



Overview of Data Science

Overview


 15 mins

Table of Contents

- [Data Science](#)
- [Introduction to VerticaPy](#)
 - [Vertica](#)
 - [Creation of VerticaPy](#)
 - [vDataFrame](#)

Before we begin a deep dive into other focused subject matters, it is essential we get an overview of the topic of Data Science. Along with that, this lesson will also introduce VerticaPy for those who have not heard of it.

What you will learn in this module?

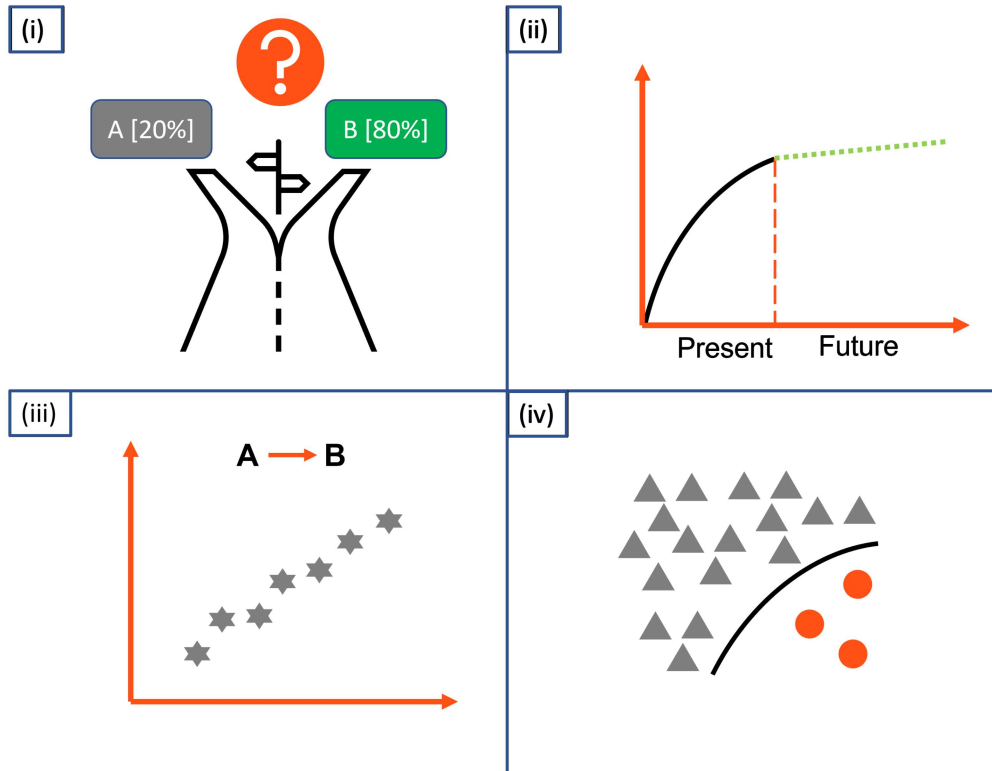
- What is Data Science?
 - What are some of the applications of Data Science?
 - What is VerticaPy?
-

Data Science

Data Science is the the practice of analyzing data to draw insights, or to predict scenarios, using a wide range of methods. It spans the entire lifespan of data, from collecting, storing, manipulating, analyzing, all the way up till decision making.

There are many advantages of Data Science for any organization including:

- (i) **Informed Decisions:** Studying the data can give insights which can be helpful in making investment decisions.
- (ii) **Predictive Analysis:** Based on other trends, forecast can be made about future events.
- (iii) **Pattern Discoveries:** By understanding patterns, groups of user can be segmented and targetted differently.
- (iv) **Find Causal Relationships:** Sometimes organizations want to find the cause-effect relationships and separate them from mere associations.



Knowledge Check: Figuring out the optimum advertisement portfolio to increase sales - in which category does it fall?

