Disclaimer: **These slides are copyrighted and strictly for personal use only**

- This document is reserved for people enrolled into the [AWS Certified Machine Learning Specialty Course by Sundog Education.]
- **Please do not share this document**, it is intended for personal use and exam preparation only, thank you.
- If you've obtained these slides for free on a website that is not the course's website, please reach out to piracy@datacumulus.com. Thanks!
- **Best of luck for the exam and happy learning!**

DataCumulus Sundog™ Education

Table of Contents

AWS Certified Machine Learning Specialty Course

MLS-C01

DataCumulus Sundog™ Education

Data Engineering

Moving, Storing and Processing data in AWS

Amazon S3 Overview

- Amazon S3 allows people to store objects (files) in “buckets” (directories)
- Buckets must have a **globally unique name**
- Objects (files) have a Key. The key is the **FULL** path:
  - <my_bucket>/**my_file.txt**
  - <my_bucket>/**my_folder1/another_folder/my_file.txt**
- This will be interesting when we look at **partitioning**
- Max object size is 5TB
- Object Tags (key / value pair – up to 10) – useful for security / lifecycle

Amazon S3 for Machine Learning

- Backbone for many AWS ML services (example: SageMaker)
- Create a "Data Lake"
  - Infinite size, no provisioning
  - 99.999999999% durability
  - Decoupling of storage (S3) to compute (EC2, Amazon Athena, Amazon Redshift Spectrum, Amazon Rekognition, and AWS Glue)
- Centralized Architecture
- Object storage => supports any file format
- Common formats for ML: CSV, JSON, Parquet, ORC, Avro, Protobuf

DataCumulus Sundog™ Education

Amazon S3 Data Partitioning

- Pattern for speeding up range queries (ex: AWS Athena)
- By Date: s3://bucket/my-data-set/year/month/day/hour/data_00.csv
- By Product: s3://bucket/my-data-set/product-id/data_32.csv
- You can define whatever partitioning strategy you like!
- Data partitioning will be handled by some tools we use (e.g. AWS Glue)

S3 Storage Classes

- Amazon S3 Standard - General Purpose
- Amazon S3 Standard-Infrequent Access (IA)
- Amazon S3 One Zone-Infrequent Access
- Amazon S3 Glacier Instant Retrieval
- Amazon S3 Glacier Flexible Retrieval
- Amazon S3 Glacier Deep Archive
- Amazon S3 Intelligent Tiering

- Can move between classes manually or using S3 Lifecycle configurations

S3 Durability and Availability

- Durability:
  - High durability (99.999999999%, 11 9's) of objects across multiple AZ
  - If you store 10,000,000 objects with Amazon S3, you can on average expect to incur a loss of a single object once every 10,000 years
  - Same for all storage classes
- Availability:
  - Measures how readily available a service is
  - Varies depending on storage class
  - Example: S3 standard has 99.99% availability = not available 53 minutes a year

S3 Standard – General Purpose

- 99.99% Availability
- Used for frequently accessed data
- Low latency and high throughput
- Sustain 2 concurrent facility failures

- Use Cases: Big Data analytics, mobile & gaming applications, content distribution…

S3 Storage Classes – Infrequent Access

- For data that is less frequently accessed, but requires rapid access when needed
- Lower cost than S3 Standard

- **Amazon S3 Standard-Infrequent Access (S3 Standard-IA)**
  - 99.9% Availability
  - Use cases: Disaster Recovery, backups

- **Amazon S3 One Zone-Infrequent Access (S3 One Zone-IA)**
  - High durability (99.999999999%) in a single AZ; data lost when AZ is destroyed
  - 99.5% Availability
  - Use Cases: Storing secondary backup copies of on-premise data, or data you can recreate

Amazon S3 Glacier Storage Classes

- Low-cost object storage meant for archiving / backup
- Pricing: price for storage + object retrieval cost

- **Amazon S3 Glacier Instant Retrieval**
  - Millisecond retrieval, great for data accessed once a quarter
  - Minimum storage duration of 90 days
- **Amazon S3 Glacier Flexible Retrieval** (formerly Amazon S3 Glacier):
  - Expedited (1 to 5 minutes), Standard (3 to 5 hours), Bulk (5 to 12 hours) – free
  - Minimum storage duration of 90 days
- **Amazon S3 Glacier Deep Archive – for long term storage:**
  - Standard (12 hours), Bulk (48 hours)
  - Minimum storage duration of 180 days

DataCumulus Sundog Education

S3 Intelligent-Tiering

- Small monthly monitoring and auto-tiering fee
- Moves objects automatically between Access Tiers based on usage
- There are no retrieval charges in S3 Intelligent-Tiering

- *Frequent Access tier (automatic):* default tier
- *Infrequent Access tier (automatic)*: objects not accessed for 30 days
- *Archive Instant Access tier (automatic):* objects not accessed for 90 days
- *Archive Access tier (optional):* configurable from 90 days to 700+ days
- *Deep Archive Access tier (optional):* config. from 180 days to 700+ days

S3 Storage Classes Comparison

	Standard	Intelligent-Tiering	Standard-IA	One Zone-IA	Glacier Instant Retrieval	Glacier Flexible Retrieval	Glacier Deep Archive
Durability	99.999999999% == (11 9's)						
Availability	99.99%	99.9%	99.9%	99.5%	99.9%	99.99%	99.99%
Availability SLA	99.9%	99%	99%	99%	99%	99.9%	99.9%
Availability Zones	>= 3	>= 3	>= 3	1	>= 3	>= 3	>= 3
Min. Storage Duration Charge	None	None	30 Days	30 Days	90 Days	90 Days	180 Days
Min. Billable Object Size	None	None	128 KB	128 KB	128 KB	40 KB	40 KB
Retrieval Fee	None	None	Per GB retrieved	Per GB retrieved	Per GB retrieved	Per GB retrieved	Per GB retrieved

https://aws.amazon.com/s3/storage-classes/

S3 Storage Classes – Price Comparison Example: us-east-1

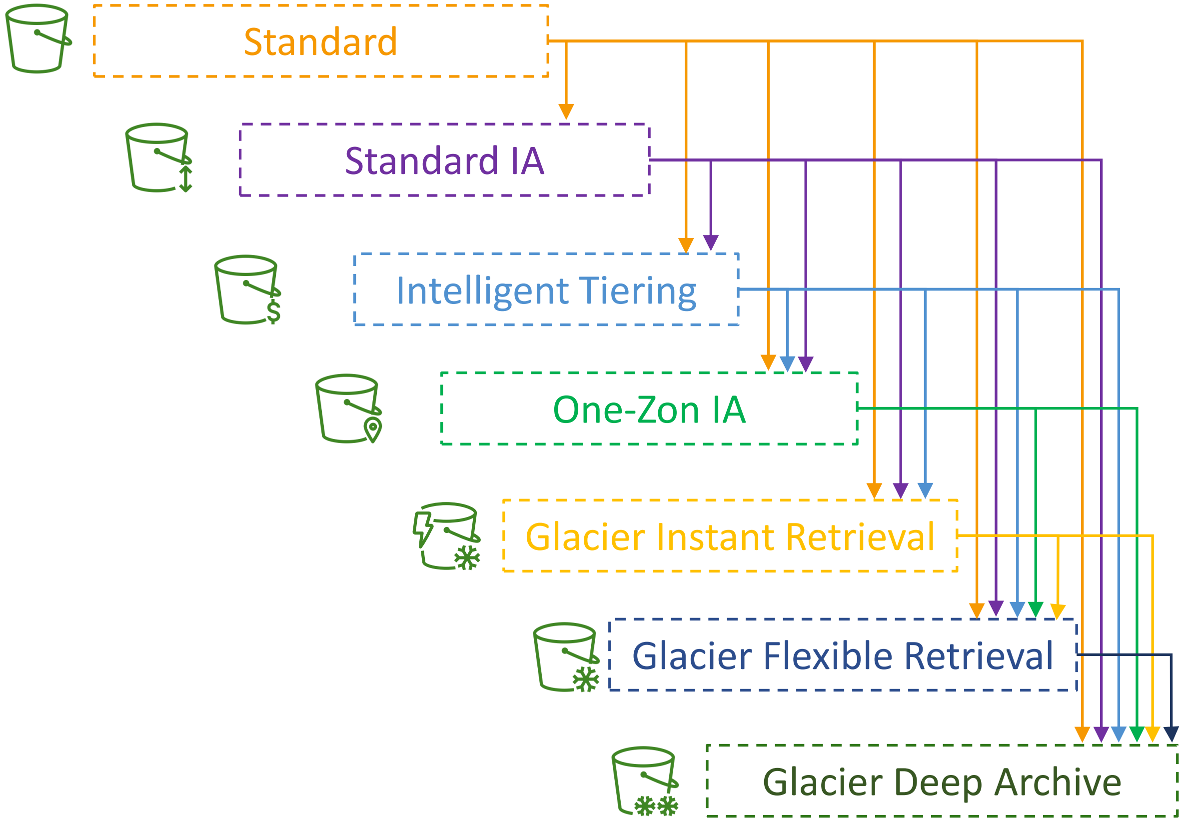
	Standard	Intelligent-Tiering	Standard-IA	One Zone-IA	Glacier Instant Retrieval	Glacier Flexible Retrieval	Glacier Deep Archive
Storage Cost (per GB per month)	$0.023	$0.0025 - $0.023	$0.0125	$0.01	$0.004	$0.0036	$0.00099
Retrieval Cost (per 1000 request)	**GET:** $0.0004 **POST:** $0.005	**GET:** $0.0004 **POST:** $0.005	**GET:** $0.001 **POST:** $0.01	**GET:** $0.001 **POST:** $0.01	**GET:** $0.01 **POST:** $0.02	**GET:** $0.0004 **POST:** $0.03 **Expedited:** $10 **Standard:** $0.05 **Bulk:** free	**GET:** $0.0004 **POST:** $0.05 **Standard:** $0.10 **Bulk:** $0.025
Retrieval Time	Instantaneous					**Expedited** (1 – 5 mins) **Standard** (3 – 5 hours) Bulk (5 – 12 hours)	**Standard** (12 hours) **Bulk** (48 hours)
Monitoring Cost (pet 1000 objects)		$0.0025					

https://aws.amazon.com/s3/pricing/

DataCumulus Sundog Education

Amazon S3 – Moving between Storage Classes

- You can transition objects between storage classes
- For infrequently accessed object, move them to **Standard IA**
- For archive objects that you don't need fast access to, move them to **Glacier or Glacier Deep Archive**
- Moving objects can be automated using a **Lifecycle Rules**



Amazon S3 – Lifecycle Rules

- **Transition Actions –** configure objects to transition to another storage class
  - Move objects to Standard IA class 60 days after creation
  - Move to Glacier for archiving after 6 months

- **Expiration actions –** configure objects to expire (delete) after some time
  - Access log files can be set to delete after a 365 days
  - **Can be used to delete old versions of files (if versioning is enabled)**
  - Can be used to delete incomplete Multi-Part uploads

- Rules can be created for a certain prefix (example: *s3://mybucket/mp3/*)*
- Rules can be created for certain objects Tags (example: *Department: Finance*)

Amazon S3 – Lifecycle Rules (Scenario 1)

- **Your application on EC2 creates images thumbnails after profile photos are uploaded to Amazon S3. These thumbnails can be easily recreated, and only need to be kept for 60 days. The source images should be able to be immediately retrieved for these 60 days, and afterwards, the user can wait up to 6 hours. How would you design this?**

- S3 source images can be on **Standard**, with a lifecycle configuration to transition them to **Glacier** after 60 days
- S3 thumbnails can be on **One-Zone IA**, with a lifecycle configuration to expire them (delete them) after 60 days

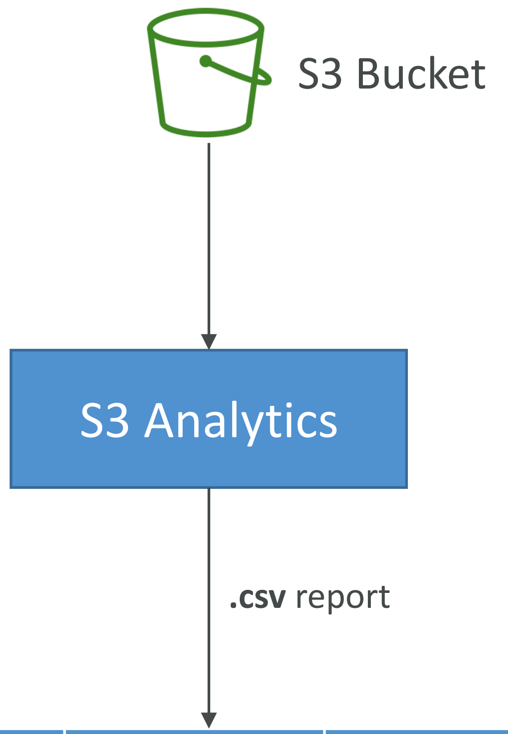
Amazon S3 – Lifecycle Rules (Scenario 2)

- **A rule in your company states that you should be able to recover your deleted S3 objects immediately for 30 days, although this may happen rarely. After this time, and for up to 365 days, deleted objects should be recoverable within 48 hours.**

- **Enable S3 Versioning** in order to have object versions, so that "deleted objects" are in fact hidden by a "delete marker" and can be recovered
- Transition the "noncurrent versions" of the object to **Standard IA**
- Transition afterwards the "noncurrent versions" to **Glacier Deep Archive**

Amazon S3 Analytics – Storage Class Analysis

- Help you decide when to transition objects to the right storage class
- Recommendations for **Standard** and **Standard IA**
  - Does NOT work for One-Zone IA or Glacier
- Report is updated daily
- 24 to 48 hours to start seeing data analysis
- Good first step to put together Lifecycle Rules (or improve them)!



Date	StorageClass	ObjectAge
8/22/2022	STANDARD	000-014
8/25/2022	STANDARD	030-044
9/6/2022	STANDARD	120-149

Amazon S3 – Security

- **User-Based**
  - **IAM Policies –** which API calls should be allowed for a specific user from IAM
- **Resource-Based**
  - **Bucket Policies –** bucket wide rules from the S3 console - allows cross account
  - **Object Access Control List (ACL) –** finer grain (can be disabled)
  - **Bucket Access Control List (ACL) –** less common (can be disabled)
- **Note:** an IAM principal can access an S3 object if
  - The user IAM permissions ALLOW it OR the resource policy ALLOWS it
  - AND there's no explicit DENY
- **Encryption:** encrypt objects in Amazon S3 using encryption keys

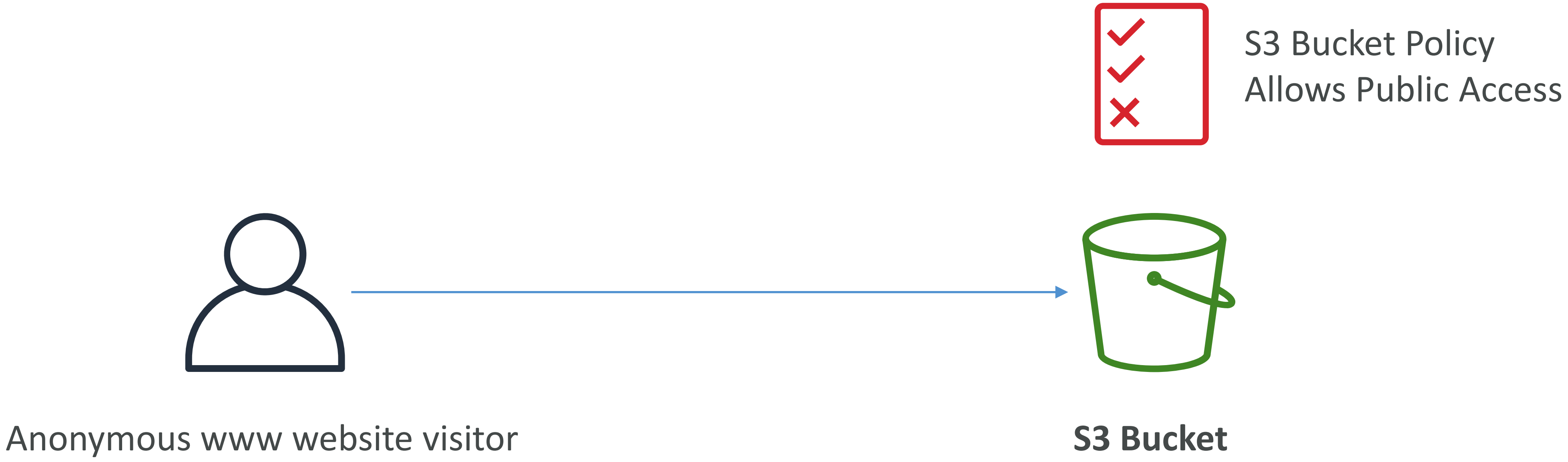
S3 Bucket Policies

- JSON based policies
  - Resources: buckets and objects
  - Effect: Allow / Deny
  - Actions: Set of API to Allow or Deny
  - Principal: The account or user to apply the policy to

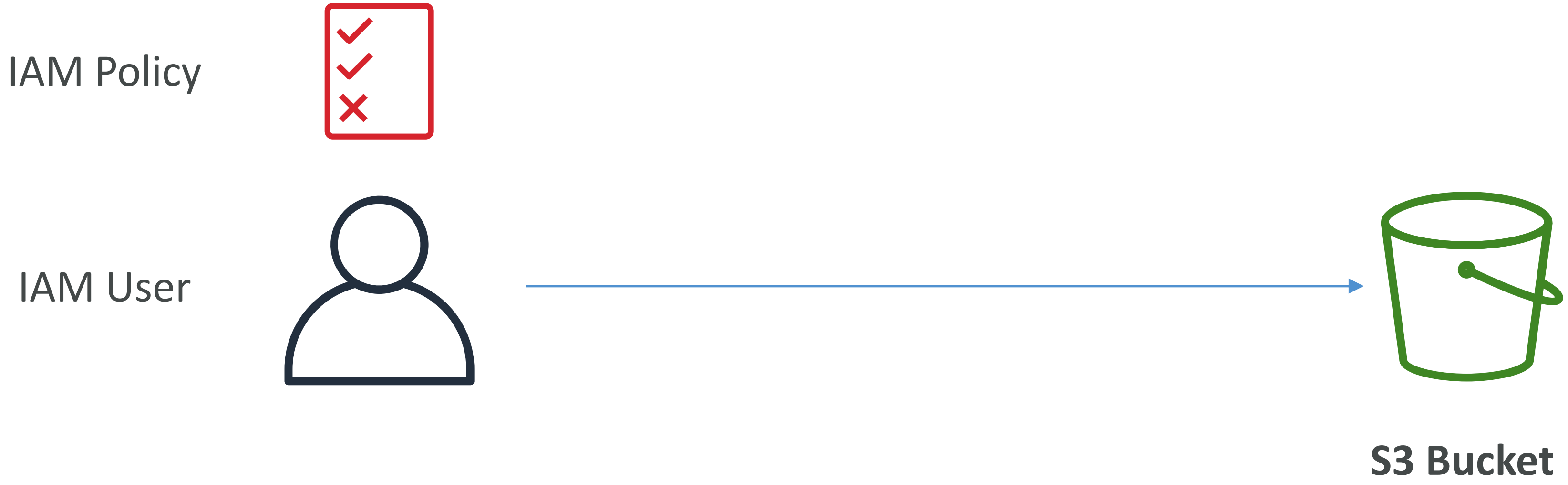
- Use S3 bucket for policy to:
  - Grant public access to the bucket
  - Force objects to be encrypted at upload
  - Grant access to another account (Cross Account)

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Sid": "PublicRead",
            "Effect": "Allow",
            "Principal": "*",
            "Action": [
                "s3:GetObject"
            ],
            "Resource": [
                "arn:aws:s3:::examplebucket/*"
            ]
        }
    ]
}
```

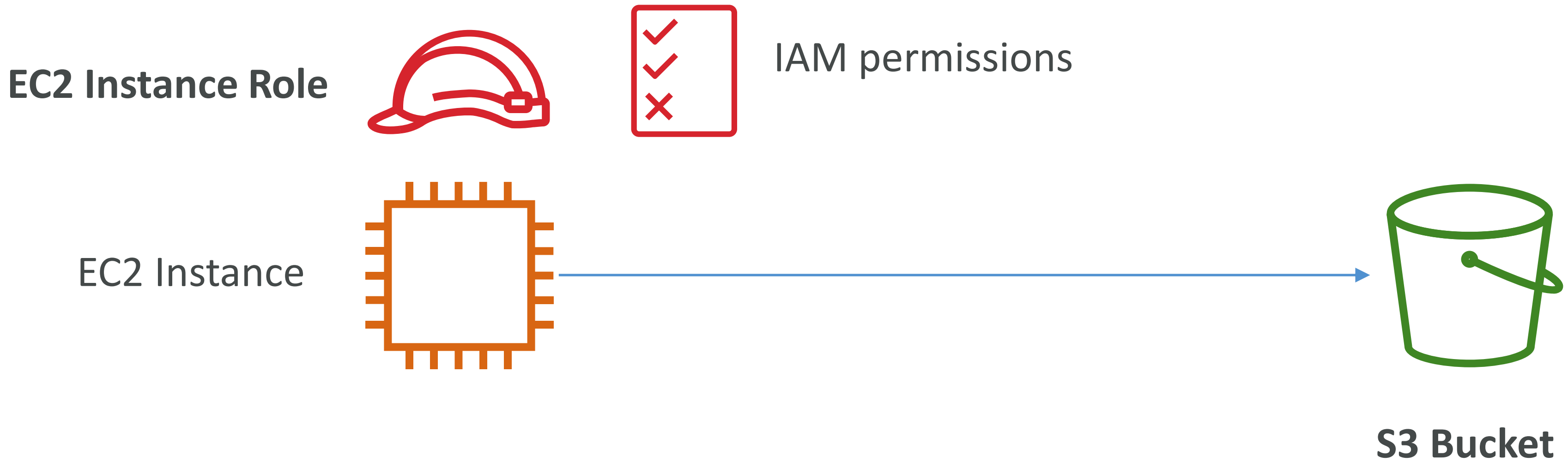
Example: Public Access - Use Bucket Policy



Example: User Access to S3 – IAM permissions



Example: EC2 instance access - Use IAM Roles



Advanced: Cross-Account Access – Use Bucket Policy

Bucket settings for Block Public Access

Block *all* public access
On

- Block public access to buckets and objects granted through *new* access control lists (ACLs)
  On
- Block public access to buckets and objects granted through *any* access control lists (ACLs)
  On
- Block public access to buckets and objects granted through *new* public bucket or access point policies
  On
- Block public and cross-account access to buckets and objects through *any* public bucket or access point policies
  On

- **These settings were created to prevent company data leaks**
- If you know your bucket should never be public, leave these on
- Can be set at the account level

Amazon S3 – Object Encryption

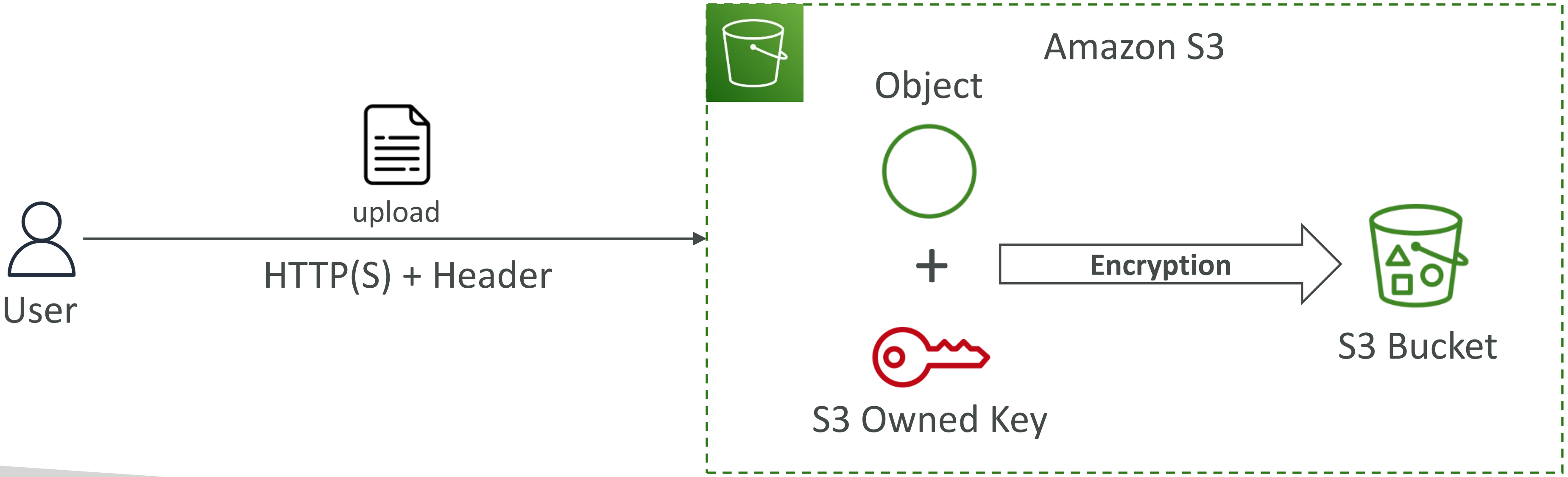
- You can encrypt objects in S3 buckets using one of 4 methods

- **Server-Side Encryption (SSE)**
  - **Server-Side Encryption with Amazon S3-Managed Keys (SSE-S3) – <u>Enabled by Default</u>**
    - Encrypts S3 objects using keys handled, managed, and owned by AWS
  - **Server-Side Encryption with KMS Keys stored in AWS KMS (SSE-KMS)**
    - Leverage AWS Key Management Service (AWS KMS) to manage encryption keys
  - **Server-Side Encryption with Customer-Provided Keys (SSE-C)**
    - When you want to manage your own encryption keys
- **Client-Side Encryption**

- It's important to understand which ones are for which situation for the exam

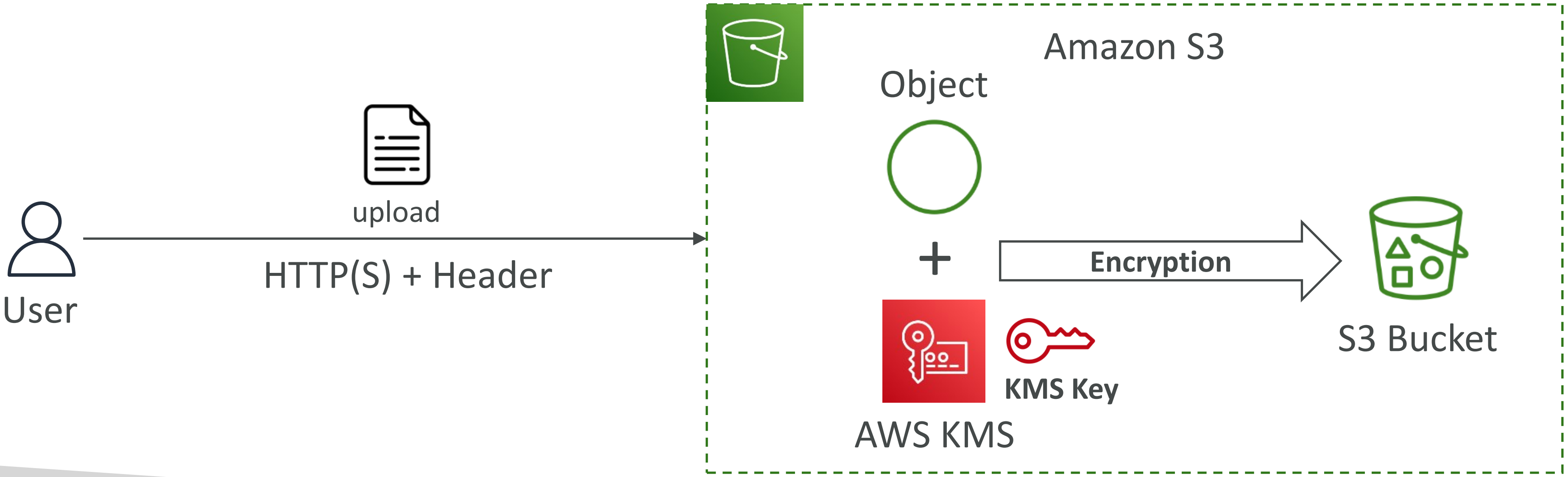
Amazon S3 Encryption – SSE-S3

- Encryption using keys handled, managed, and owned by AWS
- Object is encrypted server-side
- Encryption type is **AES-256**
- Must set header **"x-amz-server-side-encryption": "AES256"**
- **Enabled by default for new buckets & new objects**



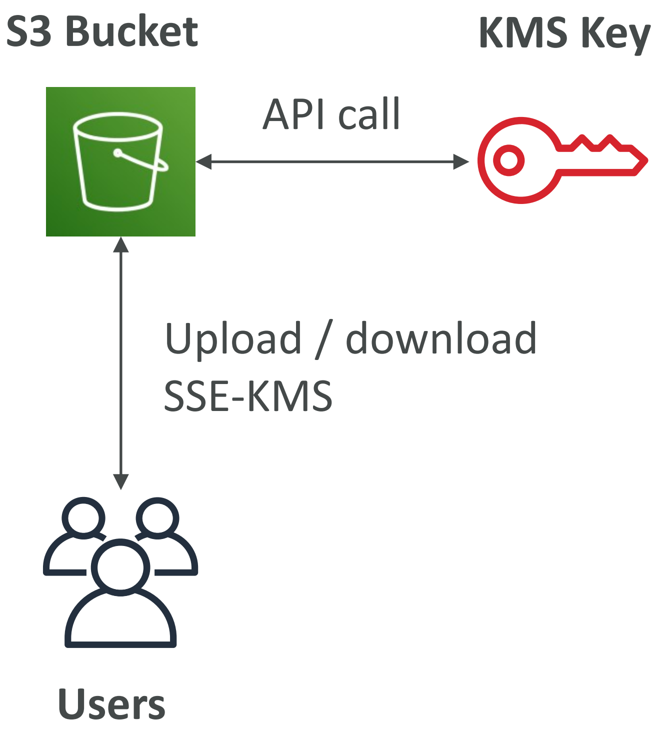
Amazon S3 Encryption – SSE-KMS

- Encryption using keys handled and managed by AWS KMS (Key Management Service)
- KMS advantages: user control + audit key usage using CloudTrail
- Object is encrypted server side
- Must set header **"x-amz-server-side-encryption": "aws:kms"**



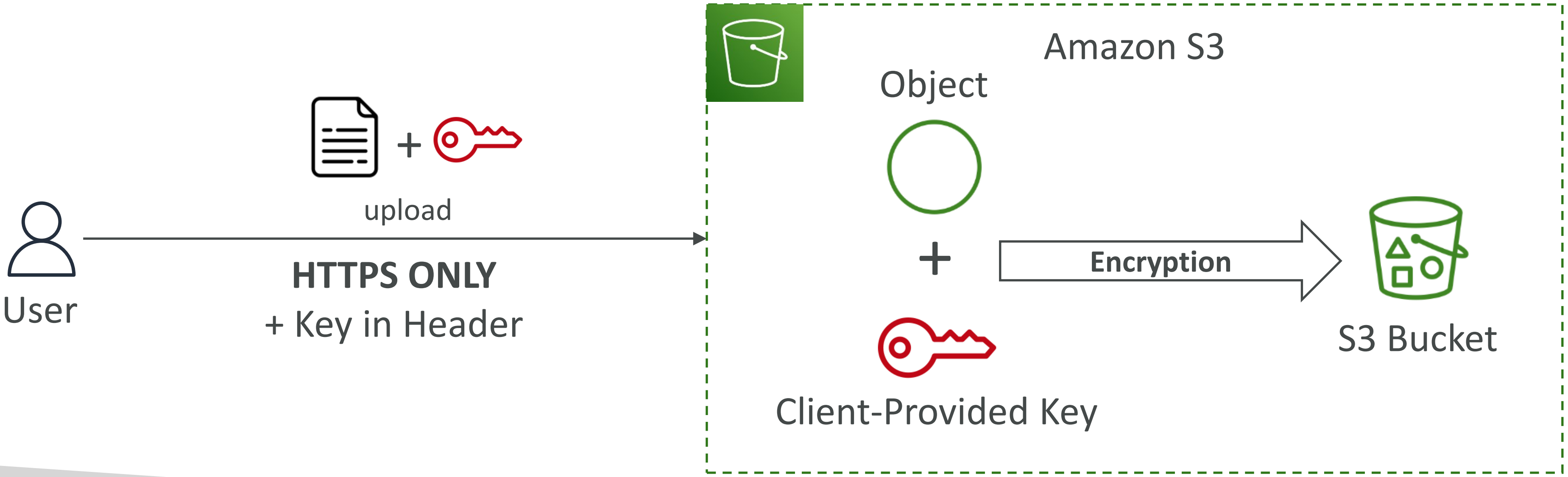
SSE-KMS Limitation

- If you use SSE-KMS, you may be impacted by the KMS limits
- When you upload, it calls the **GenerateDataKey** KMS API
- When you download, it calls the **Decrypt** KMS API
- Count towards the KMS quota per second (5500, 10000, 30000 req/s based on region)
- You can request a quota increase using the Service Quotas Console



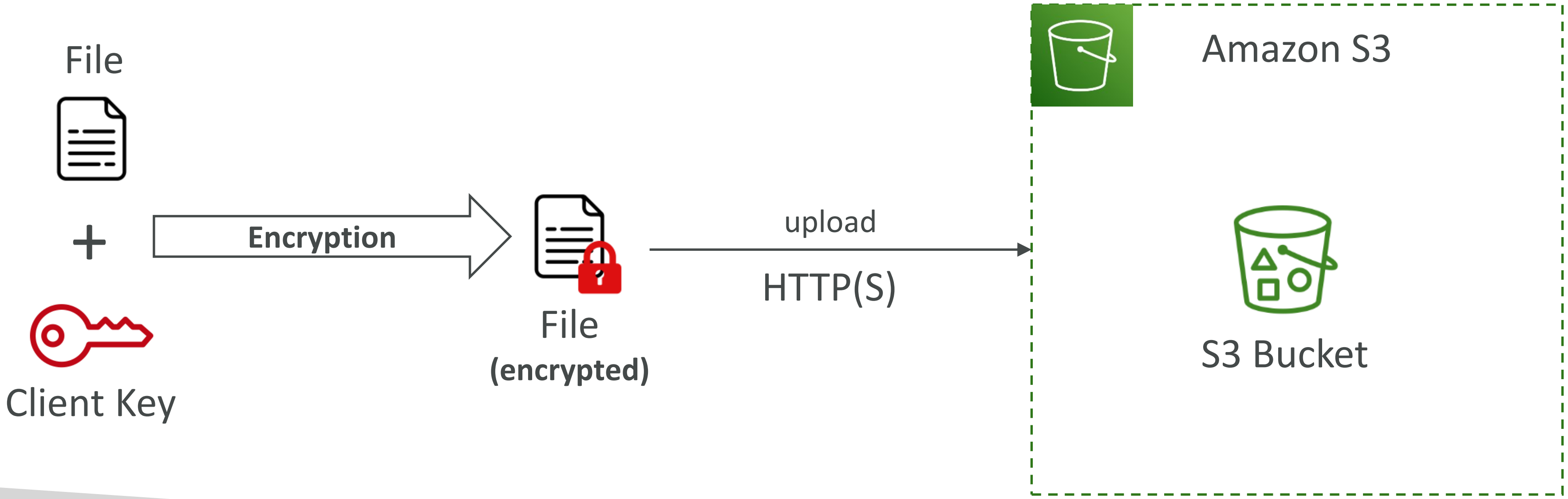
Amazon S3 Encryption – SSE-C

- Server-Side Encryption using keys fully managed by the customer outside of AWS
- Amazon S3 does **NOT** store the encryption key you provide
- **HTTPS must be used**
- Encryption key must provided in HTTP headers, for every HTTP request made



Amazon S3 Encryption – Client-Side Encryption

- Use client libraries such as **Amazon S3 Client-Side Encryption Library**
- Clients must encrypt data themselves before sending to Amazon S3
- Clients must decrypt data themselves when retrieving from Amazon S3
- Customer fully manages the keys and encryption cycle



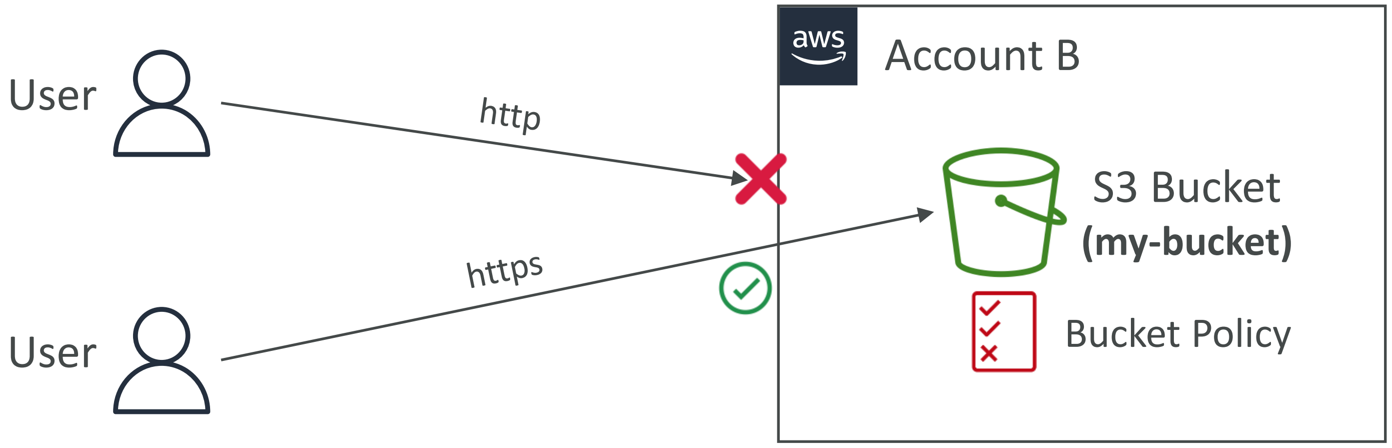
Amazon S3 – Encryption in transit (SSL/TLS)

- Encryption in flight is also called SSL/TLS

- Amazon S3 exposes two endpoints:
  - **HTTP Endpoint** – non encrypted
  - **HTTPS Endpoint** – encryption in flight

- **HTTPS is recommended**
- **HTTPS is mandatory for SSE-C**
- Most clients would use the HTTPS endpoint by default

Amazon S3 – Force Encryption in Transit aws:SecureTransport




