AWS Whitepaper

Accenture Enterprise AI – Scaling Machine Learning and Deep Learning Models

Accenture Enterprise AI – Scaling Machine Learning and Deep Learning Models: AWS Whitepaper

Table of Contents

Accenture Enterprise AI – Scaling Machine Learning and Deep Learning Models

Publication date: **July 27, 2022** (*Document revisions*)

Abstract

Today, there is a real-time, global, tectonic shift in the workplace caused by digital transformation. Accelerated by the Covid pandemic, this digital transformation has created never-seen-before opportunities and significant workplace disruption. Fully realizing the new market opportunities demands a modernized workforce. A skills gap contributed to by several factors exist in today's labor market. Some of these factors are the increase in the number of people entering the workforce each year, lack of relevant education, and the rise in technology which needs workers to be equipped with new skills to help them keep up with advancements. Addressing this widening gap between the current workforce skills and those needed for tomorrow is front and center in the minds of every C-suite.

This whitepaper outlines an innovative, scalable and automated solution using deep learning (DL) and machine learning (ML) on Amazon Web Services (AWS), to help solve the problem of bridging the existing talent and skills gap for both workers and organizations. Combining advanced data science, ML engineering, deep learning, ethical artificial intelligence (AI), and MLOps on AWS, this whitepaper provides a roadmap to enterprises and teams to help build production-ready ML solutions, and derive business value out of the same.

Are you Well-Architected?

The AWS Well-Architected Framework helps you understand the pros and cons of the decisions you make when building systems in the cloud. The six pillars of the Framework allow you to learn architectural best practices for designing and operating reliable, secure, efficient, cost-effective, and sustainable systems. Using the AWS Well-Architected Tool, available at no charge in the AWS Management Console, you can review your workloads against these best practices by answering a set of questions for each pillar.

In the Machine Learning Lens, we focus on how to design, deploy, and architect your machine learning workloads in the AWS Cloud. This lens adds to the best practices described in the Well-Architected Framework.

For more expert guidance and best practices for your cloud architecture—reference architecture deployments, diagrams, and whitepapers—refer to the AWS Architecture Center.

Introduction

Today's rapidly changing environment demands the ability for organizations to adapt to change to create a sustainable and productive workforce. Thriving in this environment requires rapid adaptation and readiness for upskilling the workforce for tomorrow.

According to VentureBeat, about 87% of ML models never make it to production. Even though 9 out of 10 business executives believe that AI will be at the center of the next technological revolution, completion, and successful production deployment is seen as a big challenge as it requires specific engineering expertise and collaboration between several teams (ML engineering, IT, Data Science, DevOps, and so on).

Accenture has built a scalable, industrialized, AI-powered solution that is a key component in helping solve the talent and skilling problem of today and tomorrow to create a productive workforce. It describes an innovative, cloud-native AWS approach that can be taken to industrialize the ML solution, and help organizations bridge the skills gap.

This whitepaper describes a technical solution (also referred to as industry solution) for building and scaling ML, and specifically, DL models for these use cases, and how Accenture is industrializing the end-to-end process to achieve the technical goals previously detailed. The technical thought process explained here can be expanded and applied to most problems in other industries. You can also use it to create a stable and sustainable Enterprise AI system.

Frictionless ideation to production

The goal of Enterprise AI and MLOps is to reduce friction and get all models from ideation to production in the shortest possible time, with as little risk as possible. Integrating AI technologies into business operations can prove to be a game-changer for organizations, with the benefits of reducing costs, boosting efficiency, generating actionable, precise insights, and creating new revenue streams. This requires not only creating efficient models, but also creating a complete end-to-end stable, resilient, and repeatable Enterprise AI system that can provide sustainable value and be amenable to continuous improvements to adapt to changing environments.

Workforce analytics use cases

Workforce analytics is an advanced set of data analysis tools and metrics for comprehensive workforce performance measurement and improvement. Workforce analytics are highly sensitive by nature and require trusted ML and DL algorithms to create scalable, productionized solutions for harnessing human potential.

To bring about a balance between responsible AI and computing at scale, Accenture created an advanced, reusable, AWS MLOps architecture for multiple workforce productivity use cases, including:

- **Solutions.AI for Talent and Skilling** — An AI-powered solution that delivers intelligent insights to help close the skills gap in any organization. Through Solutions.AI, Accenture created enterprise-wide AI solutions that deliver game-changing results, fast.
- **FutureofU: Skills. Jobs. Growth** — An AI-enabled platform solution created in collaboration with Accenture partners, who are committed to accelerating a smooth transition to help get diverse talent skilled and integrated into the AI workforce.
- **Intelligent Workforce Insights** – Uses the power of cutting-edge AI/ML technologies to develop role maps that identify declining, stable, and emerging skills, both internally and in the market, and assess roles for upskilling.

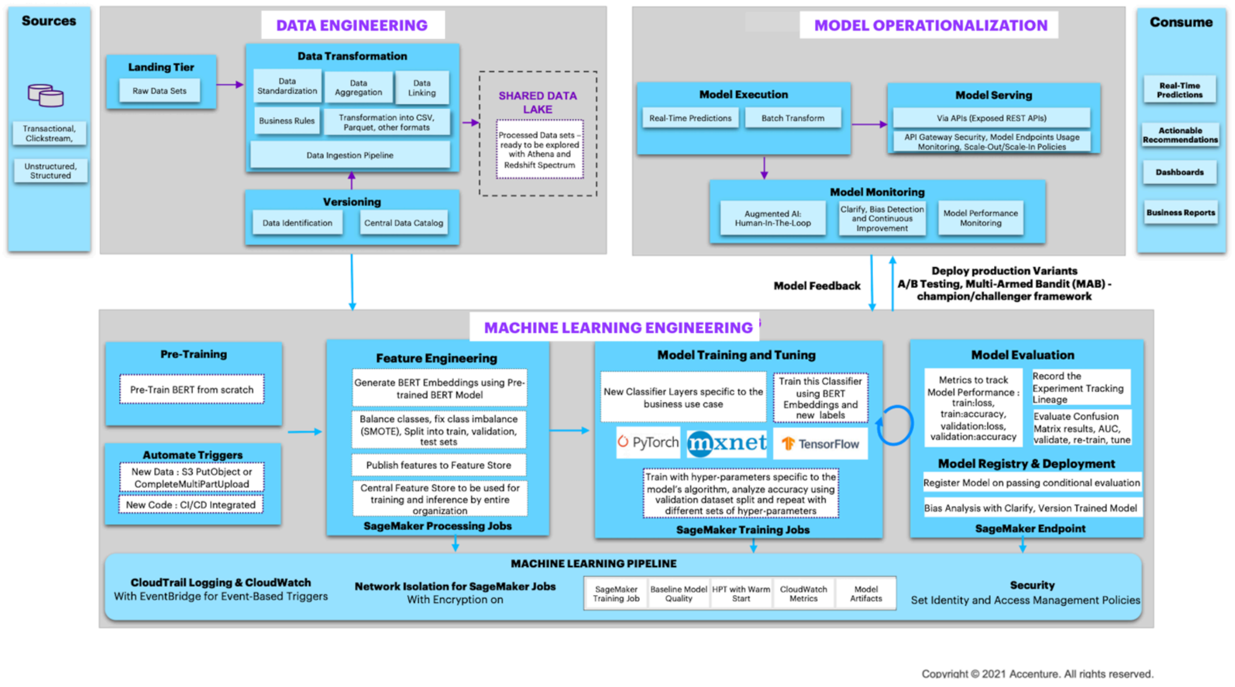
The challenge with these use cases is that the backend needs massive amount of computing resources to infer and produce the ML score results. For optimal user experience, the scoring needs to happen fast enough, with near real-time intermediate scores. All three of these use cases have three common technical traits:

- ML engineering at scale
- AI/ML automation with responsible AI
- End-to-end productionized large-scale resilient AI/ML systems

An intelligent platform approach

Deep learning is set to achieve transformation across industries and create new opportunities on a scale not seen since the industrial revolution in the 19th century. It comes with great promise, but with it are significant challenges. Most DL models are built on richer algorithmic primitives, and therefore require higher reuse between tasks, rather than training a model from scratch each time there is a new problem at hand, or a new dataset.