- ☒ Informed Decisions
- ☐ Predictive Analysis
- ☐ Pattern Discovery

submit

Data Science is generally a continuous process which can be listed as follows:

- *Step 0:* Problem Question

Understanding the business problem and figuring out the right questions to answer from the data. This step is perhaps the most crucial step because if the problem is not understood correctly then a lot of time can be spent in vain

- *Step 1: Data Collection*

In this step data is collected from different sources such as surveys, existing databases, sensors etc

- *Step 2: Data Ingestion/Storage*

Data is brought all other sources and stored into a database.

- *Step 3: Data Preparation*

Data is thoroughly examined to find discrepancies, anomalies, missing values etc. They are then treated to *clean* the data

- *Step 4: Data Exploration*

Data is visualized in different plots to understand the data and gain initial insights.

- *Step 5: Model Building*

Depending on the type of data and the problem question, models are built.

- *Step 6: Deployment and Monitoring*

The last phase is continuous as the model needs to be monitored and if needed adjusted as per the results in actual environment.

Most of these steps will be covered in some detail in the *Data Science Essentials* lesson series.

In order to make it more relatable, let us look an example of Data Science application. One of the most common examples are the targeted ads that we come across the internet. The ads seem to read our mind, whereas Data Science is not telepathy. It is only picking up cues from all the data about a user that is out there. Data Science is not restricted to business cases, but it is being increasingly used in almost all domains of our life; from schools to prisons, and from hospitals to military, data sciece is everywhere!

Lastly, let us talk about the languages that people use for Data Science. Until recently, the most popular language for Data Science was R. But python is slowly aking over with its ever-increasing libraries. SQL is also used very heavily in this domain especially for relational databases which can be efficiently queried.

Introduction to VerticaPy

Before we talk about VerticaPy, it is imperative that we know what Vertica is.

Vertica

Taking up the analogy shared above, Vertica could be thought of as a place where data is stored and analyzed. But it is not just any data warehouse, Vertica is a columnar data storage platform designed to handle large volumes of data, which enables very fast query performance in traditionally intensive scenarios.

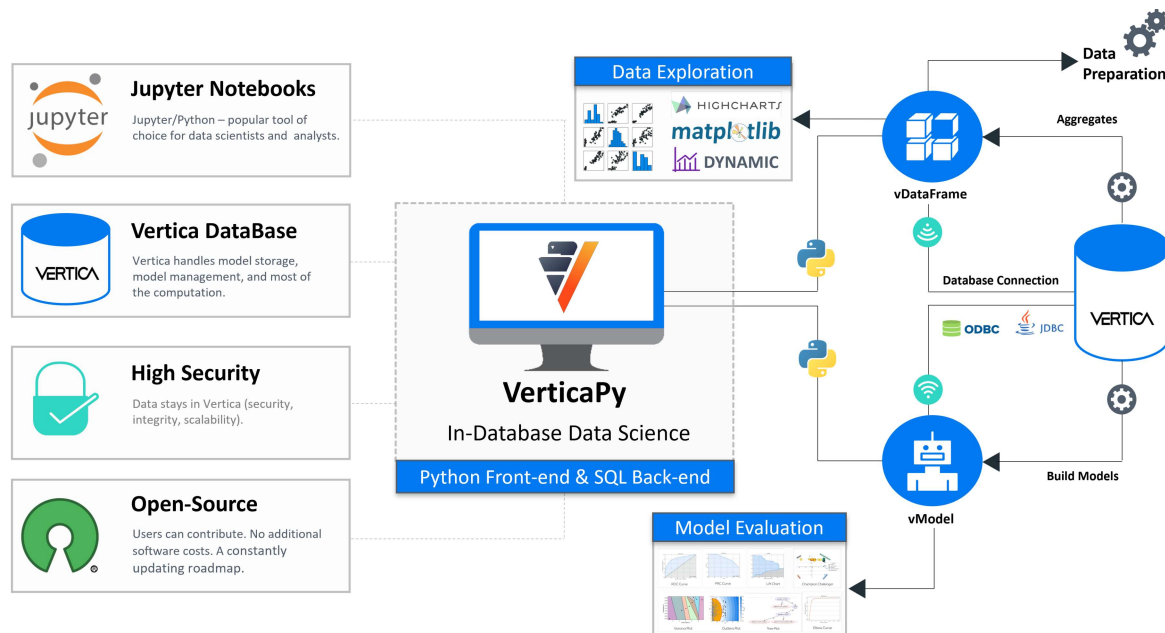
However, Vertica is based on SQL. And SQL alone isn't flexible enough to meet the needs of data scientists. Python has quickly become the most popular tool in this domain, owing much of its flexibility to its high-level of abstraction and impressively large and ever-growing set of libraries. Its accessibility has led to the development of popular and performant APIs, like pandas and scikit-learn, and a dedicated community of data scientists. To answer this customer need, [Badr Ouali](#), who was an intern at the time, came up with an ingenious solution: VerticaPy.



