Accelerate Negotiation Guidance

Version 1.1

February 2023

Accelerate Negotiation Guidance (NG)

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

In this section:

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

Overview (NG)

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 sold the same product to the same customer before, you would usually not want to go too far from the previous price. But the previous price could sometimes be sub-optimal, either accidentally or even intentionally because of some special circumstances. You would not want to blindly follow this past price in the future. 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 and avoid sticking with the same (possibly not ideal) pricing that was used in the past.

Pricefx Solution

In our Negotiation Guidance model, we build a segmentation tree using a pre-defined set of attributes called **segmentation levels** and then provide price recommendations based on the information from the whole segment. The segmentation levels are usually a combination of product attributes, customer attributes, and even 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. The segmentation levels are usually chosen manually to reflect both the past behavior and business experience and needs of the end user. Ideally, the segments should be consistent in the sense that most of the transactions within a segment should have similar pricing.

The goal of our price optimization is then to estimate the pricing potential of each segment and push the prices to higher values wherever possible.

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

Business User Reference (NG)

- Usage (NG)

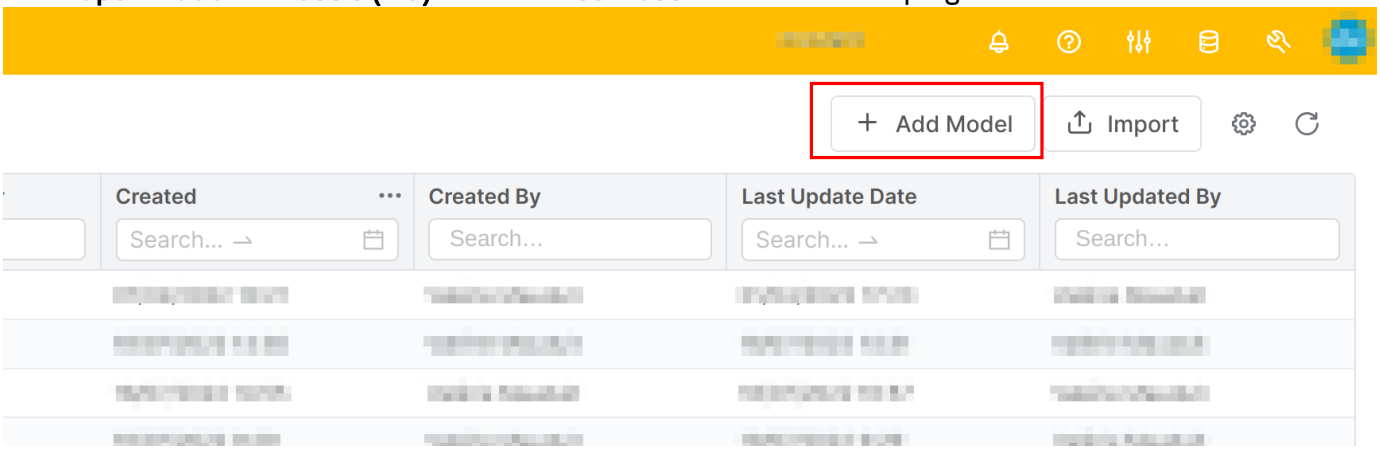
Usage (NG)

Take the following steps to configure a model:

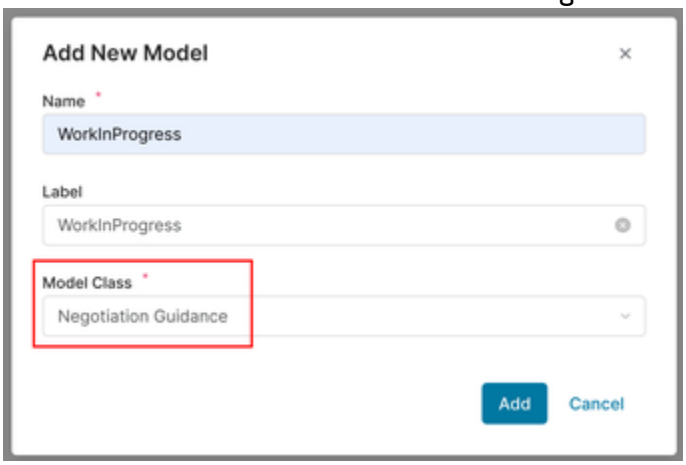
- 1. Create a Model Based on Negotiation Guidance Model Class
- 2. Set the Scope of Transactions (Definition Step)
- 3. Analyze Data in the Scope and Set Price Drivers Parameters (Analysis Step)
- 4. Configure the Segmentation (Configuration Step)
- 5. Set up Optimization Parameters (Segmentation Step)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

