The tutorial to build shared AI services

--Session 2

Suqiang Song (Jack)
Director & Chapter Leader of Data/AI Engineering @ Mastercard
jackssqcyy@gmail.com
https://www.linkedin.com/in/suqiang-song-72041716/

Agenda

Session 2: Feb. 1rd Friday 10am-12pm PT

Module 3: AI Engineering platform and AI Engineers (40 mins)

- Key factors to consider an AI Engineering platform
- architect a data pipeline framework
- Apache NiFi introduction
- Traditional AI Tribe and its challenges
- knowledges and skills are required for AI Engineer
- Growing path for an AI Engineer

Module 4: Benchmark between Spark Machine learning and Deep learning + Code Lab 2 (30 mins)

- Traditional Collaborative Filtering approach with Spark Mllib ALS (Scala)
- Build an NCF deep learning approach with Intel Analytic Zoo on Spark (Scala)

Live Demo (40 mins)

- Build an end to end AI Pipeline with Kafka, NiFi, Spark Streaming and Keras on Spark

Q & A (10 mins)

Course Prerequisites

- Install Docker at your local laptop
- Download two Docker images from shared drive URL kafka.tar and demo-whole.tar and also demo_pipeline.xml

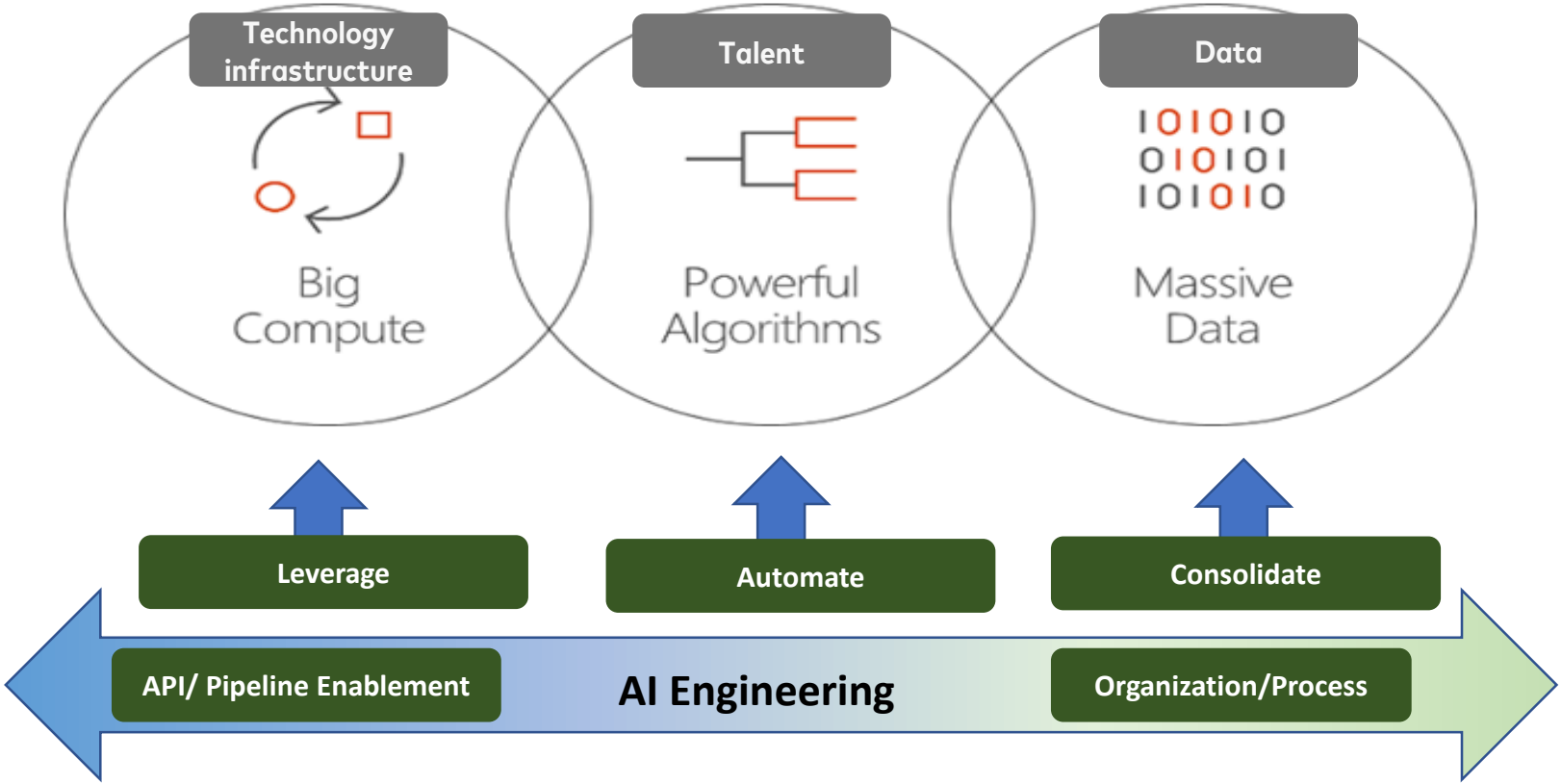
  https://1drv.ms/f/s!AsXKHMXBWUIBiBpaYk9FFjdoUifg

  passcode : jack
- Load images to your Docker environment

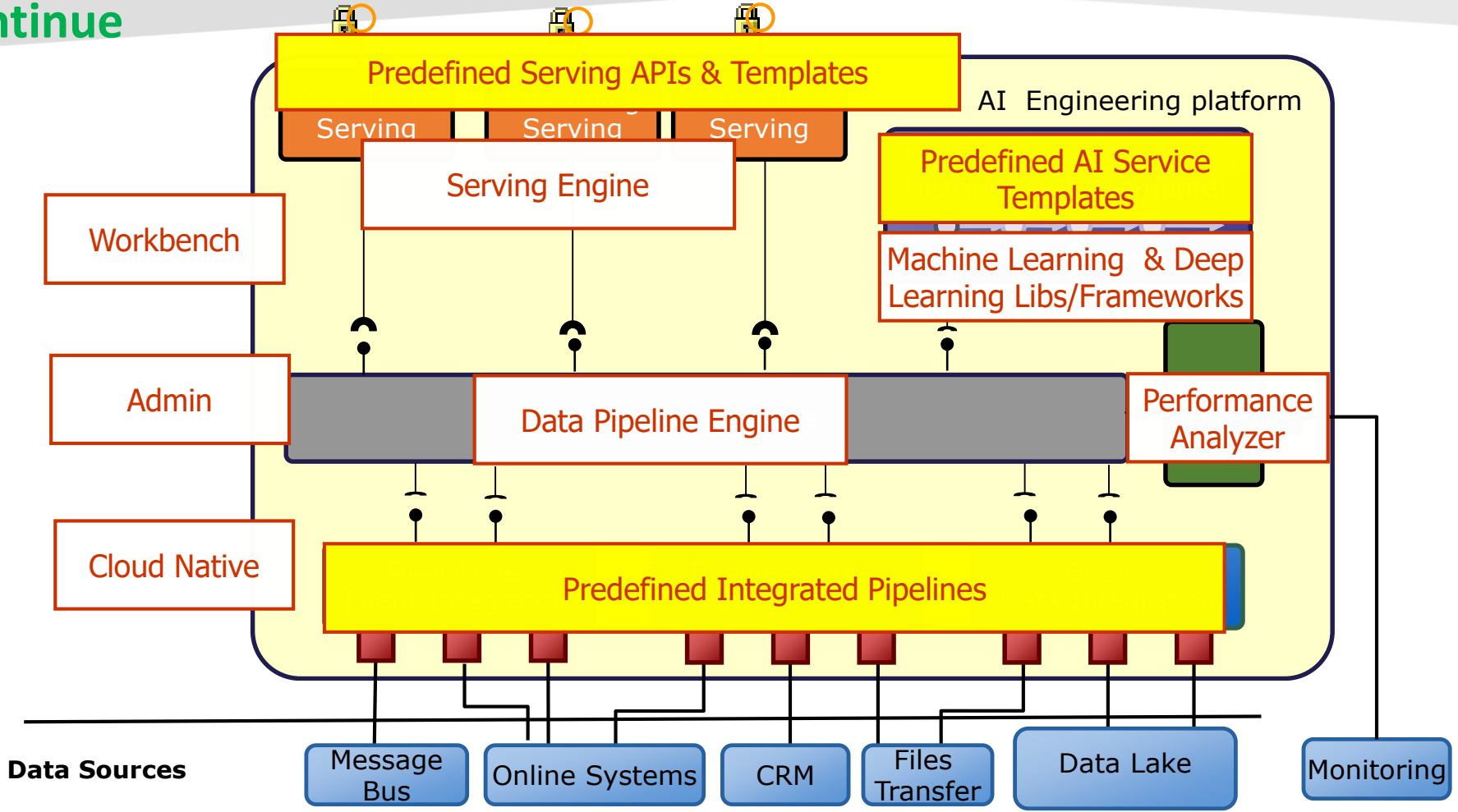
```
$ docker load -i kafka.tar
$ docker load -i demo-whole.tar
```

Module 3

Key factors to consider an AI Engineering platform

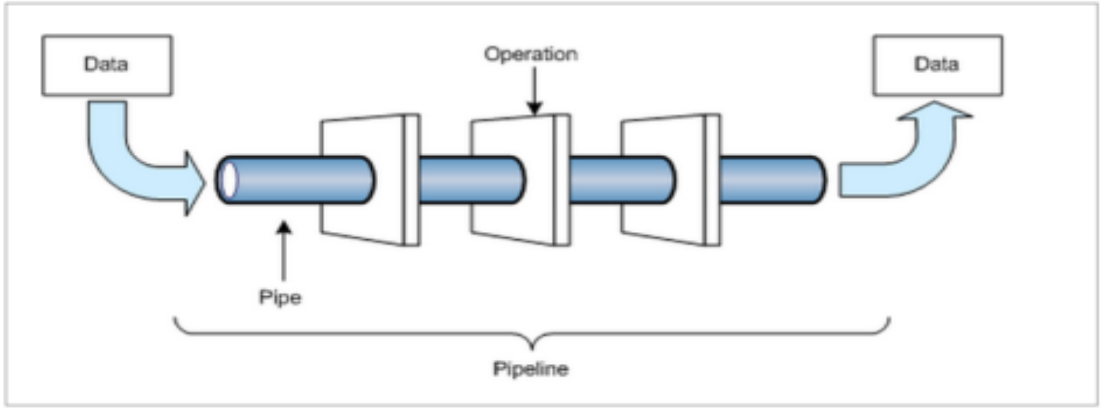


Continue

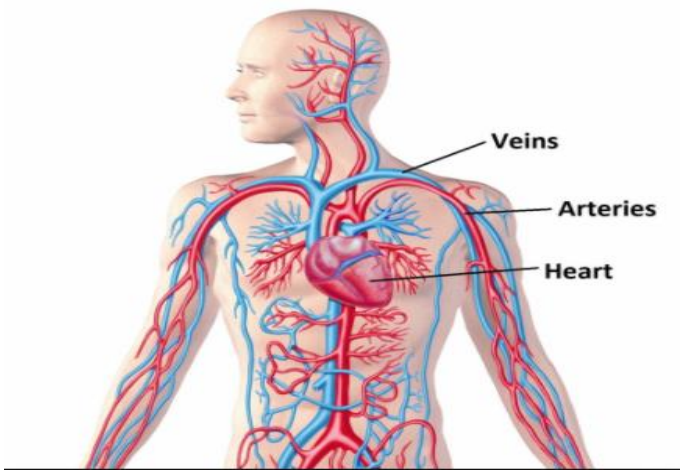

Predefined Serving APIs & Templates
Serving
Serving
Serving
AI Engineering platform
Serving Engine
Predefined AI Service Templates
Workbench
Machine Learning & Deep Learning Libs/Frameworks
Admin
Data Pipeline Engine
Performance Analyzer
Cloud Native
Predefined Integrated Pipelines
Data Sources
Message Bus
Online Systems
CRM
Files Transfer
Data Lake
Monitoring