VerticaPy

In 2016, VerticaPy started as *Vertica ML Python* when Badr joined the scalability of Vertica with the flexibility of Python. It was designed to be a python wrapper for the SQL-based Vertica. VerticaPy was set-up as an [open source project on GitHub](#) so that users can request for features, report bugs, and contribute as well. One of the first goals were to simplify Data Exploration and Data Modelling by incorporating the necessary backend-python.

Upon seeing the customer response and adoption, Badr decided to extend it to much more step by step. And gradually, almost all of the functions of Vertica are now available on Python. Not just that, but there are certain functions which are only available in VerticaPy with much more ease and flexibility. One such object is `vDataFrame`.



22

vDataFrame


The **Virtual DataFrame** is the main object and star of the library and acts as the perfect transition between small and Big Data. The principle is quite simple: As Vertica is a powerful columnar massive parallel processing (MPP) database with many built-in functions, we want it to do as much of the computation work as possible.

Indeed, columnar orientation allows for high compression, and its structure inherently avoids unnecessary parsing when retrieving data. MPP allows to parallelize our computations across the different nodes.

The best way to take advantage of your data is by simply keeping it in your Vertica database, rather than within the limitations of working memory. VerticaPy pushes all computation to your Vertica database before aggregating the final result, so you can get the best of both worlds: Vertica's power and Python's flexibility.

With Python, it's easy to add abstractions, and the vDataFrame acts as the primary abstraction layer. Simple but powerful, it'll help any user through the data science life cycle.

Check out all the efficient and easy-to-use operations of vDataFrame [here](#).

	Abc country <small>Varchar(48)</small>	123 year <small>Int</small>	123 pop <small>Numeric(13,2)</small>	Abc continent <small>Varchar(20)</small>	123 lifeExp <small>Numeric(10,6)</small>	123 gdpPercap <small>Float</small>
1	Afghanistan	1952	8425333.0	Asia	28.801	779.4453145
2	Afghanistan	1957	9240934.0	Asia	30.332	820.8530296
3	Afghanistan	1962	10267083.0	Asia	31.997	853.10071
4	Afghanistan	1967	11537966.0	Asia	34.02	836.1971382
5	Afghanistan	1972	13079460.0	Asia	36.088	739.9811058
6	Afghanistan	1977	14880372.0	Asia	38.438	786.11336
7	Afghanistan	1982	12881816.0	Asia	39.854	978.0114388
8	Afghanistan	1987	13867957.0	Asia	40.822	852.3959448
9	Afghanistan	1992	16317921.0	Asia	41.674	649.3413952
10	Afghanistan	1997	23227115.0	Asia	45.768	626.216355

Rows: 1-100 | Columns: 6

Author Name: Umar Farooq Ghumman

Author Contact: umarfarooq.ghumman@vertica.com

Resources

- Why VerticaPy