1. Create a Model Based on Negotiation Guidance Model Class

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

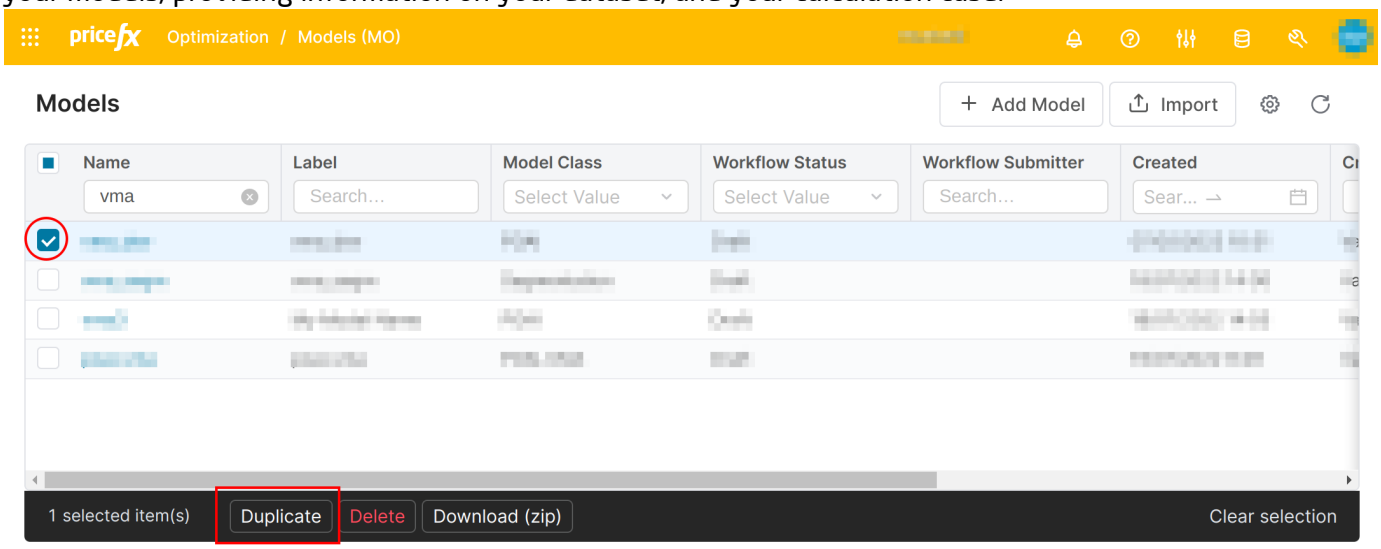


A pop-up is shown where you provide a name for your model, and the Model Class, which is *Negotiation Guidance*. Another Model Class would belong to another kind of optimization model.



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.



2. Set the Scope of Transactions (Definition Step)

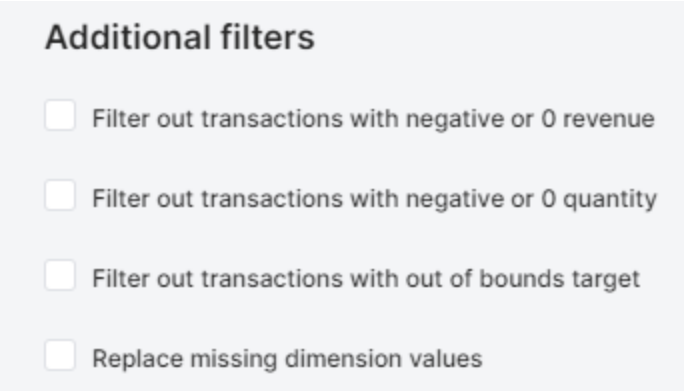
In the Definition step, you map the inputs and set the scope of the model. The user inputs are always on the left.

- **Transaction source** - Datamart used to calculate the segments. It must fill the requirements listed in Ins tallation (NG). 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 accordingly to the policy you want to simulate.
  - **Optimization Target** - Usually the margin rate or the discount rate.
  - **Target Type** - Can be "Margin%" or "Discount%".
  - **Weight Measure** *(optional)* - Field that indicates the weight of a transaction row. If not set, all the DM 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.
  - **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.

