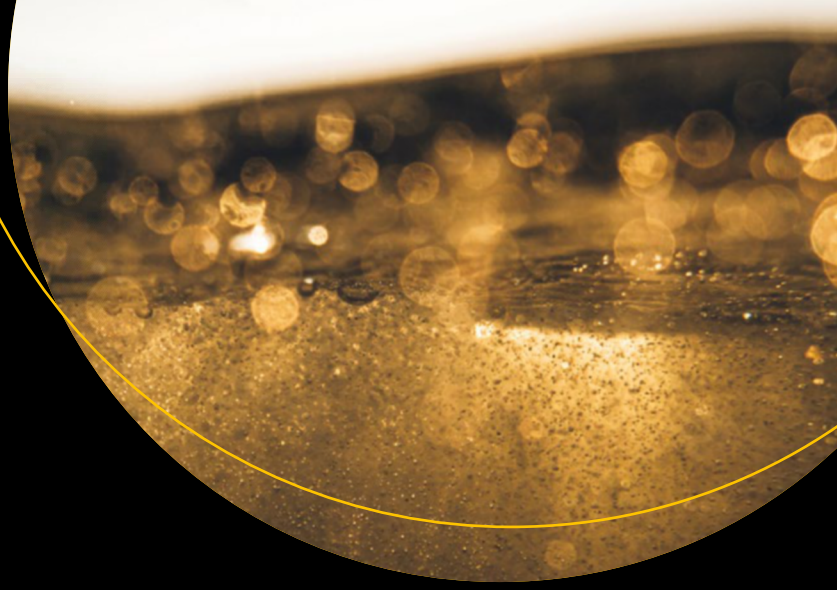

Data Engineering for Machine Learning Pipelines

From Python Libraries to ML Pipelines and Cloud Platforms

Pavan Kumar Narayanan

Apress®

Data Engineering for Machine Learning Pipelines

From Python Libraries to ML Pipelines and Cloud Platforms

Pavan Kumar Narayanan

Apress®

Data Engineering for Machine Learning Pipelines: From Python Libraries to ML Pipelines and Cloud Platforms

Pavan Kumar Narayanan
Sheridan, USA

ISBN-13 (pbk): 979-8-8688-0601-8 ISBN-13 (electronic): 979-8-8688-0602-5
https://doi.org/10.1007/979-8-8688-0602-5

Managing Director, Apress Media LLC: Welmoed Spahr
Acquisitions Editor: Celestin Suresh John
Development Editor: Laura Berendson
Editorial Project Manager: Gryffin Winkler

Cover designed by eStudioCalamar

Cover photo by Bradley Dunn on Unsplash

Distributed to the book trade worldwide by Springer Science+Business Media New York, 1 New York Plaza, Suite 4600, New York, NY 10004-1562, USA. Phone 1-800-SPRINGER, fax (201) 348-4505, e-mail orders-ny@springer-sbm.com, or visit www.springeronline.com. Apress Media, LLC is a California LLC and the sole member (owner) is Springer Science + Business Media Finance Inc (SSBM Finance Inc). SSBM Finance Inc is a **Delaware** corporation.

For information on translations, please e-mail booktranslations@springernature.com; for reprint, paperback, or audio rights, please e-mail bookpermissions@springernature.com.

Apress titles may be purchased in bulk for academic, corporate, or promotional use. eBook versions and licenses are also available for most titles. For more information, reference our Print and eBook Bulk Sales web page at http://www.apress.com/bulk-sales.

Any source code or other supplementary material referenced by the author in this book is available to readers on GitHub. For more detailed information, please visit https://www.apress.com/gp/services/source-code.

If disposing of this product, please recycle the paper

To my beloved father

Table of Contents

About the Author

Pavan Kumar Narayanan has an extensive and diverse career in the information technology industry, with a primary focus on the data engineering and machine learning domains. Throughout his professional journey, he has consistently delivered solutions in environments characterized by heterogeneity and complexity. His experience spans a broad spectrum, encompassing traditional data warehousing projects following waterfall methodologies and extending to contemporary integrations that involve application programming interfaces (APIs) and message-based systems. Pavan has made substantial contributions to large-scale data integrations for applications in data science and machine learning. At the forefront of these endeavors, he has played a key role in delivering sophisticated data products and solutions, employing a versatile mix of both traditional and agile approaches. Pavan completed his Master of Science in Computational Mathematics at Buffalo State University, Operations Research at the University of Edinburgh, and Information Technology at Northern Arizona University. He can be contacted at pavan.narayanan@gmail.com.

