MACHINE LEARNING PIPELINE

MIP-ML-08
SOHINI ROY CHOWDHURY

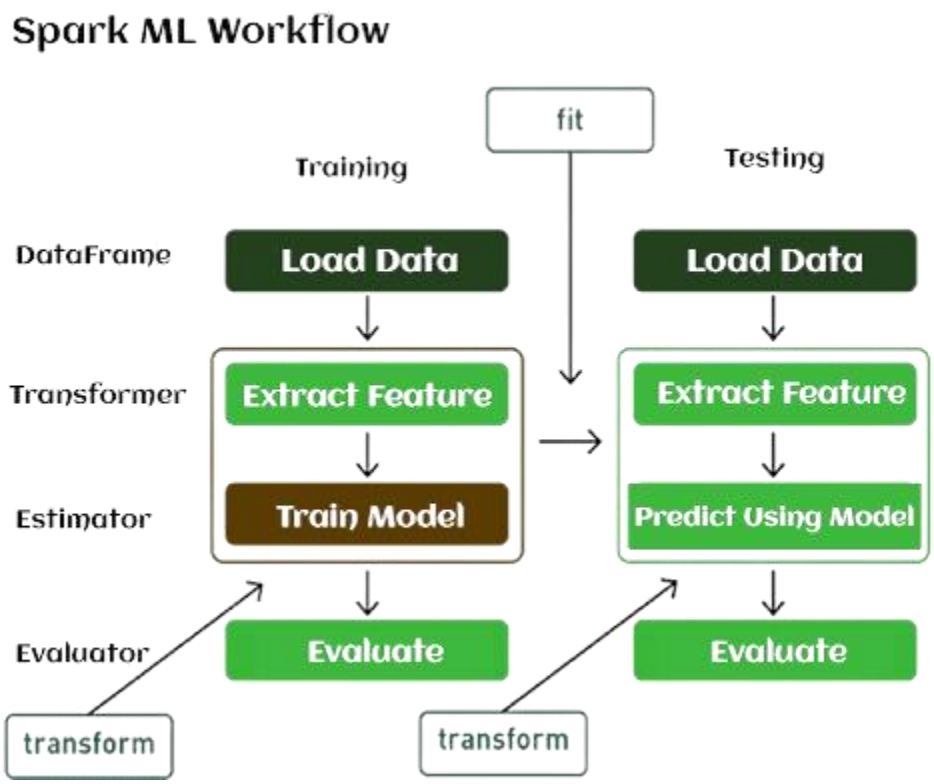
INTRODUCTION

Machine Learning pipeline refers to the complete workflow and processes of building and deploying a machine learning model. It automates and standardizes the workflow involved in creating a machine-learning model.

WHAT IS MACHINE LEARNING PIPELINE?

A Machine Learning pipeline is a process of automating the workflow of a complete machine learning task. It can be done by enabling a sequence of data to be transformed and correlated together in a model that can be analyzed to get the output. A typical pipeline includes raw data input, features, outputs, model parameters, ML models, and Predictions. Moreover, an ML Pipeline contains multiple sequential steps that perform everything ranging from data extraction and pre-processing to model training and deployment in Machine learning in a modular approach. It means that ***in the pipeline,*** **each step is designed as an independent module, and all these modules are tied together to get the final result.**