```json
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Effect": "Deny",
            "Principal": "*",
            "Action": "s3:GetObject",
            "Resource": "arn:aws:s3:::my-bucket/*",
            "Condition": {
                "Bool": {
                    "aws:SecureTransport": "false"
                }
            }
        }
    ]
}
```

Amazon S3 – Default Encryption vs. Bucket Policies

- **SSE-S3 encryption is automatically applied to new objects stored in S3 bucket**
- Optionally, you can "force encryption" using a bucket policy and refuse any API call to PUT an S3 object without encryption headers (SSE-KMS or SSE-C)

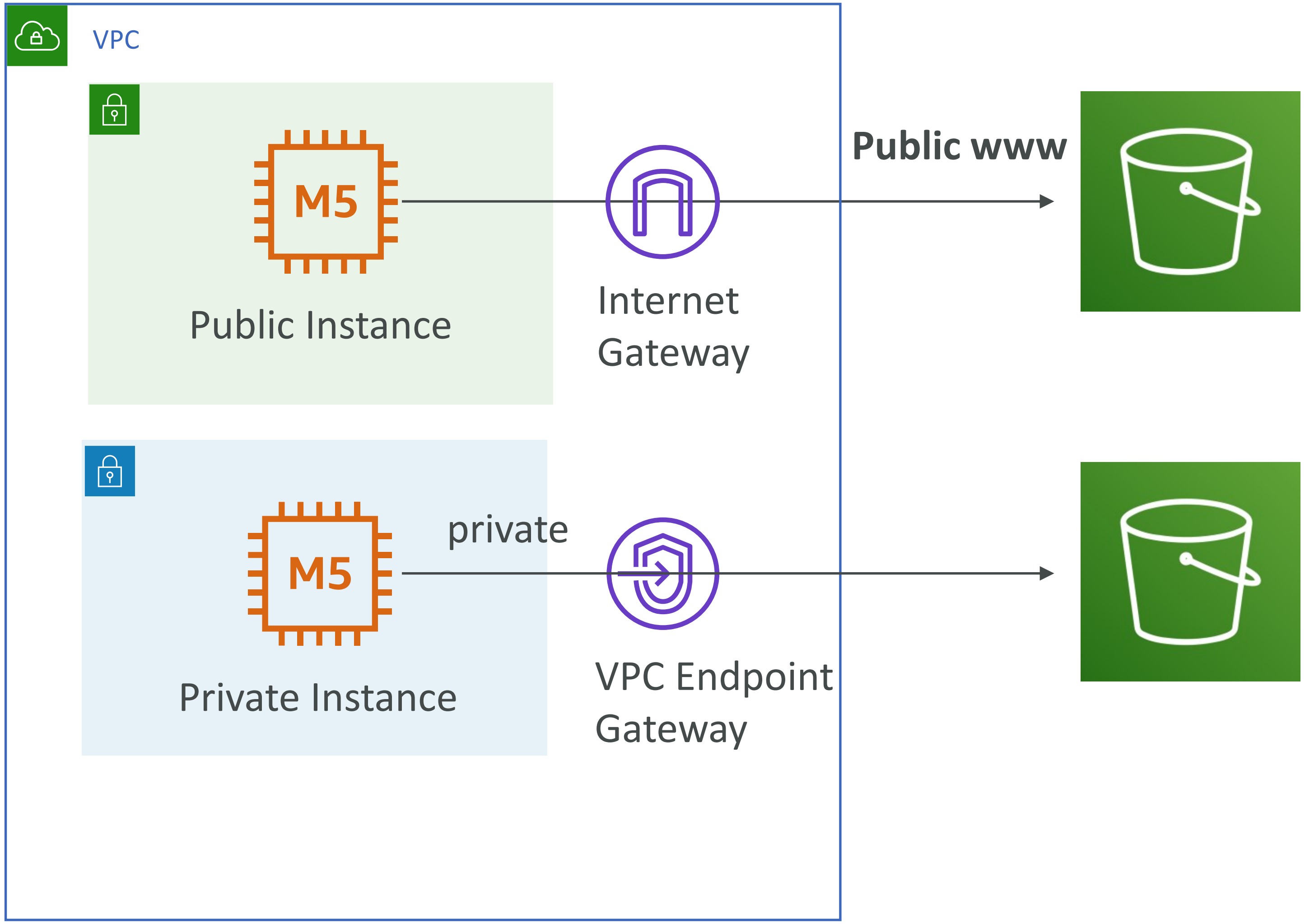
```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Effect": "Deny",
            "Action": "s3:PutObject",
            "Principal": "*",
            "Resource": "arn:aws:s3:::my-bucket/*",
            "Condition": {
                "StringNotEquals": {
                    "s3:x-amz-server-side-encryption": "aws:kms"
                }
            }
        }
    ]
}
```

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Effect": "Deny",
            "Action": "s3:PutObject",
            "Principal": "*",
            "Resource": "arn:aws:s3:::my-bucket/*",
            "Condition": {
                "Null": {
                    "s3:x-amz-server-side-encryption-customer-algorithm": "true"
                }
            }
        }
    ]
}
```

- **<u>Note:</u> Bucket Policies are evaluated before "Default Encryption"**

DataCumulus Sundog™ Education

VPC Endpoint Gateway for S3



S3 Bucket
Bucket policy by **AWS:SourceIP (public IP)**

S3 Bucket
Bucket policy by
AWS:SourceVpce
(one or few endpoints)

OR

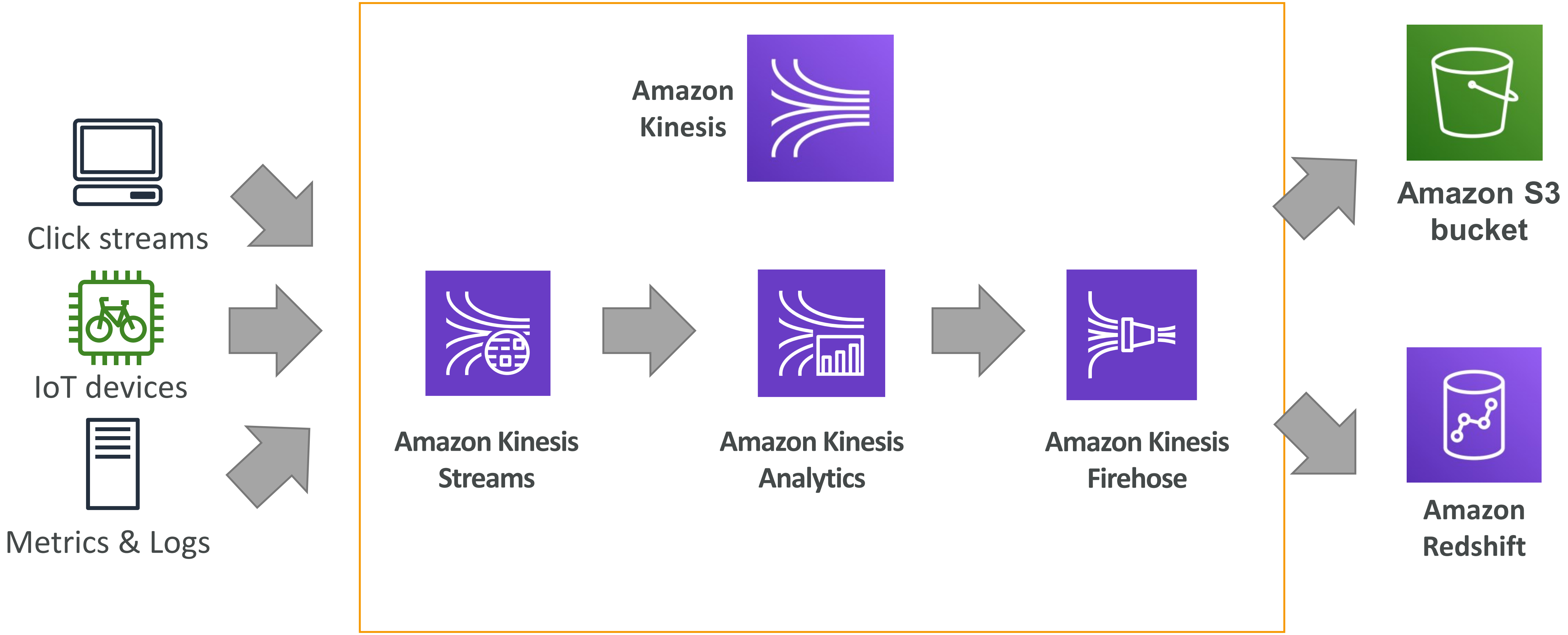
AWS:SourceVpc
(encompass all possible VPC endpoints)

AWS Kinesis Overview

- **Kinesis** is a managed alternative to Apache Kafka
- Great for application logs, metrics, IoT, clickstreams
- Great for "real-time" big data
- Great for streaming processing frameworks (Spark, NiFi, etc…)
- Data is automatically replicated synchronously to 3 AZ

- **Kinesis Streams:** low latency streaming ingest at scale
- **Kinesis Analytics:** perform real-time analytics on streams using SQL
- **Kinesis Firehose:** load streams into S3, Redshift, ElasticSearch & Splunk
- **Kinesis Video Streams:** meant for streaming video in real-time

Kinesis


Click streams
IoT devices
Metrics & Logs
Amazon Kinesis
Amazon Kinesis Streams
Amazon Kinesis Analytics
Amazon Kinesis Firehose
Amazon S3 bucket
Amazon Redshift

Kinesis Streams Overview

- Streams are divided in ordered Shards / Partitions

- Data retention is 24 hours by default, can go up to 365 days
- Ability to reprocess / replay data
- Multiple applications can consume the same stream
- Once data is inserted in Kinesis, it can't be deleted (immutability)
- Records can be up to 1MB in size

DataCumulus Sundog™ Education

Kinesis Data Streams – Capacity Modes

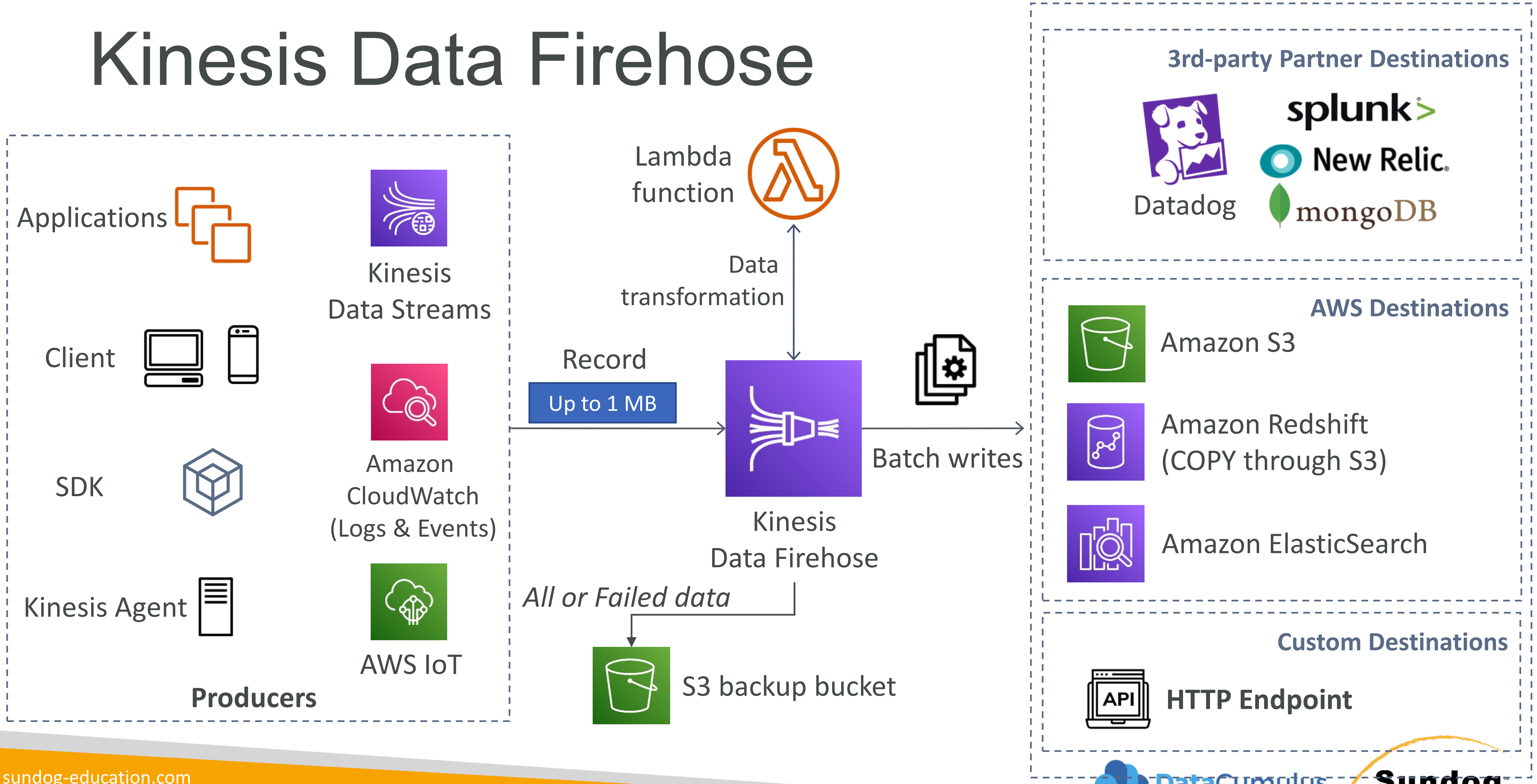
- **Provisioned mode:**
  - You choose the number of shards provisioned, scale manually or using API
  - Each shard gets 1MB/s in (or 1000 records per second)
  - Each shard gets 2MB/s out (classic or enhanced fan-out consumer)
  - You pay per shard provisioned per hour

- **On-demand mode:**
  - No need to provision or manage the capacity
  - Default capacity provisioned (4 MB/s in or 4000 records per second)
  - Scales automatically based on observed throughput peak during the last 30 days
  - Pay per stream per hour & data in/out per GB

Kinesis Data Streams Limits to know

- Producer:
  - 1MB/s or 1000 messages/s at write PER SHARD
  - "ProvisionedThroughputException" otherwise
- Consumer Classic:
  - 2MB/s at read PER SHARD across all consumers
  - 5 API calls per second PER SHARD across all consumers
- Data Retention:
  - 24 hours data retention by default
  - Can be extended to 365 days

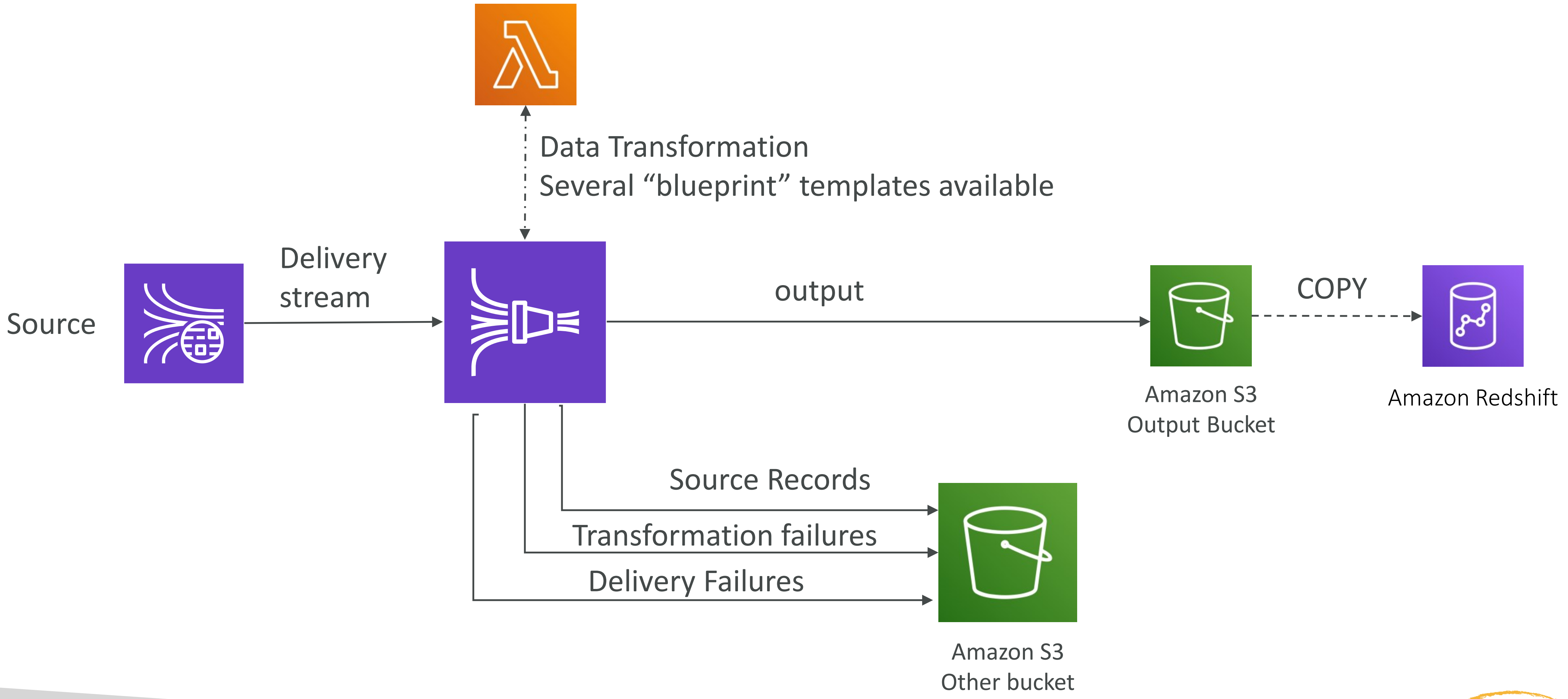
Kinesis Data Firehose



Kinesis Data Firehose

- Fully Managed Service, no administration
- **Near Real Time (Buffer based on time and size, optionally can be disabled)**
- **Data Ingestion into Redshift / Amazon S3 / ElasticSearch / Splunk**
- **Automatic scaling**
- Supports many data formats
- **Data Conversions from CSV / JSON to Parquet / ORC (only for S3)**
- Data Transformation through AWS Lambda (ex: CSV => JSON)
- Supports **compression** when target is Amazon S3 (GZIP, ZIP, and SNAPPY)
- Pay for the amount of data going through Firehose

Kinesis Data Firehose Delivery Diagram



DataCumulus Sundog™ Education

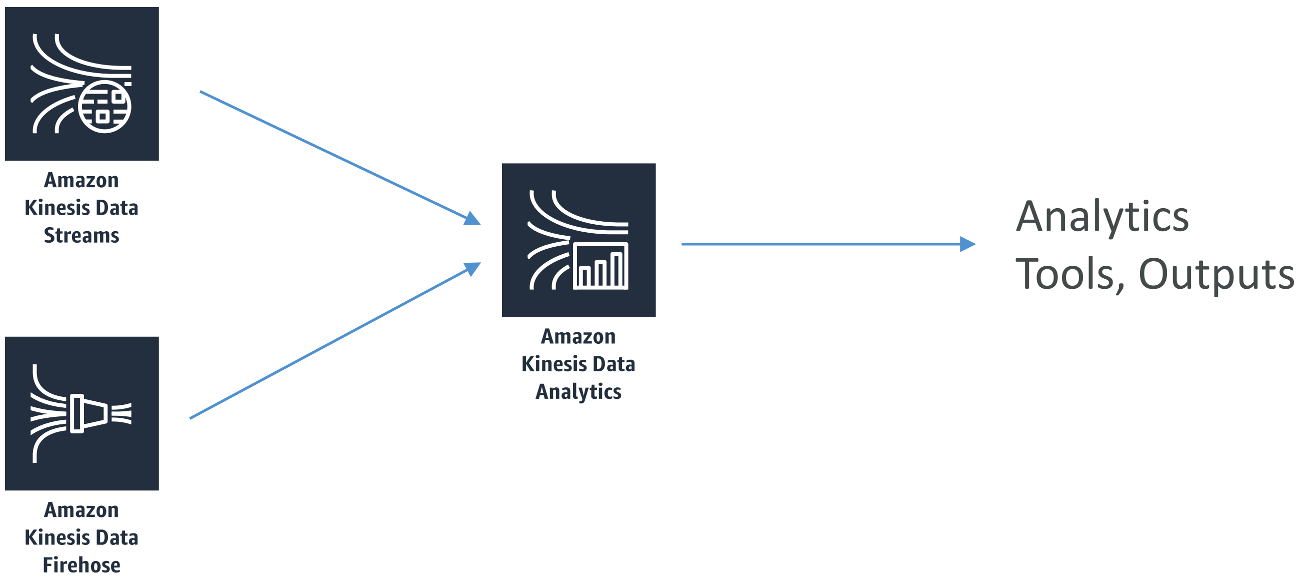
Kinesis Data Streams vs Firehose

- Streams
  - Going to write custom code (producer / consumer)
  - Real time (~200 ms latency for classic, ~70 ms latency for enhanced fan-out)
  - Automatic scaling with On-demand Mode
  - Data Storage for 1 to 365 days, replay capability, multi consumers

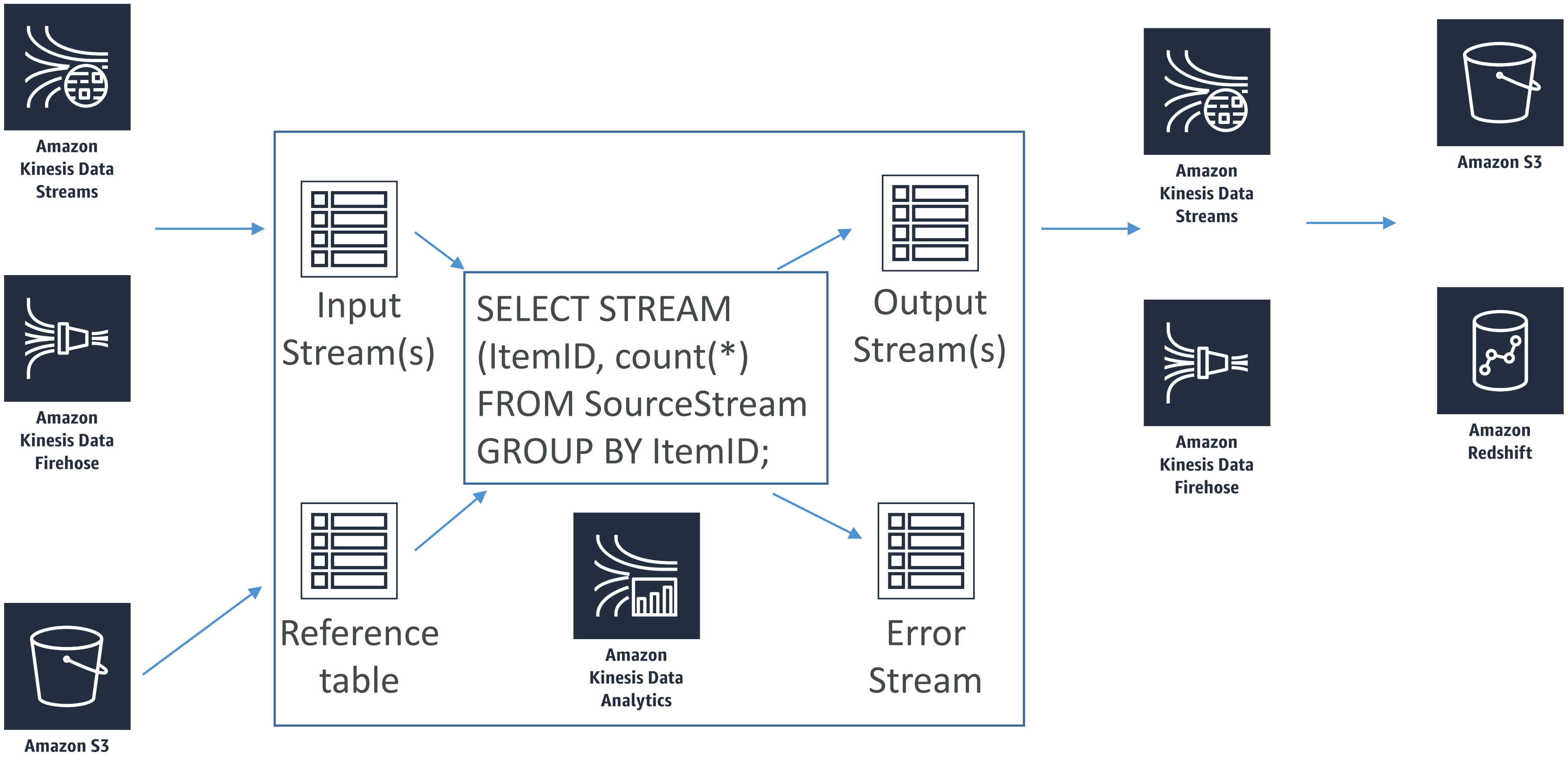
- Firehose
  - Fully managed, send to S3, Splunk, Redshift, ElasticSearch
  - Serverless data transformations with Lambda
  - **Near** real time
  - Automated Scaling
  - No data storage

DataCumulus Sundog™ Education

Kinesis Analytics, Conceptually…



Kinesis Analytics, In more depth…



Kinesis Data Analytics

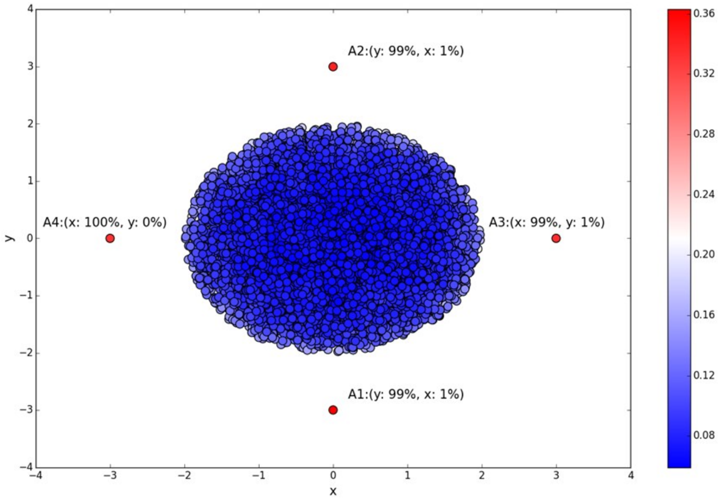
Amazon Kinesis Data Analytics

- Use cases
  - Streaming ETL: **select columns**, make simple transformations, on streaming data
  - Continuous metric generation: live leaderboard for a mobile game
  - Responsive analytics: look for certain criteria and build alerting (filtering)
- Features
  - Pay only for resources consumed (but it's not cheap)
  - Serverless; scales automatically
  - Use IAM permissions to access streaming source and destination(s)
  - SQL or Flink to write the computation
  - Schema discovery
  - Lambda can be used for pre-processing

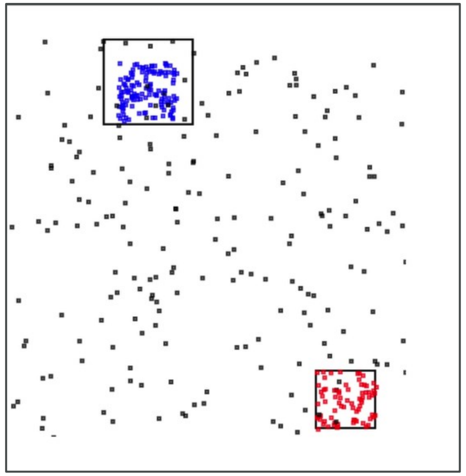
DataCumulus Sundog™ Education

Machine Learning on Kinesis Data Analytics

- **RANDOM_CUT_FOREST**
  - SQL function used for **anomaly detection** on numeric columns in a stream
  - Example: detect anomalous subway ridership during the NYC marathon
  - Uses **recent history** to compute model



- **HOTSPOTS**
  - locate and return information about relatively dense regions in your data
  - Example: a collection of overheated servers in a data center



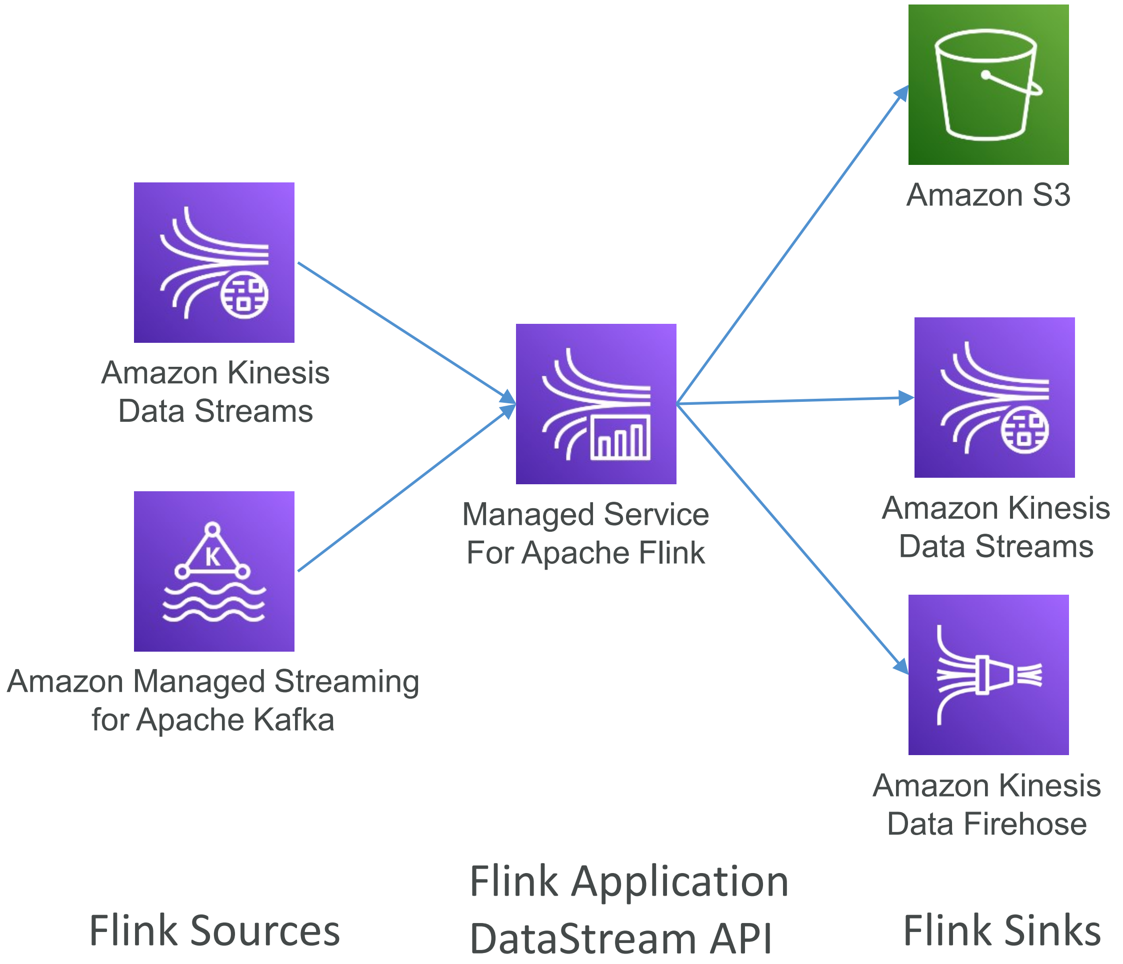
DataCumulus Sundog Education

Kinesis Data Analytics + Lambda

- AWS Lambda can be a destination as well
- Allows lots of flexibility for post-processing
  - Aggregating rows
  - Translating to different formats
  - Transforming and enriching data
  - Encryption
- Opens up access to other services & destinations
  - S3, DynamoDB, Aurora, Redshift, SNS, SQS, CloudWatch

Managed Service for Apache Flink

- Formerly Kinesis Data Analytics for Apache Flink or for Java
  - Kinesis Data Analytics always used Flink under the hood
  - But now supports Python and Scala
  - Flink is a framework for processing data streams
- MSAF integrates Flink with AWS
  - Instead of using SQL, you can develop your own Flink application from scratch and load it into MSAF via S3
- In addition to the DataStream API, there is a Table API for SQL access
- Serverless



