#CrecimientoProfesional
#AprendeConLosPioneros

Online | DMC Perú

DMC Perú | CAPACITACIÓN PROFESIONAL

Agilizando la Implementación de Modelos Predictivos

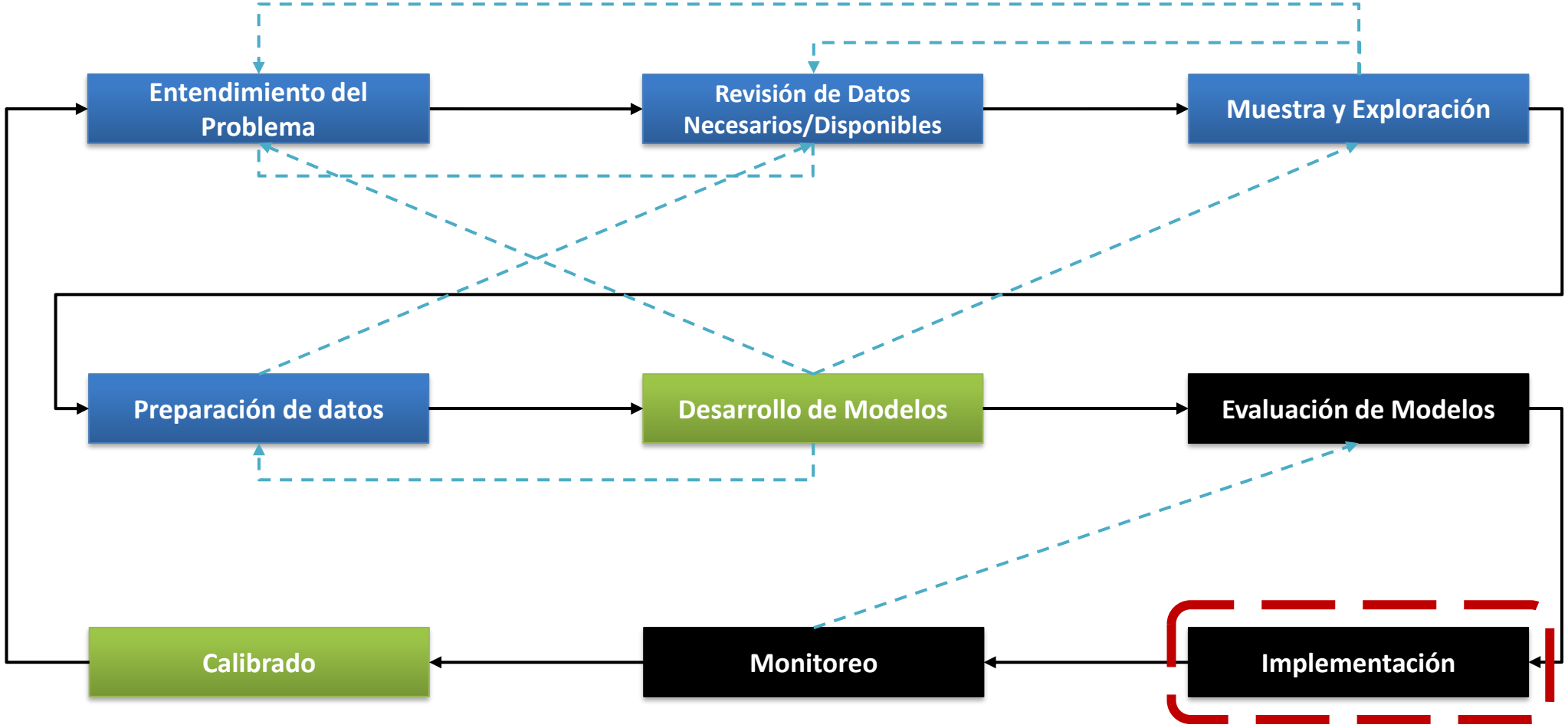
Ponente: Luis Cajachahua

Antecedentes:

https://github.com/lcajachahua/modelos-produccion

Entendiendo el Proceso Analítico

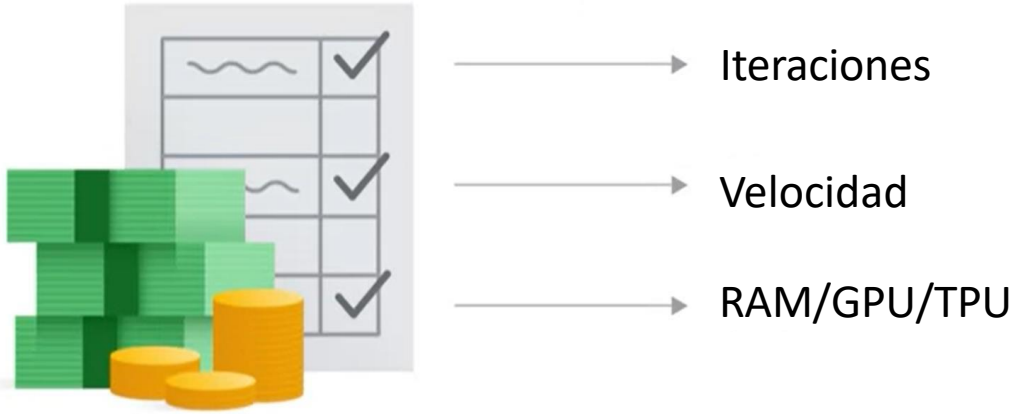
Ciclo de Vida de un Modelo



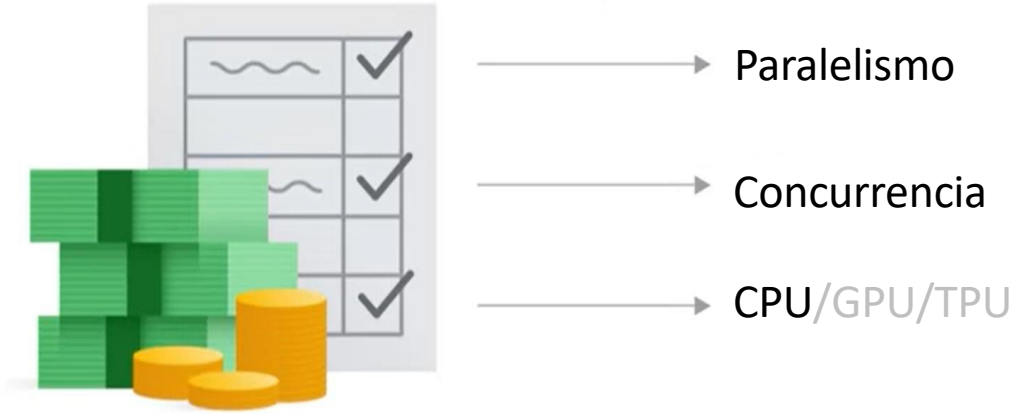
Requerimientos de un Modelo

- En el proceso analítico, existen dos fases muy distintas que demandan distintas capacidades. En la fase de entrenamiento de modelos, el proceso de optimización de parámetros y búsqueda iterativa de los óptimos globales se acelera a través del uso intensivo de la RAM o de GPU/TPU. En cambio, una vez que ya se tiene el modelo entrenado, no se necesita más la RAM/CPU/TPU, sino otras capacidades como el procesamiento masivo, escalable y paralelo.

