AWS *Enhanced Prep plan*

Contents

AWS ML Engineer Associate Curriculum Overview

- *Define quantifiable success Criteria of a ML model*
- *Learn components of SageMaker Studio*
    - *Jupyter notebook, schedule job to run Jupyter notebook*
    - *Canvas*
    - *Data (Prepare with Wrangler, Feature Store, EMR Cluster)*
    - *Jobs (training, model evaluation)*
    - *Pipeline*
    - *Model/Model Registry)*
    - *Jumpstart*
    - *Deployment or CI/CD (Inference recommender, Endpoints or Projects)*
- *Additional Things to know: Algorithms - Supervised vs Unsupervised, LLM, Responsible AI/ML, SageMaker Documentation)*

  SageMaker Documentation
    - *Feature Store*
    - *AutoML*
    - *Studio*
    - *Jupyter notebook*

Domain 1: Data Processing

1.1 Collect, Ingest, and Store Data

1.1.1 COLLECT DATA

High-Performing data

- **REPRESENTATIVE**
    - **Best practice**: *When building an ML model, it's important to feed it high-quality data that accurately reflects the real world. For example, if 20 percent of customers typically cancel memberships after a year, the data should represent that churn rate. Otherwise, the model could falsely predict that significantly more or fewer customers will cancel.*
    - **Watch for**: *If your data doesn't actually reflect the real-world scenarios that you want your model to handle, it will be difficult to identify meaningful patterns and make accurate predictions.*
- **RELEVANT**
    - **Best practice**: *Data should contain relevant attributes that expose patterns related to what you want to predict, such as membership duration for predicting cancellation rate.*
    - **Watch for**: *If irrelevant information is mixed in with useful data, it can impact the model's ability to focus on what really matters. For example, a list of customer emails in a dataset that's supposed to predict membership cancellation can negatively impact the model's performance.*
- **Feature Rich**
    - **Best practice**: *Data should include a complete set of features that can help the model learn underlying patterns. You can identify additional trends or patterns to increase accuracy by including as much relevant data as possible.*
    - **Watch for**: *Data that has limited features reduces the ability of the ML algorithm to accurately predict customer churn. For example, if the data consists of a small set of customer details and omits important data, like demographic information, it will lose accuracy and miss opportunities for detecting patterns in cancellation rate.*
- **Consistent**
    - **Best practice**: *Data must be consistent when it comes to attributes, such as features and formatting. Consistent data provides more accurate and reliable results.*
    - **Watch for**: *If the datasets come from various data sources that contain different formatting methods or metadata, the inconsistencies will impact the algorithm's ability to effectively process the data. The algorithm will be less accurate with the inconsistent data.*

Types of Data

Text

Text data, such as documents and website content, is converted to numbers for use in ML models, especially for natural language processing (NLP) tasks like sentiment analysis. Models use this numerical representation of text to analyze the data.

Tabular

Tabular data refers to information that is organized into a table structure with rows and columns, such as the data in spreadsheets and databases. Tabular data is ideal for linear regression models.

Time series

Time-series data is collected over time with an inherent ordering that is associated with data points. It can be associated with sensor, weather, or financial data, such as stock prices. It is frequently used to detect trends. For instance, you might analyze and forecast changes using ML models to make predictions based on historical data patterns.

Image

Image data refers to the actual pixel values that make up a digital image. It is the raw data that represents the colors and intensities of each pixel in the image. Image data, like data from photos, videos, and medical scans, is frequently used in machine learning for object recognition, autonomous driving, and image classification.

Formatting data

- ***Structured***
- ***Unstructured***
- ***Semi-structured***

Data formats and file types

1. Row-based data format

- *common in relational databases and spreadsheets.*
- *It shows the relationships between features*

customerID	name	age	email	last_support	subscription_active
123456789	Rosalez, Alejandro	32	alejandro_rosalez@example.com	1/11/22	false
87654321	Candella, Pat	22	pat_candella@example.com	3/26/24	true

Row-based file types

- ***CSV***

Comma-separated values (CSV) files are lightweight, space-efficient text files that represent tabular data. Each line is a row of data, and the column values are separated by commas. The simple CSV format can store different data types like text and numbers, which makes it often used for ML data. However, the simplicity of CSV comes at the cost of performance and efficiency compared to columnar data formats with more optimized for analytics.

- ***Avro RecordIO***

Avro RecordIO is a row-based data storage format that stores records sequentially. This sequential storage benefits ML workloads that need to iterate over the full dataset multiple times during model training. Additionally, Avro RecordIO defines a schema that structures the data. This schema improves data processing speeds and provides better data management compared to schema-less formats.

2. Column-based data format

- *format, queries extract insights from patterns within a column rather than the entire record, which results in efficient analysis of trends across large datasets.*

Column-based file types

- ***Parquet***

Parquet is a columnar storage format typically used in analytics and data warehouse workloads that involve large data sets. ML workloads benefit from columnar storage because data can be compressed, which improves both storage space and performance.

- ***ORC***

Optimized row columnar (ORC) is a columnar data format similar to Parquet. ORC is typically used in big data workloads, such as Apache Hive and Spark. With the columnar format, you can efficiently compress data and improve performance. These performance benefits make ORC a widely chosen data format for ML workloads.

3. Object-notation data

- *Object notation fits non-tabular, hierarchical data, such as graphs or textual data.*
- *Object-notation data is structured into hierarchical objects with features and key-value pairs rather than rows and columns.*

Object-based file types

- ***JSON***

JavaScript Object Notation (JSON) is a document-based data format that is both human and machine readable. ML models can learn from JSON because it has a flexible data structure. The data is compact, hierarchical, and easy to parse, which makes it suitable for many ML workloads.

JSON is represented in objects and arrays.

*An **object** is data defined by key-value pairs and enclosed in braces {}. The data can be a string, number, Boolean, array, object, or null*	*An **array** is a collection of values enclosed in square brackets [ ] and can contain values that are separated by commas. The following array consists of multiple objects.*

{
"customerID": 123456789,
"name": "Rosalez, Alejandro",
"age": 32,
"email": "alejandro_rosalez@example.com",
"lastSupportInteraction": "1/11/22",
"subscriptionActive": false
}
	[
	{
	"customerID": 123456789,
	"name": "Rosalez, Alejandro",
	"age": 32,
	"email": "alejandro_rosalez@example.com",
	"lastSupportInteraction": "1/11/22",
	"subscriptionActive": false
	},
	{
	"customerID": 87654321,
	"name": "Candella, Pat",
	"age": 22,
	"email": "pat_candella@example.com",
	"lastSupportInteraction": "3/26/24",
	"subscriptionActive": true
	}
	]

- ***JSONL***

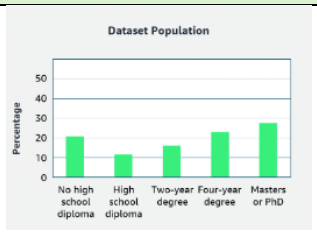
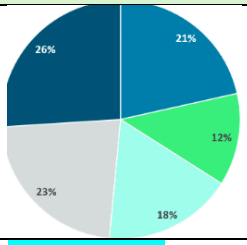
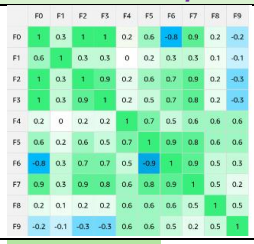
JavaScript Object Notation Lines (JSONL) is also called newline-delimited JSON. It is a format for encoding JSON objects that are separated by new lines instead of being nested. Each JSON object is written on its own line, such as in the following example.

{“customerID”: “12345678”. “name”: “Rosalez, Alejandro”, “age”: “32”, “email”:
“alejandro_rosalez@example.com”, “last_support”: “1/12/22”, “subscription_active”:
“false”}
{“customerID”: “87654321”. “name”: “Candella, Pat”, “age”: “22”, “email”:
“pat_candella@example.com”, “last_support”: “3/26/24”,“subscription_active”: “true”}

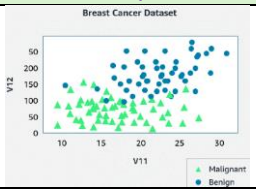
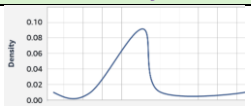
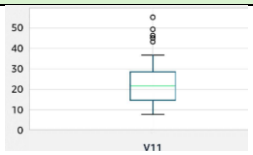
JSONL improves efficiency because individual objects can be processed without loading a larger JSON array. This improved efficiency when parsing objects results in better handling of large datasets for ML workloads. Additionally, JSONL structure can map to columnar formats like Parquet, which provides the additional benefits of those file types.

1.1.4 Graphs for data visualization

Categorical data

	Bar Charts	Pie Charts	Heat maps
Graphs			
Used for	***Comparison* analysis**	***Composition* analysis**	***Relationship* analysis**
Features	*proportion of a dataset for specific attributes.*	*Entirety of the dataset.*	*use color to depict patterns and relationships*

Numerical Data

	Scatterplots	***Histograms***	*Density Plots*	*Box Plots*
Grapgh				
Features		*Data divided into bins*	*Similar to histograms,* • *smooth the distribution of data* • *don't constrain the data to bins.*	*displaying the location of key data points, such as median, quartiles, and outliers*
Assists with	***Relationship*** *analysis*	*Distribution analysis*	*Distribution analysis*	*Distribution analysis*
Useful for	*identify distinct regions*	*overall behavior of a single feature*	• *distribution of a single feature* • *No bins but continuous distribution*	• *quickly comparing distributions* • *identifying skewness, spread, and outliers.*

1.1.2 STORE DATA

Data Storage Options

1. S3

***Features**: S3 serves as a central data lake for ingesting, extracting, and transforming data to and from other AWS services used for processing tasks. These tasks are an integral part of most ML workloads. The ability to store and retrieve data from anywhere makes Amazon S3 a key component in workflows requiring scalable, durable, and secure data storage and management.*

***Considerations**: S3 provides scalability, durability, and low cost, but it has higher latency compared to local storage. For latency-sensitive workloads, S3 might not be optimal. When deciding if S3 meets your needs, weigh its benefits against potential higher latency. With caching and proper architecture, many applications achieve excellent performance with S3, but you must account for its network-based nature.*

Use cases

- ***Data ingestion and storage***
  S3 can be used to store large datasets required for ML. Data can be ingested into S3 through streaming or batch processing. The data in S3 can then be used for ML training and inference. The scalability and durability of S3 makes it well-suited for storing the large volumes of data for effective machine learning.
- ***Model training and evaluation***
  S3 stores ML datasets and models. It provides versioning to manage different model iterations, so you can store training and validation data in S3. You can also store trained ML models. With versioning, you can manage and compare models to evaluate performance.

- ***Integration with other AWS services***
  S3 serves as a centralized location for other AWS services to access data. For example,
  - *SageMaker can access Amazon S3 data to train and deploy ML models.*
  - *Kinesis can stream data into S3 buckets for ingestion.*
  - *AWS Glue can connect to data stored in S3 for data processing purposes.*

2. EBS

***Features**: Amazon EBS is well-suited for databases, web applications, analytics, and ML workloads. The service integrates with Amazon SageMaker as a core component for ML model training and deployment. By attaching EBS volumes directly to Amazon EC2 instances, you can optimize storage for ML and other data-intensive workloads.*

***Considerations**: EBS separates storage from EC2 instances, requiring more planning to allocate and scale volumes across instances. Instance stores simplify storage by tying storage directly to the EC2 instance lifecycle. This helps to avoid separate volume management. Although EBS offers flexibility, instance stores provide streamlined, intrinsic storage management than EBS.*

Use cases

- ***High-performance storage***
  EBS provides high-performance storage for ML applications requiring fast access to large datasets. EBS offers volumes with high IOPS for quick data reads and writes. The high throughput and IOPS accelerate ML workflows and applications.
- ***Host pre-trained models***
  With EBS, you can upload, store, and access pre-trained ML models to generate real-time predictions without setting up separate hosting infrastructure.

3. EFS

***Features**:*

- *The service is designed to grow and shrink automatically as files are added or removed, so performance remains high even as file system usage changes.*
- *EFS uses the NFSv4 networking protocol to allow compute instances access to the file system across a standard file system interface. You can conveniently migrate existing applications relying upon on-premises NFS servers to Amazon EFS without any code changes.*

***Considerations**: EFS has higher pricing, but offers streamlined scaling of shared file systems. EBS offers lower costs, but there are potential performance limitations based on workload. Consider if the higher EFS costs and potential performance variability are acceptable trade-offs compared to potentially lower EBS costs but workload-dependent performance excellent performance with S3, but you must account for its network-based nature.*

Use cases

- ***Concurrent access***
  EFS allows multiple EC2 instances to access the same datasets simultaneously. This concurrent access makes Amazon EFS well-suited for ML workflows that require shared datasets across multiple compute instances.
- ***Shared datasets***
  EFS provides a scalable, shared file system in the cloud that eliminates the need for you to copy large datasets to each compute instance. Multiple instances can access data, such as ML learning libraries, frameworks, and models, simultaneously without contention. This feature contributes to faster model training and deployment of ML applications.

4. Amazon FSx

***Features**:*

- *Amazon FSx offers a rich set of features focused on reliability, security, and scalability to support ML, analytics, and high-performance computing applications.*
- *The service delivers millions of IOPS with sub-millisecond latency so you can build high-performance applications that require a scalable and durable file system.*

***Considerations**: When using Amazon FSx for ML workloads, consider potential tradeoffs. Certain file system types and workloads can increase complexity and management needs. Tightly coupling the ML workflow to a specific file system also risks vendor lock-in, limiting future flexibility.*

Use cases

- ***Two types of file systems***
  FSx is a fully managed service that provides two types of file systems: Lustre and Windows File Server. Lustre allow for high-performance workloads requiring fast storage, such as ML training datasets.

- ***Distributed architecture***
  Lustre's distributed architecture provides highly parallel and scalable data access, making it ideal for hosting large, high-throughput datasets used for ML model training. By managing infrastructure operations, including backups, scaling, high availability, and security, you can focus on your data and applications rather than infrastructure management.

Model output Storage Options

1. Training Workloads

Training workloads require high performance and frequent random I/O access to data.

- *EBS volumes are well-suited for providing the random IOPS that training workloads need. Additionally, Amazon EBS instance store volumes offer extremely low-latency data access. This is because data is stored directly on the instances themselves rather than on network-attached volumes.*

2. Inference Workloads

Need fast response times for delivering predictions, but usually don't require high I/O performance, except for real-time inference cases.

- *EBS gp3 volumes or EFS storage options are well-suited for meeting these needs.*
- *For increased low-latency demands, upgrading to EBS io2 volumes can provide improved low-latency capabilities.*

3. Real-time and streaming workloads

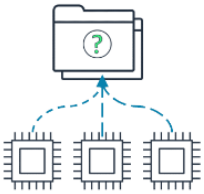
- *EFS file systems allow low latency and concurrent data access for real-time and streaming workloads. By sharing the same dataset across multiple EC2 instances, EFS provides high throughput access that meets the needs of applications requiring real-time data sharing.*

4. Dataset storage

- *S3 can be used for storing large datasets that do not need quick access, such as pretrained ML models, or data that is static or meant for archival purposes.*

Data Access Patterns

There are three common data access patterns in ML: copy and load, sequential streaming, and randomized access.

Copy and load *Data is copied from S3 to a training instance backed by EBS.*	***Sequential streaming*** *Data is streamed to instances as batches or individual records, typically from S3 to instances backed by EBS volumes.*	***Randomized access*** *Data is randomly accessed, such as with a shared file system data store, like FSx and EFS.*
		

Cost

Cost comparison

- ***S3** has the lowest cost for each gigabyte of storage based on storage classes. Storage classes are priced for each gigabyte, frequency of access, durability levels, and for each request.*
- ***EBS** has network attached storage, which is more expensive per gigabyte than Amazon S3. However, it provides lower latency, storage snapshots, and additional performance features that might be useful for ML workloads.*
- ***EFS** is a managed file service with increased costs that can link multiple instances to a shared dataset. Cost structure is designed around read and write access and the amount of gigabyte used with different storage tiers available.*
- ***FSx** pricing depends on the file system used. General price structure is around storage type used for each gigabyte, throughput capacity provisioned, and requests.*

AWS Tools for Reporting and Cost Optimization

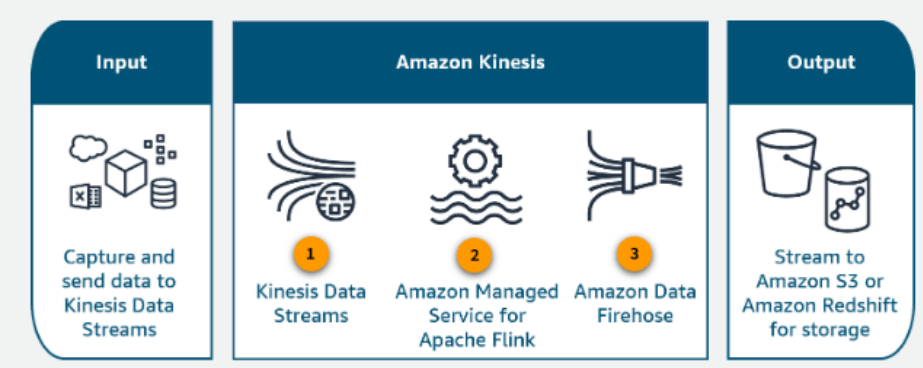
AWS provides several reporting and cost-optimization tools:

- *AWS Cost Explorer – See patterns in AWS spending over time, project future costs, identify areas that need further inquiry, observe Reserved Instance utilization, observe Reserved Instance coverage, and receive Reserved Instance recommendations.*
- *AWS Trusted Advisor – Get real-time identification of potential areas for optimization.*
- *AWS Budgets – Set custom budgets that trigger alerts when cost or usage exceed (or are forecasted to exceed) a budgeted amount. Budgets can be set based on tags and accounts as well as resource types.*
- *CloudWatch – Collect and track metrics, monitor log files, set alarms, and automatically react to changes in AWS resources.*
- *AWS CloudTrail – Log, continuously monitor, and retain account activity related to actions across AWS infrastructure at low cost.*
- *S3 Analytics – Automated analysis and visualization of S3 storage patterns to help you decide when to shift data to a different storage class.*
- *AWS Cost and Usage Report – Granular raw data files detailing your hourly AWS usage across accounts used for Do-It-Yourself (DIY) analysis (e.g., determining which S3 bucket is driving data transfer spend). The AWS Cost and Usage Report has dynamic columns that populate depending on the services you use.*

1.1.3 Data Ingestion

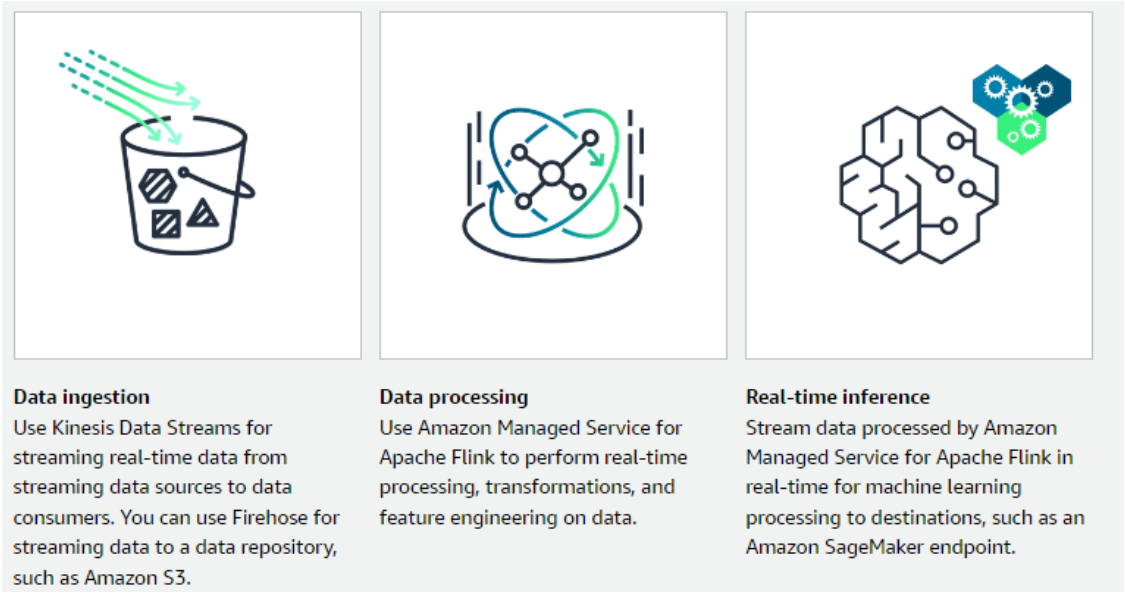
Realtime Ingestion - streaming services

Amazon Kinesis vs MSK vs Firehose



- ***Kinesis Data Streams*** *is primarily used for ingesting and processing data.*
- ***Firehose*** *provides a streamlined method of streaming data to data storage locations.*
- ***Amazon Managed Service for Apache Flink*** *provides consumption of streaming data using Apache Kafka in real-time for analysis.*

Streaming use cases



Data ingestion
Use Kinesis Data Streams for streaming real-time data from streaming data sources to data consumers. You can use Firehose for streaming data to a data repository, such as Amazon S3.

Data processing
Use Amazon Managed Service for Apache Flink to perform real-time processing, transformations, and feature engineering on data.

Real-time inference
Stream data processed by Amazon Managed Service for Apache Flink in real-time for machine learning processing to destinations, such as an Amazon SageMaker endpoint.

Data Extraction

Extraction

- ***Amazon S3 Transfer Acceleration***

Amazon S3 Transfer Acceleration uses CloudFront edge locations to accelerate large data transfers to and from S3. These transfers can help speed up data collection for ML workloads that require moving large datasets. S3 Transfer Acceleration overcomes bottlenecks like internet bandwidth and distance that can limit transfer speeds when working with large amounts of data.

- ***DMS***

AWS Data Migration Service (AWS DMS) facilitates database migration between databases or to Amazon S3 by extracting data in various formats, such as SQL, JSON, CSV, and XML. Migrations can run on schedules or in response to events for frequent data extraction. With AWS DMS, you can migrate databases between many sources and targets.

- ***AWS DataSync***

With AWS DataSync, you can efficiently transfer data between on-premises systems or AWS services by extracting data from sources, such as data file systems or network-attached storage. You can then upload data to AWS services like Amazon S3, Amazon EFS, Amazon FSx, or Amazon RDS on a scheduled or event-driven basis. DataSync facilitates moving large datasets to the cloud while reducing network costs and data transfer times.

- ***AWS Snowball***

AWS Snowball is a physical device service used to transfer large amounts of data into and out of AWS when network transfers are infeasible. Snowball devices efficiently and cost-effectively move terabytes or petabytes of data into S3

Storage

- ***S3***

With S3 serving as a highly scalable object storage service, data used for ML projects can be spread out across storage locations. Storage can be extracted and transferred to and from S3 with other AWS services. These other services include Amazon S3 Transfer Accelerator, AWS CLI, AWS SDK, AWS Snowball, AWS DataSync, AWS DMS, AWS Glue, and AWS Lambda.

- ***EBS***

EBS volumes provide storage for ML data. This data can be copied to services such as Amazon S3 or Amazon SageMaker, using tools like the AWS Management Console, AWS CLI, or AWS SDKs to manage volumes. EBS volumes store the necessary data that is then extracted and moved to other AWS services to meet ML requirements.

- ***EFS***

*Amazon EFS allows creating shared file systems that can be accessed from multiple EC2 instances, so you can share data across compute resources. You can extract data from **EFS** using AWS CLI, AWS SDKs, or with services like AWS Transfer Family and DataSync that facilitate data transfers. Amazon EFS provides the capability to share data from Amazon EC2 instances while also providing tools to conveniently move the data to other services.*

- ***RDS***

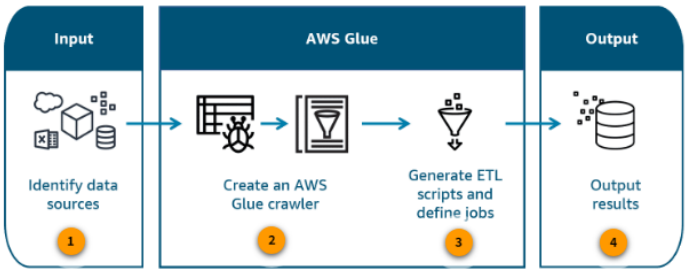
Amazon Relational Database Service (Amazon RDS) provides relational databases that can be accessed through AWS services like AWS DMS, the AWS CLI, and AWS SDKs to extract and transfer data. Amazon RDS is a common source for extracting relational data because it offers managed database instances that streamline data access.

- ***DynamoDB***

Amazon DynamoDB is a fully managed NoSQL database service provided by AWS. You can extract data using various AWS services like AWS DMS, AWS Glue, and AWS Lambda. You can use programmatic tools, such as the AWS CLI and AWS SDK, to process and analyze the data outside of DynamoDB. Data extraction allows DynamoDB data to be integrated with other platforms for further processing.

Data Merging

1. ***AWS Glue is a fully managed ETL service that you can use to prepare data for analytics and machine learning workflows.***



Best for: Glue works for ETL workloads from varied data sources into data lakes like Amazon S3.

Steps

a) Identify: Identify data sources: *AWS Glue can be used to combine or transform large datasets using Apache Spark. It can efficiently process large structured and unstructured datasets in parallel. AWS Glue integrates with services like S3, Redshift, Athena, or other JDBC compliant data stores.*

b) ***Create an AWS Glue crawler:*** *AWS Glue crawlers scan data and populate the AWS Glue Catalog.*

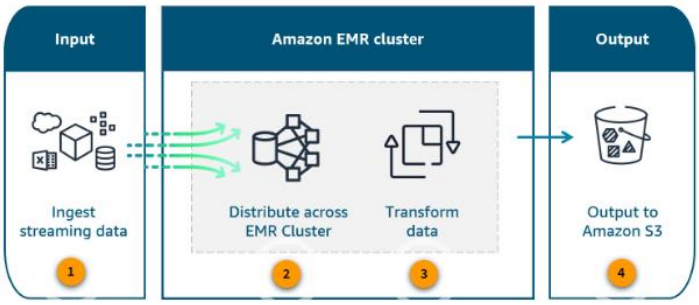
c) Generate ETL scripts and define jobs: *Jobs run the ETL scripts to extract, transform, and load the data, which can start on demand or can be scheduled to run at specific intervals.*

d) ***Clean and transformed data*** *is written back to S3 or to another data store, such as Amazon Redshift.*

2. ***Amazon EMR***

Amazon EMR is a service for processing and analyzing large datasets using open-source tools of big data analysis, such as Apache Spark and Apache Hadoop. It applies ETL methodologies to ensure the product is flexible and scalable. Amazon EMR integrates data from multiple sources into one refined platform, making the transformation of data cost-effective and quick.

Best for: Amazon EMR is best suited for processing huge datasets in the petabyte range.



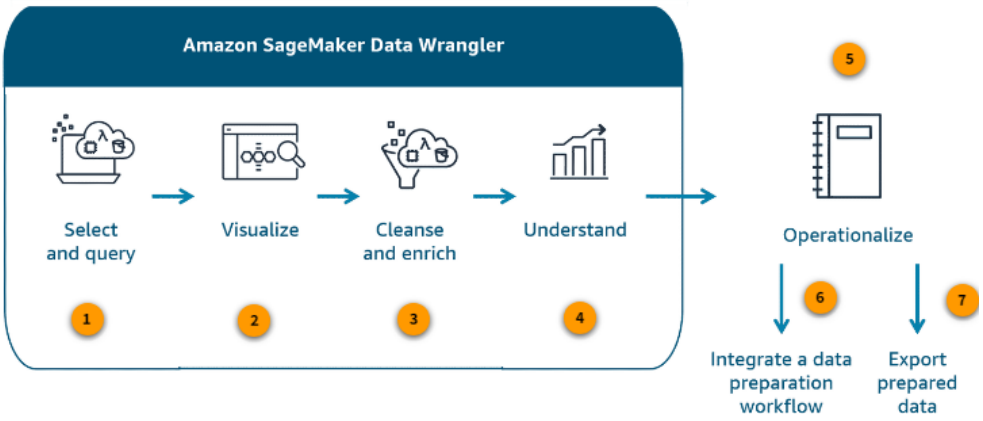
STEPS

- ***Ingest streaming sources:*** *ETL is done using Apache Spark Streaming APIs. This makes it possible to source data in real time from places such as Apache Kafka and Amazon Kinesis. Data is received and combined in real time.*
- ***Distribute across EMR cluster:*** *EMR clusters are made up of various nodes, each of which are configured specifically to handle parallel tasks and data processing.*
- ***Generate ETL scripts and define jobs:*** *At the end of the data processing lifecycle, you can use the Python, Scala, or SQL development environments. These environments give you powerful, flexible methods for scripting data workflows and for making data filtering, transformation, and aggregation more convenient.*

- ***Output to Amazon S3:*** *After processing and transforming the data, the results are outputted in an Amazon S3 bucket.*

- ***Amazon SageMaker Data Wrangler***

  Amazon EMR is a service for processing and analyzing large datasets using open-source tools of big data analysis, such as Apache Spark and Apache Hadoop. It applies ETL methodologies to ensure the product is flexible and scalable. Amazon EMR integrates data from multiple sources into one refined platform, making the transformation of data cost-effective and quick.



3.Clean and enrich
Cleanse and explore data, perform feature engineering with built-in data transforms, and detect statistical bias with Amazon SageMaker Clarify.
6.Integrate a data preparation workflow
Use Amazon SageMaker Pipelines to integrate a data preparation workflow.
7.Export prepared data
Export data to SageMaker Feature Store or S3.

When to use which (Data Wrangler vs Glue vs EMR)

Feature	AWS Glue	Amazon EMR	SageMaker Data Wrangler
Purpose	Serverless ETL service	Big data processing platform	ML-focused data preparation
Ease of Use	Visual ETL designer	Requires cluster setup and management	Visual interface, no coding required
Ideal Vol.	• Medium to large datasets	• Ideal for very large datasets	• Small to medium datasets
ML Integration	Can prepare data for ML, but not specialized	Can run ML frameworks, but requires setup	Tightly integrated with SageMaker ML workflow
Ideal Use Cases	• Batch data transformations • Data catalog management • Serverless data integration	• Batch and real-time processing • Complex big data processing	• Quick data exploration and visualization • Data cleaning and transformation for ML models

Troubleshooting

Scalability issues

- ***Capacity issues with data destinations***

With EFS, FSx, and S3, you can seamlessly scale storage increasing or decreasing in size.

