MACHINE LEARNING PIPELINES

TIM AJAR KECERDASAN ARTIFISIAL

OUTLINE

- DEFINITION
- BENEFIT OF USING MACHINE LEARNING PIPELINES
- COMPONENTS OF MACHINE LEARNING PIPELINES

DEFINITION

Machine Learning is an approach to data analysis that involves building and adapting models, which allow programs to learn through experience

Pipeline is split into stages. The stages are interconnected to form a pipe in such a way that instructions enter at one end, progress through the stages, and exit at the other end

Machine learning pipelines refer to a series of interconnected steps or processes that are designed to automate and streamline the workflow involved in training, evaluating, and deploying machine learning models

BENEFIT OF USING MACHINE LEARNING PIPELINES

UNATTENDED RUNS

The pipeline allows to schedule different steps to run in parallel in a reliable and unattended way. It means you can focus on other tasks simultaneously when the process of data modeling and preparation is going on

EASY DEBUGGING

Using pipeline, there is a separate function for each task(such as different functions for data cleaning and data modeling). Therefore, it becomes easy to debug the complete code and find out the issues in a particular step.

EASY TRACKING AND VERSIONING

We can use a pipeline to explicitly name and version the data sources, inputs, and output rather than manually tracking data and outputs for each iteration.

FAST EXECUTION

As we discussed above, in the ML pipeline, each part of the workflow acts as an independent element, which allows the software to run faster and generate an efficient and high-quality output.

BENEFIT OF USING MACHINE LEARNING PIPELINES (CONT.)

COLLABORATION

Using pipelines, data scientists can collaborate over each phase of the ML design process and can also work on different pipeline steps simultaneously.

REUSABILITY

We can create pipeline templates for particular scenarios and can reuse them as per requirement. For example, creating a template for retraining and batch scoring.

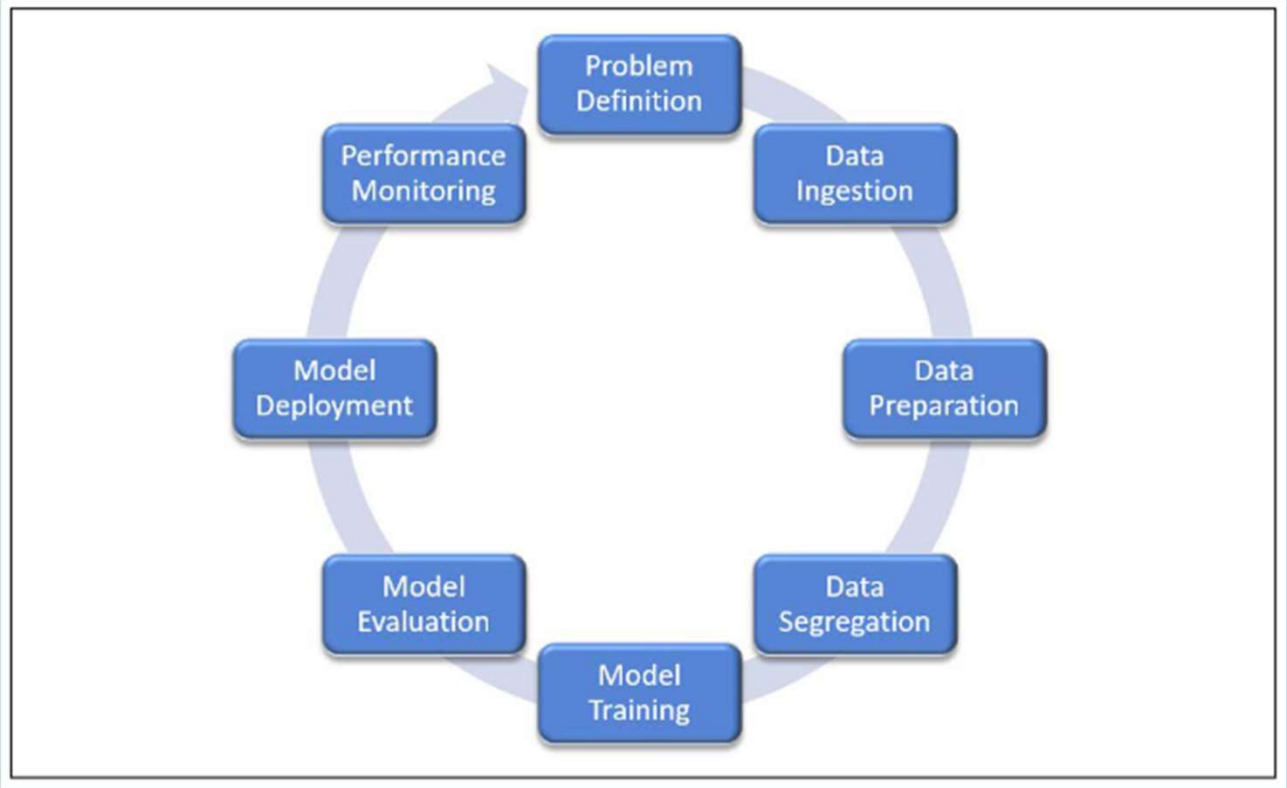
HETEROGENEOUS COMPUTE

We can use multiple pipelines which are reliably coordinated over heterogeneous computer resources as well as different storage locations. It allows making efficient use of resources by running separate pipelines steps on different computing resources, e.g., GPUs, Data Science VMs, etc.