Data Flow Pipeline

- Flow-based ”programming”
- Source –Channel-Sink “structure”
- Ingest Data from various sources and Transform data to various destinations
- Extract – Transform – Load
- High-Throughput, straight-through data flows
- Data Governance
- Combine Batch and Stream-Processing
- Visual coding with flow editor
- Event Processing (ESP & CEP)



Source: Confluent



Architect a data pipeline framework

What is the DFX ?

Along with functional requirements, there are various quality attributes.

The difference in these attributes can make the product very different.

Such as Tesla and Leaf

DFX is Design For Quality Attributes

The X(quality attributes) for data pipeline framework includes

- Clustering
- High Availability & Recovery
- Delivery Guarantee
- Data Buffering ,Flow Control and Back Pressure
- Data Governance
- Usability
- Extensibility
- Multi-Tenancy
- Version Control & Deployment
- Security
- Monitoring & Diagnostic Capabilities
- Integration Capabilities
- Cloud Native
- Performance , Latency and Throughputs

Example : High Availability and Recovery

- **High availability**
  - **Pipeline** level : Each step or processor at flow that is likely to encounter failures will have a "failure" routing relationship
  - **Pipeline** Failure is handled by looping that failure relationship back on the same step or to new steps
  - **Node** level failover will depends on a "cluster coordinator" and a "primary node" elected
  - **Pipeline** failover between nodes ?

- **Recovery**
  - Replay : Content repository should be designed to act as a rolling buffer of history which supports replay every well
  - Data Recovery after failover , the eventual consistency
  - Breakpoint resume : Last-saved offset , how you resume the pipeline from the broken pieces after fixed

Example :
Data Buffering ,Flow Control and Back Pressure

- **Buffering with Prioritization**
  - Configure a prioritizer per connection, such as FirstInFristOut , NewestFirst,OldestFirst etc..
  - Determine what is important for your data – time based, arrival order,
    importance of a data set
  - Funnel many connections down to a single connection to prioritize across
    data sets
  - Develop your own prioritizer if needed

- **Flow Control & Back-Pressure**
  - Configure back-pressure such as expiration, threshold for per connection
  - Based on number of flows or total size of flows
  - Upstream processor no longer scheduled to run until below threshold

Example : Security

Yes , you don't want your CEO to be testified before Congress ☺

- **Control Plane**
  - Pluggable authentication :2-Way SSL, LDAP, Kerberos
  - File-based authority provider out of the box
  - Multiple roles to defines access controls
- **Data Plane**
  - Optional 2-Way SSL between cluster nodes
  - Optional 2-Way SSL on Site-To-Site (or Edge-to-Edge) connections
  - Encryption/Decryption of data through processors
- **Data privacy and compliance**
  - PCI/PII compliance
  - GDPR (General Data Protection Regulation)

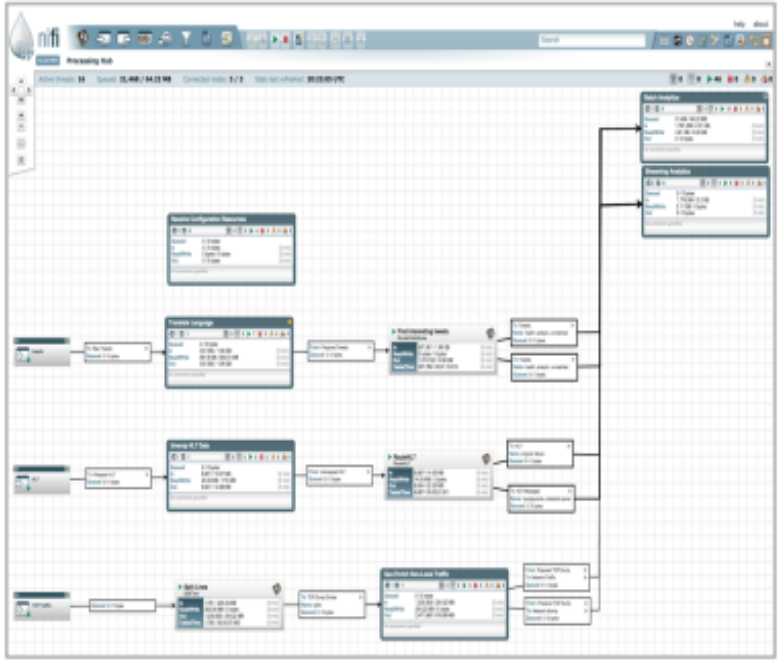
Example : Multi-Tenancy

Ability for multiple groups of entities (people or systems) to command, control, and observe state of different parts of the dataflow

- **Multi-tenant Authorization**
  - Enable a self-service model for dataflow management, allowing each team or organization to manage flows with a full awareness of the rest of the flow, to which they do not have access.
- **Multi-tenant isolation and Separated SLA/QoS**
  - Data is absolutely critical and it is loss intolerant
  - Enables the fine-grained flow specific configuration to each tenant
  - Data Buffering ,Flow Control and Back Pressure should be considered at tenant level
- **Multi-tenant isolated resources management**
  - Integrate with 3rd popular resources management framework such as Yarn
  - Split up the functionalities of resource management and job scheduling/monitoring into separate daemons

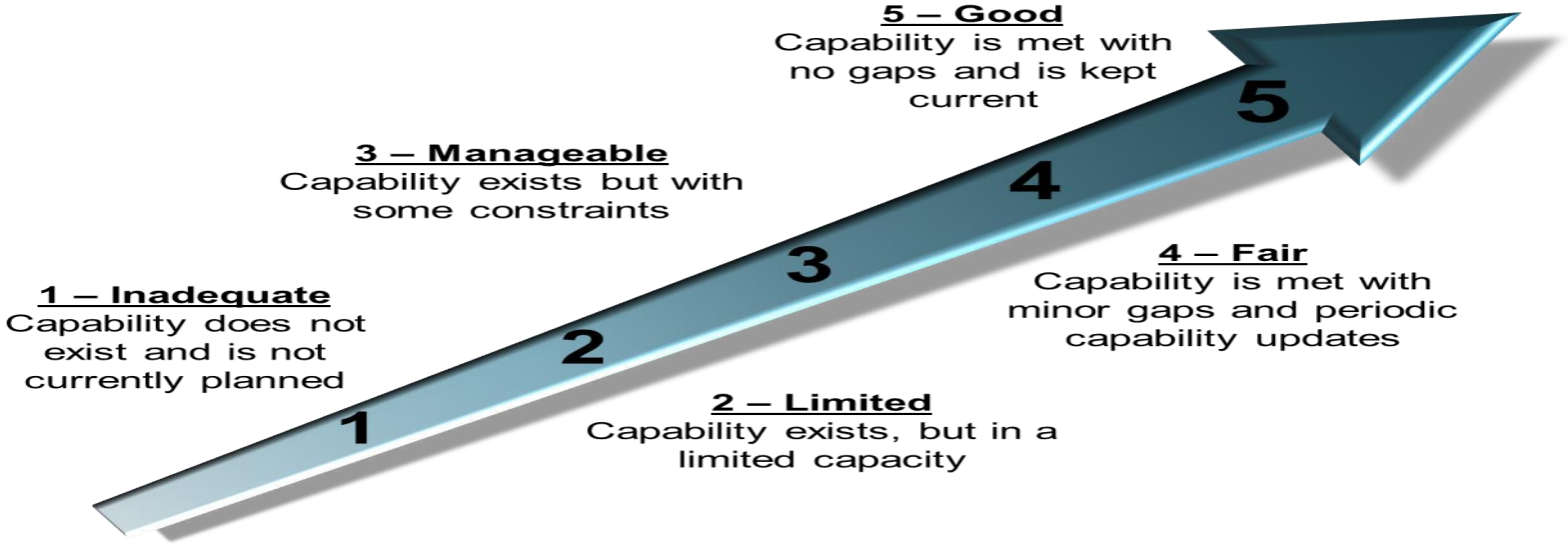
Apache NiFi

- Powerful and reliable **Distributed Real-time computation engine**
- Directed graphs of **data routing and transformation**
- Web-based **User Interface** for creating, monitoring, & controlling data flows
- Highly configurable - **modify data flow at runtime**, dynamically prioritize data
- **Data Provenance** tracks data through entire system
- **Easily extensible** through development of custom components



[1] https://nifi.apache.org/

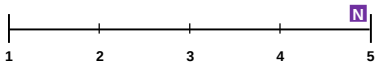
Technology Assessment Score Definition



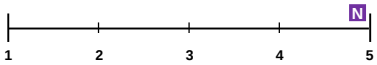
Assessment Score Ratings

N NiFi

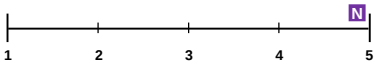
Clustering



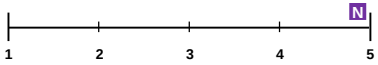
High Availability and Recovery



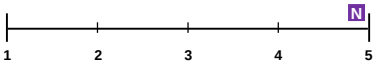
Delivery Guarantee



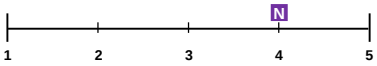
Data Buffering ,Flow Control and Back Pressure



