Cristian Scutaru

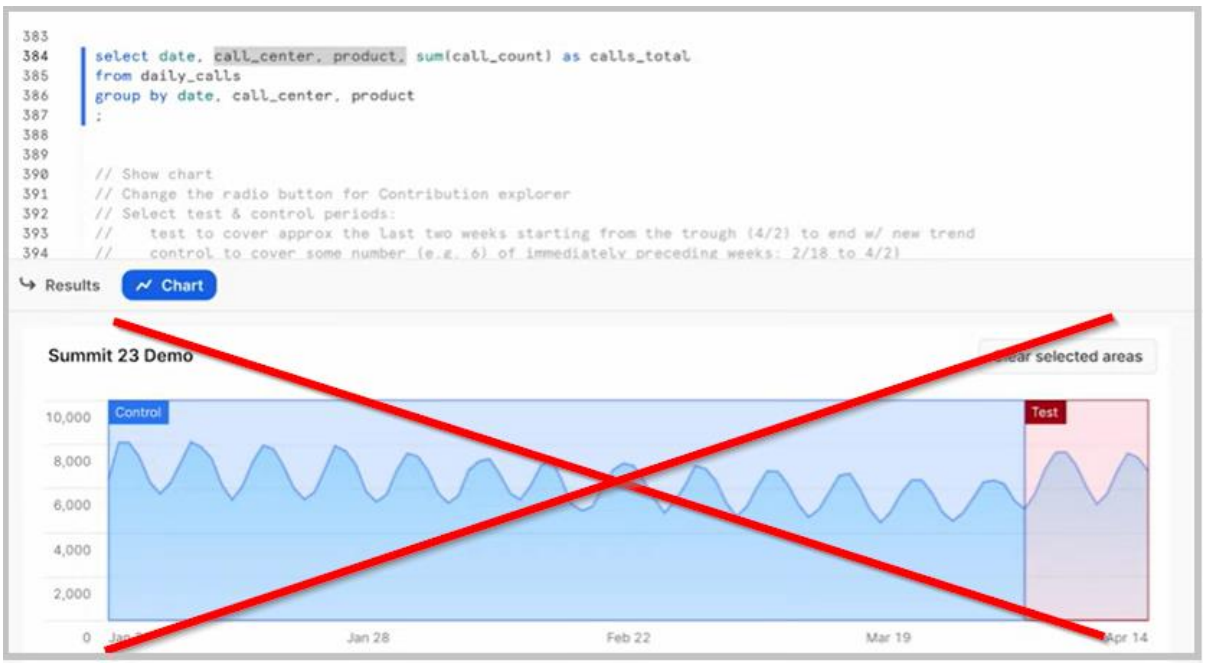
.

4 min read

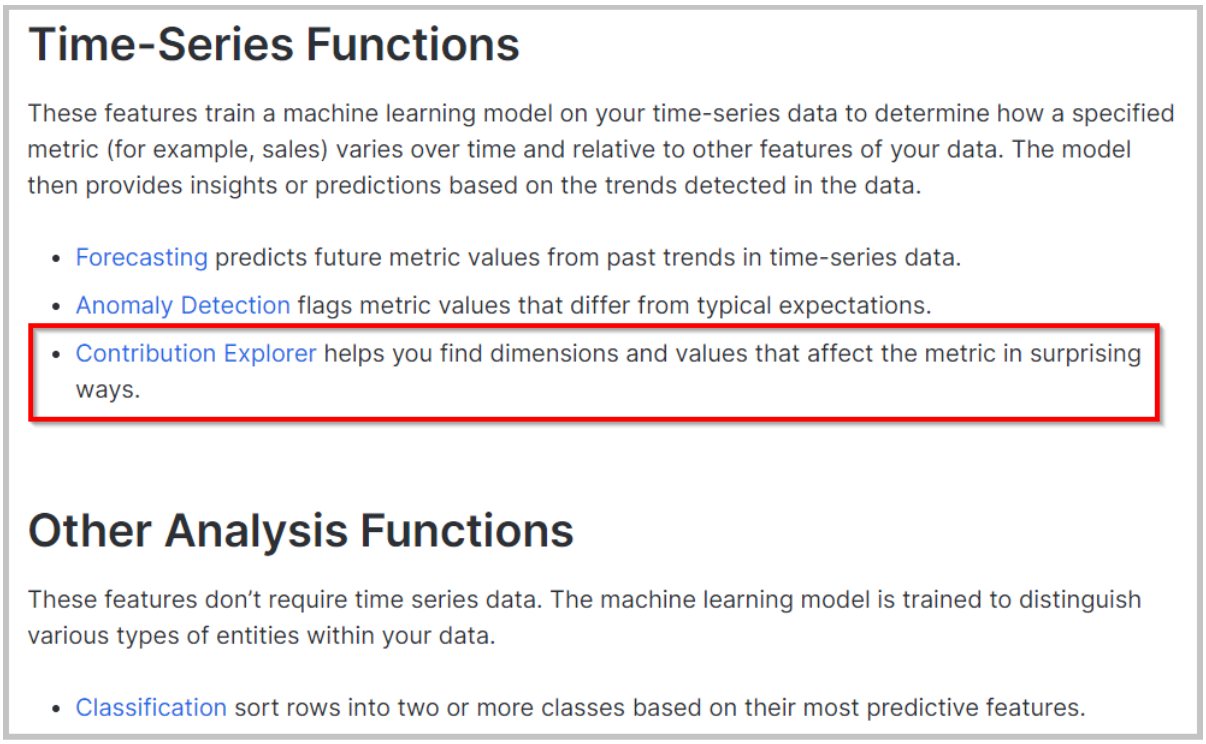
.