Entrenamiento



Scoring



Procesamiento Analítico

Los requerimientos técnicos de Entrenamiento son distintos

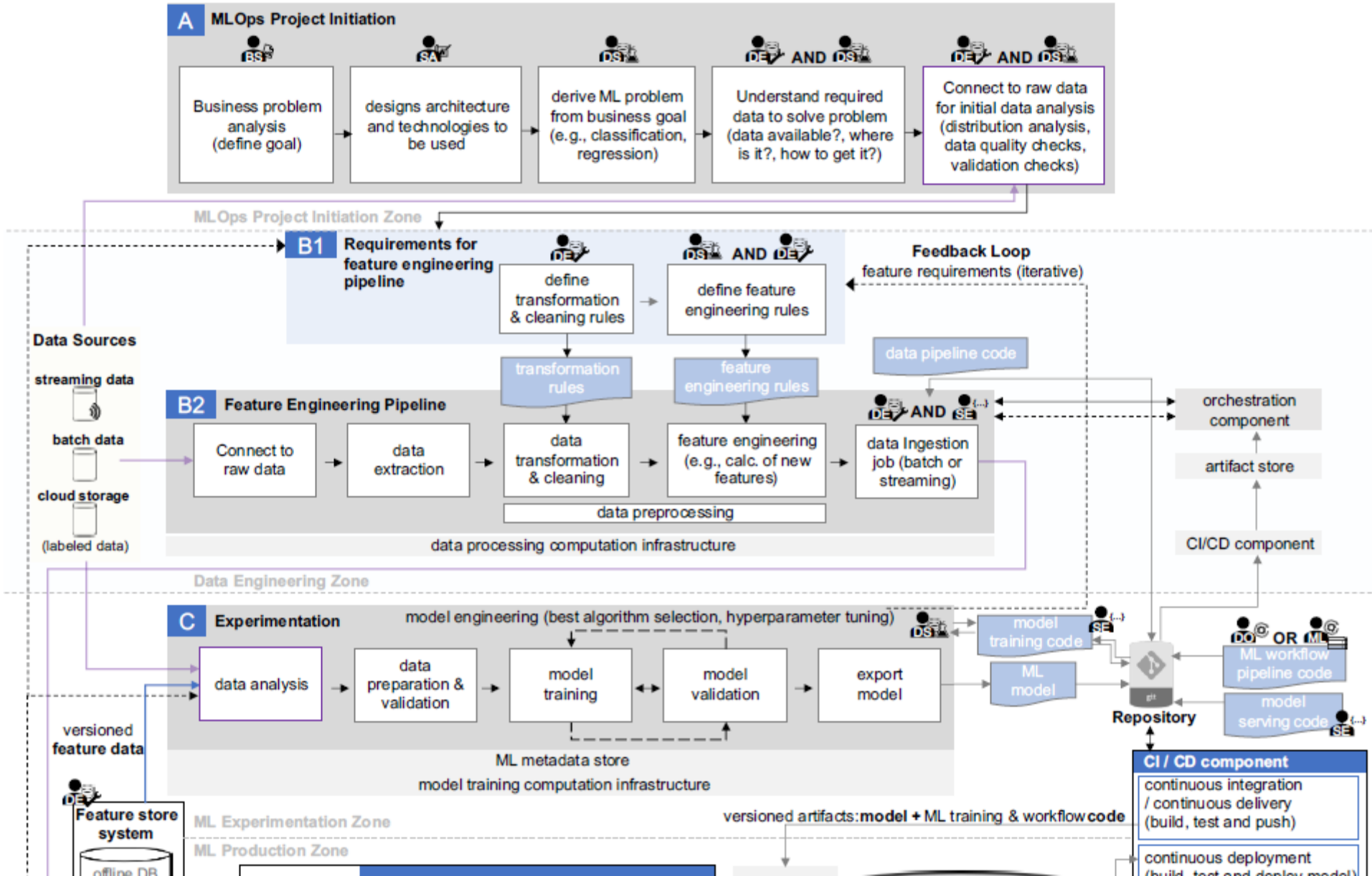
Algorithm	Training (time complexity)	Prediction (time complexity)	Auxiliary space complexity	Comment
Linear regression	$O(nm^2+m^3)$ [1]	O(m)	O(m)	Low run time and space complexity! Suitable for low latency problems
Logistic regression	O(nm) [2]	O(m)	O(m)	Low run time and space complexity! Suitable for low latency problems
Naive Bayesian	O(n*m)	O(c*m)	O(c*m)	Works well with high dimensional data. Mainly used in NLP problems with large dimensionality. Also used to compare with more complex algorithms used for NLP.
Decision Trees	O(nlog(n)d)	O(d)	O(p)	
Gradient Boosting	O(nlog(n)dr)	O(dr)	O(pr+br)	br could be dropped as br<<pr
Random Forest	$O(nlog(n)dn_{tree})$	$O(dn_{tree})$	$O(pn_{tree})$	
Support Vector Machines	$O(n^2m+n^3)$	$O(mn_{sv})$	$O(n_{sv})$	High training time but low space complexity. Suitable for low latency problems