Data Governance

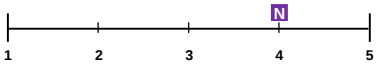


Usability

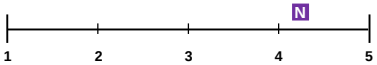


Extensibility

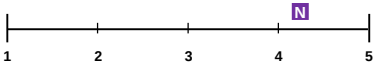
Multi-Tenancy



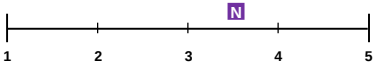
Version Control & Deployment



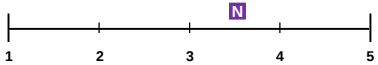
Authentication & Authorization



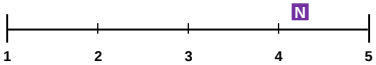
Encryption and decryption



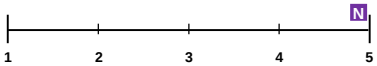
Monitoring & Diagnostic



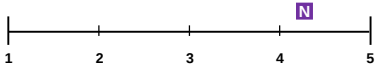
Integration capabilities



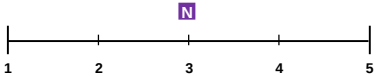
Cloud Native



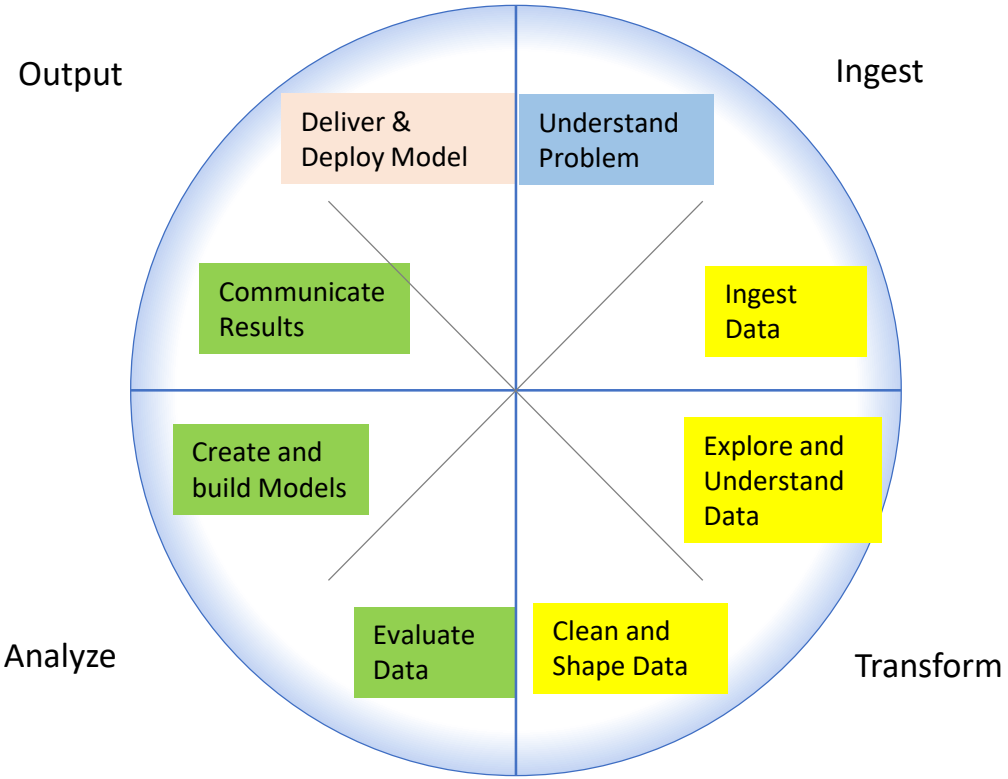
Performance, Latency and Throughputs (Real Time & Streaming)

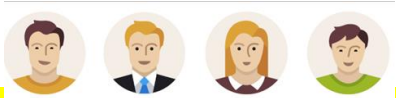


Performance, Latency and Throughputs (Batch files / DB actions)



Traditional AI Tribe





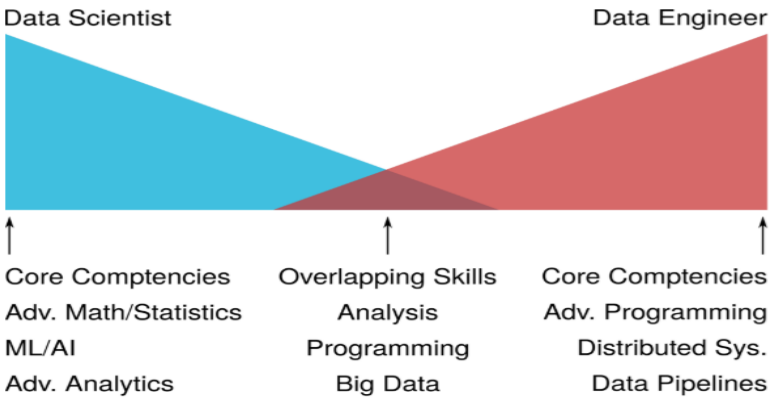
Data Engineer
Architect how data is organized & ensure operability

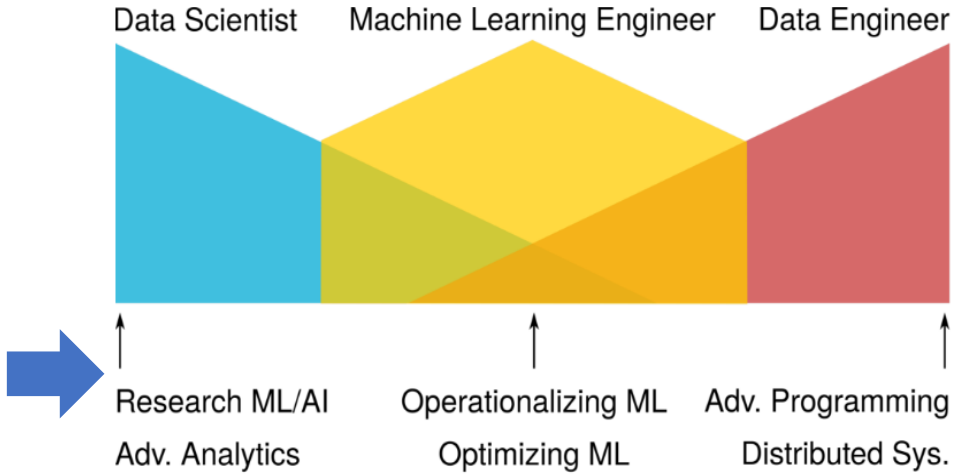
Data Scientist
Deep analytics and modeling for hidden insights

Business Analyst
Work with data to apply insights to business strategy

App Developer
Integrates data & insights with existing or new applications

Trends





About You

You have significant experience architecting, designing, and building scalable cloud-native software platforms that embed ML/AI models, and have hands-on experience with operationalizing and optimizing ML based end-to-end solutions. You have a good understanding of mainstream machine learning algorithms and experience using popular frameworks to train, test, and deploy machine learning models. You're are an excellent communicator and are able to convey complex ideas to different audiences. Above all, you're excited about working with data, and you know how to realize your ideas with code.

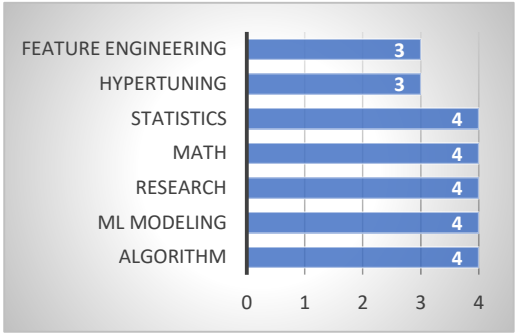
https://vmware.wd1.myworkdayjobs.com/VMware/job/USA-California-Palo-Alto/Staff-Machine-Learning-Engineer_R1813174?from=timeline

Responsibilities

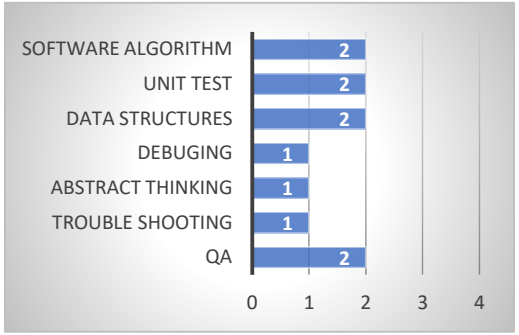
- Architect, design and implement highly scalable ML pipelines (data ingestion, feature extraction, training, testing and validation, inference, and continual learning) in production environments using cloud-native methodologies.
- Stay up to date with state-of-the-art technologies for large scale data processing and machine learning; work with devops team to choose the right tools and operational architecture.
- Have a thorough understanding of the larger ecosystem that encompasses the ML pipeline and implement appropriate interfaces for upstream and downstream services to use.
- Build necessary frameworks and tools, such as logging and A/B testing, to monitor the performance of ML pipelines and models, and constantly look for ways in which the overall product performance can be improved.
- Work closely with data scientists during all stages of data science projects including data cleansing, verifying integrity of data, and developing ML algorithms to address specific use-cases.
- Build prototypes of systems to demonstrate feasibility of proposed ML algorithms.
- Monitor and analyze the effectiveness of new features after they are introduced in the product.
- Design and build abstractions to easily integrate ML models into production data pipelines; enforce software best practices so that the ML code is easy to debug and maintain.
- Work with teams across all functions including Data Science, Data Engineering, and Product Management.