View on Medium

Why Snowflake's TOP_INSIGHTS is NOT a Time-Series Function!



The relatively new **Contribution Explorer** feature in ***Snowflake Cortex***—with its **TOP_INSIGHTS** ML-powered function—is advertised everywhere as a *time-series related* ML-powered function, but it is not:



To make things confusing, this Snowflake PM presented it in the context of a split time-series as well, only few months ago:

https://medium.com/media/206db8e4e4a96fcd980fef59a252a6b1/href

Unlike the **Time Series Forecasting** and **Anomaly Detection** classes—which require a date in strict time series format in their constructor (sorted, with no gaps), see the TIMESTAMP_COLNAME parameter—you do not pass ***at all*** any such field to TOP_INSIGHTS:

```
SNOWFLAKE.ML.TOP_INSIGHTS(
  <categorical_dimensions>, <continuous_dimensions>,
  <metric>, <label> )
```

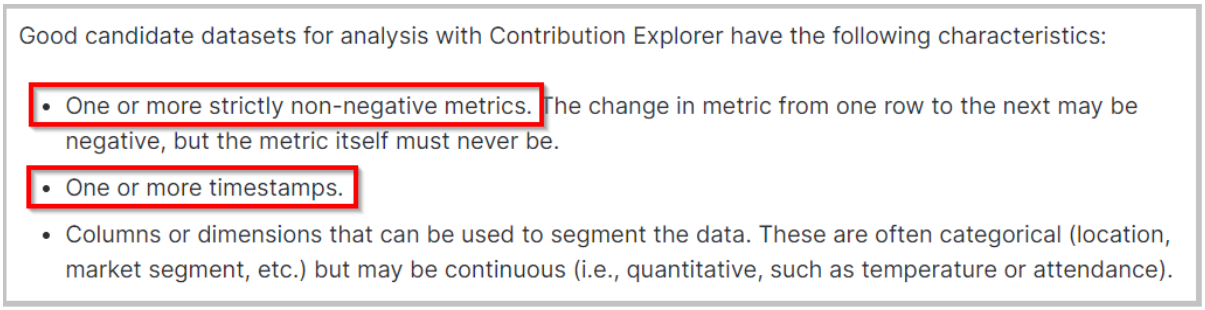
> ***All that TOP_INSIGHTS cares about is to get two unsorted collections of rows, one for training (or "control"), the other one for test. And the function will let you know what dimensions and segments (specific values, or range of values) contributed to the collective change in the test set, when compared to the control set overall. We do not care about patterns, seasonality, or any other things specifics to a strictly ordered time-series.***

TOP_INSIGHTS does not care AT ALL if you collected these points from a sorted time series or not. In fact, it's even confusing and misleading to associate these unordered

collections of rows with a time series at all, to make people believe that a time-series is required here.