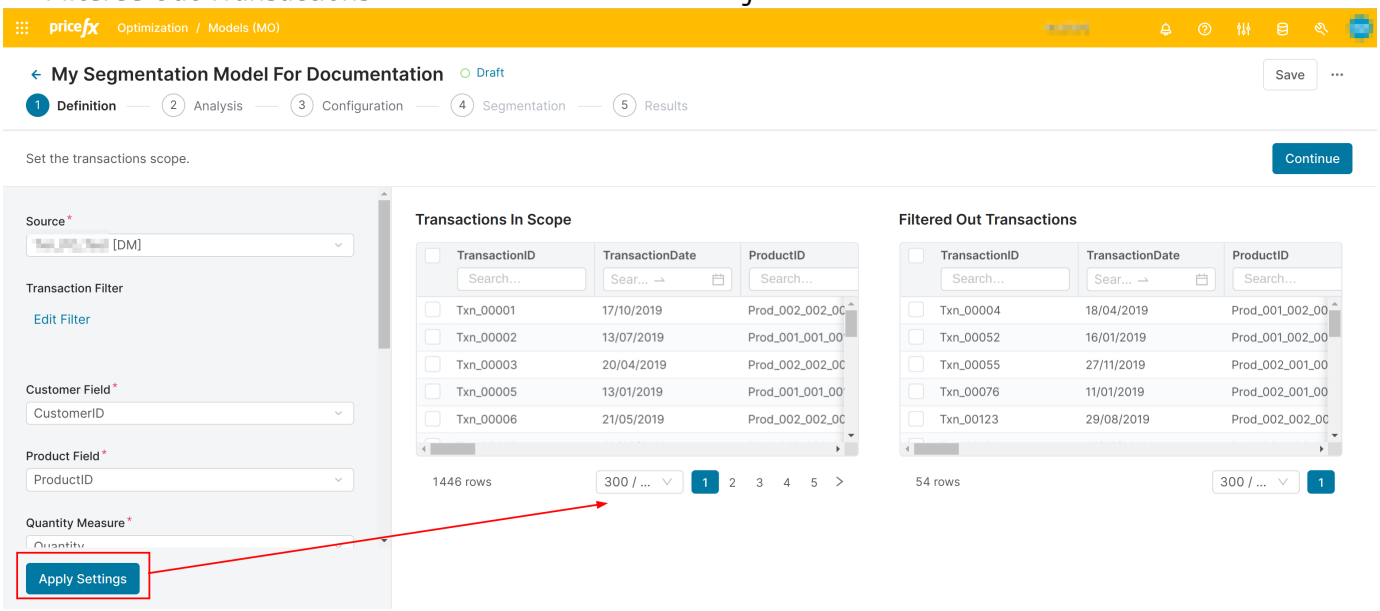
If needed, create some calculated fields in the transaction Datamart: revenue, margin, margin rate (i.e., margin/revenue).

Use the check-boxes for additional filters ensuring calculations without exceptions:



Once you apply the settings, the right panel provides:

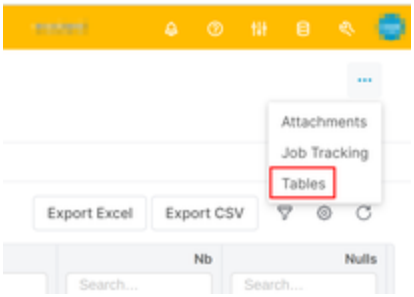
- **Transactions in Scope** - Data that are in the scope of the segmentation.
- **Filtered Out Transactions** - Data that are filtered out by the set transactions filter.



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)

When you arrive at the Analysis step from the Definition step, the model first runs a calculation to prepare the data. It can take some minutes, depending on the size of the source Datamart. The goal of this calculation is to format and save the data that will be used in the next steps of the model. 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 source Datamart, taking into account the filtered data in the scope. For any data, you get the min and max value, the number of nulls, the number of different values (cardinality), and information about the type of the field. 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

In this tab, you choose the fields on which the price drivers are calculated. The price drivers evaluate the importance of any dimension of the data to forecast the optimization target (i.e. generally the margin percentage). These values 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 dimensions 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.

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.

4. Configure the Segmentation (Configuration Step)

When you arrive at the Configuration step from the Analysis step, the model first runs a calculation to get the importance of the price drivers. In this step, you are now able to set the parameters of the segmentation itself.

The first section, **Select and sort segmentation dimensions**, is a table of the dimensions available for the segmentation. The importance columns display the result of the price drivers calculation. If a dimension was not selected in the price drivers selection, its importance is 0. The higher the importance, the more the dimension is accurate to provide a good segmentation. But due to business concerns, you may want to use other dimensions and it is possible. For each dimension, its cardinality is also provided. Check all the fields that you want to use for the segmentation. You can also drag and drop the lines of the input matrix to define the order of the segmentation levels. You *must* select at least one level of segmentation. Note also that dimensions with null values cannot be selected as segmentation levels (to avoid issues at later stages).

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

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 calculated quantity.

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

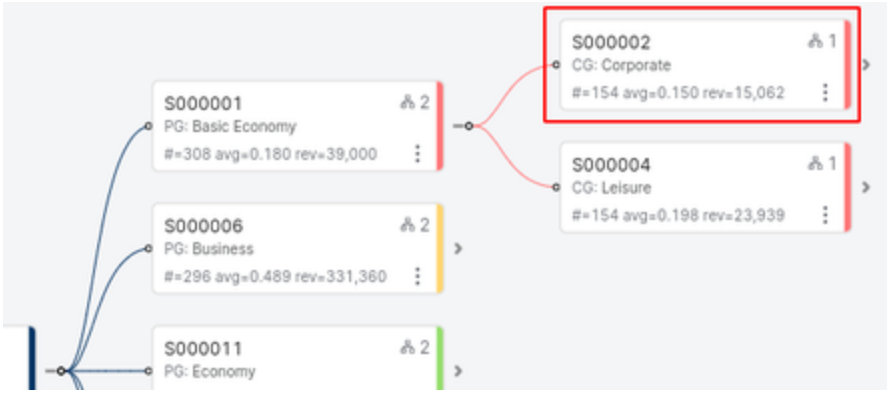
Click the **Continue** button (top right) to go to the Segmentation step.