DataCumulus Sundog Education

Common use-cases

- Streaming ETL
- Continuous metric generation
- Responsive analytics

Amazon Kinesis Data Analytics

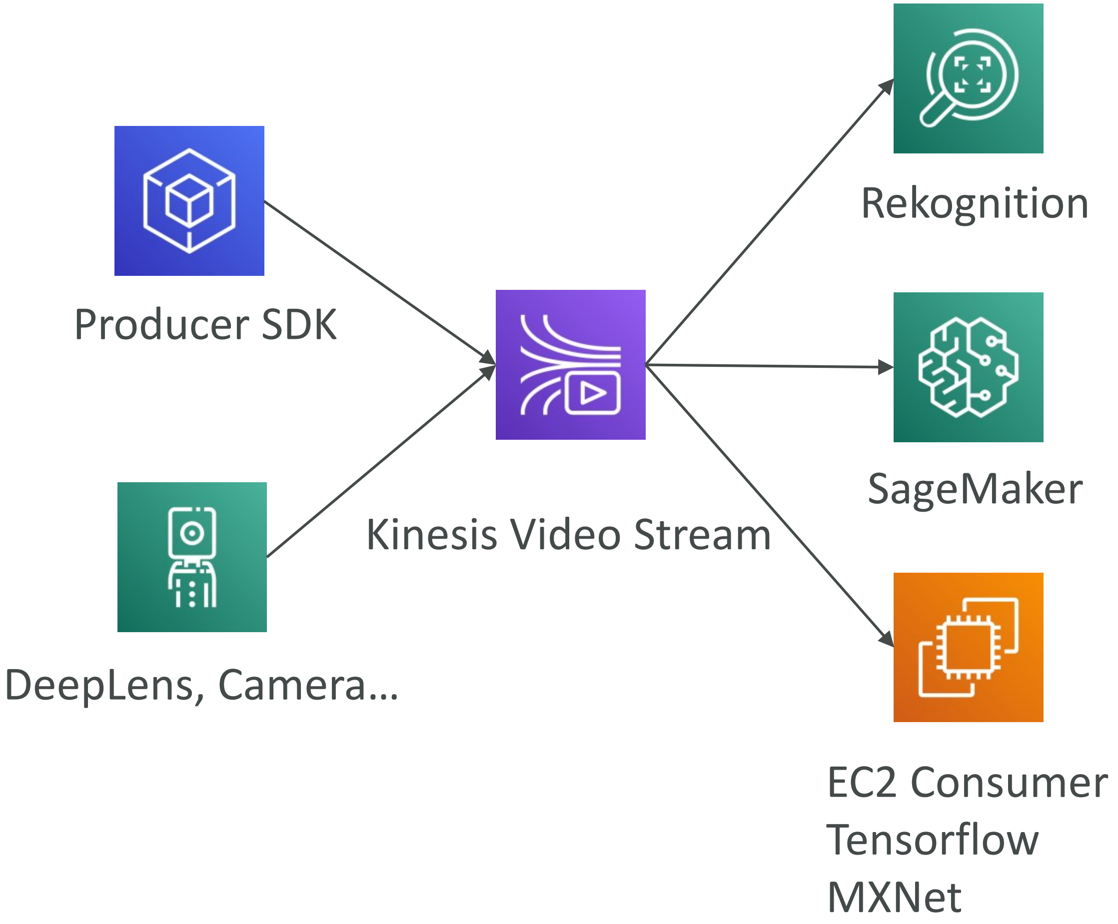
Kinesis Analytics

- Pay only for resources consumed (but it's not cheap)
  - Charged by Kinesis Processing Units (KPU's) consumed per hour
  - 1 KPU = 1 vCPU + 4GB
- Serverless; scales automatically
- Use IAM permissions to access streaming source and destination(s)
- Schema discovery

DataCumulus Sundog™ Education

Kinesis Video Stream

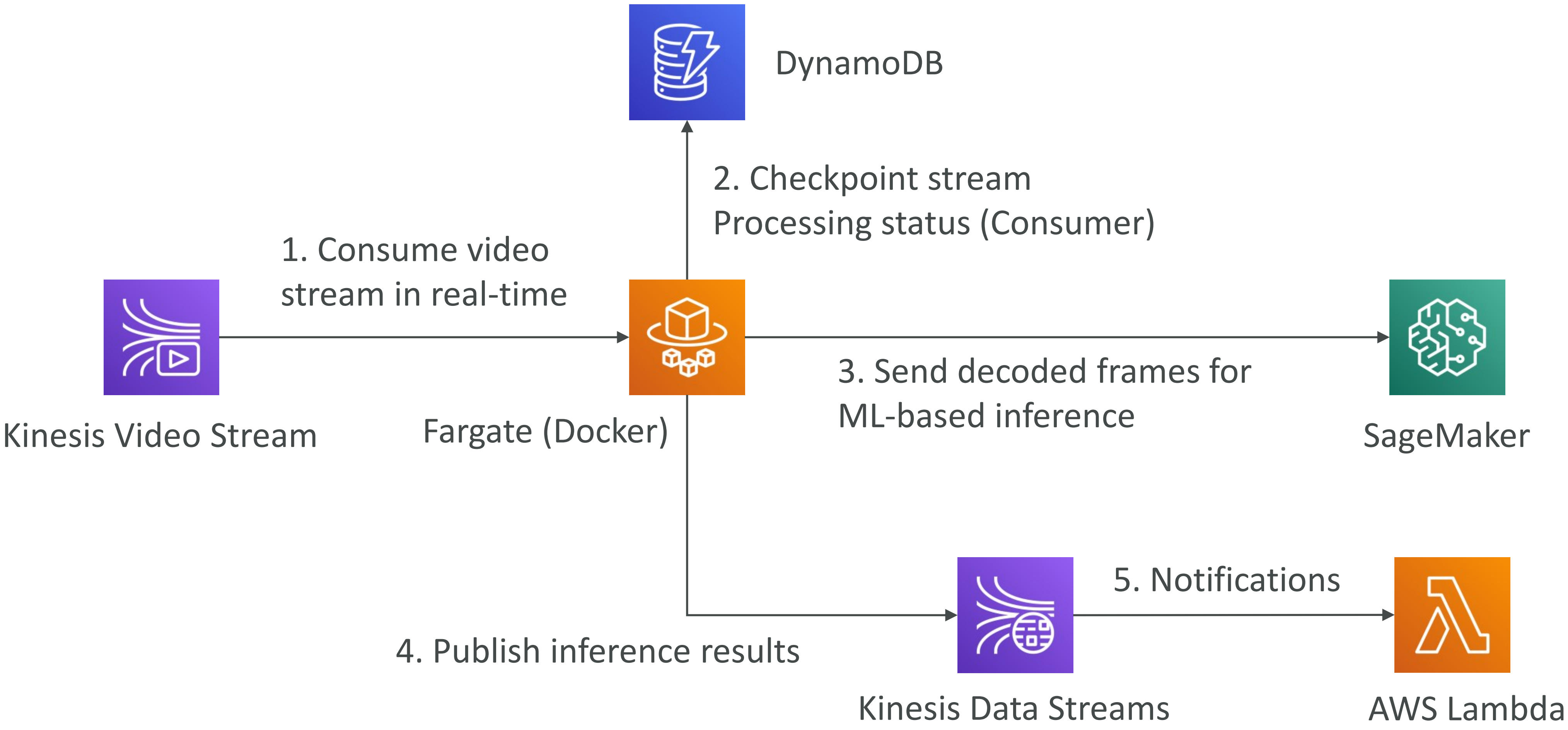
- Producers:
  - security camera, body-worn camera, **AWS DeepLens**, smartphone camera, audio feeds, images, RADAR data, **RTSP camera**.
  - One producer per video stream
- Video playback capability
- Consumers
  - build your own (MXNet, Tensorflow)
  - **AWS SageMaker**
  - **Amazon Rekognition Video**
- Keep data for 1 hour to 10 years



DataCumulus Sundog Education

Kinesis Video Streams use cases

https://aws.amazon.com/blogs/machine-learning/analyze-live-video-at-scale-in-real-time-using-amazon-kinesis-video-streams-and-amazon-sagemaker/

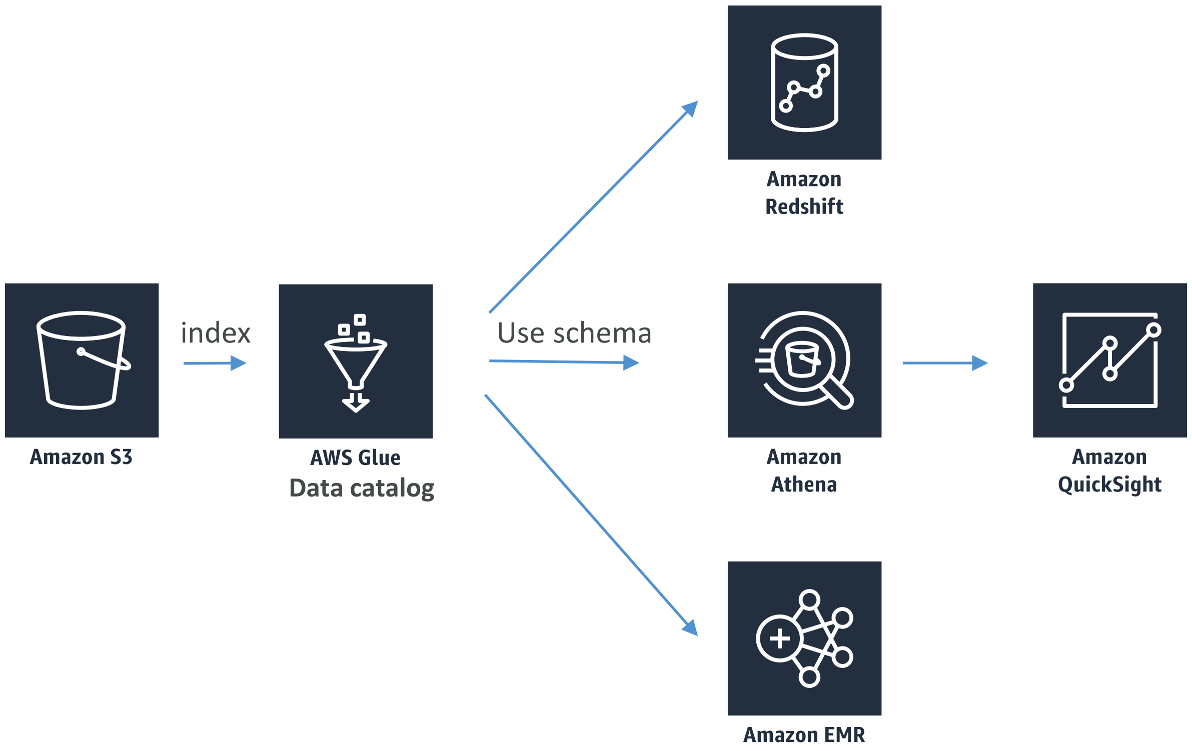


Kinesis Summary – Machine Learning

- Kinesis Data Stream: create real-time machine learning applications
- Kinesis Data Firehose: ingest massive data near-real time
- Kinesis Data Analytics: real-time ETL / ML algorithms on streams
- Kinesis Video Stream: real-time video stream to create ML applications

Glue Data Catalog

- Metadata repository for all your tables
  - Automated Schema Inference
  - Schemas are versioned
- Integrates with Athena or Redshift Spectrum (schema & data discovery)
- **Glue Crawlers** can help build the **Glue Data Catalog**



DataCumulus
Sundog™ Education

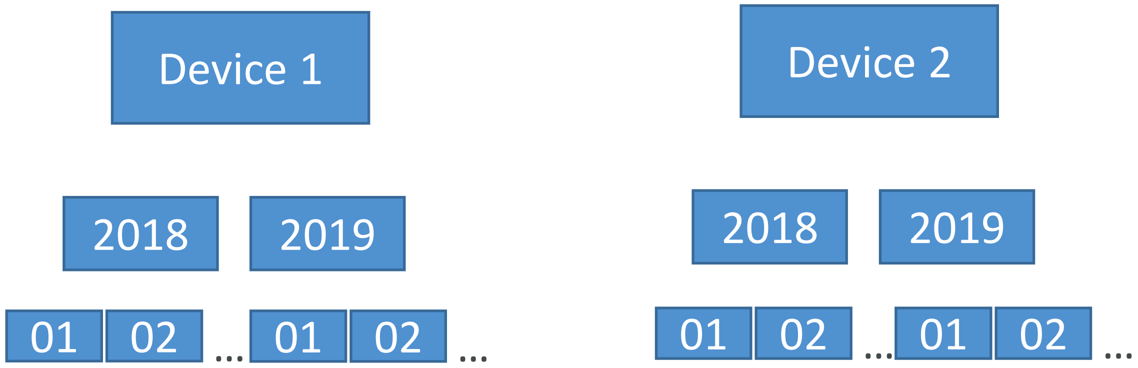
Glue Data Catalog - Crawlers

- **Crawlers** go through your data to infer schemas and partitions
- Works JSON, Parquet, CSV, relational store
- Crawlers work for: S3, Amazon Redshift, Amazon RDS
- Run the Crawler on a Schedule or On Demand

- Need an IAM role / credentials to access the data stores

Glue and S3 Partitions

- Glue crawler will extract partitions based on how your S3 data is organized
- Think up front about how you will be querying your data lake in S3
- Example: devices send sensor data every hour
- Do you query primarily by **time ranges**?
  - If so, organize your buckets as s3://my-bucket/dataset/**yyyy/mm/dd/device**
- Do you query primarily by **device**?
  - If so, organize your buckets as s3://my-bucket/dataset/**device/yyyy/mm/dd**



Glue ETL

- Transform data, Clean Data, Enrich Data (before doing analysis)
  - Generate ETL code in Python or Scala, you can modify the code
  - Can provide your own Spark or PySpark scripts
  - Target can be S3, JDBC (RDS, Redshift), or in Glue Data Catalog
- Fully managed, cost effective, pay only for the resources consumed
- Jobs are run on a serverless Spark platform

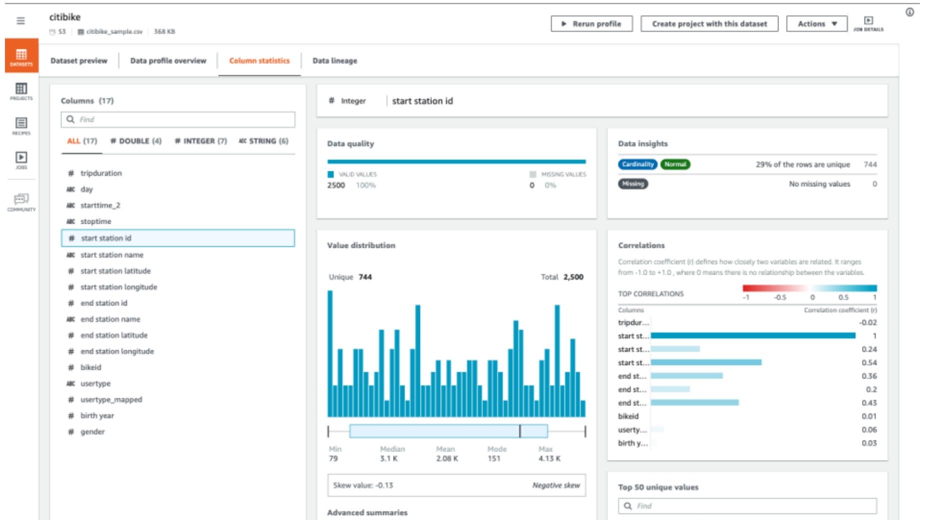
- Glue Scheduler to schedule the jobs
- Glue Triggers to automate job runs based on “events”

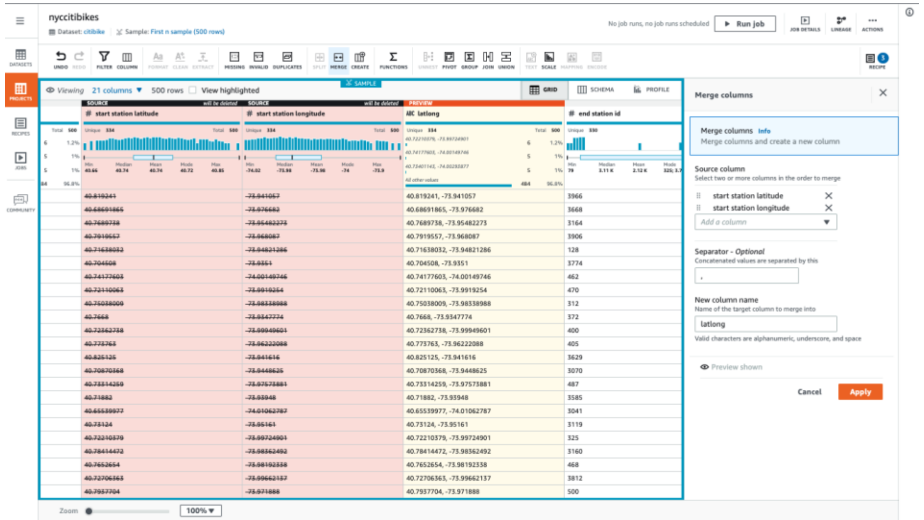
Glue ETL - Transformations

- Bundled Transformations:
  - DropFields, DropNullFields – remove (null) fields
  - Filter – specify a function to filter records
  - Join – to enrich data
  - Map - add fields, delete fields, perform external lookups
- Machine Learning Transformations:
  - **FindMatches ML:** identify duplicate or matching records in your dataset, even when the records do not have a common unique identifier and no fields match exactly.
- Format conversions: CSV, JSON, Avro, Parquet, ORC, XML
- Apache Spark transformations (example: K-Means)

AWS Glue DataBrew

- Allows you to clean and normalize data **without writing any code**
- Reduces ML and analytics data preparation time by up to 80%
- Data sources include S3, Redshift, Aurora, Glue Data Catalog…
- +250 ready-made transformations to automate tasks
  - Filtering anomalies, data conversion, correct invalid values…





https://aws.amazon.com/glue/features/databrew/

AWS Data Stores for Machine Learning

- <u>Redshift:</u>
  - Data Warehousing, SQL analytics (OLAP - Online analytical processing)
  - Load data from S3 to Redshift
  - Use Redshift Spectrum to query data directly in S3 (no loading)
- <u>RDS, Aurora:</u>
  - Relational Store, SQL (OLTP - Online Transaction Processing)
  - Must provision servers in advance

AWS Data Stores for Machine Learning

- DynamoDB:
  - NoSQL data store, serverless, provision read/write capacity
  - Useful to store a machine learning model served by your application

- S3:
  - Object storage
  - Serverless, infinite storage
  - Integration with most AWS Services

AWS Data Stores for Machine Learning

- OpenSearch (previously ElasticSearch):
  - Indexing of data
  - Search amongst data points
  - Clickstream Analytics

- ElastiCache:
  - Caching mechanism
  - Not really used for Machine Learning

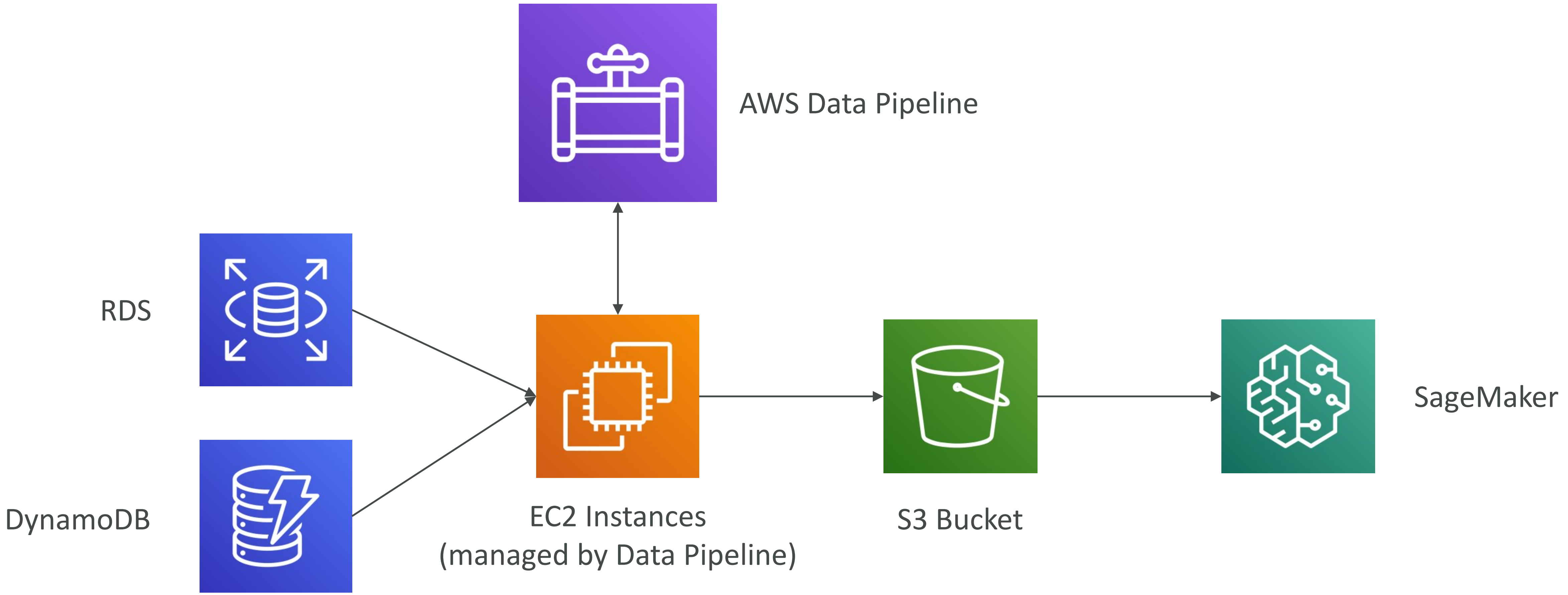
DataCumulus Sundog Education

AWS Data Pipeline Features

- Destinations include S3, RDS, DynamoDB, Redshift and EMR
- Manages task dependencies
- Retries and notifies on failures
- Data sources may be on-premises
- Highly available

DataCumulus Sundog Education

Data Pipeline example



AWS Data Pipeline vs Glue

- Glue:
  - Glue ETL - Run Apache Spark code, Scala or Python based, focus on the ETL
  - Glue ETL - Do not worry about configuring or managing the resources
  - Data Catalog to make the data available to Athena or Redshift Spectrum
- Data Pipeline:
  - Orchestration service
  - More control over the environment, compute resources that run code, & code
  - Allows access to EC2 or EMR instances (creates resources in your own account)

AWS Batch

- Run batch jobs as Docker images
- Dynamic provisioning of the instances (EC2 & Spot Instances)
- Optimal quantity and type based on volume and requirements
- No need to manage clusters, fully **serverless**
- You just pay for the underlying EC2 instances

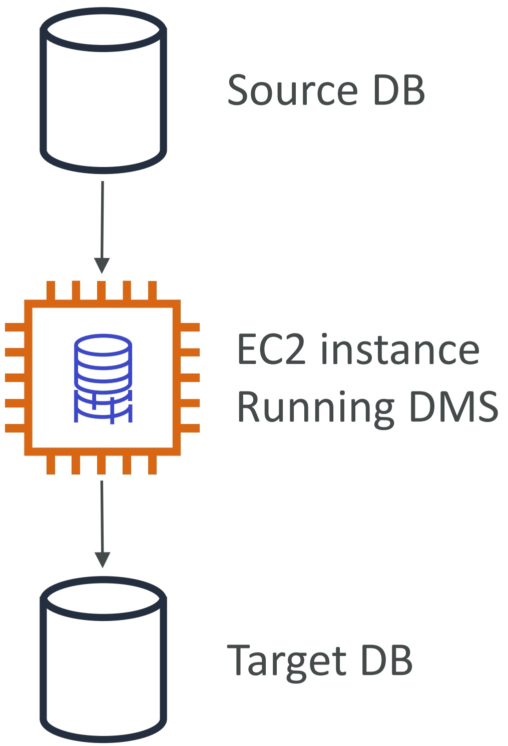
- Schedule Batch Jobs using CloudWatch Events
- Orchestrate Batch Jobs using AWS Step Functions

AWS Batch vs Glue

- Glue:
  - Glue ETL - Run Apache Spark code, Scala or Python based, focus on the ETL
  - Glue ETL - Do not worry about configuring or managing the resources
  - Data Catalog to make the data available to Athena or Redshift Spectrum
- Batch:
  - For any computing job regardless of the job (must provide Docker image)
  - Resources are created in your account, managed by Batch
  - For any non-ETL related work, Batch is probably better

DMS – Database Migration Service

- Quickly and securely migrate databases to AWS, resilient, self healing
- The source database remains available during the migration
- Supports:
  - Homogeneous migrations: ex Oracle to Oracle
  - Heterogeneous migrations: ex Microsoft SQL Server to Aurora
- Continuous Data Replication using CDC
- You must create an EC2 instance to perform the replication tasks



AWS DMS vs Glue

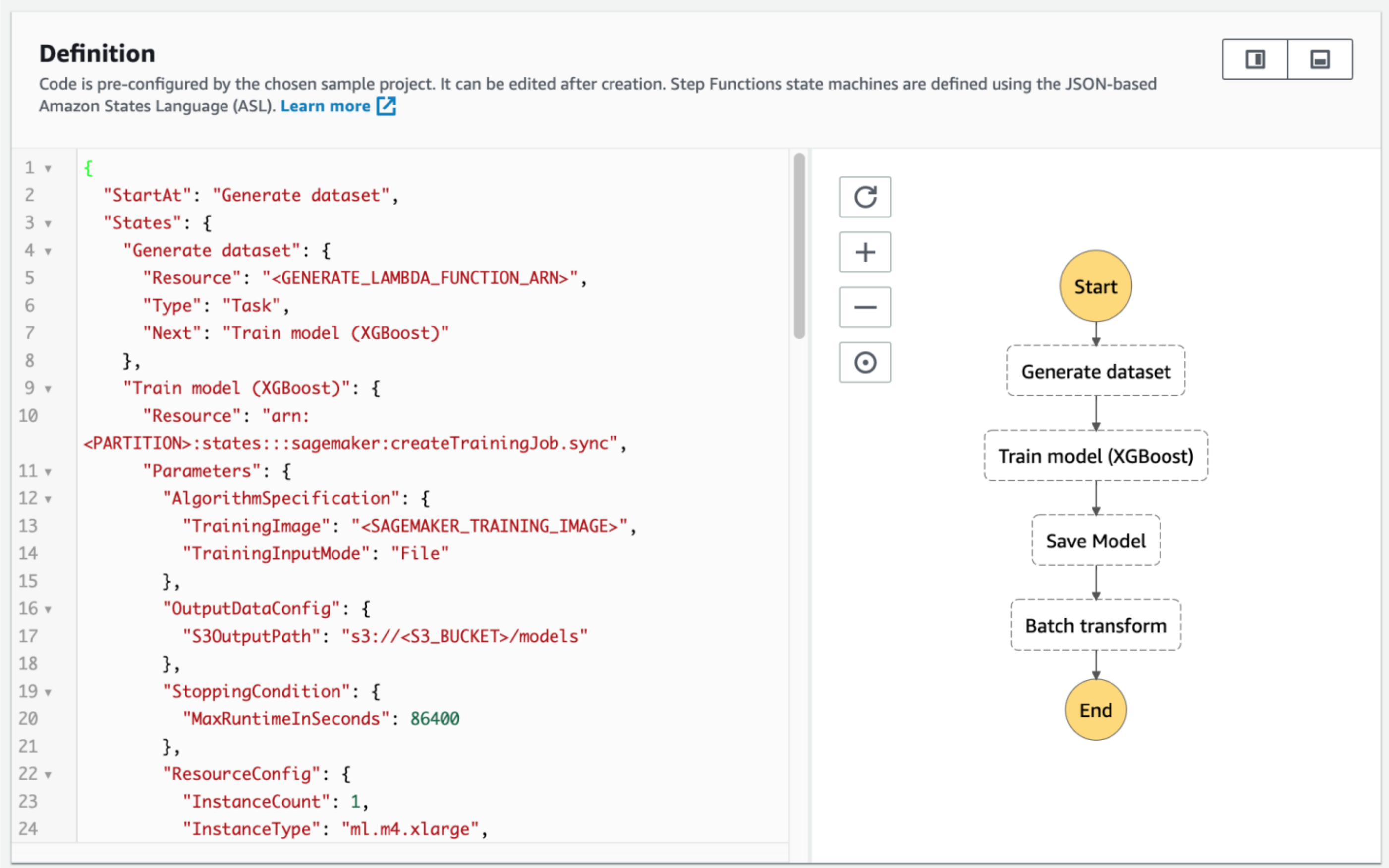
- Glue:
  - Glue ETL - Run Apache Spark code, Scala or Python based, focus on the ETL
  - Glue ETL - Do not worry about configuring or managing the resources
  - Data Catalog to make the data available to Athena or Redshift Spectrum

- AWS DMS:
  - Continuous Data Replication
  - No data transformation
  - Once the data is in AWS, you can use Glue to transform it

AWS Step Functions

- **Use to design workflows**
- Easy visualizations
- Advanced Error Handling and Retry mechanism outside the code
- Audit of the history of workflows
- Ability to “Wait” for an arbitrary amount of time
- Max execution time of a State Machine is 1 year

Step Functions – Examples
Train a Machine Learning Model

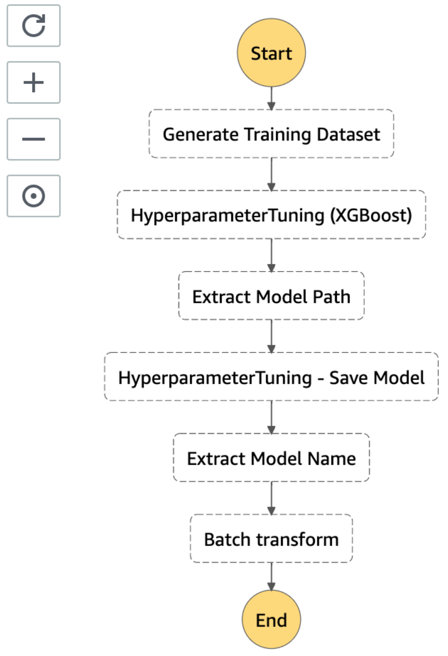


Step Functions – Examples
Tune a Machine Learning Model

Definition

Code is pre-configured by the chosen sample project. It can be edited after creation. Step Functions state machines are defined using the JSON-based Amazon States Language (ASL). **Learn more**

```
{
    "StartAt": "Generate Training Dataset",
    "States": {
        "Generate Training Dataset": {
            "Resource": "<GENERATE_LAMBDA_FUNCTION_ARN>",
            "Type": "Task",
            "Next": "HyperparameterTuning (XGBoost)"
        },
        "HyperparameterTuning (XGBoost)": {
            "Resource": "arn:
<PARTITION>:states:::sagemaker:createHyperParameterTuningJob.sync",
            "Parameters": {
                "HyperParameterTuningJobName.$": "
<JOB_NAME_FROM_LAMBDA>",
                "HyperParameterTuningJobConfig": {
                    "Strategy": "Bayesian",
                    "HyperParameterTuningJobObjective": {
                        "Type": "Minimize",
                        "MetricName": "validation:rmse"
                    },
                    "ResourceLimits": {
                        "MaxNumberOfTrainingJobs": 2,
                        "MaxParallelTrainingJobs": 2
                    },
                    "ParameterRanges": {
```

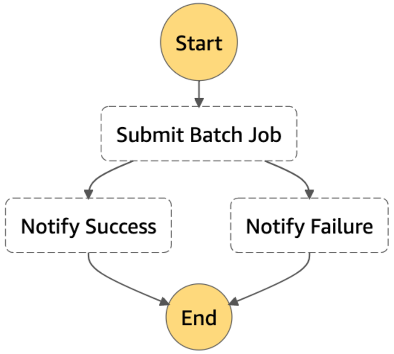


Step Functions – Examples
Manage a Batch Job

Definition

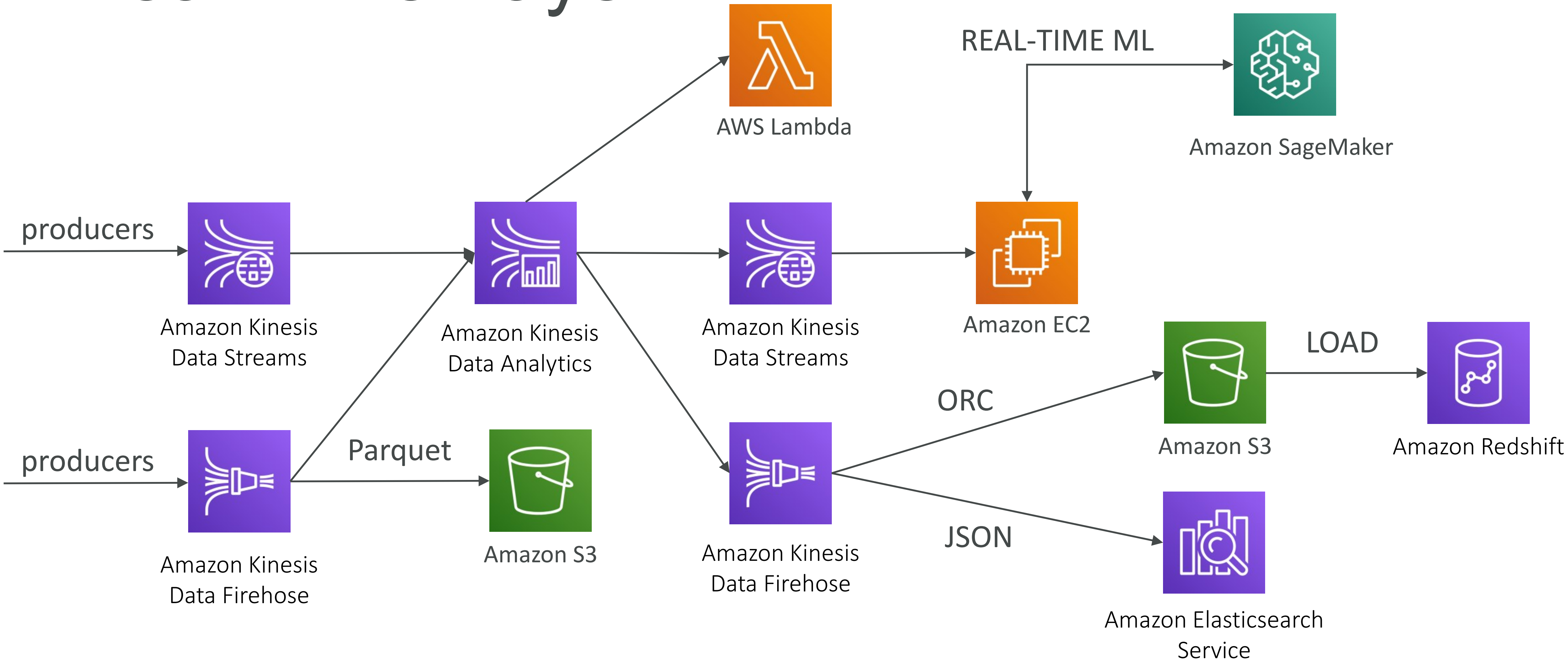
Code is pre-configured by the chosen sample project. It can be edited after creation. Step Functions state machines are defined using the JSON-based Amazon States Language (ASL). Learn more

```
{
  "Comment": "An example of the Amazon States Language for notification on an AWS Batch job completion",
  "StartAt": "Submit Batch Job",
  "TimeoutSeconds": 3600,
  "States": {
    "Submit Batch Job": {
      "Type": "Task",
      "Resource": "arn:<PARTITION>:states:::batch:submitJob.sync",
      "Parameters": {
        "JobName": "BatchJobNotification",
        "JobQueue": "<BATCH_QUEUE_ARN>",
        "JobDefinition": "<BATCH_JOB_DEFINITION_ARN>"
      },
      "Next": "Notify Success",
      "Catch": [
        {
          "ErrorEquals": [ "States.ALL" ],
          "Next": "Notify Failure"
        }
      ]
    },
    "Notify Success": {
      "Type": "Task",
      "Resource": "arn:<PARTITION>:states:::sns:publish",
```