- ***Latency issues, IOPs, and data transfer times***

High latency, insufficient IOPs, or slow data transfer times significantly impact performance of storage systems and data ingestion. Network bottlenecks, undersized provisioned storage volumes, or using inefficient methods of ingestion can arise from these factors.

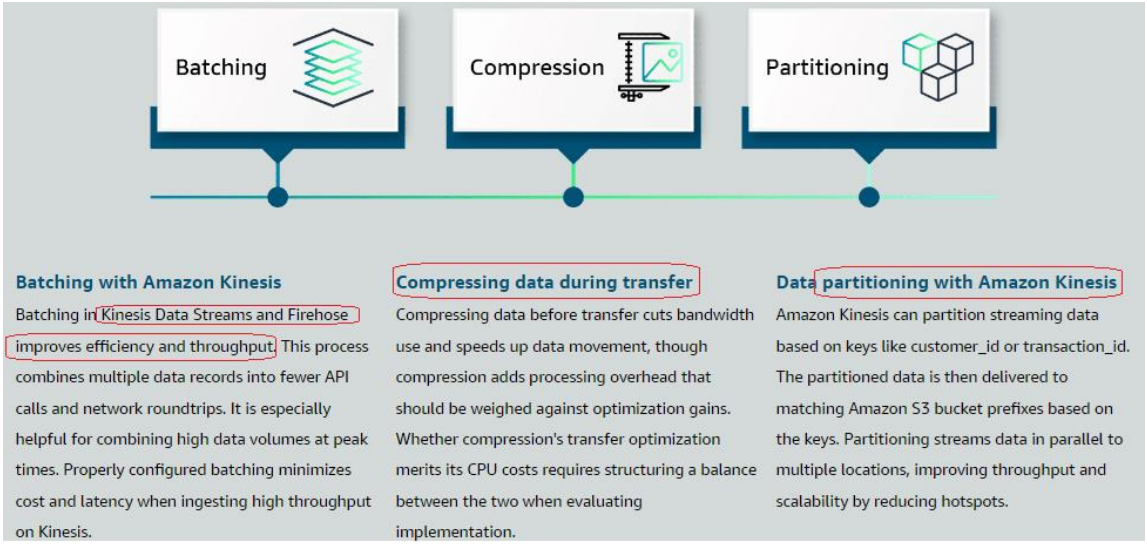
Consider optimizing network configurations or use AWS services with improved performance capabilities, such as provisioned IOPS volumes for EBS. Using techniques, such as compression or batching, can also lead to improved data-transfer efficiency.

- ***Uneven distribution of data access***

Hotspots or bottlenecks in storage systems can be caused by uneven distribution of data access resulting in performance degradation or data availability issues.

AWS services, such as S3 and EFS, automatically distribute data and provide load balancing capabilities. Data partitioning is another strategy that can be implemented, which distributes data across multiple storage resources, reducing the likelihood of hotspots.

Ingestion modifications



DO THE ASSESSMENT!!

1.1.4 Summary

Keywords/Concepts	AWS Service/Option
• Data lake, central storage, scalable, durable • Large static datasets, archival	Amazon S3
High-performance storage, real-time predictions, pre-trained models	Amazon EBS
• Concurrent access, shared datasets, scalable file system • Real-time/streaming workloads	Amazon EFS
High-performance computing, distributed architecture, Lustre	Amazon FSx
Training workloads, high IOPS, random access	• EBS (especially io2) • Instance Store
Inference workloads (standard)	• EBS gp3 • EFS
Inference workloads (low-latency)	EBS io2
Copy and load pattern Sequential streaming pattern	S3 to EBS
Randomized access pattern	FSx or EFS
Columnar storage, data compression	Parquet, ORC
Row-based storage, sequential records	CSV, Avro RecordIO
Hierarchical data, flexible structure	JSON, JSONL

Data Types and Processing

Data Type	Suitable For	Not Suitable For
Tabular data	Linear regression, classification	Complex pattern recognition in unstructured data
Columnar data	Data analysis, efficient querying	Frequent record updates
Time series data	Trend analysis, forecasting	Non-temporal pattern recognition
Image data	Object recognition, image classification	Text-based analysis
Text data	Natural language processing, sentiment analysis	Image or numerical analysis

Data Formats

Format	Suitable For	Storage Type	Characteristics
CSV	*Simple tabular data, easy human readability*	*Row-based*	*space-efficient*
Avro RecordIO	*ML workloads requiring multiple dataset iterations*		*schema-defined*
Parquet	*Large-scale data analysis*	*columnar*	*Efficient compression, fast querying*
ORC	*Big data analysis (Hive, Spark)*		*Optimized for large-scale data processing*
JSON/JSONL	*Hierarchical, non-tabular data*	*Hierarchical*	*Flexible structure, easy parsing*

Data Visualization

See Above

AWS Storage Options

Service	Advantages	Limitations
Amazon S3	*Scalable, durable, central data lake*	*Higher latency compared to local storage*
Amazon EBS	*High IOPS, suited for databases and ML training*	*Limited to single EC2 instance, requires volume management*
Amazon EFS	*Concurrent access, scalable shared file system*	*Higher costs, performance dependent on network speed*
Amazon FSx	*Lowest latency, high-performance computing*	*Potential vendor lock-in, complex management for some file system types*

Data Access Patterns

Pattern	Best With	Characteristics
Copy and Load	*S3 to EBS*	*Data copied entirely before processing*
Sequential Streaming	*S3 to EBS*	*Data streamed in batches or individual records*
Randomized Access	*EFS, FSx*	*Random data access, shared file system*

Use Case Recommendations

Use Case	Recommended Service	Reason
Training Workloads	• *EBS (io2)* • *Instance Store*	*High IOPS, low-latency random access*
Inference Workloads (standard)	• *EBS (gp3)* • *EFS*	*Balance of performance and cost*
Inference Workloads (low-latency)	*EBS (io2)*	*Higher IOPS for faster response times*
Real-time/Streaming Workloads	*EFS*	*Concurrent access, shared datasets*
Large Static Datasets	*S3*	*Cost-effective for infrequently accessed data*
Distributed Processing	• *EFS* • *FSx*	*Concurrent access from multiple instances*

1.2 Transform Data (Data Cleaning, Categorical encoding, Feature Engineering)

Remember

- ***Data cleaning*** *focuses on handling issues like missing data and outliers.*
- ***Categorical encoding*** *- Used to convert values into numeric representations.*
- ***Feature engineering*** *focuses on modifying or creating new features from the data, rather than encoding features.*

1.2.1 Data Cleaning

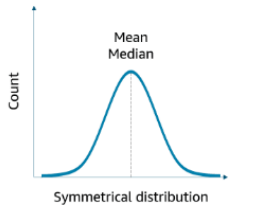
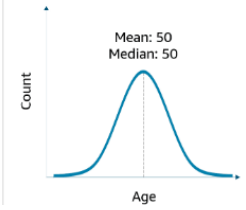
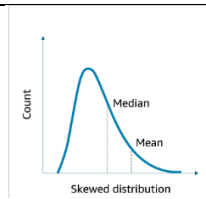
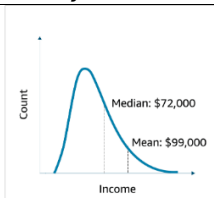
Incorrect and Duplicate Data

deduplication *- The process of automating data duplication removal. Deduplication works by scanning datasets for duplicate information, retaining one copy of the information, and replacing the other instances with pointers that direct back to the stored copy. Deduplication can drastically increase storage capacity by keeping only unique information in your data repositories.*

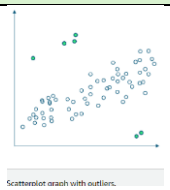
Data Outliers

Methods

1. Calculating mean and median

Mean	*Median*
The mean is the average of all the values in the dataset. Mean can be a useful method for understanding your data when the data is symmetrical. *For example, a symmetrical set of data that contains ages of respondents might reflect that both the mean and median of the dataset is 50 years old.*	*The median is the value in the dataset that divides the values into two equal halves.* *If your data is skewed or contains outliers, the median tends to provide the better metric for understanding your data as it relates to central tendency.* *For example, a dataset that contains income level might contain outliers. The mean might skew toward higher or lower values, while the median would provide a more accurate picture of the data's central tendency.*
 	 

2. Identifying natural and artificial outliers

Natural outliers	***Artificial outliers***	
Natural outliers are data points that are accurate representations of data, but are extreme variations of the central data points. For example, in a dataset that includes height measurements of individuals, an extremely tall individual would represent a natural outlier.	*Artificial outliers are anomalous data points in your dataset due to error or improper data collection. For example, a faulty sensor in a thermometer might produce a body temperature that is unrealistically high or low compared to expected body temperatures.*	

Participant ID	Age	Income	Education	State	Flu shot	Outbreak zone?
123456	39	45,000/year	Four-year degree	NY	Yes	No
123457	23	1,500/month	Baccalauréat	MN	No	Yes
123458	78		Masters/PhD		Yes	Yes
123459	20	3,000,000/year	High-school diploma	CA	No	No
123450	154	53,000/year		Masters/PhD		
123456	39	45,000/year	Four-year degree	NY	Yes	No

1 ***Artificial outlier -*** *This data is in the correct format for the Age column, but an entry of 154 is unrealistic. In this case, it makes the most sense to delete this entry from your data.*

2 ***Natural outlier*** *- Although three million dollars a year is drastically more than the rest of the salaries in our dataset, this number is still plausible. In this case, you can choose to transform the outlier and reduce the outlier's influence on the overall dataset. You will learn about that type of data transformation later in this course.*

Incomplete and Missing Data

There are some key steps you can take to address incomplete and missing values in your dataset.

a) Identify missing values

There are certain Python libraries, such as Pandas, that you can use to check for missing values.

b) Determine why values are missing

Before you can determine how to treat the missing values, it's important to investigate which mechanisms caused the missing values. The following are three common types of missing data:

- ***Missing at Random (MAR):*** *The probability that a data point is missing depends only on the observed data, not the missing data.*

  Example*: In a dataset of student test scores, scores are missing for some students who were absent that day. Absence is related to performance.*

- ***Missing Completely at Random (MCAR):*** *The probability that a data point is missing does not depend on the observed or unobserved data.*

  Example*: In an employee survey, some people forgot to answer the question about their number of siblings. Their missing sibling data does not depend on any values.*

- ***Missing Not at Random (MNAR):*** *The probability that a data point is missing depends on the missing data itself.*

  Example: In a financial audit, companies with accounting irregularities are less likely to provide complete records. The missing data depends on the sensitive information being withheld.

- ***Drop missing values***

Depending on what is causing your missing values, you will decide to either drop the missing values or impute data into your dataset.

One of the most straightforward ways to deal with missing values is to remove the rows of data with missing values. You can accomplish this by using a Pandas function. Dropping rows or columns will make the dataset non-missing. However, the risk of dropping rows and columns is significant.

Issues

  - *If you have hundreds of rows or columns of data, all of that missing data might cause bias in your model predictions.*
  - *If you drop too much data, you might not have enough features to feed the model.*

- ***Impute values***

Missing values might be related to new features that haven't included your dataset yet. After you include more data, those missing values might be highly correlated with the new feature. In this case, you would deal with missing values by adding more new features to the dataset. If you determine the values are missing at random, data imputation, or inputting the data into your dataset, is most likely the best option.

One common way to impute missing values is to replace the value with the mean, median, or most frequent value. You would select the mean, median, or most frequent value for categorical variables. You would select the mean or median for numerical variables. Choosing the mean, median, or most frequent value depending on your business problem and data collection procedures.

1.2.2 Categorical encoding

Categorical encoding is the process of manipulating text-based variables into number-based variables.

When to encode

Not all categorical variables need to be encoded. Depending on your use case, different ML algorithms might not require you to encode your variables.

For instance, a random forest model can handle categorical features directly. You would not need to encode values, such as teal, green, and blue, as numeric values.

Encoding Types (or types of Categorical values)

- ***Binary** categorical values refer to values that are one of two options. For example, a column of data might be true or false, such as if someone attended an event.*
- ***Nominal**, or multi-categorical, values are category values where order does not matter. A set of data that contains different geographic locations, such as states or cities, might be considered multi-categorical.*
- ***Ordinal**, or ordered, values are category values where the order does matter, like what size of drink you order at a coffee shop: small, medium, or large.*

Encode Techniques

Not all categorical variables

- ***Label encoding** converts categorical values into binary numbers*

Category value	Encoded value
Teal	0
Blue	1
Green	2

- ***One Hot encoding** : creating a new binary feature for each unique category value. Rather than assigning a different value for each data point like binary encoding, one-hot encoding sets the feature to 1 or 0 depending on if the category that applies to a given data point.*

Category value	Teal	Blue	Green
Teal	1	0	0
Blue	0	1	0
Green	0	0	1

***When to use which:** **Label coding** might not be the best technique if there are a lot of categories. In One hot encoding, these additional columns might grow your dataset so much that it makes it difficult to analyze efficiently.*

1.2.3 Feature Engineering

Feature engineering is a method for transforming raw data into more informative features that help models better capture underlying relationships in the data.

Feature Engineering by data type (numeric, text, image, and sound data types)

We only cover numeric and text here

***Numeric feature engineering** involves transforming numeric values for the model and is often accomplished by grouping different numeric values together.*

***Text feature** engineering involves transforming text for the model and is often accomplished by splitting the text into smaller pieces.*

1. ***Numerical Feature Engineering***
   - ***Purpose:*** *Aims to transform the numeric values so that all values are on the same scale.*
   - ***Why:*** *This method helps you to take large numbers and scale them down, so that the ML algorithm can achieve quicker computations and avoid skewed results.*

   a) ***Feature Scaling****:*

Normalization	***Standardization***
rescales the values (often between 0 and 1)	*Similar, but **mean** of 0 and **standard deviation** of 1*
	When to use: Reduces the negative effect of outliers.

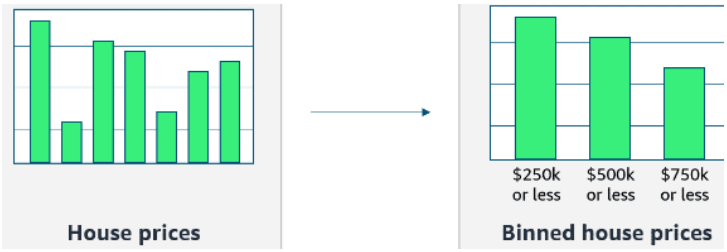
Price (in dollars)	price_scaled
100,000	0.1
437,000	0.437
1,000,000	1

Price (in dollars)	price_scaled
100,000	-1.15
437,000	0
1,000,000	1.15

   b) ***Binning****:*

   The data is divided into these bins based on value ranges, thus transforming a numeric feature into a categorical one.

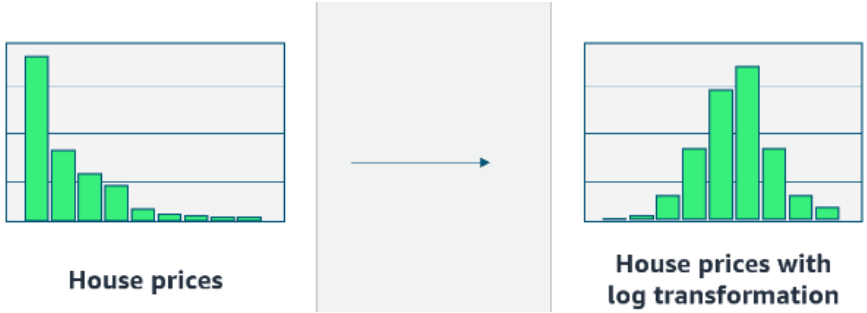
   When*: numerical data when the exact difference in numbers is not important, but the general range is a factor.*



   c) ***Log transformation****:*

   The most common logarithmic functions have bases of 10 or e, where e is approximately equal to 2.71828. Logarithmic functions are the inverse of exponential functions and are useful for modeling phenomena where growth decreases over time, like population growth or decay.

   When*: When skewed numeric data, or multiple outliers. Basically, log compresses data to use lower numbers*



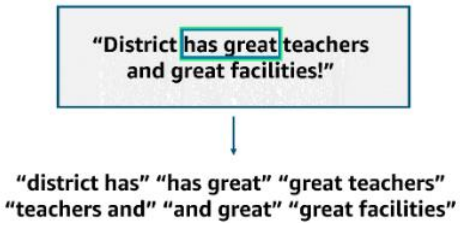
   For example, the log of $10,000 would be around 4 and the log of $10,000,000 would be around 7. Using this method, the outliers are brought much closer to the normal values in the remainder of the dataset.

2. *Text Engineering*

a) ***Bag of Words**: The bag-of-words model does not keep track of the sequence of words, but counts the number of words in each observation. Bag-of-words uses tokenization to create a statistical representation of the text.*

ID	Word	Count
1	District	1
2	has	1
3	great	2
4	teachers	1
5	and	1
6	facilities	1

b) ***N-gram: builds off of bag-of-words by producing a group of words of n size.***



c) ***Temporal data:** Temporal or time series data is data involving time, like a series of dates. Temporal data can come in many formats, and is often a mix of numeric and text data.*

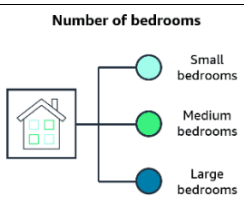
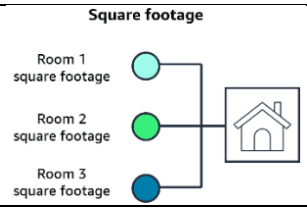
Listing date	Listing date	is_summer	day_of_month	month	year
June 3, 2014	2014-06-03	1	3	6	2014
06/05/18	2018-06-05	1	5	6	2018
31 Nov, 23	2023-11-31	0	31	11	2023
8-1-24	2024-08-01	1	1	8	2024

When to use which

Bag of Words	*create a statistical representation of the text*
N-grams	*Phrase of n-size important (like sentiment analysis)*
Temporal data	*Capture key trends, cycles, or patterns over time*

Feature Selection Techniques

- ***Feature splitting & Feature combining***

Feature splitting	***Feature combining***
breaks down features into multiple derived features	
	

- ***Principal component analysis***

Statistical technique that you can use for dimensionality reduction

Property number	Square footage	Bedrooms	Bathrooms	Price (in dollars)	Tax rate	State
1	1,500	3	2	250,000	0.82%	NC
2	2,000	4	3	350,000	0.87%	WA
3	1,000	2	1	150,000	1.61%	WI
4	3,000	5	4	450,000	2.08%	IL
5	2,500	4	3	400,000	1.63%	NE

1 ***Principal component – Size{|:*** *The first component accounts for the physical attributes of the home that includes square footage, bedrooms, and bathrooms*

2 ***Principal component - Cost:*** *The second component captures the financial factors that include price and tax rate.*

X. AWS Tools for Data Transformation

X.1. Data Labeling with AWS

1. Mechanical Turk

Purpose:

- ***Image annotation:***
- ***Text annotation:***
- ***Data collection:***
- ***Data cleanup:***
- ***Transcription:***

2. SageMaker Ground Truth

Uses Mechanical Turk and other data processing methods to streamline the data preparation process even further.

Purpose

- ***Image annotation:***
- ***Text annotation:***
- ***Object Detection***
- ***Named Entity Recognition***

3. SageMaker Ground Truth Plus

Fully managed data labeling service using expert labelers.

When to use which

Mechanical Turk *On-demand access to workers, lower costs, fast turnaround times, task flexibility, and quality control capabilities.*
SageMaker Ground Truth *Higher quality, or Object detection or NER, public sourcing*
SageMaker Ground Truth Plus *Production labeling workflows, sensitive data, complex tasks, and custom interfaces*

X.2. Data Ingestion with AWS

1. Data Wrangler

***Purpose**: visual, code-free tool for data preprocessing and feature engineering*

Steps

- ***Clean data:***
- ***Feature engineering:** Combine columns and apply formulas, etc.*
- ***Fixing formatting issues:** Missing headers, encoding problems, etc.*
- ***Reducing data size:** For large datasets*
- ***Automating transformations:***

2. SageMaker Feature Store

***What**:*

- *Managed repository for storing, sharing, and managing features.*
- ***Storing features** saves time by eliminating redundant feature engineering efforts.*

Steps

- *Automated data preprocessing*
- *Centralized feature repository:*
- *Management of feature pipelines:*
- *Standardized features:*
- *Caching for performance:*

When to use which

- ***Use Data Wrangler:***
  - *Initial data exploration*
  - *one-time transformations*
  - *when working directly in notebooks.*
- ***Use Feature Store***
  - *Moving to production*
  - *Sharing features across models or teams*
  - *when you need low-latency feature serving for online inference.*

X.3. Data Transformation with AWS

1. Data Glue

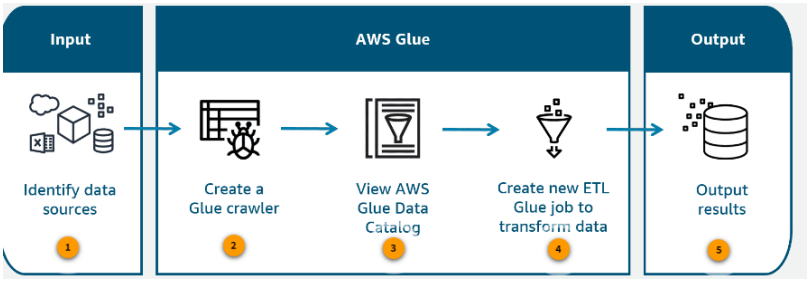
Purpose:

- *AWS Glue auto-generates Python code to handle issues like distributed processing, scheduling, and integration with data sources.*
- *AWS Glue DataBrew is visual data preparation tool for cleaning, shaping, and normalizing datasets.*

Use cases

- *Automated ETL pipelines*
- *Data integration and ingestion:*
- *Data cleansing and standardization*
- *Feature engineering:*
- *Final pretraining data preparation*

Steps



2. SageMaker Data Wrangler

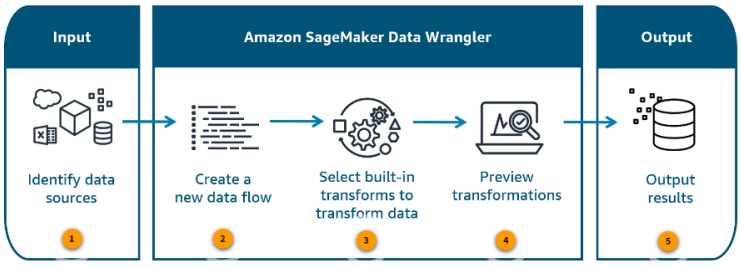
Purpose:

- *We know SageMaker Data Wrangler to ingest data.*
- *SageMaker Data Wrangler can also help explore, clean, and preprocess data without writing code.*

Use Cases

- *Clean Data & Fix formatting issues*
- *Feature Engineering*
- *Reducing data size*
- *Automating transformations*

Steps



When to use:

- ***Glue**: Production ETL pipelines, large-scale data processing, scheduled jobs*
- ***Data Wrangler:** Exploratory data analysis, quick transformations, ML data prep in SageMaker*

3. *For Streaming data*

Lambda	*Spark on Amazon EMR*
• *Data normalization:* • *Data filtering:* • *Transcoding media:*	• *Real-time analytics:* • *Anomaly detection:* • *Monitoring dashboards*

1.3 Validate Data and Prepare for Modeling

1.3.1 VALIDATE DATA

Basics

Bias Metrics

	Class imbalance (CI)	**Difference in proportion of labels (DPL)**
	Occurs when distribution of classes in the training data is skewed (one class significantly less represented than the others).	*Compare the distribution of labels in data*
Understand	*If CI +ve, advantaged **group** is relatively overrepresented in this dataset.* *If CI -ve, advantaged **group** is relatively underrepresented in this dataset.*	*If DPL +ve, one **class** significantly higher proportion.* *If DPL -ve, one **class** significantly less proportion.*

Data Validation Strategies

	Resampling	*Synthetic data generation*	*Data augmentation*
Add/Update	Adding data	Adding Data	Transform data
Auto/Manual	Manually	Algorithmically	Algorithmically
How	**Oversample/under sample**	**Creating new artificial data points**	
Technique(s)		SMOTE	GAN
Type of Data	Numeric	Text based	Image

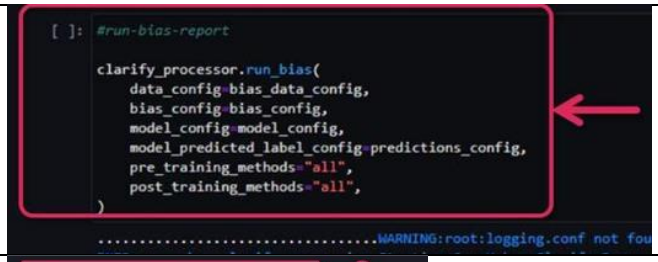
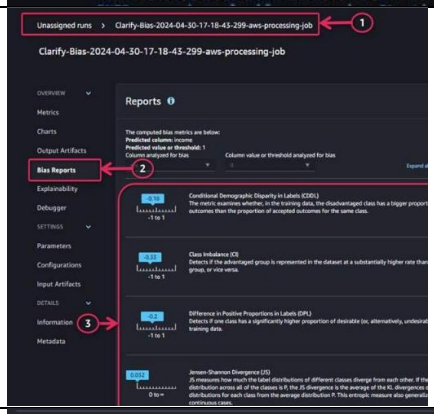
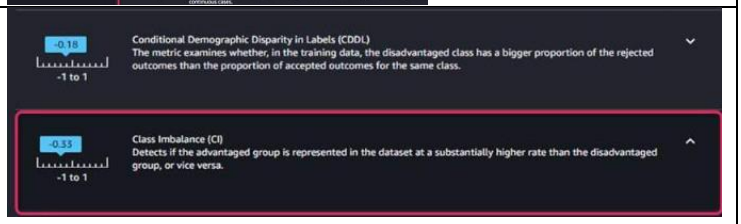
AWS tools for Data Validation

	Glue Data Quality	*Glue DataBrew*	*Comprehend*
What	Managed monitoring	Visual data prep tool	NLP tool
Use cases	• Data Validation • Data quality • Automated scheduling • Data quality dashboards	• Data profiling • Built-in transformations • Custom transformations	• Entity recognition • Language detection • Topic modeling

SageMaker Clarify

How it works

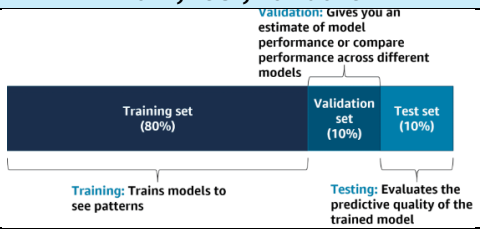
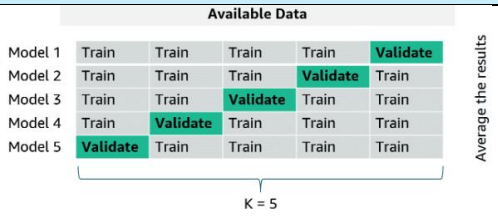
- *Create a bias report using the configuration for pre-training and post-training analysis*
- *Assess the bias report by considering the class imbalance (CI) and difference in proportion of labels (DPL).*

1. ***Set up the bias report:*** *To set up your bias report configurations, use the BiasConfig to provide information on which columns contain the facets with the sensitive group of sex, what the sensitive features might be using facets_values_or_threshold, and what the desirable outcomes are using labels_values_or_threshold.*	
2. ***Run the bias report****: create the bias report using the configuration for pretraining and post-training analysis. This step takes approximately 15-20 minutes*	
3. ***Access the bias report:***	
4. ***Assess the Class Imbalance: The*** *CI shows -0.33 as the bias metric. CI measures if the advantaged group, men, is represented in the dataset at a substantially higher rate than the disadvantaged group, women, or the other way around. The negative value demonstrated indicates that the advantaged group, men, is relatively underrepresented in this dataset example.*	
5. ***Assess the Difference in Positive Proportion of Labels:*** *Review the DPL. This metric detects a label imbalance between classes that might cause unwarranted biases during training.*	

1.3.2 PREPARE FOR MODELLING

Dataset Splitting, Shuffling, and Augmentation

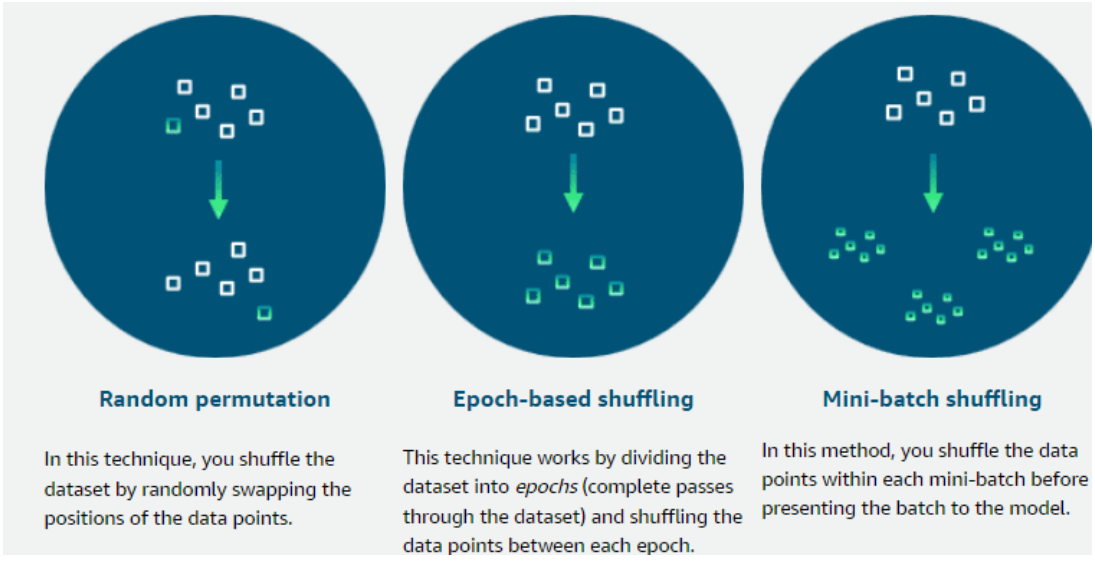
Data Splitting Techniques

	Train, test, validate	Cross-validation
		
Best for	• Easy implementation • Provides quick estimate of model performance	• Uses **the entire dataset** for training and testing, maximizing data usage • Reduces variance in performance estimation **by averaging results across multiple iterations**
Limitations	• Performance estimate might have variance due to dependency on specific examples in the test set • Not suitable for small datasets because it might lead to overfitting or underfitting	• Computationally more expensive, especially for large datasets • Might be sensitive to class imbalances if not stratified properly
Example	•	**K-fold cross-validation**

Dataset shuffling

Benefits of data shuffling: *Dataset shuffling plays a crucial role in mitigating biases that might arise from the inherent structure of the data. By introducing randomness through shuffling, you can help the model be exposed to a diverse range of examples during training.*

Data shuffling techniques:



Dataset Augmentation

Data augmentation works by creating new, realistic training examples that expand the model's understanding of the data distribution. Dataset augmentation involves artificially expanding the size and diversity of a dataset

Data augmentation techniques:

- *Image-based Augmentation*
  - *Flipping, rotating, scaling, or shearing images*
  - *Adding noise or applying color jittering*
  - *Mixing or blending images to create new, synthetic examples*
- *Text-based Augmentation*
  - *Replacing words with synonyms or antonyms*
  - *Randomly inserting, deleting, or swapping words*
  - *Paraphrasing or translating text to different languages*
  - *Using pre-trained language models to generate new, contextually relevant text*
- *Time series Augmentation*
  - *Warping or scaling the time axis*
  - *Introducing noise or jitter to the signal*
  - *Mixing or concatenating different time-series segments*
  - *Using generative models to synthesize new time-series data*

When to use which

- ✓ ***Data-splitting (Train, Test, Validate):***
  - ***Pros:*** *Clear separation of data, prevents data leakage*
  - ***Cons:*** *Reduces amount of data available for training*
- ✓ ***Cross-validation:***
  - ***Pros:*** *Makes efficient use of all data, robust performance estimate*
  - ***Cons:*** *Computationally expensive, may not be suitable for very large datasets*
- ✓ ***Data shuffling:***
  - ***Pros:*** *Reduces bias, improves generalization*
  - ***Cons:*** *May not be appropriate for time-series data where order matters*
- ✓ ***Data augmentation:***
  - ***Pros:*** *Increases dataset size, improves model robustness*
  - ***Cons:*** *May introduce artificial patterns if not done carefully*

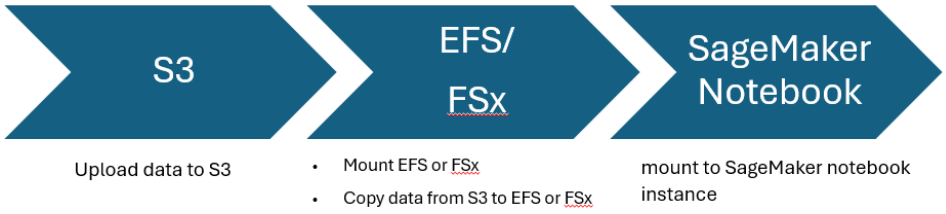
AWS services for pre-training data configuration

Final formatting process

SageMaker built-in algorithms for formatting

- **CSV**: Many built-in algorithms in SageMaker:
  - XGBoost
  - linear learner
  - DeepAR.
- **RecordIO-protobuf:** Commonly used for image data, where each record represents an image and its associated metadata.