For businesses to derive value, ML and DL models need to be productionized, run at scale, and reused across organizations. Lack of scalability, repeatability, and manual processes diminish any value that would be otherwise realized from these powerful models.


Sources
Transactional, Clickstream,
Unstructured, Structured
DATA ENGINEERING
Landing Tier
Raw Data Sets
Data Transformation
Data Standardization
Data Aggregation
Data Linking
Business Rules
Transformation into CSV, Parquet, other formats
Data Ingestion Pipeline
Versioning
Data Identification
Central Data Catalog
SHARED DATA LAKE
Processed Data sets – ready to be explored with Athena and Redshift Spectrum
MODEL OPERATIONALIZATION
Model Execution
Real-Time Predictions
Batch Transform
Model Serving
Via APIs (Exposed REST APIs)
API Gateway Security, Model Endpoints Usage Monitoring, Scale-Out/Scale-In Policies
Model Monitoring
Augmented AI: Human-In-The-Loop
Clarify, Bias Detection and Continuous Improvement
Model Performance Monitoring
Consume
Real-Time Predictions
Actionable Recommendations
Dashboards
Business Reports
Model Feedback
Deploy production Variants A/B Testing, Multi-Armed Bandit (MAB) - champion/challenger framework
MACHINE LEARNING ENGINEERING
Pre-Training
Pre-Train BERT from scratch
Automate Triggers
New Data : S3 PutObject or CompleteMultiPartUpload
New Code : CI/CD Integrated
Feature Engineering
Generate BERT Embeddings using Pre-trained BERT Model
Balance classes, fix class imbalance (SMOTE), Split into train, validation, test sets
Publish features to Feature Store
Central Feature Store to be used for training and inference by entire organization
SageMaker Processing Jobs
Model Training and Tuning
New Classifier Layers specific to the business use case
Train this Classifier using BERT Embeddings and new labels
PyTorch
mxnet
TensorFlow
Train with hyper-parameters specific to the model's algorithm, analyze accuracy using validation dataset split and repeat with different sets of hyper-parameters
SageMaker Training Jobs
Model Evaluation
Metrics to track Model Performance : train:loss, train:accuracy, validation:loss, validation:accuracy
Record the Experiment Tracking Lineage
Evaluate Confusion Matrix results, AUC, validate, re-train, tune
Model Registry & Deployment
Register Model on passing conditional evaluation
Bias Analysis with Clarify, Version Trained Model
SageMaker Endpoint
MACHINE LEARNING PIPELINE
CloudTrail Logging & CloudWatch With EventBridge for Event-Based Triggers
Network Isolation for SageMaker Jobs With Encryption on
SageMaker Training Job
Baseline Model Quality
HPT with Warm Start
CloudWatch Metrics
Model Artifacts
Security Set Identity and Access Management Policies
Copyright © 2021 Accenture. All rights reserved.

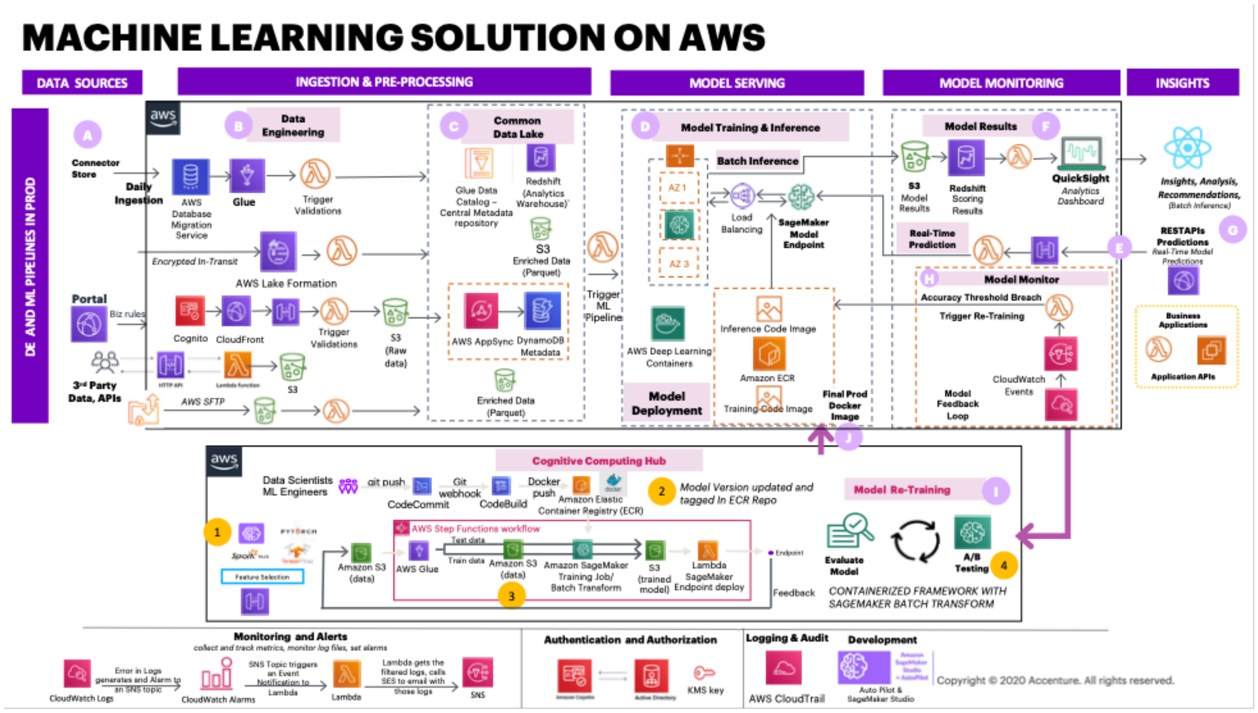
Complete solution for scaling and productionizing DL models with automated pipelines

This proposed architecture in the previous diagram is designed to help achieve three goals:

- Greater, systematic reuse of features and architectures
- Reduces manual processes
- Increases speed to market

ML architecture on AWS

Expanding on the previous architecture, the following architecture is a drill-down of how a complete solution can be built with various AWS components and connected for a seamless, resilient, production-grade solution that is driven by performance and easy to maintain.



Complete solution with cloud-native AWS components

(Copyright 2022 © Accenture. All rights reserved)

Feature engineering

Many DL and ML models are used for the workforce productivity solution; however, text classification and sentence prediction are inherently the main classifiers you need. Given the superior performance of neural language models, and because it enables machines to understand qualitative information, it fits the need of building neural network-based DL models for assessing peoples' skills proficiency, and for recommending new career pathways.

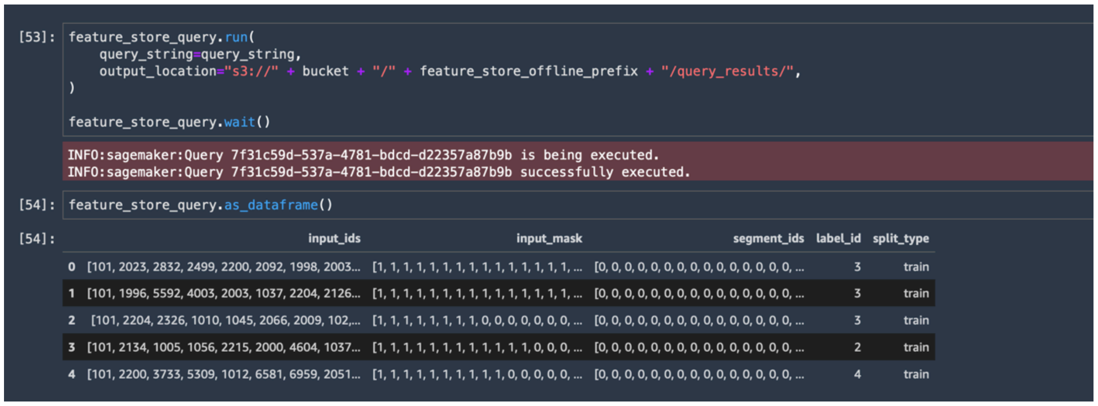
Bidirectional Encoder Representations from Transformers (BERT) is the first Natural Language Processing (NLP) technique to rely solely on a self-attention mechanism, which is made possible by the bidirectional transformers at the center of BERT's design. This is significant because a word may change meaning as a sentence develops. Each word added augments the overall meaning of the sentence, and the context may completely alter the meaning of a specific word.