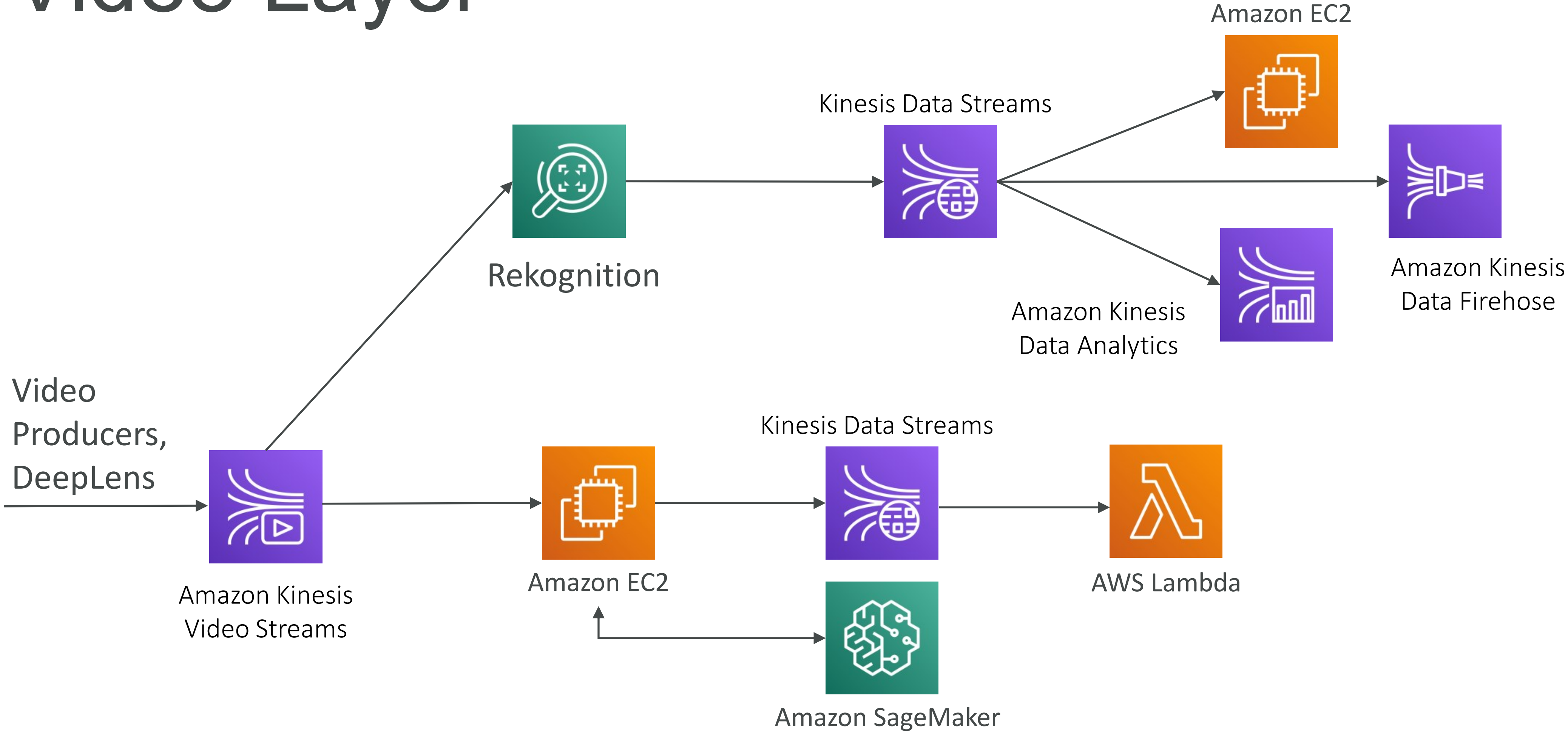


DataCumulus Sundog Education

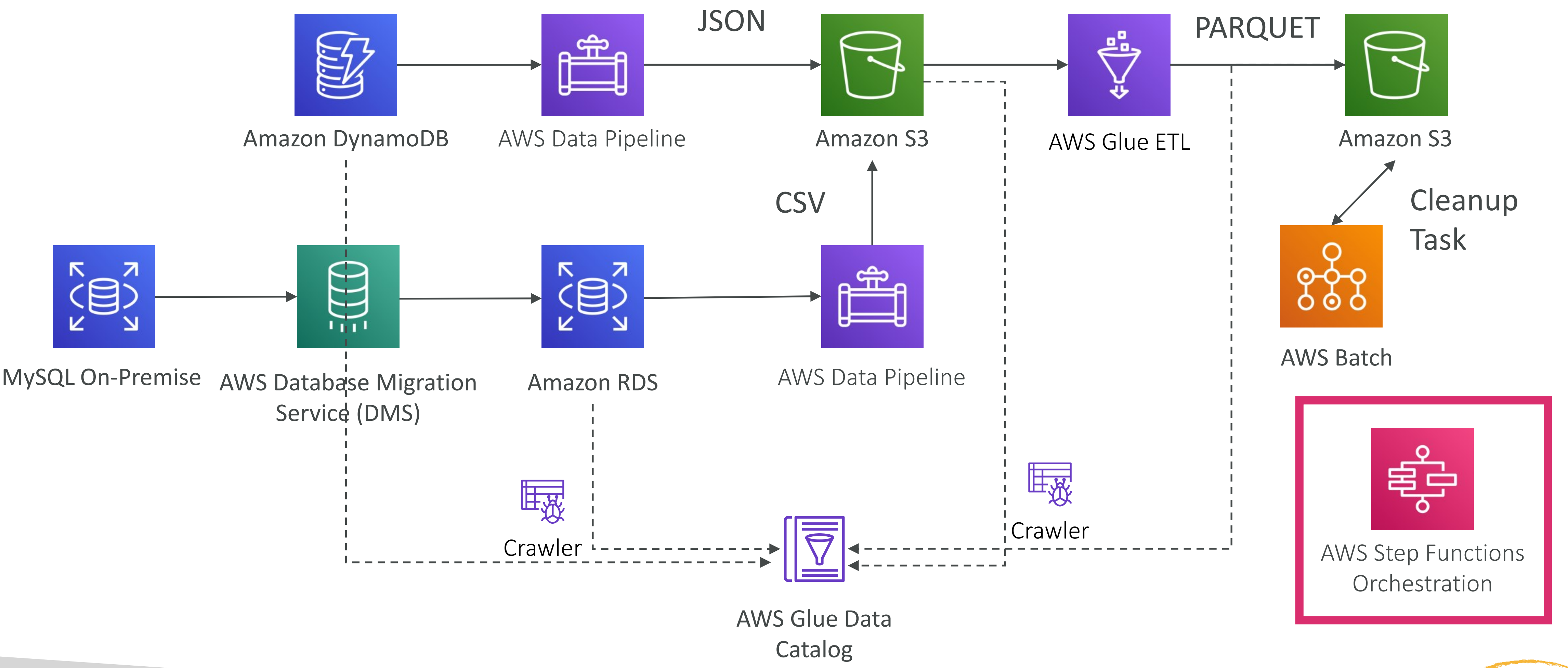
Full Data Engineering Pipeline
Real-Time Layer



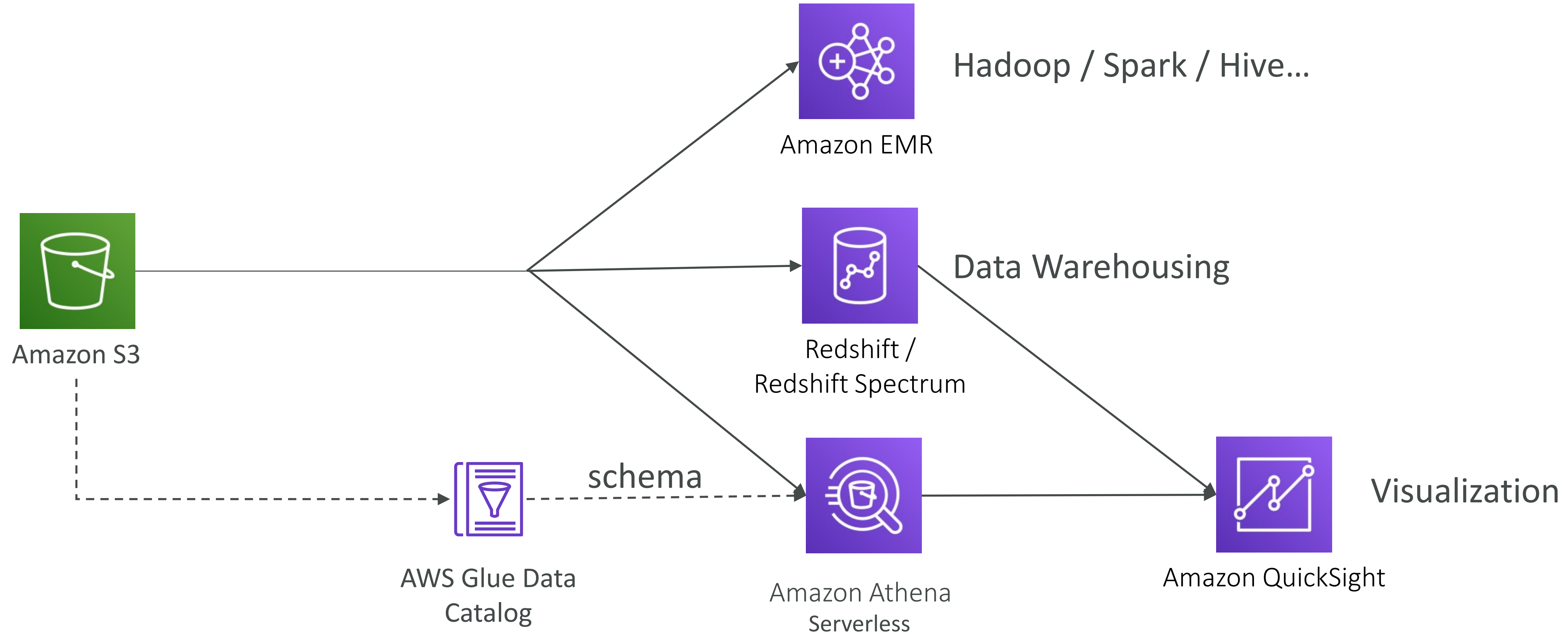
Full Data Engineering Pipeline
Video Layer



Full Data Engineering Pipeline Batch Layer

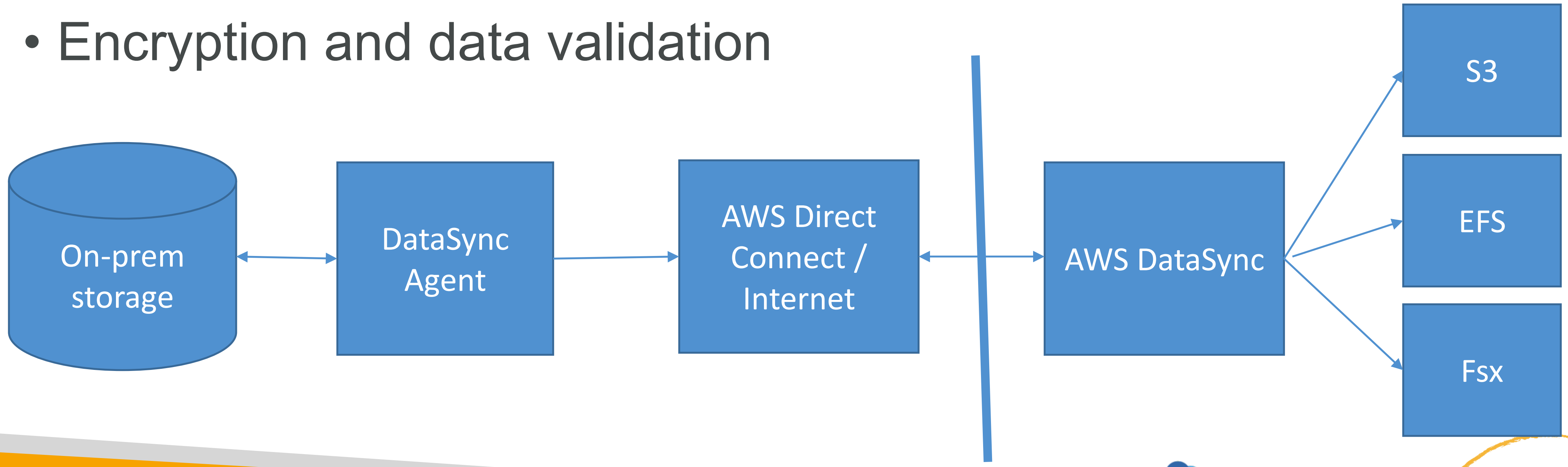


Full Data Engineering Pipeline
Analytics layer



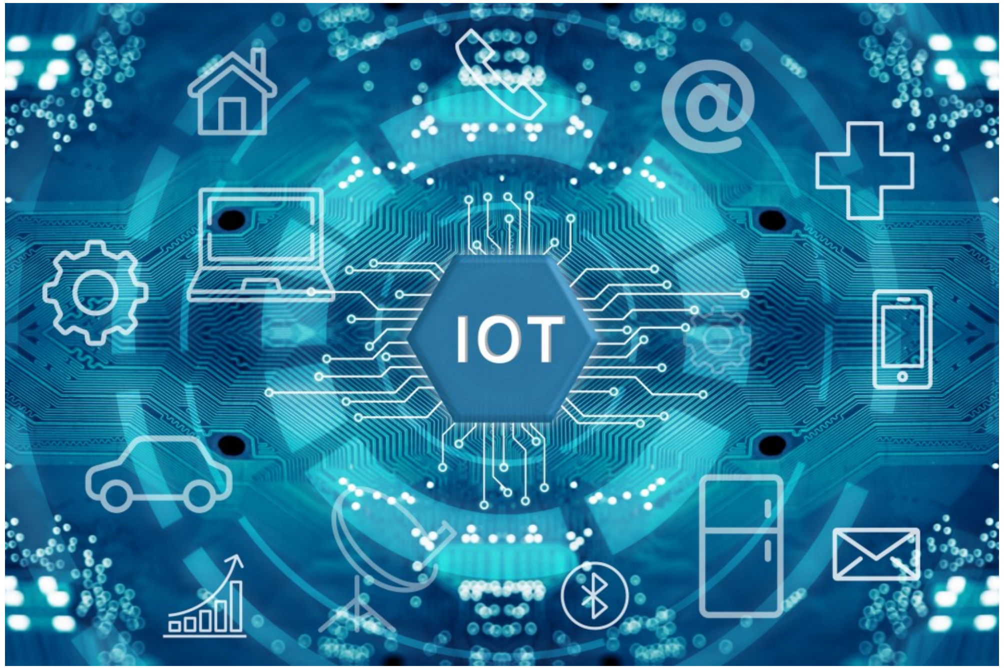
AWS DataSync

- For data migrations from on-premises to AWS storage services
- A DataSync Agent is deployed as a VM and connects to your internal storage
  - NFS, SMB, HDFS
- Encryption and data validation



MQTT

- An Internet of Things (IOT) thing
- Standard messaging protocol
- Think of it as how lots of sensor data might get transferred to your machine learning model
- The AWS IoT Device SDK can connect via MQTT



Exploratory Data Analysis

Python is becoming ubiquitous

- But the test will not test your Python knowledge.

```
In [1]: %matplotlib inline
        import numpy as np
        import pandas as pd

        df = pd.read_csv("PastHires.csv")
        df.head()
```

Out[1]:

	Years Experience	Employed?	Previous employers	Level of Education	Top-tier school	Interned	Hired
0	10	Y	4	BS	N	N	Y
1	0	N	0	BS	Y	Y	Y
2	7	N	6	BS	N	N	N
3	2	Y	1	MS	Y	N	Y
4	20	N	2	PhD	Y	N	N

Pandas

- A Python library for slicing and dicing your data
  - Data Frames
  - Series
  - Interoperates with **numpy**

```
In [12]: df[['Years Experience', 'Hired']][:5]
Out[12]:
```

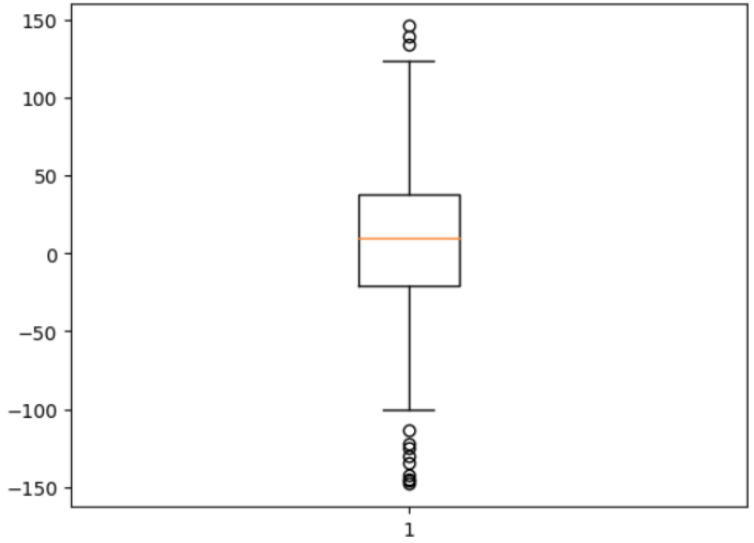
	Years Experience	Hired
0	10	Y
1	0	Y
2	7	N
3	2	Y
4	20	N

```
In [14]: degree_counts = df['Level of Education'].value_counts()
         degree_counts
Out[14]: BS     7
         PhD    4
         MS     2
         Name: Level of Education, dtype: int64
```

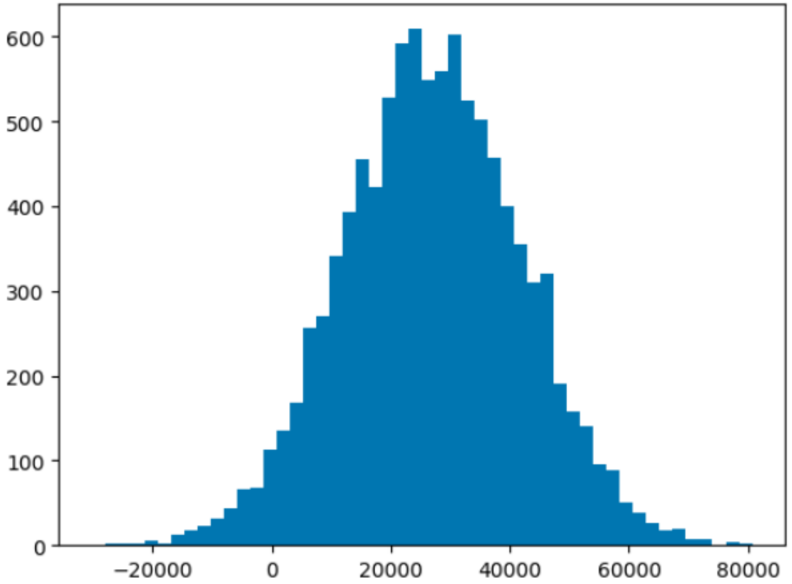

DataCumulus Sundog™ Education

Matplotlib

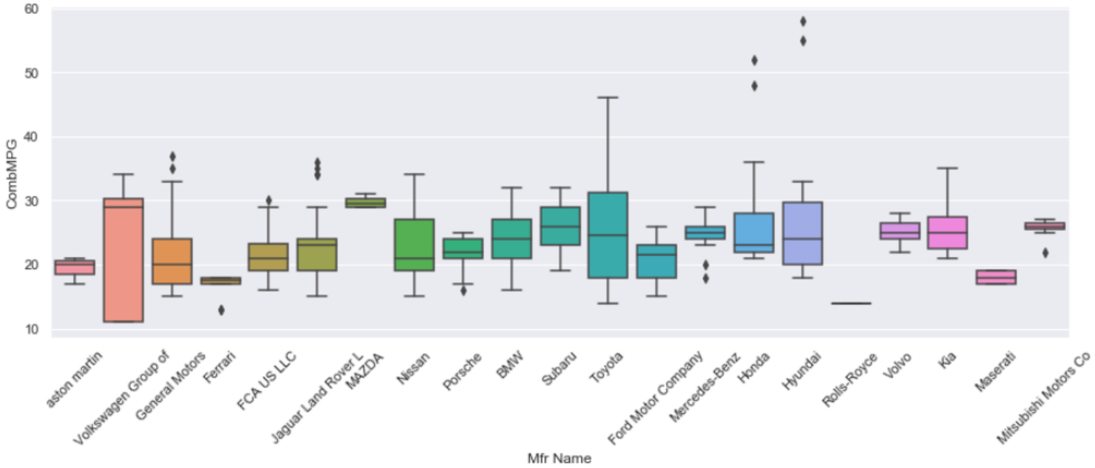
In [13]:
```
uniformSkewed = np.random.rand(100) * 100 - 40
high_outliers = np.random.rand(10) * 50 + 100
low_outliers = np.random.rand(10) * -50 - 100
data = np.concatenate((uniformSkewed, high_outliers, low_outliers))
plt.boxplot(data)
plt.show()
```



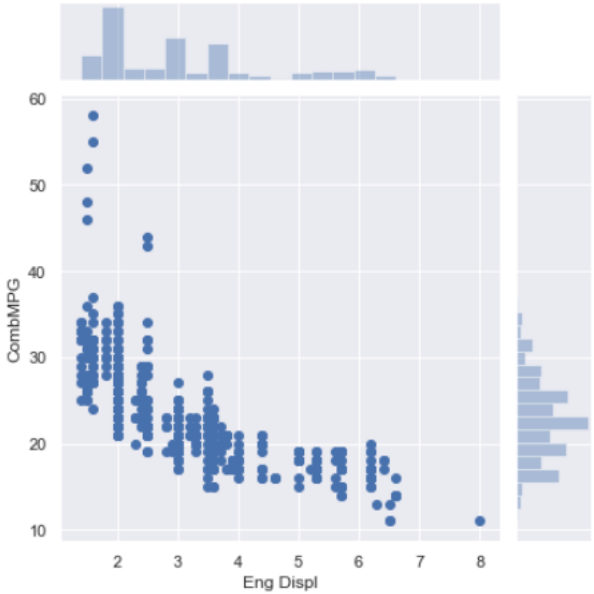
In [12]:
```
incomes = np.random.normal(27000, 15000, 10000)
plt.hist(incomes, 50)
plt.show()
```

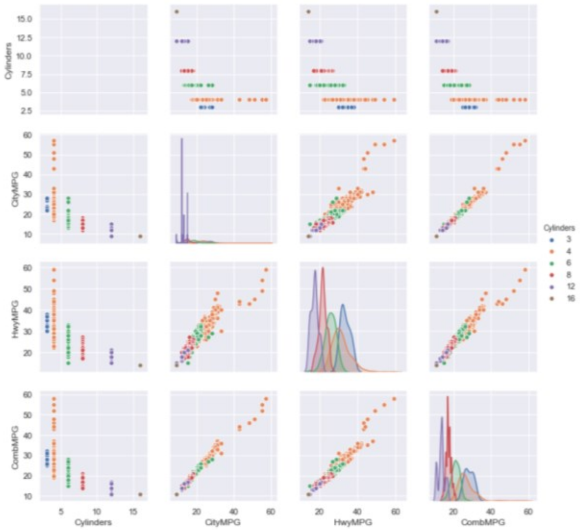


Seaborn

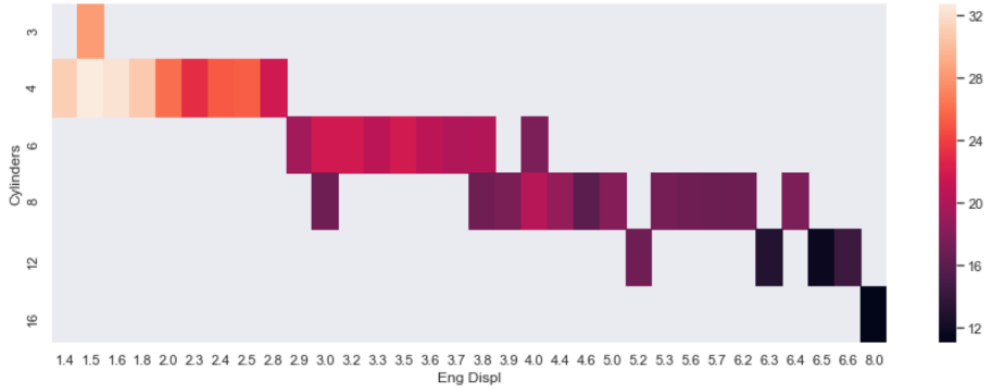



```
In [9]: sns.jointplot(x="Eng Displ", y="CombMPG", data=df)
Out[9]: <seaborn.axisgrid.JointGrid at 0x1d8a32f3e80>
```






```
In [14]: df2 = df.pivot_table(index='Cylinders', columns='Eng Displ', values='CombMPG', aggfunc='mean')
         sns.heatmap(df2)
Out[14]: <matplotlib.axes._subplots.AxesSubplot at 0x1d8a31ceac8>
```



sundog-education.com
datacumulus.com

scikit_learn

- Python library for machine learning models

Ensemble learning: using a random forest

We'll use a random forest of 10 decision trees to predict employment of specific candidate profiles:

In [7]:
```
from sklearn.ensemble import RandomForestClassifier

clf = RandomForestClassifier(n_estimators=10)
clf = clf.fit(X, y)

#Predict employment of an employed 10-year veteran
print (clf.predict([[10, 1, 4, 0, 0, 0]]))
#...and an unemployed 10-year veteran
print (clf.predict([[10, 0, 4, 0, 0, 0]]))
```

```
[1]
[0]
```

In [7]:
```
from sklearn import preprocessing

scaler = preprocessing.StandardScaler()
all_features_scaled = scaler.fit_transform(all_features)
all_features_scaled
```

Out[7]:
```
array([[ 0.7650629 ,  0.17563638,  1.39618483,  0.24046607],
       [ 0.15127063,  0.98104077,  1.39618483,  0.24046607],
       [-1.89470363, -1.43517241, -1.157718  ,  0.24046607],
       ...,
       [ 0.56046548,  0.98104077,  1.39618483,  0.24046607],
       [ 0.69686376,  0.98104077,  1.39618483,  0.24046607],
       [ 0.42406719,  0.17563638,  0.11923341,  0.24046607]])
```

Jupyter notebooks

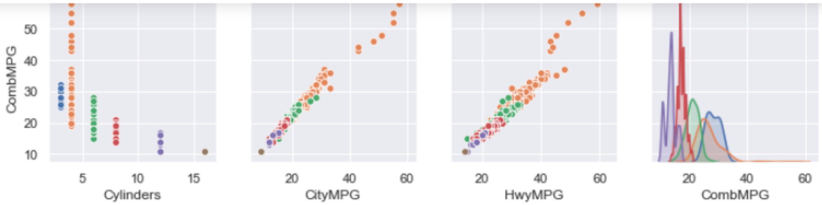
jupyter Seaborn Last Checkpoint: 06/13/2019 (autosaved)

Logout

File Edit View Insert Cell Kernel Widgets Help

Trusted | Python 2

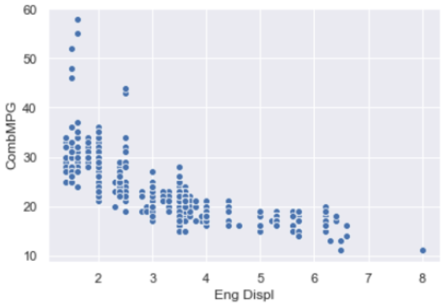
Run | Markdown



By studying the results above, you can see there is a relationship between number of cylinders and MPG, but MPG for 4-cylinder vehicles ranges really widely. There also appears to be a good linear relationship between the different ways of measuring MPG values, until you get into the higher MPG ratings.

Seaborn 1.9 also includes "scatterplot", which is exactly what it sounds like. It plots individual data points across two axes of your choosing, so you can see how your data is distributed across those dimensions.

```
In [8]: sns.scatterplot(x="Eng Displ", y="CombMPG", data=df)
Out[8]: <matplotlib.axes._subplots.AxesSubplot at 0x1d8a304dc18>
```



Seaborn also offers a "jointplot", which combines a scatterplot with histograms on both axes. This lets you visualize both the individual data points and the distribution across both dimensions at the same time.

Let’s look at an example

Many Flavors of Data

Major Types of Data

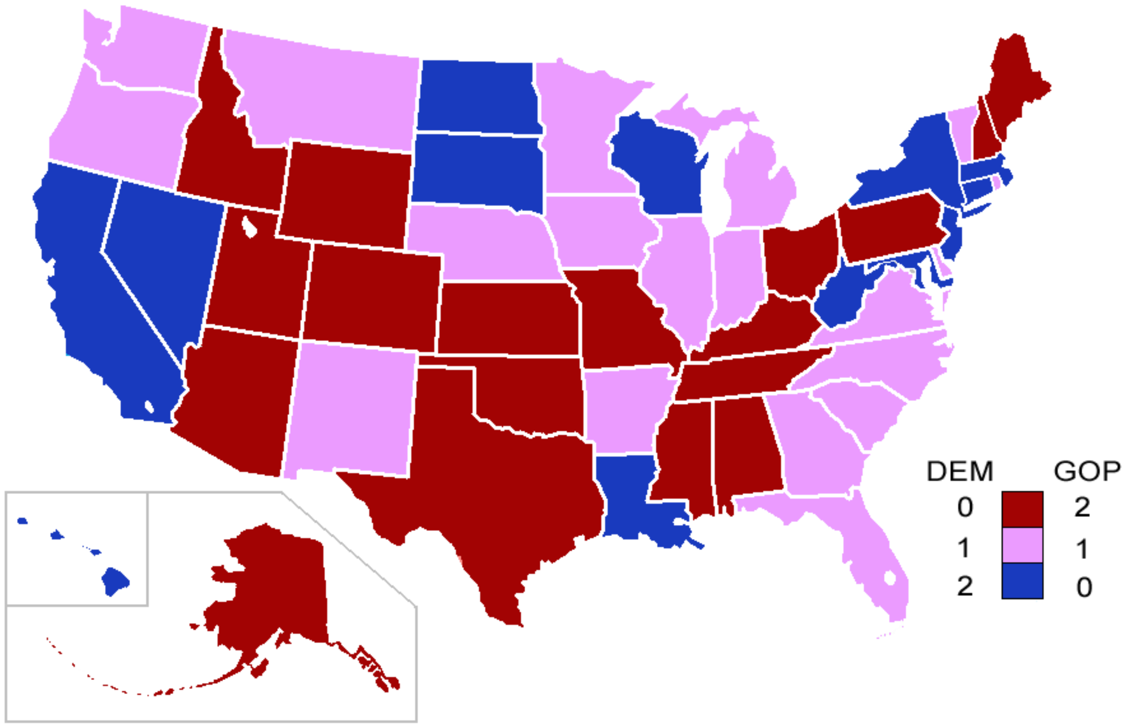
- Numerical
- Categorical
- Ordinal

Numerical

- Represents some sort of quantitative measurement
  - Heights of people, page load times, stock prices, etc.
- Discrete Data
  - Integer based; often counts of some event.
    - How many purchases did a customer make in a year?
    - How many times did I flip “heads”?
- Continuous Data
  - Has an infinite number of possible values
    - How much time did it take for a user to check out?
    - How much rain fell on a given day?

Categorical

- Qualitative data that has no inherent mathematical meaning
  - Gender, Yes/no (binary data), Race, State of Residence, Product Category, Political Party, etc.
- You can assign numbers to categories in order to represent them more compactly, but the numbers don't have mathematical meaning



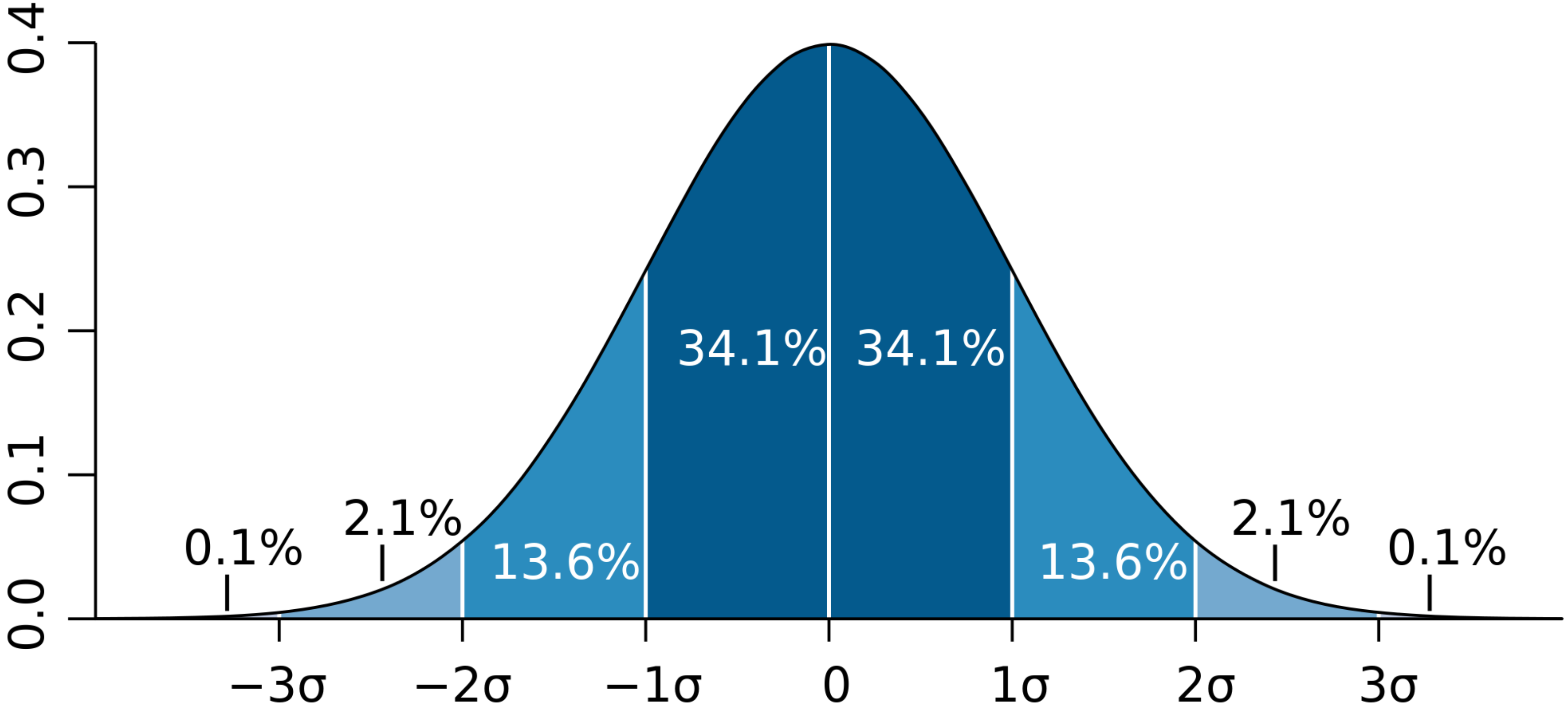
DataCumulus
Sundog Education

Ordinal

- A mixture of numerical and categorical
- Categorical data that has mathematical meaning
- Example: movie ratings on a 1-5 scale.
  - Ratings must be 1, 2, 3, 4, or 5
  - But these values have mathematical meaning; 1 means it's a worse movie than a 2.