5. Set up Optimization Parameters (Segmentation Step)

When you arrive at the Segmentation step from the Configuration step, the model first runs a calculation to define the segmentation tree and all the segments. It can take some minutes, depending on the size of the source data. This step is made of three tabs: *Tree View*, *Indicators*, and *Optimization Setup*.

Tree View

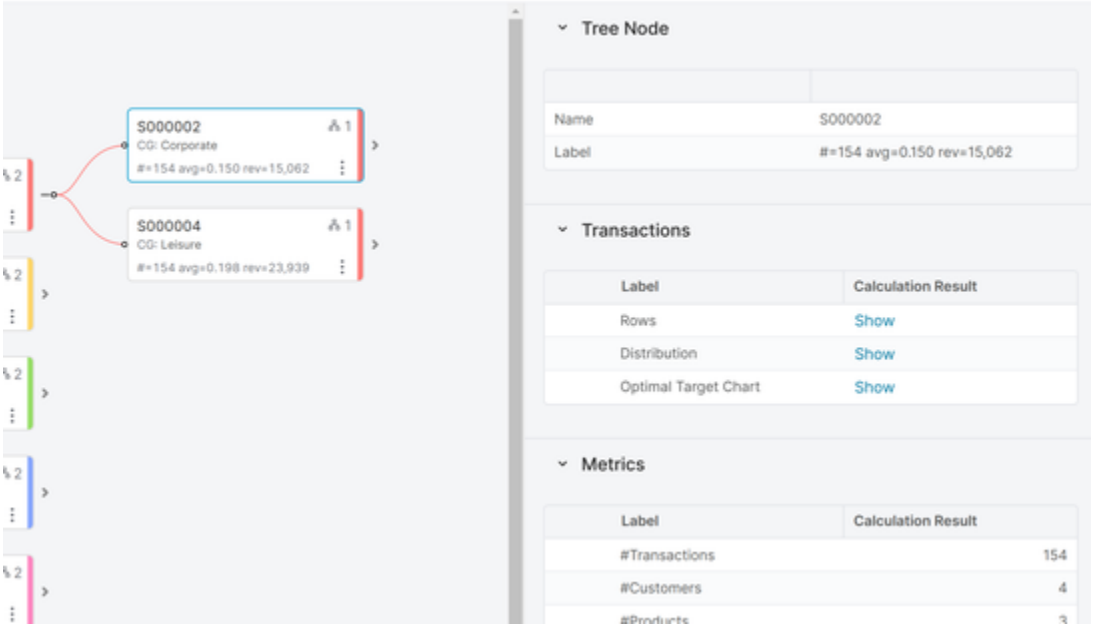
This is a dynamic view of the segmentation tree. You can expand and collapse it and get information about any segment.



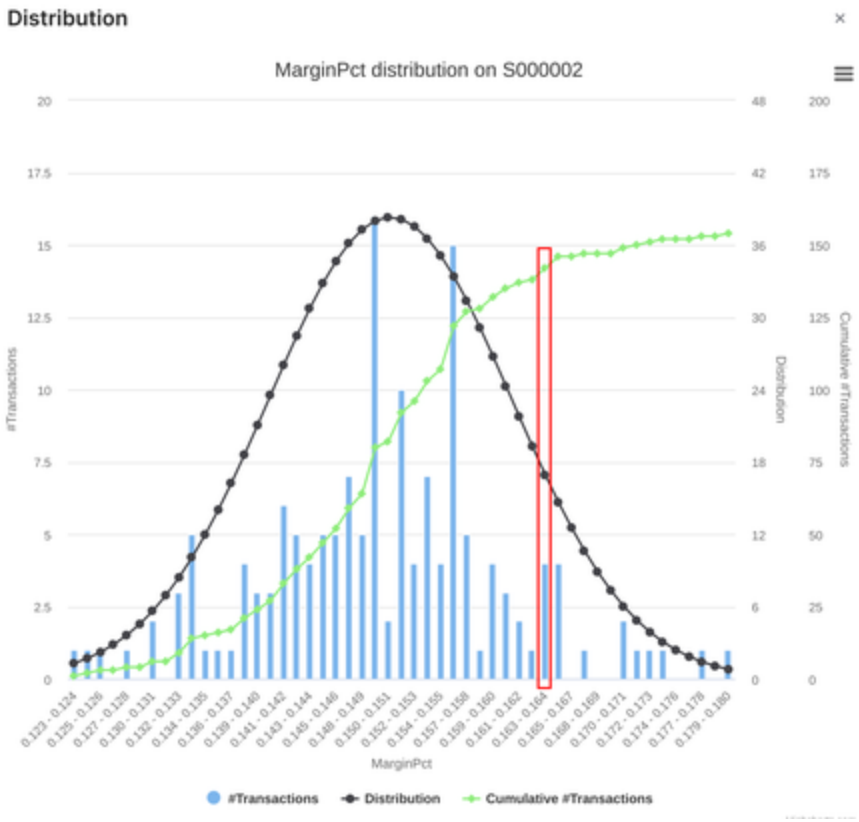
For each segment, i.e. node of the tree, the following information is directly given:

- 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 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.