The feature store

One of the key needs for the industry use cases listed in this whitepaper is to provide C-suite and organizations with a roadmap to accelerate, scale, and sustain digital adoption. To enable individual talent mobility using AI, it is necessary to collect data points at the individual level. Making AI models understand people's strengths, interests, and other personal criteria result in providing better career recommendations that benefit the workforce and organizations alike. One of the first steps in the journey of creating a productionized, stable AI/ML platform is to focus on a centralized feature store.

After Amazon SageMaker Processing applies the transformations defined in the SageMaker Data Wrangler, the normalized features are stored in an offline feature store so the features can be shared and reused consistently across the organization among collaborating data scientists. This means SageMaker Processing and Data Wrangler can be used to generate features, and then store them in a feature store. This standardization is often key to creating a normalized, reusable set of features that can be created, shared, and managed as input into training ML models. You can use this feature consistency across the maturity spectrum, whether you are a startup or an advanced organization with an ML center of excellence.

The Amazon SageMaker Feature Store is accessible across the organization for different teams to collaborate, promoting reuse, reducing overall cost, and avoiding silos with duplicate work efforts. The following query is a sample of the central Feature Store created with BERT embeddings. A SageMaker Feature Group and a Feature Store are created. Multiple downstream teams can retrieve and use features from this central store instead of redoing feature engineering repeatedly, adding to the organization's operational costs and non-standardization issues.



```
[53]: feature_store_query.run(
          query_string=query_string,
          output_location="s3://" + bucket + "/" + feature_store_offline_prefix + "/query_results/",
      )

      feature_store_query.wait()

      INFO:sagemaker:Query 7f31c59d-537a-4781-bdcd-d22357a87b9b is being executed.
      INFO:sagemaker:Query 7f31c59d-537a-4781-bdcd-d22357a87b9b successfully executed.

[54]: feature_store_query.as_dataframe()

[54]:
```

	input_ids	input_mask	segment_ids	label_id	split_type
0	[101, 2023, 2832, 2499, 2200, 2092, 1998, 2003...	[1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, ...	[0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, ...	3	train
1	[101, 1996, 5592, 4003, 2003, 1037, 2204, 2126...	[1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, ...	[0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, ...	3	train
2	[101, 2204, 2326, 1010, 1045, 2066, 2009, 102,...	[1, 1, 1, 1, 1, 1, 1, 1, 0, 0, 0, 0, 0, 0, ...	[0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, ...	3	train
3	[101, 2134, 1005, 1056, 2215, 2000, 4604, 1037...	[1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 0, 0, 0, ...	[0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, ...	2	train
4	[101, 2200, 3733, 5309, 1012, 6581, 6959, 2051...	[1, 1, 1, 1, 1, 1, 1, 1, 1, 0, 0, 0, 0, 0, ...	[0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, ...	4	train

Feature Store with BERT embeddings ready for reuse across the organization

The algorithms

There are three key pillars to building successful ML applications. If not done correctly, in spite of all the state-of-the-art continuous integration/continuous delivery (CI/CD), feature store, feature engineering, and graphics processing unit (GPU)-accelerated DL or automated pipelines, the end-to-end Enterprise AI platform is bound to fail:

- The quality of the data
- The minimum level of complexity employed to solve the problem
- The ability of the solution to be measured and monitored

Data engineering and data quality

The talent and skilling industry use case requires over 20 data sources to be ingested from. One of the main challenges is to fix data quality before feeding the raw datasets into your DL models for classification and recommendation. Data quality issues can deeply impact not just the data engineering pipelines, but all the ML pipelines downstream as well. Deequ helps in analyzing the datasets across all the stages of feature engineering, training and deployment. The training-serving skew is aptly shown by Deequ, by detecting deviation from baseline statistics. Deequ can create schema constraints and statistics for each input feature. Completeness, Correlation, Uniqueness and Compliance Deequ metrics can be tracked in the `MetricsRepository`, and Spark processing alerts can be set on detecting anomalies for immediate actions.

Hyper-parameter tuning (HPT)

As hyper-parameters control how the ML algorithm learns the model parameters during training, it's important to define optimization metrics and create SageMaker Hyper Parameter Tuning jobs to converge on the best combination of hyper-parameters. Based on ML build experience, AWS recommends using Bayesian hyper-parameter optimization strategy over manual, random, or grid search, as it usually yields better results using fewer computer resources.

For the talent and skilling industry use cases defined earlier, the DL models need to classify millions of jobs and skills to predict a good match and user learning sequence. The following details are some of the things that we found useful and were key in our thought leadership for creating AI solutions. We define the objective metric that the HPT job will try to optimize, which is validation accuracy for the talent and skilling use cases. The following is an example code snippet for the metrics definition:

```
objective_metric_name = "validation:accuracy"

metrics_definitions = [
    {"Name": "train:loss", "Regex": "loss: ([0-9\\.]+)"},
    {"Name": "train:accuracy", "Regex": "accuracy: ([0-9\\.]+)"},
    {"Name": "validation:loss", "Regex": "val_loss: ([0-9\\.]+)"},
    {"Name": "validation:accuracy", "Regex": "val_accuracy: ([0-9\\.]+)"},
]
```

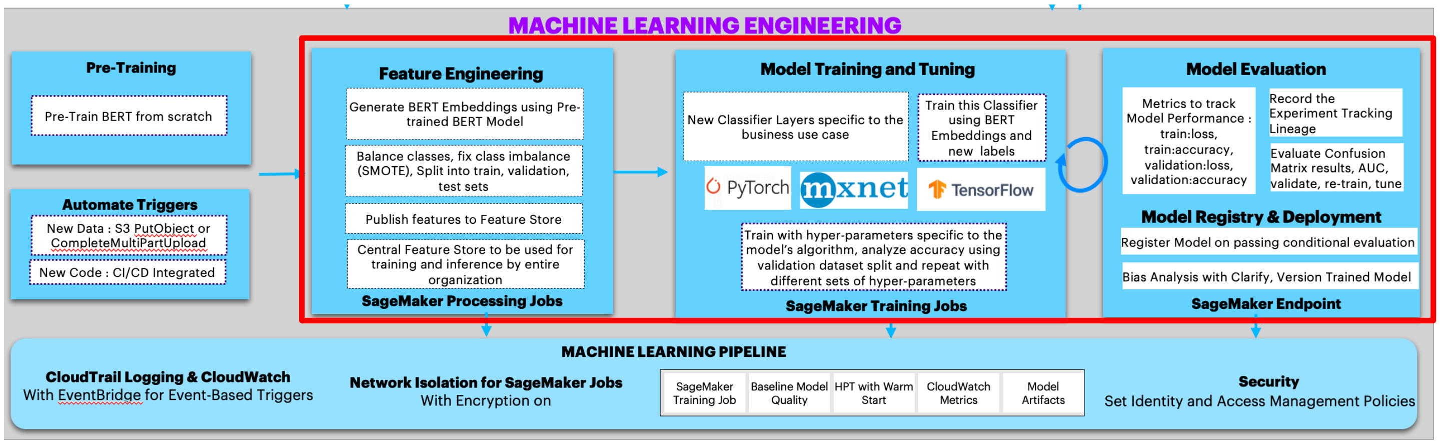
Next, we set the `HyperparameterTuner` with `estimator` and `Hyperparameter` ranges.

A crucial setting is the early_stopping_type, which you set so that SageMaker can stop the tuning job when it starts to overfit, and can help save cost of the overall tuning job. Inaccurate hyperparameter tuning can not only result in excessive costs, but also an ineffective model even after hours of training.

```
objective_metric_name = "validation:accuracy"

tuner = HyperparameterTuner(
    estimator=estimator,
    objective_type="Maximize",
    objective_metric_name=objective_metric_name,
    hyperparameter_ranges=hyperparameter_ranges,
    metric_definitions=metrics_definitions,
    max_jobs=2,
    max_parallel_jobs=10,
    strategy="Bayesian",
    early_stopping_type="Auto",
)
```

Combining all of this together, you have the following build and training process taking BERT as an example. Other DL models built with PyTorch, MXNet, or TensorFlow follow the same process. It is essential to get the following three stages (within the box under MACHINE LEARNING ENGINEERING) correct to move on to productionizing the system with large scale model deployments.