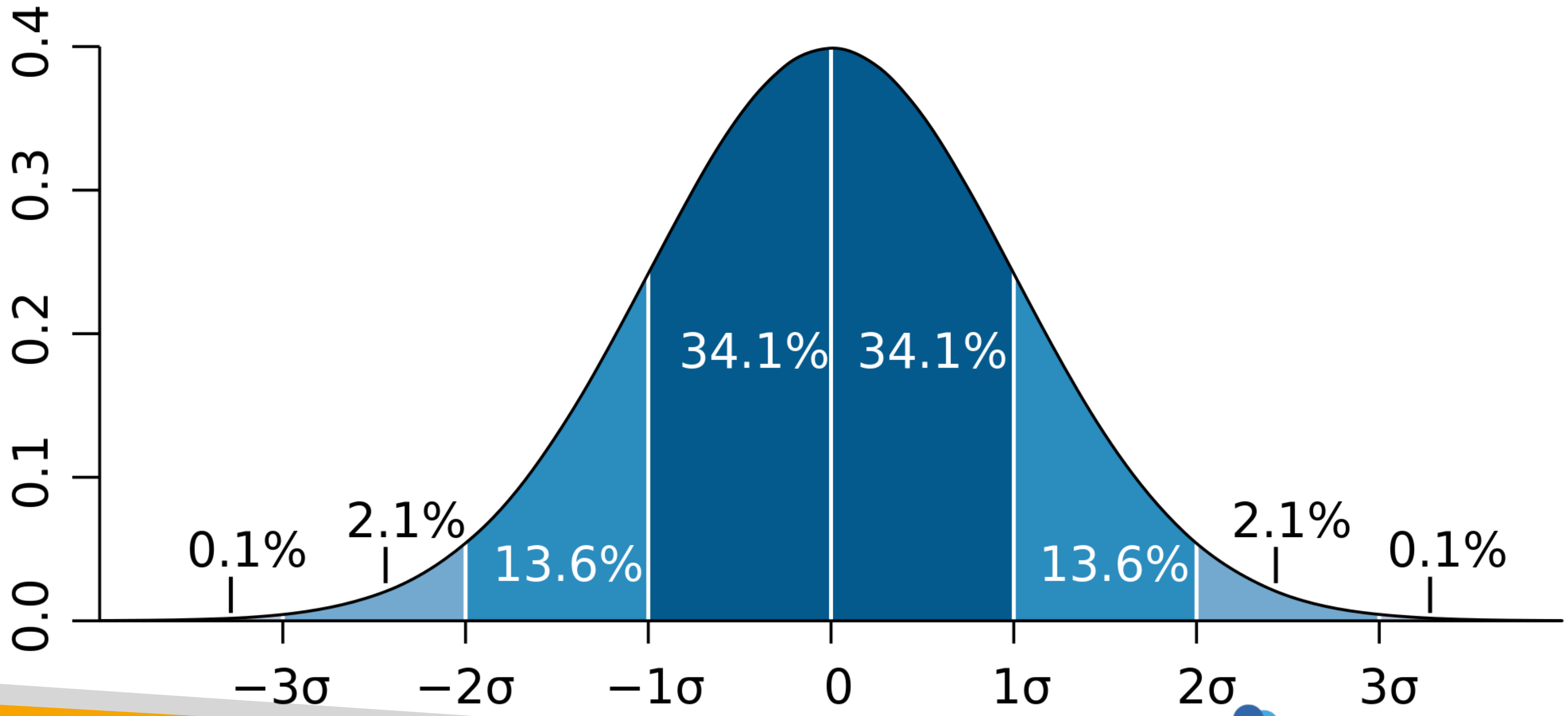
Data Distributions

A “normal distribution”

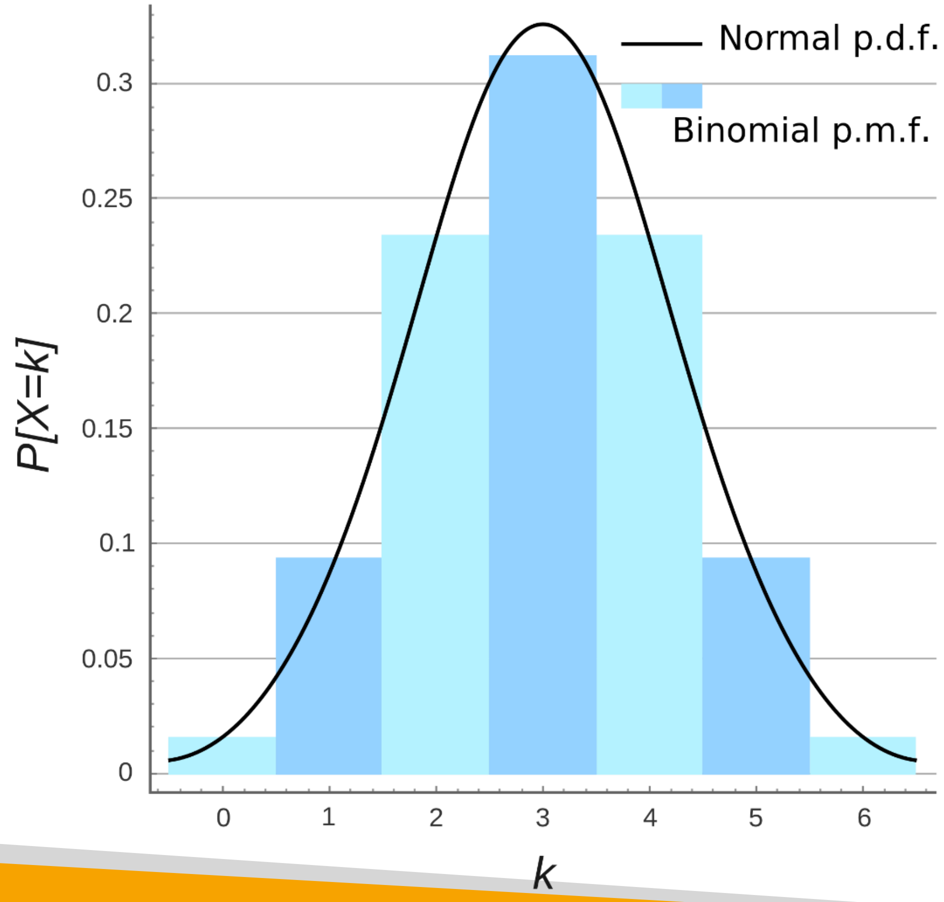


DataCumulus Sundog Education

Gives you the probability of a data point falling within some given range of a given value.



Probability Mass Function



Poisson Distribution

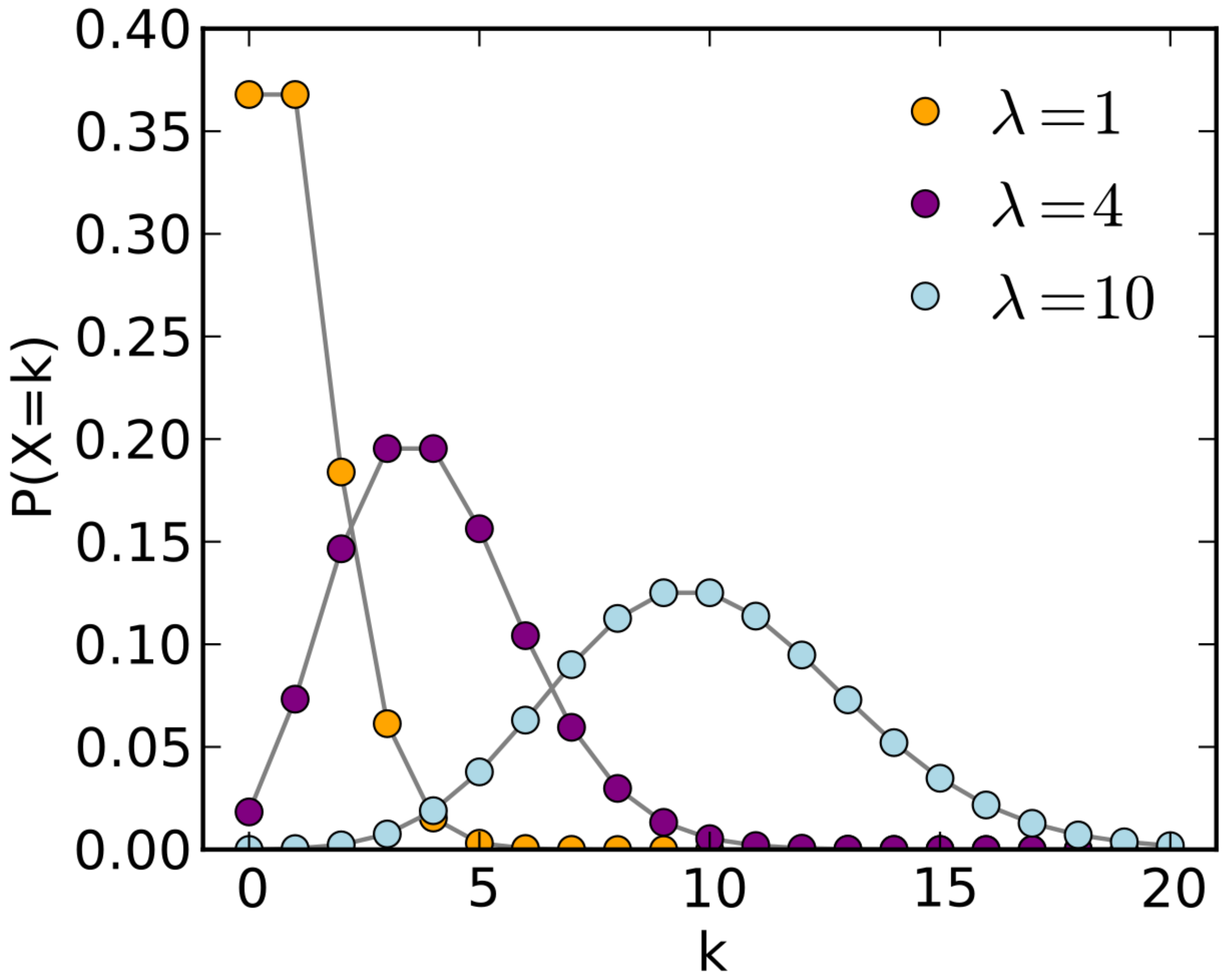
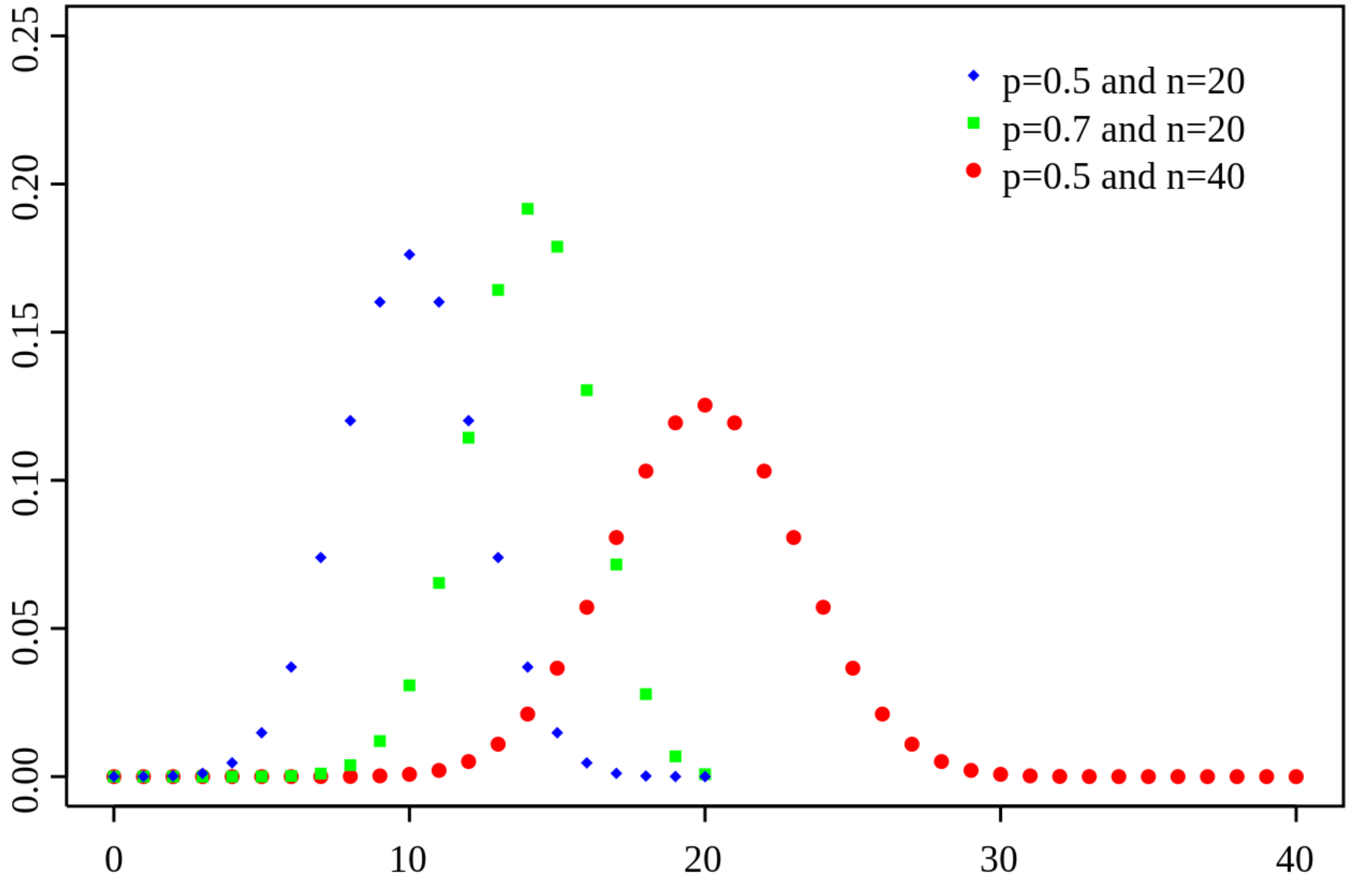


Image: Skbkekas, CC BY 3.0

Binomial Distribution



DataCumulus
Sundog Education

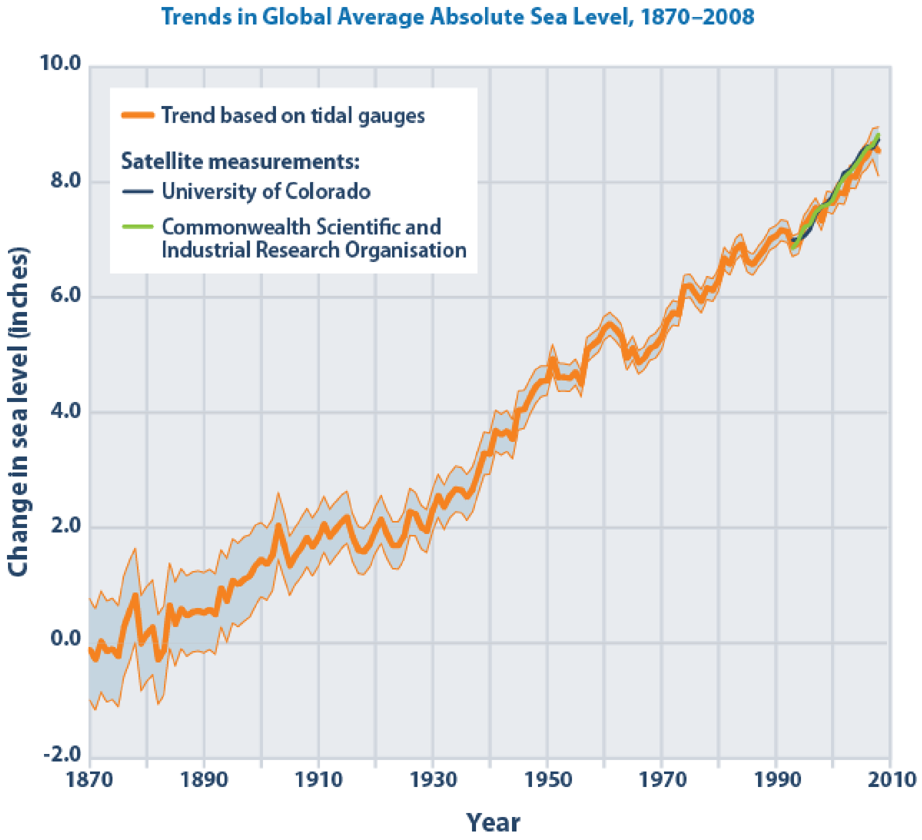
Bernoulli Distribution

- Special case of binomial distribution
- Has a single trial (n=1)
- Can think of a binomial distribution as the sum of Bernoulli distributions

Time Series Analysis

DataCumulus
Sundog™
Education

Trends

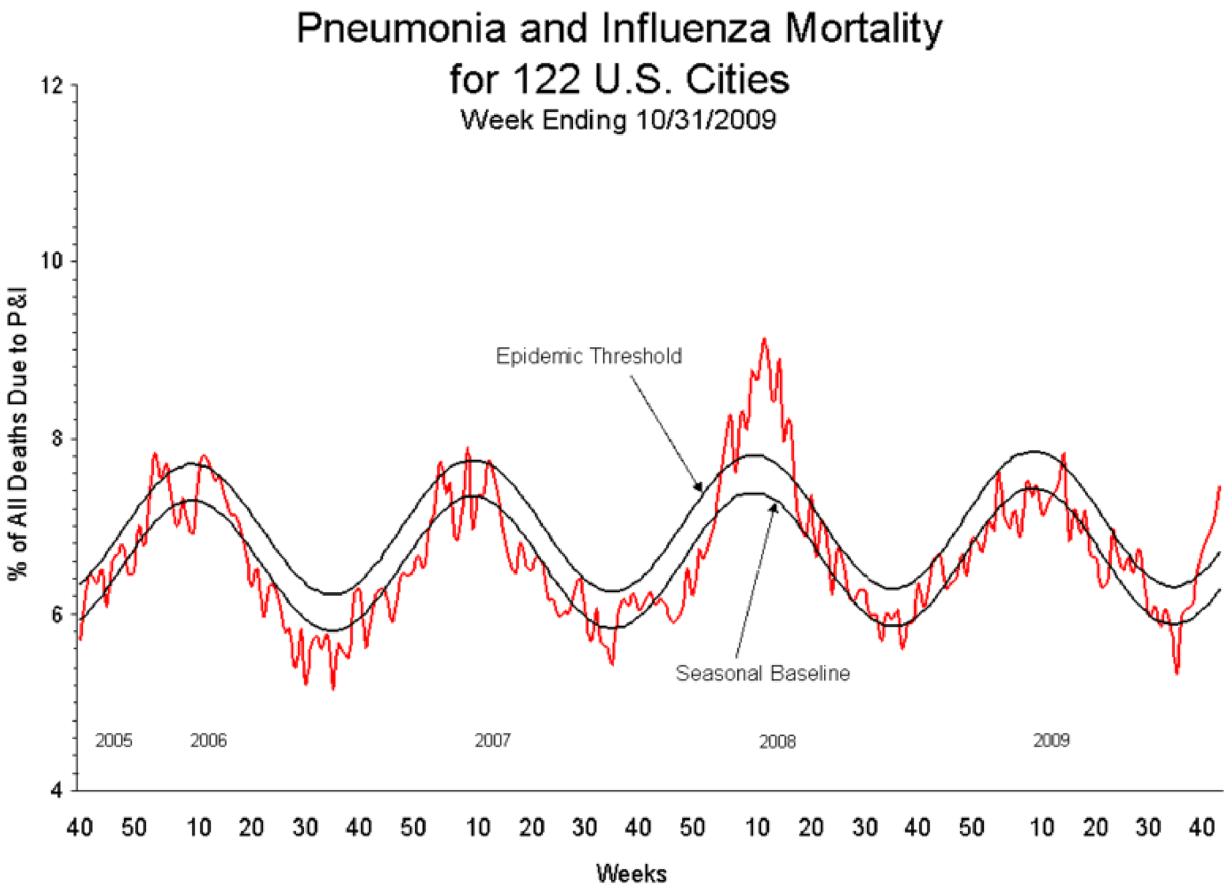


Data sources:
- CSIRO (Commonwealth Scientific and Industrial Research Organisation). 2009. Sea level rise. Accessed November 2009. http://www.cmar.csiro.au/sealevel.
- University of Colorado at Boulder. 2009. Sea level change: 2009 release #2. http://sealevel.colorado.edu.

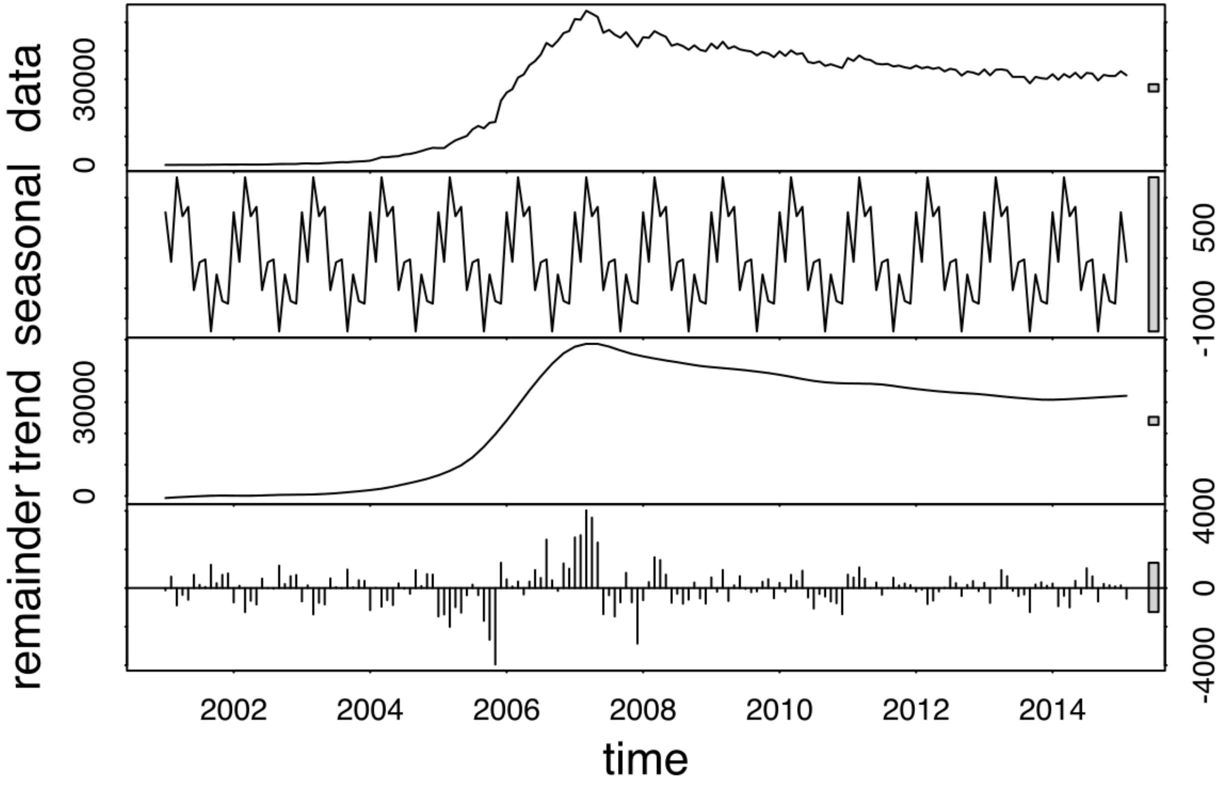
For more information, visit U.S. EPA's "Climate Change Indicators in the United States" at www.epa.gov/climatechange/science/indicators.

DataCumulus Sundog Education

Seasonality



You can have both



Noise

- Some variations are just random in nature
- Seasonality + Trends + Noise = time series
  - Additive model
  - Seasonal variation is constant
- Or sometimes, seasonality * trends * noise – trends sometimes amplify seasonality and noise.
  - Multiplicative model
  - Seasonal variation increases as the trend increases

Amazon Athena

Serverless interactive queries of S3 data

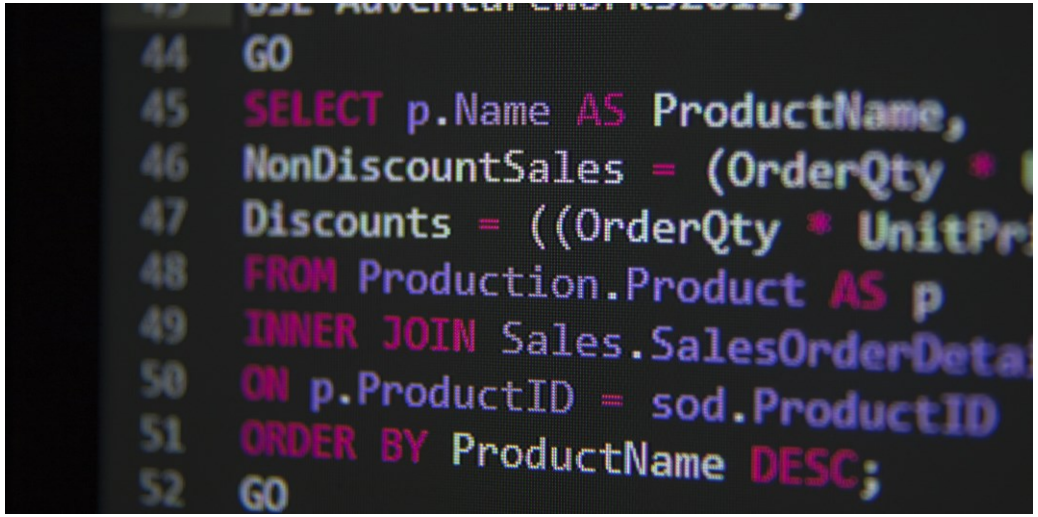
The picture can't be displayed.

What is Athena?

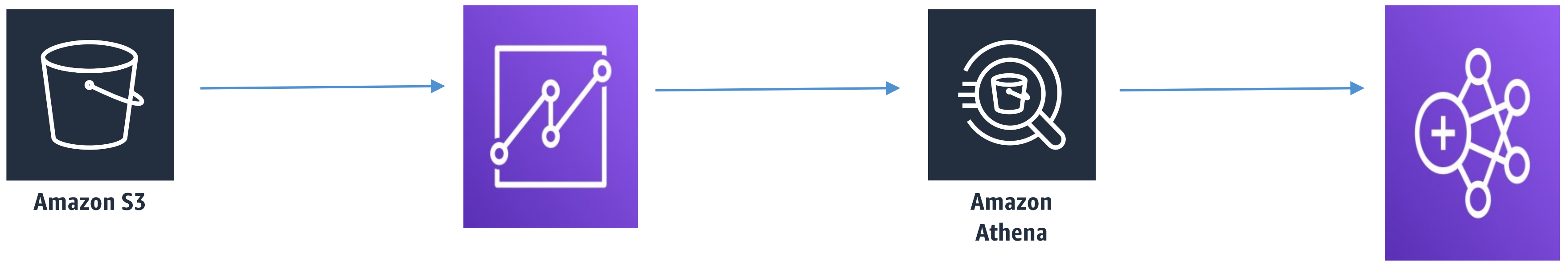
- Interactive query service for S3 (SQL)
  - No need to load data, it stays in S3
- Presto under the hood
- Serverless!
- Supports many data formats
  - CSV (human readable)
  - JSON (human readable)
  - ORC (columnar, splittable)
  - Parquet (columnar, splittable)
  - Avro (splittable)
- Unstructured, semi-structured, or structured

Some examples

- Ad-hoc queries of web logs
- Querying staging data before loading to Redshift
- Analyze CloudTrail / CloudFront / VPC / ELB etc logs in S3
- Integration with Jupyter, Zeppelin, RStudio notebooks
- Integration with QuickSight
- Integration via ODBC / JDBC with other visualization tools



Athena + Glue



Athena cost model

- Pay-as-you-go
  - $5 per TB scanned
  - Successful or cancelled queries count, failed queries do not.
  - No charge for DDL (CREATE/ALTER/DROP etc.)
- Save LOTS of money by using columnar formats
  - ORC, Parquet
  - Save 30-90%, and get better performance
- Glue and S3 have their own charges

DataCumulus Sundog Education

Athena Security

- Access control
  - IAM, ACLs, S3 bucket policies
  - AmazonAthenaFullAccess / AWSQuicksightAthenaAccess
- Encrypt results at rest in S3 staging directory
  - Server-side encryption with S3-managed key (SSE-S3)
  - Server-side encryption with KMS key (SSE-KMS)
  - Client-side encryption with KMS key (CSE-KMS)
- Cross-account access in S3 bucket policy possible
- Transport Layer Security (TLS) encrypts in-transit (between Athena and S3)

Athena anti-patterns

- Highly formatted reports / visualization
  - That's what QuickSight is for
- ETL
  - Use Glue instead

Amazon QuickSight

Business analytics and visualizations in the cloud

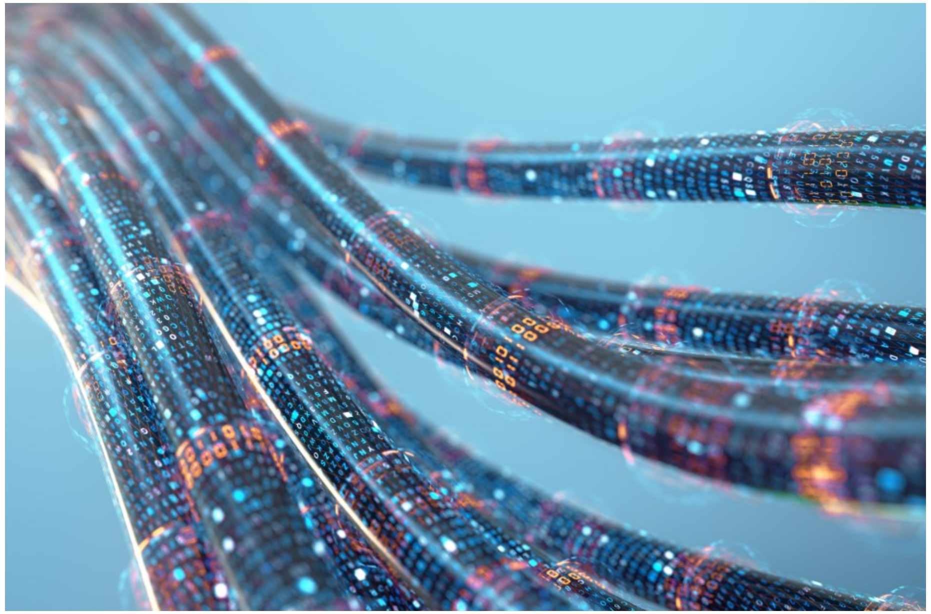
DataCumulus Sundog™ Education

What is QuickSight?

- Fast, easy, cloud-powered business analytics service
- Allows all employees in an organization to:
  - Build visualizations
  - Perform ad-hoc analysis
  - Quickly get business insights from data
  - Anytime, on any device (browsers, mobile)
- Serverless

QuickSight Data Sources

- Redshift
- Aurora / RDS
- Athena
- EC2-hosted databases
- Files (S3 or on-premises)
  - Excel
  - CSV, TSV
  - Common or extended log format
- AWS IoT Analytics
- Data preparation allows limited ETL



SPICE

- Data sets are imported into SPICE
  - Super-fast, Parallel, In-memory Calculation Engine
  - Uses columnar storage, in-memory, machine code generation
  - Accelerates interactive queries on large datasets
- Each user gets 10GB of SPICE
- Highly available / durable
- Scales to hundreds of thousands of users

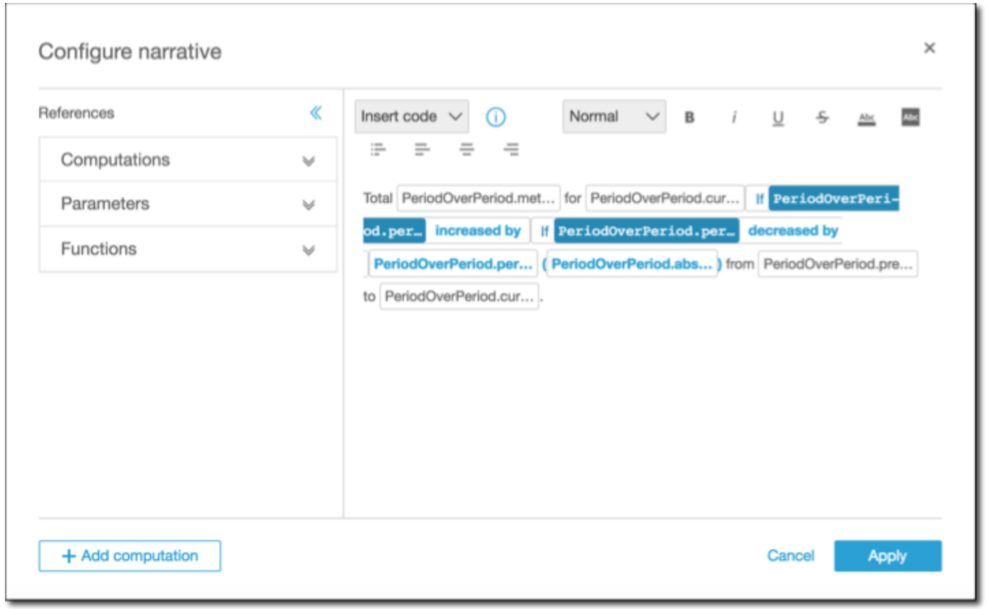
QuickSight Use Cases

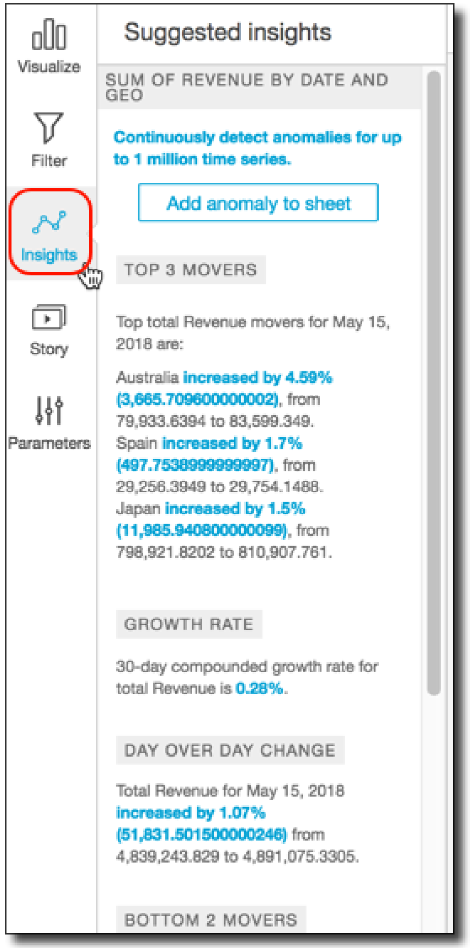
- Interactive ad-hoc exploration / visualization of data
- Dashboards and KPI's
- Analyze / visualize data from:
  - Logs in S3
  - On-premise databases
  - AWS (RDS, Redshift, Athena, S3)
  - SaaS applications, such as Salesforce
  - Any JDBC/ODBC data source

DataCumulus Sundog™ Education

Machine Learning Insights

- Anomaly detection
- Forecasting
- Auto-narratives





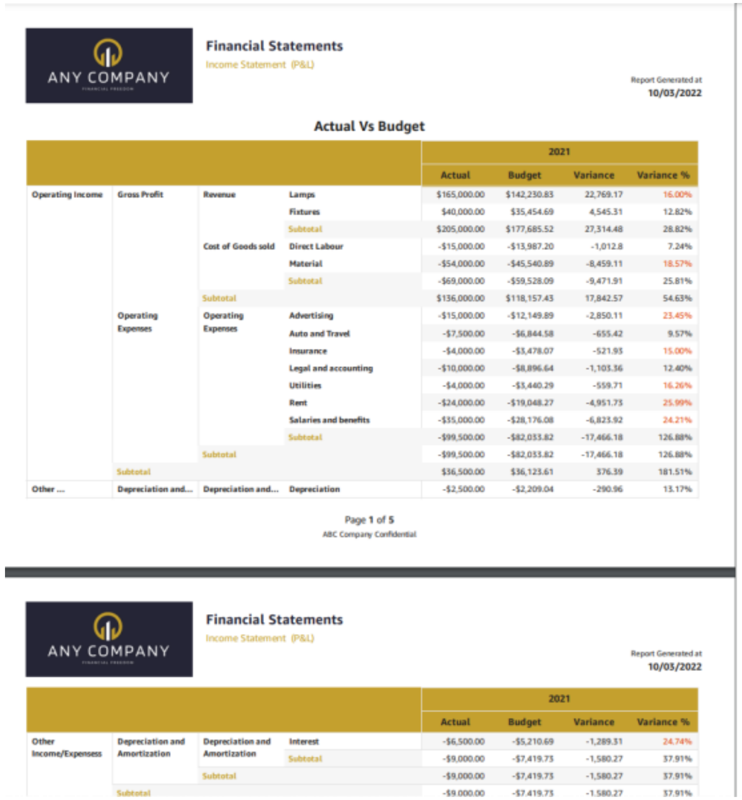
Quicksight Q

- Machine learning-powered
- Answers business questions with Natural Language Processing
  - "What are the top-selling items in Florida?"
- Offered as an add-on for given regions
- Personal training on how to use it is required
- Must set up *topics* associated with *datasets*
  - Datasets and their fields must be NLP-friendly
  - How to handle dates must be defined