Steps after formatting:



- **Upload data to Amazon S3**
- **Mount Amazon EFS or Amazon FSx**
- **Copy data from Amazon S3 to EFS** or **FSx**
  - Use AWS data transfer utilities or custom script
  - Verify data integrity by checking file sizes and checksums after the transfer is complete.
- **Load the data into your ML training resource**
  - With Amazon EFS -> you would create an EFS file system and mount it to your SageMaker notebook instance or training job. Copy dataset files into the EFS file system. Then in your training script, load the data by accessing the Amazon EFS mount path.
  - For Amazon FSx -> create a Lustre file system and attach it to your SageMaker resource. Copy the data files to the FSx Lustre file system. In your training script, load the data by accessing the Amazon FSx mount path.
  - Note that both Amazon EFS and Amazon FSx for Lustre provide shared file storage that can be accessed from multiple Amazon Elastic Compute Cloud (EC2) instances at the same time.
- **Monitor, refine, scale, automate, and secure**
  - When your data is loaded into your resource, you will continue to monitor, refine, scale, automate, and secure your ML workloads. Monitoring, refining, scaling, automating, and securing your workloads is a complex and involved part of the ML lifecycle.
  - Implement data lifecycle management strategies, such as archiving or deleting old or unused data.
  - Consider using AWS services like AWS Step Functions, AWS Lambda, Amazon Managed Workflows for Apache Airflow (Amazon MWAA), and AWS CodePipeline to automate and orchestrate your data workflows.

Domain 2: Data Transformation

2.1 Choose a modelling approach

2.1.1 AWS Model Approaches

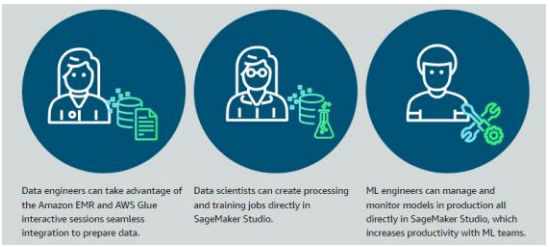
AWS AI/ML stack:

- AWS AI services: NO fine-tune option
- AWS ML services: fine-tune option
- Customized ML model solutions (using AWS infrastructure and frameworks)

2.1.1 SageMaker Offerings

Studio

2. ***Roles and Persona***



Choice of IDEs

Jupyterlab	Code Editor	RStudio

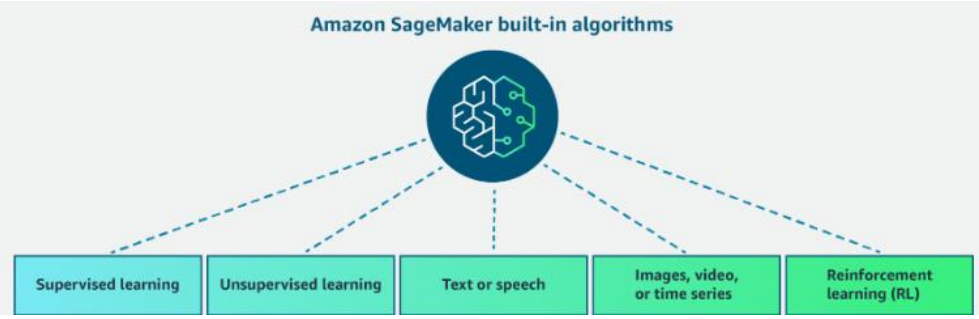
Choice of IDEs SageMaker notebook instances

SageMaker notebook instances initiate Jupyter servers on Amazon Elastic Compute Cloud (Amazon EC2) and provide preconfigured kernels with the following packages:

- Amazon SageMaker Python SDK, AWS SDK for Python (Boto3)
- AWS Command Line Interface (AWS CLI)
- Conda
- Pandas
- Deep learning framework libraries
- Other libraries for data science and ML

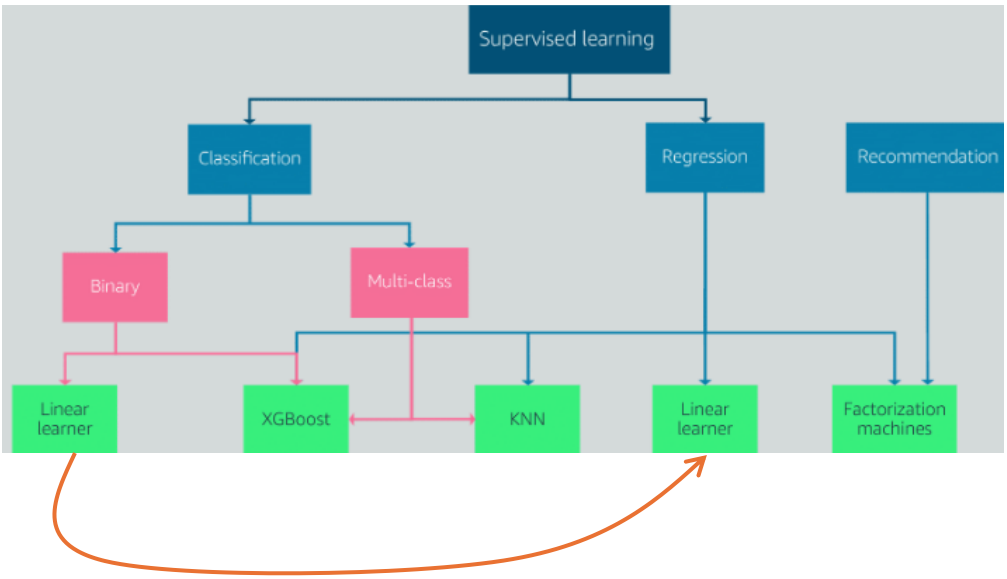
2.1.1 SageMaker Model types

SageMaker notebook instances initiate Jupyter servers on Amazon Elastic Compute Cloud (Amazon EC2) and provide preconfigured

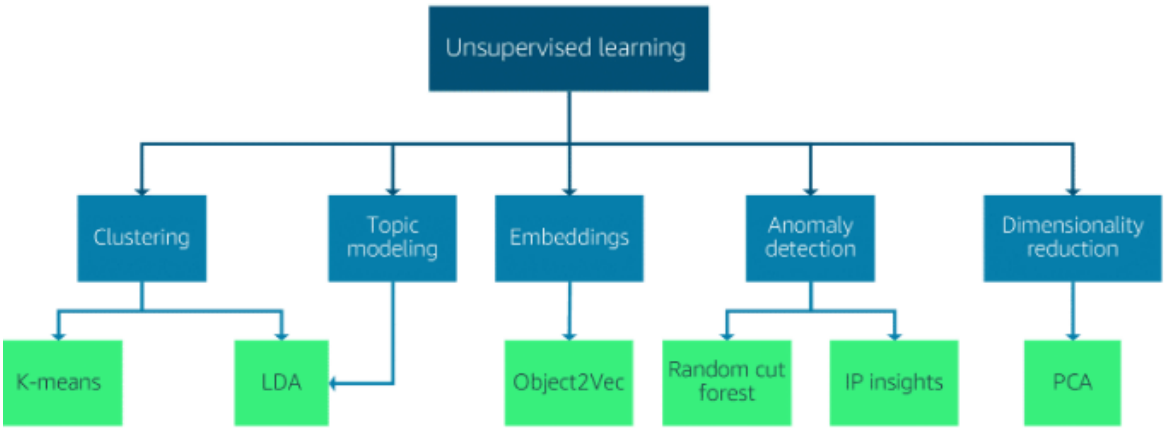


1. Supervised

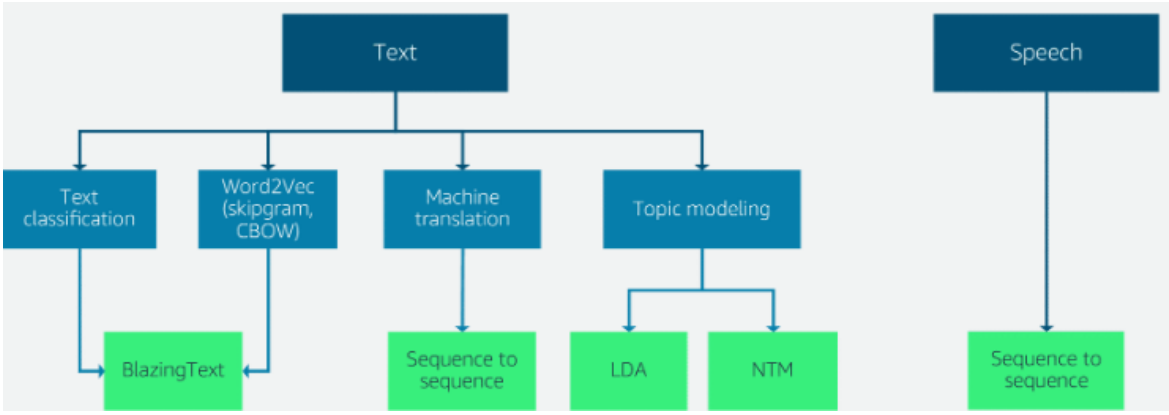
Algorithm	*Classification: Binary*	*Classification: Multi-class*	*Regression*
Linear Learner	*Yes*	*No*	*Yes*
XGBoost	*Yes*	*Yes*	*Yes*
K-Nearest Neighbors	*No*	*Yes*	*Yes*
Factorization Machines	*No*	*No*	*Yes*



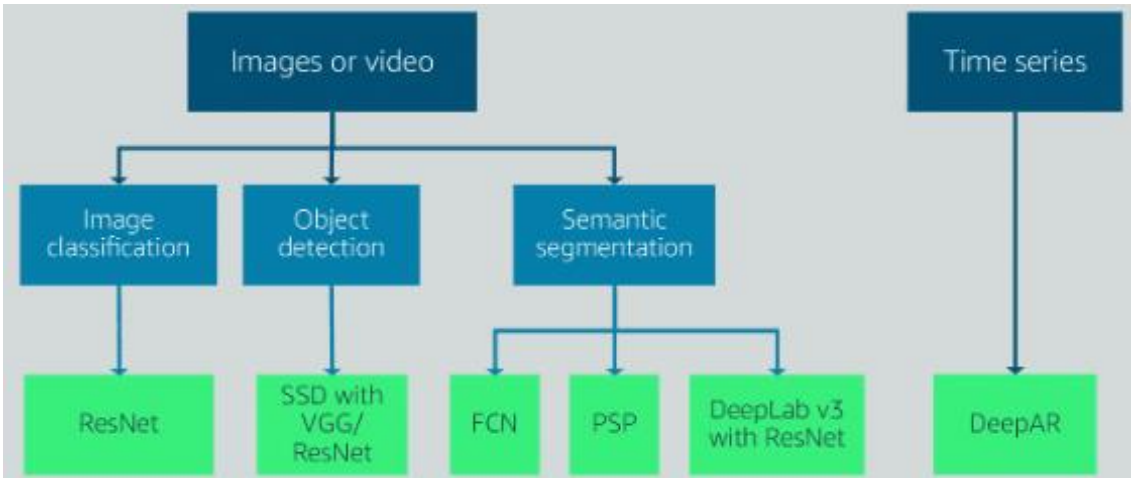
2. *Unsupervised*



3. *Text or speech data*



4. *Images and video (or time series data)*



5. *Reinforcement learning (RL)*

To train RL models in SageMaker RL, use the following components:

- **A deep learning (DL) framework.** Currently, SageMaker supports RL in TensorFlow and Apache MXNet.
- **An RL toolkit.** An RL toolkit manages the interaction between the agent and the environment and provides a wide selection of state of the art RL algorithms. SageMaker supports the Intel Coach and Ray RLlib toolkits. For information about Intel Coach, see https://nervanasystems.github.io/coach/(opens in a new tab). For information about Ray RLlib, see https://ray.readthedocs.io/en/latest/rllib.html(opens in a new tab).
- **An RL environment.** You can use custom environments, open-source environments, or commercial environments. For information, see RL Environments in Amazon SageMaker(opens in a new tab).

2.1.3 SageMake AutoML

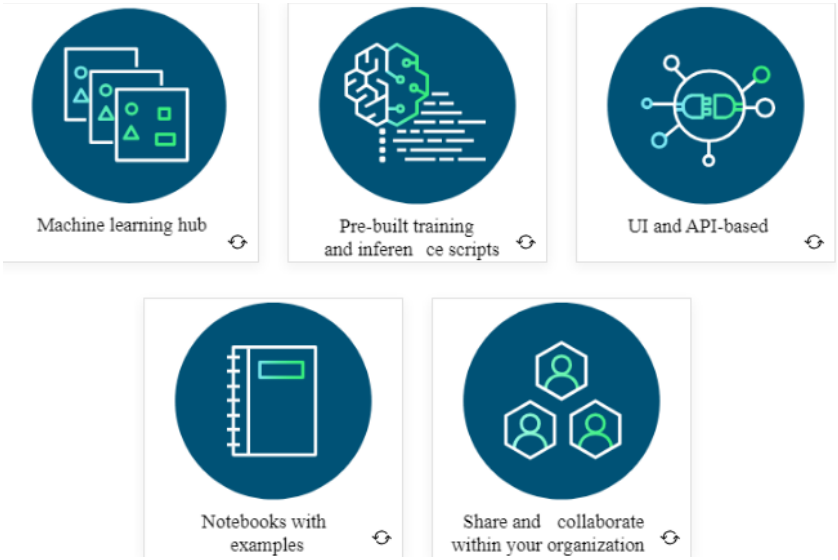
SageMaker

- **Data analysis and processing**: SageMaker Autopilot identifies your specific problem type, handles missing values, normalizes your data, selects features, and prepares the data for model training.
- **Model selection**: SageMaker Autopilot explores a variety of algorithms. SageMaker Autopilot uses a cross-validation resampling technique to generate metrics that evaluate the predictive quality of the algorithms based on predefined objective metrics.
- **Hyperparameter optimization**: SageMaker Autopilot automates the search for optimal hyperparameter configurations.
- **Model training and evaluation**: SageMaker Autopilot automates the process of training and evaluating various model candidates.
    - It splits the data into training and validation sets, and then it trains the selected model candidates using the training data.
    - Then it evaluates their performance on the unseen data of the validation set.
    - Lastly, it ranks the optimized model candidates based on their performance and identifies the best performing model.
- **Model deployment:** After SageMaker Autopilot has identified the best performing model, it provides the option to deploy the model. It accomplishes this by automatically generating the model artifacts and the endpoint that expose an API. External applications can send data to the endpoint and receive the corresponding predictions or inferences.

2.1.3 SageMake JumpStart

SageMaker JS is a ML hub with foundation models, built-in algorithms, and prebuilt ML solutions that you can deploy with a few clicks.

Features



Foundation Models

With JumpStart foundation models, many models are available such as:

- *Jurassic models from AI21*
- *Stable Diffusion from Stability.ai*
- *Falcon from HuggingFace*
- *Llama from Meta*
- *AlexaTM from Amazon*

JumpStart industry-specific solutions

- ***Demand forecasting***

Amazon SageMaker JumpStart provides developers and data science teams ready-to-start AI/ML models and pipelines. SageMaker JumpStart is ready to be deployed and can be used as-is. For demand forecasting, SageMaker JumpStart comes with a pre-trained, deep learning-based forecasting model, using Long- and Short-Term Temporal Patterns with Deep Neural Networks (LSTNet).

- ***Credit rating prediction***

Amazon SageMaker JumpStart solution uses Graph-Based Credit Scoring to construct a corporate network from SEC filings (long-form text data).

- ***Fraud detection***

Detect fraud in financial transactions by training a graph convolutional network with the deep graph library and a SageMaker XGBoost model.

- ***Computer vision***

Amazon SageMaker JumpStart supports over 20 state-of-the-art, fine-tunable object detection models from PyTorch hub and MxNet GluonCV. The models include YOLO-v3, FasterRCNN, and SSD, pre-trained on MS-COCO and PASCAL VOC datasets.

Amazon SageMaker JumpStart also supports image feature vector extraction for over 52 state-of-the-art image classification models including ResNet, MobileNet, EfficientNet from TensorFlow hub. Use these new models to generate image feature vectors for their images. The generated feature vectors are representations of the images in a high-dimensional Euclidean space. They can be used to compare images and identify similarities for image search applications.

- ***Extract and analyze data from documents***

JumpStart provides solutions for you to uncover valuable insights and connections in business-critical documents. Use cases include text classification, document summarization, handwriting recognition, relationship extraction, question and answering, and filling in missing values in tabular records.

- ***Predictive maintenance***

The AWS predictive maintenance solution for automotive fleets applies deep learning techniques to common areas that drive vehicle failures, unplanned downtime, and repair costs.

- ***Churn prediction***

After training this model using customer profile information, you can take that same profile information for any arbitrary customer and pass it to the model. You can then have it predict whether that customer is going to churn or not. Amazon SageMaker JumpStart uses a few algorithms to help with this. LightGBM, CatBoost, TabTransformer, and AutoGluon-Tabular used on a churn prediction dataset are a few examples.

- ***Personalized recommendations***

Amazon SageMaker JumpStart can perform cross-device entity linking for online advertising by training a graph convolutional network with a deep graph library.

- ***Healthcare and life sciences***

You could use the model to summarize long documents with LangChain and Python. The Falcon LLM is a large language model, trained by researchers at the Technology Innovation Institute (TII) on over 1 trillion tokens using AWS. Falcon has many different variations, with its two main constituents Falcon 40B and Falcon 7B, comprised of 40 billion and 7 billion parameters, respectively. Falcon has fine-tuned versions trained for specific tasks, such as following instructions. Falcon performs well on a variety of tasks, including text summarization, sentiment analysis, question answering, and conversing.

- ***Financial pricing***

Many businesses dynamically adjust pricing on a regular basis to maximize their returns. Amazon SageMaker JumpStart has solutions for price optimization, dynamic pricing, option pricing, or portfolio optimization use cases. Estimate price elasticity using Double Machine Learning (ML) for causal inference and the Prophet forecasting procedure. Use these estimates to optimize daily prices.

- ***Causal inference***

Researchers can use machine learning models such as Bayesian networks to represent causal dependencies and draw causal conclusions based on data.

2.1.5 Bedrock

Use cases

Text generation

Create new pieces of original content, such as short stories, essays, social media posts, and webpage copy.

Chatbots

Build conversational interfaces, such as chatbots and virtual assistants, to enhance the user experience for your customers. Build assistants that understand user requests, automatically break down tasks, engage in dialogue to collect information, and take actions to fulfill the request.

Search

Search for and synthesize relevant information to answer questions and provide recommendations from a large corpus of text and image data.

Text summarization

Get concise summaries of long documents such as articles, reports, research papers, technical documentation, and even books to quickly and effectively extract important information.

Image generation

Quickly create realistic and visually appealing images for ad campaigns, websites, presentations, and more from language prompts.

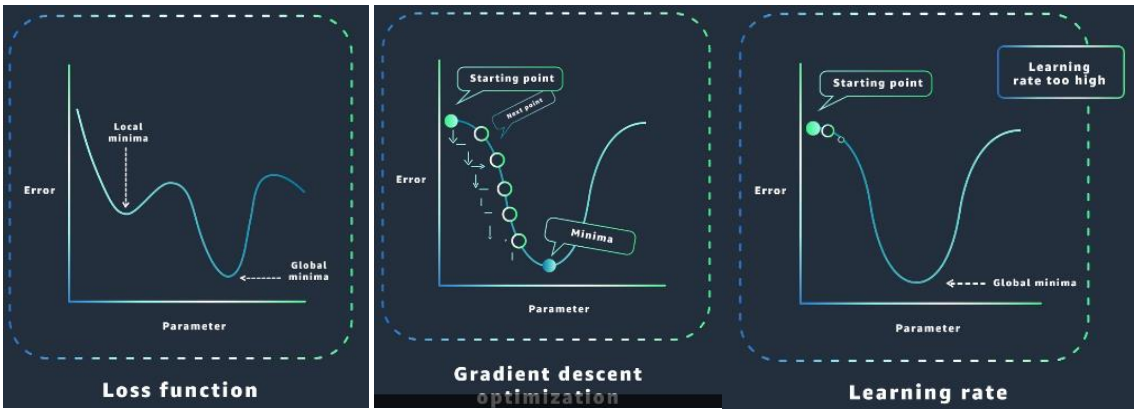
Personalization

Help customers find what they're looking for with more relevant and contextual product recommendations than word matching.

2.2 Train Models

2.2.1 Model Training Concepts

Minimizing loss:



- Loss function needs to be minimum (Global minimum)
- Gradient descent optimization used to reach "global minima" loss
- Hyperparameter tuning uses Gradient descent to reach "global minima" loss
    - Too much tuning can result in "Overshoot", learning rate too high
    - Too less tuning can result in "Undershooting", learning rate too small

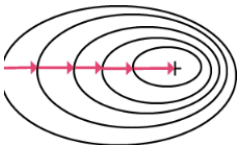
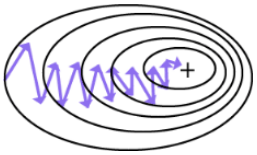
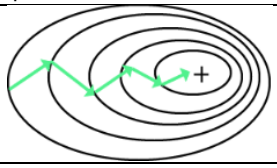
(Measuring) Loss function:

Root mean square error	Log-likelihood loss
• The most basic form of a loss function, commonly used in regression tasks such as predicting continuous values is known as Root Mean Square Error (RMSE). • A regression task can be used to predict home prices.	• A variation of a loss function is log-likelihood loss, also known as cross-entropy loss, is used in logistic regression. • With log-likelihood loss, instead of the raw probabilities of predictions of each class, the logarithm of probabilities is considered.
$\sqrt{\frac{\sum_{i=1}^{n}\left(Y_{target,i} - Y_{pred,i}\right)^2}{n}}$	$-(y \log p + (1-y)\log(1-p))$

When to use which

Log-likelihood loss is an algorithm used for classification tasks, where the goal is to predict whether an input belongs to one of two or more classes. For example, you might use logistic regression to predict whether an email is spam.

Optimizing - Reducing Loss function:

Optimization technique	***Gradient descent***	***Stochastic gradient descent***	***Mini-batch gradient descent***
Weights updated	Every epoch	Every datapoint	Every batch
Speed of each epoch calculation	Slowest	Fast	Slower
Gradient steps	Smooth updates toward the minima	Noisy or erratic updates toward the minima	Less noisy or erratic updates toward the minima
			

Gradient descent

As mentioned, gradient descent only updates weights after it's gone through all of the data, also known as an epoch. Of the three variations covered here

- gradient descent has the slowest speed to finding the minima as a result, but
- also has the fewest number of steps to reach the minima.

In stochastic gradient descent or SGD, you update your weights for each record you have in your dataset.

Stochastic Gradient Descent (SGD)

For example, if you have 1000 data points in your dataset, SGD will update the parameters 1000 times. With gradient descent, the parameters would be updated only once in every epoch.

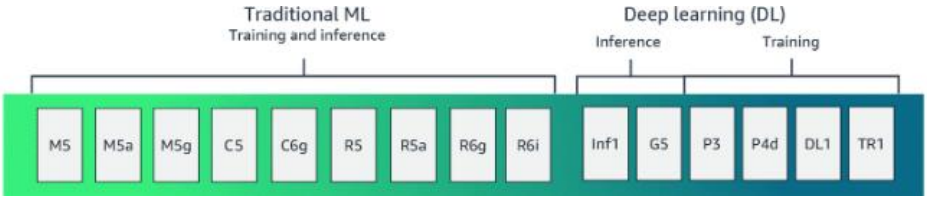
- SGD leads to more parameter updates and, therefore, the model will **get closer to the minima more quickly.**
- One drawback of SGD, however, is that it will oscillate in different directions, unlike gradient descent, hence lot more steps.

Mini-batch gradient descent

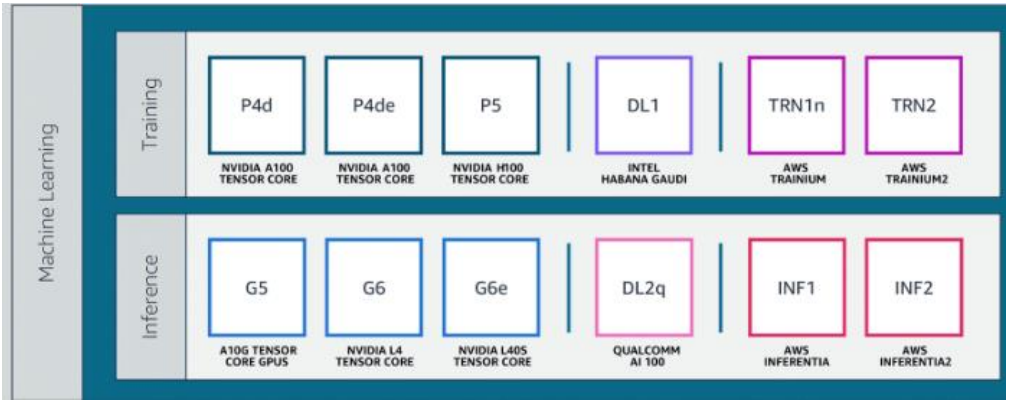
Hybrid of gradient descent and SGD, this approach uses a smaller dataset or a batch of records, also called batch size, to update your parameters.

- Mini-batch gradient descent updates more than gradient descent while having less erratic or noisy updates as compared to SGD. The user-defined batch size helps you fit the smaller dataset into memory. Having a smaller dataset helps the algorithms run on almost any average computer that you might be using.

2.2.2 Compute Environment



AWS Instances for ML:



AWS offers solutions for a variety of specific ML tasks, and this permits you to optimize on your particular use case scenarios.