MACHINE LEARNING ENGINEERING
Pre-Training
Pre-Train BERT from scratch
Automate Triggers
New Data : S3 PutObject or CompleteMultiPartUpload
New Code : CI/CD Integrated
Feature Engineering
Generate BERT Embeddings using Pre-trained BERT Model
Balance classes, fix class imbalance (SMOTE), Split into train, validation, test sets
Publish features to Feature Store
Central Feature Store to be used for training and inference by entire organization
SageMaker Processing Jobs
Model Training and Tuning
New Classifier Layers specific to the business use case
Train this Classifier using BERT Embeddings and new labels
PyTorch
mxnet
TensorFlow
Train with hyper-parameters specific to the model's algorithm, analyze accuracy using validation dataset split and repeat with different sets of hyper-parameters
SageMaker Training Jobs
Model Evaluation
Metrics to track Model Performance : train:loss, train:accuracy, validation:loss, validation:accuracy
Record the Experiment Tracking Lineage
Evaluate Confusion Matrix results, AUC, validate, re-train, tune
Model Registry & Deployment
Register Model on passing conditional evaluation
Bias Analysis with Clarify, Version Trained Model
SageMaker Endpoint
MACHINE LEARNING PIPELINE
CloudTrail Logging & CloudWatch
With EventBridge for Event-Based Triggers
Network Isolation for SageMaker Jobs
With Encryption on
SageMaker Training Job
Baseline Model Quality
HPT with Warm Start
CloudWatch Metrics
Model Artifacts
Security
Set Identity and Access Management Policies

Complete ML engineering process and fine-tuning deep learning models

Model registry

It is important to catalog models to explain the model predictions and insights. It is also important that all models promoted to production are cataloged, all model versions managed, metadata such as training metrics are associated with a model, and the approval status of a model is managed. This is especially needed when organizations want to move from ad-hoc one-off proof-of-concepts to embedding AI in their enterprise systems with multiple teams, doing daily DL experiments. This is implemented in the solution using SageMaker Model Registry.

Optimization

DL is simple in essence. In the last few years, AWS has achieved astonishing results on machine-perception problems with the help of simple parametric models trained with gradient descent (GD). Extending that, all that is needed at the core is sufficiently large parametric models trained with GD on a large dataset.

Creating a DL algorithm or identifying the algorithm to use and fine-tune is the first step. The next step for an enterprise is to derive business value out of the algorithm. That can be achieved only when the models are appropriately industrialized, scaled, and continuously improved. Ill-performing models negatively impact a business or organization's bottom line. In Accenture's talent and skilling solution, there are over 50 models running in Production, making a large-scale, smooth, operationalization process a necessity.

Optimization drivers

DL has positioned itself as an AI revolution and is here to stay. Some of the benefits of using DL models are:

- Reusability
- Scalability

Optimizing and scaling ML and DL models in production is a crucial set of tasks, and one that must be done with finesse. To maximize the benefits listed previously, a proper implementation approach must be taken.

Following are details on how it should be implemented for the industry use cases, taking the example of a few of the models. The same approach can be used for scaling many other DL models for new problems.

Fine-tuning and reuse of models

Periodically, businesses get updated training data from market intelligence data sources on new market trends. There is always a need to optimize the hyper-parameters of the TensorFlow BERT classifier layer. For such cases, where the tuning job must be run again with an updated dataset or a new version of the algorithm, warm start with `TRANSFER_LEARNING` as the start type helps reuse the previous HPT job results, but along with new hyperparameters. This speeds up converging on the best model faster.

This is particularly important in Enterprise AI systems, as multiple teams may want to reuse the models created. Training DL models from scratch requires a lot of GPU, compute, and storage resources. Model reuse across the organization helps in reducing costs. Therefore, a useful technique for model reuse is fine-tuning. Fine-tuning methodology involves unfreezing a few of the top layers of a frozen model base for feature extraction, and then jointly training both the newly added part of the model, which is the fully connected classifier and top layers. With this, a model can be reused for a different problem, and does not have to be re-trained, saving costs for the company.

In the following sections, you will see how you can implement and scale the model fine-tuning strategies previously discussed, while maintaining a laser focus on the business metrics we need to attain.

`WarmStartConfig` uses one or more of the previous hyper-parameters tuning job runs called the *parent jobs,* and needs a `WarmStartType`.

```
from sagemaker.tensorflow import TensorFlow

estimator = TensorFlow(
    entry_point="tf_bert_reviews.py",
    source_dir="src",
    role=role,
    instance_count=train_instance_count,
    instance_type=train_instance_type,
    volume_size=train_volume_size,
    py_version="py37",
    framework_version="2.3.1",
    hyperparameters={
        "epochs": epochs,
        "epsilon": epsilon,
        "validation_batch_size": validation_batch_size,
        "test_batch_size": test_batch_size,
        "train_steps_per_epoch": train_steps_per_epoch,
        "validation_steps": validation_steps,
        "test_steps": test_steps,
        "use_xla": use_xla,
        "use_amp": use_amp,
        "max_seq_length": max_seq_length,
        "enable_sagemaker_debugger": enable_sagemaker_debugger,
        "enable_checkpointing": enable_checkpointing,
        "enable_tensorboard": enable_tensorboard,
        "run_validation": run_validation,
        "run_test": run_test,
        "run_sample_predictions": run_sample_predictions,
    },
    input_mode=input_mode,
    metric_definitions=metrics_definitions,
```

Setting up `HyperparameterTuner` with `WarmStartConfig`, including new hyper-parameter ranges.

```
objective_metric_name = "train:accuracy"
tuner = HyperparameterTuner(
    estimator=estimator,
    objective_type="Maximize",
    objective_metric_name=objective_metric_name,
    hyperparameter_ranges=hyperparameter_ranges,
    metric_definitions=metrics_definitions,
    max_jobs=2,
    max_parallel_jobs=1,
    strategy="Bayesian",
    early_stopping_type="Auto",
    warm_start_config=warm_start_config,
)
```

Scaling with distributed training

For efficient parallel computing during distributed training, employ both data parallelism and model parallelism. SageMaker supports distributed PyTorch. You can use the Hugging Face Transformers library that natively supports the SageMaker distributed training framework for both TensorFlow and PyTorch. The SageMaker built-in, distributed, all-reduce communication strategy should be used to achieve data parallelism by scaling PyTorch training jobs to multiple instances in a cluster.

Avoiding common missteps to reduce rework

The biggest driving factor in making a successful productionized ML project that has minimal to no amount of rework is a collaborative involvement between the ML team and the business unit.

Secondly, transforming data science prototype scripts from experimentation phase to modular performant code for production is a deeply involved task, and if not done right, will not produce a stable production system.

Finally, the ecosystem of ML engineering and MLOps is a culmination of multiple processes and standards from within DevOps, adding in ML-specific tooling and domain-specific elements, thereby building repeatable, resilient, production-capable data science solutions on the cloud. These three tenets alone distinguish a matured AI/ML enterprise from one that has just started in the journey of using ML for deriving business value.

For industry solutions, as mentioned in the *Workforce analytics use cases* section of this document, following are some optimizations that have proved useful to having an efficient, enterprise-grade, end-to-end, industrialized, ML solution:

- Remove monolithic prototype scripts
- Identify difficult-to-test code in large, tightly coupled code bases
- Introduce effective encapsulation and abstraction techniques

In a scaled, industrialized, production version, the full end-to-end automated data engineering and ML engineering pipeline is the product built on the data science scripts in the experimentation phase.

Machine learning pipelines

Despite many companies going all in on ML, hiring massive teams of highly compensated data scientists, and devoting huge amounts of financial and other resources, their projects end up failing at high rates.

Moving from a single laptop setup toward a scalable, production-grade, data science platform is a completely different challenge from the proof-of-concept stage, and arguably, one of the most difficult, as it involves collaborating with different teams across an organization. Accenture has devised a scalable and unique approach for the Accenture talent and skilling AI solutions discussed in this whitepaper, to go from prototype to full-scale productionized systems in a very short period of time; enhancing “speed to market” and generating value.