Algorithm	Training (time complexity)	Prediction (time complexity)	Auxiliary space complexity	Comment
K Means	O(nmki)	O(mk)	O(nm+km)	
KNN	O(1)	O(nm)	O(nm)	
Hierarchial clustering	$O(n^3) \Rightarrow O(n^2log(n))$	O(mn)	$O(n^2)$	High training time, high runtime, and high space complexities. Not suitable for low latency.
DBSCAN	$O(n^2) \Rightarrow O(nlog(n))$	O(mn)	O(n)	Better than heirarchial clustering interms on complexities
PCA	$O(nm \times min(n, m) + m^3)$	O(lm)	O(lm)	The complexity of covariance matrix computation is O(nm×min(n, m)). Its eigenvalue decomposition is O(m^3). O(lm) is for the transformation matrix
t-SNE	$O(n^2)$	O(nq)	$O(n^2)$	t-SNE involves significant amount of computations and hence the algorithm takes a lot of time and space for the computation. t-SNE has a quadratic time and space complexity in the number of data points. **t-SNE requires $O(3n^2)$ of memory**

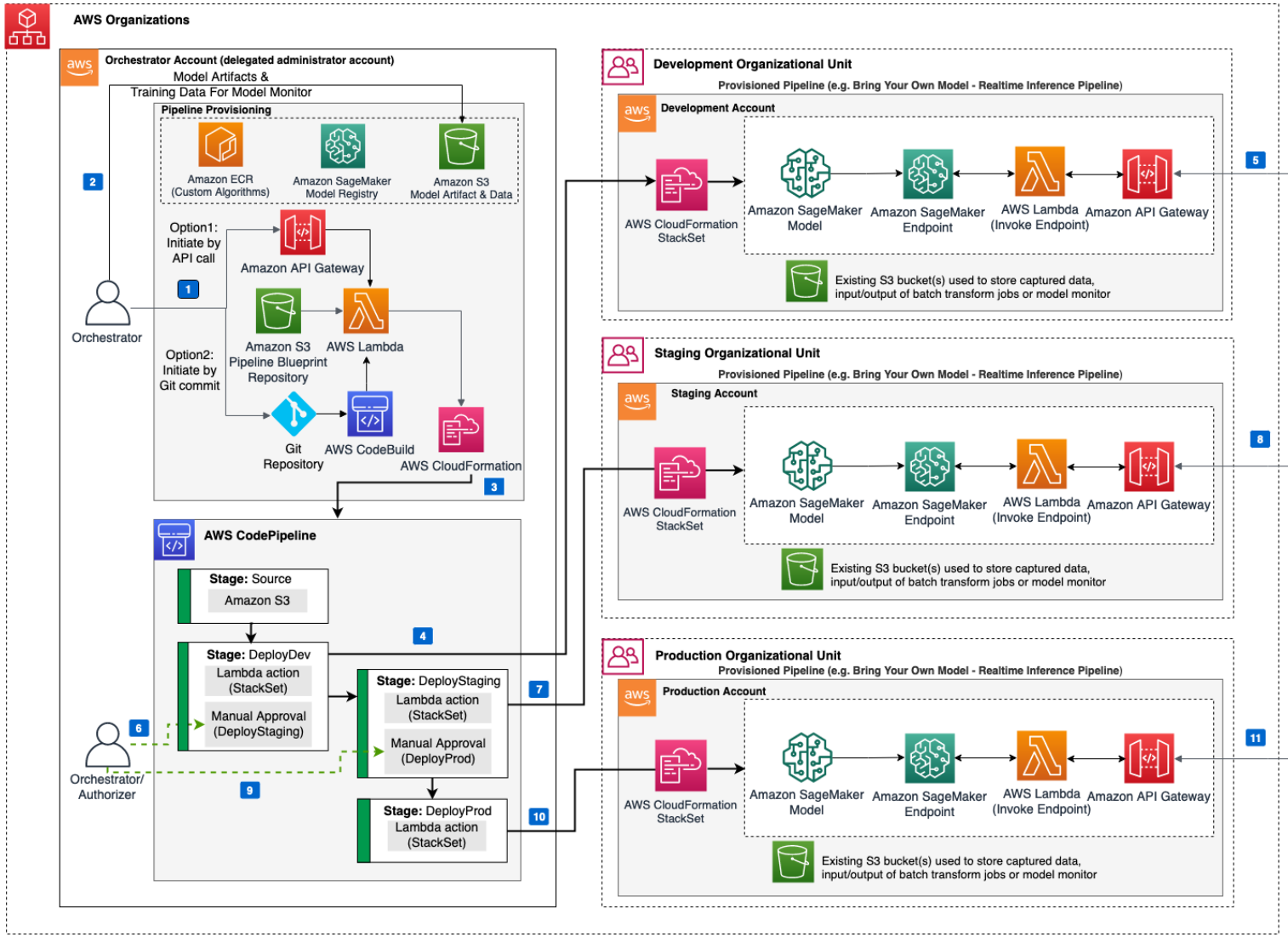
Por eso no tiene tanto sentido utilizar las mismas tecnologías para Entrenamiento y Scoring