Try to pass together M rows for the control set and N rows for the test, in any arbitrary order, and even interlaced. Then mixed them up, in a different arbitrary order, and call TOP_INSIGHTS again: it will return the exact same results, for every returned values (contributors, metrics, etc.).

It's also wrong and misleading what they say here (in their online doc as well), as **a timestamp will not influence at all the end result**. In fact, you don't even need any timestamp, as I said before:


Good candidate datasets for analysis with Contribution Explorer have the following characteristics:

- One or more strictly non-negative metrics. The change in metric from one row to the next may be negative, but the metric itself must never be.
- One or more timestamps.
- Columns or dimensions that can be used to segment the data. These are often categorical (location, market segment, etc.) but may be continuous (i.e., quantitative, such as temperature or attendance).

Quick Demo

The online samples are scarce at this time, but there are at least two examples where the datasets have no timestamps at all. And even the metric is not required. I used a similar dataset myself, to determine the likelihood of a bank customer to subscribe a term deposit.

Here are the same 7 rows that I passed to TOP_INSIGHTS in two separate calls, but interlaced and sorted in different ways. The last Y column separates the control and test sets:

	JOB	MARITAL	AGE	Y
1	management	married	59	FALSE
2	admin.	married	39	FALSE
3	blue-collar	divorced	53	FALSE
4	blue-collar	married	31	FALSE
5	blue-collar	married	44	FALSE
6	blue-collar	married	52	TRUE
7	technician	married	31	TRUE

	JOB	MARITAL	AGE	Y
1	blue-collar	married	31	FALSE
2	technician	married	31	TRUE
3	admin.	married	39	FALSE
4	blue-collar	married	44	FALSE
5	blue-collar	married	52	TRUE
6	blue-collar	divorced	53	FALSE
7	management	married	59	FALSE

We have no target *metric* being investigated (we always pass 1.0 for that parameter), as we just want to see how the categorical job and marital dimensions, and the continuous age dimension, contributed to the overall changes in the test set, when compared to the control set. Because this is what the function does in fact. Here's what's usually returned by ***both*** calls:

	CONTRIBUTOR	···	ACTUAL	EXPECTED	RATIO
1	["age <= 52.5"]		2	1	166%
2	["job = admin."]		0	0	0%
3	["age > 48.0", "job = management", "marital = married"]		0	0	0%
4	["job = blue-collar", "marital = divorced"]		0	0	0%
5	["age > 52.5"]		0	0	0%
6	["marital = divorced"]		0	0	0%
7	["age <= 48.0", "job = blue-collar", "marital = married"]		0	0	0%
8	["job = management"]		0	0	0%
9	["job = management", "marital = married"]		0	0	0%
10	["job = admin.", "marital = married"]		0	0	0%

Just be aware it's a statistical ML algorithm behind—apparently using decision trees—and the result may not be always the same. **But it does not depend on the order of rows.**

The query I used (uncomment "ORDER BY age" for a different sort order of the same input rows):

```
WITH source AS (
```

```
    (SELECT job, marital, age, y
    FROM TEST.PUBLIC.MARKETING
    WHERE not y
    ORDER BY duration ASC
    LIMIT 5)
    UNION
    (SELECT job, marital, age, y
    FROM TEST.PUBLIC.MARKETING
    WHERE y
    ORDER BY duration DESC
    LIMIT 2)
    -- ORDER BY age
),
input AS (
    SELECT
        {
            'job': job,
            'marital' : marital
        } AS cat_dims,
        {
            'age': age
        } AS cont_dims,
        1.0::float AS metric,
        y::boolean AS label
```

```
    FROM source),
analysis AS (
    SELECT res.*
    FROM input,
        TABLE(SNOWFLAKE.ML.TOP_INSIGHTS(cat_dims, cont_dims, metric,
label)
        OVER (PARTITION BY 0)) res)
SELECT contributor,
    metric_test AS actual,
    TRUNC(expected_metric_test) AS expected,
    TRUNC(relative_change * 100) || '%' AS ratio
FROM analysis
WHERE ABS(relative_change - 1) > 0.4
    AND NOT ARRAY_TO_STRING(contributor, ',') LIKE '%not%'
ORDER BY actual DESC
LIMIT 10;
```