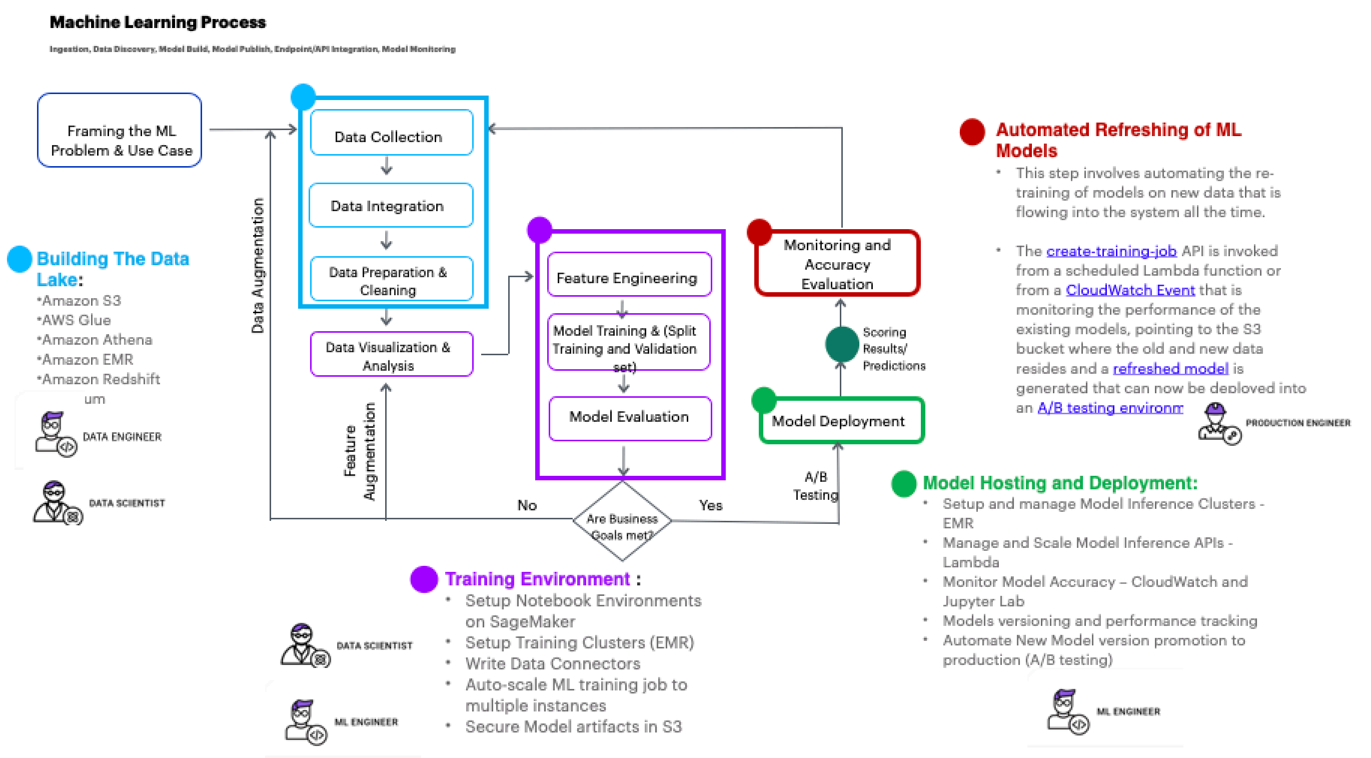
ML technical debt may rapidly accumulate. Without strict enforcement of ML engineering principles (built on rigorous software engineering principles) to data science code may result in a messy pipeline, and managing these pipelines – detecting errors and recovering from failures – becomes extremely difficult and costly. Comprehensive live monitoring of system behavior in near real time, combined with automated response, is critical for long-term system reliability.

This and the following sections address these problems, and provide a solution.

Going from POC to large-scale deployments

The main challenges for companies looking to move beyond the realm of basic AI proof of concepts (POCs), manual data science POCs, and pilot programs to Enterprise AI, can be grouped around the need to achieve the following at Enterprise level:

- Repeatability
- Scalability
- Transparency/Explainability


Machine Learning Process
Ingestion, Data Discovery, Model Build, Model Publish, Endpoint/API Integration, Model Monitoring
Framing the ML Problem & Use Case
Data Collection
Data Integration
Data Preparation & Cleaning
Data Visualization & Analysis
Data Augmentation
Feature Augmentation
Feature Engineering
Model Training & (Split Training and Validation set)
Model Evaluation
Are Business Goals met?
No
Yes
A/B Testing
Model Deployment
Scoring Results/ Predictions
Monitoring and Accuracy Evaluation
Building The Data Lake:
•Amazon S3
•AWS Glue
•Amazon Athena
•Amazon EMR
•Amazon Redshift
DATA ENGINEER
DATA SCIENTIST
Training Environment :
Setup Notebook Environments on SageMaker
Setup Training Clusters (EMR)
Write Data Connectors
Auto-scale ML training job to multiple instances
Secure Model artifacts in S3
DATA SCIENTIST
ML ENGINEER
Automated Refreshing of ML Models
PRODUCTION ENGINEER
Model Hosting and Deployment:
Setup and manage Model Inference Clusters - EMR
Manage and Scale Model Inference APIs - Lambda
Monitor Model Accuracy – CloudWatch and Jupyter Lab
Models versioning and performance tracking
Automate New Model version promotion to production (A/B testing)
ML ENGINEER

- This step involves automating the re-training of models on new data that is flowing into the system all the time.
- The create-training-job API is invoked from a scheduled Lambda function or from a CloudWatch Event that is monitoring the performance of the existing models, pointing to the S3 bucket where the old and new data resides and a refreshed model is generated that can now be deployed into an A/B testing environm

ML workflow and process with multiple teams that need to collaborate to create a complete ML solution in production

Applying software engineering principles to data science

The single biggest driving factor in making a successful project that has the fewest amount of rework is that of collaborative involvement between the ML team and the business unit. The second biggest factor to ensure success is communication within the ML teams.

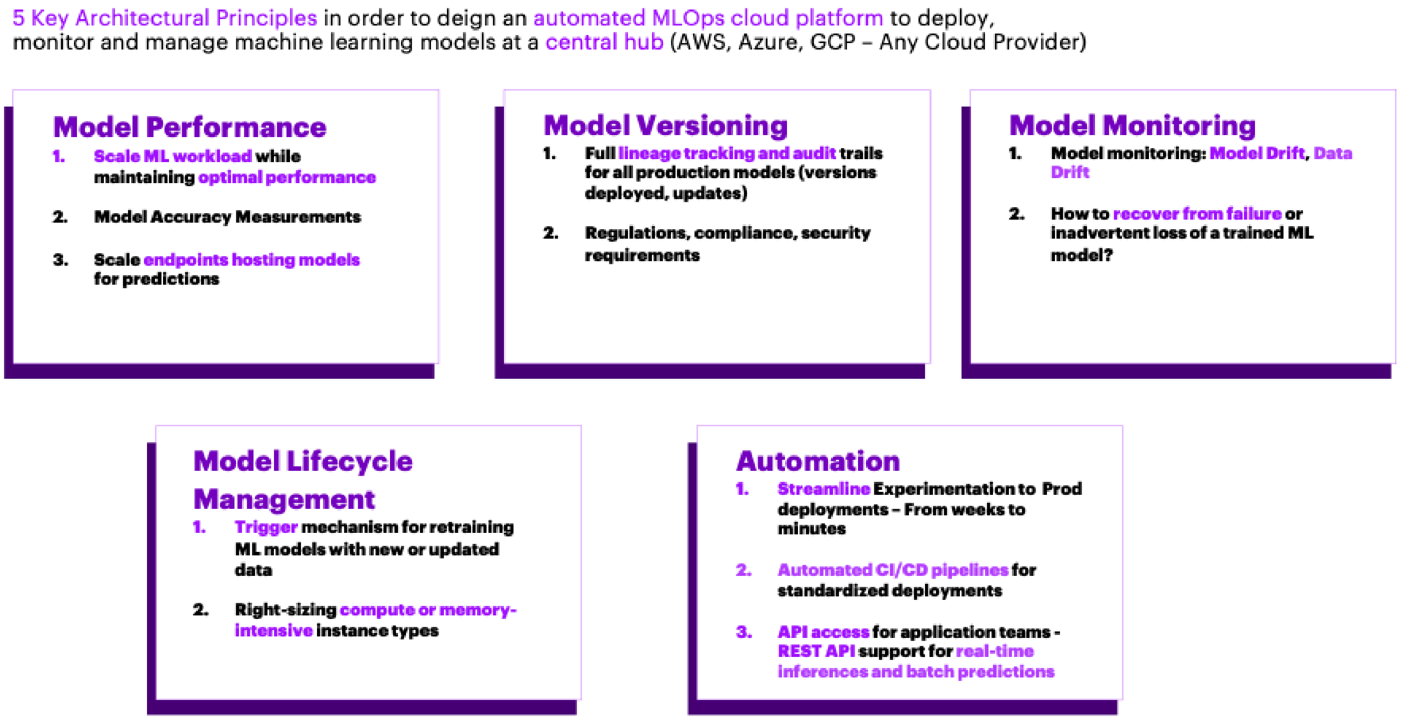
Decades ago, software engineers refined their processes from large-scale waterfall implementations to a more flexible agile process. That was a game-changer in terms of bringing ideas to market faster and more efficiently. Similarly, now ML engineering seeks to define a new set of practices and tools that will optimize the whole unique realm of software development for data analysts and scientists.