AWS Container Services:

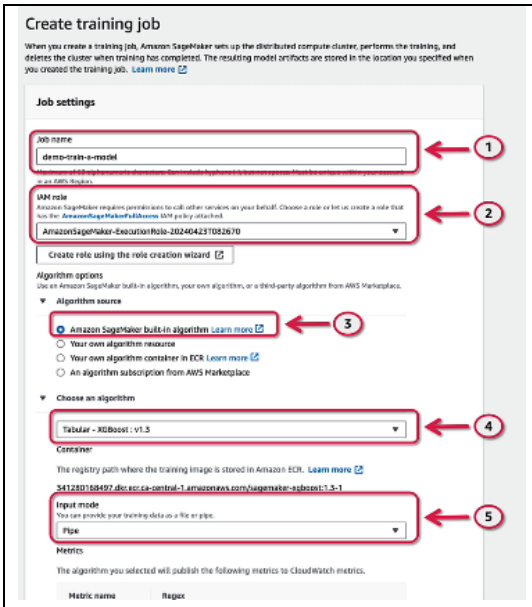
		Keyword
Amazon ECS	*ECS simplifies the process of running and managing containerized applications on AWS, offering various deployment options, and seamlessly integrating with other AWS services.*	*General/custom Container*
Amazon EKS	*Amazon EKS provides a fully managed Kubernetes control plane and seamless integration with other AWS services*	*Kubernetes*
AWS Fargate		*Fully Managed Container*
Amazon ECR	*ECR makes it easy to store, manage, and deploy container images.*	*Container Registry*

Containers in SageMaker for ML model generation

1. ***SageMaker managed container images***
   - You can use built-in training algorithms included in these containers, ML Framework, settings, libraries, and dependencies included in these container images, but provide your own custom training algorithms. This approach is referred to as script mode.
2. ***Customer-managed container images (BYOC)***
   - You can build your own container using the Bring Your Own Container (BYOC) approach if you need more control over *the algorithm, framework, dependencies, or settings.*
   - *Some industries might require BYOC or BYOM to meet regulatory and auditing requirements.*

2.2.3 Train a model

Create training job

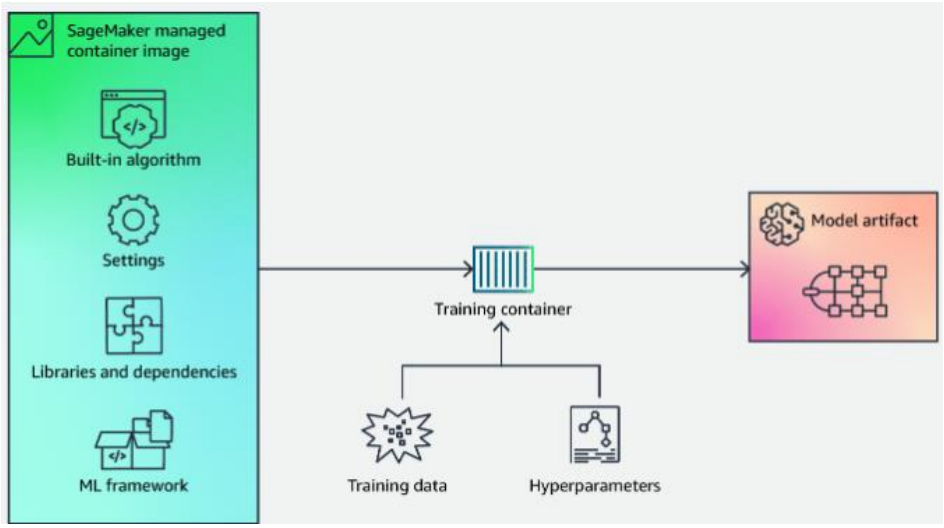


- Create IAM role
- Chose algorithm source (built-in, etc.)
- Choose algorithm
- Configure compute resource
- Set hyperparameters (+ default)
- Specify data type
- Choose Data source (default S3)

Model created

- Store in S3
- Package and distribute
- Register model (in registry)

Train a model



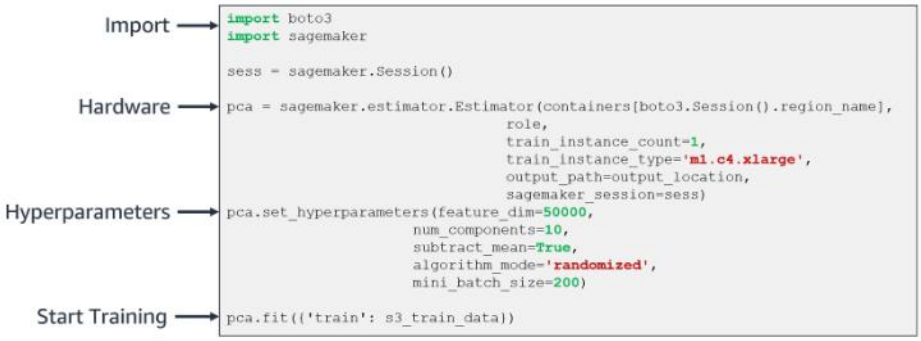
For built-in algorithms, the only inputs you need to provide are the

- *training data*
- *hyperparameters*
- *compute resources.*

Amazon SageMaker training options

When it comes to training environments, you have several to choose from:

- *Create a training job using SageMaker console (see the Creating a Training Job Using the Amazon SageMaker Console lesson for an example using this method).*
- *Use AWS SDKs for the following:*
  - *The high-level SageMaker Python SDK*
  - *The low-level SageMaker APIs for the SDK for Python (Boto3) or the AWS CLI*



Training data sources

- S3
- Amazon EFS
- Amazon FSx for Lustre

Training ingestion modes

	Pipe mode	File mode	Fast File mode
What	SageMaker streams data directly from Amazon S3 to the container, without downloading the data to the ML storage volume	SageMaker will download the training data from S3 to the provisioned ML storage volume. Then it will mount the directory to the Docker volume for the training container.	SageMaker can stream data directly from S3 to the container with no code changes. Users can author their training script to interact with these files as though they were stored on disk.
Pros	Improve training performance by reducing the time spent on data download	In a distributed training setup ,the training data is distributed uniformly across the cluster.	Fast File mode works best when the data is read sequentially.
Cons		Manually ensure ML storage volume has sufficient capacity to accommodate data from Amazon S3.	Augmented manifest files are not supported. The startup time is lower when there are fewer files in the S3 bucket provided.

When to use which

	Pipe mode	File mode	Fast File mode
large training datasets	X		X
training algorithm reads data sequentially		X	X

Amazon SageMaker training – Script mode

Amazon SageMaker script mode provides the flexibility to develop custom training and inference code while using industry-leading machine learning frameworks.

Steps to bring your own script using SageMaker script mode

1. Use your local laptop or desktop with the SageMaker Python SDK. You can get different instance types, such as CPUs and GPUs, but are not required to use the managed notebook instances.
2. Write your training script.
3. Create a SageMaker estimator object, specifying the
    a) training script
    b) instance type
    c) other configurations.
4. Call the *fit* method on the estimator to start the training job, passing in the training and validation data channels.
5. SageMaker takes care of the rest. It pulls the image from Amazon Elastic Container Registry (Amazon ECR) and loads it on the managed infrastructure.
6. Monitor the training job and retrieve the trained model artifacts once the job is complete.

Example

```
import sagemaker
from sagemaker.pytorch import PyTorch

# Define the PyTorch estimator
estimator = PyTorch(
    entry_point="train.py",
    role=sagemaker.get_execution_role(),
    instance_count=1,
    instance_type="ml.p3.2xlarge",
    framework_version="1.12.0",
    py_version="py38",
)
# Launch the training job
estimator.fit({"train": "s3://my-bucket/train_data"})
```

In this example, the *PyTorch* estimator is configured with the training script using the entry_point: *train.py*, instance type *ml.p3.2xlarge*, and other settings. The *fit* method is called to launch the training job, passing in the location of the training data.

Reducing training time

Amazon SageMaker script mode provides the flexibility to develop custom training and inference code while using industry-leading machine learning frameworks.

a) *Early stopping:*

Early stopping is a regularization technique that shuts down the training process for a ML model when the model's performance on a validation set stops improving.

How early stopping works in Amazon SageMaker

Amazon SageMaker provides a seamless integration of early stopping into its hyperparameter tuning functionality, so users can use this technique with minimal effort. Here is how early stopping works in SageMaker:

a) ***Evaluating objective metric after each epoch**: During the training process, SageMaker evaluates the specified objective metric (for example, accuracy, loss, F1-score) for each epoch or iteration of the training job.*

b) ***Comparing to running median of previous training jobs***

c) ***Stopping current job if performing worse than median**:*

b) *Distributed training*

A. ***Data parallelism** is the process of splitting the training set in mini-batches evenly distributed across nodes. Thus, each node only trains the model on a fraction of the total dataset.*

Done in SageMaker using SageMaker distributed data parallelism (SMDDP) library

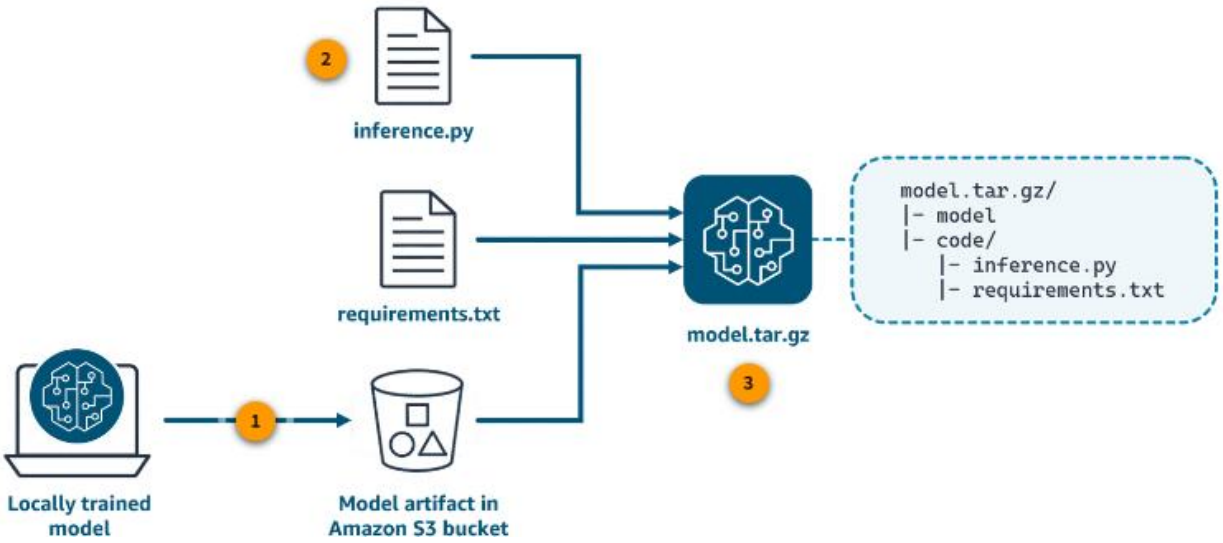
B. ***Model parallelism** is the process of splitting a model up between multiple instances or nodes.*

SageMaker model parallelism library v2 (SMP v2)

Guidance on choosing data parallelism compared to model parallelism

- *If model can fit on a single GPU's memory but your dataset is large, **data parallelism** is the recommended approach. It splits the training data across multiple GPUs or instances for faster processing and larger effective batch sizes.*
- *If model is too large to fit on a single GPU's memory, **model parallelism** becomes necessary. It splits the model itself across multiple devices, enabling the training of models that would otherwise be intractable on a single GPU.*

Building a deployable model package



Step 1: *Upload your model artifact to Amazon S3.*

Step 2: *Write a script that will run in the container to load the model artifact. In this example, the script is named inference.py. This script can include custom code for generating predictions, as well as input and output processing. It can also override the default implementations provided by the pre-built containers.*

To install additional libraries at container startup, add a requirements.txt file that specifies the libraries to be installed by using pip.

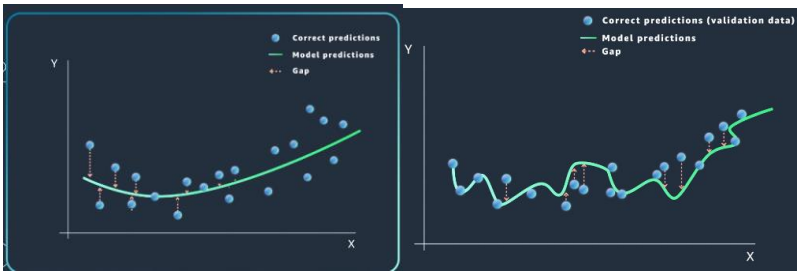
Step 3: *Create a model package that bundles the model artifact and the code. This package should adhere to a specific folder structure and be packaged as a tar archive, named model.tar.gz, with gzip compression.*

2.3 Refine Models

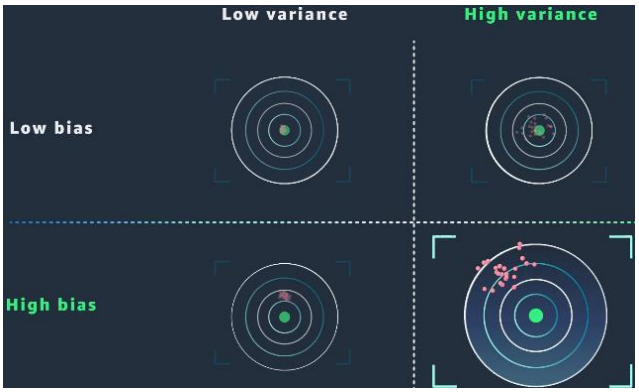
2.3.1 Evaluating Model Performance

Bias and Variance

a) **What are these**



Bias ***Variance***



Common cause of high model bias vs Variance

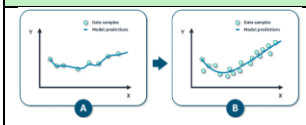
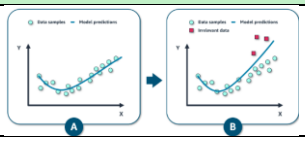
Bias	***Variance***
The model is too simple	The model is too complex
Incorrect modeling or feature engineering	Too much irrelevant data in training dataset
Inherited bias from the training dataset	Model trained for too long on training dataset

2.3.2 Model Fit (Overfitting and Underfitting)

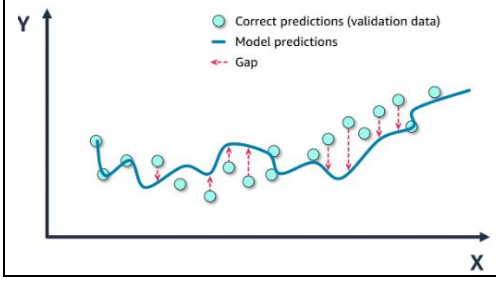
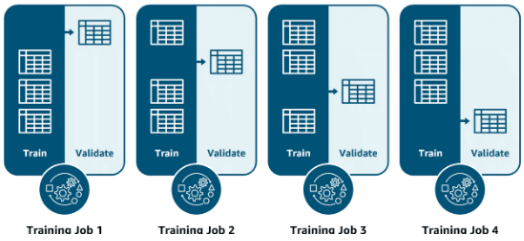
1. Overfit/Underfit

- **Overfit**

1. Reasons

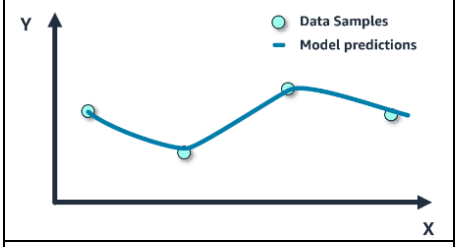
Training data too small	Too much irrelevant data	Excessive training time	Overly complex architecture
			
		Prolonged training on the same data can cause model to memorize training examples instead of learning underlying patterns.	*A model with too many parameters (weights and biases) can start memorizing the training data and noise.*

2. Detecting model overfitting

Check for model variance	Use K-fold cross-validation
If your model performed well with the training set but poorly with the validation set, it indicates high variance and overfitting.	You split the input data into k subsets of data, also called folds. You train multiple models on this dataset. For each model, you change which fold is set aside to be the evaluation dataset. This process is repeated k times
	

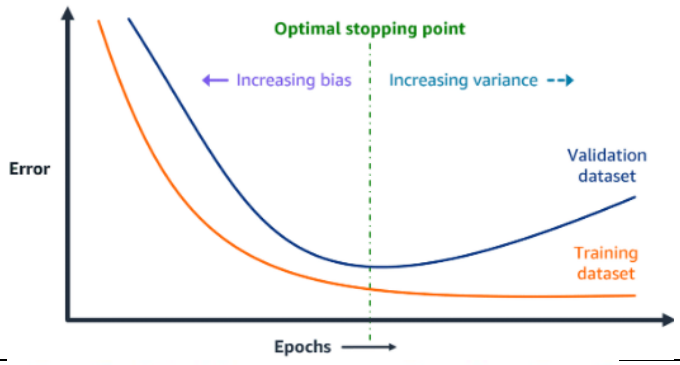
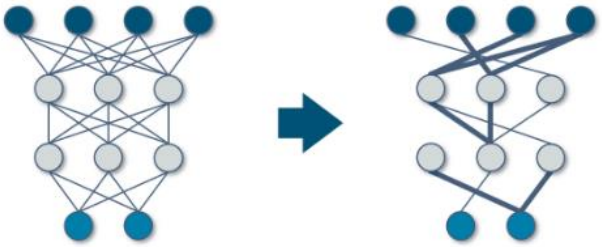
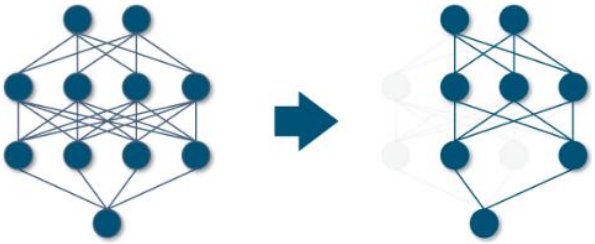
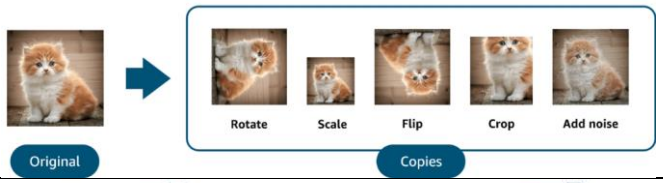
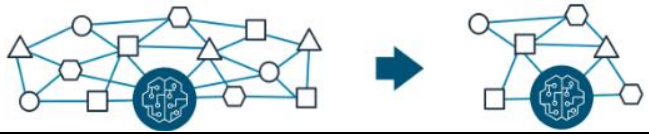
- **Underfit**

Reasons

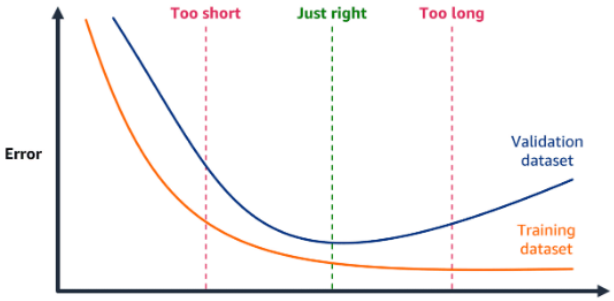
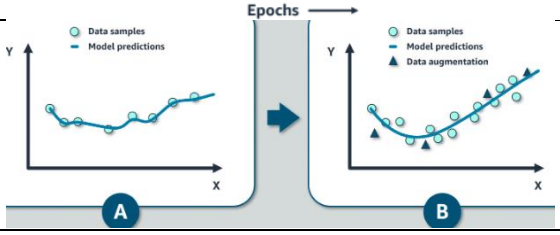
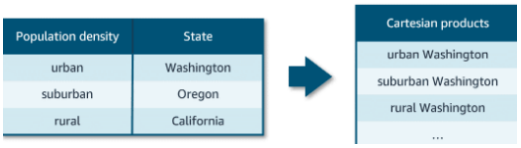
Insufficient Data	Insufficient Training Time	Excessive training time
	*Model might not have had the opportunity to learn the necessary patterns and relationships in the data*	*A model with too few parameters (weights and biases) will likely not be able to accurately capture the nonlinear relationships or intricate patterns within the data instead of learning underlying patterns.*

b) Preventing Overfitting and Underfitting

a) Remediating Overfitting

Early stopping Pauses the training process before the model learns the noise in the data.	
Pruning Aims to remove weights that don't contribute much to the training process	
Regularization	**a) Dropout** Randomly drops out (or sets to 0), a number of neurons in each layer of the neural network during each epoch.  **b) L1 regularization** Push the weights of less important features to zero. **c) L2 regularization** Results in smaller overall weight values (and stabilizes the weights) when there is high correlation between the input features.
Data augmentation	perform data augmentation to increase the diversity of the training data 
Model architecture simplification	

b) Remediating Underfitting

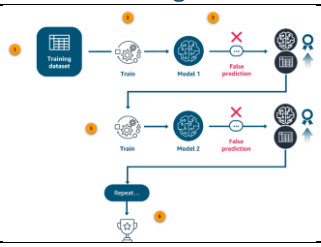
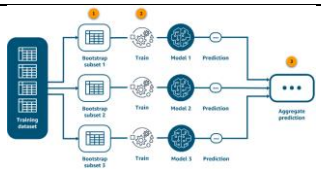
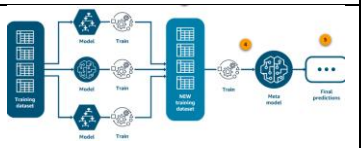
Train for an appropriate length of time	
Use a larger number of data points	
Increase model flexibility	**a) Add New Domain -specific features** For example, if you have length, width, and height as separate variables, you can create a new volume feature to be a product of these three variables. b) Add Cartesian Products Consider generating new features through Cartesian products.  **c) Change Feature Engineering** Adjusting types of feature processing techniques can increase model flexibility. E.g., in NLP tasks, you increase the size of n-grams, etc. **d) Decrease Regularization** Such as reducing the regularization strength or using a different regularization technique,

c) *Combining models for improved performance*

***Ensembling**: Process of combining the predictions or outputs of multiple machine learning models to create a more accurate and robust final prediction.*

The idea behind ensembling is that by combining the strengths of different models, the weaknesses of individual models can be mitigated. This leads to improved overall performance.

The following are commonly used ensembling methods:

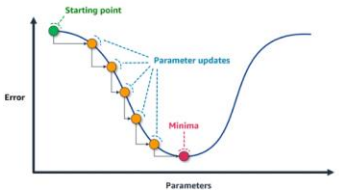
	Boosting	Bagging	Stacking
	trains different machine learning models sequentially	combines multiple models trained on different datasets	combines both
When	**Accuracy**	**Interpretability**	
Prevents	• **Overfitting** • underfitting	• **overfitting**	
			
	a) Adaptive Boosting (AdaBoost) b) Gradient Boosting (GB) c) Extreme Gradient Boosting (XGBoost)		

Boosting algorithm		
Adaptive Boosting (AdaBoost)	**Gradient Boosting (GB)**	**Extreme Gradient Boosting (XGBoost)**
classification	• classification • regression	• classification • regression • large datasets and big data applications.
Bagging (bootstrap aggregation)		
Random forests		
Stacking		
??		

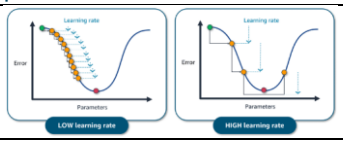
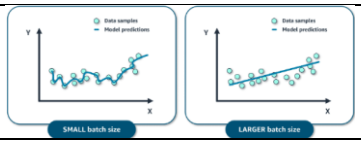
2.3.3 Hyperparameter Tuning

Benefits of Hyperparameter tuning

a) Impact of Hyperparameter tuning on model performance



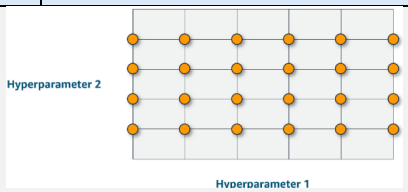
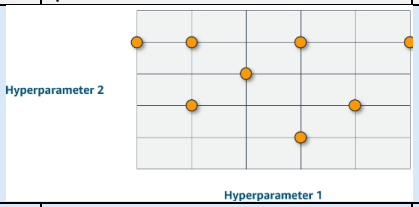
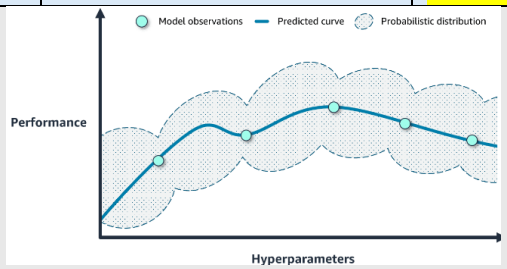
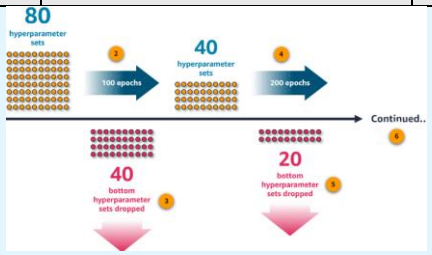
b) Types of hyperparameters for tuning

Gradient Descent algorithm			
	Learning Rate	Batch Size	Epochs
	Determines the step size taken by the algorithm during each iteration. This controls the rate at which the training job updates model parameters.	# of examples used in each iteration	# of passes through the entire training dataset
			
Careful	If the learning rate is too high, the algorithm might overshoot the optimal solution and fail to converge.	A larger batch size can lead to faster convergence but might require more computational resources.	However, too many can result in overfitting.

Neural networks			
# of layers	# of neurons in each layer	Choice of activation functions	Regularization Techniques
more layers -> more complex	more neurons -> more processing power	introduce non-linearity into the neural network Common activation functions include: • Sigmoid function • Rectified Linear Unit (ReLU) • Hyperbolic Tangent (Tanh) • Softmax function	helps prevent overfitting . Common regularization techniques • L1 /L2 regularization • Dropout • Early stopping
increasing the depth of a network risks overfitting.	Increasing number of neurons risks overfitting		

Decision Tree		
Maximum Depth of tree	# of neurons in each layer	Choice of activation functions
helps manage complexity of the model and prevent overfitting	Sets a threshold that the data must meet before splitting a node. prevents the tree from creating too many branches. This also helps to prevent overfitting	Options to select how algorithm evaluates node splits: • **Gini impurity:** measures purity of data and the likelihood that data could be misclassified. • **Entropy:** Measures randomness of data. The child node that reduces entropy the most is the split that should be used.

Hyperparameter tuning techniques

		Pros	Cons	When
Manual		When you have a good understanding of the problem at hand	time-consuming	Domain knowledge, and prior experience with similar problems
Grid search	 **Systematic and exhaustive approach to hyperparameter tuning. It involves defining all possible hyperparameter values and training and evaluating the model for every combination of these values.**			
		Reliable technique, especially for smaller-scale problems.	Computationally expensive.	Small scale and accuracy
Random search				
		More efficient than Grid Search	Optimum hyperparameter combination could be missed.	
Bayesian optimization	 **Uses the performance of previous hyperparameter selections to predict which of the subsequent values are likely to yield the best results.**			
		• can handle composite objectives • can also converge faster than random search.	• More complex to implement. • Works sequentially, so difficult to scale.	multiple objectives and/or speed.
Hyperband	 **Dynamically allocates resources to well-performing configurations and stops underperforming ones early.**			
		• can train multiple models in parallel • can be a more efficient allocation of compute resources than grid search or random search	• Not for regular alogos	• Only be used for iterative algorithms like neural networks • For limited resources

Hyperparameter tuning using SageMaker AMT

STEPS

1. Define your environment and resources, such as output buckets, training set, and validation set.
2. Specify the hyperparameters to tune and the range of values to use for each of the following: **alpha**, **eta**, **max_depth**, **min_child_weight**, and **num_round**.
3. Identify the objective metric that SageMaker AMT will use to gauge model performance.
4. Configure and launch the SageMaker AMT tuning job, including completion criteria to stop tuning after the criteria have been met.
5. Identify the best-performing model and the hyperparameters used in its creation.
6. Analyze the correlation between the hyperparameters and objective metrics.

2.3.4 Managing Model Size

Model Size Overview

a) Model Size considerations

Bigger and more complex models can achieve higher accuracy on training data. However, there are several tradeoffs involved with large model sizes that must be considered.

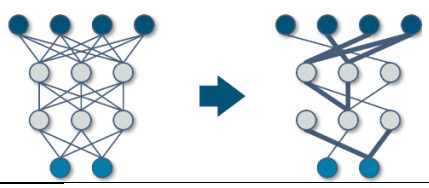
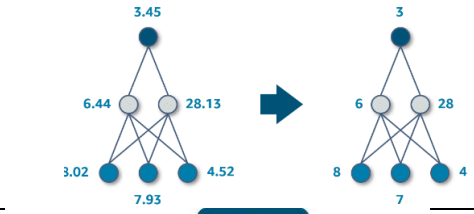
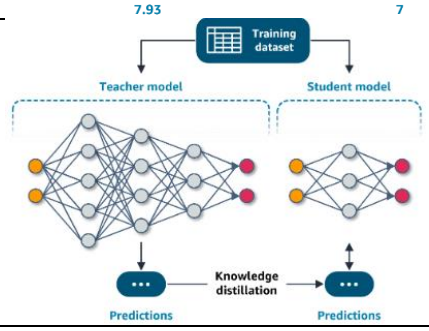
Pros

Smaller models	Larger models
Smaller models have several advantages, including faster training times, reduced memory usage, and lower computational costs. They can be particularly useful in real-time or resource-constrained scenarios where prediction speed and low latency are desired.	Larger models might perform better because they are more likely to have captured more relationships in the data.

Cons

Smaller models	Larger models
Small models might not perform as well because they are less likely to have captured complex patterns in the training data.	More relationships often come at the expense of faster deployment times, prediction latency, and greater compute resource requirements.

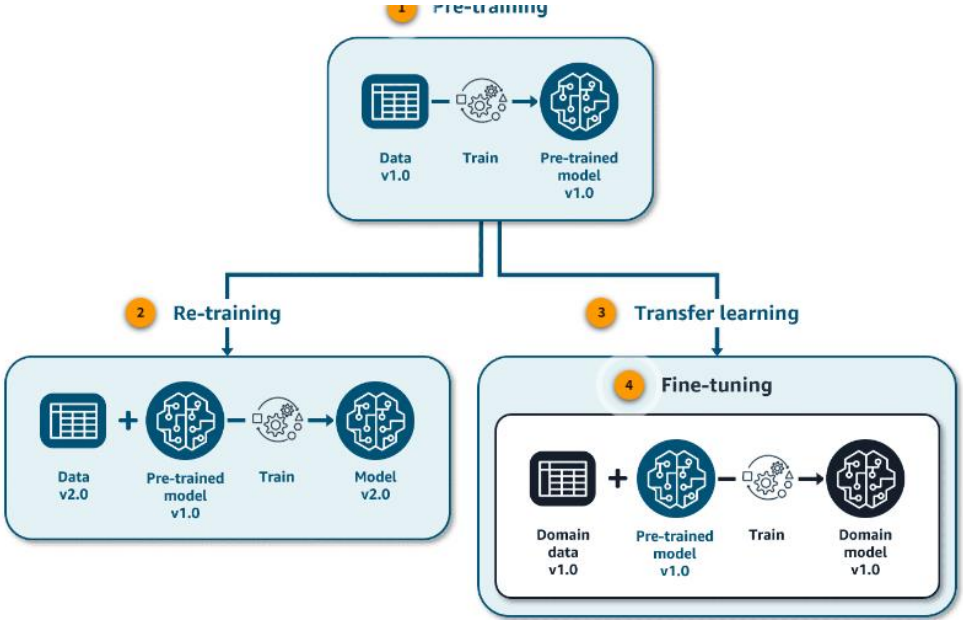
b) Model size reduction technique: Compression

Pruning Pruning is a technique that removes the least important parameters or weights from a model.	
Quantization Quantization changes the representation of weights to its most space-efficient representation. E.g., instead of a 32-bit floating-point representation of weight, quantization has the model use an 8-bit integer representation.	
Knowledge distillation With distillation, a larger teacher model transfers knowledge to a smaller student model. The student model is trained on the same dataset as the teacher. However, the student model is also trained on the teacher model's knowledge of the data.	