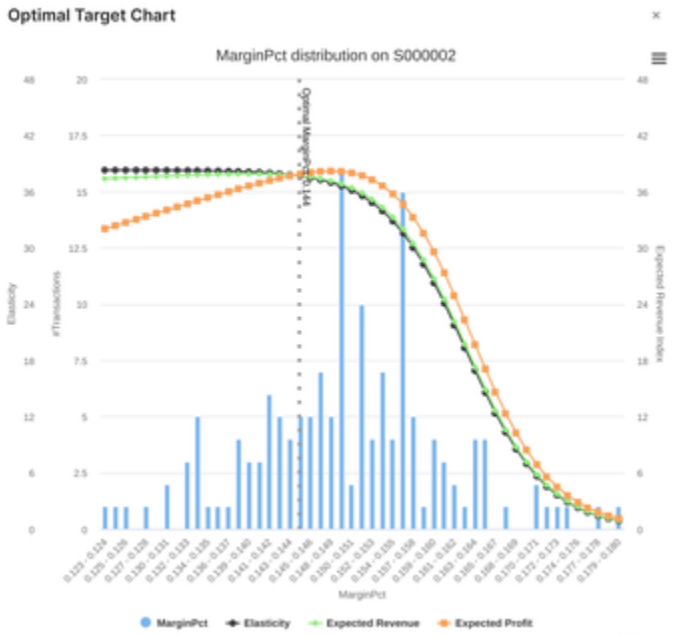


In addition to the information you already have in the tree view, you have some other metrics of the segment. In the segment Transactions, you also have access to the sample of input data present in this segment, and two histograms. The first one is called *Distribution*.



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. Here, 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. Here, the average value is 15% and the normal curve does not fit well on the segment.

The second histogram is called *Optimal Target Chart*.



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. Read below for more details.

Indicators

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

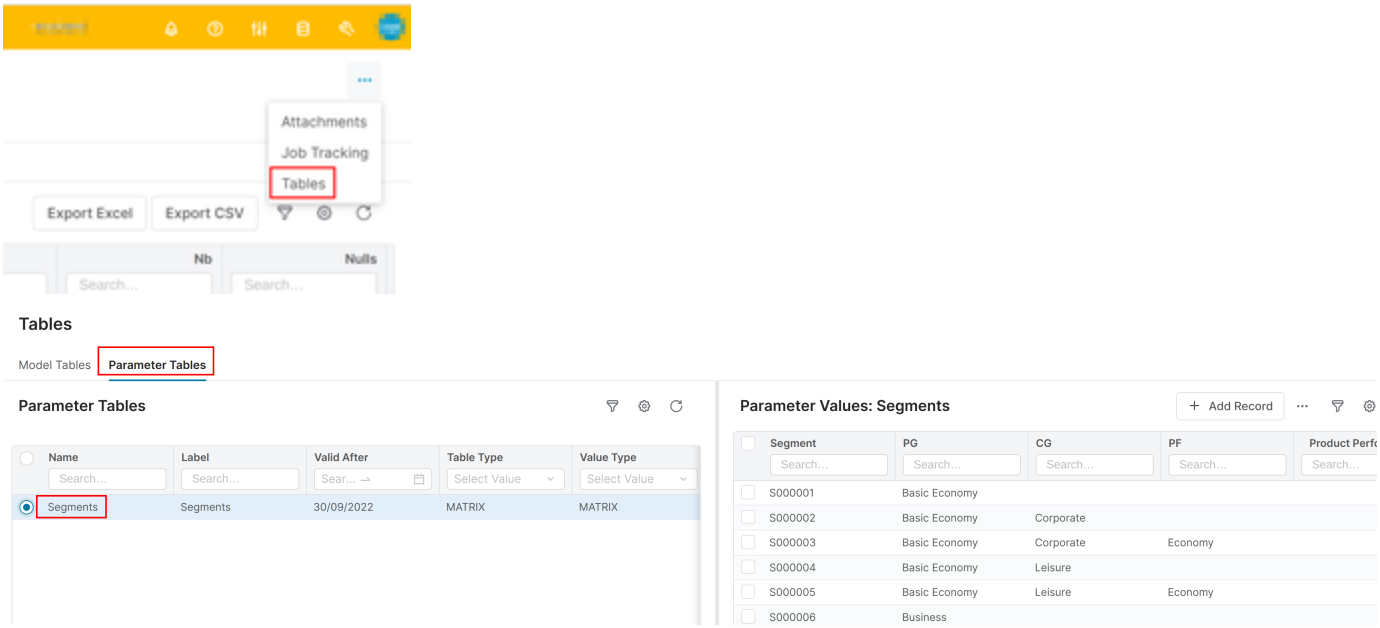
- **Details by segment** - Table that summarizes the metrics of all the segments in one place.
- **Segmentation overview** - Table that shows what amount of data is kept by each level of segmentation and how much this level of segmentation describes the target metric variations.
- **Fitting of Elasticity** - Provides the average divergence to normal distribution. Each segment is fitted to a normal distribution and a metric calculates the divergence between the values in the segment and the closest normal distribution. The value provided here is the average of all the segments' divergences. The higher this value, the less the segments fit normal distributions.
- **Elasticity fit chart** - Scatter chart where 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** - Table that provides the divergence to normal distribution (numerical value) for all the segments.