About the Technical Reviewer

Suvoraj Biswas is a leading expert and emerging thought leader in enterprise generative AI, specializing in architecture and governance. He authored the award-winning book *Enterprise GENERATIVE AI Well-Architected Framework & Patterns*, acclaimed by industry veterans globally. Currently an enterprise solutions architect at Ameriprise Financial, a 130-year-old Fortune 500 organization, Biswas has over 19 years of IT experience across India, the United States, and Canada. He has held key roles at Thomson Reuters and IBM, focusing on enterprise architecture, architectural governance, DevOps, and DevSecOps.

Biswas excels in designing secure, scalable enterprise systems, facilitating multi-cloud adoption, and driving digital transformation. His expertise spans Software as a Service (SaaS) platform engineering, Amazon Web Services (AWS) cloud migration, generative AI implementation, machine learning, and smart IoT platforms. He has led large-scale, mission-critical enterprise projects, showcasing his ability to integrate advanced cutting-edge technologies into practical business strategies.

Introduction

The role of data engineering has become increasingly crucial. Data engineering is the foundation on which organizations build their reporting, analytics, and machine learning capabilities. Data engineering as a discipline transforms raw data into reliable and insightful informational resources that provide valuable insights to organizations. However, data engineering is much more than that. It is a complex and iterative process.

If an enterprise project can be seen as constructing a multi-storied building, then data engineering is where the execution happens—from architecting and designing the structure of the data solution to laying the data storage foundation; installing the data processing frameworks; developing pipelines that source, wrangle, transform, and integrate data; and developing reports and dashboards while ensuring quality and maintenance of the systems, pipelines, reports, and dashboards, among several other things.

This book is designed to serve you as a desk reference. Whether you are a beginner or a professional, this book will help you with the fundamentals and provide you with job-ready skills to deliver high-value proposition to the business. This book is written with an aim to bring back the knowledge gained from traditional textbook learning, providing an organized exploration of key data engineering concepts and practices. I encourage you to own a paper copy. This book will serve you well for the next decade in terms of concepts, tools, and practices.

Though the content is exhaustive, the book covers around 60% of data engineering topics. Data engineering is a complex and evolving discipline; there are so many more topics that are not touched on. As a reader, I encourage you to contact me and share your remarks and things you witnessed in the data space that you wish to share. I am thankful to those who have already written to me, and I wish to see more of these efforts from the readers and enthusiasts.

Having said that, this book covers a wide range of topics that are practiced in modern data engineering, with a focus on building data pipelines for machine learning development. This book is organized into the following topics:

- Data wrangling: Techniques for cleaning, transforming, and preparing data for analysis using Pandas, Polars, and GPU-based CuDF library
- Data validation: Ensuring data quality and integrity throughout the pipeline using Pydantic, Pandera, and Great Expectations
- Concurrency computing: Leveraging parallel processing for improved performance using Dask and DaskML
- APIs: Designing and working with APIs for data exchange using FastAPI
- Streaming pipelines: Designing and working with real-time data flows efficiently using Kafka
- Workflow orchestration: Managing complex data workflows and dependencies using cron, Airflow, and Prefect
- Cloud computing: Utilizing cloud services for scalable data engineering solutions using AWS, Azure, and Google Cloud Platform (GCP)

The book covers major cloud computing service providers. The reason is that organizations are increasingly moving toward a multi-cloud approach. And so, being equipped with knowledge on diverse cloud environments is crucial for your career success. This book delivers that.