MLOps

MLOps (Genérico)



MLOps (Ejemplo Cloud)



- Tiempo
- Capacidad
- Roles Escasos
- Factura Imprevisible

Formatos Estándar para Acelerar la Implementación

Formatos Estándar

Los formatos estándar en Ciencia de Datos existen desde hace más de 20 años, siendo los últimos en los que se han posicionado como una alternativa interesante para lidiar con las limitaciones de las plataformas analíticas en cuestiones de escalabilidad.

Ventajas:

- Reduce la necesidad de un entorno productivo analítico separado del resto, lo que representa costo en infraestructura, licencia y capacidades. En muchos casos este entorno puede ser costoso debido a los requerimientos adicionales de escalabilidad y concurrencia.
- Reduce la dependencia de perfiles profesionales difíciles de reclutar y mantener en la organización.
- Simplifica la implementación de modelos, además de eliminar la necesidad de mover datos a otra plataforma para realizar el proceso de scoring.
- Pueden utilizarse en combinación con diversas bases de datos, a través de funciones SQL o de scripts en Java o C.

Formatos Estándar

Standard Format

PMML
NNVM
ONNX
NNEF

Single stack

Execution

Optimization

Tooling (Viz, analysis, ...)

PMML

- El lenguaje de marcado de modelos predictivos (Predictive Model Markup Language o PMML) es un formato de intercambio de modelos predictivos basado en XML concebido por el Dr. Robert Lee Grossman en 1998, entonces director del Centro Nacional de Minería de Datos de la Universidad de Illinois en Chicago. Las versiones posteriores han sido desarrolladas por Data Mining Group.
- PMML proporciona una forma para que las aplicaciones analíticas describan e intercambien modelos predictivos producidos por algoritmos de aprendizaje automático y minería de datos. Admite modelos comunes como la regresión logística, árboles de decisión, árboles ensamblados, redes neuronales y otros algoritmos avanzados.
- Dado que PMML es un estándar basado en XML, la especificación viene en forma de esquema XML. PMML en sí mismo es un estándar maduro con más de 30 organizaciones que han anunciado productos compatibles con PMML. Entre ellas tenemos SAS, R, Python, Knime, SPSS, Dataiku, RapidMiner, Weka, Tibco Data Science/Statistica.

