
        [
                sourceQuery: myQuery,
                mapping: [
                        quantity: 'myQuantityField',
                        optimization_target: 'marginPct',
                        segmentationLevels: ['productFamily',
'channel', 'country'],
                        otherFields: ['product', 'customer_id'],
                ]
        ]
)['AsBatchQuery']
api.getDatamartContext().executeSqlQuery(batchSqlQuery)
```

Common reasons to modify the logic

A model evaluation is the main link between any place in the partition and the model. So any output that should be provided to another module must be in this logic.

This logic provides a complex SQL query that fetches a lot of information by navigating into the segmentation tree. Some preprocessing or post-processing could be added in the query, depending on specific needs.

The logic is based on some library tools accessible from **NG2_Lib/BatchQueryUtils**. These tools are useful to fetch the segmented data in other parts of the model, inside the model itself.

Release Notes (Optimization - Negotiation Guidance)

Optimization - Negotiation Guidance 2.0.1

This document summarizes fixes introduced in the Accelerate Negotiation Guidance Package release version.

Version	2.0.1
Release Date	Dec 11, 2023

New Features and Improvements

New Feature Description	ID
It is possible to switch off the mode in which all variables are set as exposed: there is a new AdvancedParameterTable in parameter tables where you set 'OE_exposed_variables' to 'false'.	PFPCS-7629
When installing the accelerator from PlatformManager, engines configuration steps are now the first steps of the accelerator installation. This to prevent errors when trying to run a model while the accelerator installation has not been completed.	PFPCS-7700

Fixed Issues

Bug Description	ID
The query_segment evaluation does not handle well the case when no segment is found.	PFPCS-7724
When using Greenplum database and running the model, there is an error "type "jsonpath" does not exist".	PFPCS-7733