Optimization Setup

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

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 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.

How the Percentile Values Are Used?

Each segment is defined with three percentiles. The corresponding values of the target metric for these percentiles are calculated. Then 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 to the floor percentile value.
- If a transaction was done with an optimization metric between the floor percentile value and the target percentile one, it is increased to the target percentile value.
- If a transaction was done with an optimization metric higher than the target percentile value, it is left as it is.

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.

6. Manage the Segmentation Results (Result step)

When you arrive at the Results step from the Segmentation step, the model first runs the optimization calculation of each segment, it may take some minutes, depending on the size of the model. Then four tabs are provided: *Impact*, *Tree View*, *Recommendations* and *Evaluation*.

Impact

This dashboard displays the optimization result for the strategy floor-target-ceiling. In the case that we consider an optimization metric to increase, like a margin, all the transactions whose optimization metric is lower than the floor value are pushed to the floor value, and all the transactions whose value is between floor and target are pushed to the target value. If the optimization metric has to decrease, the corresponding behavior in a decreasing direction happens, with a decrease to the ceiling or to the target value. The left panel is to choose the granularity of the bar charts.

Tree View

This view is the same as https://pricefx.atlassian.net/wiki/spaces/ACCDEV/pages/4163239969#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.

Recommendations

This tab 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 transactions 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/ACCDEV/pages/4163239969#Percentiles.
- **Floor Percentile**, **Target Percentile**, **Ceiling Percentile** - Explained in https://pricefx.atlassian.net/wiki/spaces/ACCDEV/pages/4163239969#Percentiles.
- **Floor**, **Target**, **Ceiling** - Values of the optimization metric for each of the defined percentiles.
- **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.

Evaluation

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

From Another Module

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).

Admin User Reference (NG)

- Installation (NG)

Installation (NG)

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*. 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.
2. Go to **Marketplace** > **Accelerator Packages**.
3. Find **Optimization - Negotiation Guidance**.
4. Select your Target Partition from the drop-down menu.
5. Set up Datamart mapping.
   - 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).
6. Click **Continue** and wait until the deployment is complete.

Technical User Reference (NG)

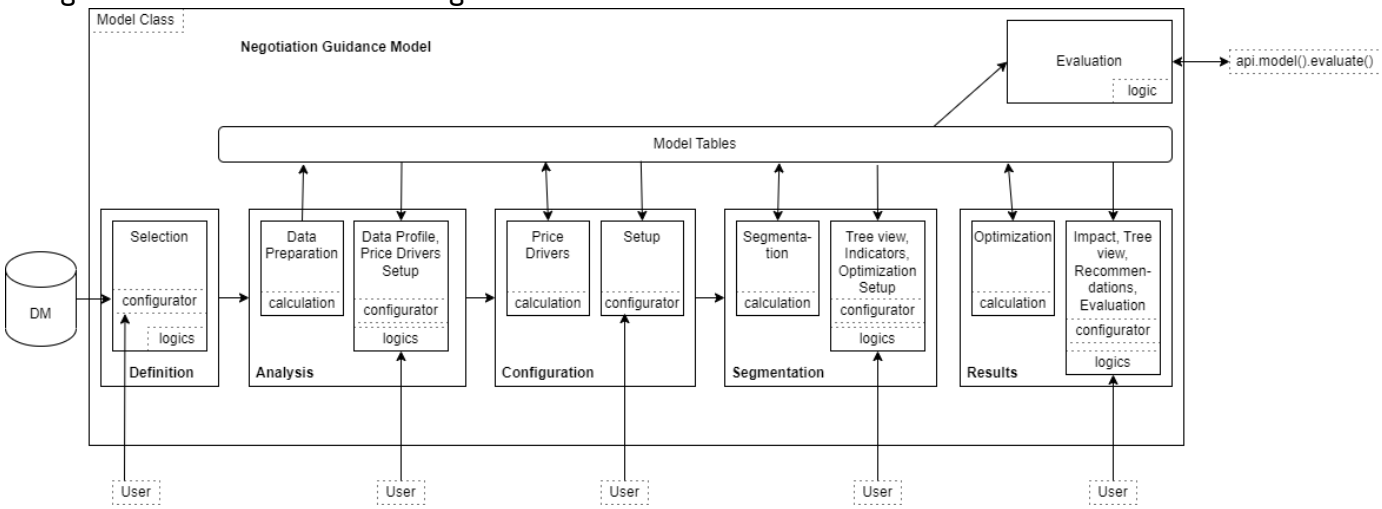
This section details the ModelClass and the logics that the Negotiation Guidance Accelerator deploys. For each step, its aim, outputs, and the main reasons to modify the logics are explained.