Ratings for traditional data scientist

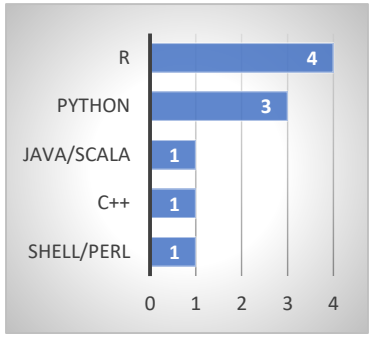
Data Mining

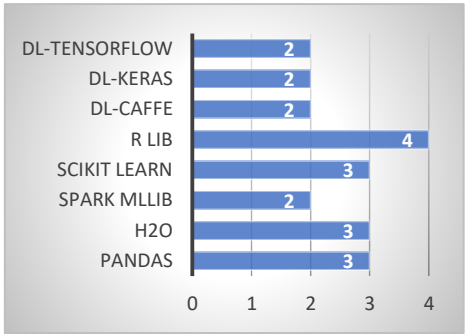


Programing / Coding

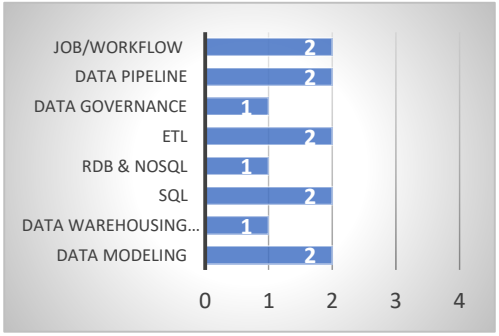


Languages

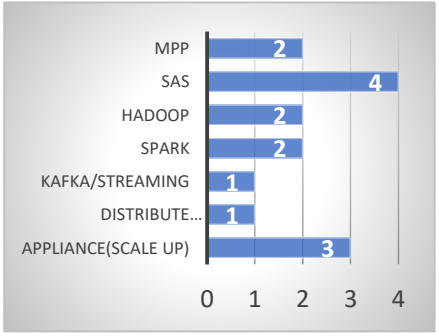




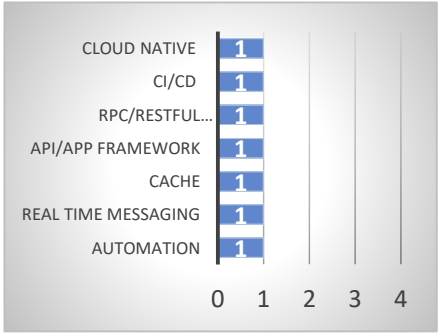
Data Engineering

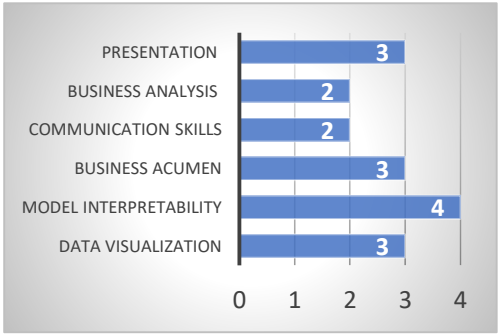


Big Data Stacks



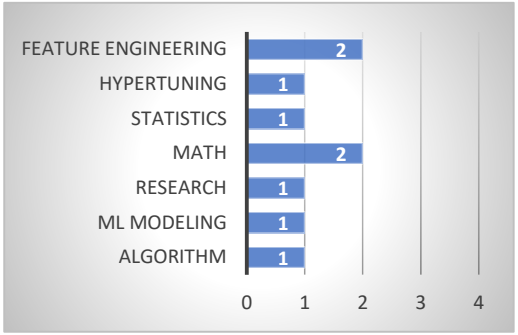
APIs /Services/App (Model Serving)



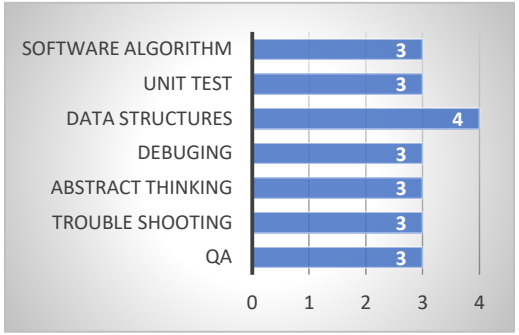


Ratings for traditional data engineer

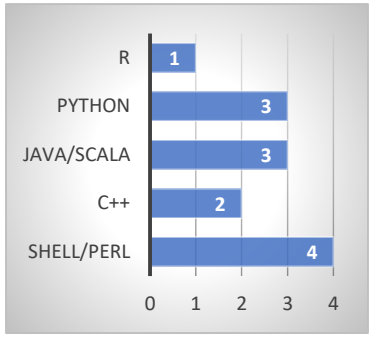
Data Mining

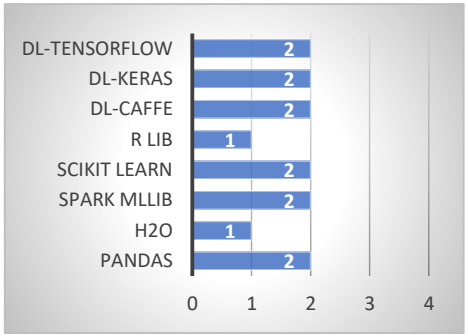


Programing / Coding

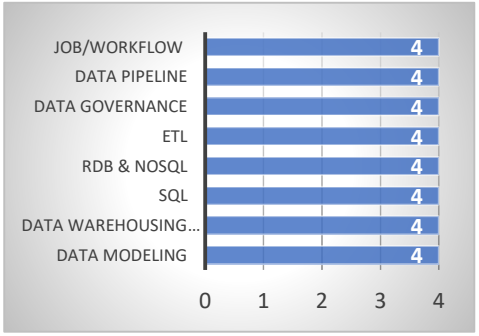


Languages

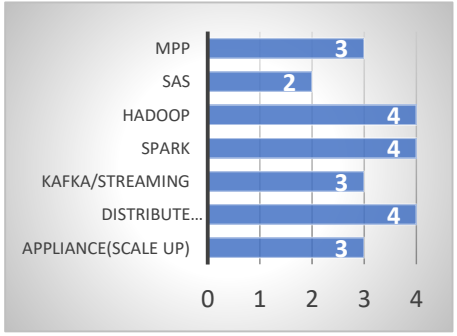




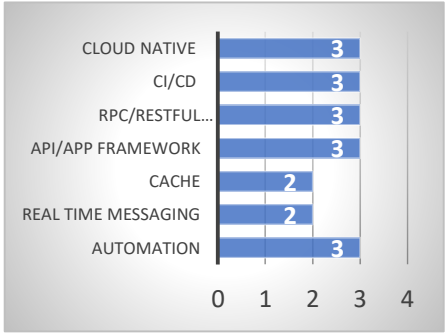
Data Engineering

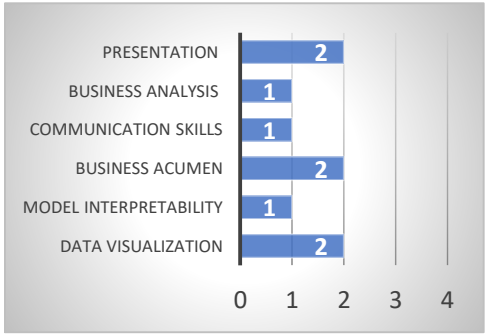


Big Data Stacks



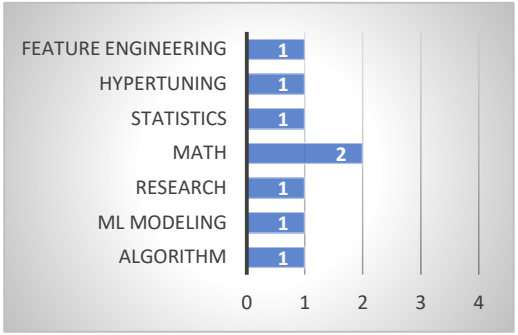
APIs /Services/App (Model Serving)



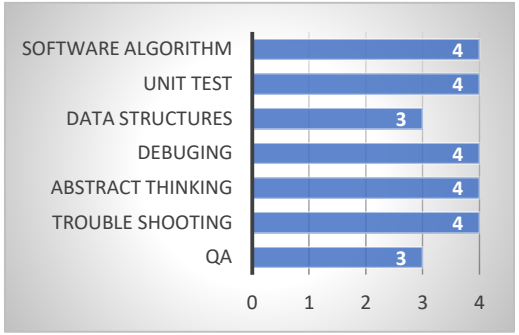


Ratings for traditional application developer

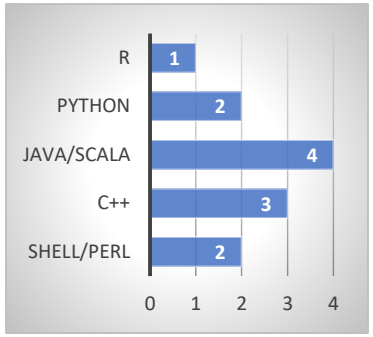
Data Mining



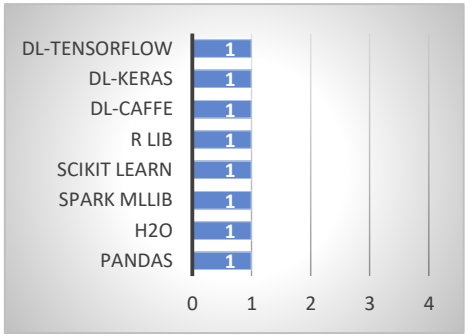
Programing / Coding



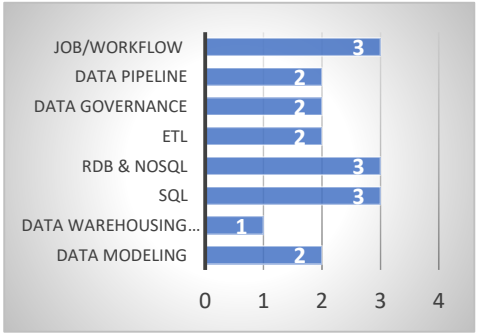
Languages



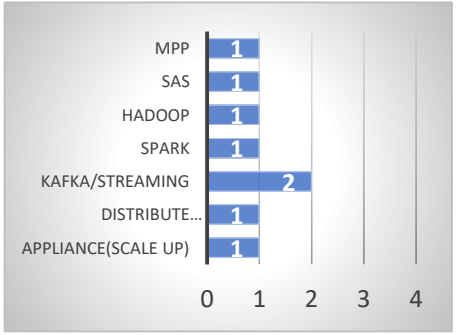
ML Framework



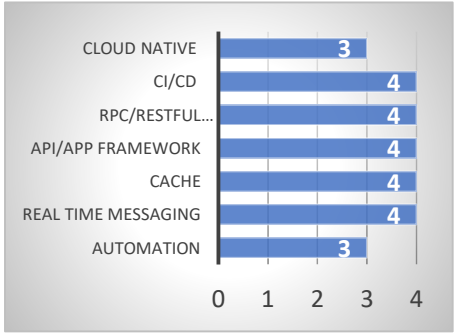
Data Engineering

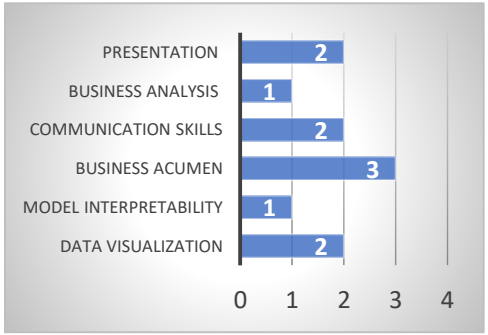


Big Data Stacks



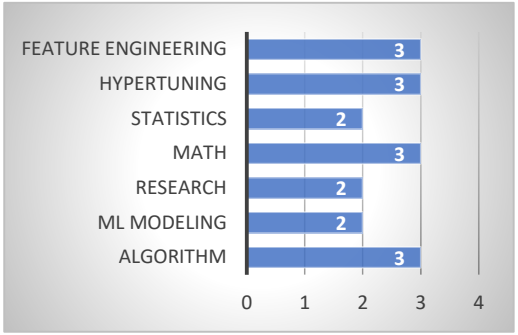
APIs /Services/App (Model Serving)