WHY PIPELINES?

- Pipelines let us focus on various tasks while different stages run in an unattended way in parallel or in sequence.
- Our engineers use available resources efficiently, running separate steps on several compute targets.
- It's possible to reuse pipeline templates.
- We don't need to track data and result paths manually or handle pipeline templates, which leads to increased productivity.
- The modular nature of pipelines lets ml solutions develop faster and with higher quality.
- Last but not least: pipelines allow engineers to cooperate in all ml areas, working collectively on its steps, making them a perfect tool for effective teamwork.

COMPONENTS OF MACHINE LEARNING PIPELINES



PROBLEM DEFINITION

This might be the most critical step when setting up your pipeline. Time spent here can save you orders of magnitude of time on the later stages of the pipeline. Asking and framing the right question is paramount.

Consider the following cautionary tale:

"Bob spent years planning, executing, and optimizing how to conquer a hill. Unfortunately, it turned out to be the wrong hill."

DATA INGESTION

Collecting the right data for your pipeline might be a tremendous undertaking. Depending on the problem you are trying to solve, obtaining relevant datasets might be quite difficult.

Another important consideration is to decide how will the data be sourced, ingested, and stored:

- What data provider or vendor should we use? Can they be trusted?
- How will it be ingested? Hadoop, Impala, Spark, just Python, and so on?
- Should it be stored as a file or in a database?
- What type of database? Traditional RDBMS, NoSQL, graph.
- Should it even be stored? If we have a real-time feed into the pipeline, it might not even be necessary or efficient to store the input.
- What format should the input be? Parquet, JSON, CSV

DATA PREPARATION

- Data cleansing
- Filtration
- Aggregation
- Augmentation
- Consolidation
- Missing value
- Duplicate records or value

DATA SEGREGATION

In order to train a model using the processed data, it is recommended to split the data into two subsets:

- Training data
- Testing data

and sometimes into three:

- Training data
- Validation data
- Testing data

MODEL TRAINING

After we train our model with various algorithms comes another critical step. It is time to select which model is optimal for the problem at hand. We don't always pick the best performing model. An algorithm that performs well with the training data might not perform well in production because it might have overfitted the training data. At this point in time, model selection is more of an art than a science but there are some techniques that are explored further to decide which model is best.

MODEL EVALUATION

Once we split the data it is now time to run the training and test data through a series of models and assess the performance of a variety of models and determine how accurate each candidate model is.

This is an iterative process and various algorithms might be tested until you have a model that sufficiently answers your question.

MODEL DEPLOYMENT

Once a model is chosen and finalized, it is now ready to be used to make predictions. It is typically exposed via an API and embedded in decision-making frameworks as part of an analytics solution. Following are a variety of model deployment architectures:

	RESTful API Architecture	Shared DB Architecture	Streaming Architecture	Mobile App Architechture
Training Method	Batch	Batch	Streaming	Streaming
Prediction Method	Real-time	Batch	Streaming	Real-time
Result Delivery	Via RESTful API	Via Shared Database	Streaming via Message Queue	Via in-process API on mobile device
Prediction Latency	Low	High	Very low	Low
System Maintanability	Medium	Easy	Difficult	Medium

MODEL DEPLOYMENT (CONT.)

The architecture chosen for the model deployment it is a good idea to use the following principles:

- Reproducibility - Store all the model inputs and outputs, as well as all relevant metadata such as configuration, dependencies, geography, and time zones. This is especially important for domains that are highly regulated, banking, for example.
- Automation - As early as possible, automate as much as possible of the training and model publishing.
- Extensibility - If models need to be updated on a regular basis, a plan needs to be put in place from the beginning.
- Modularity - As much as possible, modularize your code and make sure that controls are put in place to faithfully reproduce the pipelines across environments (DEV, QA, TEST).
- Testing - Allocate a significant part of your schedule for testing the machine learning pipeline. Automate the testing as much as possible and integrate into your process from the beginning.

PERFORMANCE MONITORING

Once a model makes it into production, our work is not finished. Moving a model into production may not be easy, there are various steps involved in getting the model into production.

The model is continuously monitored to observe how it behaved in the real world and calibrated accordingly. New data is collected to incrementally improve it. Similarly, monitoring a deployed machine learning model requires attention from various perspectives to make sure that the model is performing.

PERFORMANCE MONITORING (CONT.)

Performance in the data science context does not mean how fast the model is running, but rather how accurate are the predictions.

Data scientists monitoring machine learning models are primarily looking at a single metric: drift. Drift happens when the data is no longer a relevant or useful input to the model. Data can change and lose its predictive value.

If the data drifts, the prediction results will become less accurate because the input features are out of date or no longer relevant.

THANK YOU