DataCumulus Sundog Education

Quicksight Paginated Reports

- Reports designed to be printed
- May span many pages
- Can be based on existing Quicksight dashboards
- New in Nov 2022



ANY COMPANY

Financial Statements

Income Statement (P&L)

Report Generated at
10/03/2022

Actual Vs Budget

				2021			
				Actual	Budget	Variance	Variance %
Operating Income	Gross Profit	Revenue	Lamps	$165,000.00	$142,230.83	22,769.17	16.00%
			Fixtures	$40,000.00	$35,454.69	4,545.31	12.82%
			Subtotal	$205,000.00	$177,685.52	27,314.48	28.82%
		Cost of Goods sold	Direct Labour	-$15,000.00	-$13,987.20	-1,012.8	7.24%
			Material	-$54,000.00	-$45,540.89	-8,459.11	18.57%
			Subtotal	-$69,000.00	-$59,528.09	-9,471.91	25.81%
		Subtotal		$136,000.00	$118,157.43	17,842.57	54.63%
	Operating Expenses	Operating Expenses	Advertising	-$15,000.00	-$12,149.89	-2,850.11	23.45%
			Auto and Travel	-$7,500.00	-$6,844.58	-655.42	9.57%
			Insurance	-$4,000.00	-$3,478.07	-521.93	15.00%
			Legal and accounting	-$10,000.00	-$8,896.64	-1,103.36	12.40%
			Utilities	-$4,000.00	-$3,440.29	-559.71	16.26%
			Rent	-$24,000.00	-$19,048.27	-4,951.73	25.99%
			Salaries and benefits	-$35,000.00	-$28,176.08	-6,823.92	24.21%
			Subtotal	-$99,500.00	-$82,033.82	-17,466.18	126.88%
		Subtotal		-$99,500.00	-$82,033.82	-17,466.18	126.88%
	Subtotal			$36,500.00	$36,123.61	376.39	181.51%
Other ...	Depreciation and...	Depreciation and...	Depreciation	-$2,500.00	-$2,209.04	-290.96	13.17%

Page 1 of 5

ABC Company Confidential

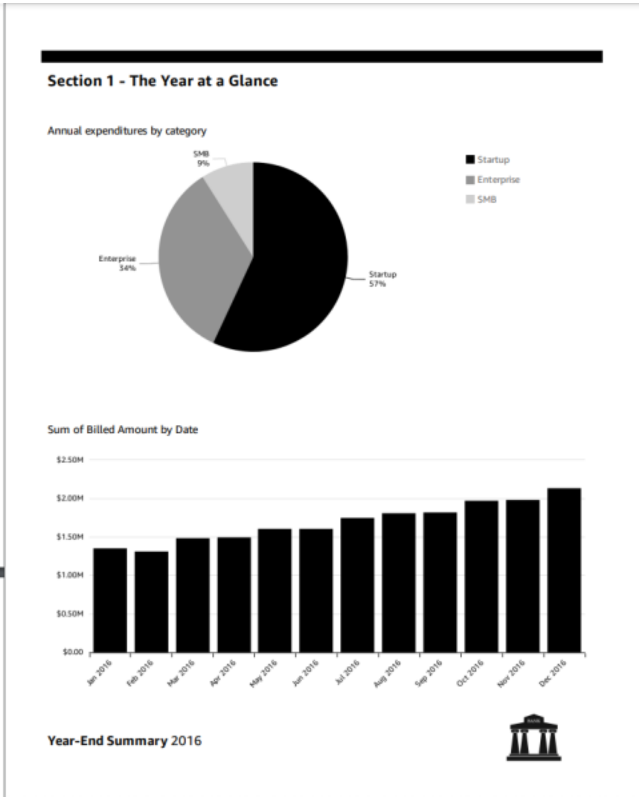
ANY COMPANY

Financial Statements

Income Statement (P&L)

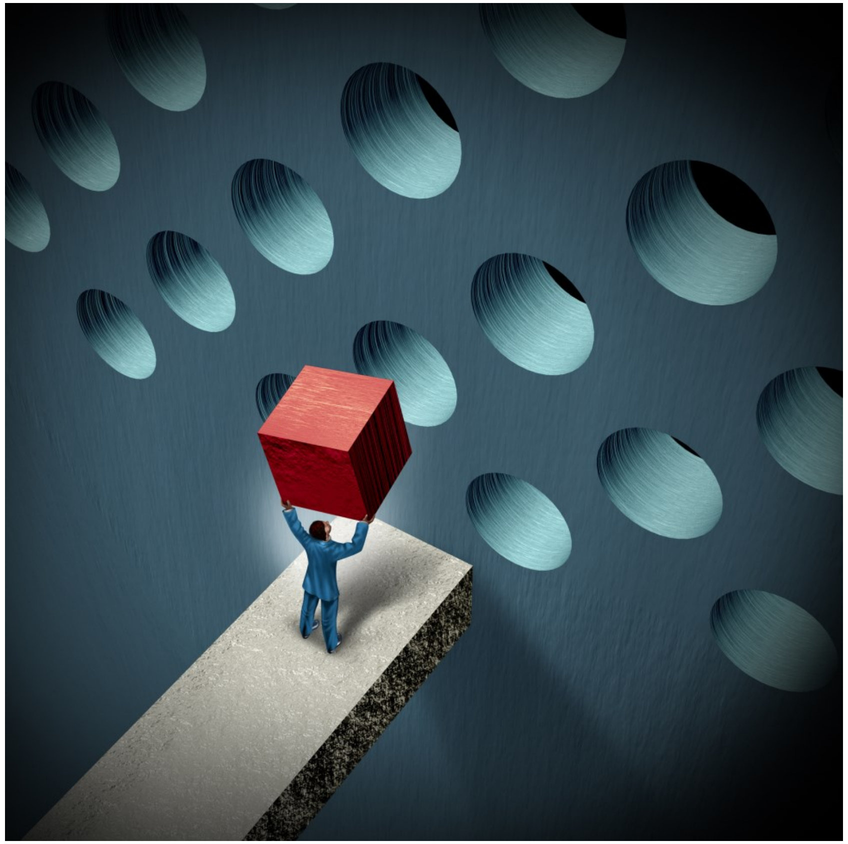
Report Generated at
10/03/2022

				2021			
				Actual	Budget	Variance	Variance %
Other Income/Expenses	Depreciation and Amortization	Depreciation and Amortization	Interest	-$6,500.00	-$5,210.69	-1,289.31	24.74%
			Subtotal	-$9,000.00	-$7,419.73	-1,580.27	37.91%
		Subtotal		-$9,000.00	-$7,419.73	-1,580.27	37.91%
	Subtotal			-$9,000.00	-$7,419.73	-1,580.27	37.91%



QuickSight Anti-Patterns

- ~~Highly formatted canned reports~~
  - ~~QuickSight is for ad-hoc queries, analysis, and visualization~~
  - No longer true with paginated reports!
- ETL
  - Use Glue instead, although QuickSight can do some transformations



QuickSight Security

- Multi-factor authentication on your account
- VPC connectivity
  - Add QuickSight's IP address range to your database security groups
- Row-level security
  - Column-level security too (CLS) – Enterprise edition only
- Private VPC access
  - Elastic Network Interface, AWS Direct Connect

DataCumulus Sundog Education

QuickSight User Management

- Users defined via IAM, or email signup
- SAML-based single sign-on
- Active Directory integration (Enterprise Edition)
- MFA

QuickSight Pricing

- Annual subscription
  - Standard: $9 / user /month
  - Enterprise: $18 / user / month
  - Enterprise with Q: $28 / user / month
- Extra SPICE capacity (beyond 10GB)
  - $0.25 (standard) $0.38 (enterprise) / GB / month
- Month to month
  - Standard: $12 / user / month
  - Enterprise: $24 / user / month
  - Enterprise with Q: $34 / user / month
- Additional charges for paginated reports, alerts & anomaly detection, Q capacity, readers, and reader session capacity.
- Enterprise edition
  - ~~Encryption at rest~~ (now included in standard as well)
  - Microsoft Active Directory integration

DataCumulus Sundog™ Education

QuickSight Dashboards

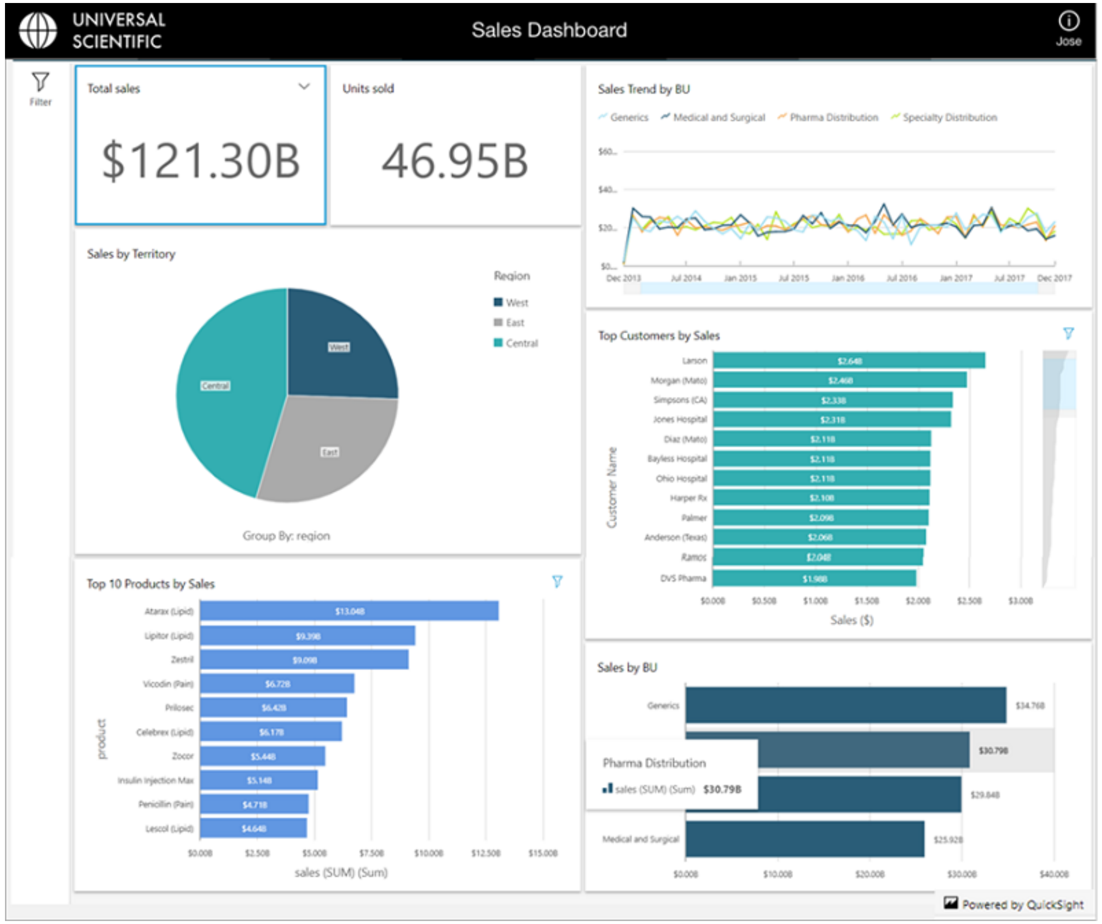


Image: AWS Big Data Blog

DataCumulus Sundog Education

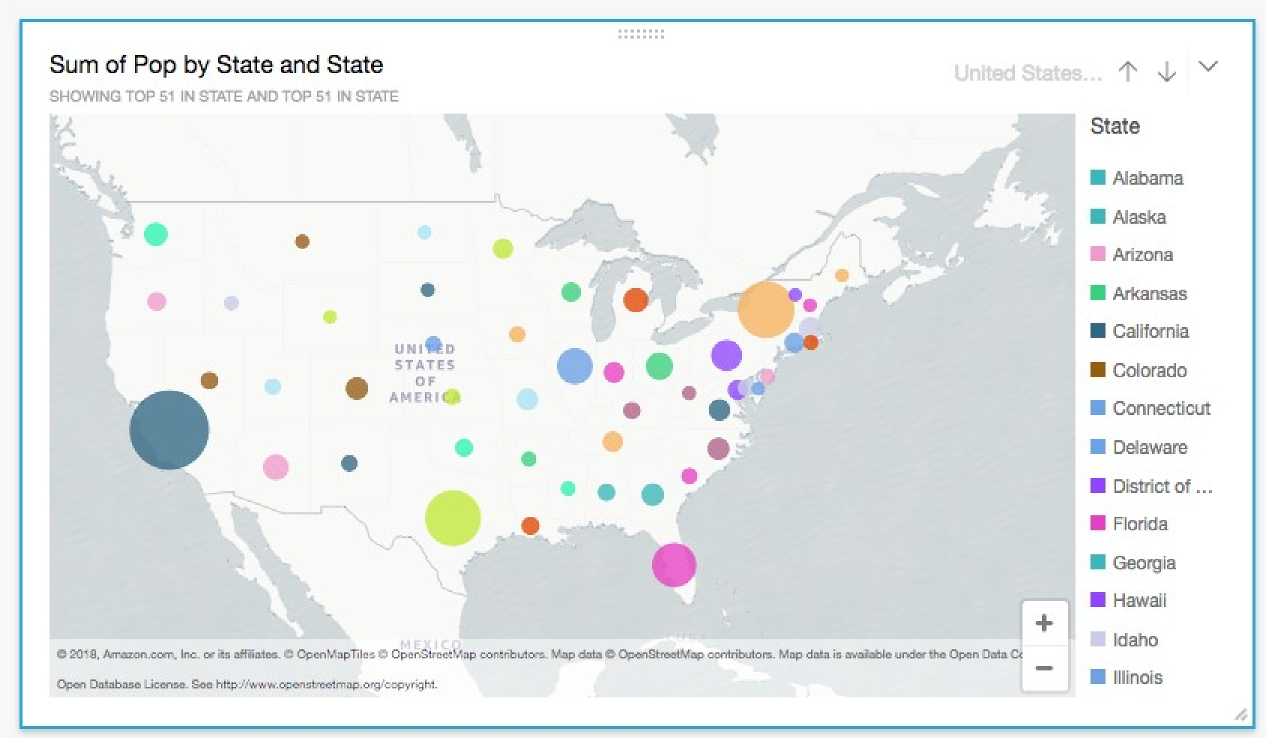
QuickSight Visual Types

- AutoGraph
- Bar Charts
  - For comparison and distribution (histograms)
- Line graphs
  - For changes over time
- Scatter plots, heat maps
  - For correlation
- Pie graphs, tree maps
  - For aggregation
- Pivot tables
  - For tabular data

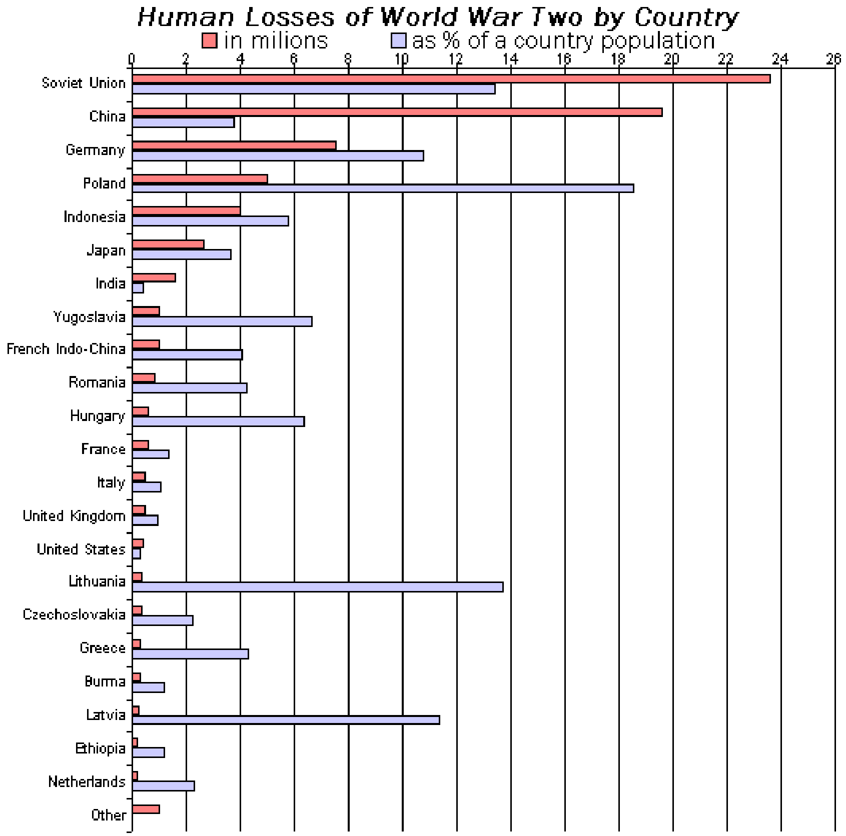
DataCumulus Sundog Education

Additional Visual Types

- KPIs
- Geospatial Charts (maps)
- Donut Charts
- Gauge Charts
- Word Clouds



Bar Charts: Comparison, Distribution



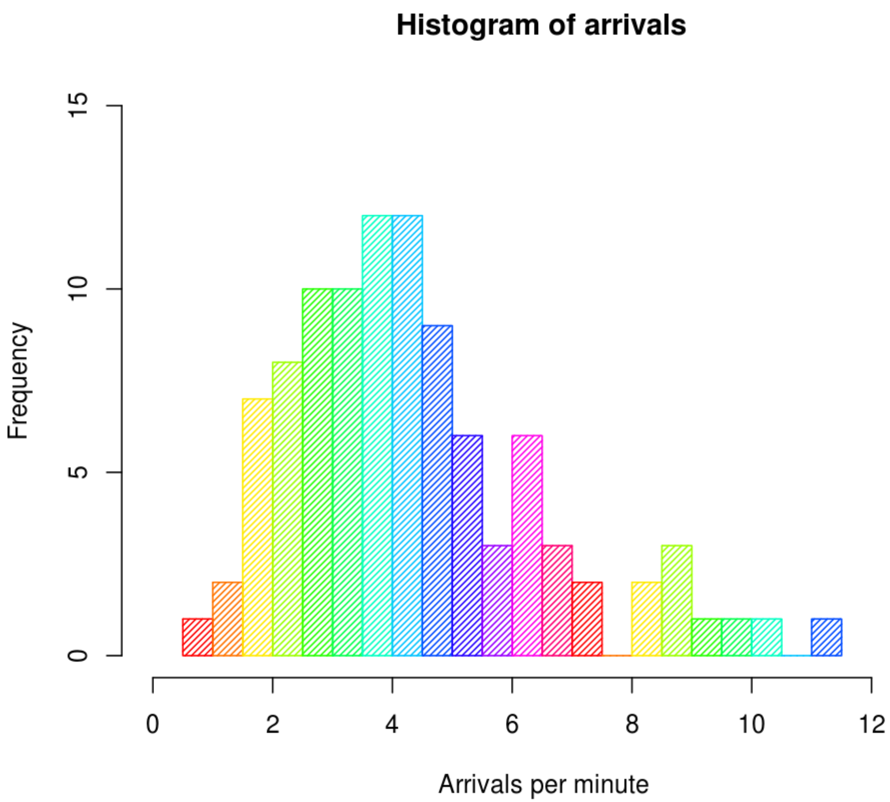
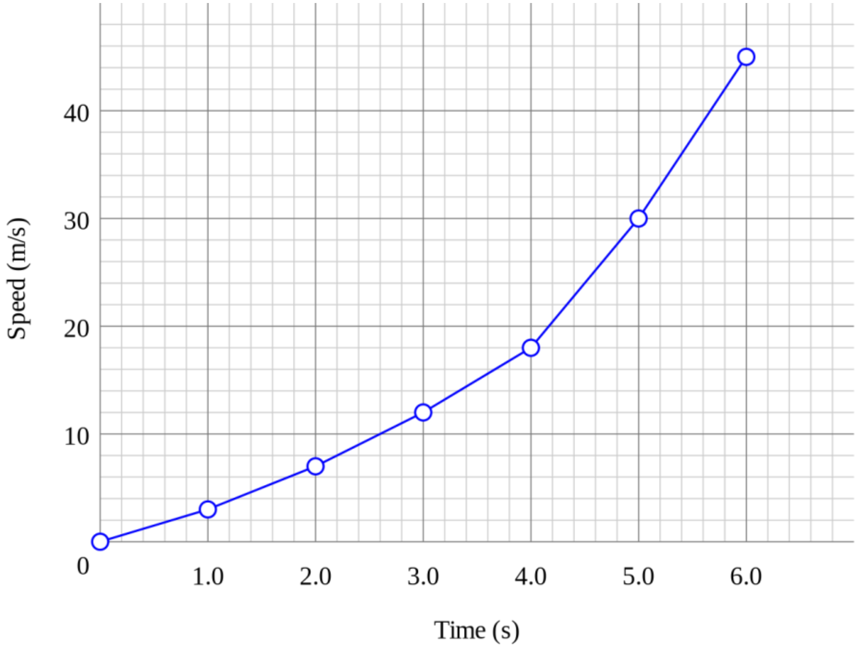
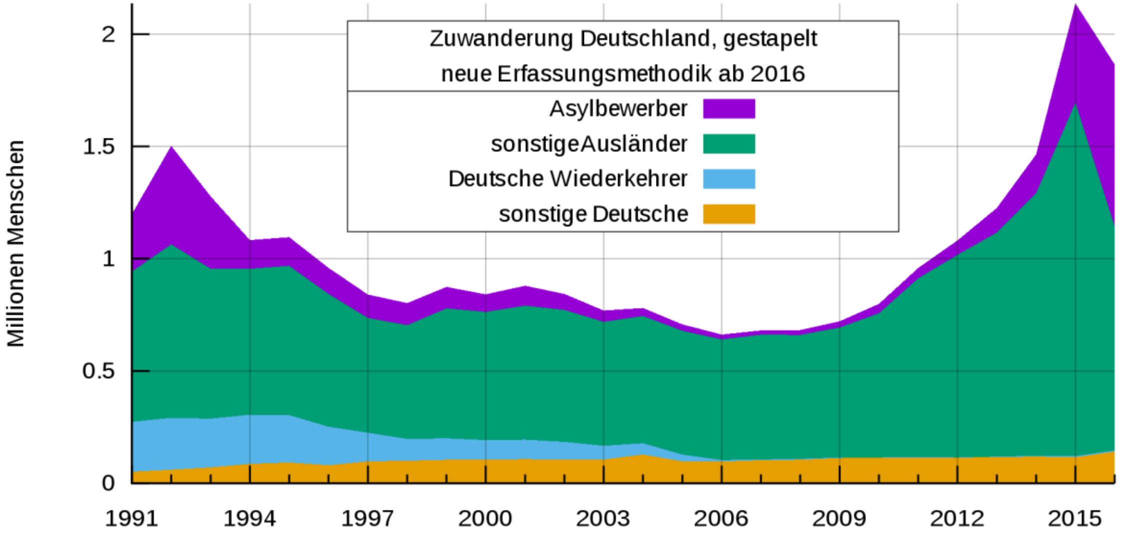


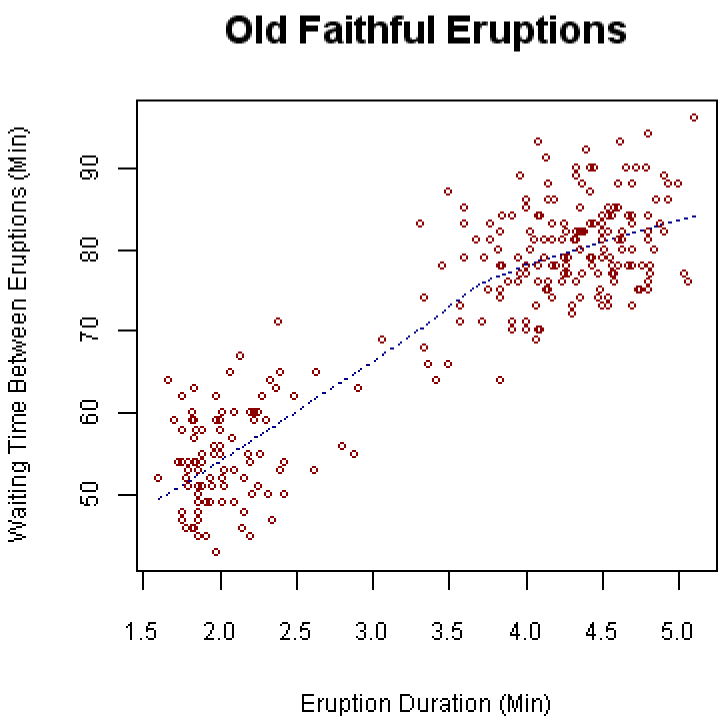
Image: DanielPenfield, Wikipedia CC BY-SA 3.0

Line Charts: Changes over Time



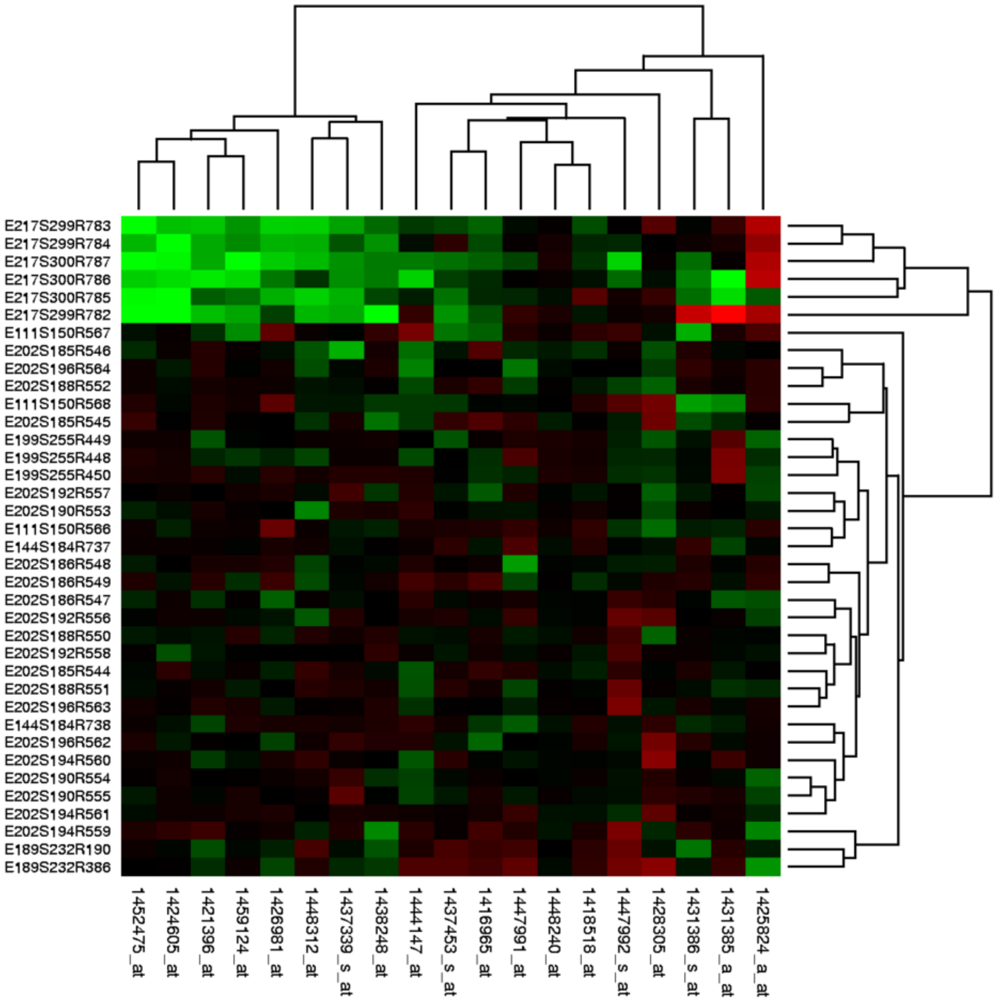


Scatter Plots: Correlation



DataCumulus Sundog Education

Heat Maps: Correlation



DataCumulus Sundog Education

Pie Charts: Aggregation

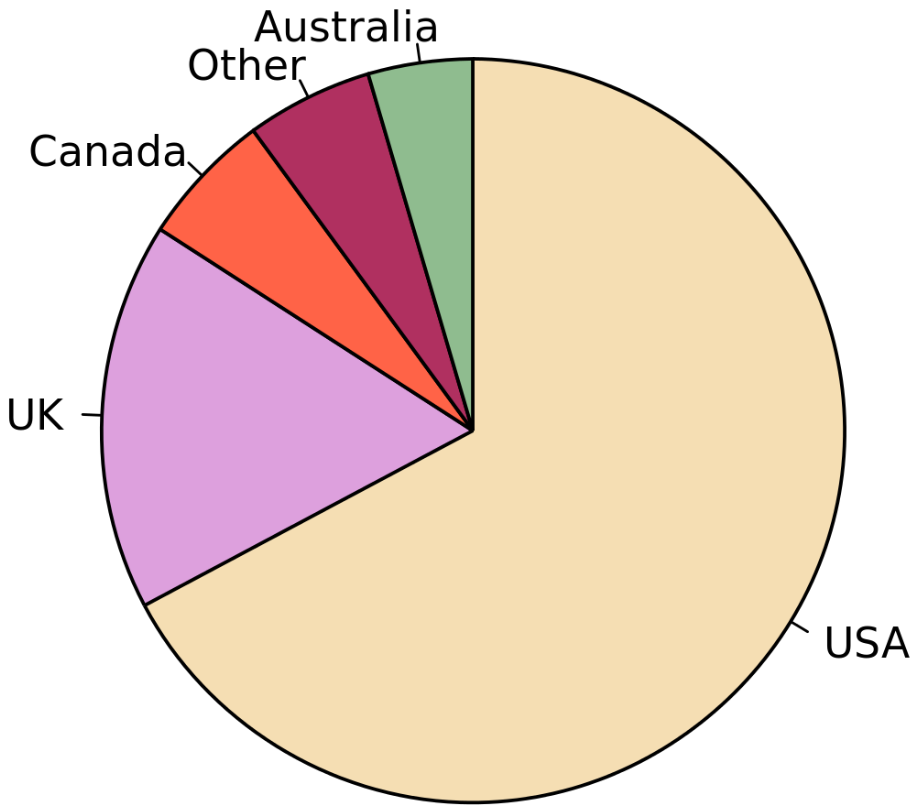
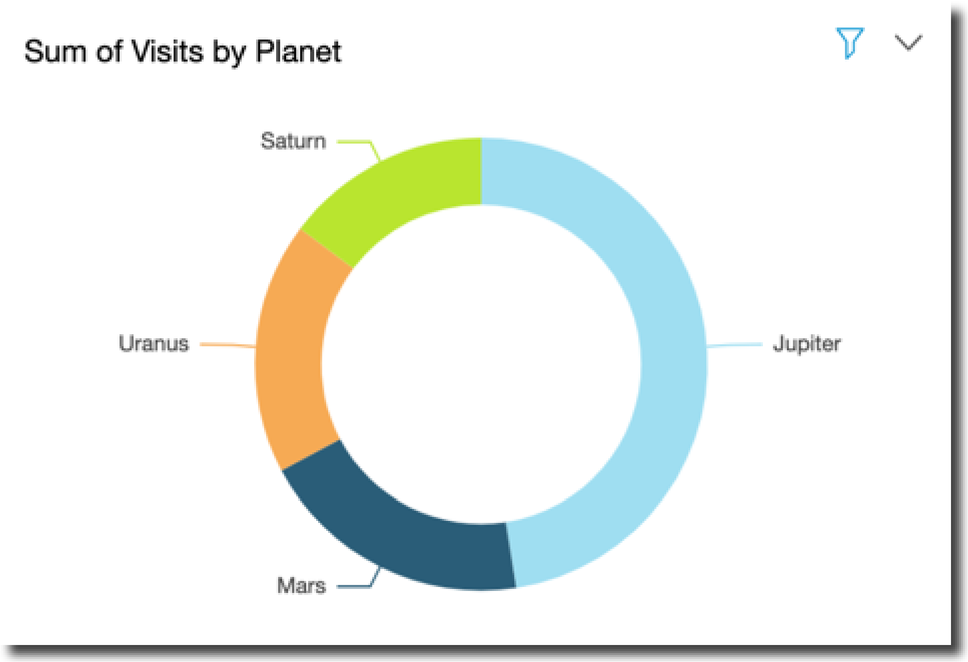
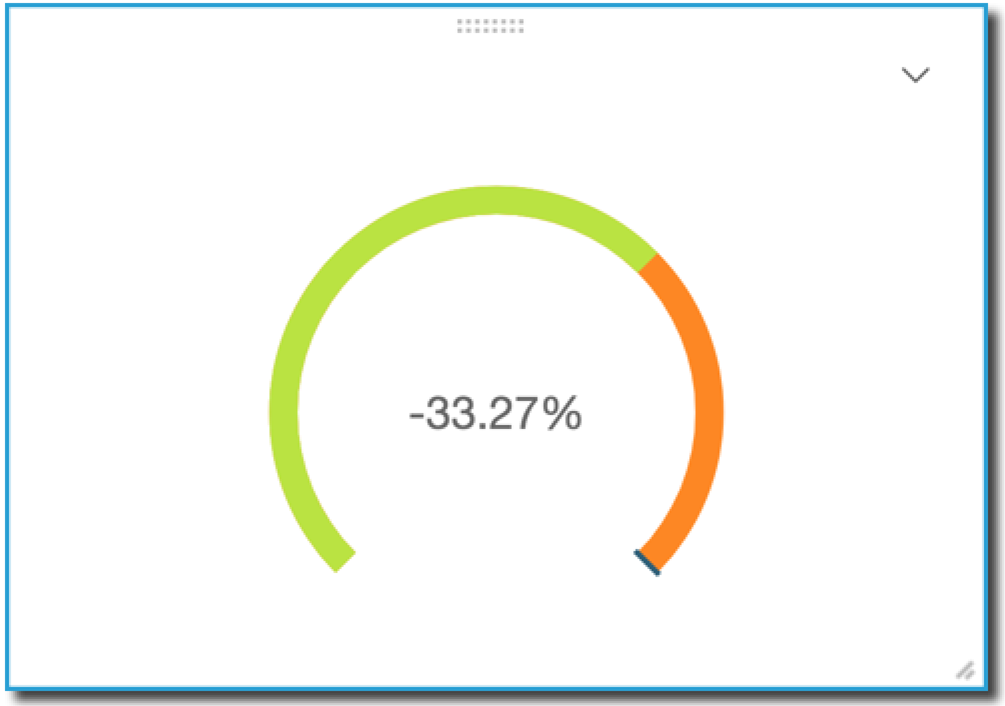


