April 04, 2023

Simplify Building and Operationalization of Machine Learning Models with Informatica Model Serve

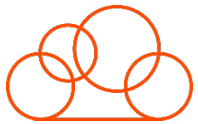
- Pawan Lawale, Principal Product Manager

Informatica

Housekeeping Tips

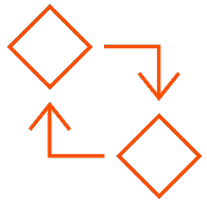
- Today's Webinar is scheduled for 1 hour
- The session will include a webcast and then your questions will be answered live at the end of the presentation
- All dial-in participants will be muted to enable the speakers to present without interruption
- Questions can be submitted to "All Panelists" via the Q&A option and we will respond at the end of the presentation
- The webinar is being recorded and will be available on our INFASupport YouTube channel and Success Portal - where you can download the slide deck for the presentation. The link to the recording will be emailed as well.
- Please take time to complete the post-webinar survey and provide your feedback and suggestions for upcoming topics.

Feature Rich Success Portal



Bootstrap trial and POC Customers

Enriched Customer Onboarding experience



Product Learning Paths and Weekly Expert Sessions

Informatica Concierge

Tailored training and content recommendations

Informatica

More Information

Success Portal

https://success.informatica.com

Communities & Support

https://network.informatica.com

Documentation

https://docs.informatica.com

University

https://www.informatica.com/in/services-and-training/informatica-university.html

Informatica

Safe Harbor

The information being provided today is for informational purposes only. The development, release, and timing of any Informatica product or functionality described today remain at the sole discretion of Informatica and should not be relied upon in making a purchasing decision.

Statements made today are based on currently available information, which is subject to change. Such statements should not be relied upon as a representation, warranty or commitment to deliver specific products or functionality in the future.

Informatica

Model Serve

Pawan Lawale
Principal Product Manager

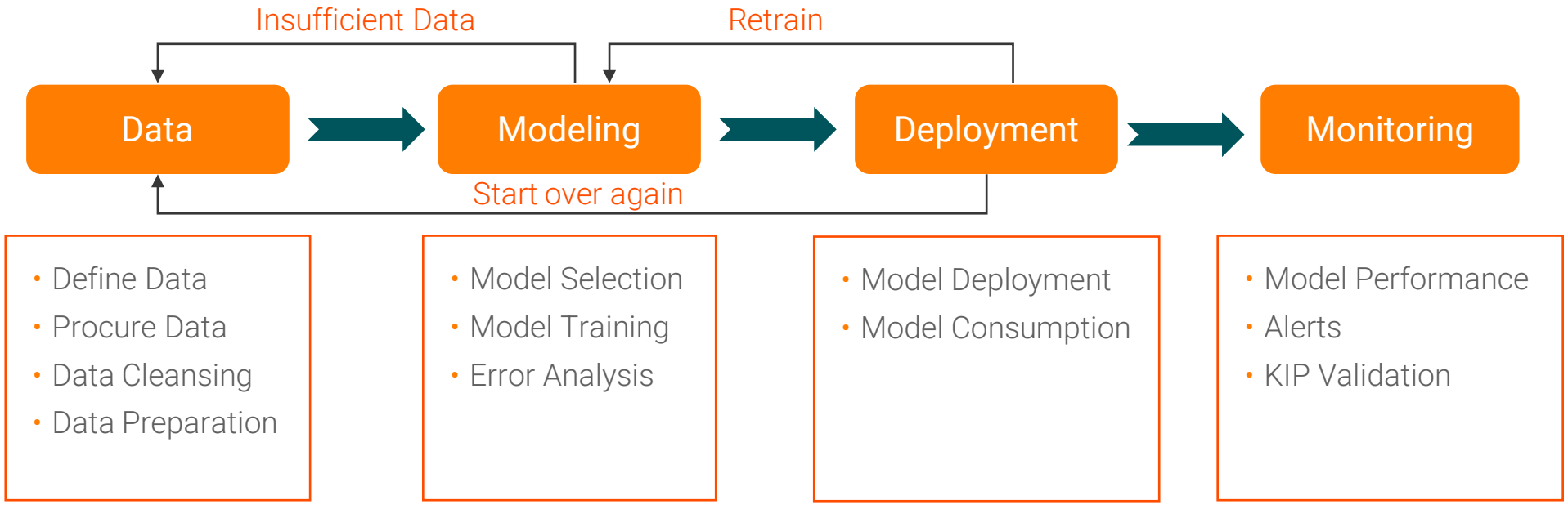
Agenda

- Introduction to MLOps
- Machine Learning Use Cases
- Challenges in ML Model Operationalization
- Informatica’s solution to MLOps
- InfaCore Overview
- Model Serve Platform
- Demo
- Q&A

What Is MLOps?