ONNX

- ONNX es el acrónimo de Open Neural Network Exchange.
- Proporciona un formato de código abierto para modelos de IA, tanto de Machine Learning como de Deep Learning.
- Fue desarrollado por un consorcio (Microsoft, Google, AWS) y ahora es administrado por Microsoft.
- Almacena datos en un formato de llamada protobuf (búfer de protocolo), el mismo formato de archivo utilizado en Tensorflow y Caffe.
- Sirve como mediador entre una variedad de marcos como pytorch, scikit-learn, tensorflow y una variedad de entornos de destino (cpu, gpu y otros proveedores de ejecución)

Otros Formatos Estándar

MOJO

- Esta funcionalidad de H2O permite convertir los modelos que construye en MOJO, que luego se pueden implementar para efectuar scoring en otras plataformas.
- Los MOJO son compatibles con los modelos Deep Learning, DRF, GBM, GLM, GAM, GLRM, K-Means, PCA, Stacked Ensembles, SVM, Word2vec, Isolation Forest, XGBoost, CoxPH y RuleFit. Todos los modelos generados por AutoML son compatibles.
- Los MOJO se pueden usar en aplicaciones Java y también se pueden usar en R/Python para generar predicciones para datos almacenados en un marco de datos R/Python en memoria o en un archivo CSV.

MLeap

- Es un formato de serialización común y un motor de ejecución para canalizaciones de aprendizaje automático.
- Es compatible con Spark, Scikit-learn y Tensorflow para entrenar canalizaciones y exportarlas a un paquete MLeap.
- Las canalizaciones serializadas (paquetes) se pueden deserializar nuevamente en Spark para la puntuación en modo batch o el tiempo de ejecución de MLeap para potenciar los servicios de API en tiempo real.

¿Por qué usar formatos estándar es más rápido que MLOps?

MLOps	BYOM
Generating Scripts for Training in Prod Environment - Train Script - Validation Script	Not required, you can reuse your development script (Process or Notebook) to train the final and validate the model
Run formatting tests (lenting in Flake8 or similar)	Not required
Run functional tests (all processes producing expected results)	Not required
Create the requirements.txt list of libraries	Not required
Create the CI/CD process using specific tools	Not required
Configure the production environment using Kubernetes, Docker and Base Images	Not required
Retrieve the Project from Production Environment	Version control could store scripts and models as files
Build the Container and install the Libraries	Not required
Run Train & Validation Scripts	Not required, done in Development
Adding the trained model to Model Registry	Load the Model in the Database using standard formats
Generate the Scoring Script, formatting and functional tests	Generate BTEQ process for Scoring (SQL)
Option A: Schedule the execution (Batch)	**Option A:** Schedule using cron, Vantage Workflow or Control-M
Option B: Create API deployment script, considering security, data protection and scalability	**Option B:** Use the Vantage API Service or Cloud Provider Services to Deploy the Model

Ejemplo: Implementando un LLM en Teradata

A custom model is trained on enterprise data for domain specific use-cases (such as Q&A model for virtual assistant). It is not necessarily large)

1 **Prepare analytic data set for modeling**

Prepare dataset from text data

TERADATA VANTAGE

Sample Data

API for model training

2 **Build custom model with any tool**

Allow data scientists freedom to use preferred tools to train a custom language models

TensorFlow, PyTorch, AzureML, LLaMA Meta, Google BERT, Amazon Bedrock, OpenAI GPT

3 **Export model for scoring in production**

ONNX

BYOM

Model Interchange File

API for model scoring

Model score

API

4 **Score entire data set in parallel on Vantage**

Demo de Formatos Estándar

DIIC Perú | CAPACITACIÓN **PROFESIONAL**

Online | DIIC Perú