Image: M.W. Toews, Wikipedia, CC BY-SA 4.0

Donut Charts: Percentage of Total Amount



Gauge Charts: Compare values in a measure



Tree Maps: Heirarchical Aggregation

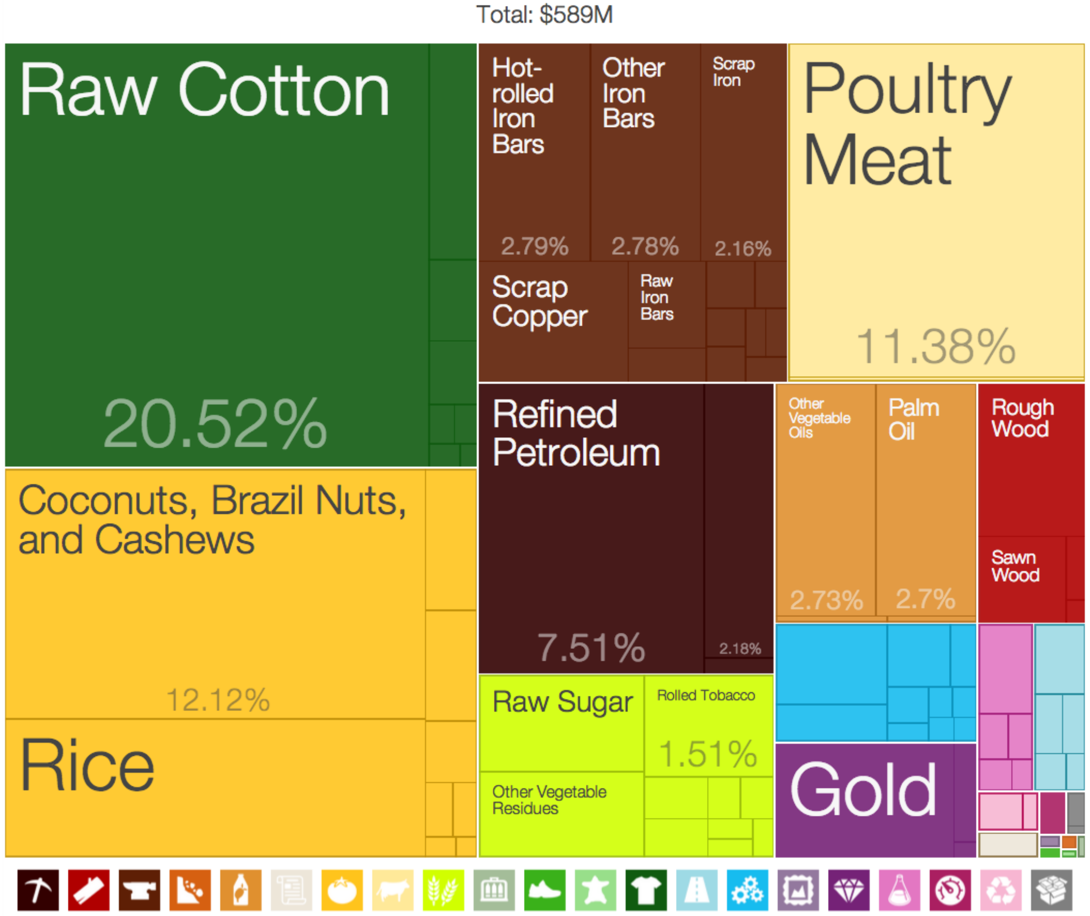


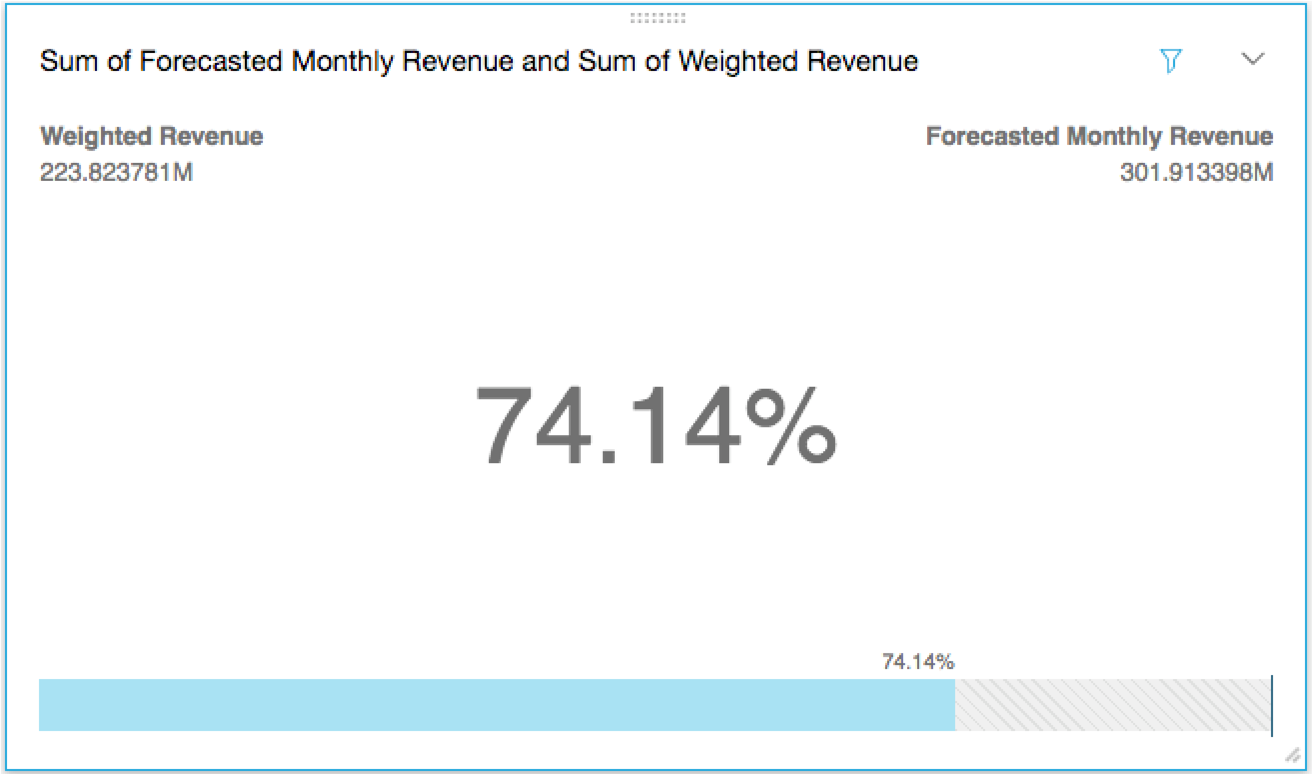
Image: Harvard-MIT Observatory of Economic Complexity, Wikipedia, CC-BY-SA 3.0

Pivot Tables: Tabular Data

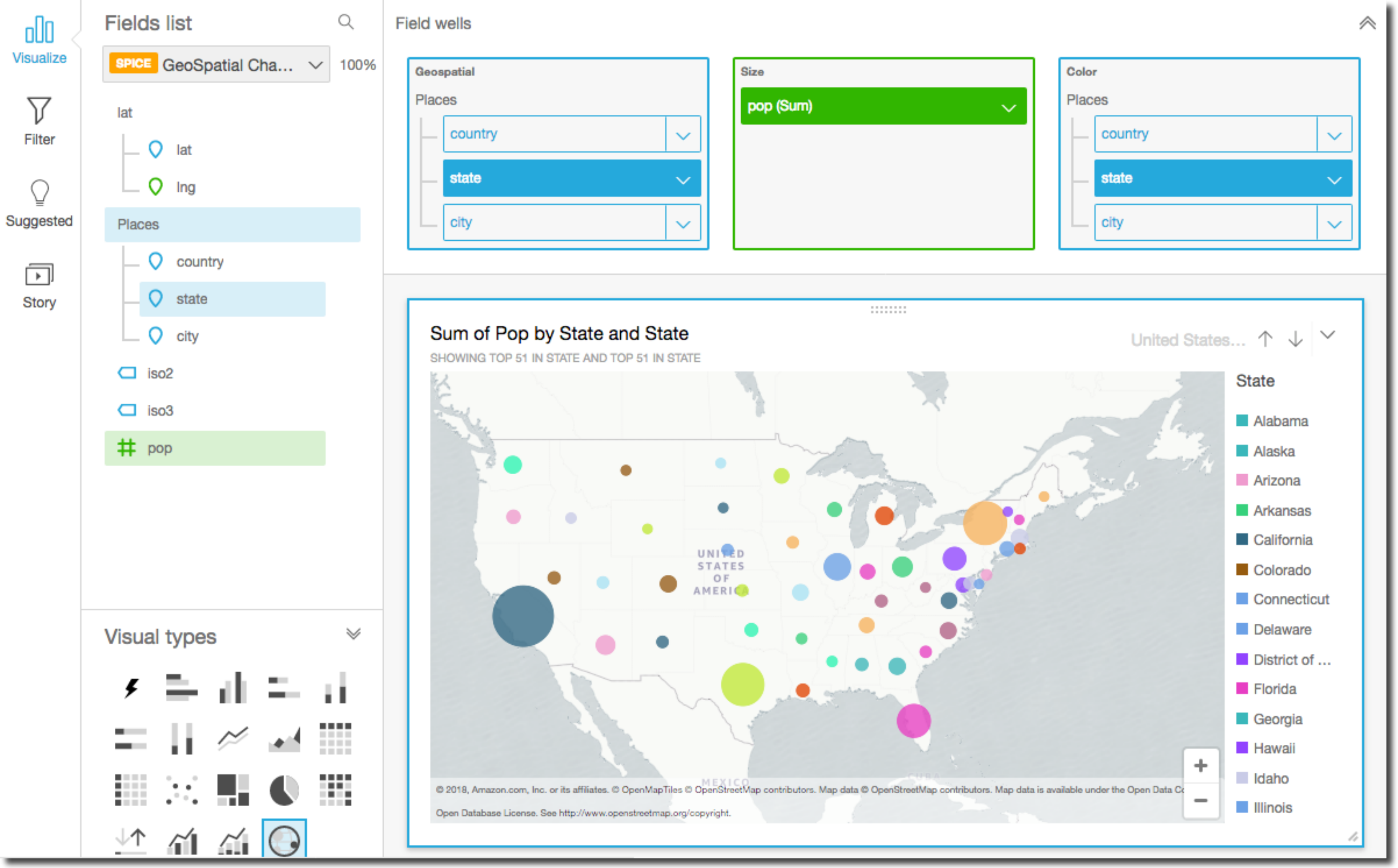
	A	B	C	D	E	F	G
1	**Region**	**Gender**	**Style**	**Ship Date**	**Units**	**Price**	**Cost**
2	East	Boy	Tee	1/31/2005	12	11.04	10.42
3	East	Boy	Golf	1/31/2005	12	13	12.6
4	East	Boy	Fancy	1/31/2005	12	11.96	11.74
5	East	Girl	Tee	1/31/2005	10	11.27	10.56
6	East	Girl	Golf	1/31/2005	10	12.12	11.95
7	East	Girl	Fancy	1/31/2005	10	13.74	13.33
8	West	Boy	Tee	1/31/2005	11	11.44	10.94
9	West	Boy	Golf	1/31/2005	11	12.63	11.73
10	West	Boy	Fancy	1/31/2005	11	12.06	11.51
11	West	Girl	Tee	1/31/2005	15	13.42	13.29
12	West	Girl	Golf	1/31/2005	15	11.48	10.67

Sum of Units	Ship Date					
Region	1/31/2005	2/28/2005	3/31/2005	4/30/2005	5/31/2005	6/30/2005
East	66	80	102	116	127	125
North	96	117	138	151	154	156
South	123	141	157	178	191	202
West	78	97	117	136	150	157
(blank)						
Grand Total	363	435	514	581	622	640

KPI's: compare key value to its target value

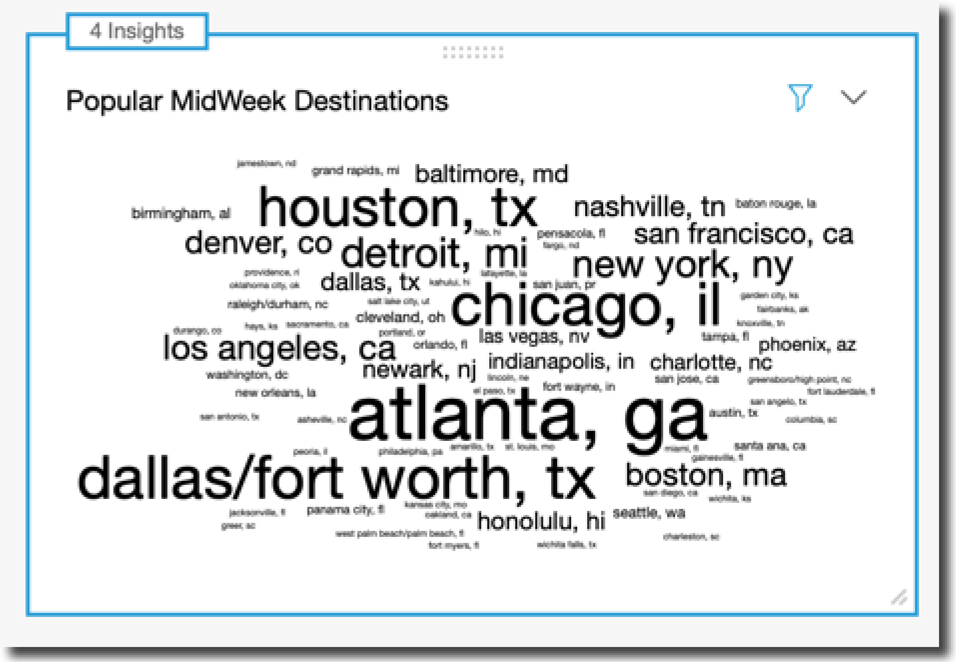


Geospatial Charts



DataCumulus Sundog Education

Word Clouds: word or phrase frequency



EMR

Elastic MapReduce

DataCumulus Sundog™ Education

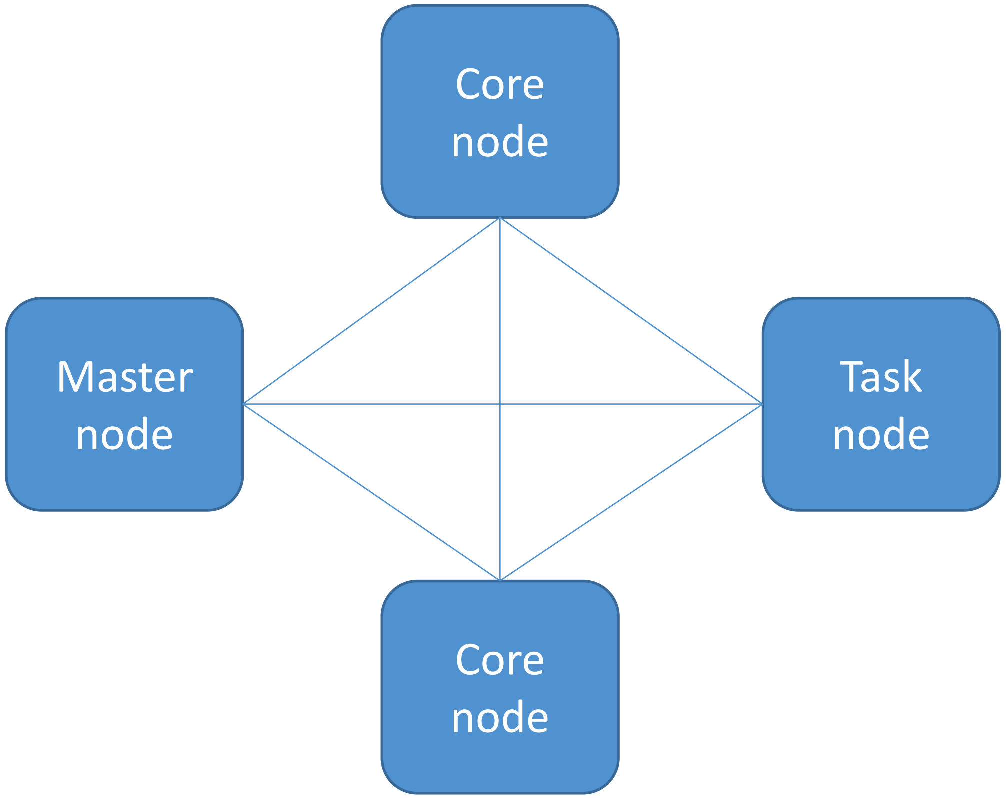
What is EMR?

- Elastic MapReduce
- Managed Hadoop framework on EC2 instances
- Includes Spark, HBase, Presto, Flink, Hive & more
- EMR Notebooks
- Several integration points with AWS

Amazon EMR

DataCumulus Sundog™ Education

An EMR Cluster



- **Master node**: manages the cluster
  - Single EC2 instance
- **Core node**: Hosts HDFS data and runs tasks
  - Can be scaled up & down, but with some risk
- **Task node**: Runs tasks, does not host data
  - No risk of data loss when removing
  - Good use of **spot instances**

EMR Usage

- Transient vs Long-Running Clusters
  - Can spin up task nodes using Spot instances for temporary capacity
  - Can use reserved instances on long-running clusters to save $
- Connect directly to master to run jobs
- Submit ordered steps via the console
- EMR Serverless lets AWS scale your nodes automatically

EMR / AWS Integration

- Amazon EC2 for the instances that comprise the nodes in the cluster
- Amazon VPC to configure the virtual network in which you launch your instances
- Amazon S3 to store input and output data
- Amazon CloudWatch to monitor cluster performance and configure alarms
- AWS IAM to configure permissions
- AWS CloudTrail to audit requests made to the service
- AWS Data Pipeline to schedule and start your clusters

EMR Storage

- HDFS
- EMRFS: access S3 as if it were HDFS
  - **EMRFS Consistent View** – Optional for S3 consistency
  - Uses DynamoDB to track consistency
- Local file system
- EBS for HDFS

DataCumulus Sundog™ Education

EMR promises

- EMR charges by the hour
  - Plus EC2 charges
- Provisions new nodes if a core node fails
- Can add and remove tasks nodes on the fly
- Can resize a running cluster's core nodes

DataCumulus Sundog™ Education

So… what's Hadoop?

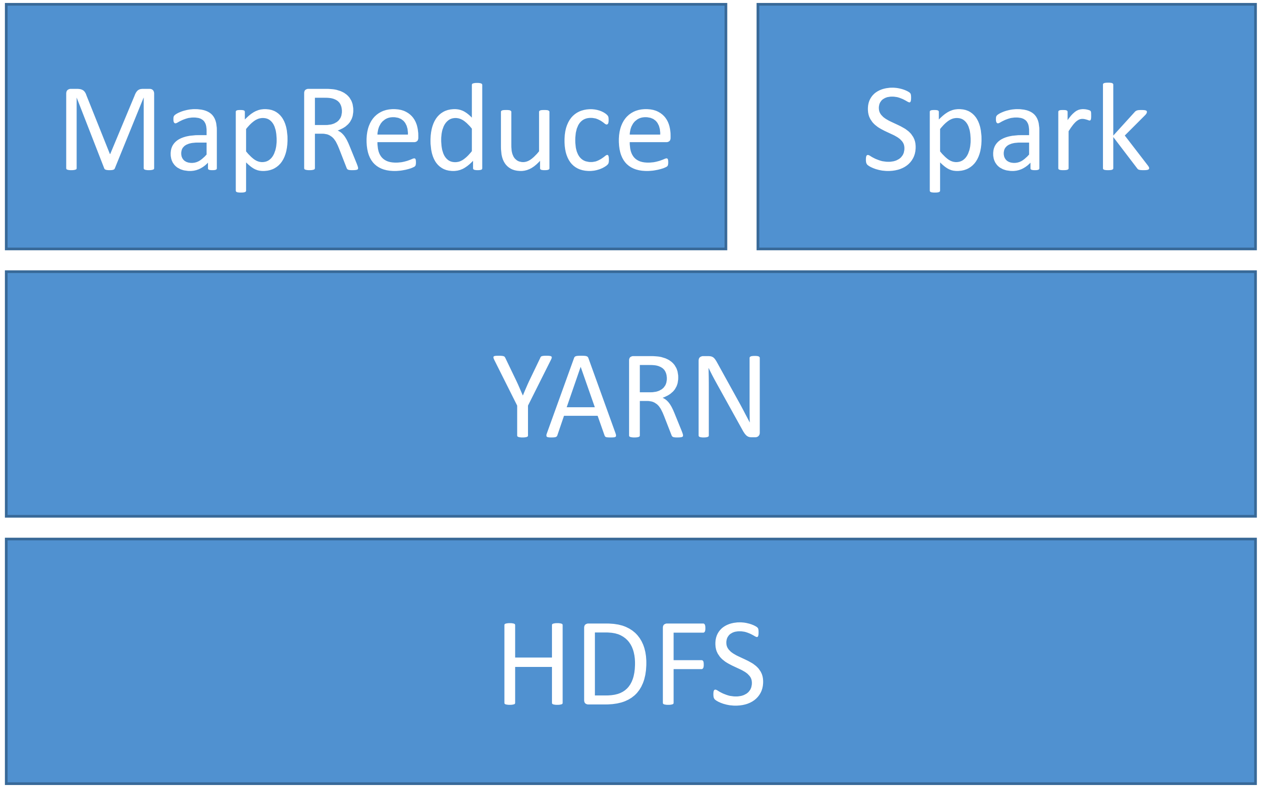
MapReduce

YARN

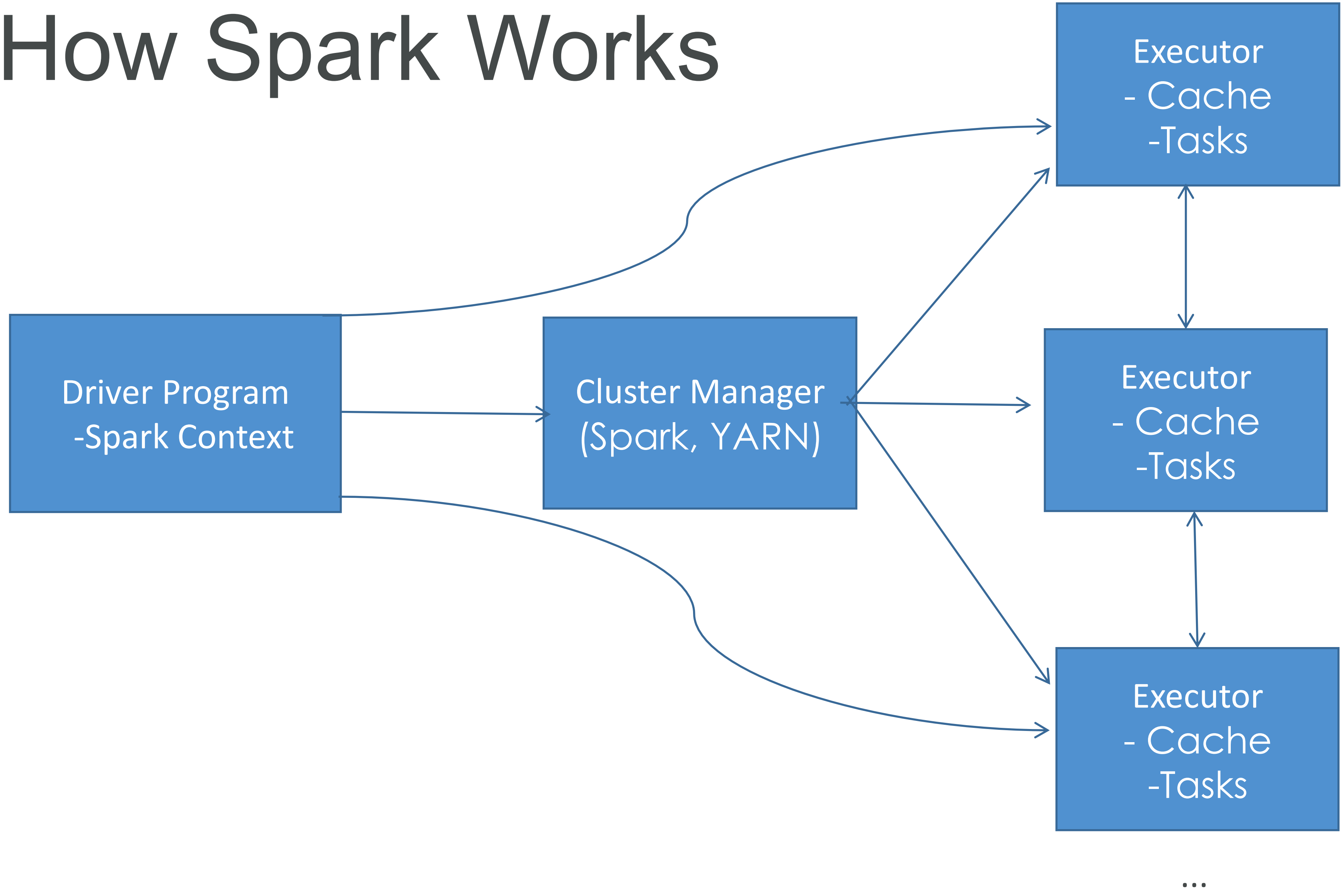
HDFS

DataCumulus Sundog™ Education

Apache Spark



How Spark Works



Spark Components

Spark Streaming

Spark SQL

MLLib

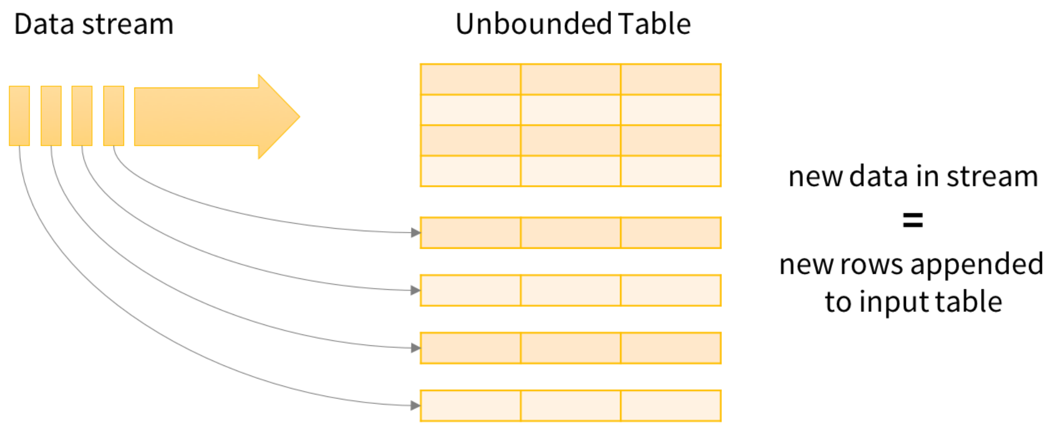
GraphX

SPARK CORE

Spark MLLib

- Classification: logistic regression, naïve Bayes
- Regression
- Decision trees
- Recommendation engine (ALS)
- Clustering (K-Means)
- LDA (topic modeling)
- ML workflow utilities (pipelines, feature transformation, persistence)
- SVD, PCA, statistics

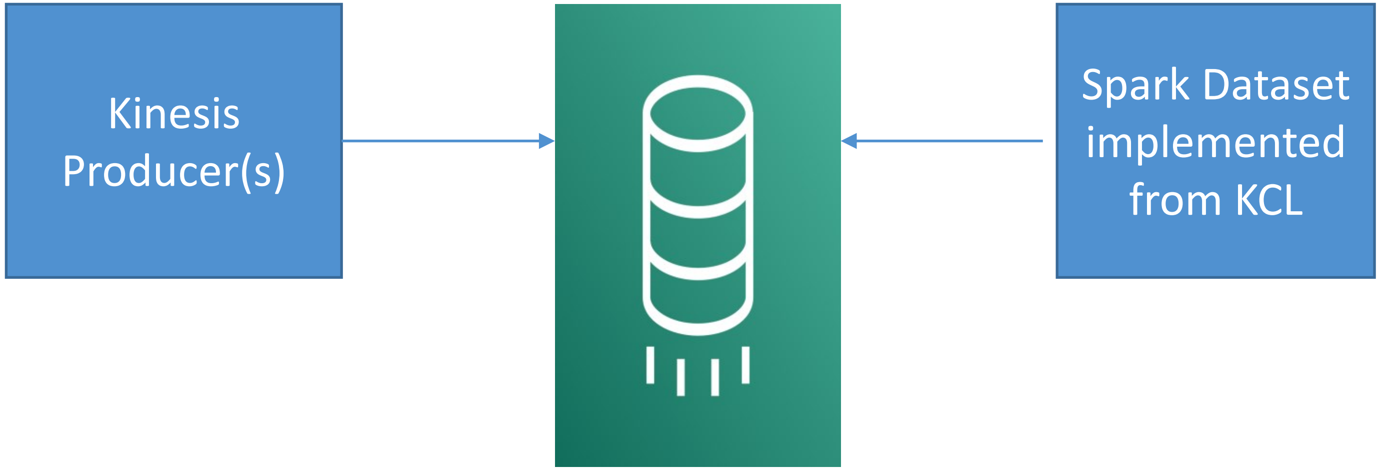
Spark Structured Streaming



Data stream as an unbounded Input Table

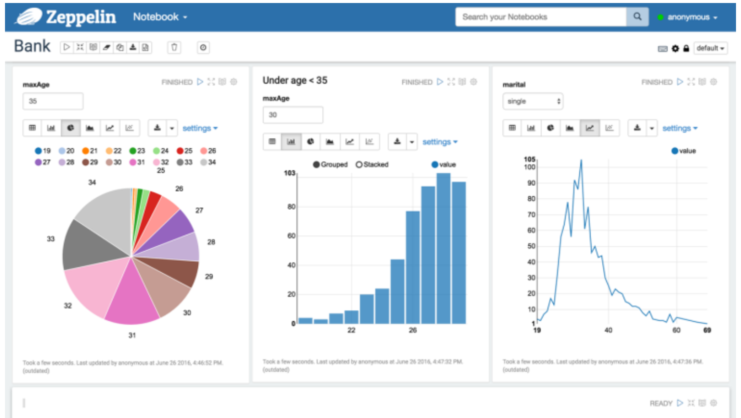
```
val inputDF = spark.readStream.json("s3://logs")
inputDF.groupBy($"action", window($"time", "1 hour")).count()
        .writeStream.format("jdbc").start("jdbc:mysql//...")
```

Spark Streaming + Kinesis



Zeppelin + Spark

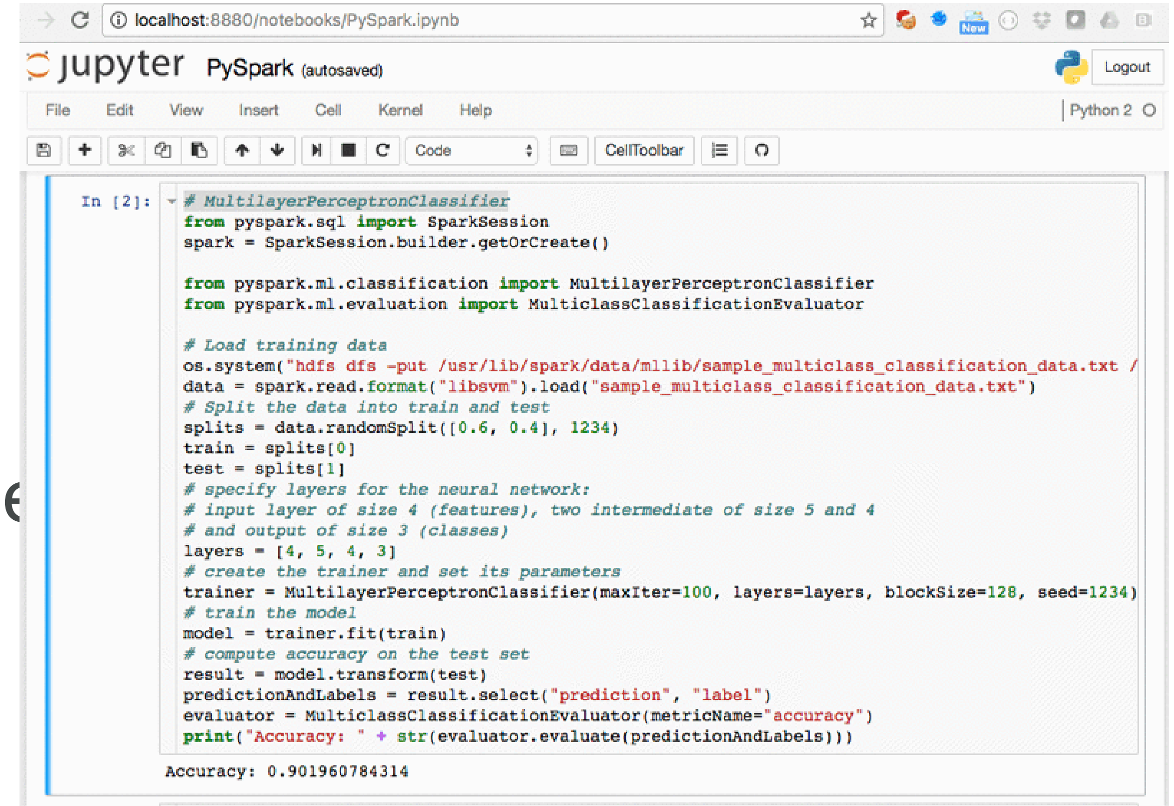
- Can run Spark code interactively (like you can in the Spark shell)
  - This speeds up your development cycle
  - And allows easy experimentation and exploration of your big data
- Can execute SQL queries directly against SparkSQL
- Query results may be visualized in charts and graphs
- Makes Spark feel more like a data science tool!



DataCumulus Sundog Education

EMR Notebook

- Similar concept to Zeppelin, with more AWS integration
- Notebooks backed up to S3
- Provision clusters from the notebook!
- Hosted inside a VPC
- Accessed only via AWS console



EMR Security

- IAM policies
- Kerberos
- SSH
- IAM roles
- Security configurations may be specified for Lake Formation
- Native integration with Apache Ranger
  - For data security on Hadoop / Hive

EMR: Choosing Instance Types

- Master node:
  - m4.large if < 50 nodes, m4.xlarge if > 50 nodes
- Core & task nodes:
  - m4.large is usually good
  - If cluster waits a lot on external dependencies (i.e. a web crawler), t2.medium
  - Improved performance: m4.xlarge
  - Computation-intensive applications: high CPU instances
  - Database, memory-caching applications: high memory instances
  - Network / CPU-intensive (NLP, ML) – cluster computer instances
  - Accelerated Computing / AI – GPU instances (g3, g4, p2, p3)
- Spot instances
  - Good choice for task nodes
  - Only use on core & master if you're testing or very cost-sensitive; you're risking partial data loss

Feature Engineering

What is feature engineering?