In this section:

Negotiation Guidance Model Class

The Negotiation Guidance (NG) 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 https://pricefx.atlassian.net/wiki/spaces/UNITY/pages/4122084177 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 NG Accelerator follow a standard naming convention: first *NG_* 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 *NG_NodeEval_TreeView.* In the end, there is a library logic named *NG_Lib*.

Library

The logic is **NG_Lib**.

Aim of the logic

Ng_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:

- `LabelsLib`, `TablesLib`, and `ParametersLib` - Lists of the labels, names of the tables, and names of the parameters used in all the places in the code.

- `DefinitionLib`, `AnalysisLib`, `ConfigurationLib`, and `SegmentationLib` - Sets of tools dedicated to each step of the negotiation guidance.
- `CalculationsLib` - Groups tools to deal with the percentiles and all the main formulas that could be useful to move between costs, margins, prices, and discounts.
- `ElasticityLib` - Groups the methods to deal with different elasticity functions and evaluations.
- `BeneficMetricsLib` - Set of functions to evaluate the projected values for a given change of the optimization target.
- `CustomLib` - Defines the formulas to get the most important pricing values changes if the optimization target is of a given type.
- `ImpactLib` - Set of functions used by the tab Impact of the step Results.
- `ErrorLib` - Deals with a calculation failure.

It is accessed via the calls on `libs.NG_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: **NG_1_Def_Eval_Selection** and **NG_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 **NG_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 **NG_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 **NG_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 **NG_2_Ana_Eval_Price_Drivers_Setup_Configurator**.

Aim of the logic

This logic is a pure configurator. Its aim is to choose the attributes 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 **NG_3_Con_Calc_PriceDrivers** and there is one tab with related evaluation logic.

Calculation: PriceDrivers

The logic is **NG_3_Con_Calc_PriceDrivers**.

Aim of the logic

This logic calculates the price drivers, i.e. the importance of each attribute when a regression tree is built on top of them, using the optimization target as the variable to predict.

Outputs of the calculation

The logic writes the table PriceDrivers. The table provides the importance of each simulated attribute.

Common reasons to modify the logic

The parameters to define the classification and regression tree can be changed. It is also possible to change the way to calculate the attributes' importance.

Setup Tab

The logic is **NG_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.

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.

Segmentation Step

The calculation logic is **NG_4_Seg_Calc_Segmentation** and there are three tabs with related evaluation logics: **Tree View**, **Indicators**, and **Optimization Setup**.

Calculation: Segmentation

The logic is **NG_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 NG_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 NG_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 NG_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 NG_Lib, where some functions have been created to deal with this Parameters Table.

Tree View Tab

The logic is **NG_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 NG_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 **NG_4_Seg_Eval_Indicators**.

Aim of the logic

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

Optimization Setup Tab

The logic is **NG_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 (percentile values) 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 logic is **NG_5_Res_Calc_Optimization** and there are four tabs with related evaluation logics: **Impact**, **Tree View**, **Recommendations**, and **Evaluation**.

Calculation: Optimization

The logic is **NG_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), and an optimized optimization metric value based on the percentiles (more details in Usage (NG)). Then the optimized value is used to calculate the global impact in the segment if the optimization strategy based on these thresholds would have been applied to the input transaction dataset.

Outputs of the calculation

The *Impact* Model Table, containing the global revenue and margin difference if the optimization policy based on thresholds would have been applied.

Common reasons to modify the logic

You might want to define the optimization process differently than the rule explained in Usage (NG).

Impact Tab

The logics are **NG_5_Res_Eval_Impact** and **NG_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 input allows one to choose the level of aggregation. 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 **NG_NodeEval_TreeView**. It is exactly the one used in https://pricefx.atlassian.net/wiki/spaces/ACCDEV/pages/4266000499#Tree-View-Tab, please refer to this section. It uses the same data but also some new fields, calculated by the NG_5_Seg_Calc_Optimization calculation.

Recommendations Tab

The logic is **NG_5_Res_Eval_Recommendations**.

Aim of the logic

This dashboard returns exactly the Model Table *Segments*, with all the optimized metrics.

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 **NG_5_Res_Eval_Evaluation**.

Aim of the logic

This tab simulates the 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.

Evaluations

The model has two evaluations: `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.