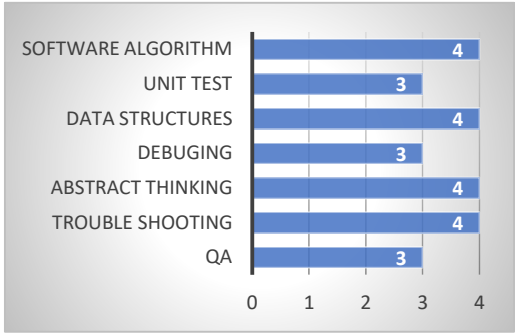


Ratings for modern AI engineer

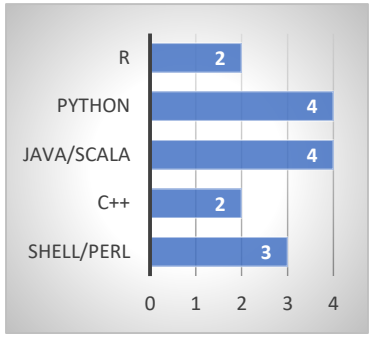
Data Mining

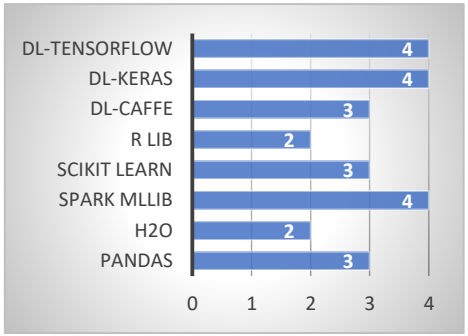


Programing / Coding

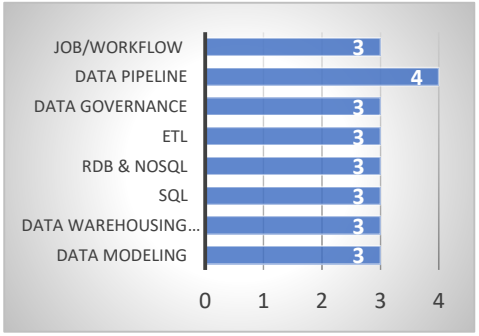


Languages

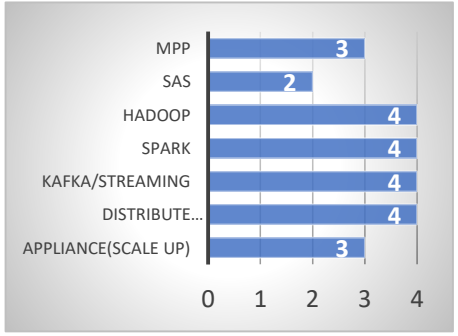




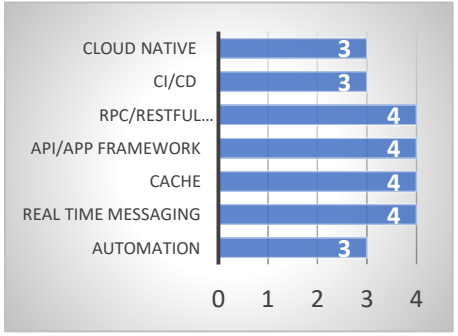
Data Engineering

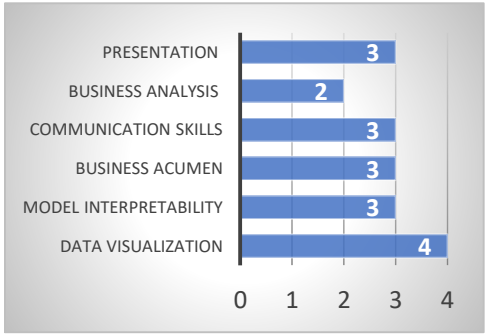


Big Data Stacks

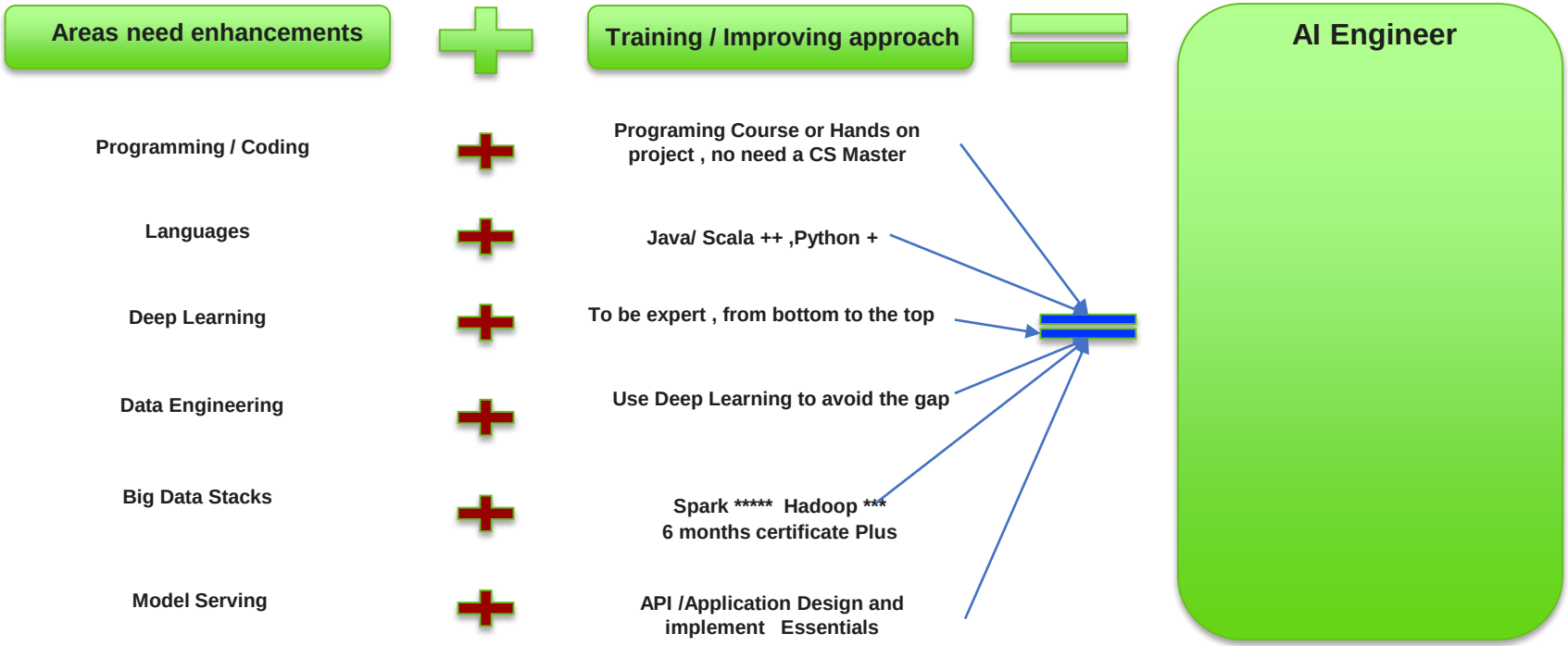


APIs /Services/App (Model Serving)