The idea of ML engineering and MLOps as a paradigm is rooted in the application of similar principles that DevOps has to software development.

Machine learning automation through pipelines

For the Accenture workforce productivity and talent and skilling use cases listed in this whitepaper, you see that automated, repeatable, reproducible, parametrized data engineering and ML engineering pipelines help to track model runs, lineage, versions, and timings across all the stages, from data ingestion to feature engineering to model training and post-deployment.

Automated ML pipelines ensure one vital feature and requirement of a stable live enterprise AI platform: repeatability and reproducibility. Instead of manual ad-hoc Python scripts that may propagate data quality issues from sources, pipelines help ensure the issues are caught, handled, retried, or logged at every step of the pipeline.



5 Key Architectural Principles in order to deign an automated MLOps cloud platform to deploy, monitor and manage machine learning models at a central hub (AWS, Azure, GCP – Any Cloud Provider)

Model Performance

1. Scale ML workload while maintaining optimal performance
2. Model Accuracy Measurements
3. Scale endpoints hosting models for predictions

Model Versioning

1. Full lineage tracking and audit trails for all production models (versions deployed, updates)
2. Regulations, compliance, security requirements

Model Monitoring

1. Model monitoring: Model Drift, Data Drift
2. How to recover from failure or inadvertent loss of a trained ML model?

Model Lifecycle Management

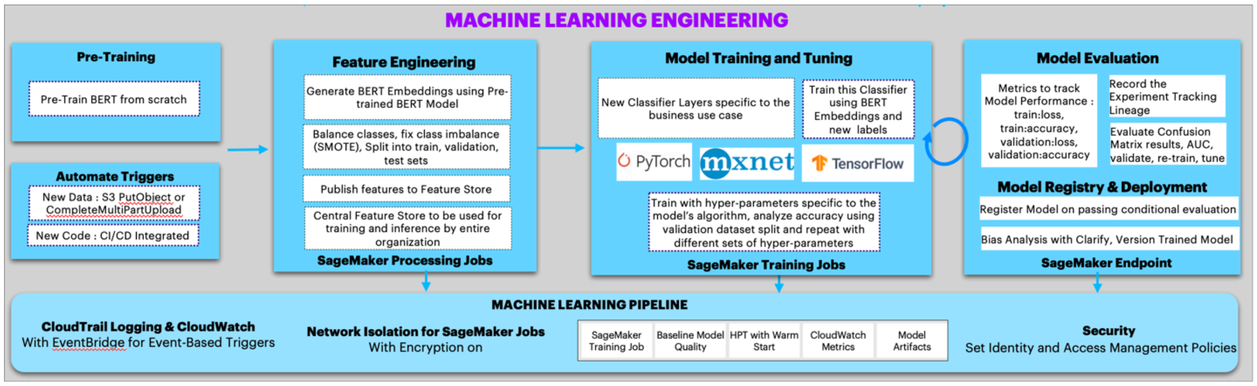
1. Trigger mechanism for retraining ML models with new or updated data
2. Right-sizing compute or memory-intensive instance types

Automation

1. Streamline Experimentation to Prod deployments – From weeks to minutes
2. Automated CI/CD pipelines for standardized deployments
3. API access for application teams - REST API support for real-time inferences and batch predictions

The design principles for a functioning MLOps platform

Implementing these design principles, the following diagram summarizes a modern and effective way to implement pipelines on AWS. This section describes the key ML processes, and the stages for creating a modern and performant ML pipeline on AWS.

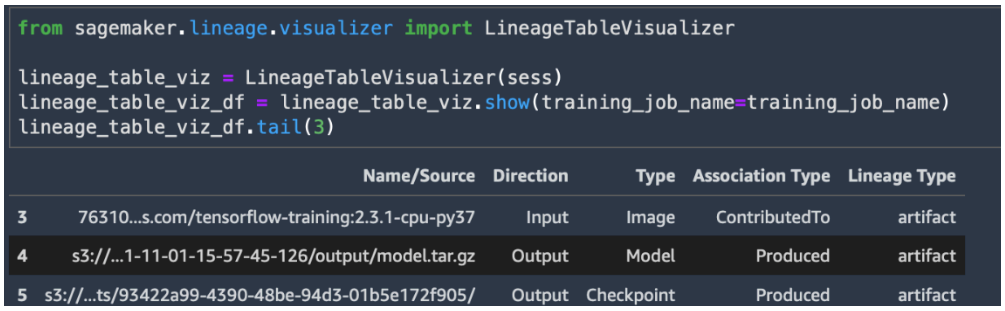

MACHINE LEARNING ENGINEERING
Pre-Training
Pre-Train BERT from scratch
Automate Triggers
New Data : S3 PutObject or CompleteMultiPartUpload
New Code : CI/CD Integrated
Feature Engineering
Generate BERT Embeddings using Pre-trained BERT Model
Balance classes, fix class imbalance (SMOTE), Split into train, validation, test sets
Publish features to Feature Store
Central Feature Store to be used for training and inference by entire organization
SageMaker Processing Jobs
Model Training and Tuning
New Classifier Layers specific to the business use case
Train this Classifier using BERT Embeddings and new labels
PyTorch
mxnet
TensorFlow
Train with hyper-parameters specific to the model's algorithm, analyze accuracy using validation dataset split and repeat with different sets of hyper-parameters
SageMaker Training Jobs
Model Evaluation
Metrics to track Model Performance : train:loss, train:accuracy, validation:loss, validation:accuracy
Record the Experiment Tracking Lineage
Evaluate Confusion Matrix results, AUC, validate, re-train, tune
Model Registry & Deployment
Register Model on passing conditional evaluation
Bias Analysis with Clarify, Version Trained Model
SageMaker Endpoint
MACHINE LEARNING PIPELINE
CloudTrail Logging & CloudWatch With EventBridge for Event-Based Triggers
Network Isolation for SageMaker Jobs With Encryption on
SageMaker Training Job
Baseline Model Quality
HPT with Warm Start
CloudWatch Metrics
Model Artifacts
Security
Set Identity and Access Management Policies

The ML engineering process and steps to create modern and effective ML pipelines on AWS

Tracking lineage

Model artifact lineage should be recorded and tracked at every stage. This is the only way you can tie back the model to the dataset it was trained on, the features used, the validation dataset used, and the approval process and deployment history. For tracking and looking at the lineage artifacts for our processing job, you can do the following in SageMaker, and the same should be set up as part of the pipeline artifact lineage.

SageMaker Lineage Tracking API has lineage traversal, contexts, and associations among other concepts that help in creating a lineage history for all stages of SageMaker Pipelines, training jobs, endpoints, SageMaker processing jobs, and models, as shown in the following image. Lineage of feature engineering jobs can and should be tracked in the same way. Amazon SageMaker ML Lineage Tracking helps in storing information about all the stages of a ML workflow from data preparation to model deployment.