If you are a data engineer looking to expand your skillset and move up the ladder, a data scientist who seeks to understand how things work under the hood, a software engineer who is already doing the work, or a student who is looking to gain a position, this book is essential for you. Even if you are in a functional domain or a founder of a start-up, interested in setting up a team and good data engineering practice, this book will help you from start to end. This book is a guide that will help with learning for people at every stage (from novice to expert).

I highly encourage you not to seek solutions from chat bots during learning. At the time of writing this (June 2024), many bot programs either provide legacy code or steer you into their own direction and approach. It may seem better because of the way information is presented, but it slows down your learning and professional development. Even to this date, I find search engines to be more effective, which take me to Q&A sites where one can witness how a solution has been arrived at and why other approaches didn't succeed. Furthermore, I find Google search to be effective in terms of locating a certain method or a function in an API documentation that may be better than the website itself.

At the end of this book, you will be equipped with skills, knowledge, and practices that can be derived from a decade of experience in data engineering. Many vendors and cloud computing service providers provide you with free credits if you register for the first time. I encourage you to make use of these free credits, explore various tools and services, and gain knowledge. And when you are done, please remember to clean up your workspace, shut down any virtual machines and databases, take a backup of your work, and delete your workspace in the cloud to prevent additional charges.

CHAPTER 1

Core Technologies in Data Engineering

Introduction

This chapter is designed to get you started. As we begin our journey into advanced data engineering practices, it is important to build a solid foundation in key concepts and tools. This chapter is designed to provide a refresher on Python programming and software version control using GitHub and a high-level review of structured query language (SQL). While these may seem not connected and different from each other, they play an important role in developing effective data engineering pipelines. Python's data ecosystem, software version controlling systems using Git, and working with relational database systems using SQL will certainly solidify your foundations and help you tackle data engineering challenges. These are challenging and complex topics, especially reviewing them together; I hope you persist, learn, and practice on your own.

By the end of the chapter, you will

- Understand some of the key Python programming concepts relevant to advanced software engineering practices.
- Gain an appreciation toward software version control systems and git; create git repositories; and perform essential version control operations like branching, forking, and submitting pull requests.
- Have enough knowledge to set up your own code repository in GitHub.

P. K. Narayanan, *Data Engineering for Machine Learning Pipelines*,
https://doi.org/10.1007/979-8-8688-0602-5_1

- Review fundamental SQL concepts for querying and manipulating data in database systems.
- Identify advanced SQL concepts and best practices for writing efficient and optimized SQL code.

Engineering data pipelines is not easy. There is a lot of complexity and it involves leveraging analytical thinking. The complexity includes but is not limited to varying data models, intricate business logic, tapping into various data sources, and so many other tasks and projects that make up the profession. Often, data engineers do build machine learning models to identify anomalies and build advanced imputations on the data. I have taken some of the topics in my experience that would serve value to the data engineer or software developer who is engineering pipelines.

Python Programming

There are two major versions of the Python programming language, Python 2 and Python 3. Python 2 has reached end of life for support. Python 3 is the de facto language version for Python programming. The Python programming language (version 3.*) is still the most preferred language for development, comes with extensive community-backed data processing libraries, and is widely adopted. In this section, let us take some time to look at some of the Python programming concepts that may help, in addition to your basic programming knowledge. These are discussed at a high level. I encourage you to perform further investigation into these classes, methods, and arguments that can be passed into these methods.

F Strings

Python f strings allow you to easily insert variables and expressions within a print string. When compared with traditional string formatting, f strings are a lot more readable and easy to understand. You can create an f string by placing the character F in front of the string, before quotes. The character is case insensitive, meaning you can either insert an F or an f. Here is an example:

```
Greetings = f"Hello, it is a beautiful day here, at the park today"
```

Now you can insert variables and expressions within the f string. We have shown the preceding f string without inserts for demonstration purposes. Here is how you can perform that:

```
Fruit = "berries"
Cost = 1.5
Number = 3

print(f"the {fruit} costs {cost} per piece, totals to $ {cost*number}
dollars")
```

will return

```
the oranges costs 1.5 per piece, totals to $ 4.5 dollars
```

Python Functions

In programming, functions are a set of instructions that can be executed together to accomplish a specific task or operation. It may or may not accept inputs and may or may not return output. Let us look at an example of a function:

```
def add(a,b):
    c = a+b
    return c
```