MLOps

MLOps (Machine Learning Operationalization) enables the operationalization of the end-to-end pipeline that supports the continuous delivery and continuous integration of ML models in a production environment.[2]



Source (Gartner)
[1] Magic Quadrant for Data Science and Machine Learning Platforms 2020
[2] ID G00725627 Use Gartner's 3-Stage MLOps Framework to Successfully Operationalize Machine Learning Projects

Machine Learning Use Case

Informatica

Use Case: ABC Financial Services Corp.

Scenario

ABC Financial Services Corp. is a specialist in loan/mortgage services. They need to validate loan applications for potential default. This is an extremely slow and complicated process.

Goal

- Use ML to augment the static rules to find out potential loan defaults
- Use the historical loan information to develop/train the ML model which can be used to predict potential default based on the loan application details
- The prediction of ML model will be available as one of the key metrics that will be used to decide whether to give loan or not
- This ML model will be used by multiple applications such as:
  - Informatica Mapping for batch processing
  - Java-based web application

How Are They Solving This Use Case Today?

Income Details

Education Details

Personal Details

Informatica

Cleansing validation, DQ, Complex transformation

	Y
	N
	Y
	N

Data for Training

Informatica

Prepare

Ellen
Data Engineer

Expert in building complex Data Quality, Cleansing.
Use Tools such as CDI, and CDI-E.

Apply Algorithms/Multiple Experiments with Data sets to get desired accuracy

	Y
	N

Proper Data for training

M

Trained Model

jupyter python

Build

Sam
Data Scientist

Expert in building ML models.
Uses Python with Jupyter Notebook.

Monitor and Alerts

M

Trained Model

Python Wrapper

Web apps

Informatica Data Pipelines

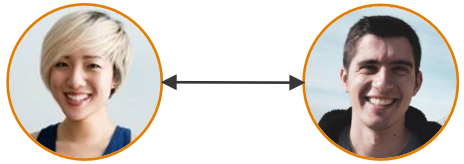
Deploy and Consume

Chris
Central IT

Expert in productionizing and provide post-production support.
Uses built-in applications for monitoring and alerting along with other productivity tools

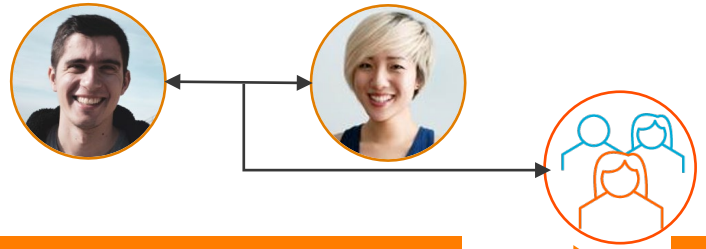
Informatica

Challenges Faced at Each Step



Prepare and Build

- Sam trains a model which has low accuracy and realizes that training data needs to be cleaned. He asks Ellen to do so; she sends cleaned data back to Sam
- A lot of back and forth between Sam and Ellen takes days or weeks and leads to overall delay



Consume and Deploy

- **Language Mismatch**: Trained model is in Python, but consuming applications are built in Java and Informatica mappings
- **Knowledge Gap**: Developers of the consuming applications are not skilled in Python
- Duplication of efforts in data preparation between Prepare and Consume step

Monitor and Alert

- **Model Drifts**
- Machine learning model is only trained on a sample of dataset. Finding whether there are differences in data attributes between Training and Serving, which is leading to incorrect results