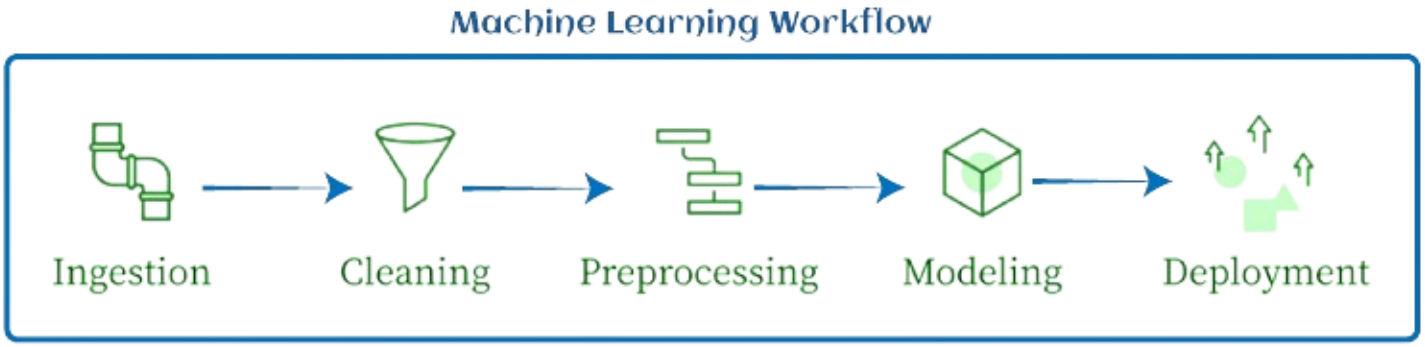
The ML pipeline is a high-level API for MLlib within the "spark.ml" package. A typical pipeline contains various stages. However, there are two main pipeline stages:



1. **Transformer:** It takes a dataset as an input and creates an augmented dataset as output. For example, A tokenizer works as Transformer, which takes a text dataset, and transforms it into tokenized words.
2. **Estimator:** An estimator is an algorithm that fits on the input dataset to generate a model, which is a transformer. For example, regression is an Estimator that trains on a dataset with labels and features and produces a logistic regression model.

IMPORTANCE OF MACHINE LEARNING PIPELINE

To understand the importance of a Machine learning pipeline, let's first understand a typical workflow of an ML task:



A typical workflow consists of **Ingestion, Data cleaning, Data pre-processing, Modelling, and deployment.**

In ML workflow, all these steps are run together with the same script. It means the same script will be used to extract data, clean data, model, and deploy. However, it may generate issues while trying to scale an ML model. These issues involve:

- If we need to deploy multiple versions of the same model, we need to run the complete workflow cycle multiple times, even when the very first step, i.e., ingestion and preparation, are exactly similar in each model.
- If we want to expand our model, we need to copy and paste the code from the beginning of the process, which is an inefficient and bad way of software development.
- If we want to change the configuration of any part of the workflow, we need to do it manually, which is a much more time-consuming process.

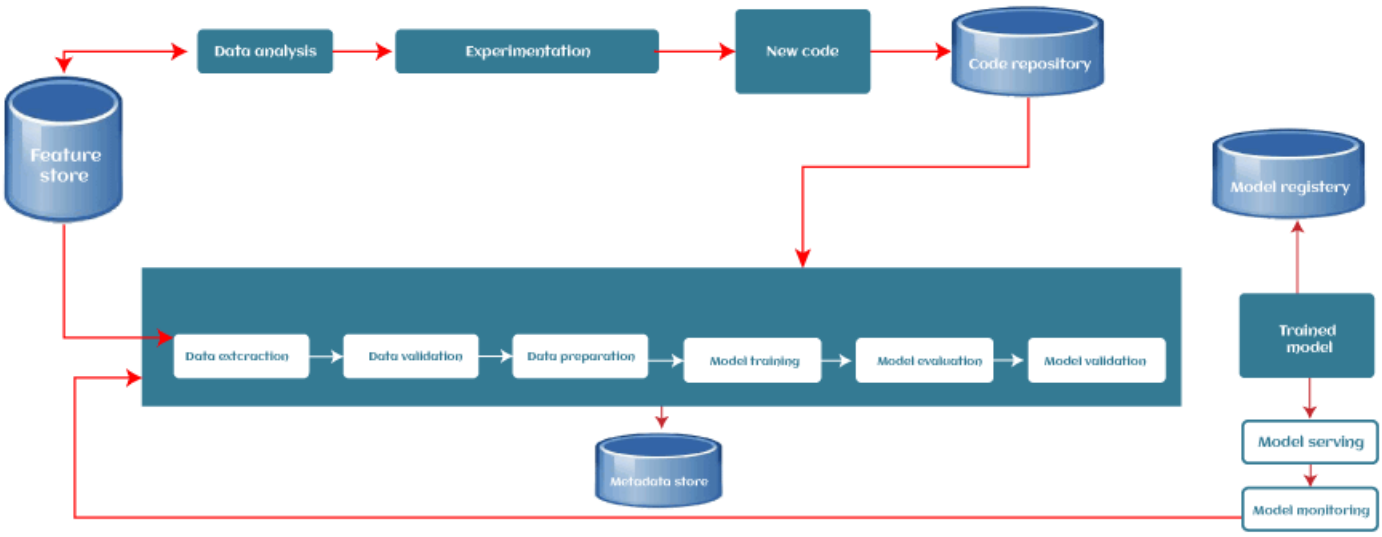
For solving all the above problems, we can use a Machine learning pipeline. With the ML pipeline, each part of the workflow acts as an **independent module**. So, whenever we need to change any part, we can choose that specific module and use that as per our requirement.