Query Segment Evaluation

This is based on the logic **NG_5_Res_Eval_Evaluation** and is detailed in the paragraph https://pricefx.atlassian.net/wiki/spaces/ACCDEV/pages/4266000499/Technical+User+Reference+NG#Evaluation-Tab.

Batch Evaluation

The logic is **NG_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 NG_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.

Release Notes (NG)

Negotiation Guidance 1.1

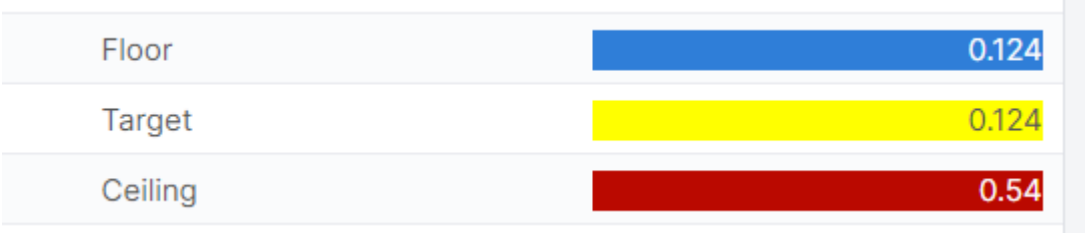
his document summarizes major improvements and fixes introduced in the Accelerate Negotiation Guidance Package release version.

Version	1.1
Release Date	Feb 16, 2023

Table of contents:

New Features and Improvements

Description	ID
Handling null values in Weight Measure has been improved: • Improved error message for null total weight for a segment. • Added option to filter out null weight if a weight column is selected in the definition inputs. • Added more checks against degenerate cases in Fit when weight is null.	PFUN-14727
Elasticity and optimal price calculations will not produce an error, even if the data in the segment will be out of validity range for those calculations. This can occur e.g. if the customer would like to use target metric of unsupported type (currently supported are Margin% and Discount%), and is not interested in the elasticity-based calculations. Then target metric can attain values out of validity ranges of elasticity calculations (for instance Margin% should be between 0 and 1). The elasticity-based calculations would not produce an error and result in most relevant number even in such cases. Break-even calculation will not produce a run-time exception even if the input data would lead to an arithmetic error in break-even calculations.	PFUN-15990
It is easier now to filter out transactions that create issues during the negotiation guidance process. In a new section "Data handling" of the Definition tab there are the following options: • Filter out transactions with negative or 0 revenue • Filter out transactions with negative or 0 quantity • Filter out transactions with out of bounds target	PFUN-16869

• Define min/mas target values • Replace missing dimension values with a defined text	
Elasticities and optimization results of a large set of data are evaluated now in a single batch query.	PFUN-17156
In the Results step in the Tree View tab, the fields with the optimization value for Floor /Target/Ceiling in the segment detail have now a background color to help users check the status quickly. 	PFUN-17758

Fixed Issues

Description	ID
Floor Percentile default value is set to 30 instead of 50 in case of Target Type Margin %.	PFUN-16642
Ceiling Percentile default value is set to 70 instead of 50 in case of Target Type Discount %.	PFUN-16643
Result step fails with the error "Query timed out".	PFUN-17166
Segmentation fails with the error "Division by zero".	PFUN-17385
When using a weight for a model, it is incorrectly applied to sample_count (labelled #Transactions) field from Segments table.	PFUN-17440
Segmentation fails with the error "Ambiguous method overloading for method java. lang.Long#minus".	PFUN-17474
When we have dimensions with null values and they are selected as segmentation levels, NG creates duplicated segments.	PFUN-17641
Segmentation fails with unhelpful error message "CreateSegmentsTable : ERROR (@38): Cannot access tail() for an empty iterable" when all data are filtered out at the segment table creation.	PFUN-17706
When a column selected for quantity in the transactions table contains floating values (and not integers), the Segmentation step fails with an unhelpful error in the UpdateSegmentsTable element.	PFUN-17715
When running segmentation, the percentile is not well calculated with floor/target /stretch having the same values.	PFUN-17870
Calculation gets stuck in Segmentation.	PFUN-17892

The “Replace missing dimension values with” input in the Definition step no longer has its correct type string, but is now a numeric input instead.	PFUN-18174
When running the segmentation stage, you can occasionally get the “out of range for type double precision” error.	PFUN-18282
Out of the existing functions that can be used to read values in a SQL query none currently works for both `api.getDatamartContext.executeSqlQuery` (uses JOOQ) and `model.loadTable` (does not use JOOQ).	PFUN-18285
In the Definition step, if you delete the default value of the ”Replace missing dimension values with” user input (the input is left blank), then apply settings and run the model, the user input is then filled with ”__missing__” (meaning it is no longer empty).	PFUN-18333
Price Waterfall elasticity from NG model does not work if there are more than 9 segmentation levels.	PFUN-18485
In an NG model, if a segmentation level contains null values, then by default the model will fill the data with a default string. But if you use ”null” in a query, you get no result. Instead, you need to use the default string in the join conditions.	PFUN-18486