2.3.5 Refining Pre-trained models

Benefits of Fine tuning

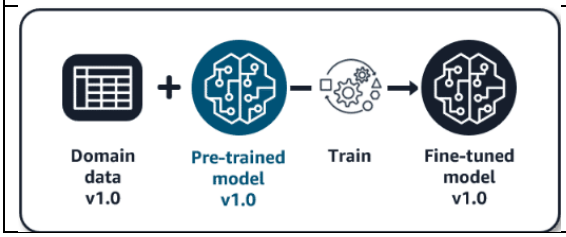
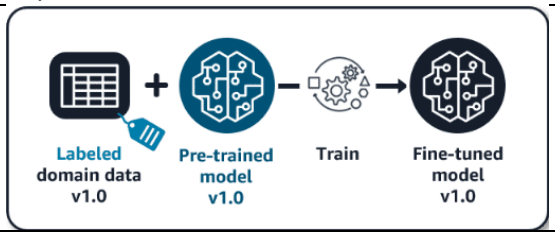
a) Where fine-tuning fits in the training process



Reasons for fine-tuning

- To customize your model to your specific business needs
- To work with domain-specific language, such as industry jargon, technical terms, or other specialized vocabulary
- To have enhanced performance for specific tasks
- To have accurate, relative, and context-aware responses in applications
- To have responses that are more factual, less toxic, and better aligned to specific requirements

b) Fine-tuning approaches

Domain adaption	***Instruction adaption***
Adapting foundation models to specific tasks by using limited domain-specific data	*Uses labeled examples to improve the performance of a pre-trained foundation model on a specific task.*
	

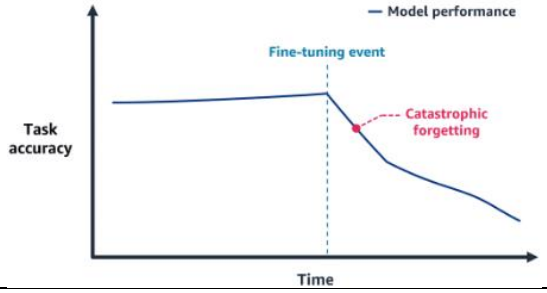
Fine tuning Models with Custom Datasets on AWS

With a Custom Dataset Using Amazon SageMaker JumpStart	*With a Custom Dataset Using Amazon Bedrock*
1. Navigate to the model detail card of your choice in SageMaker JumpStart. 2. Edit your model artifact location. 3. Enter your custom dataset location. 4. Adjust the hyperparameters of the training job. 5. Specify the training instance type. 6. Start the fine-tuning job.	1. Choose a custom model in Amazon Bedrock. 2. Create a fine-tuning job. 3. Configure the model details. 4. Configure the job. 5. Select your custom dataset. 6. Adjust the hyperparameters.

Catatrosphic Forgetting Prevention

Catastrophic forgetting occurs when a model is trained on a new task or data, and it forgets previously learned knowledge.

a) Detecting

Plot your model's performance over time. If the model's performance on specific tasks decreases significantly after training on new data, it might be a sign of catastrophic forgetting. 	Make sure your validation sets are representative of historic patterns in the data that are still relevant to the problem.

b) Preventing

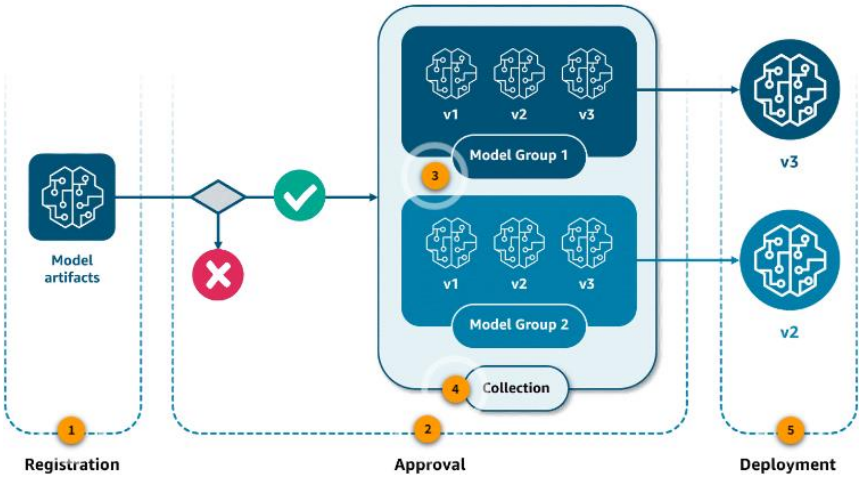
To prevent catastrophic forgetting, consider the following techniques:

1. **Elastic weight consolidation** (EWC): regularization technique that predicts which weights are important to performing previously learned tasks. It adds a penalty term to the loss function that protects these weights when the model is fine-tuned or re-trained on new task-specific data. Monitoring the EWC can indicate how much the model is forgetting older knowledge.
2. **Rehearsal**: This approach includes samples from the original training set during the fine-tuning or re-training process. During this process, the model rehearses the previous task to help it retain the learned knowledge.
3. **Model design**: You can also design your model with the appropriate amount of complexity to learn and retain patterns in the data. You can also use enough features to make sure your model captures diverse patterns in the data that differentiate between tasks.
4. **Renate**: This is an open source Python library for automatic model re-training of neural networks. Instead of working with a fixed training set, the library provides continual learning algorithms that can incrementally train a neural network as more data becomes available.

2.3.6 Model Versioning

Benefits of SageMaker Model Registry

a) SageMaker Model Registry



b) Benefits

- **Catalog models for production**
- **Manage model versions**
- **Control the approval status of models within your ML pipeline**

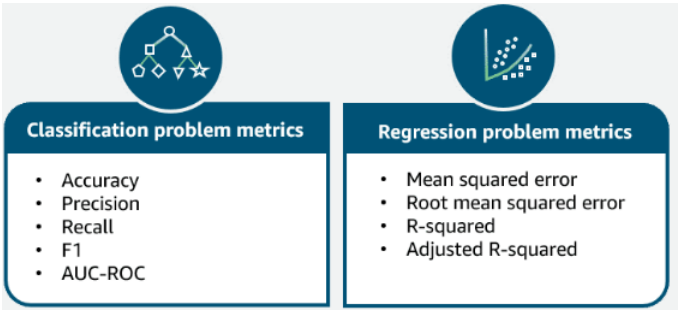
Registering and Deploying models with SageMaker Model Registry

a) SageMaker Model Registry

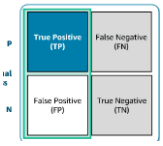
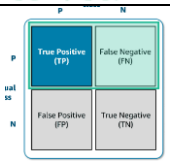
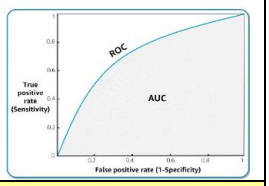
2.4 Analyze Model Performance

2.4.1 Model Evaluation

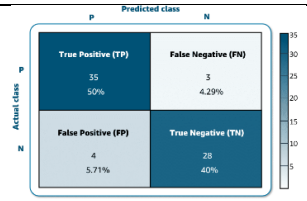
Model Metrics and Evaluation Techniques



a) Classical Algorithm problems

Accuracy	***Precision***	***Recall***	***F1 score***	*AUC Curve*
# of matching predictions to the total number of instances .	Proportion of positive that are correct.	proportion of correct sets that are identified as positive.	precision + recall	D
$\frac{TP+TN}{(TP+TN+FP+FN)}$	$\frac{TP}{TP+FP}$	$\frac{TP}{TP+FN}$	$\frac{2 \cdot Precision \cdot Recall}{Precision + Recall}$	
				
	Cost of false positives is high	Cost of false negatives is high (Better to have false +**ves**)		
	classification Emails as spam or not	(e.g. diagnose cancer)		

New one: Heat Maps

graphical representations that use color coding to visualize the performance of a model		identify the areas of interest that have the most impact when your model is making predictions.

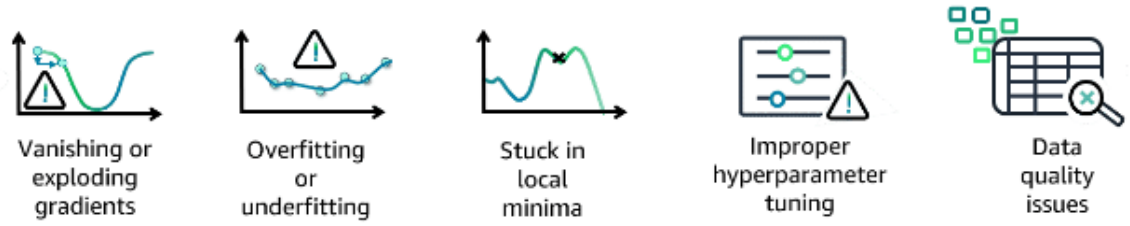
b) Regression Algorithm problems

Metric	Description	When to Use
Mean Squared Error (MSE)	Average of squared differences between predicted and actual values	• When larger errors should be penalized more • For comparing models (lower is better) • When the scale of errors is important
Root Mean Square Error (RMSE)	Square root of MSE, in the same units as the target variable	• When you want the error in the same units as the target variable • For easier interpretation of the error magnitude • When comparing models with different scales
R-Squared (R^2)	Proportion of variance in the dependent variable explained by the independent variables	• To understand how well the model fits the data • When you want a metric bounded between 0 and 1 • For comparing models across different datasets
Adjusted R-Squared	Modified version of R-Squared that adjusts for the number of predictors in the model	• When comparing models with different numbers of predictors • To penalize overly complex models • In feature selection processes

Model Convergence

Convergence refers to the ability of a model to reach an optimal solution during the training process. Failure to converge can lead to suboptimal performance, overfitting, or even divergence, where the model's performance deteriorates over time.

a) Impact of convergence

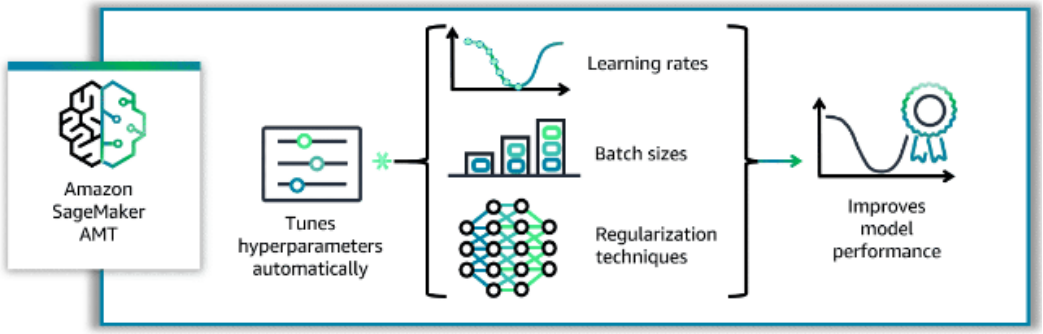


b) How SageMaker AMT (Compiler) helps in convergence issues of convergence

This is where SageMaker AMT can help. It can automatically tune models by finding the optimal combination of hyperparameters, such as

i. learning rate schedules
ii. initialization techniques
iii. regularization methods.

Improve CNN



How SageMaker AMT improves issues with local maxima and local minima

When training a deep CNN for image classification tasks can encounter saddle points or local minima. This is because the loss function landscape in high-dimensional spaces can be complex. Having multiple local minima and saddle points can trap the optimization algorithm, leading to suboptimal convergence.

This is where SageMaker Training Compiler can help. It can automatically apply optimization techniques like

- tensor remapping
- operator fusion
- kernel optimization.

Debug Model Convergence with SageMaker Debugger

SageMaker Clarify and Metrics Overview

Bias metrics give visibility into model evaluation process

Data bias metrics	• **Class Imbalance**: Measures the imbalance in the distribution of classes/labels in your training data. • **Facet Imbalance:** Evaluates the imbalance in the distribution of facets or sensitive attributes, such as age, gender, or race across different classes or labels. • **Facet Correlation:** Measures the correlation between facets or sensitive attributes and the target variable.
Model bias metrics	• **Differential validity:** Evaluates the difference in model performance such as accuracy, precision, and recall across different facet groups. • **Differential prediction bias:** Measures the difference in predicted outcomes or probabilities for different facet groups, given the same input features. • **Differential feature importance:** Analyzes the difference in feature importance across different facet groups, helping to identify potential biases in how the model uses features for different groups.
Model explainability metrics	• **SHAP (SHapley Additive exPlanations):** Provides explanations for individual predictions by quantifying the contribution of each feature to the model's output. • **Feature Attribution**: Identifies the most important features contributing to a specific prediction, helping to understand the model's decision-making process. • **Partial Dependence Plots (PDPs)**: Visualizes the marginal effect of one or more features on the model's predictions, helping to understand the relationship between features and the target variable.
Data quality metrics	• **Missing Data:** Identifies the presence and distribution of missing values in your training data. • **Duplicate Data:** Detects duplicate instances or rows in your training data. • **Data Drift**: Measures the statistical difference between the training data and the data used for inference or production, helping to identify potential distribution shifts.

Domain 3: Selecta a deployment infrastructure

Model building and deployment infrastructure involves the following:

- Handling training and optimizing the machine learning models
- Data processing, feature engineering, model training, validation, and optimization
- Requires compute resources like GPUs for quickly training complex models
- Outputting the final trained model artifacts that will be used for inference

Inference infrastructure involves the following:

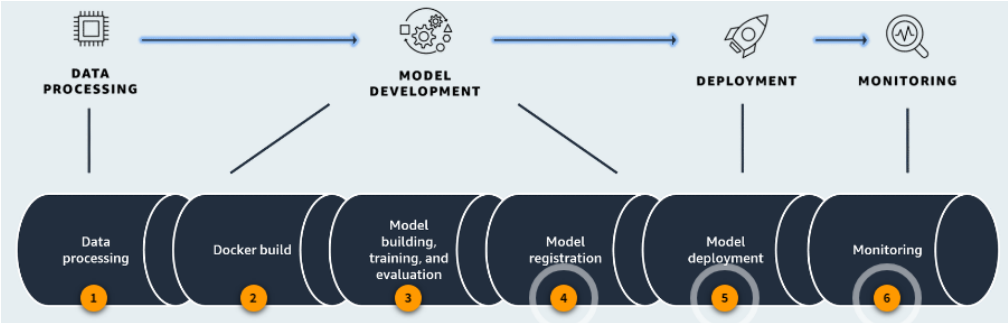
- Hosting the deployed trained models and handles running inference
- Receiving new unseen data, runs it through the models, and returns predictions and results
- Focusing on low latency, high throughput inference serving and scales to handle high query volumes without affecting latency
- Can be hosted on the cloud, on-premises, or at edge locations

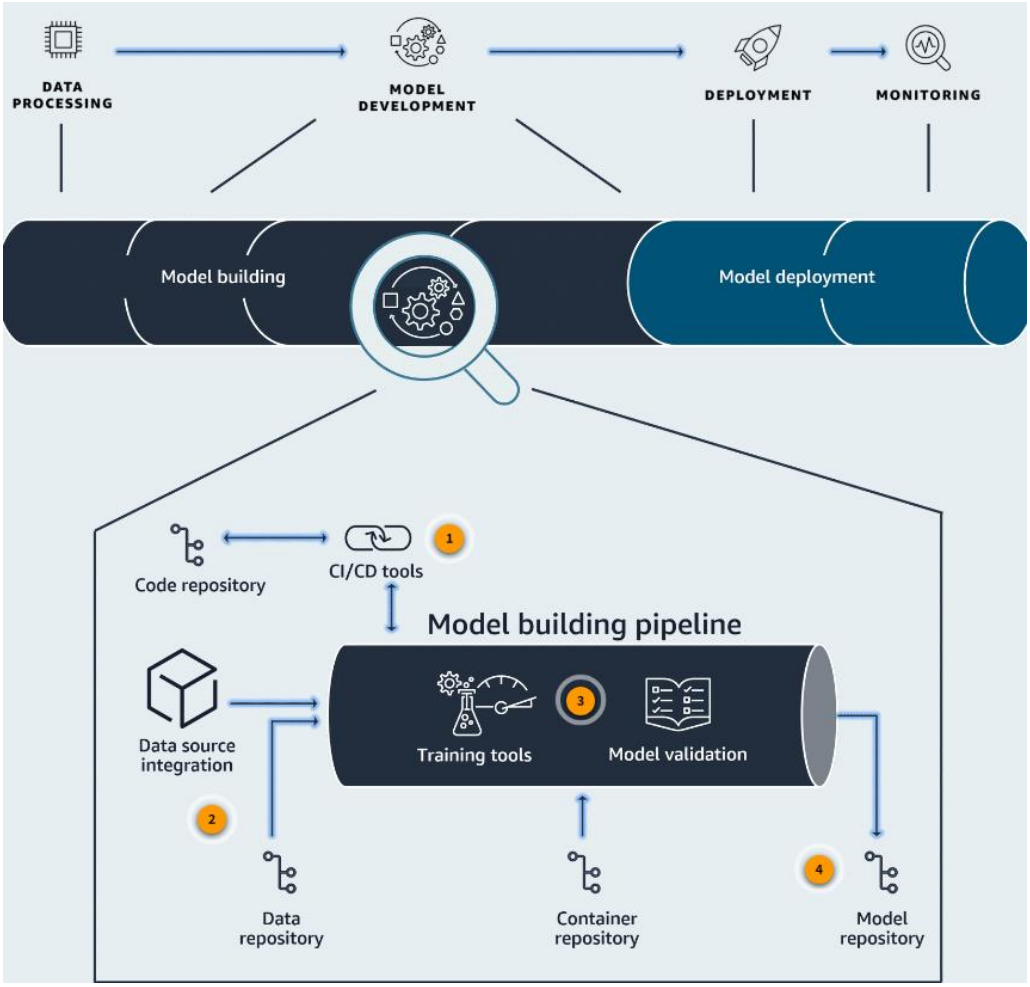
3.1 Select a Deployment Infrastructure

3.1.1 Model building & Deployment Infra

Building a Repeatable Framework

a) Example pipeline sequences -Options





Workflow Orchestration Options

a) Comparisons

Deployment Option	When to Use
SageMaker Pipelines	• When working entirely within the AWS SageMaker ecosystem • For end-to-end ML workflows that need to be automated and managed at scale
AWS Step Functions	• For serverless orchestration of ML pipelines • When you need to integrate ML workflows with other AWS services • For complex workflows with branching and parallel execution • When you want visual representation of workflow
Amazon MWAA (Managed Workflows for Apache Airflow)	• When you're familiar with Apache Airflow and prefer DAG-based workflows • For complex scheduling requirements • When you need to integrate with both AWS and non-AWS services
MLflow	• When you need an open-source platform for the complete ML lifecycle • For tracking experiments, packaging code into reproducible runs, and sharing and deploying models • When working in a multi-cloud or hybrid cloud environment • When you want to use a tool that integrates well with many ML frameworks and libraries
Kubernetes	• For container orchestration of ML workflows and deploying ML models at scale • When you need fine-grained control over resource allocation and scheduling • For multi-cloud or hybrid cloud deployments • When you want to leverage Kubernetes' extensive ecosystem (e.g., Kubeflow for ML-specific workflows

*b) **Comparisons:** AWS Controllers for Kubernetes (ACK) and SageMaker Components for Kubeflow Pipelines.*

AWS Controllers for Kubernetes (ACK)	SageMaker Components for Kubeflow Pipelines
• SageMaker Operators for Kubernetes facilitate the processes for developers and data scientists who use Kubernetes to train, tune, and deploy ML models in SageMaker. • You can install SageMaker Operators on your Kubernetes cluster in Amazon Elastic Kubernetes Service (Amazon EKS). • You can create SageMaker jobs by using the Kubernetes API and command-line Kubernetes tools, such as kubectl.	• You can move your data processing and training jobs from the Kubernetes cluster to the SageMaker ML-optimized managed service. • You have an alternative to launching your compute-intensive jobs from SageMaker. • You can create and monitor your SageMaker resources as part of a Kubeflow Pipelines workflow. • Each of the jobs in your pipelines run on SageMaker instead of the local Kubernetes cluster so you can take advantage of key SageMaker features.

3.1.2 Inference Infrastructure

Deployment Considerations & Deployment Infrastructure

a) Deployment Targets

***Best practice**: When*

	Benefits	Keep in mind	Choose when..	Use case
SageMaker endpoints	• Fully managed service • Convenient to deploy and scale • Built-in monitoring and logging • Supports various ML frameworks	• Not as customizable as other options • Potentially higher cost than other options	• You want a fully managed solution with minimal operational overhead and don't require advanced customization.	• A bank decides to use SageMaker endpoints to deploy ML models that detect fraud.
EKS	• Highly scalable and flexible • Supports advanced deployment scenarios • Supports custom configurations	• Possible higher operational overhead • Steeper learning curve to manage tool effectively	• You need advanced deployment scenarios and customized configurations, and you have the resources to manage the Kubernetes cluster	• A biomedical company uses EKS clusters to process DNA sequencing data. •
ECS	• Managed container orchestration service • Convenient to scale • Integrates well with other AWS services • Can run in Batch mode	• Limited advanced features compared to Kubernetes • Vendor lock-in	• You want a managed container orchestration service with good AWS integration and you don't require advanced Kubernetes features	• A renewable energy firm uses Amazon ECS to scale solar energy forecasting workloads.
Lambda	• Serverless • Automatically scales • Low operational overhead	• Limited run time • Cold starts can impact latency • Not suitable for long-running or complex models	• You have lightweight, low-latency models and want a serverless, pay-per-use solution	• A telehealth company uses Lambda functions for appointment reminders.

Choosing a model inference strategy

a) Amazon SageMaker inference options

SageMaker provides multiple inference options, including real-time, serverless, batch, and asynchronous to suit different workloads.

Inference Option	Description	When to Choose
Real-time	For low latency, high throughput requests	• When you need immediate responses (e.g., real-time fraud detection) • For applications requiring consistent, low-latency predictions • When your model can handle requests within milliseconds • For high-traffic applications with steady request rates
Serverless	Handles intermittent traffic without managing infrastructure	• For unpredictable or sporadic workloads • When you want to avoid managing and scaling infrastructure • For cost optimization in scenarios with variable traffic • For dev/test environments or proof-of-concept deployments
Asynchronous	Queues requests and handles large payloads	• For time-insensitive inference requests • When dealing with large input payloads (e.g., high-resolution images) • For long-running inference jobs (up to 15 minutes) • When you need to decouple request submission from processing
Batch Transform	Processes large offline datasets	• For offline predictions on large datasets • When you need to process data in bulk (e.g., nightly batch jobs) • For scenarios where real-time predictions are not required • When you want to precompute predictions for faster serving

Container and Instance Types for Inference

a) Choosing the right container for Inference

Approach	Description	When to Choose
SageMaker managed container images	Pre-built containers with inference logic included	• When using standard ML frameworks (e.g., TensorFlow, PyTorch, Scikit-learn) • For quick deployment without custom code • When built-in inference logic meets your needs • To leverage SageMaker's optimizations and best practices
Your own inference code	Custom containers with your specific inference logic	• When you need custom preprocessing or postprocessing • For proprietary algorithms or frameworks not supported by SageMaker • When you require specific dependencies or libraries • For full control over the inference environment

b) Choosing the right compute resources (AWS instance)

Instance family	Workload type
t family	Short jobs or notebooks
m family	Standard CPU to memory ratio
r family	Memory-optimized
c family	Compute-optimized
p family	Accelerated computing, training, and inference
g family	Accelerated inference, smaller training jobs
Amazon Elastic Inference	Cost-effective inference accelerators

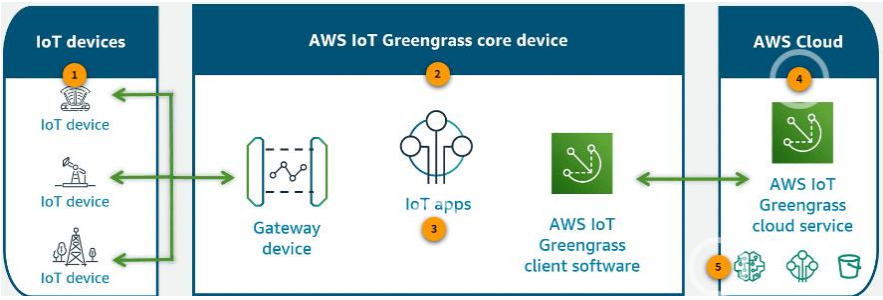
When to choose CPU, GPU of Inf2

Amazon EC2 P5 instances	GPU-based instances:	C2 Inf2 instances:
• High serial performance • Cost efficient for smaller models • Broad support for models and frameworks	• High throughput at desired latency • Cost efficient for high utilization • Good for deep learning, large models	• Accelerator designed for ML inference • High throughput at lower cost than GPUs • Ideal for models that AWS Neuron SDK supports

Optimizing Deployment with Edge Computing

a) *Using edge devices - AWS Options*

AWS IoT Greengrass



Amazon SageMaker Neo

b) *When to use which*

AWS IoT Greengrass	SageMaker Neo
• **Run at the edge**: Bring intelligence to edge devices, such as for anomaly detection in precision agriculture or powering autonomous devices. • **Manage applications:** Deploy new or legacy apps across fleets using any language, packaging technology, or run time. • **Control fleets:** Manage and operate device fleets in the field locally or remotely using MQTT or other protocols. • **Process locally**: Collect, aggregate, filter, and send data locally.	• **Optimize models for faster inference:** SageMaker Neo can optimize models trained in frameworks like TensorFlow, PyTorch, and MXNet to run faster with no loss in accuracy. • **Deploy models to SageMaker and edge devices:** SageMaker Neo can optimize and compile models to run on SageMaker hosted inference platforms, like SageMaker endpoints. As you've learned, it can also help you to run models on edge devices, such as phones, cameras, and IoT devices. • **Model portability:** SageMaker Neo can convert compiled models between frameworks, such as TensorFlow and PyTorch. Compiled models can also be run across different platforms and hardware, helping you to deploy models to diverse target environments. • **Compress model size:** SageMaker Neo quantizes and prunes models to significantly reduce their size, lowering storage costs and improving load times. This works well for compressing large, complex models for production.

3.2 Create and Script Infrastructure

These pillars provide a consistent and scalable designs.

The security pillar

- create ML solutions that anonymize sensitive data, such as personally identifiable information
- guides the configuration of least-privileged access to your data and resources
- suggests configurations for your AWS account structures and Amazon Virtual Private Clouds to provide isolation boundaries around your workloads.

The reliability pillar

- helps construct ML solutions that are resistant to disruption while recovering quickly
- guides you to design data processing workflows to be resilient to failures by implementing error handling, retries, and fallback mechanisms
- **recommends data backups, and versioning**.

The performance efficiency pillar

- focuses on the efficient use of resources to meet requirements.
- help you optimize ML training and tuning jobs by selecting the most suitable EC2 instance types for a particular task, running model inference using edge computing to minimize latency and maximize performance.

The cost optimization pillar

- focuses on building and operating systems that minimize costs
- In the data processing stage -> guides storage resource selection and tools for automation such as Amazon SageMaker Data Wrangler.
- During model development -> rightsizing compute resources
- Finally, during model deployment -> auto scaling

The sustainability pillar

- focuses on environmental impacts (energy consumption, efficient resource usage)

The operational excellence pillar

- focuses on the efficient operation, performance visibility, and continuous improvement

3.2.1 Methods for Provisioning Resources

IAC

a) Tools

Tool	Description	Language Support	Multi-Cloud Support	Typical Use Cases
CloudFormation	AWS-native IaC service	JSON, YAML	AWS only	• AWS-only • Teams familiar with AWS ecosystem • Simple to moderate complexity deployments
CDK (Cloud Development Kit)	IaC framework that compiles to CloudFormation	TypeScript, Python, Java, C#, Go	AWS only (can be extended)	• Teams with strong programming skills • Complex AWS infrastructures • Reusable ML infrastructure components
Terraform	Open-source IaC tool	HCL, JSON	Excellent	• Multi-cloud ML deployments • Hybrid cloud scenarios • Teams preferring declarative syntax
Pulumi	Modern IaC platform	TypeScript, Python, Go, .NET	Excellent	• Requires complex logic • Teams preferring familiar programming languages • Multi-cloud, complex architectures

Working with CloudFormation

a) Template

"**AWSTemplateFormatVersion**" : "2010-09-09"	**Format version** This first section identifies the AWS CloudFormation template version to which the template conforms.
"**Description**" : "Write details on the template."	**Description** This text string describes the template.

"**Metadata**" : {
"**Instances**" : {"Description" : "Info on instances"},
"**Databases**" : {"Description" : " Info about dbs"}
}
	Metadata
	These objects provide additional information about the template.

"**Parameters**" : {
"**InstanceTypeParameter**" : {
"Type" : "String",
"Default" : "t2.micro",
"**AllowedValues**" : ["t2.micro", "m1.small"],
"Description" : "Enter t2.micro or m1.small"
}
}
	Parameters
	Values **passed to your template** when you create or update a stack. You can refer to parameters from the Resources and Outputs sections of the template.

"**Rules**" : {
"Rule01": {
"**RuleCondition**": {
...
},
"**Assertions**": [
...
]}
}
	Rules
	Rules validate parameter values passed to a template during a stack creation or stack update.

"**Mappings**" : {
"Mapping01" : {
"Key01" : {
"**Name**" : "Value01"
}, ...
}}
	Mappings
	These are map keys and associated values that you can use to specify conditional parameter values. This is similar to a lookup table

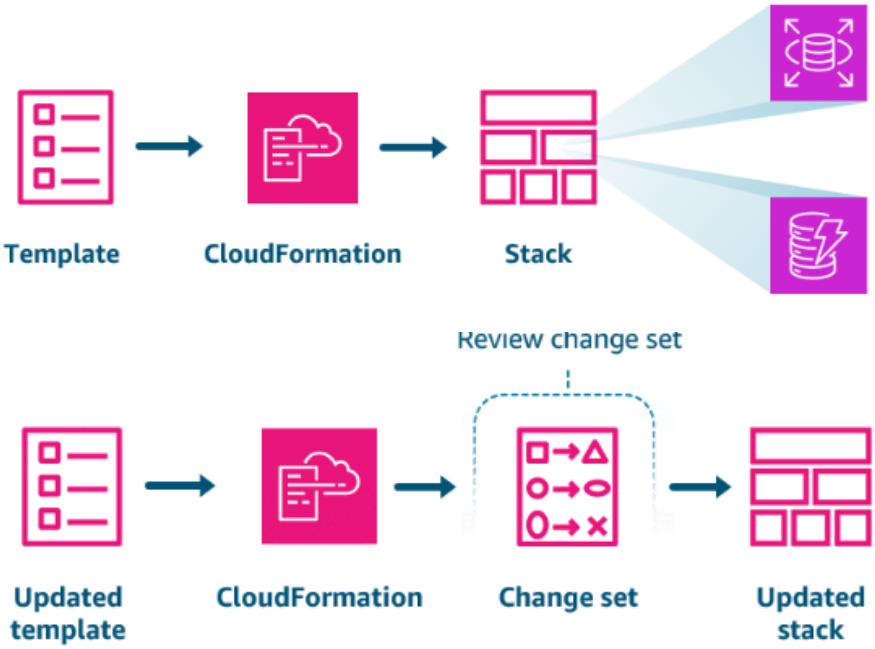
"**Conditions**" : {
"MyLogicalID" : {*Intrinsic function*}
}
	Conditions
	Control whether certain resources are created, or whether certain resource properties are assigned a value during stack creation or an update.