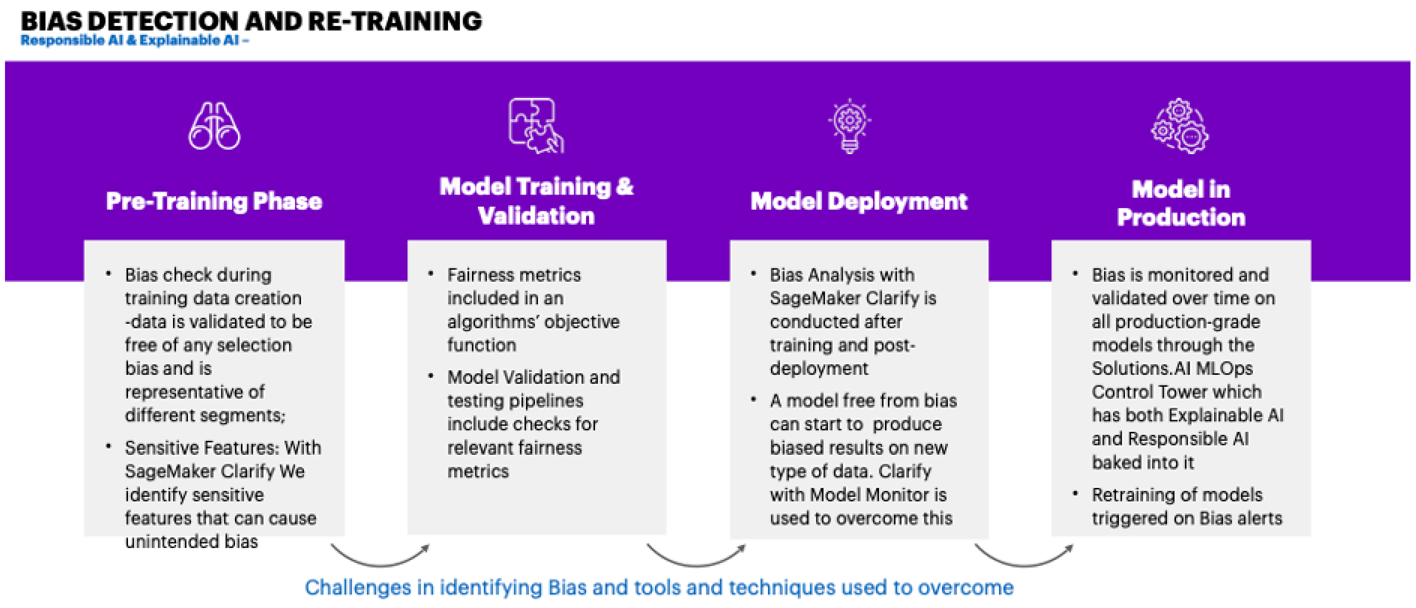
```
from sagemaker.lineage.visualizer import LineageTableVisualizer

lineage_table_viz = LineageTableVisualizer(sess)
lineage_table_viz_df = lineage_table_viz.show(training_job_name=training_job_name)
lineage_table_viz_df.tail(3)
```

	Name/Source	Direction	Type	Association Type	Lineage Type
3	76310...s.com/tensorflow-training:2.3.1-cpu-py37	Input	Image	ContributedTo	artifact
4	s3://...1-11-01-15-57-45-126/output/model.tar.gz	Output	Model	Produced	artifact
5	s3://...ts/93422a99-4390-48be-94d3-01b5e172f905/	Output	Checkpoint	Produced	artifact

Tracking model lineage and pipeline lineage with Amazon SageMaker

Monitoring for performance and bias


BIAS DETECTION AND RE-TRAINING
Responsible AI & Explainable AI –
Pre-Training Phase
Model Training & Validation
Model Deployment
Model in Production
Challenges in identifying Bias and tools and techniques used to overcome

Pre-Training Phase

- Bias check during training data creation -data is validated to be free of any selection bias and is representative of different segments;
- Sensitive Features: With SageMaker Clarify We identify sensitive features that can cause unintended bias

Model Training & Validation

- Fairness metrics included in an algorithms' objective function
- Model Validation and testing pipelines include checks for relevant fairness metrics

Model Deployment

- Bias Analysis with SageMaker Clarify is conducted after training and post-deployment
- A model free from bias can start to produce biased results on new type of data. Clarify with Model Monitor is used to overcome this

Model in Production

- Bias is monitored and validated over time on all production-grade models through the Solutions.AI MLOps Control Tower which has both Explainable AI and Responsible AI baked into it
- Retraining of models triggered on Bias alerts

Bias detection with SageMaker Clarify and nearly continuous monitoring with SageMaker Model Monitor

DL models can be heavily impacted by data bias. Model and data bias detection and rectification should be constant underlying themes in an Enterprise AI system. You can use SageMaker Clarify extensively to perform evaluation for detecting data bias, during featuring engineering for evaluating feature importance, for assessing model bias during training/hyper-parameter tuning and finally with SageMaker Model Monitor to take actions on live models.

An important need is to generate reports throughout all these stages to maintain transparency of the process. Using SageMaker Data Wrangler along with SageMaker Clarify, you can generate reports that explain the features that are considered important, the choices made by the model, and the reasoning behind the predictions.

Data drift, covariate, label drifts, and concept shift mandate nearly continuous monitoring and updates to models. Nearly continuous monitoring of model insights, measuring them, and testing their production effectiveness are critical steps to achieve a successful Enterprise AI. Baseline metrics first need to be calculated during model training and thresholds should be set for Kullback–Leibler (KL), Kolmogorov-Smirnov (KS), and Linear Programming (LP) Key Performance Indicators (KPIs).

SageMaker Model Monitor detects drift on live model output which can be integrated with SageMaker Clarify for various detections and takes corrective actions, such as re-training models, introducing new production variants, and retiring older non-performing models.

Because our focus is always on the end-state, the models and all supporting infrastructure and data feeds that are employed to solve a problem need to have a measurable characteristic tied to business outcomes.

Post-training bias metrics

The following metrics are helpful for detecting and explaining our model predictions:

- Proportions in Predicted Labels (DPPL)
- Disparate Impact (DI)
- Difference in Conditional Acceptance (DCAcc)
- Difference in Conditional Rejection (DCR)

For all post-training data and model bias metrics, you can use SageMaker Clarify. Taking the trained model, you choose the feature to analyze for bias and determine the conditional rejection (DCR) and other metrics listed previously. Because SageMaker Clarify supports SHAP, you can determine the contribution of each feature to the model prediction. This helps in performing feature attribution and generating the model explainability report with SageMaker Clarify. This provides you with all the information you need for detecting drifts in feature attribution and model explainability on live model endpoints – a key business goal in ensuring you have responsible AI baked in into all your solutions.

Monitoring performance

Data quality monitoring

Earlier, this whitepaper described a process of creating data-quality baseline using Deequ, which helps detect drift in the statistical characteristics of input data being sent to the live model. If this step is not accurately performed in the ML and DL pipelines, the rest of the components downstream are deeply impacted, resulting in incorrect or sub-optimal results.

Dealing with drifts

Models are bound to drift; it is only a matter of time. It can be gradual or rapid, based on the circumstances. There are various types of drifts to deal with once your models are in production.

- **Feature drift, Label drift and Concept drift**, which can be handled by either revisiting feature engineering, model re-training, or training on new data.
- **Prediction drift and Feedback drift**, which require more involved approach of new versions of the models as otherwise it may impact the business objectives.

Regardless of the types and causes of drifts, it is vital to monitor model metrics associated with training for passive retraining, and the model attribution data for active retraining. Monitoring these shifts in model efficacy can help with early intervention (Model Monitor), explainable analytics reports (SageMaker Clarify), and the ability to resolve the issue in a way that will not cause disruption to the project.

Augmented AI

While you have multiple continually running automated ML pipelines that save the final batch predictions and insights in a curated zone, it is essential to insert human supervision and guidance in the automated AI/ML workflows. Humans can provide the necessary critical quality assurances before pushing sensitive models into production to help the models learn better. Use Amazon Augmented AI (A2I) along with SageMaker Ground Truth to bring together ML and humans, to provide automation while continuously improving your models.

With Amazon SageMaker, A2I, Ground Truth, Amazon CloudFront, and Amazon Cognito, you have a fully functional web application validation system that lets your functional domain experts and technical leads validate samples of model predictions, which in turn helps quickly optimize and fine-tune your predictive models.

