VerticaPy Newsletter

With the release of VerticaPy versions 1.0.0 and 1.0.1, we are thrilled to announce many new features and additions to VerticaPy! The below sections highlight what's new in each of these versions. For more information about these new features, including examples, check out the new and improved VerticaPy documentation.

VerticaPy 1.0.0

Support

- Python versions 3.10-3.12 are now supported, with a minimum requirement of Python 3.9.
- Several modules have been deprecated. It is recommended to transition to the updated syntax. For more information, see the documentation.

Machine Learning Support

- We've added support for many Vertica algorithms, including
  - Isolation Forest
  - KPrototypes
  - PoissonRegressor
  - AR
  - MA
  - ARMA
  - ARIMA
  - TfidfVectorizer
- With the newly supported `XGBClassifier.features_importance()` method, you can find the feature importance for XGBoost models. Link
- New classification metrics, which use various averaging techniques, are now available for multiclass data/models.
- We have integrated model versioning and model tracking into VerticaPy, allowing users to register and work with models inside the database. Click here for more information.
- For better consistency, `verticapy.machine_learning.model_selection.statistical_tests.seasonal_decompose` now handles multiple variables using the ROW data type. Link

Bug Fixes

Various bug fixes, including adjustments to R squared, Prevalence Threshold, and improvements to several methods—such as `vDataFrame.balance()`.

Other

- A new dataset, "Africa Education", has been added to `verticapy.datasets`. Link
- `vDataFrame.SQL` has been deprecated. Now, `verticapy.vDataFrame` can be used to directly create a vDataFrame from SQL queries.

For example:

```
 import verticapy as vp
vp.vDataFrame(
"(SELECT pclass, embarked, AVG(survived) FROM public.titanic GROUP BY 1, 2) x"
)
```

Click here for more information.

- Import structures have been updated. The code was completely restructured for better readability and collaboration.

VerticaPy 1.0.1

Along with the above 1.0.0 improvements, VerticaPy version 1.0.1 includes the following updates:

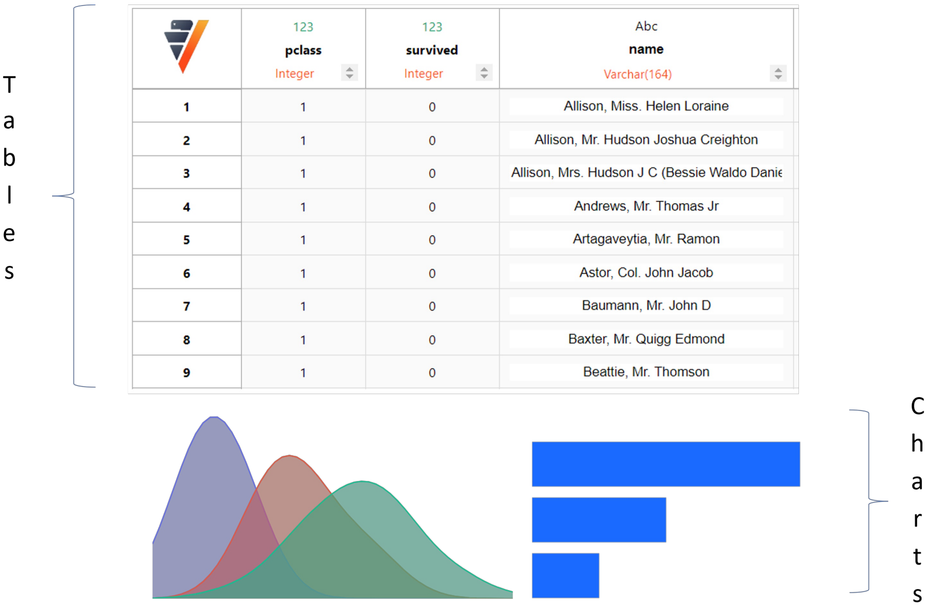
Options

The `verticapy.set_option()` function now allows you to set the following options:

- `max_cellwidth`: Maximum width of VerticaPy table cells.