"**Transform**" : {
set of transforms
}
	Transform
	For **serverless** applications, transform specifies the version of the AWS SAM to use.

"**Resources**" : {
"Logical ID of resource" : {
"**Type**" : "Resource type",
"**Properties**" : {
Set of properties
}}
}
	Resources
	This section specifies the stack resources, and their properties that you would like to provision. You can refer to resources in the Resources and Outputs sections of the template.
	Note: This is the only required section of the template.

"**Outputs**" : {
"Logical ID of resource" : {
"**Description**" : "Information on the value",
"**Value**" : "Value to return",
"**Export**" : {
"**Name**" : "Name of resource to export"
}}}
	Outputs
	Describe the values that are returned whenever **you** view your stack's properties. For example, you can declare an output for an Amazon S3 bucket name and then call the aws cloudformation describe-stacks AWS CLI command to view the name**.**

b) CF Stacks



c) Provisioning stacks using CloudFormation templates

$ **aws cloudformation** create-stack \
  --**stack-name myteststack** \
  --**template-body** file:///home/testuser/**mytemplate.json** \
  --**parameters** ParameterKey=Parm1,ParameterValue=test1 ParameterKey=Parm2,ParameterValue=test2

Working with CDK

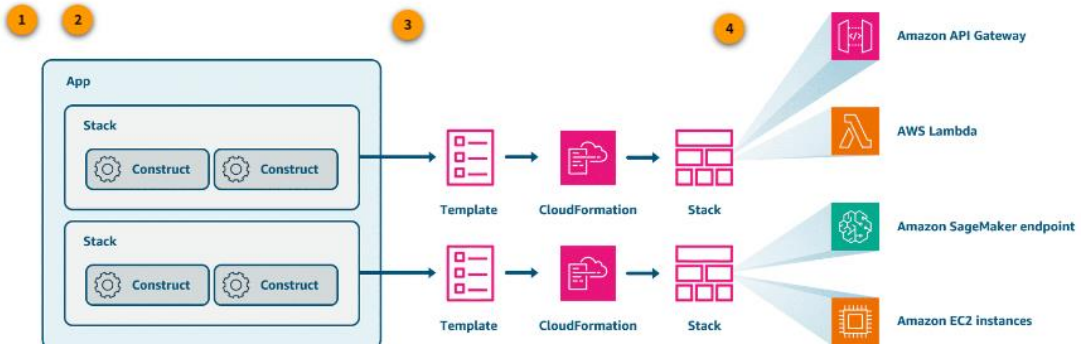
The AWS CDK consists of two primary parts:

- **AWS CDK Construct Library:** This library contains a collection of pre-written modular and reusable pieces of code called constructs. These constructs represent infrastructure resources and collections of infrastructure resources.
- **AWS CDK Toolkit:** This is a command line tool for interacting with CDK apps. Use the AWS CDK Toolkit to create, manage, and deploy your AWS CDK projects.

a) CDK Construct level comparisons

Level	*Description*	*Abstraction*	*Ease of Use*	*Typical Use Case*
L1	*Direct CF resources representation*	*Low*	*Low*	*full control over CF resources*
L2	*Logical grouping of L1 resources*	*Medium*	*Medium*	*most common*
L3	*High-level abstractions that represent complete solutions*	*High*	*High*	*Quickly deploy common architectural patterns*

b) CDK LifeCycle



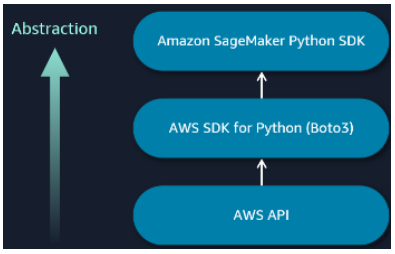
1	**cdk init** When you begin your CDK project, you create a directory for it, run **cdk init**, and specify the programming language used: • **mkdir** my-cdk-app • **cd** my-cdk-app • **cdk init** app --language typescript
2	**cdk bootstrap** You then run **cdk bootstrap** to prepare the environments into which the stacks will be deployed. This creates the special dedicated AWS CDK resources for the environments.
3	**cdk synth** Creates the CloudFormation templates using the **cdk synth** command.
4	**cdk deploy** Finally, you can run the **cdk deploy** command to have CloudFormation provision resources defined in the synthesized templates.

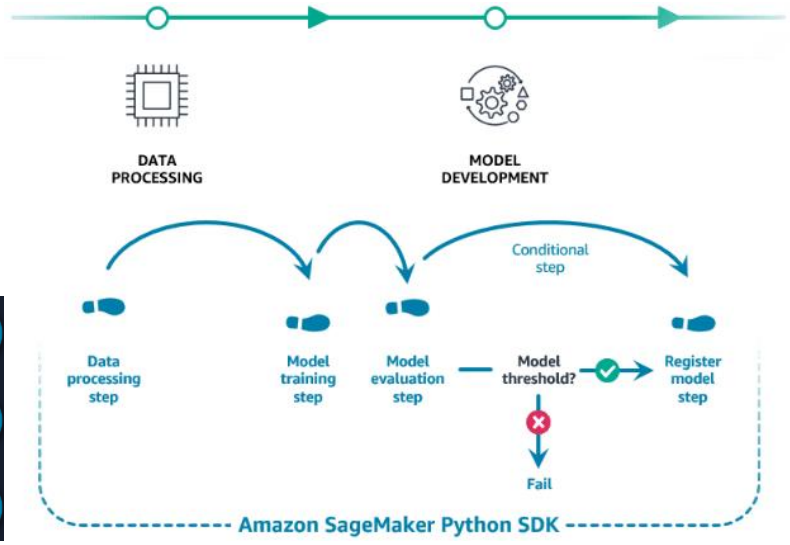
Comparing CF and CDK

The AWS CDK consists

	AWS CloudFormation	AWS CDK
Authoring experience	CF only uses JSON or YAML templates to define your infrastructure resources.	M modern programming languages, like **Python, TypeScript, Java, C#, and Go**.
IaC approach	CloudFormation templates are declarative. You define the desired state of your infrastructure and CloudFormation handles the provisioning and updates.	AWS CDK provides an imperative approach to generating CloudFormation templates, which are declarative, means you can introduce logic and conditions that determine the resources to provision in your infrastructure.
Debugging and troubleshooting	Troubleshooting CloudFormation templates requires learning specific CloudFormation error handling and messages.	With the CDK, you can use the debugging capabilities of your chosen programming language, making it more convenient to identify and fix issues in your infrastructure code.
Reusability and modularity	create nested stacks and cross-stack references, resulting in modular and reusable infrastructure designs. However, this approach can become complex and difficult to manage as your infrastructure grows.	Supports programming languages that you can use to apply object-oriented programming principles. This makes it more convenient to create modular and reusable IaC code blocks for your infrastructure.
Community support	CloudFormation has been around for a longer time and has a larger community for support. It also has a variety of third-party tools and resources.	Newer offering than AWS CloudFormation, but it is rapidly gaining adoption.
Learning curve	steeper learning curve for developers who are used to a more programmatic approach over a template-driven approach.	If you're already familiar with programming languages like Python or TypeScript, AWS CDK will have a gentler learning curve.

3.2.2 Deploying and Hosting Models

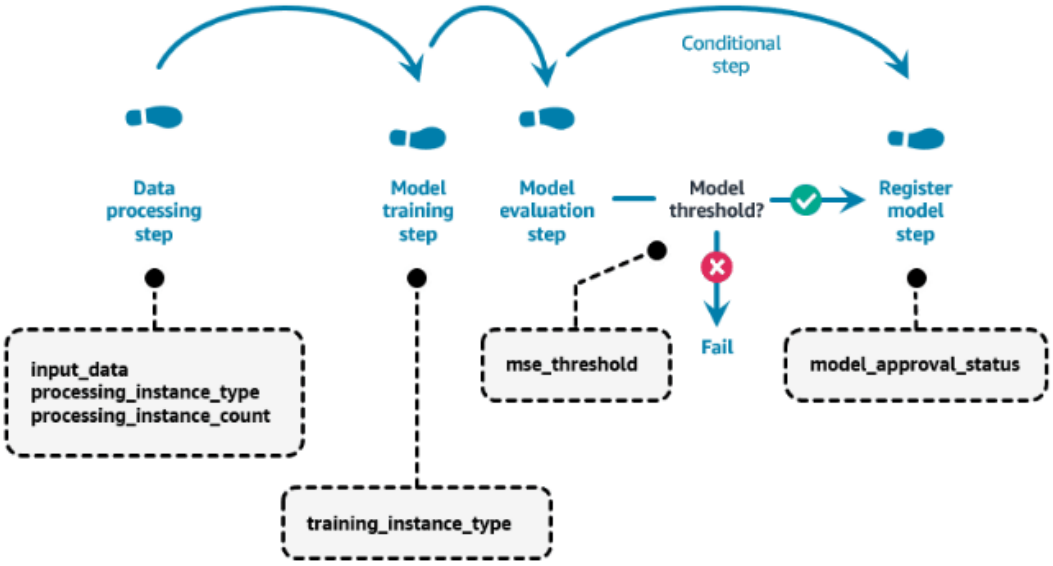




SageMaker Python SDK

a) Creating pipelines with the SageMaker Python SDK to orchestrate workflows

```
pipeline = Pipeline(
    name=pipeline_name,
    parameters=[input_data, processing_instance_type,
                processing_instance_count, training_instance_type,
                mse_threshold, model_approval_status],
    steps = [step_process, step_train, step_evaluate, step_conditional]
)
```



b) Automating common tasks with the SageMaker Python SDK

Preparing data (the.run())

With Amazon SageMaker Processing, you can run processing jobs for data processing steps in your machine learning pipeline. Processing jobs accept data from Amazon S3 as input and store data into Amazon S3 as output.

I. Creating the Processor

To define a processing job, you first create a Processor. The following example instantiates the **SKLearnProcessor()** class, which streamlines using **scikit-learn** in your data processing step:

```
from sagemaker.sklearn.processing import SKLearnProcessor

sklearn_processor = SKLearnProcessor(
framework_version="0.20.0",
role="[Your SageMaker-compatible IAM role]",
instance_type="ml.m5.xlarge",
instance_count=1,
)
```

II. Running the Processor

You then use the **.run()** method on the processor to run a processing job.

```
from sagemaker.processing import ProcessingInput, ProcessingOutput

sklearn_processor.run(
    code="preprocessing.py",
    inputs=[
        ProcessingInput(source="s3://your-bucket/path/to/your/data",
destination="/opt/ml/processing/input"),
    ],
    outputs=[
        ProcessingOutput(output_name="train_data",
source="/opt/ml/processing/train"),
        ProcessingOutput(output_name="test_data", source="/opt/ml/processing/test"),
    ],
    arguments=["--train-test-split-ratio", "0.2"],
)
```

III. Adding the data preprocessing step to a SageMaker Pipeline

Finally, you define a data preprocessing step in your pipeline using **ProcessingStep()**:

```
step_process = ProcessingStep(
    name="MyProcessingStep",
    ...,
    step_args = sklearn_processor.run(
    code="preprocessing.py",
    inputs=[
        ProcessingInput(source="s3://your-bucket/path/to/your/data",
destination="/opt/ml/processing/input"),
    ],
    outputs=[
        ProcessingOutput(output_name="train_data",
source="/opt/ml/processing/train"),
        ProcessingOutput(output_name="test_data", source="/opt/ml/processing/test"),
    ],
    arguments=["--train-test-split-ratio", "0.2"],    ),
)
```

Training Models (the.fit())

You can run model training jobs using the SageMaker Python SDK. The following model training job example manages the training script, framework, training instance, and training data input.

I. Creating the estimator for the training job

To define a model training job, you instantiate the **estimator** class. This class encapsulates training on SageMaker. The following code creates a model training job using the **MXNet()** class to train a model using the MXNet framework:

```
from sagemaker.mxnet import MXNet

mxnet_estimator = MXNet('train.py',
                        role='SageMakerRole',
                        instance_type='ml.p2.xlarge',
                        instance_count=1,
                        framework_version='1.2.1')
```

II. Running the training job

After you create the estimator, you can then use the **.fit()** method to run the training job. This method takes an argument that identifies the path to the training data. In this example, the training dataset is stored in Amazon S3:

```
mxnet_estimator.fit('s3://my_bucket/my_training_data/')
```

III. Creating a model training step in SageMaker Pipelines

Finally, you define a model training step in your pipeline using the **TrainingStep()** method:

```
step = TrainingStep(
    name="MyTrainingStep",
    step_args= mxnet_estimator.fit('s3://my_bucket/my_training_data/')
)
```

Deploying Models (deploy())

You can use SageMaker Python SDK to deploy a SageMaker model endpoint using the **deploy()** and **predict()** methods. You start by defining your endpoint configuration. The following code shows the configuration for a serverless endpoint:

```
from sagemaker.serverless import ServerlessInferenceConfig
serverless_config = ServerlessInferenceConfig(
    memory_size_in_mb=4096,
    max_concurrency=10,
)
```

You use this configuration in a **deploy()** method. If the model is already created, you use the **Model** class to create a SageMaker model from a model artifact. The model artifact location in Amazon S3 and the code to use to perform inference as the **entry_point** is specified:

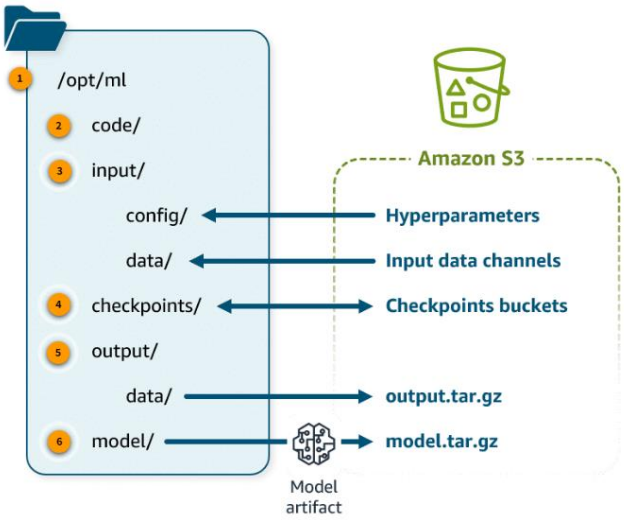
```
serverless_predictor = model.deploy(serverless_inference_config=serverless_config)
```

After deployment is complete, you can use the predictor's **predict()** method to invoke the serverless endpoint:

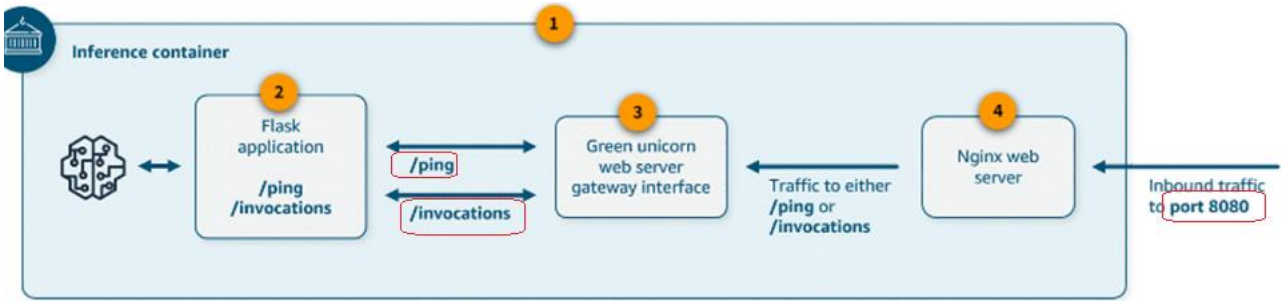
```
response = serverless_predictor.predict(data)
```

c) Building and Maintaining Containers

Training Container



Inference container



Below example is for Python script

Point	File
1	**serve.py** when the container is started for hosting. It starts the inference server, including the nginx web server and Gunicorn as a Python web server gateway interface.
2	**predictor.py** This Python script contains the logic to load and perform inference with your model. It uses Flask to provide the /ping and /invocations endpoints.
3	**wsgi.py** This is a wrapper for the Gunicorn server.
4	**nginx.conf** This is a script to configure a web server, including listening on port 8080. It forwards requests containing either /ping or /invocations paths to the Gunicorn server.

When creating or adapting a container for performing real-time inference, your container must meet the following requirements:

- Your container must include the path **/opt/ml/model**. When the inference container starts, it will import the model artifact and store it in this directory.

  Note: This is the same directory that a training container uses to store the newly trained model artifact.

- Your container must be configured to run as an executable. Your Dockerfile should include an ENTRYPOINT instruction that defines an executable to run when the container starts, as **ENTRYPOINT** ["<language>", "<executable>"]
  e.g. **ENTRYPOINT** ["python", "serve.py"]

- Your container must have a web server listening on **port 8080**.

- Your container must accept **POST** requests to the **/invocations** and **/ping** real-time endpoints.

- The requests that you send to these endpoints must be returned with 60 seconds and have a size <6 MB.

Auto scaling strategy

a) SageMaker model auto scaling methods

Scaling Method	Description	Use Case	Key Features
Target tracking scaling policy	Adjusts capacity to maintain a specified metric near a target value	When you want to maintain a specific metric (e.g., CPU utilization) at a target level	• Specify a metric and target value • Automatically adds or removes capacity • Good for maintaining consistent performance
Step scaling policy	Defines multiple policies for scaling based on specific metric thresholds	When you need more granular control over scaling actions at different metric levels	• Define multiple thresholds and corresponding scaling actions • More aggressive response to demand changes • Allows fine-tuning of scaling behavior
Scheduled scaling policy	Scales resources based on a predetermined schedule	When demand follows a predictable pattern (daily, weekly, monthly, yearly)	• Set one-time or recurring schedules • Use cron expressions with start and end times • Ideal for known traffic patterns
On-demand scaling	Manually increase or decrease the number of instances	For unpredictable or one-off events that require manual intervention	• Full manual control over scaling • Useful for new product launches, unexpected traffic spikes, or special promotions • Flexibility to respond to unforeseen circumstances

3.3 Automate Deployment

3.3.1 Introduction to DevOps

Code repositories

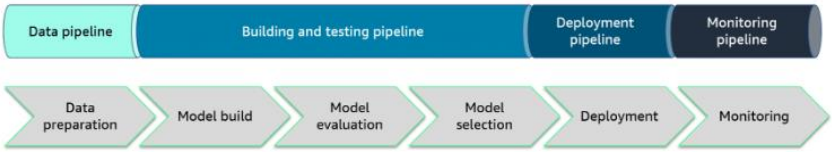
a) GitHub vs GitLab

Feature	GitHub	GitLab
Hosting Options	Cloud-hosted, GitHub Enterprise (self-hosted)	Cloud-hosted, Self-hosted (Community and Enterprise Editions)
CI/CD	GitHub Actions (built-in)	GitLab CI/CD (built-in)
Project Management	Projects, Kanban boards	Issue boards, Epics, Roadmaps
Third-party Integrations	Extensive marketplace	Fewer, but strong built-in tools
Open Source	Many open-source projects	Fully open-source core

3.3.2 CI/CD: Applying DevOps to MLOps

Introduction to MLOps

a) CICD in ML Lifecycle



b) Teams in the ML process

- **Data engineers**: Data engineers are responsible for data sourcing, data cleaning, and data processing. They transform data into a consumable format for ML and data scientist analysis.
- **Data scientists:** Responsible for model development including model training, evaluation, and monitoring to drive insights from data.
- **ML engineers: R**esponsible for MLOps - model deployment, production integration, and monitoring. They standardize ML systems development (Dev) and ML systems deployment (Ops) for continuous delivery of high-performing production ML models.

c) Nonfunctional requirements in ML

- **Consistency**
- **Flexibility**: Accommodates a wide range of ML frameworks and technologies to adapt to changing requirements.
- **Reproducibility**
- **Reusability**:
- **Scalability**:
- **Auditability**: Provides comprehensive logs, versioning, and dependency tracking of all ML artifacts for transparency and compliance.
- **Explainability**: Incorporates techniques that promote decision transparency and model interpretability.

d) Comparing the ML workflow with DevOps

Feature	DevOps	MLOps
Code versioning	✓	✓
Compute environment	✓	✓
CI/CD	✓	✓
Monitoring in production	✓	✓
Data provenance		✓
Datasets		✓
Models		✓
Model building with workflows		✓
Model deployment workflows		✓

Automating testing in CI/CD Pipelines
SageMaker projects with CI/CD practices

- **Unit tests**
  validate smaller components like individual functions or methods.
- **Integration tests**
  can check that pipeline stages, including data ingestion, training, and deployment, work together correctly. Other types of integration tests depend on your system or architecture.
- **Regression tests**
  In practice, regression testing is re-running the same tests to make sure something that used to work was not broken by a change.

Version Control Systems: Getting started with Git
SageMaker projects with CI/CD practices

Command	Description
git init	Initializes a new Git repository in the current directory
git clone [repository_url]	Creates a local copy of a remote repository
git add [file(s)]	Stages the specified file(s) for the next commit
git commit -m "commit message"	Records the staged changes with a descriptive commit message
git push	Uploads local committed changes to a remote repository
git pull	Fetches and merges changes from a remote repository into the local branch
git branch	Lists all available branches in the repository
git checkout [branch_name]	Switches to the specified branch
git merge [branch_name]	Merges the specified branch into the current branch

Continuous Flow Structures : Automate deployment

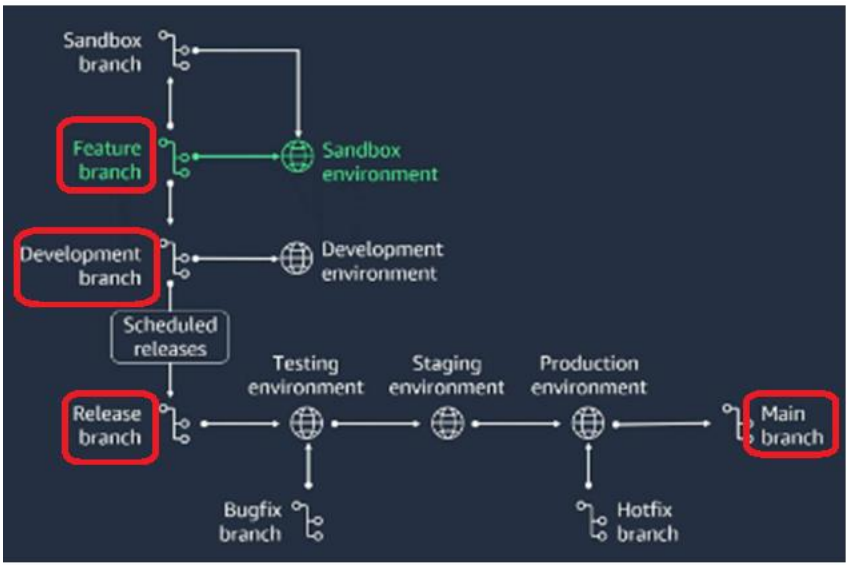
a) *Key components*

1) ***Model training and versioning:***
2) ***Model packaging and containerization:***
3) ***Continuous integration (CI):***
4) ***Monitoring and observability:***
5) ***Rollback and rollforward strategies:***

b) *Gitflow and GitHub flows*

Feature	Gitflow	GitHub Flow
Complexity	More complex	Simpler
Main Branches	main and develop	Single main branch
Feature Development	Feature branches from develop	Feature branches from main
Release Process	Dedicated release branches	Direct to main via pull requests
Hotfixes	Separate hotfix branches	Treated like features
Suited For	Scheduled releases, larger projects	Continuous delivery, smaller projects
Integration Branch	develop branch	N/A (uses main)
Learning Curve	Steeper	Flatter
Flexibility	More rigid structure	More flexible

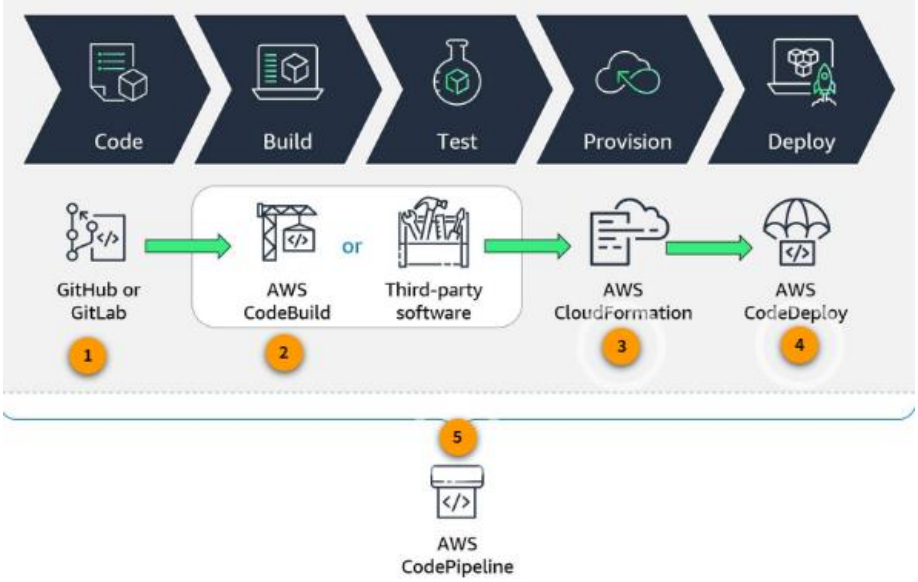
c) *GitFlow*



3.3.3 AWS Software Release Processes

Continuous Delivery Services

a) AWS CI/CD Pipeline



CodePipeline *Provides configurable manual approval gates to control releases, detailed monitoring capabilities, and granular permissions to manage pipeline access, as we will explore further.*	• ***Each AWS account: 1000 pipelines*** • ***The number of actions in a single pipeline: 500.*** • ***The size of input artifact for a single action: 1 GB*** • ***#of custom actions for/region/account: 50.*** • ***# of webhooks/region/account : 300.***
CodeBuild *CodeBuild sets service quotas and limits on builds and compute fleets. These quotas are for each supported AWS Region for each AWS account.*	• ***detailed logging, auto-scaling capacity, and high availability for builds*** • ***integrates with other AWS services like CodePipeline and ECR for end-to-end CI/CD workflows*** • ***artifacts can be stored in S3 or other destinations*** • ***build can be monitored through the CodeBuild console, Amazon CloudWatch, and other methods*** • ***fine-grained access controls for build projects using IA) policies.***
CodeDeploy *CodeDeploy is a deployment service that provides automated deployments, flexible deployment strategies, rollback capabilities, and integration with other AWS services to help manage the application lifecycle across environments.*	• ***facilitates automated deployments to multiple environments*** • ***supporting deployment strategies like blue/green, in-place, and canary deployments*** • ***provides rollback capabilities,*** • ***detailed monitoring and logging, and integration with services like EC2, Lambda, and ECS.***

Best Practices for Configuring & Troubleshooting

Service	Purpose	Key Configuration Steps	Troubleshooting Tips
CodeBuild	Compiles source code, runs unit tests, and produces deployment-ready artifacts	1. Create a CodeBuild project 2. Define build specification 3. Configure build environment 4. Set up build artifacts 5. Configure CloudWatch Logs (optional)	• Validate buildspec file • Verify IAM permissions • Review service limits • CloudTrail **Unique** • Check for network issues • Check CodeBuild logs
CodeDeploy	Automates application deployments to various compute platforms	1. Set up IAM role 2. Create CodeDeploy app 3. Create deployment group 4. Define deployment configuration	**Unique** • Review CodeDeploy logs • Verify CodeDeploy agent • Validate AppSpec file • Check instance health • Analyze the rollback reason
CodePipeline	Models, visualizes, and automates software release steps	1. Create CodePipeline pipeline 2. Add source stage 3. Add build stage 4. Add deploy stage 5. Review and create pipeline	• Validate build specifications ((buildspec.yml) • Verify input configuration • Use CloudTrail • Check the IAM permissions **Unique** • Use CloudWatch • Examine runtime details • Check pipeline history

Automating Data Integration in ML Pipeline

Code Pipeline vs Step Functions

Aspect	*AWS CodePipeline*	*AWS Step Functions*
Primary Purpose	*CI/CD* and release automation	Workflow orchestration and coordination
Workflow Type	Linear, predefined stages	Complex, branching workflows with conditional logic
Best For	Standard software deployment pipelines	Complex, multi-step processes and microservices orchestration

MLOps with Code Pipeline and Step Functions

- **MLOps Overview:**
  - Set of practices and tools for streamlining ML model deployment, monitoring, and management
  - Focuses on automating ML workflows in production environments
- **AWS Step Functions:**
  - Fully managed visual workflow service for building distributed applications
  - Represents pipeline stages (preprocessing, training, evaluation, deployment) as task states
  - Manages control flow between stages
  - Can integrate with Lambda, AWS Batch, and other AWS services
- **AWS CodePipeline:**
  - Fully managed continuous delivery service
  - Automates release pipelines for MLOps workflows
  - Represents each stage of the pipeline as an action
- **Integration of Step Functions and CodePipeline:**
  a) CodePipeline invokes MLOps pipeline based on events (e.g., new model version commit)
  b) Pipeline stages include source code management, model building, testing, and deployment
  c) CodePipeline starts Step Functions state machine to initiate MLOps workflow
  d) Can pass input data or parameters to configure the workflow
- **Benefits of Integration:**
  - Enables efficient movement of models through development lifecycle
  - Automates the entire process from training to production deployment
  - Provides flexibility in configuring and managing complex ML workflows

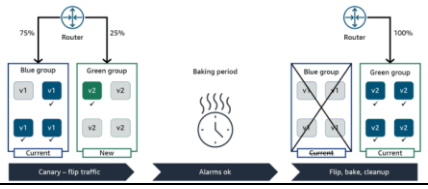
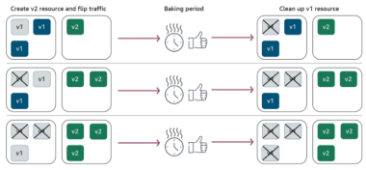
Deployment strategies

All at once
Shifts all endpoint traffic from the existing blue fleet to the new green fleet in a single step. The pre-specified CloudWatch alarms then monitor the green fleet for issues during the baking period. If no alarms trip, SageMaker terminates the old blue fleet after the baking period.

Canary
Traffic is shifted to a new fleet in two steps. First, a small portion of traffic (the canary) is shifted to the new fleet and monitored during a baking period. If the canary succeeds, the remainder of traffic is shifted from the old fleet to the new fleet, and the old fleet is terminated.

Linear
Traffic is shifted gradually in a pre-specified number of equal, incremental steps. The number of steps and percentage of traffic shifted at each step can be customized. This extends basic canary shifting from 2 steps to n linearly spaced steps for more gradual and customizable traffic shifting.