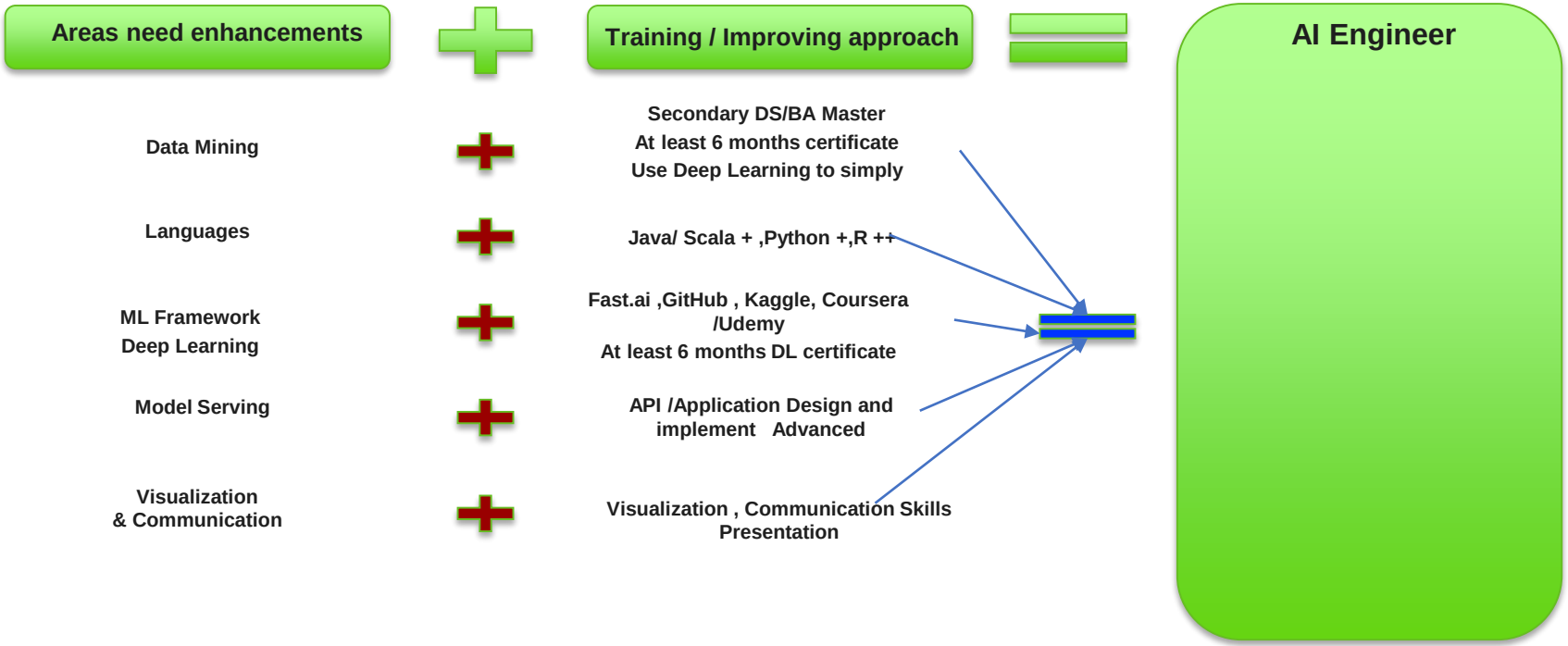




Growing path 1 : traditional data scientist - > AI Engineer



Growing path 2 : traditional data engineer - > AI Engineer

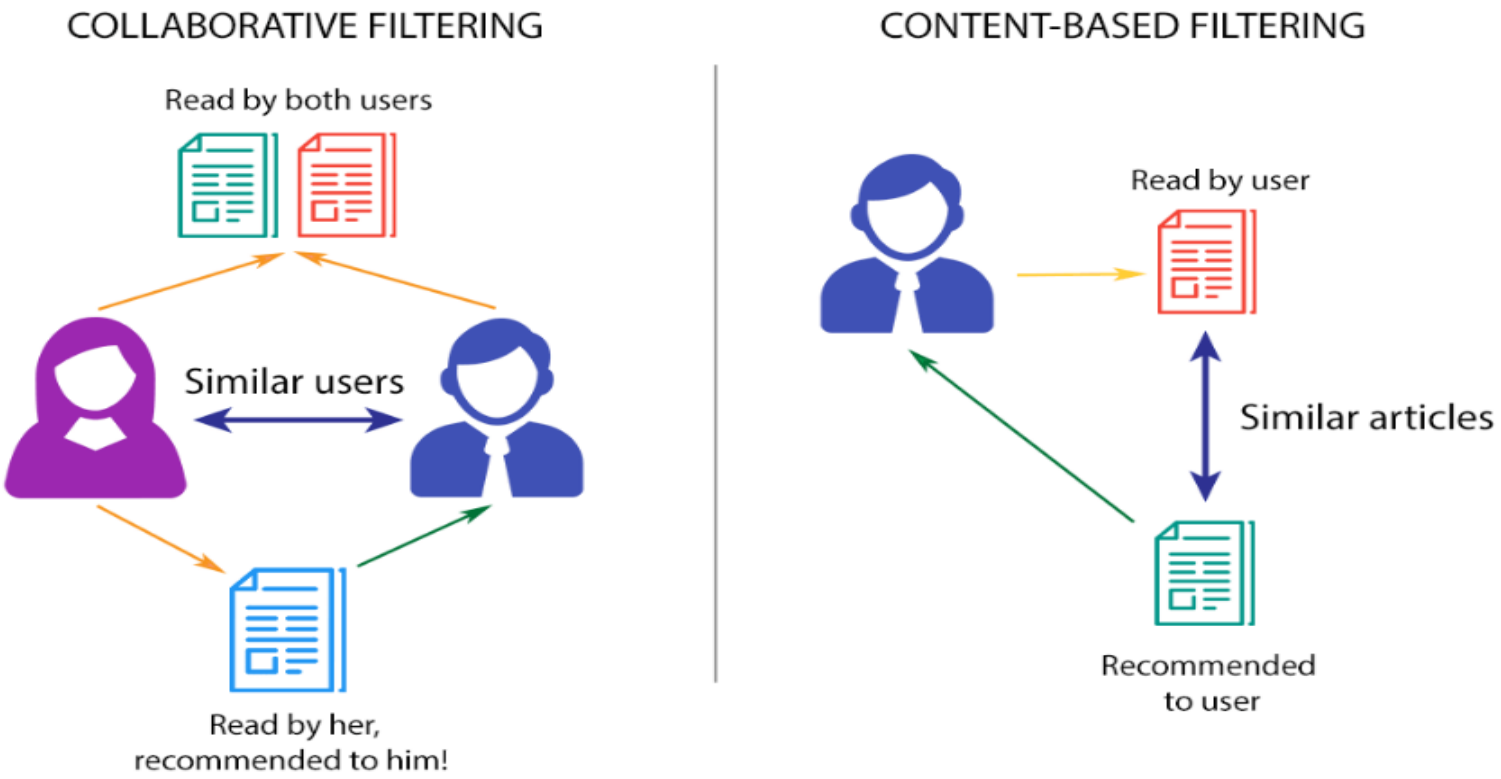


Growing path 3 : traditional application developer - > AI Engineer

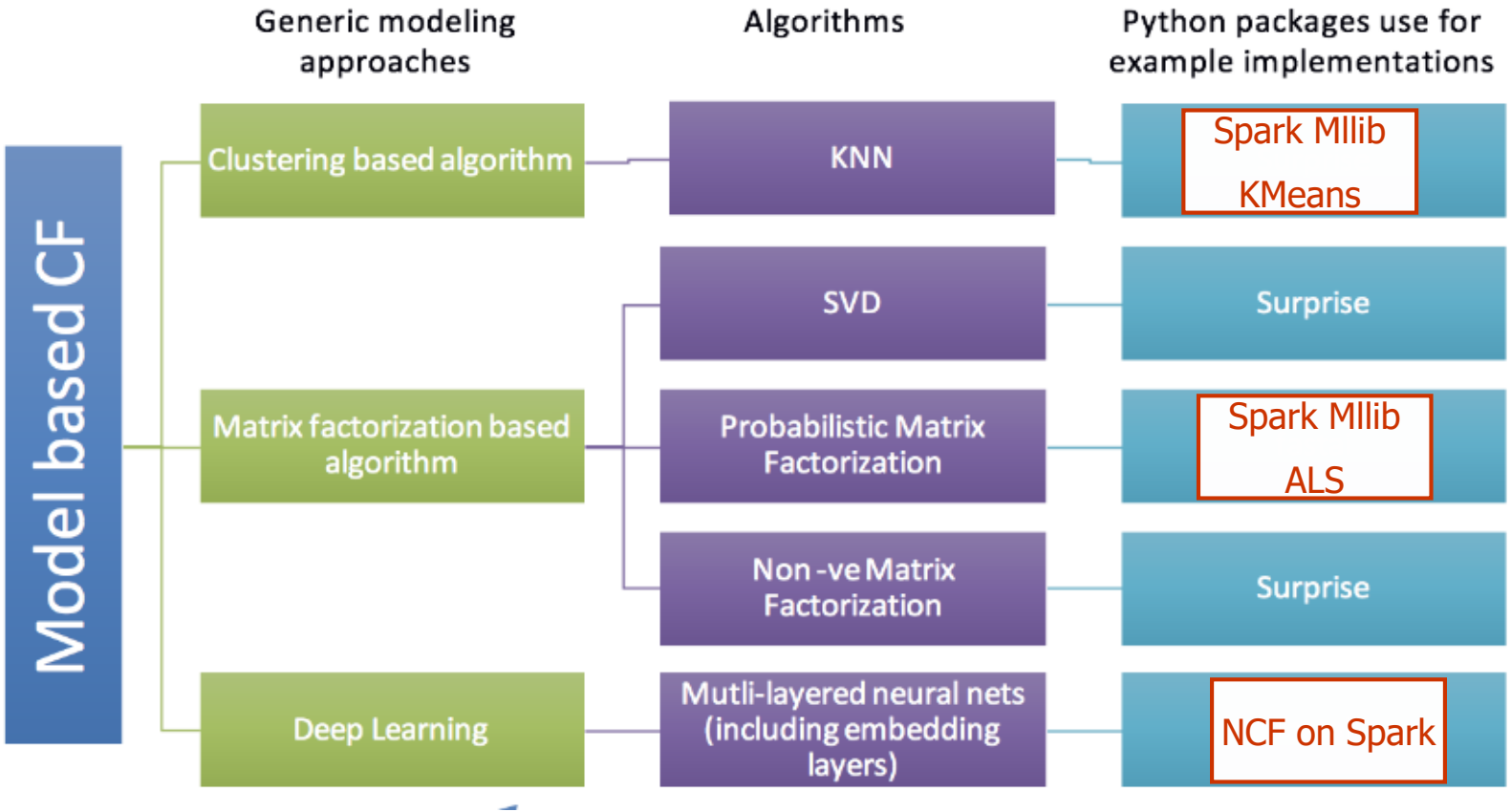
Module 4+ Code Lab2

https://github.com/jack1981/AaaSDemo

Collaborative Filtering -- concept



Collaborative Filtering -- Model selection

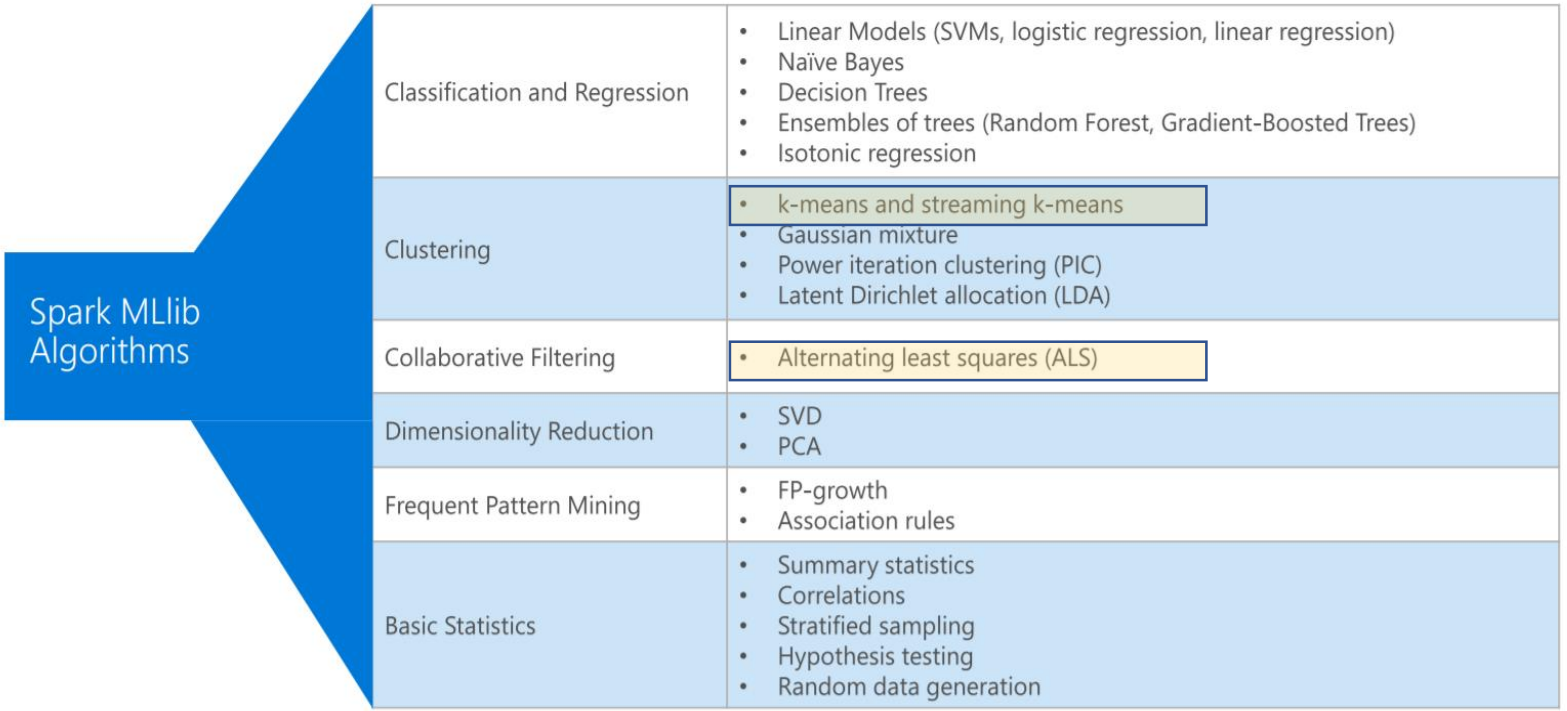


Spark Mllib

Enables Parallel, Distributed ML for large datasets on Spark Clusters

- Offers a set of parallelized machine learning algorithms for ML
- Supports Model Selection (hyper parameter tuning) using Cross Validation and Train-Validation Split.
- Supports Java, Scala or Python apps using Data Frame-based API

Spark Mllib Algorithms

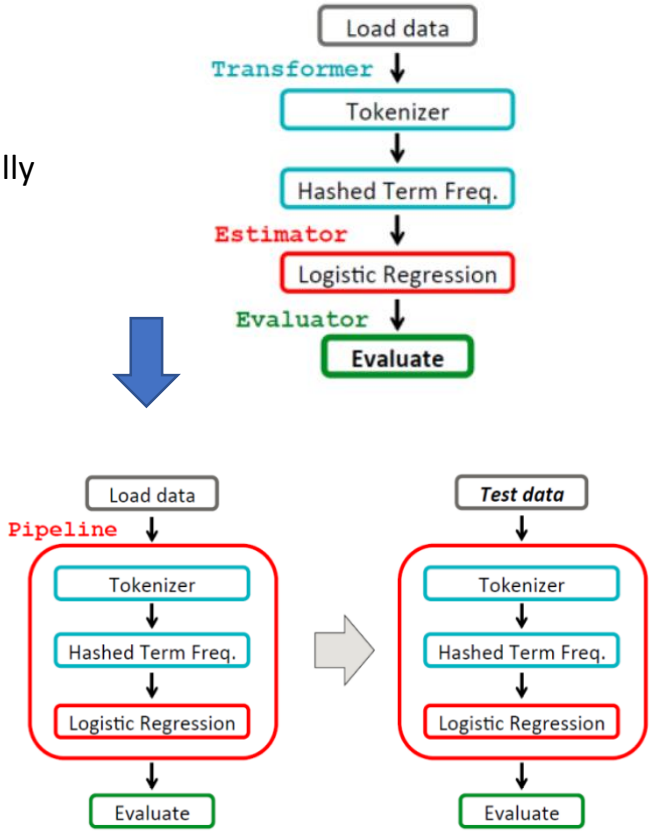


Classification and Regression	• Linear Models (SVMs, logistic regression, linear regression) • Naïve Bayes • Decision Trees • Ensembles of trees (Random Forest, Gradient-Boosted Trees) • Isotonic regression
Clustering	• k-means and streaming k-means • Gaussian mixture • Power iteration clustering (PIC) • Latent Dirichlet allocation (LDA)
Collaborative Filtering	• Alternating least squares (ALS)
Dimensionality Reduction	• SVD • PCA
Frequent Pattern Mining	• FP-growth • Association rules
Basic Statistics	• Summary statistics • Correlations • Stratified sampling • Hypothesis testing • Random data generation