Human-in-the-loop workflows

In the workforce productivity use cases, one important business requirement is an unbiased, efficient, and effective model for assessing a person's current skills based on CV, professional profile, so on. While you can use custom entity recognition and customize BERT layers (not just the classifier top layers) for roles, skills, titles, and so on, it is essential to integrate human oversight in the entire workflow involving many models in the pipeline (custom models, Amazon Textract, Amazon Comprehend, fine-tuned BERT) and decision-making process, so that your models learn better.

It is important to know how to make this workflow function well, as it will impact the business outcomes if an appropriate automated workflow is not put in place to fix low-confidence predictions and improve the models. For example, in the industry use case, there has been a regular need to parse a document with a person's work history and professional skills, which can be in virtually any format and predicting skill proficiency scores.

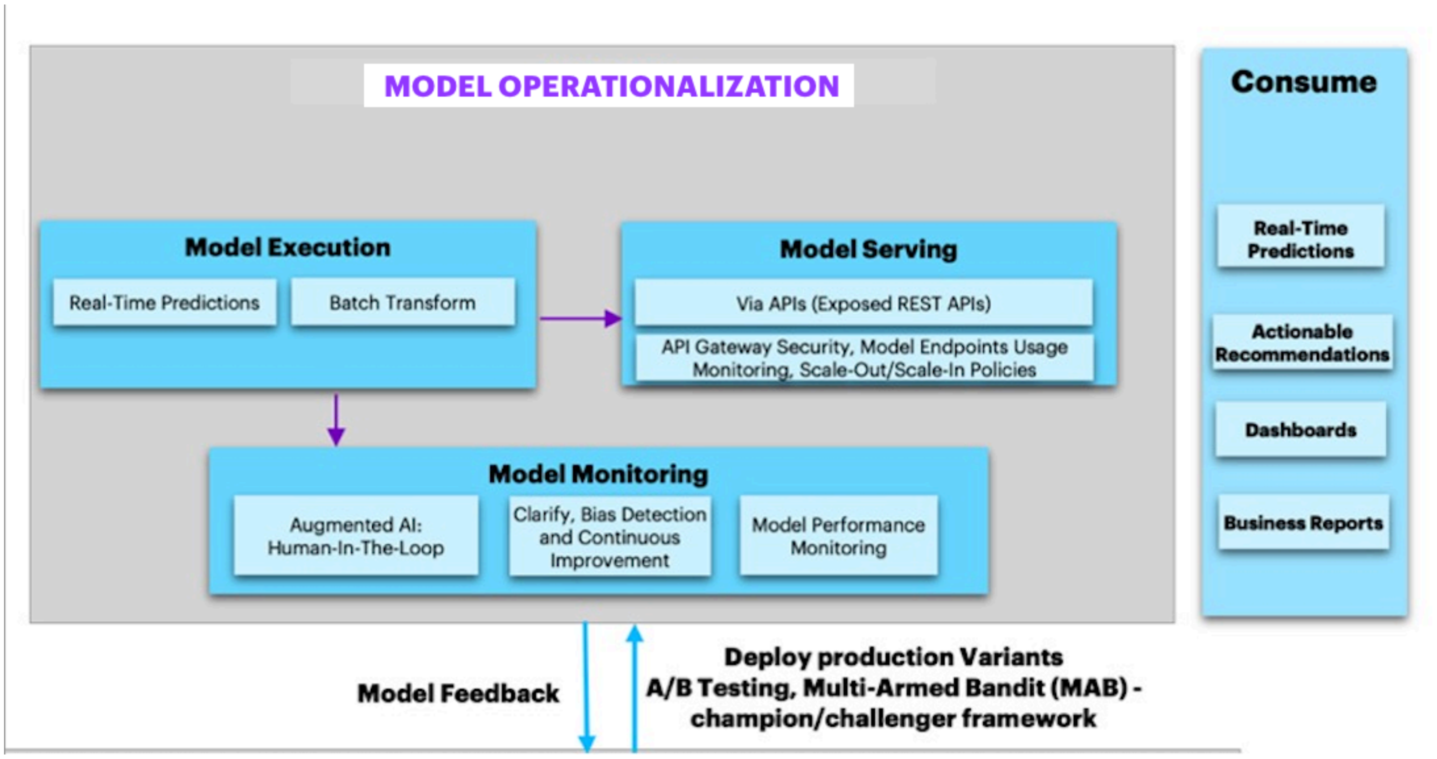
Using Amazon Textract (which offers AI powered extraction of text and structured data from documents) and a series of models in an inference pipeline, AWS was able to provide the recommended insights. This also allowed AWS to integrate human judgement into the workflow wherever needed with A2I, to help the models learn better and improve over time.

A front-end web application integrated with augmented AI instills confidence in the predictions and recommendations being made by the models. This is especially important in creating and maintaining a matured, productionized, enterprise AI system such as the ones in our use case. The Ground Truth labels that were acquired using A2I and human-in-the-loop workflow are actively used with SageMaker Model Monitor to continuously evaluate concept drift. For all other drifts in model bias, feature importance, and explainability, use Model Monitor to compare against the baseline provided to each SageMaker deployed endpoint.

Updating model versions

Post-deployment of our models, it is important to track business metrics that we are aiming to optimize. The business metrics are usually tied to revenue generation, sales, number of user clicks, purchases, or as in cases found in this whitepaper, on the efficacy of the workforce productivity recommendations or sales forecast shown to C-suite users. In this solution, AWS recommends storing the business metrics in Amazon DynamoDB.

When new versions of the models are available, it is important to see which variant performs better before routing all traffic to the new model. Models tested during the engineering process and the same models running “in the wild” may produce different results, and you don’t want to switch your use base to the new models without first testing them sufficiently with live, unseen, real traffic. Use Canary testing, A/B testing, and Multi-Armed Bandit (MAB) strategies in a champion-challenger framework, to choose the best new model among the various SageMaker hosted production variants.


MODEL OPERATIONALIZATION
Model Execution
Real-Time Predictions
Batch Transform
Model Serving
Via APIs (Exposed REST APIs)
API Gateway Security, Model Endpoints Usage Monitoring, Scale-Out/Scale-In Policies
Model Monitoring
Augmented AI: Human-In-The-Loop
Clarify, Bias Detection and Continuous Improvement
Model Performance Monitoring
Consume
Real-Time Predictions
Actionable Recommendations
Dashboards
Business Reports
Model Feedback
Deploy production Variants A/B Testing, Multi-Armed Bandit (MAB) - champion/challenger framework

Deploying production variants and continuous MLOps in production

Conclusion

Taking data science algorithms from experimentation to large-scale production requires an end-to-end ML engineering process in place to maintain and manage an Enterprise AI system. This becomes a more involved process for scaling DL models, as in the use cases described here. This whitepaper describes an approach to scale ML and DL models in production and reduce the time from “ideation to production”, while minimizing cost and maximizing gains for the organization.

Contributors

Contributors to this document include:

- Nitu Nivedita, Senior Manager - Solutions.AI, Accenture
- Paul A. Barrett, Managing Director - Solutions.AI, Accenture
- Soonam Kurian, Senior Solutions Architect, Amazon Web Services

Further reading

- Data Science on AWS by Chris Fregly and Antje Barth - Released April 2021, Publisher(s): O'Reilly Media, Inc.
- Machine Learning Engineering in Action by Ben Wilson, Manning Publications, Published March 2022, ISBN 9781617298714
- Deep Learning with Python by François Chollet, Manning Publications; Published November 2017, ISBN 9781617294433
- AWS Architecture Center

About Accenture

Accenture is a global professional services company with leading capabilities in digital, cloud and security. Combining unmatched experience and specialized skills across more than 40 industries, we offer Strategy and Consulting, Technology and Operations services and Accenture Song — all powered by the world's largest network of Advanced Technology and Intelligent Operations centers. Our 699,000 people deliver on the promise of technology and human ingenuity every day, serving clients in more than 120 countries. We embrace the power of change to create value and shared success for our clients, people, shareholders, partners, and communities. Visit us at accenture.com.

Document revisions

To be notified about updates to this whitepaper, subscribe to the RSS feed.

Change	Description	Date
Initial publication	Whitepaper published.	July 27, 2022

Notices

AWS Glossary

For the latest AWS terminology, see the AWS glossary in the *AWS Glossary Reference*.