Comparison deployment

Blue/green deploy	***Canary deployment***	***Rolling deployment***
maintaining two identical prod environments, one blue (existing) and one green (new), and gradually shifting traffic between them.	Gradually rolling out a new model version to a small portion of users	Gradually replace previous deployment of model version with new version by updating endpoint in configurable batch sizes
		
• When you need instant rollback capability • For critical applications requiring zero downtime • When your application can handle sudden traffic shifts	• To test new features with a subset of users • When you want to gather user feedback before full release • For applications with high traffic where you want to minimize risk	• When you have stateful application • For large-scale deployments where cost is an issue • When you can tolerate having mixed versions temporarily

The baking period is a set time for monitoring the green fleet's performance before completing the full transition, making it possible to rollback if alarms trip. This period builds confidence in the new deployment before permanent cutover.

3.3.4 Retraining models

Retraining models

a) Retraining mechanisms

Automated retraining pipelines	Invoke model retraining when new data becomes available	
Scheduled retraining	Periodic jobs to retrain the model at regular intervals	
Drift detection and invoked retraining	invoke retraining when the model's performance starts to degrade.	SageMaker Model Monitor + Lambda (SageMaker Model Monitor can detect model drift, and Lambda can be used to initiate the retraining process.)
Incremental learning	Incremental learning allows the model to be updated with new data without completely retraining the model from scratch.	• XGBoost • Linear Learner (AWS SageMaker supports several algorithms for this, as above)
Experimentation and A/B testing	Retraining can be paired with experimentation and A/B testing to compare various model versions	AWS SageMaker + Personalize (AWS SageMaker and Amazon Personalize can be used to deploy and manage these experiments.)

b) Catastrophic forgetting during retraining and transfer learning

Catastrophic forgetting is a phenomenon that occurs in machine learning models, particularly in the context of continual or lifelong learning.

***Catastrophic forgetting is a type of over-fitting*.** The model learns the training data too well such that it no longer performs well on other data.

Why does catastrophic forgetting happen?

a. **Retraining and optimized training:** The primary reason for catastrophic forgetting is that the parameters of the model are typically updated to optimize for the current task. These updates effectively overwrite the knowledge acquired from previous tasks.
b. **Transfer Learning:** Transfer learning is an ML approach where a pre-trained model, which was trained on one task, is fine-tuned for a related task. Organizations can make use of transfer learning to retrain existing models on new, related tasks using a smaller dataset.

Solving catastrophic forgetting

Method	Main Idea	Advantages	Limitations
Rehearsal-based	Retrain on a subset of old data along with new data	• Directly addresses forgetting • Conceptually simple	• Requires storing old data • Can be computationally expensive
Architectural	Modify network architecture to accommodate new tasks	• Can be very effective • Doesn't require old data	• May increase model complexity • Can be challenging to design
Replay-based	Generate synthetic data to represent old tasks	• Doesn't need storing old data • Work well with Gen AI models	• Quality depends on model • May not capture all aspects of old data
Regularization-based	Add constraints to limit changes to important parameters	• Doesn't require old data or architecture changes • Often computationally efficient	• May limit learning of new tasks • Determining importance can be challenging

Configuring Inferencing Jobs

a) Inference types

Real-time inference
Real-time inference is ideal for inference workloads where you have real-time, interactive, low latency requirements. These endpoints are fully managed and support autoscaling. For more information, see Real-time inferencing.

Serverless inference
Serverless inference can be used to serve model inference requests in real time without directly provisioning compute instances or configuring scaling policies to handle traffic variations. For more information, see Serverless Inference.

Asynchronous inference
Asynchronous inference queues incoming requests for asynchronous processing. This option is ideal for requests with large payload sizes, long processing times, and near-real-time latency requirements. The latency requirement is one reason to choose asynchronous over batch. For more information, see

Batch transform
Batch transform provides offline inference for large datasets. Batch transform is helpful for preprocessing datasets or gaining inference from large datasets. For more information, see Use Batch Transform.

Differences between training and inferencing

Training	*Inferencing*
Requires high parallelism with large batch processing for higher throughput	*Usually run on a single input in real time*
More compute- and memory- intensive	*Less compute- and memory-intensive*
Standalone item not integrated into application stack	*Integrated into application stack workflows*
Runs in the cloud	*Runs on different devices at the edge and cloud*
Typically runs less frequently and on an as-needed basis	*Runs **an indefinite amount of time***
Compute capacity requirements are typically predictable, so wouldn't require auto scaling	*Compute capacity requirements might be dynamic and unpredictable, **so would require auto scaling***

Domain 4: Monitor Model

4.1 Monitor Model Performance and Data Quality

4.1.1 Monitoring Machine Learning Solutions

Importance of Monitoring in ML

a) Machine Learning Lens: AWS Well-Architected Framework: Best practices and design principles

***Best practice**: When*

Optimize resources	***Resource pooling***	*Sharing compute, storage, and networking resources*
	Caching	
	Data management	*data compression, partitioning, and lifecycle management*
Scale ML workloads based on demand	***AWS Auto Scaling***	*SageMaker built-in scaling. AWS Auto-Scaling*
	Lambda	
Reduce Cost	***Monitor usage and costs***	*resource tagging*
	monitor ROI	
Enable continuous improvement	***Establish Feedback Loops***	
	Monitor Performance	*SageMaker Model Monitor (Drift) CloudWatch alerts (deviations)*
	Automate Retraining	

Detecting Drift in Monitoring

a) Drift Types

Drift Type	*Description*	*Causes*	*Implications*
Data Quality Drift	Production data distribution differs from training data distribution	• Real-world data not as curated as training data • Changes in data collection processes • Shifts in real-world conditions	• Model accuracy decreases • Predictions become less reliable
Model Quality Drift	Model predictions differ from actual ground truth labels	• Changes in the underlying relationship between features and target • Model decay over time • Concept drift	• Decreased model performance • Inaccurate predictions
Bias Drift	Increase in bias affecting model predictions over time	• Training data too small or not representative • Incorporation of societal assumptions in training data • Exclusion of important data points • Changes in real-world data distribution	• Model overgeneralization • Unfair or discriminatory predictions • Ethical concerns • New groups in production
Feature Attribution Drift	Changes in the contribution of individual features to model predictions	• Shifts in feature importance over time • Changes in the underlying problem domain • Introduction of new, more predictive features	• Model may rely on less relevant features • Decreased interpretability • Potentially reduced performance

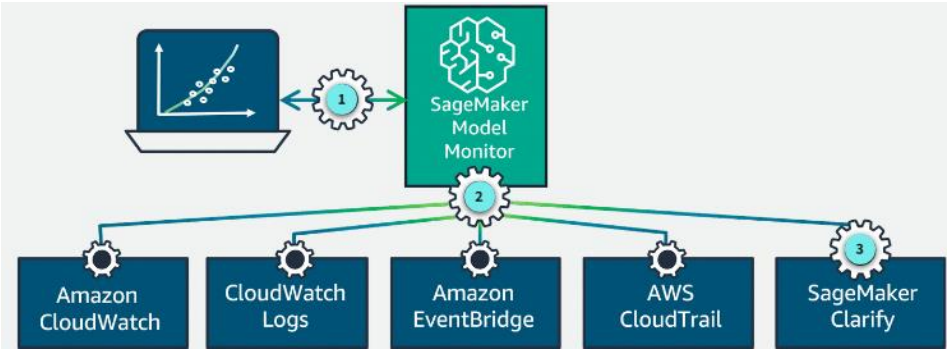
Note: *Bias inverse of variance, which is the level of small fluctuations or noise common in complex data sets. Bias tends to cause model predictions to overgeneralize, and variance tends to cause models to undergeneralize. Increasing variance is one method for reducing the impact of bias.*

b) Monitoring Drift

Monitoring Type	What It Monitors	How It Monitors
Data Quality Monitoring	• Missing values • Outliers • Data types • Statistical metrics (mean, std dev, etc.) • Data distribution	• Implement data validation checks • Calculate statistical metrics • Compare metrics with baseline values • Use data drift detection techniques (e.g., Kolmogorov-Smirnov tests, Maximum Mean Discrepancy)
Model Quality Monitoring	• Evaluation metrics (accuracy, precision, recall, F1, AUC, etc.) • Prediction confidence • Performance across different subpopulations	• Calculate evaluation metrics on held-out test set or production data sample • Implement confidence thresholding or uncertainty estimation • Flag low-confidence predictions • Monitor performance on different data subsets
Model Bias Drift Monitoring	• Bias metrics (disparate impact, fairness, etc.) • Performance across sensitive groups	• Calculate bias metrics for different sensitive groups • Compare bias metrics with baseline values or thresholds • Implement bias mitigation techniques (e.g., adversarial debiasing, calibrated equalized odds)
Feature Attribution Drift Monitoring	• Feature importance scores • Statistical metrics of feature attributions	• Use interpretability techniques (e.g., SHAP) to calculate feature attributions • Calculate statistical metrics on feature attributions • Compare metrics with baseline values • Identify features with significantly changed attributions

SageMaker Model Monitor

Integration



SageMaker - Monitoring for Data Quality Drift

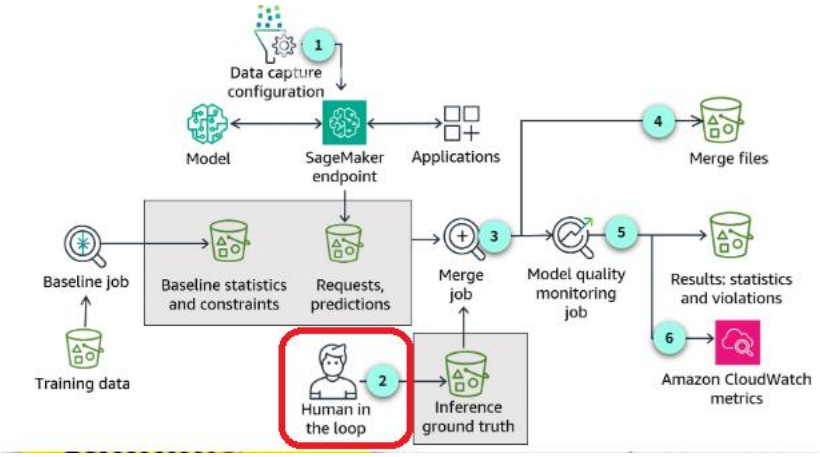
STEPS

1. **Initiate data capture on the endpoint**
2. **Create a baseline**

   start a baseline processing job with the suggest_baseline method of the ModelQualityMonitor object using the SageMaker Python SDK.
3. **Schedule data quality monitoring jobs**
4. **Integrate data quality monitoring with Cloudwatch**
5. **Interpret results and analyze findings**

   The report is generated as the **constraint_violations.json** file. The SageMaker Model Monitor prebuilt container provides the following violation checks.
   - data_type_check
   - completeness_check
   - baseline_drift_check
   - missing_column_check
   - extra_column_check
   - categorical_values_check

SageMaker - Monitoring for Model Quality Drift using Model Monitor



To monitor model quality, SageMaker Model Monitor requires the following inputs:

1. Baseline data
2. Inference input and predictions made by the deployed model
3. Amazon SageMaker Ground Truth associated with the inputs to the model

SageMaker - Monitoring for Bias using Clarify

Statistical bias drift occurs when the data used for training differs from the data encountered during prediction, leading to potentially biased outcomes. This is prominent when training data changes over time. In this lesson, you will learn about AWS services that help you monitor for statistical bias drift.

Post-training bias metrics in SageMaker Clarify help us answer two key questions:

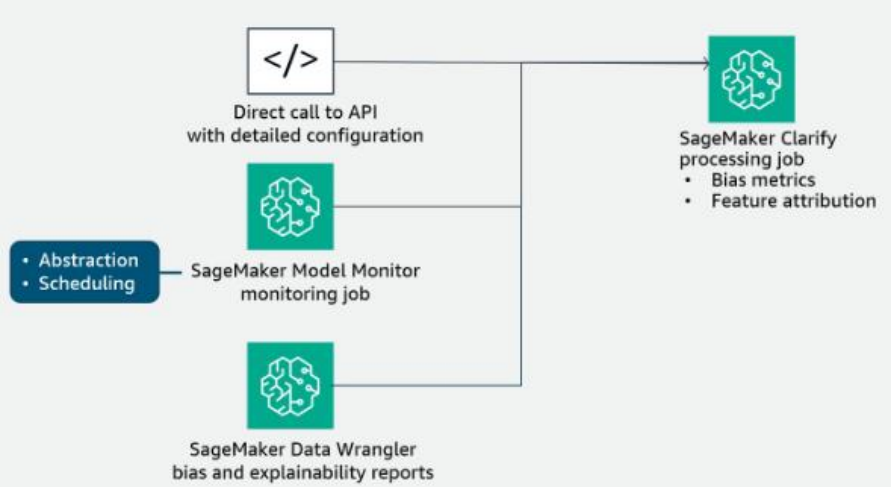
- Are all facet values represented at a similar rate in positive (favorable) model predictions?
- Does the model have similar predictive performance for all facet values?

SageMaker Model Monitor automatically does the following:

- **Merges** the prediction data with SageMaker Ground Truth labels
- **Computes** baseline statistics and constraints
- **Inspects** the merged data and generates bias metrics and violations
- **Emits** CloudWatch metrics to set up alerts and triggers
- **Reports and alerts** on bias drift detection
- **Provides** reports for visual analysis

How it works: It quantifies the contribution of each input feature (for example, audio characteristics) to the model's predictions, helping to explain how the model arrives at its decisions.

Options for using SageMaker Clarify

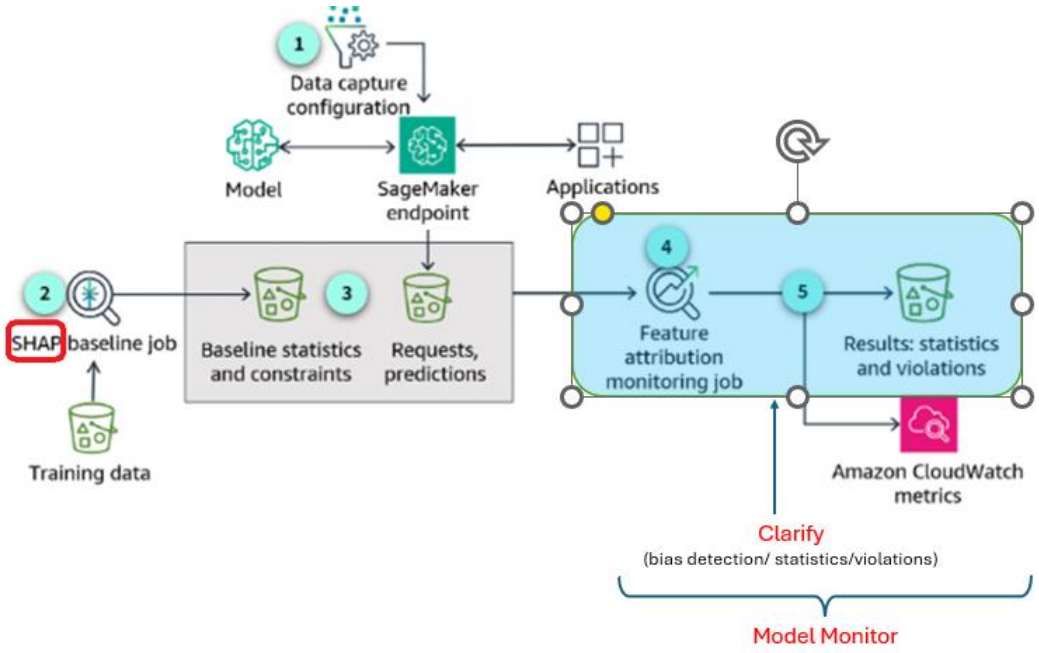


When to use which

Method	Description	When to Use
SageMaker Clarify Directly	Configure and run Clarify processing job using SageMaker Python SDK API	• For one-time or ad-hoc bias analysis • When you need full control over the analysis configuration • For integrating bias analysis into custom workflows
SageMaker Model Monitor + Clarify	Integrate Clarify with Model Monitor for continuous bias monitoring	• When you want to automate bias detection in production • If you need to set up alerts for bias drift
SageMaker Data Wrangler	Utilize Clarify within Data Wrangler during data preparation	• During the data preparation phase • When you want to identify potential bias early in the ML pipeline • If you're already using Data Wrangler for data preprocessing

SageMaker - Monitoring for Feature Attribution Drift (Model Monitor + Clarify)

Feature attribution refers to understanding and quantifying the contribution or influence of each feature on the model's predictions or outputs. It helps to identify the most relevant features and their relative importance in the decision-making process of the model.



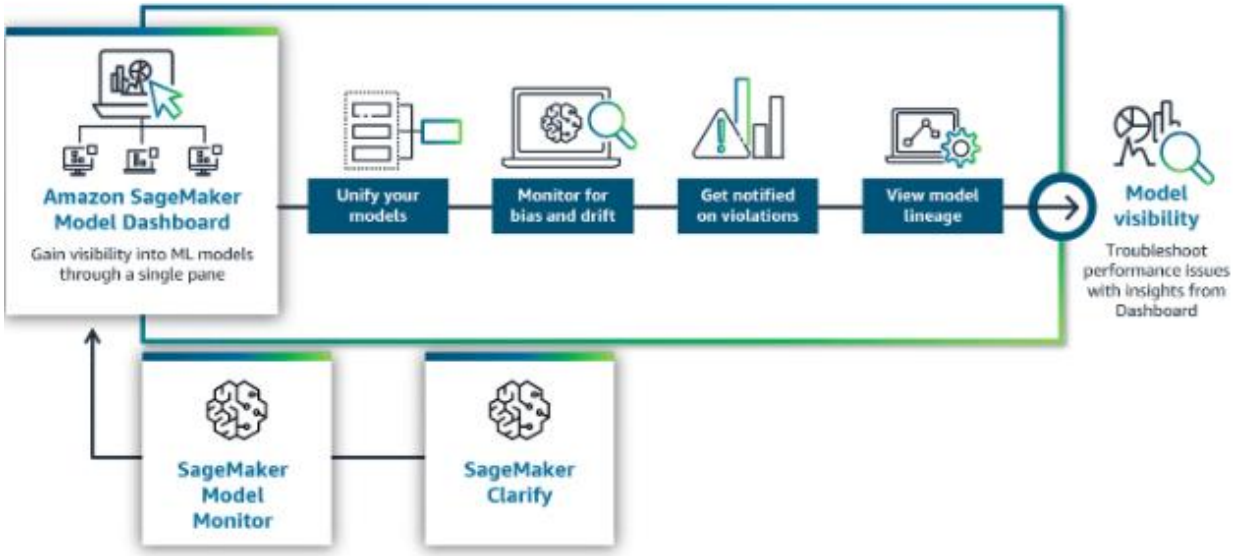
Uses SHAP

SageMaker Clarify provides feature attributions based on the concept of Shapley value. This is a game-theoretic approach that assigns an importance value (SHAP value) to each feature for a particular prediction.

Here's how it works:

1. **SageMaker Clarify:** This is the core component that performs the actual bias detection and generate quality metrics and violations
2. **SageMaker Model Monitor:** This is the framework that can use Clarify's capabilities to perform continuous monitoring of deployed models.

SageMaker Model Dashboard



Features

1. **Alerts** :

   How it helps: The dashboard provides a record of all activated alerts, allowing the data scientist to review and analyze past issues.

Alert criteria depend upon two parameters:

- **Datapoints to alert**: Within the evaluation period, how many runtime failures raise an alert?
- **Evaluation period:** The # of most recent monitoring executions to consider when evaluating alert status.

2. **Risk rating**

A user-specified parameter from the model card with a low, medium, or high value.

3. **Endpoint performance**

You can select the endpoint column to view performance metrics, such as:

- **CpuUtilization**: The sum of each individual CPU core's utilization from 0%-100%.
- **MemoryUtilization**: The % of memory used by the containers on an instance, 0%-100%.
- **DiskUtilization**: The % of disk space used by the containers on an instance, 0%-100%.

4. **Most recent batch transform job**

This information helps you determine if a model is actively used for batch inference.

5. **Model lineage graphs**

When training a model, SageMaker creates a **model lineage graph**, a visualization of the entire ML workflow from data preparation to deployment.

6. **Links to model details**

The dashboard links to a model details page where you can explore an individual model.

Model Monitor vs SageMaker Dashboard vs Clarify: When to use which one

Tool	*Description*	*Why to use*	*When to Use*
Model Monitor	Continuous monitoring of ML models in production	• data and model quality issues • model drift	• To set up automated alerts for performance degradation • When you need to monitor resource utilization • Monitor real-time endpoints, batch transform, On-demand monitoring job
SageMaker Dashboard	Centralized view of SageMaker resources and jobs	•	• For a high-level overview of all SageMaker activities • To track training jobs, endpoints, and notebook instances
SageMaker Clarify	Bias detection and model explainability tool	• Detecting Bias • Triggers statistics and Violations report	• To detect bias in training data and model predictions • When you need to explain model decisions • For regulatory compliance requiring model transparency • To improve model fairness and accountability

4.1.2 Remediating Problems Identified by Monitoring

Automated remediations and notifications

- **Stakeholder notifications:** When monitoring metrics indicate changes that impact business KPIs or the underlying problem
- **Data Scientist notification:** You can use automated notifications to **data scientists** when your monitoring detects data drift or when expected data is missing.
- **Model retraining:** Configure your model training pipeline to **automatically retrain models when monitoring detects drift, bias, or performance degradation**.
- **Autoscaling:** You use resource utilization metrics gathered by infrastructure monitoring to initiate autoscaling actions.

Model retraining strategies

Strategy	When to Use	Advantages	Considerations
Event-driven	• When drift is detected above a certain threshold • In response to significant changes in data or performance	• Timely response to changes • Efficient use of resources	• May be frequent if thresholds are too sensitive • Retraining can be expensive and time-consuming
On-demand	• When market conditions change significantly • In response to new competitors or strategies	• Allows for human judgment in decision-making • Can incorporate business context	• Requires constant monitoring by data scientists or stakeholders • May lead to delayed responses
Scheduled	• When there are known seasonal patterns • For maintaining model accuracy over time	• Predictable maintenance schedule • Can anticipate and prepare for retraining periods	• May retrain unnecessarily if no significant changes occur • Might miss sudden, unexpected changes

4.2 Monitor and Optimize Infrastructure and Costs

4.2.1 Monitor Infrastructure

Monitor Performance Metrics - CloudWatch vs Model Monitor

Feature	SageMaker Model Monitor	CloudWatch Logs
Purpose	*Continuous monitoring of ML models in production*	*Monitoring, storing, and accessing log files*
Key Capabilities	(all 4 ML monitoring types) • Data quality monitoring • Model quality monitoring • Bias drift monitoring • Feature attribution drift monitoring	• Log collection from various sources • Log storage in S3 • Pattern recognition • Log anomaly detection
Monitoring Types	• Real-time endpoint monitoring • Batch transform job monitoring • On-schedule monitoring for async batch jobs	• EC2 instances • CloudTrail • Amazon Route 53 • Other sources
Alert System	Set alerts for deviations in model quality	Notifications based on preset thresholds
Customization	• Pre-built monitoring capabilities (no coding) • Custom analysis options	Customizable log patterns and anomaly detection

Monitoring vs. Observability

	Monitoring	Observability
Definition	Continuous collection and analysis of metrics	Deep insights into internal state and behavior of ML systems
Focus	Detecting anomalies and deviations	Understanding complex interactions and dependencies
Key Activities	• Collecting metrics • Logging • Alerting	• Analyzing system behavior • Identifying root causes • Reasoning about system health
Techniques	• • Metric collection • Threshold-based alerting • Basic log analysis	• Distributed tracing • Structured logging • Advanced data visualization
Outcome	Detect issues and invoke alerts or automated actions	Provide deeper insights for troubleshooting and optimization
Scope	Primarily focused on predefined metrics and thresholds	Enables asking and answering questions about system behavior

Monitoring Tools (for Performance and Latency)

Feature	AWS X-Ray	CloudWatch Lambda Insights	CloudWatch Logs Insights	QuickSight
Purpose	Trace information about responses and calls in applications	In-depth performance monitoring for Lambda fns only	Interactive log analytics service	BI and data visualization service
Key Features	• Works across AWS and third-party services • Generates detailed service graphs • Identifies performance bottlenecks	• Monitors metrics (memory, duration, invocation count) • Provides detailed logs and traces • Helps identify bottlenecks in Lambda functions	• Interactive querying and analysis of log data • Correlates log data from different sources • Visualizes time series data • Supports aggregations, filters, and regex	• Interactive dashboards • ML-powered insights • Supports various data sources
Compatible Services	EC2, ECS, Lambda, Elastic Beanstalk	Lambda	Any service that generates logs in CloudWatch	Various AWS services and external data sources
ML Use Cases	• Analyze bottlenecks in ML systems • Trace requests in ML applications (e.g., chatbot inference)	• Monitor and optimize ML models deployed as Lambda functions • Identify root causes of Lambda function issues	• Analyze logs from ML workloads • Identify patterns and anomalies in ML system behavior	• Create dashboards for ML experiment results • Analyze and present insights from ML predictions
Visualization	Service maps, Trace views	Performance dashboards, Trace details	Time series graphs, Log event views	Interactive dashboards, Charts, Graphs
Primary Benefit	End-to-end request tracing and bottleneck identification	Detailed Lambda function performance insights	Flexible, interactive log analysis and visualization	Comprehensive data visualization and business intelligence

SageMaker w/ EventBridge

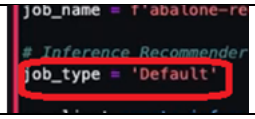
Actions that can be automatically invoked using EventBridge:

a) Invoking an AWS Lambda function
b) Invoking Amazon EC2 run command (not create or deploy)
c) Relaying event to Kinesis Data Streams
d) Activating an AWS Step Functions state machine.
e) **Notifying** SNS topic or an AWS Server Migration Service (AWS SMS) queue.

4.2.2 Optmize Infrastructure

Inference Recommender types

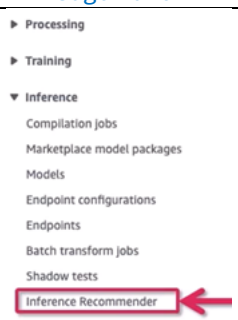
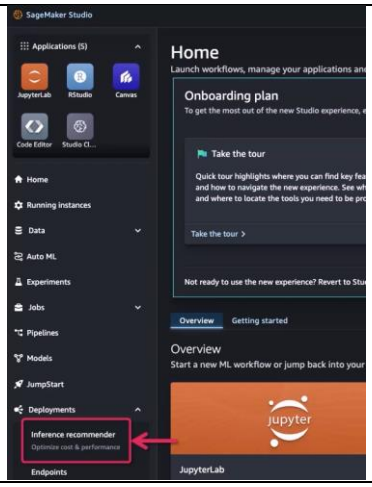
a) Inference Recommendation Types

Default	Advanced
Endpoint Recommender	Endpoint Recommender + Inference Recommender
45 mins	2 hours
	

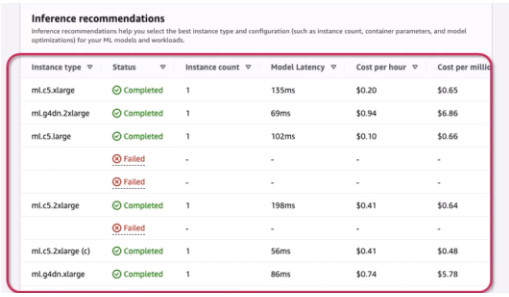
b) Endpoint Recommender vs Inference Recommender

	Endpoint Recommender	Inference Recommender
Output	list (or ranking) of prospective instances	Same
	run a set of load tests.	based on a custom load test.
What you need to do	**-N/A -**	your desired ML instances or a serverless endpoint, provide a custom traffic pattern, and provide requirements for latency and throughput

c) How to start

SageMaker	Inference Recommender
	

d) Sample Recommender output



4.2.3 Optmize Costs

Inference Recommender types

Option	Description	Best For	Cost Savings	Example Use Case
Spot Instances	Spare EC2 capacity at lower prices; can be interrupted	Interruptible workloads	Up to 90% vs On-Demand	Data preprocessing or batch processing
On-Demand Instances	Pay-per-use with no long-term commitment	Short-term, unpredictable workloads	None (baseline)	Real-time inference services
Reserved Instances	Discounted rates for 1 or 3-year commitments	Steady-state, predictable workloads	Up to 72% vs On-Demand	Long-running ML training jobs
Capacity Blocks	Reserved capacity for AWS Outposts or Wavelength Zones	Ensuring capacity during peak demand	Varies	ML workloads in on-premises environments
Savings Plans for SageMaker	Commit to a specific compute usage for 1 or 3 years	Flexible, recurring SageMaker usage	Up to 64% vs On-Demand	Regular model training and deployment

4.3 Secure AWS ML Resources

4.3.1 Securing ML Resources

Access Control using IAM

a) Roles vs Policies

Category	Type	Description	Key Responsibilities/Features
User Roles	Data Scientist/ ML Engineer	Provides access for experimentation	Access to S3, Athena , SageMaker Studio
	Data Engineer	Provides access for data management	Access to S3, Athena, AWS Glue, EMR
	MLOps Engineer	Provides access for ML operations	Access to SageMaker, CodePipeline, CodeBuild, CloudFormation, ECR, Lambda, Step Functions
Service Roles	SageMaker Execution	Allows SageMaker to perform tasks on behalf of users	General SageMaker operations
	Processing Job	Specific to SageMaker processing jobs	Data processing tasks
	Training Job	Specific to SageMaker training jobs	Model training tasks
	Model	Specific to SageMaker model deployment	Model deployment and hosting
IAM Policies	Identity-based	Attached to IAM users, groups, or roles	Define actions allowed on specific resources
	Resource-based	Attached to resources (e.g., S3 buckets)	Control who can access specific resources