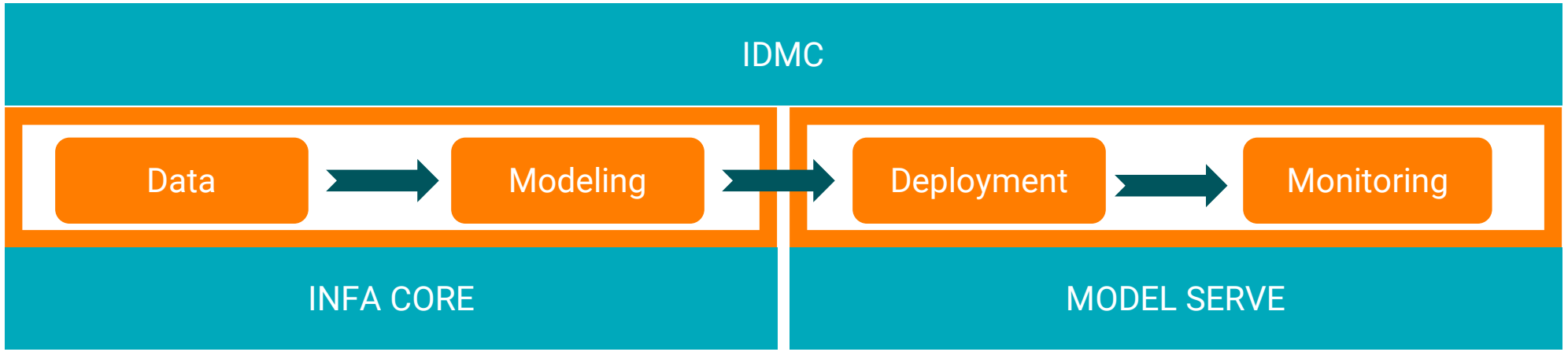
Informatica

Enterprise AI/ML Projects **Today!**

Only 15% are successful

Informatica’s solution to MLOps

Informatica provides end-to-end MLOps solution with the help of InfaCore API platform and Model Serve.



Informatica

InfaCore - Overview

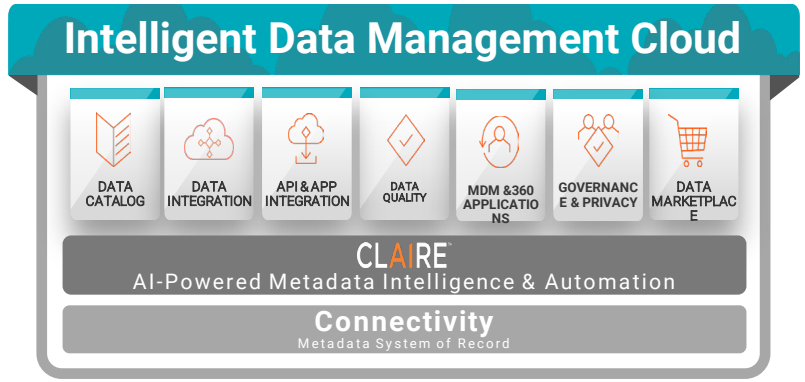
An open , extensible ,embeddable framework for democratization of data management for all developers and all data driven applications.

Initial Features
Connectivity
Advance Transformations
CLAIRE Capabilities
Operationalization

IDE Plugins

Wrappers – Programming languages

INFACORE API Layer



Informatica

Example for Simplified UX using InfaCore for Low code persona

Scenario

Hierarchy processing

Hand code/low code

```
from pyspark.sql import SparkSession
from pyspark.dbutils import DBUtils
from pyspark.sql.functions import col
from pyspark.sql.types import StructType

spark = SparkSession.builder.getOrCreate()

# declare dummy data to demonstrate how the collapse mechanism works
jsonStrings = ['{"car":{"color":"red", "model":"jaguar"},"name":"Jo","address":{"city":"Houston","state":"Texas", "zip":{"first":1234,"second":4321}}}']
otherPeopleRDD = spark.sparkContext.parallelize(jsonStrings)
df = spark.read.json(otherPeopleRDD)

# Recursively iterates over the schema, creating an array of arrays, whereby each item
# of the master array, is an array of column names
#
# For example, lets say there are three columns of which two are hierarchical and the following schema/structure
#    name
#    address
#      street
#    details
#      age
#      gender
#
# The function will return the following array:
# [["name"],["address","street"],["address","town"],["details","age"],["details","gender"]]
def get_all_columns_from_schema(source_schema):
  branches = []
  def inner_get(schema, ancestor=None):
    if ancestor is None: ancestor = []
    for field in schema.fields:
      branch_path = ancestor+[field.name]
      if isinstance(field.dataType, StructType):
        inner_get(field.dataType, branch_path)
      else:
        branches.append(branch_path)

  inner_get(source_schema)

  return branches

# collapse_columns is passed the dataframe schema, which is then passes
# to get_all_columns_from_schema.  On return, it iterates through the array
# of columns in order to build up the select list that will be used
# to collapse the hierarchical columns into a single 2d structure
#
# for example, lets say _all_columns has the following array: [["name"],["address","street"]]
# after iterating through the array, the function response will be
# [col("name"), col("address.street").alias("address_street")]
def collapse_columns(source_schema, columnFilter=None):
  _columns_to_select = []
  if columnFilter is None: columnFilter = ""
  _all_columns = get_all_columns_from_schema(source_schema)
  for column_collection in _all_columns:
    if (len(columnFilter) > 0) & (column_collection[0] != columnFilter):
        continue

    if len(column_collection) > 1:
      _columns_to_select.append(col('.'.join(column_collection)).alias('_'.join(column_collection)))
    else:
      _columns_to_select.append(col(column_collection[0]))

  return _columns_to_select

# as above but for individual columns
def collapse_column(source_df, source_column):
    column_name = ""
    if isinstance(source_column, Column):
      column_name = source_column.name
    else:
      column_name = source_column

    return collapse_columns(source_df.schema, column_name)

# returns a dataframe that has been collapsed.  Input is the dataframe to be collapsed
def collapse_to_dataframe(source_df):
  return source_df.select(collapse_columns(source_df.schema))
```