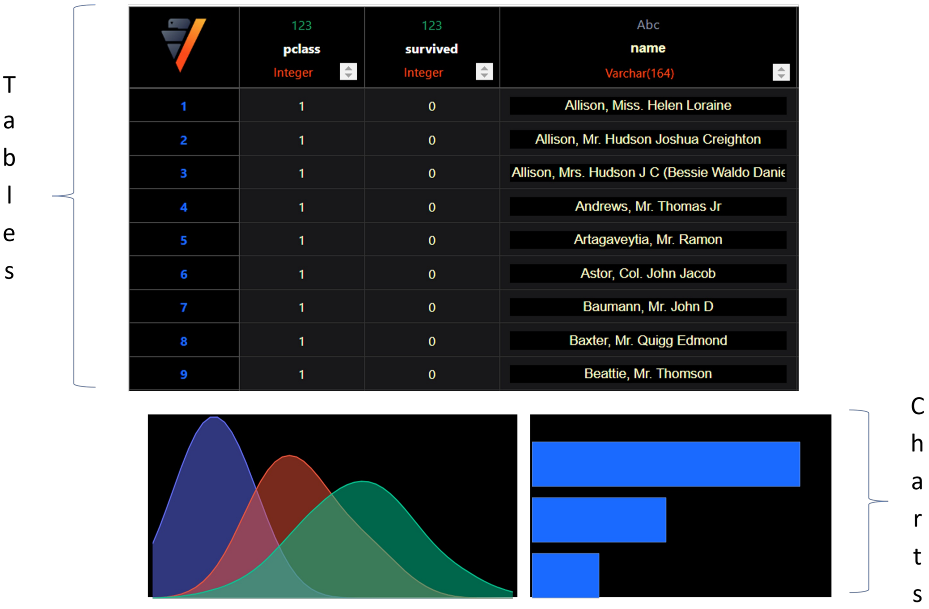
- `max_tableheight` : Maximum height of VerticaPy tables.
- `theme` : Set the display theme for VerticaPy objects to 'light' or 'dark'. 'dark' is recommended for night use, and 'light' is the default.

The default theme is "Light". It is recommended for daily use:



	123 pclass Integer	123 survived Integer	Abc name Varchar(164)
1	1	0	Allison, Miss. Helen Loraine
2	1	0	Allison, Mr. Hudson Joshua Creighton
3	1	0	Allison, Mrs. Hudson J C (Bessie Waldo Danie
4	1	0	Andrews, Mr. Thomas Jr
5	1	0	Artagaveytia, Mr. Ramon
6	1	0	Astor, Col. John Jacob
7	1	0	Baumann, Mr. John D
8	1	0	Baxter, Mr. Quigg Edmond
9	1	0	Beattie, Mr. Thomson

On the other hand, the "Dark" theme is suited for night-time use:



	123 pclass Integer	123 survived Integer	Abc name Varchar(164)
1	1	0	Allison, Miss. Helen Loraine
2	1	0	Allison, Mr. Hudson Joshua Creighton
3	1	0	Allison, Mrs. Hudson J C (Bessie Waldo Danie
4	1	0	Andrews, Mr. Thomas Jr
5	1	0	Artagaveytia, Mr. Ramon
6	1	0	Astor, Col. John Jacob
7	1	0	Baumann, Mr. John D
8	1	0	Baxter, Mr. Quigg Edmond
9	1	0	Beattie, Mr. Thomson

For switching the themes, following syntax can be used:

```
 import verticapy as vp
vp.set_option("theme", "dark")
```

Diagnostics

The `verticapy.performance.vertica.qprof.QueryProfiler` class offers an extended set of functionalities, enabling the creation of complex trees with multiple metrics. This can help in finding ways to improve the performance of slow-running queries.

Other

The docstrings throughout the documentation have been enriched with examples and further details, providing an improved doc experience.