When you call this function

```
print(add(2,3)
```

it will return

```
5
```

Let us look at a bit more substantial function. Here, we have a function that takes Celsius as input and calculates Fahrenheit as output:

```
def f_calc(celsius):
    return ((9/5 * celsius) + 32)
```

This is the same as the following function. Both are syntactically equal. We will discuss this way of defining functions in another section:

```
def f_calc(celsius: int) -> float:
    return ((9/5 * celsius) + 32)
```

In the preceding code, we have passed a variable named celsius of type integer. Inside the function we are calculating a value and returning that value in floating point type. This is how functions can accept input parameters, process them, and provide output.

Advanced Function Arguments

*args

Let us look at using the "*args" parameter in a function. The arguments parameter in functions can be used to pass a variable number of arguments to a function. If you have the syntax "*args" once, you can pass several numbers of variables, without having to define each one of them. Let us look at two examples in contrast to understand this concept better:

```
def add(a,b):
    c = a+b
    return c
```

When you run this

```
add(2,3)
```

it will return

```
5
```

What if we do not know how many input parameters we need? We can use arguments here:

```
def add_args(*args):
    total_sum = 0
    for i in args:
        total_sum += i
    return total_sum
4
```

So when you run this

```
add_args(3,4,5,6,7)
```

it will return

```
25
```

Note You can also pass a list to the function and still obtain the functionality. But with arguments, you can call an empty function and return none.

**kwargs

This is quite similar to arguments, and the difference is that you can have a key associated with each item in the argument:

```
def add_kwargs(**kwargs):
    total_sum = 0
    for i in kwargs.values():
    total_sum += i
    return total_sum
```

Use of args and kwargs can be highly beneficial in enhancing the flexibility of a function, complementing the decorator implementation and passing a varying number of parameters as inputs.

Lambda Functions

In Python, lambdas are short for lambda expressions. They are inline functions that evaluate one single expression. That single expression is evaluated when the expression is called. Let us look at this with an illustration:

```
calculate_fahrenheit = lambda celsius: 9/5*celsius + 32
```

So when you call this function

```
calculate_fahrenheit(27)
```

it will return

```
80.6
```

Decorators in Python

Decorators in Python are a way to add additional functionality to your existing function. The decorator itself is a function. In the parameter column, it would take another function, as an input parameter. In calculus terms, it is equal to the function of a function. If math is not your cup of tea, then let's look at a simple example.

Here is a test decorator:

```
def testdec(func):
    def wrapper():
        print("+++Beginning of Decorator Function+++")
        func()
        print("+++End of Decorator Function+++")
    return wrapper
```

Let us try to use this decorator on a sample function:

```
@testdec
def hello():
    print("Hello!")
```

When we run this function

```
hello()
```

it will provide us

```
+++Beginning of Decorator Function+++
Hello!
+++End of Decorator Function+++
```

Type Hinting

Type hinting is a Python feature that lets you specify the data types of the variable declarations, arguments in functions, and the variables that these functions return. Type hints make it easier to understand what data types to expect and which function returns what type of data. The idea behind type hinting is better readability, code maintainability, and being able to debug and troubleshoot errors faster. Type hints are not mandatory; you can still pass an argument without specifying the data type (Python will still check them during runtime).

Here is a simple example demonstrating the type hinting functionality in Python:

```
def some_function(a: float, b: float) -> float:
    c = a+b
    return c
```

Typing Module

The typing module supports type hints for advanced data structures with various features. Here is an example for a dictionary:

```
From typing import Dict

var2: Dict[int, str] = {234: "JohnDoe"}
```

Notice the usage of "Dict." It serves the same purpose as "dict," a dictionary type offered by core Python. "Dict" does not provide additional functionality except you have to specify the data type for both key and value aspects of a dictionary. If you are using Python 3.9 or later, you can use either "Dict" or "dict."

Let us look at another example called the Union, where a variable could be one of few types:

```
from typing import Union

var4: list[Union[int, float]] = [3, 5, 5.9, 6.2]
```