Informatica Mapping

InfaCore API code

Infacore.parse_hierarchy(input, type_def_sch)

Informatica

Model Serve

Informatica

Model Serve

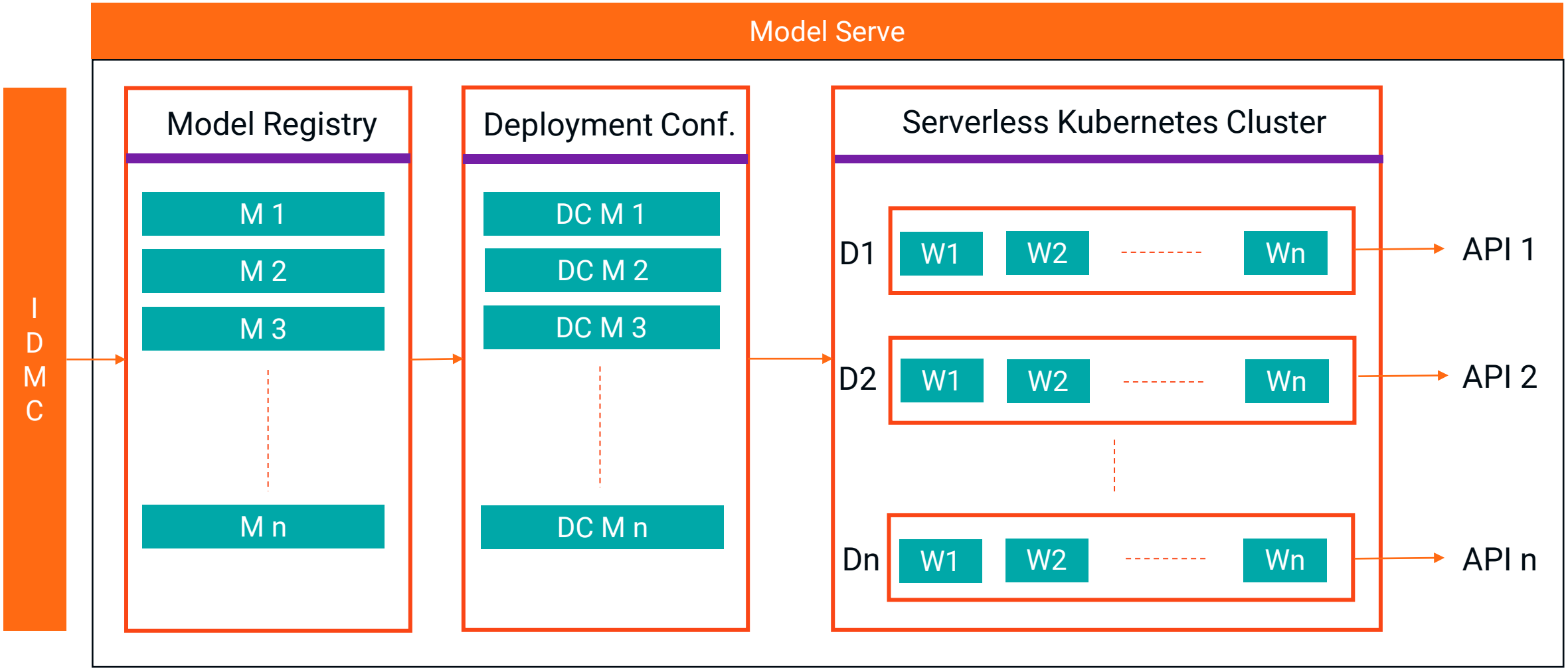
Key Highlights

- Centralized model registry & management within IDMC
- Framework/Compute Agnostic platform for Operationalizing any model at scale
- One click Elastic Serving of registered ML models
- Monitoring and Alerting of served models.

Benefits

- Operationalization of ML Models at Scale
- Ease of Maintenance of ML Lifecycle

Model Serve Architecture



Informatica

Demo

Q & A

Informatica