We can understand it with an example. Building any ML model requires a huge amount of data to train the model. As data is collected from different resources, it is necessary to clean and pre-process the data, which is one of the crucial steps of an ML project. However, whenever a new dataset is included, we need to perform the same pre-processing step before using it for training, and it becomes a time-consuming and complex process for ML professionals.

To solve such issues, ML pipelines can be used, which can remember and automate the complete pre-processing steps in the same order.

MACHINE LEARNING PIPELINE STEPS

On the basis of the use cases of the ML model and the requirement of the organization, each machine learning pipeline may be different to some extent. However, each pipeline follows/works upon the general workflow of Machine learning, or there are some common stages that each ML pipeline includes. Each stage of the pipeline takes the output from its preceding stage, which acts as the input for that particular stage. A typical ML pipeline includes the following stages:



1.Data Ingestion

Each ML pipeline starts with the Data ingestion step. In this step, the data is processed into a well-organized format, which could be suitable to apply for further steps. This step does not perform any feature engineering; rather, this may perform the versioning of the input data.

2. Data Validation

The next step is data validation, which is required to perform before training a new model. Data validation focuses on statistics of the new data, e.g., range, number of categories, distribution of categories, etc. In this step, data scientists can detect if any anomaly present in the data. There are various data validation tools that enable us to compare different datasets to detect anomalies.

3. Data Pre-processing

Data pre-processing is one of the most crucial steps for each ML lifecycle as well as the pipeline. We cannot directly input the collected data to train the model without pr-processing it, as it may generate an abrupt result.

The pre-processing step involves preparing the raw data and making it suitable for the ML model. The process includes different sub-steps, such as Data cleaning, feature scaling, etc. The product or output of the data pre-processing step becomes the final dataset that can be used for model training and testing. There are different tools in ML for data pre-processing that can range from simple Python scripts to graph models.

4. Model Training & Tuning

The model training step is the core of each ML pipeline. In this step, the model is trained to take the input (pre-processed dataset) and predicts an output with the highest possible accuracy.

However, there could be some difficulties with larger models or with large training data sets. So, for this, efficient distribution of the model training or model tuning is required.

This issue of the model training stage can be solved with pipelines as they are scalable, and a large number of models can be processed concurrently.

5. Model Analysis

After model training, we need to determine the optimal set of parameters by using the loss of accuracy metrics. Apart from this, an in-depth analysis of the model's performance is crucial for the final version of the model. The in-depth analysis includes calculating other metrics such as precision, recall, AUC, etc. This will also help us in determining the dependency of the model on features used in training and explore how the model's predictions would change if we altered the features of a single training example.

6. Model Versioning

The model versioning step keeps track of which model, set of hyperparameters, and datasets have been selected as the next version to be deployed. For various situations, there could occur a significant difference in model performance just by applying more/better training data and without changing any model parameter. Hence, it is important to document all inputs into a new model version and track them.

7. Model Deployment

After training and analyzing the model, it's time to deploy the model. An ML model can be deployed in three ways, which are:

- Using the Model server,
- In a Browser
- On Edge device

However, the common way to deploy the model is using a model server. Model servers allow to host multiple versions simultaneously, which helps to run A/B tests on models and can provide valuable feedback for model improvement.