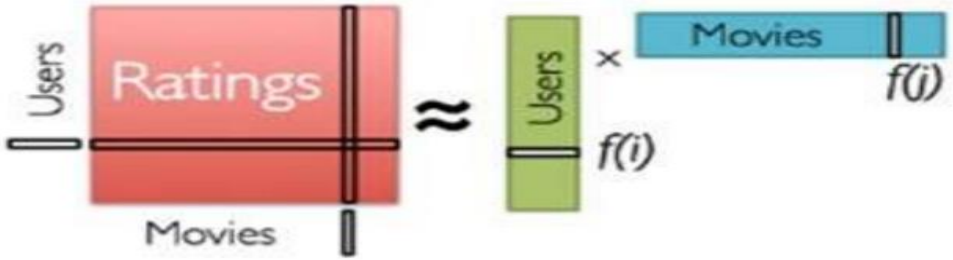
Spark Mllib ML Pipeline

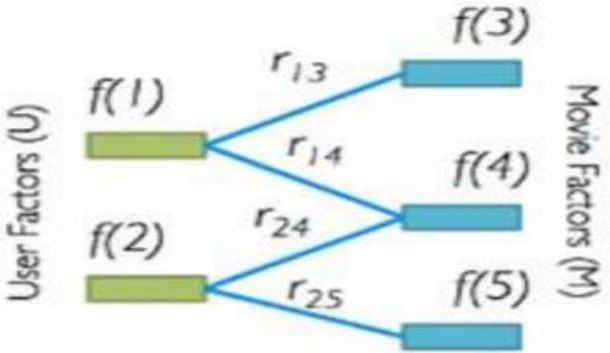
DataFrame: Spark ML uses DataFrame rather than regular RDD as they hold a variety of data types (e.g. feature vectors, true labels, and predictions).
Transformer: a transformer converts a DataFrame into another DataFrame usually by appending columns. (since Spark DataFrame is immutable, it actually creates a new DataFrame). The implement method for a transformer is "transform()".
Estimator: An Estimator is an algorithm which can be fit on a DataFrame to produce a Transformer. Implements method fit() taking a DataFrame and a model (also a transformer) as input.
Pipeline: Chains multiple Transformers and Estimators each as a stage to specify an ML workflow. These stages are run in order, and the input DataFrame is transformed as it passes through each stage.
Parameter: All Transformers and Estimators now share a common API for specifying parameters.
Evaluator: Evaluate model performance. The Evaluator can be

- RegressionEvaluator for regression problems,
- BinaryClassificationEvaluator for binary data, or
- MulticlassClassificationEvaluator for multiclass problems.

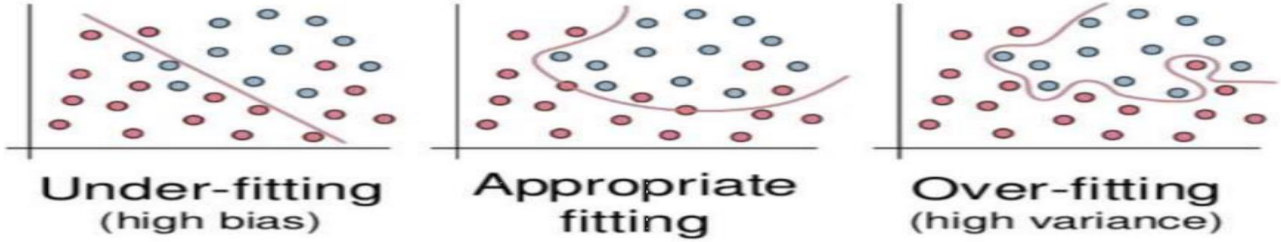


Alternating Least Squares (ALS) Spark ML

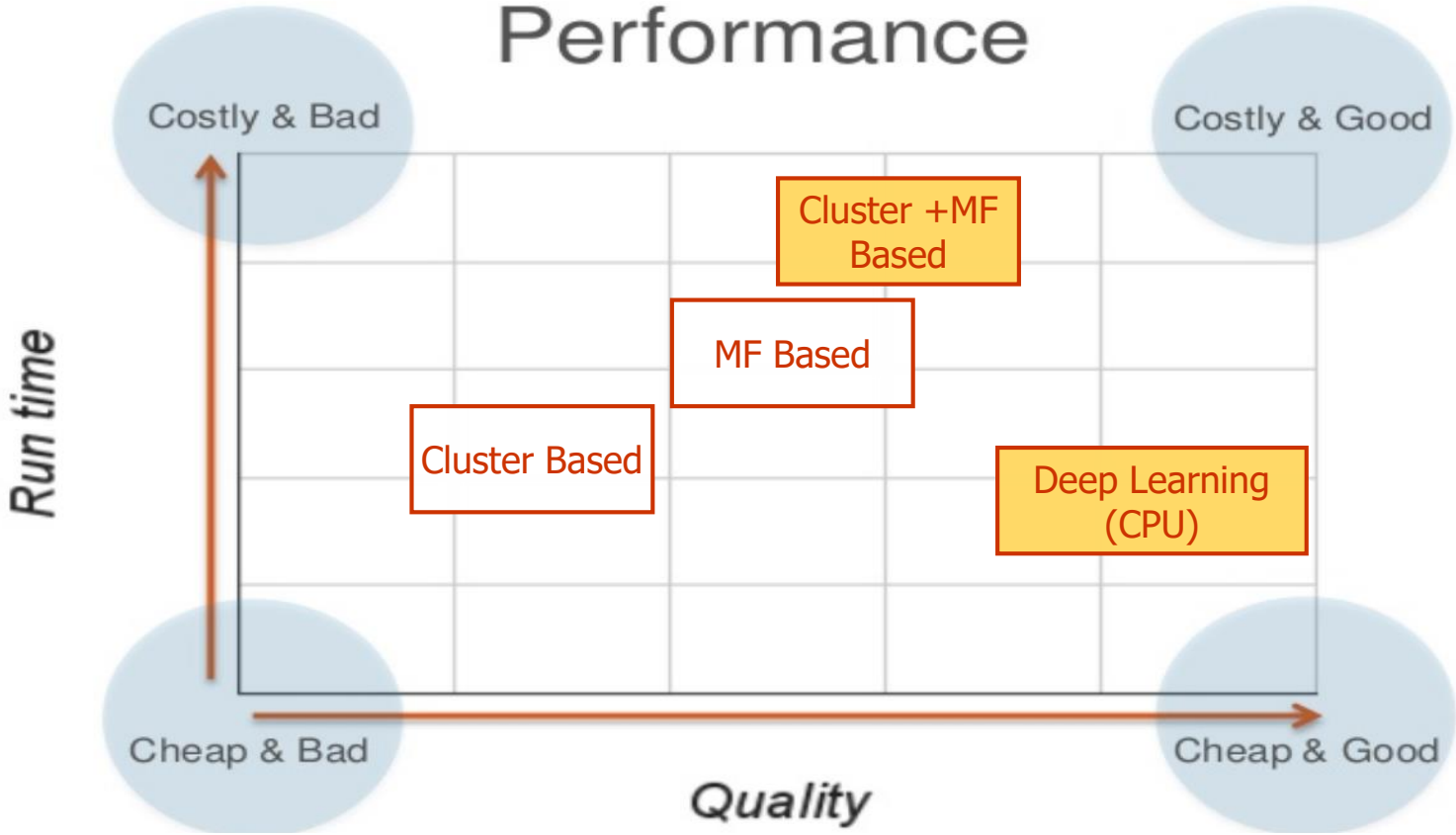




- Hyperparameters which can be adjusted:
  - **rank** = the number of latent factors in the model
  - **maxIter** = the maximum number of iterations
  - **regParam** = the regularization parameter