IAM Policy – Examples for ML workflows

ID	Purpose	Key Permissions	Resource Scope	Notes
1	Least privilege access for ML workflow	• SageMaker: CreateTrainingJob, CreateModel • S3: GetObject, PutObject • ECR: BatchGetImage • CloudWatch: PutMetricData `"Effect": "Allow", "Action": [ "sagemaker:CreateTrainingJob", "sagemaker:DescribeTrainingJob", "sagemaker:StopTrainingJob" ], "Resource": "*" }`	Specific ARNs for each service	Adheres to principle of least privilege
2	Read metadata of ML resources	• machinelearning:Get* • machinelearning:Describe* `"Version": "2012-10-17", "Statement": [{ "Effect": "Allow", "Action": ["machinelearning:Get*"], "Resource": ["arn:aws:machinelearning:us-east-1:555555555555:mlmodel/-ML-5555" "arn:aws:machinelearning:us-east-1:666666666666:mlmodel/-ML-6666" "arn:aws:machinelearning:us-east-1:777777777777:mlmodel/-ML-7777" "arn:aws:machinelearning:us-east-1:888888888888:mlmodel/-ML-8888" "arn:aws:machinelearning:us-east-1:555555555555:mlmodel/-ML-5555"] }, { "Effect": "Allow", "Action": ["machinelearning:Describe*"], "Resource": ["*"`	Specific MLModel ARNs for Get* * (all) for Describe*	Allows reading metadata but not modifying resources
3	Create ML resources	• machinelearning:CreateDataSourceFrom* • machinelearning:CreateMLModel • machinelearning:CreateBatchPrediction • machinelearning:CreateEvaluation	* (all)	Cannot be restricted to specific resources
4	Manage real-time endpoints and predictions	• machinelearning:CreateRealtimeEndpoint • machinelearning:DeleteRealtimeEndpoint • machinelearning:Predict	Specific MLModel ARN	Allows management of endpoints for a specific model

Detailed examples

1. *identity-based policy used in a machine learning use case*

```
{
"Version": "2012-10-17",
"Statement": [
{
"Effect": "Allow",
"Action": [
"s3:GetObject",
"s3:PutObject"
],
"Resource": [
"arn:aws:s3:::ml-data-bucket/*"
]
},
{
"Effect": "Allow",
"Action": [
"sagemaker:CreateTrainingJob",
"sagemaker:DescribeTrainingJob",
"sagemaker:StopTrainingJob"
],
```

```
{
"Effect": "Allow",
"Action": [
"cloudwatch:PutMetricData"
],
"Resource": "*",
"Condition": {
"StringEquals": {
"cloudwatch:namespace": "SageMaker"
}
}
},
{
"Effect": "Allow",
"Action": [
"logs:CreateLogGroup",
"logs:CreateLogStream",
"logs:PutLogEvents"
],
"Resource": [
"arn:aws:logs:*:*:log-group:/aws/sagemaker/*"
```

2. *Allow users to read machine learning resources metadata*

```
"Statement": [{
  "Effect": "Allow",
  "Action": [
    "machinelearning:Get*"
  ],
  "Resource": [
    "arn:aws:machinelearning:us-east-1:555555555555:mlmodel/-ML-5555"
    "arn:aws:machinelearning:us-east-1:666666666666:mlmodel/-ML-6666"
    "arn:aws:machinelearning:us-east-1:777777777777:mlmodel/-ML-7777"
    "arn:aws:machinelearning:us-east-1:888888888888:mlmodel/-ML-8888"
    "arn:aws:machinelearning:us-east-1:555555555555:mlmodel/-ML-5555"
  ]
},
{
  "Effect": "Allow",
  "Action": [
    "machinelearning:Describe*"
  ],
  "Resource": [
    "*"
```

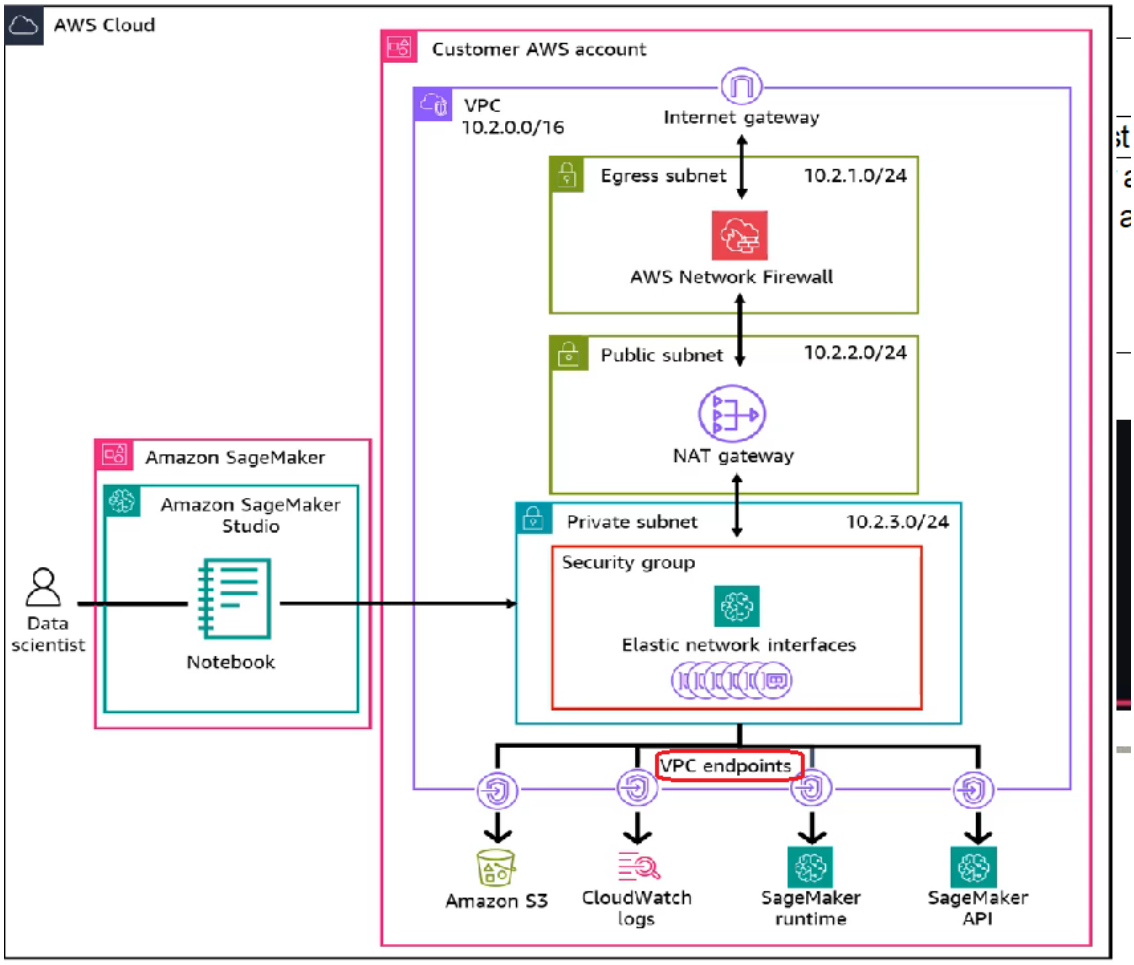
3. *Allow users to create machine learning resources*

```
"Version": "2012-10-17",
"Statement": [{
  "Effect": "Allow",
  "Action": [
    "machinelearning:CreateDataSourceFrom*",
    "machinelearning:CreateMLModel",
    "machinelearning:CreateBatchPrediction",
    "machinelearning:CreateEvaluation"
  ],
  "Resource": [
    "*"
  ]
```

4. *Allow users to create /delete real-time endpoints and perform real-time predictions on an ML model*

```
"Version": "2012-10-17",
"Statement": [{
  "Effect": "Allow",
  "Action": [
    "machinelearning:CreateRealtimeEndpoint",
    "machinelearning:DeleteRealtimeEndpoint",
    "machinelearning:Predict"
  ],
  "Resource": [
        "arn:aws:machinelearning:us-east-1:555555555555:mlmodel/-ML-5555'
  ]
```

Detailed examples



To ensure access only from VPC, **use VPC Endpoints for:**

- S3
- CloudWatch Logs
- SageMaker runtime
- SageMaker API

4.3.3 SageMaker Compliance & Governance

AWS Services for Compliance and Governance

Service	Purpose	Key Features	ML-Related Use Case
AWS Artifact	Provide on-demand access to AWS compliance reports and agreements	• Self-service portal • Access to compliance documentation	Access HIPAA compliance reports for healthcare ML projects
AWS Config	Monitor & Evaluate AWS resource configurations	• Continuous monitoring • Automated configuration evaluation	Monitor SageMaker resource configurations for compliance with security policies
Audit Manager	Continuously audit AWS usage for risk and compliance assessment	Streamlined auditing process, against regulations and standards	Assess compliance of ML workflows with industry standards
Security Hub	View of security alerts and posture	Centralized security alerts	Monitor security posture across ML workflows and resources
Amazon Inspector	Automated vulnerability management	Continuous scanning for vulnerabilities	Scan container images in ECR for ML model deployments
AWS Service Catalog	Create and manage catalogs of pre-approved resources	Governance-compliant resource catalogs	Create catalogs of compliant SageMaker resources and ML models

Amazon SageMaker Governance Tools Summary

Tool	*Purpose*	*Key Features*
SageMaker Role Manager	*Simplify access control*	• Define minimum permissions for ML activities • Quick setup & Streamlined access management
SageMaker Model Cards	*Document and share model information*	• Record intended uses • Document risk ratings
SageMaker Model Dashboard	*Provide overview of models*	• Unified view of all models in account • Monitor model behavior in production
SageMaker Assets	*Streamline ML asset management*	• Publish ML and data assets • Share assets across teams
Model Governance and Explainability	*Ensure compliance and transparency*	• Protect data and workloads • Ensure compliance with standards • Enhance model interpretability

Compliance certifications and regulatory Frameworks

Governance /Framework	***Description***	***AWS Services to Use***
ISO 27001	Information Security Management System standard	• AWS Config • AWS Security Hub
SOC 2	Service Organization Control for service organizations	• AWS Artifact • AWS Config • SageMaker Model Cards
PCI-DSS	Payment Card Industry Data Security Standard	• AWS Config • AWS WAF • Amazon Inspector
HIPAA	Health Insurance Portability and Accountability Act	• AWS Artifact • AWS Security Hub • AWS Config
FedRAMP	Federal Risk and Authorization Management Program	• AWS CloudTrail • AWS Config

Note: AWS Config common to all

4.3.3 Security Best Practices for CI/CD Pipelines

CI/CD pipeline stages

***Best practice**: When*

,

CI/CD Stage	*Security Tools/Practices*
Pre-Commit	• pre-commit hooks (scripts) • IDE plugins to ○ analyze code, detect issues ○ provide recommendations for improvements. ○ handle linting, formatting, beautifying, and securing *code*.
Commit	*Static Application Security Testing (SAST),*
Build	*Software Composition Analysis (SCA)* ○ identifies the open-source packages used in code ○ defining vulnerabilities and potential compliance-based issues ○ scan infrastructure as code *(IaC) manifest files*
Test	• Dynamic Application Security Testing (DAST) • Interactive Application *Security Testing (IAST)* ○ *Combine the advantages of SAST and DAST* tools.
Deploy	• Penetration testing
Monitor	• Red/Blue/Purple teaming

4.3.4 Implement Security & Compliance w/ Monitoring, Logging and Auditing

CloudTrail for ML Resource Monitoring and Logging

Use Case	Description	Benefits
Compliance Auditing	Generate audit trails using CloudWatch Logs and CloudTrail	• Demonstrate compliance with regulations • Meet internal policy requirements
Resource Optimization	Monitor resource utilization metrics	• Optimize ML workloads • Prevent resource abuse and DoS attacks
Incident Response	Investigate and respond to security incidents	• Identify unauthorized access attempts • Detect and respond to data breaches
Anomaly Detection	Implement ML models to detect unusual patterns	• Identify potential security threats • Detect deviations in monitoring data

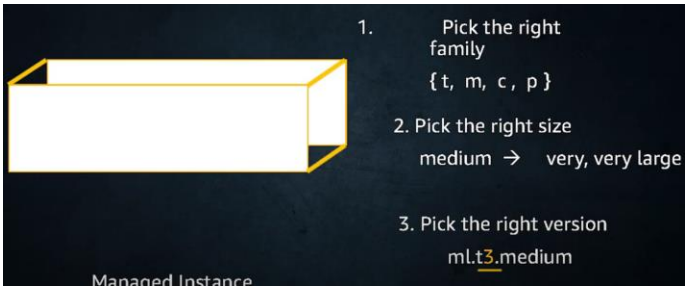
SageMaker Security Troubleshooting and Debugging Summary

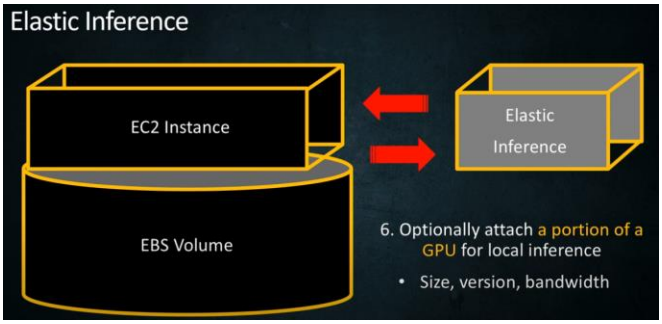
Tool/Feature	Purpose	Key Information Provided	Use Case
CloudTrail Logs	Monitor API calls	• Caller identity • Timestamps • API details	Identify unauthorized API calls to SageMaker resources
Data Event Logs	Monitor data plane operations	Input/output data for training and inference	Verify if unauthorized entities accessed model data
IAM Policies	Manage access control	Permissions granted for SageMaker resources and operations	Identify overly permissive policies, ensure least privilege
VPC Flow Logs	Monitor network traffic	Network traffic to/from SageMaker resources	Identify suspicious IP addresses or communication patterns
Encryption Settings	Ensure data protection	• Encryption status (at rest and in transit) • AWS KMS key configurations	Verify proper data encryption and key management
AWS PrivateLink	Enhance network security	Private connections between VPC and SageMaker	Ensure traffic remains within AWS network

Domain X: Misc

X.1 SageMaker Deep Dive

X.1.1 Fully Managed Notebook Instances with Amazon SageMaker





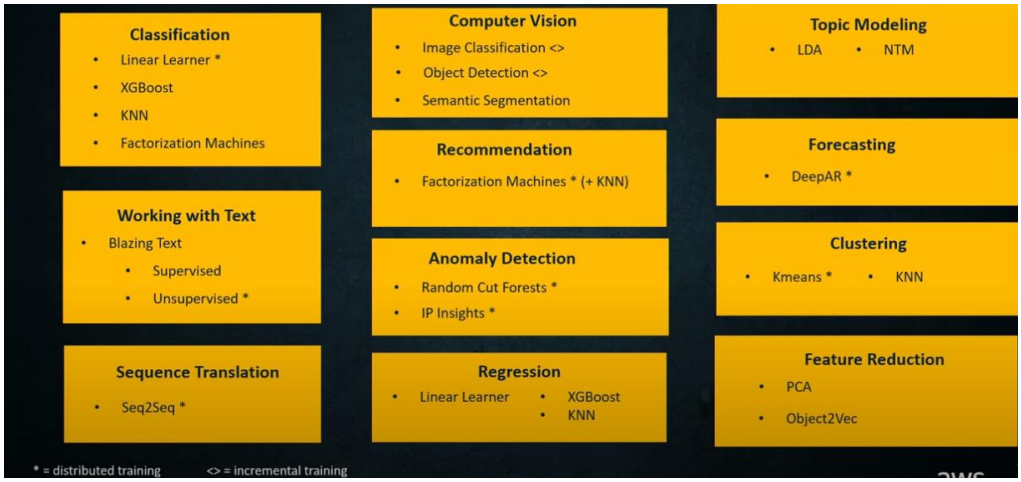
Elastic Inference

Elastic Inference is a service that allows attaching a portion of a GPU to an existing EC2 instance.2 This approach is particularly useful when running inference locally on a notebook instance.2 By selecting an appropriate Elastic Inference configuration based on size, version, and bandwidth, users can accelerate their inference tasks without needing a full GPU.2

Use Cases for Elastic Inference

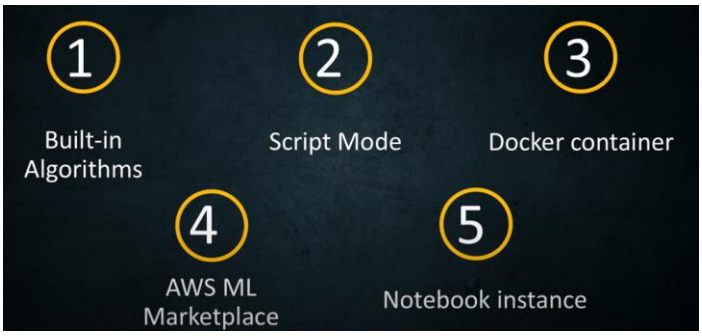
- You need to run inference tasks locally on your notebook instance.
- Your workload benefits from GPU acceleration but doesn't require a full GPU.
- You want to optimize cost by only paying for the portion of GPU resources used.

X.1.2 SageMaker Built-in Algorithms



Task Category	Algorithms	Supervised/UnSupervised
Classification	• Linear Learner (distributed) • XGBoost • KNN • Factorization Machines	Supervised
Regression	• Linear Learner • XGBoost • KNN	Supervised
Computer Vision	• Object Detection (incremental) • Semantic Segmentation	Supervised
Working with Text	• BlazingText	Supervised / Unsupervised
Sequence Translation	• Seq2Seq (distributed)	Supervised
Recommendation	• Factorization Machines (distributed) • KNN	Supervised
Anomaly Detection	• Random Cut Forests (distributed) • IP Insights (distributed)	Unsupervised / Semi-supervised
Topic Modeling	• LDA • NTM	Unsupervised
Forecasting	• DeepAR (distributed)	Supervised
Clustering	• K-means (distributed) • KNN	Unsupervised
Feature Reduction	• PCA • Object2Vec	Unsupervised / Semi-supervised

X.1.3 SageMaker Training types



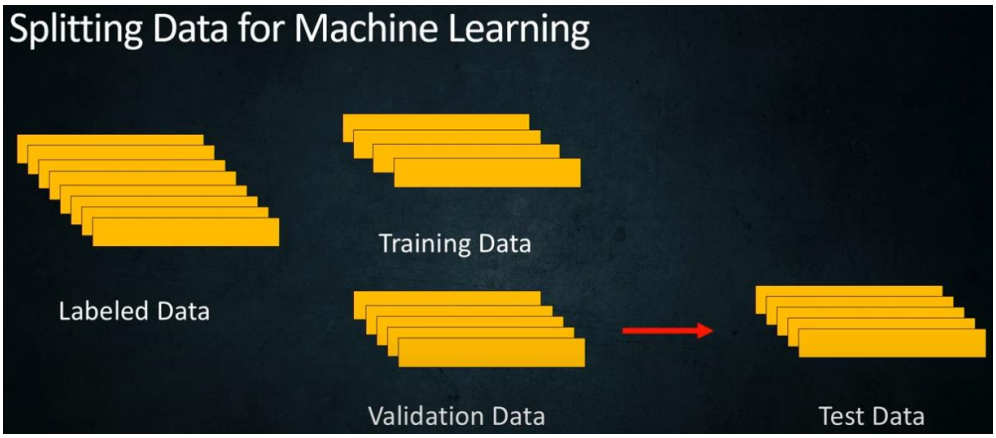
Training Type	Description	When to Use
1. Built-in Algorithms	Pre-configured algorithms provided by Amazon SageMaker, optimized for performance and ease of use	• Working on common ML tasks (e.g., classification, regression) • When you need a quick start without deep ML expertise
2. Script Mode	Custom training scripts using popular ML frameworks (e.g., TensorFlow, PyTorch, Scikit-learn)	• You have existing scripts in popular ML frameworks • For customizing model architecture while leveraging SageMaker's infrastructure
3. Docker Container	Custom Docker containers with your own algorithms or environments	• Need complete control over training env. • Custom or proprietary algorithms • For complex, multi-step training pipelines
4. AWS ML Marketplace	Pre-built algorithms and models from third-party vendors available through the AWS Marketplace	• Need industry-specific or specialized models • When you want to explore alternative solutions without building from scratch
5. Notebook Instance	Interactive development and training using Jupyter notebooks on managed instances	• During the initial stages of model development • When you need an interactive environment for debugging and visualization

Key Considerations:

- Skill Level: Built-in Algorithms and Marketplace for beginners, Script Mode and Containers for more advanced users
- Customization Needs: From low (Built-in) to high (Containers)
- Development Speed: Notebooks for rapid prototyping, Built-in for quick deployment, Containers for complex but reproducible setups
- Scale: Consider moving from Notebooks to other options as your data and model complexity grow.

X.1.4 Train Your ML Models with Amazon SageMaker

Splitting Data for ML



Create Notebook Instance → **Prepare Data** → Train & Validate the Model → Build the Model → Deploy the Model

Applying dummies as one-hot encoding to convert categorical features to numeric.

```
model_data = pd.get_dummies(churn)
model_data = pd.concat(
        [model_data["Churn?_True."],
        model_data.drop(
            ["Churn?_False.",
             "Churn?_True."],
             axis=1)],
        axis=1 )
```

Training data, and subsequent validation data, are stored in diff S3 buckets

```
s3_input_train = TrainingInput(
        s3_data=S3_train_location,
        content_type="csv" )
s3_input_validation = TrainingInput(
        s3_data= S3_validate_location,
        content_type="csv" )
```

IAM Role
# of instances
Type of instance

Training data, and subsequent validation data, are stored in diff S3 buckets

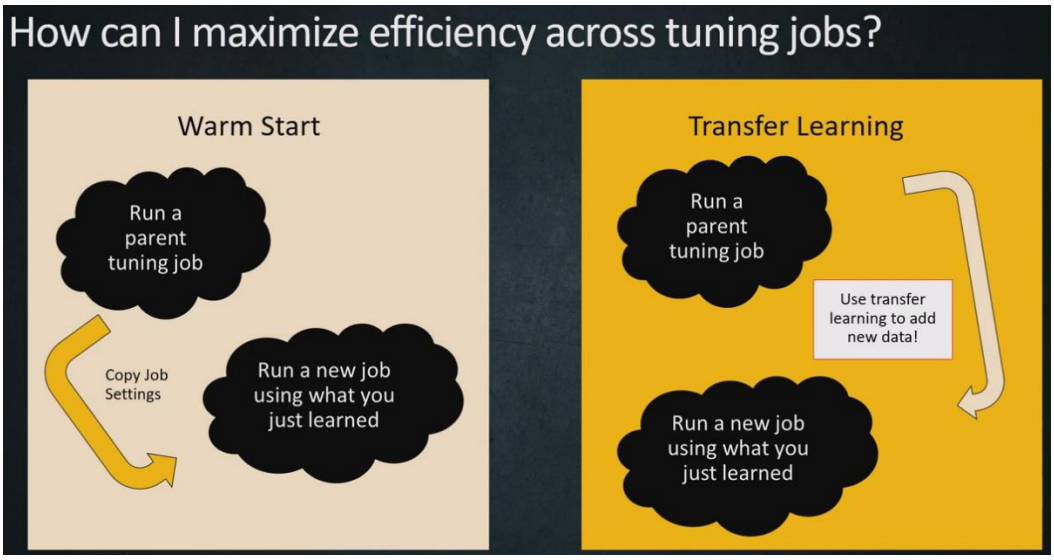
```
sess = sagemaker.Session()

xgb = sagemaker.estimator.Estimator
    (container,
     role,
     train_instance_count=1,
     train_instance_type='ml.m4.xlarge',
     output_path=s3_output_location,
     sagemaker_session=sess)

xgb.set_hyperparameters(
                max_depth=5,
                eta=0.2,
                gamma=4,
                min_child_weight=6,
                subsample=0.8,
                silent=0,
                objective='binary:logistic',
                num_round=100)

xgb.fit({
        'train': s3_input_train,
        'validation': s3_input_validation})
```

```
xgb_predict = xgb.deploy(
    initial_instance_count=1,
    instance_type='ml.t2.medium')
xgb_predict.content_type = 'text/csv'
xgb_predict.serializer = csv_serializer
```

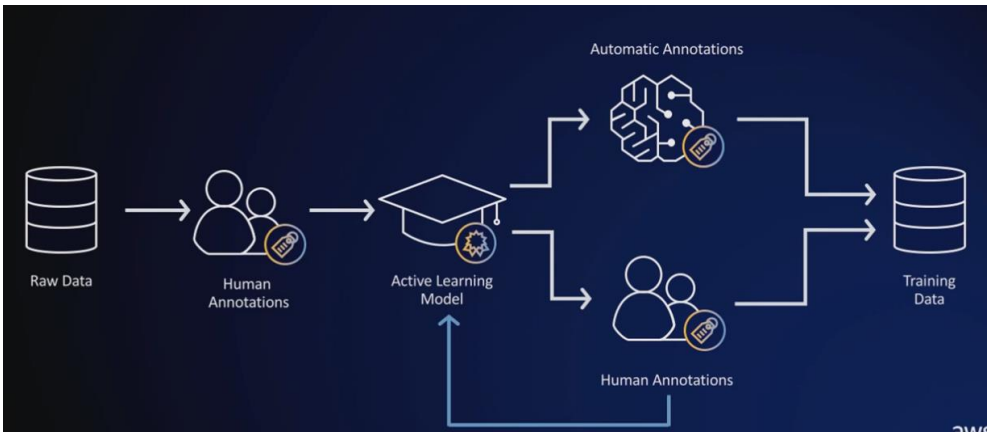
Deploy also involves creating an API endpoint

X.1.5 Tuning Your ML Models with Amazon SageMaker

Maximizing Efficiency across tuning jobs



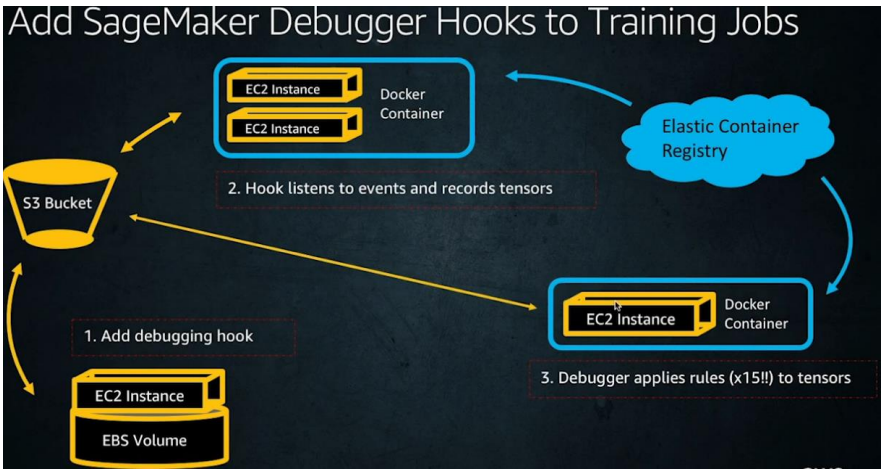
X.1.6 Tuning Your ML Models with Amazon SageMaker



How to automate

Put a check to see if Accuracy falls below a % (e.g. > 80%), invoke Human in the loop

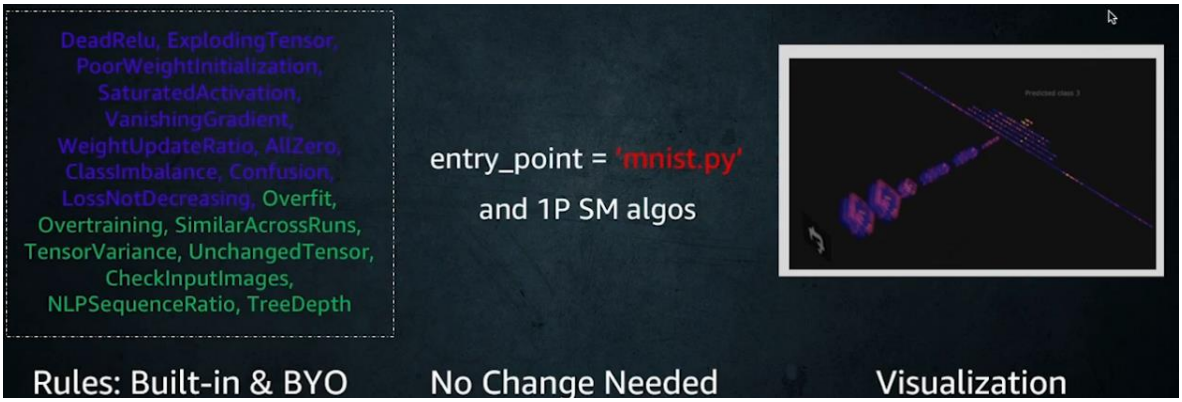
X.1.6 Add Debugger to Training Jobs in Amazon SageMaker



How it works

1. Add debugging hook:
    - An EC2 instance with an attached EBS volume is used to initiate the process.
    - The debugging hook is added to the training job configuration.
2. Hook listens to events and records tensors:
    - Docker containers running on EC2 instances are used for the training job.
    - The hook listens for specific events during the training process and records tensor data.
3. Debugger applies rules to tensors:
    - Another EC2 instance with a Docker container is used for debugging.
    - The debugger applies predefined rules (mentioned as "x15!!" in the image) to the recorded tensor data.

Benefits of debugger



1. Comprehensive Built-in Rules/Algorithms: The debugger offers a wide range of built-in rules to detect common issues in machine learning models, such as:
    - DeadRelu, ExplodingTensor, PoorWeightInitialization
    - SaturatedActivation, VanishingGradient
    - WeightUpdateRatio, AllZero, ClassImbalance
    - Confusion, LossNotDecreasing, Overfit
    - Overtraining, SimilarAcrossRuns
    - TensorVariance, UnchangedTensor
    - CheckInputImages, NLPSequenceRatio, TreeDepth

2. Customizable (BYO - Bring Your Own): Users can create and add their own custom debugging rules.
3. Easy Integration: The entry point is 'mnist.py' and it works with SageMaker's built-in algorithms (1P SM algos), suggesting easy integration with existing SageMaker workflows.
4. No Code Changes Required: The "No Change Needed" text implies that adding debugging capabilities doesn't require modifying the existing model code.
5. **Visualization**: The debugger provides visualization capabilities, as indicated by the image on the right, which appears to show a tensor or weight distribution.
6. **Real-time Monitoring**: The variety of rules suggests that the debugger can monitor various aspects of model training in real-time, helping to identify issues as they occur.

X.1.7 Deployment using SageMaker

Deployment Strategy	Description	When to Use
Blue/Green Deployment with Linear Traffic Shifting	Gradually shift traffic from the old version (blue) to the new version (green) over time	• When you need fine-grained control over the traffic shift • For critical applications requiring minimal risk • When you have the resources to run two full environments simultaneously
Canary Deployment	Release a new version to a small subset of users before rolling it out to the entire infrastructure	• When you want to test in production with real users • For early detection of issues before full deployment • When you have a diverse user base
A/B Testing	Run two versions simultaneously and compare their performance based on metrics	• When you want to test specific features or changes • When you need to optimize based on user behavior or business metrics
Rolling Deployment	Gradually replace instances of the old version with the new version	• When you have limited resources and can't run two full environments • For applications that can handle mixed versions • When you need to minimize downtime