8. Feedback Loop

Each pipeline forms a closed-loop to provide feedback. With this close loop, data scientists can determine the effectiveness and performance of the deployed models. This step could be automated or manual depending on the requirement. **Except for the two manual review steps (the model analysis and the feedback step), we can automate the entire pipeline.**

Benefits of having an end-to-end Machine-Learning Pipeline

1. **Ensure reproducibility:** By running the channel repeatedly on similar inputs, consistent outputs can be obtained, ensuring reproducibility and reliability in machine learning.
2. **Simplify workflow:** The pipeline automates multiple steps in the machine learning workflow. This reduces the need for manual intervention from the data science team, making the process more efficient and streamlined.

3. **Accelerate deployment:** The pipeline helps reduce the time data and models take to the production phase. This enables faster development of Machine learning solutions and quicker integration into real-world applications.
4. **Enable focus on innovation:** With modular components and automation in place, the pipeline frees the data science team to focus more on developing new solutions rather than spending excessive time maintaining existing ones.
5. **Facilitate reusability:** The pipeline allows for easy reusability of components in the machine learning workflow. Specific steps can be reused to create and deploy end-to-end solutions seamlessly integrated with systems without starting from scratch each time.

Best Practices for Building ML Pipelines:

Building end-to-end ML pipelines requires careful planning, design, and implementation to ensure efficiency, scalability, and maintainability. Here are some best practices to consider:

Modular Design: Break down the pipeline into modular components, each responsible for a specific task such as data preprocessing, feature engineering, model training, and deployment. This modular design promotes code reusability, scalability, and ease of maintenance.

Automation: Automate repetitive tasks such as data ingestion, preprocessing, and model training using workflow management tools like Apache Airflow, Luigi, or Kubeflow. Automation reduces manual errors, accelerates development cycles, and enhances productivity.

Scalability: Design the pipeline to handle large volumes of data efficiently by leveraging distributed computing frameworks such as Apache Spark or Dask. Scalable pipelines can accommodate growing datasets and computational demands, ensuring robust performance in production environments.

Version Control: Implement version control for pipeline components, datasets, and model artifacts using platforms like Git or DVC (Data Version Control). Version control facilitates collaboration, reproducibility, and tracking of changes throughout the ML lifecycle.

Monitoring and Logging: Integrate logging and monitoring mechanisms to track pipeline performance, detect anomalies, and troubleshoot issues in real-time. Monitoring metrics such as latency, throughput, and error rates enable proactive maintenance and continuous improvement of the pipeline.

Security and Compliance: Ensure data security, privacy, and regulatory compliance throughout the pipeline by implementing encryption, access controls, and compliance measures such as GDPR or HIPAA. Secure pipelines protect sensitive information and mitigate risks associated with data breaches or unauthorized access.

ML PIPELINE TOOLS

There are different tools in Machine learning for building a Pipeline. Some are given below along with their usage:

Steps while building the pipeline	Tools
Obtaining the Data	Managing the Database - PostgreSQL, MongoDB, DynamoDB, MySQL. Distributed Storage - Apache Hadoop, Apache Spark/Apache Flink.
Scrubbing / Cleaning the Data	Scripting Language - SAS, Python, and R. Processing in a Distributed manner - MapReduce/ Spark, Hadoop. Data Wrangling Tools - R, Python Pandas
Exploring / Visualizing the Data to find the patterns and trends	Python, R, MATLAB, and Weka.
Modeling the data to make the predictions	Machine Learning algorithms - Supervised, Unsupervised, Reinforcement, Semi-Supervised, and Semi-unsupervised learning. Important libraries - Python (Scikit learn) / R (CARET)
Interpreting the result	Data Visualization Tools -ggplot, Seaborn, D3.JS, Matplotlib, Tableau.

Conclusion:

Machine learning pipelines play a pivotal role in streamlining the end-to-end process of building, deploying, and maintaining ML models in real-world applications. By orchestrating tasks such as data preprocessing, feature engineering, model training, and deployment, ML pipelines enable organizations to leverage the power of AI for data-driven decision-making, automation, and innovation across industries.

As the demand for scalable, efficient, and reliable ML solutions continues to grow, the development of robust and flexible pipelines becomes increasingly imperative. By embracing best practices, emerging technologies, and collaborative methodologies, practitioners can unlock the full potential of machine learning pipelines and drive transformative impact in the era of AI-driven digital transformation.