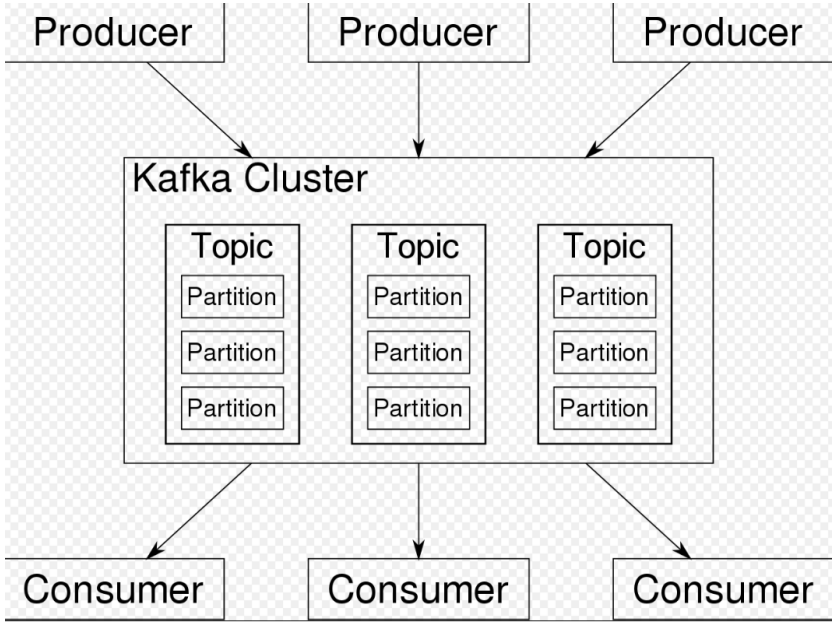
Collaborative Filtering -- Trade Offs

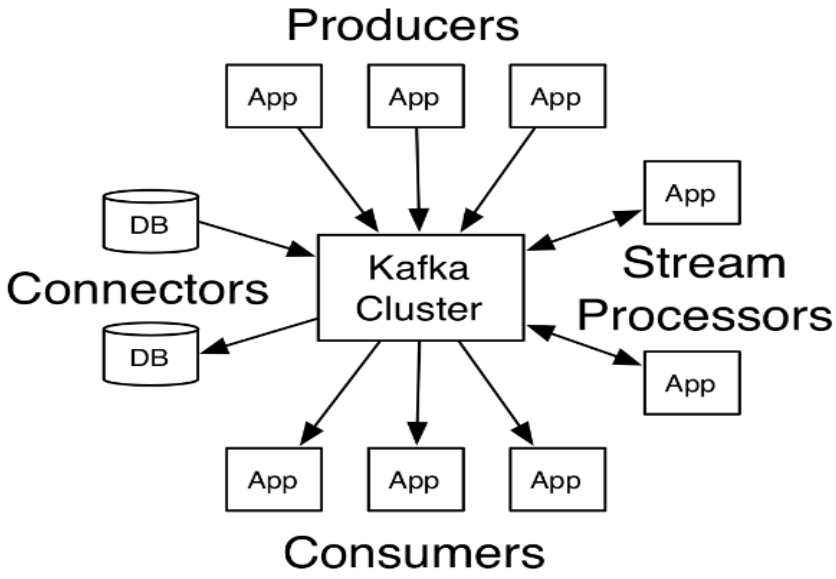


Code Time !

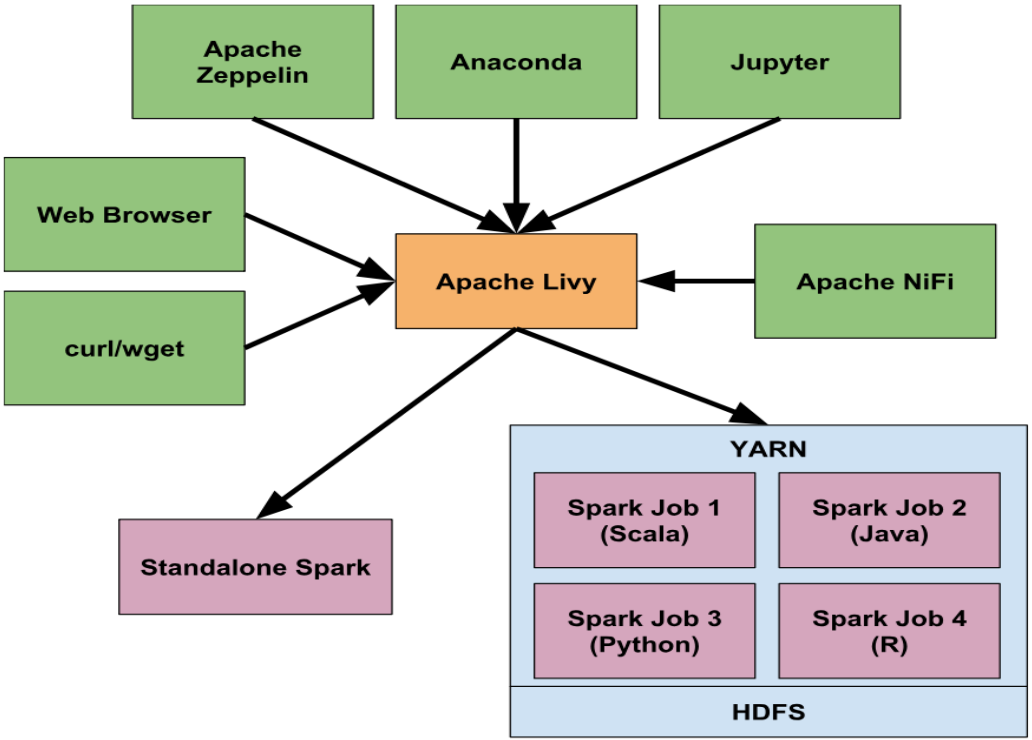
Live Demo (Build an end to end AI Pipeline with Kafka, NiFi, Spark Streaming and Keras on Spark)

Kafka

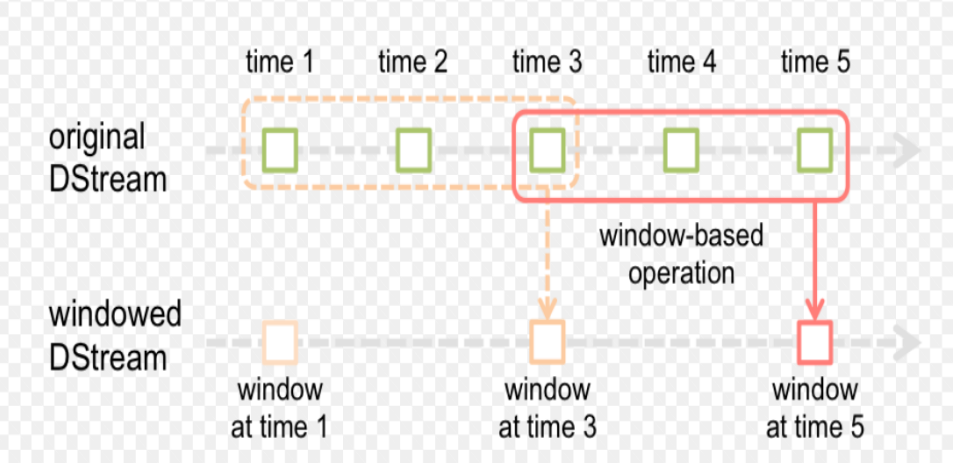

Producer
Producer
Producer
Kafka Cluster
Topic
Partition
Partition
Partition
Topic
Partition
Partition
Partition
Topic
Partition
Partition
Partition
Consumer
Consumer
Consumer

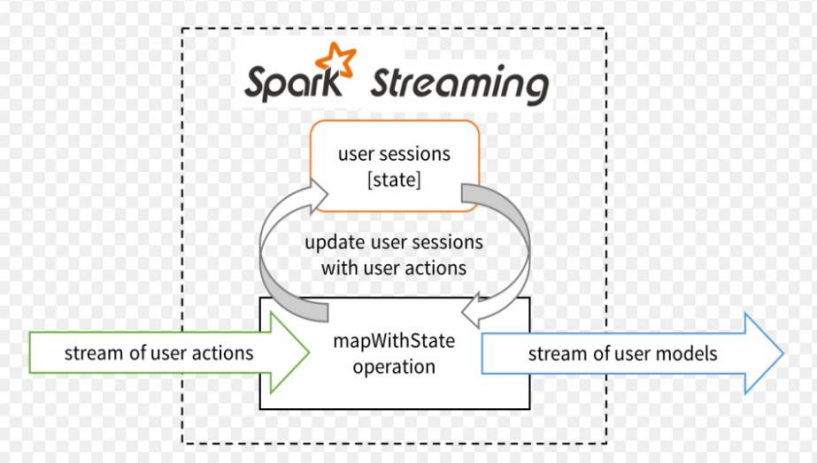

Producers
App
App
App
App
DB
Connectors
Kafka
Cluster
Stream
Processors
DB
App
App
App
App
Consumers

Livy


Apache Zeppelin
Anaconda
Jupyter
Web Browser
curl/wget
Apache Livy
Apache NiFi
Standalone Spark
YARN
Spark Job 1 (Scala)
Spark Job 2 (Java)
Spark Job 3 (Python)
Spark Job 4 (R)
HDFS

Spark Streaming





Q & A