- Applying your knowledge of the data – and the model you're using - to create better features to train your model with.
  - Which features should I use?
  - Do I need to transform these features in some way?
  - How do I handle missing data?
  - Should I create new features from the existing ones?
- You can't just throw in raw data and expect good results
- This is the art of machine learning; where expertise is applied
- "Applied machine learning is basically feature engineering" – Andrew Ng

DataCumulus Sundog Education

The Curse of Dimensionality

- Too many features can be a problem – leads to sparse data
- Every feature is a new dimension
- Much of feature engineering is selecting the features most relevant to the problem at hand
  - This often is where domain knowledge comes into play
- Unsupervised dimensionality reduction techniques can also be employed to distill many features into fewer features
  - PCA
  - K-Means

DataCumulus Sundog Education

Imputing Missing Data: Mean Replacement

- Replace missing values with the mean value from the rest of the column (columns, not rows! A column represents a single feature; it only makes sense to take the mean from other samples of the same feature.)
- Fast & easy, won't affect mean or sample size of overall data set
- Median may be a better choice than mean when outliers are present
- But it's generally pretty terrible.
  - Only works on column level, misses correlations between features
  - Can't use on categorical features (imputing with most frequent value can work in this case, though)
  - Not very accurate

In [3]:
```
import pandas as pd

masses_data = pd.read_csv('mammographic_masses.data.txt', 
masses_data.head()
```

Out[3]:

	BI-RADS	age	shape	margin	density	severity
0	5.0	67.0	3.0	5.0	3.0	1
1	4.0	43.0	1.0	1.0	NaN	1
2	5.0	58.0	4.0	5.0	3.0	1
3	4.0	28.0	1.0	1.0	3.0	0
4	5.0	74.0	1.0	5.0	NaN	1

In [6]:
```
mean_imputed = masses_data.fillna(masses_data.mean())
mean_imputed.head()
```

Out[6]:

	BI-RADS	age	shape	margin	density	severity
0	5.0	67.0	3.0	5.0	3.000000	1
1	4.0	43.0	1.0	1.0	2.910734	1
2	5.0	58.0	4.0	5.0	3.000000	1
3	4.0	28.0	1.0	1.0	3.000000	0
4	5.0	74.0	1.0	5.0	2.910734	1

Imputing Missing Data: Dropping

- If not many rows contain missing data…
  - …and dropping those rows doesn't bias your data…
  - …and you don't have a lot of time…
  - …maybe it's a reasonable thing to do.
- But, it's never going to be the right answer for the "best" approach.
- Almost anything is better. Can you substitute another similar field perhaps? (i.e., review summary vs. full text)

In [3]:
```
import pandas as pd

masses_data = pd.read_csv('mammographic_masses.data.
masses_data.head()
```

Out[3]:

	BI-RADS	age	shape	margin	density	severity
0	5.0	67.0	3.0	5.0	3.0	1
1	4.0	43.0	1.0	1.0	NaN	1
2	5.0	58.0	4.0	5.0	3.0	1
3	4.0	28.0	1.0	1.0	3.0	0
4	5.0	74.0	1.0	5.0	NaN	1

In [7]:
```
mean_imputed = masses_data.dropna()
mean_imputed.head()
```

Out[7]:

	BI-RADS	age	shape	margin	density	severity
0	5.0	67.0	3.0	5.0	3.0	1
2	5.0	58.0	4.0	5.0	3.0	1
3	4.0	28.0	1.0	1.0	3.0	0
8	5.0	57.0	1.0	5.0	3.0	1
10	5.0	76.0	1.0	4.0	3.0	1

Imputing Missing Data: Machine Learning

- KNN: Find K “nearest” (most similar) rows and average their values
  - Assumes numerical data, not categorical
  - There are ways to handle categorical data (Hamming distance), but categorical data is probably better served by…
- Deep Learning
  - Build a machine learning model to impute data for your machine learning model!
  - Works well for categorical data. Really well. But it’s complicated.
- Regression
  - Find linear or non-linear relationships between the missing feature and other features
  - Most advanced technique: MICE (Multiple Imputation by Chained Equations)

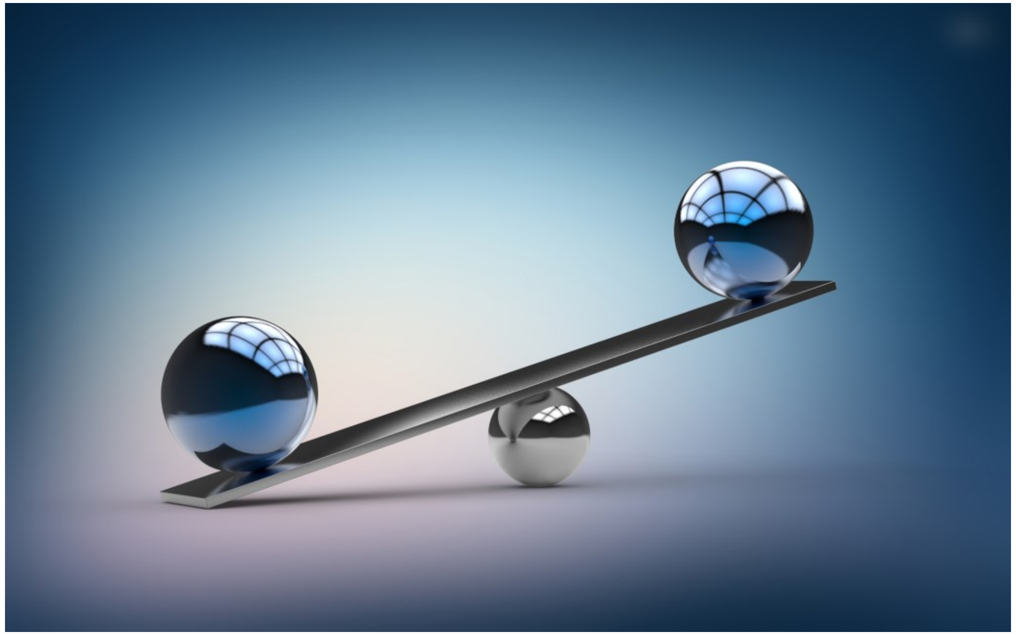
Imputing Missing Data: Just Get More Data

- What's better than imputing data? Getting more real data!
- Sometimes you just have to try harder or collect more data

Handling Unbalanced Data

What is unbalanced data?

- Large discrepancy between "positive" and "negative" cases
  - i.e., fraud detection. Fraud is rare, and most rows will be not-fraud
  - Don't let the terminology confuse you; "positive" doesn't mean "good"
    - It means the thing you're testing for is what happened.
    - If your machine learning model is made to detect fraud, then fraud is the positive case.
- Mainly a problem with neural networks



Oversampling

- Duplicate samples from the minority class
- Can be done at random

Undersampling

- Instead of creating more positive samples, remove negative ones
- Throwing data away is usually not the right answer
  - Unless you are specifically trying to avoid “big data” scaling issues

SMOTE

- Synthetic Minority Over-sampling TEchnique
- Artificially generate new samples of the minority class using nearest neighbors
  - Run K-nearest-neighbors of each sample of the minority class
  - Create a new sample from the KNN result (mean of the neighbors)
- Both generates new samples and undersamples majority class
- Generally better than just oversampling

Adjusting thresholds

- When making predictions about a classification (fraud / not fraud), you have some sort of threshold of probability at which point you'll flag something as the positive case (fraud)
- If you have too many false positives, one way to fix that is to simply increase that threshold.
  - Guaranteed to reduce false positives
  - But, could result in more false negatives

Handling Outliers

Variance measures how “spread-out” the data is.

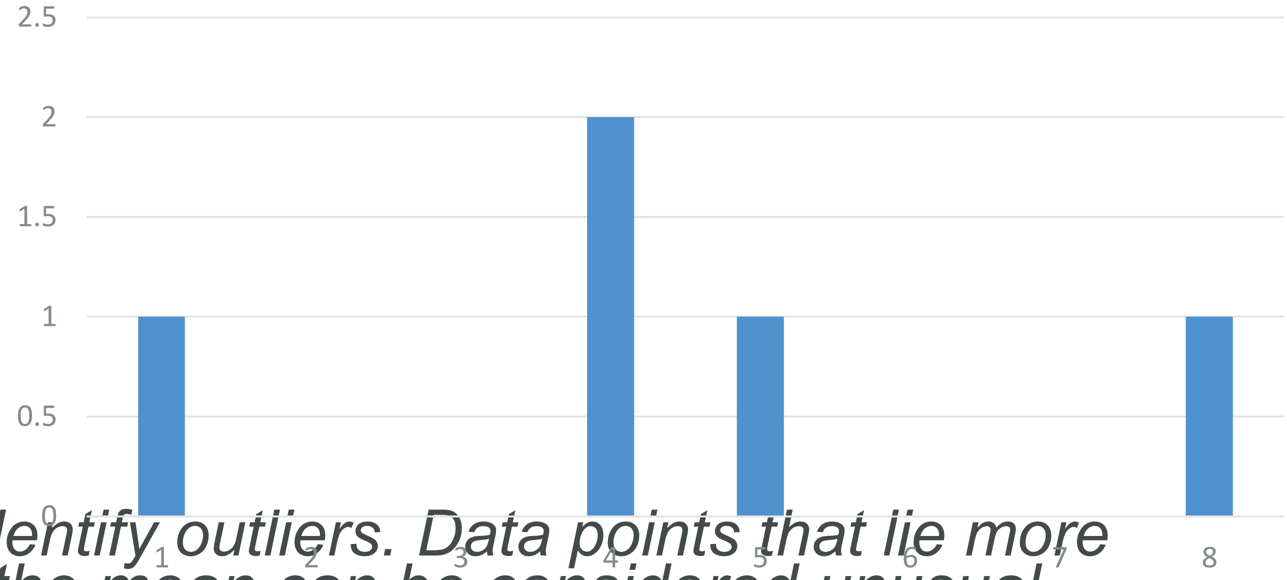
- Variance (σ^2) is simply the **average of the squared differences from the mean**
- Example: What is the variance of the data set (1, 4, 5, 4, 8)?
  - First find the mean: (1+4+5+4+8)/5 = 4.4
  - Now find the differences from the mean: (-3.4, -0.4, 0.6, -0.4, 3.6)
  - Find the squared differences: (11.56, 0.16, 0.36, 0.16, 12.96)
  - Find the average of the squared differences:
    - σ^2= (11.56 + 0.16 + 0.36 + 0.16 + 12.96) / 5 = 5.04

Standard Deviation σ is just the square root of the variance.

$\sigma^2 = 5.04$

$$\sigma = \sqrt{5.04} = 2.24$$

So the standard deviation of (1, 4, 5, 4, 8) is 2.24.



This is usually used as a way to identify outliers. Data points that lie more than one standard deviation from the mean can be considered unusual.

You can talk about how extreme a data point is by talking about "how many sigmas" away from the mean it is.

DataCumulus Sundog Education

Dealing with Outliers

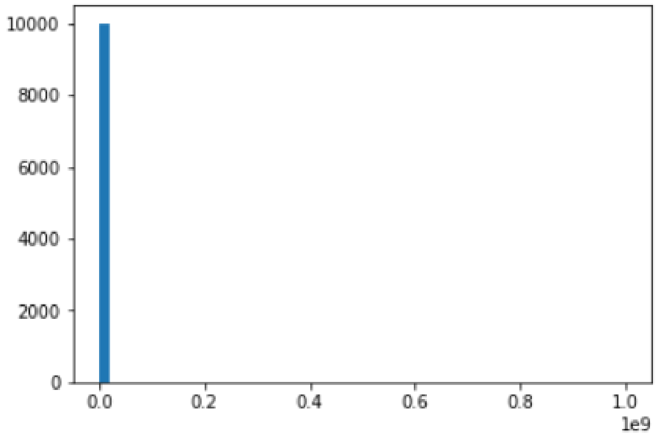
- Sometimes it's appropriate to remove outliers from your training data
- Do this responsibly! Understand why you are doing this.
- For example: in collaborative filtering, a single user who rates thousands of movies could have a big effect on everyone else's ratings. That may not be desirable.
- Another example: in web log data, outliers may represent bots or other agents that should be discarded.
- But if someone really wants the mean income of US citizens for example, don't toss out billionaires just because you want to.

Dealing with Outliers

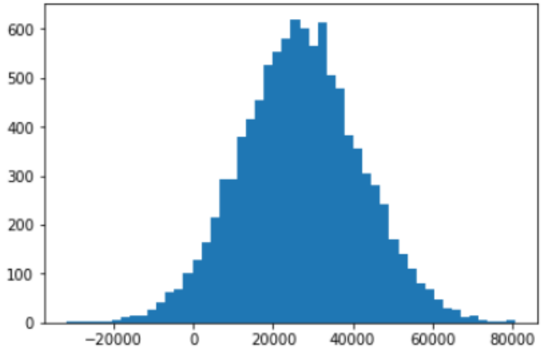
- Our old friend standard deviation provides a principled way to classify outliers.
- Find data points more than some multiple of a standard deviation in your training data.
- What multiple? You just have to use common sense.
- Remember AWS's Random Cut Forest algorithm creeps into many of its services – it is made for outlier detection
    - Found within QuickSight, Kinesis Analytics, SageMaker, and more

Example: Income Inequality

```
%matplotlib inline
import numpy as np

incomes = np.random.normal(27000, 15000, 10000)
incomes = np.append(incomes, [1000000000])

import matplotlib.pyplot as plt
plt.hist(incomes, 50)
plt.show()
```




```
def reject_outliers(data):
    u = np.median(data)
    s = np.std(data)
    filtered = [e for e in data if (u - 2 * s < e < u + 2 * s)]
    return filtered

filtered = reject_outliers(incomes)

plt.hist(filtered, 50)
plt.show()
```



That looks better. And, our mean is more, well, meangingful now as well:

```
np.mean(filtered)
```

```
26894.643431053548
```

Binning

- Bucket observations together based on ranges of values.
- Example: estimated ages of people
  - Put all 20-somethings in one classification, 30-somethings in another, etc.
- Quantile binning categorizes data by their place in the data distribution
  - Ensures even sizes of bins
- Transforms numeric data to ordinal data
- Especially useful when there is uncertainty in the measurements

DataCumulus Sundog Education

Transforming

- Applying some function to a feature to make it better suited for training
- Feature data with an exponential trend may benefit from a logarithmic transform
- Example: YouTube recommendations
  - A numeric feature x is also represented by $x^2$and $\sqrt{x}$
  - This allows learning of super and sub-linear functions
  - (ref: https://static.googleusercontent.com/media/research.google.com/en//pubs/archive/45530.pdf)

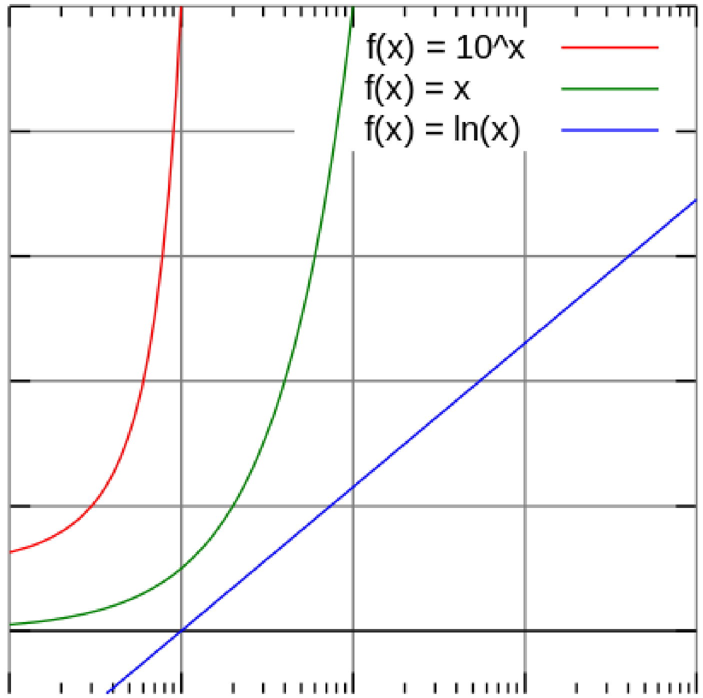
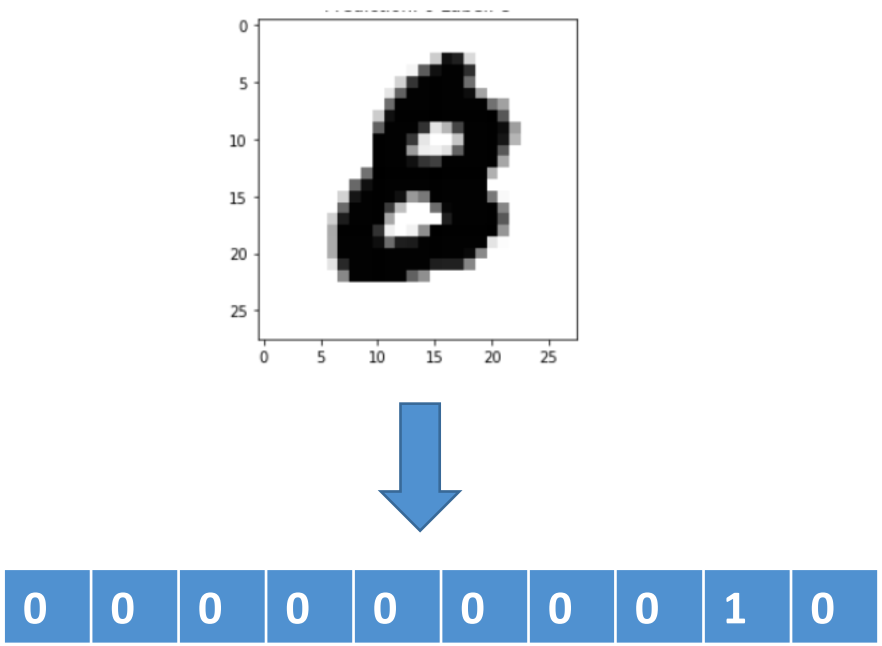


Image: By Autopilot - Own work, CC BY-SA 3.0,
https://commons.wikimedia.org/w/index.php?curid=10733854

Encoding

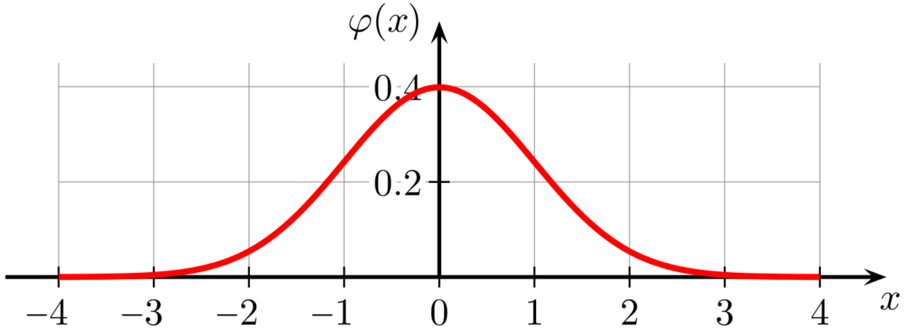
- Transforming data into some new representation required by the model
- One-hot encoding
  - Create “buckets” for every category
  - The bucket for your category has a 1, all others have a 0
  - Very common in deep learning, where categories are represented by individual output “neurons”



DataCumulus Sundog Education

Scaling / Normalization

- Some models prefer feature data to be normally distributed around 0 (most neural nets)
- Most models require feature data to at least be scaled to comparable values
  - Otherwise features with larger magnitudes will have more weight than they should
  - Example: modeling age and income as features – incomes will be much higher values than ages
- Scikit_learn has a preprocessor module that helps (MinMaxScaler, etc)
- Remember to scale your results back up



Geek3 [CC BY 3.0 (https://creativecommons.org/licenses/by/3.0)]

Shuffling

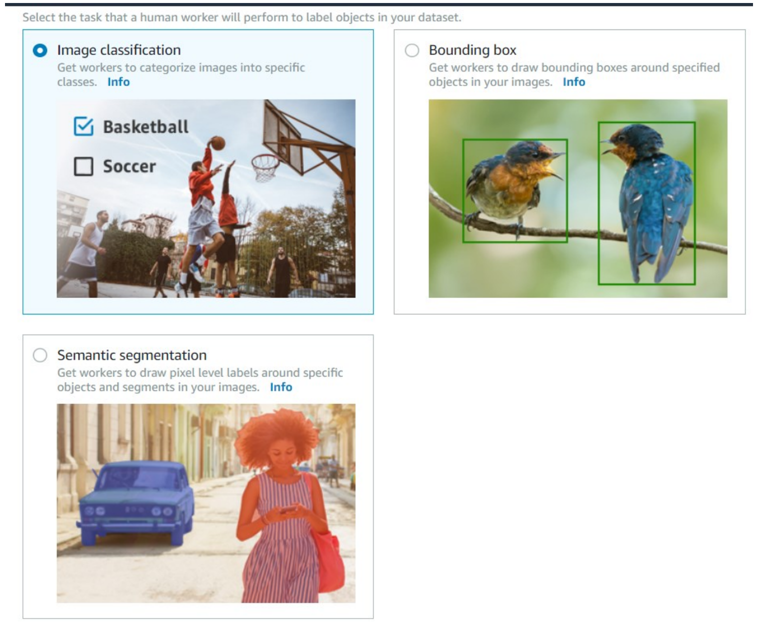
- Many algorithms benefit from shuffling their training data
- Otherwise they may learn from residual signals in the training data resulting from the order in which they were collected

SageMaker Ground Truth

DataCumulus Sundog™ Education

What is Ground Truth?

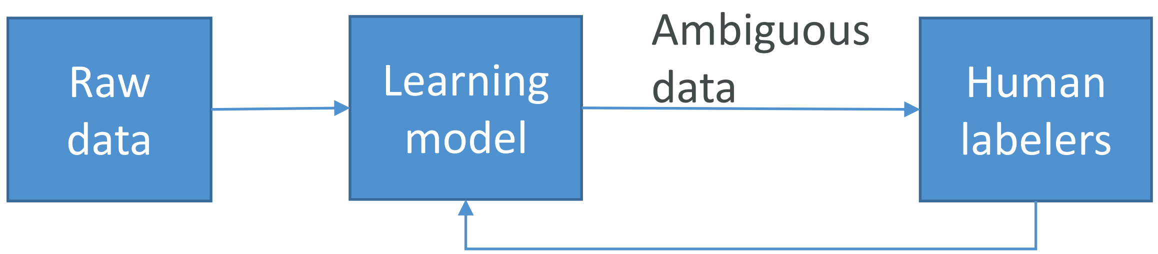
- Sometimes you don't have training data at all, and it needs to be generated by humans first.
- Example: training an image classification model. Somebody needs to tag a bunch of images with what they are images of before training a neural network
- Ground Truth manages humans who will label your data for training purposes



DataCumulus Sundog Education

But it’s more than that

- Ground Truth creates its own model as images are labeled by people
- As this model learns, only images the model isn’t sure about are sent to human labelers
- This can reduce the cost of labeling jobs by 70%



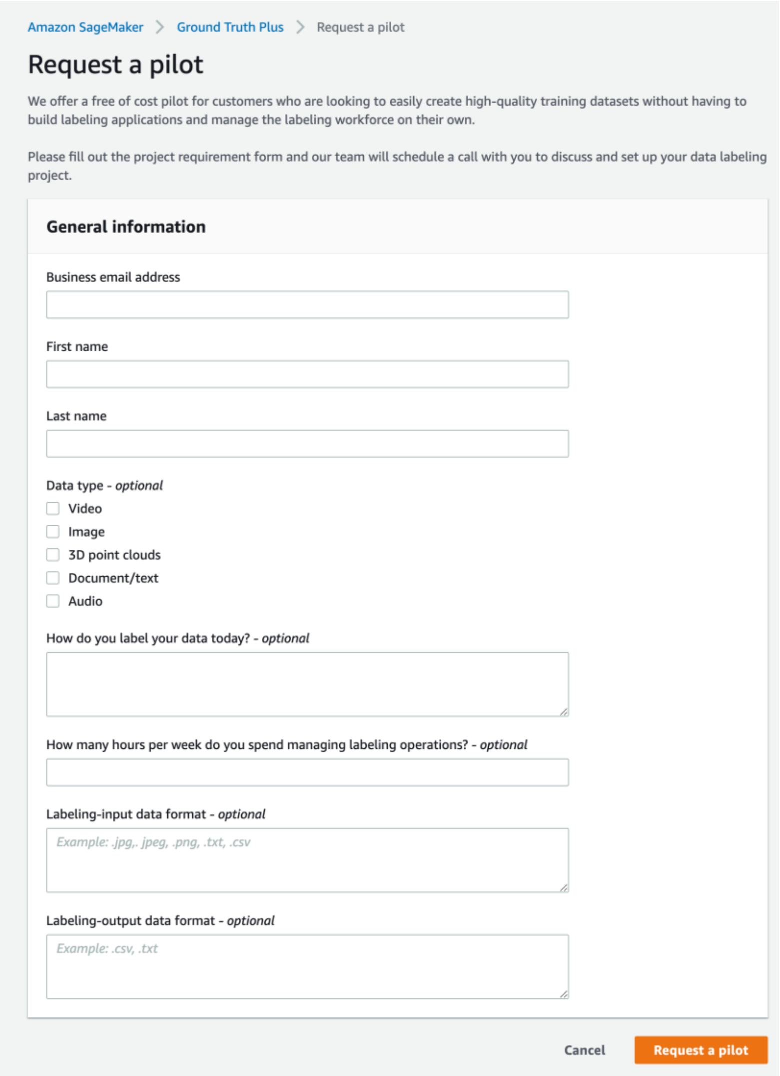
Who are these human labelers?

- Mechanical Turk
- Your own internal team
- Professional labeling companies

DataCumulus Sundog™ Education

Ground Truth Plus

- Turnkey solution
- “Our team of AWS Experts” manages the workflow and team of labelers
    - You fill out an intake form
    - They contact you and discuss pricing
- You track progress via the Ground Truth Plus Project Portal
- Get labeled data from S3 when done



Other ways to generate training labels

- Rekognition
  - AWS service for image recognition
  - Automatically classify images
- Comprehend
  - AWS service for text analysis and topic modeling
  - Automatically classify text by topics, sentiment
- Any pre-trained model or unsupervised technique that may be helpful

Lab: Preparing Data for TF-IDF on Spark and EMR

TF-IDF

- Stands for *Term Frequency* and *Inverse Document Frequency*
- Important data for search – figures out what terms are most relevant for a document
- Sounds fancy!

TF-IDF Explained

- *Term Frequency* just measures how often a word occurs in a document
  - A word that occurs frequently is probably important to that document's meaning
- *Document Frequency* is how often a word occurs in an entire set of documents, i.e., all of Wikipedia or every web page
  - This tells us about common words that just appear everywhere no matter what the topic, like "a", "the", "and", etc.

TF-IDF Explained

- So a measure of the relevancy of a word to a document might be:

$$\frac{Term\ Frequency}{Document\ Frequency}$$

Or: Term Frequency * Inverse Document Frequency

That is, take how often the word appears in a document, over how often it just appears everywhere. That gives you a measure of how important and unique this word is for this document

TF-IDF In Practice

- We actually use the log of the IDF, since word frequencies are distributed exponentially. That gives us a better weighting of a words overall popularity
- TF-IDF assumes a document is just a “bag of words”
  - Parsing documents into a bag of words can be most of the work
  - Words can be represented as a hash value (number) for efficiency
  - What about synonyms? Various tenses? Abbreviations? Capitalizations? Misspellings?
- Doing this at scale is the hard part
  - That’s where Spark comes in!

Unigrams, bigrams, etc.

- An extension of TF-IDF is to not only compute relevancy for individual words (terms) but also for *bi-grams* or, more generally, <u>*n-grams*</u>.
- "I love certification exams"
  - Unigrams: "I", "love", "certification", "exams"
  - Bi-grams: "I love", "love certification", "certification exams"
  - Tri-grams: "I love certification", "love certification exams"

DataCumulus Sundog Education

Sample TF-IDF matrix with unigrams and bigrams

	I	Love	Certification	Exams	Puppies	I love	Love certification	Love puppies	Certification exams
"I love certification exams"									
"I love puppies"									

Using TF-IDF

- A very simple search algorithm could be:
  - Compute TF-IDF for every word in a corpus
  - For a given search word, sort the documents by their TF-IDF score for that word
  - Display the results

DataCumulus Sundog™ Education

Let’s use TF-IDF on Wikipedia

WIKIPEDIA
The Free Encyclopedia

DataCumulus Sundog Education

Modeling, Part 1: General Deep Learning and Machine Learning

DataCumulus

Sundog™ Education

Deep Learning 101

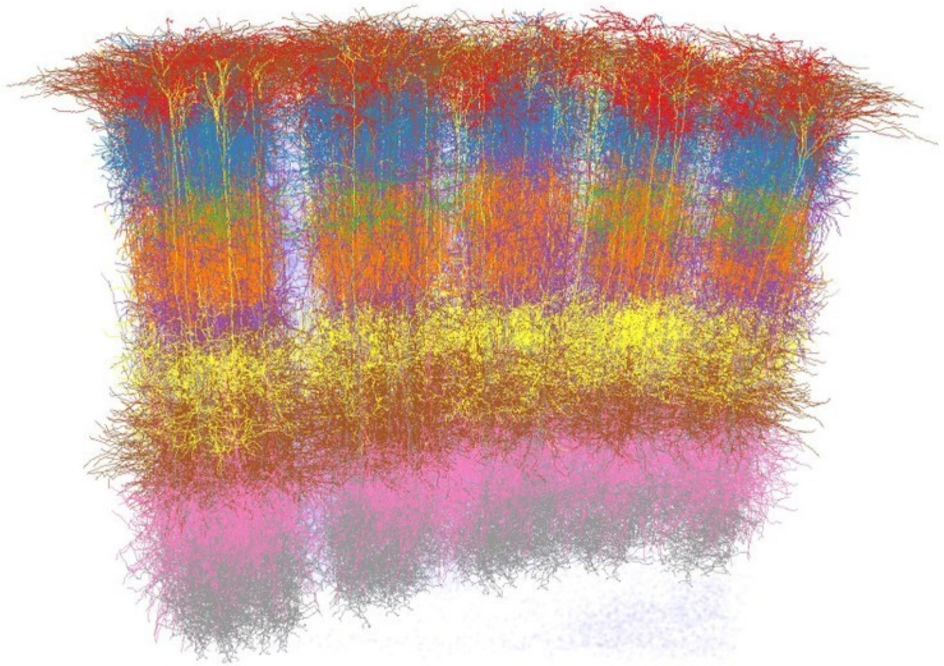
And AWS Best Practices

The biological inspiration

- Neurons in your cerebral cortex are connected via axons
- A neuron “fires” to the neurons it’s connected to, when enough of its input signals are activated.
- Very simple at the individual neuron level – but layers of neurons connected in this way can yield learning behavior.
- Billions of neurons, each with thousands of connections, yields a mind

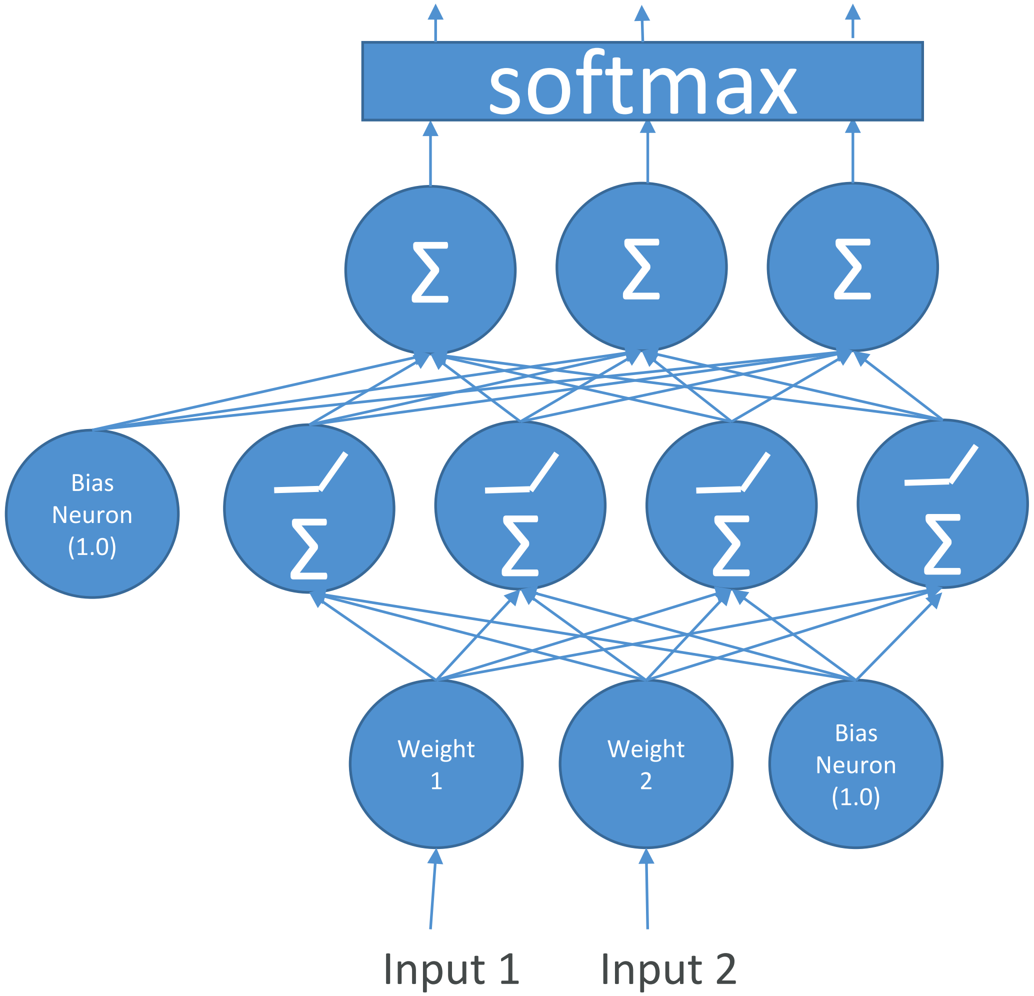
Cortical columns

- Neurons in your cortex seem to be arranged into many stacks, or “columns” that process information in parallel
- “mini-columns” of around 100 neurons are organized into larger “hyper-columns”. There are 100 million mini-columns in your cortex
- This is coincidentally similar to how GPU’s work…



(credit: Marcel Oberlaender et al.)

Deep Neural Networks



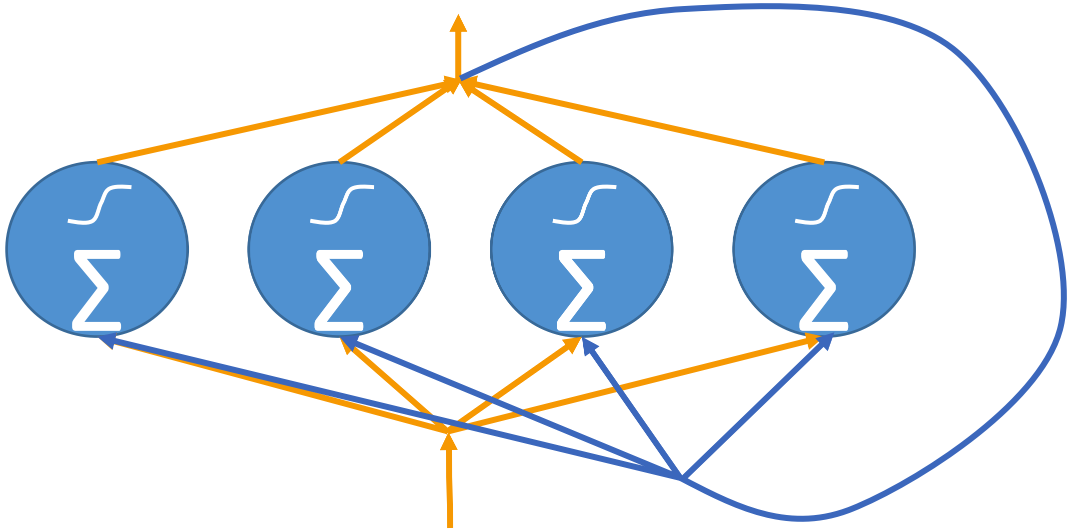
Deep Learning Frameworks

- Tensorflow / Keras
- MXNet

```
model = Sequential()

model.add(Dense(64, activation='relu', input_dim=20))
model.add(Dropout(0.5))
model.add(Dense(64, activation='relu'))
model.add(Dropout(0.5))
model.add(Dense(10, activation='softmax'))
sgd = SGD(lr=0.01, decay=1e-6, momentum=0.9,
       nesterov=True)
model.compile(loss='categorical_crossentropy',
       optimizer=sgd, metrics=['accuracy'])
```

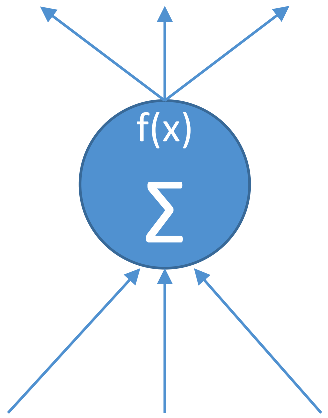
Types of Neural Networks

- Feedforward Neural Network
- Convolutional Neural Networks (CNN)
  - Image classification (is there a stop sign in this image?)
- Recurrent Neural Networks (RNNs)
  - Deals with sequences in time (predict stock prices, understand words in a sentence, translation, etc)
  - **LSTM**, GRU



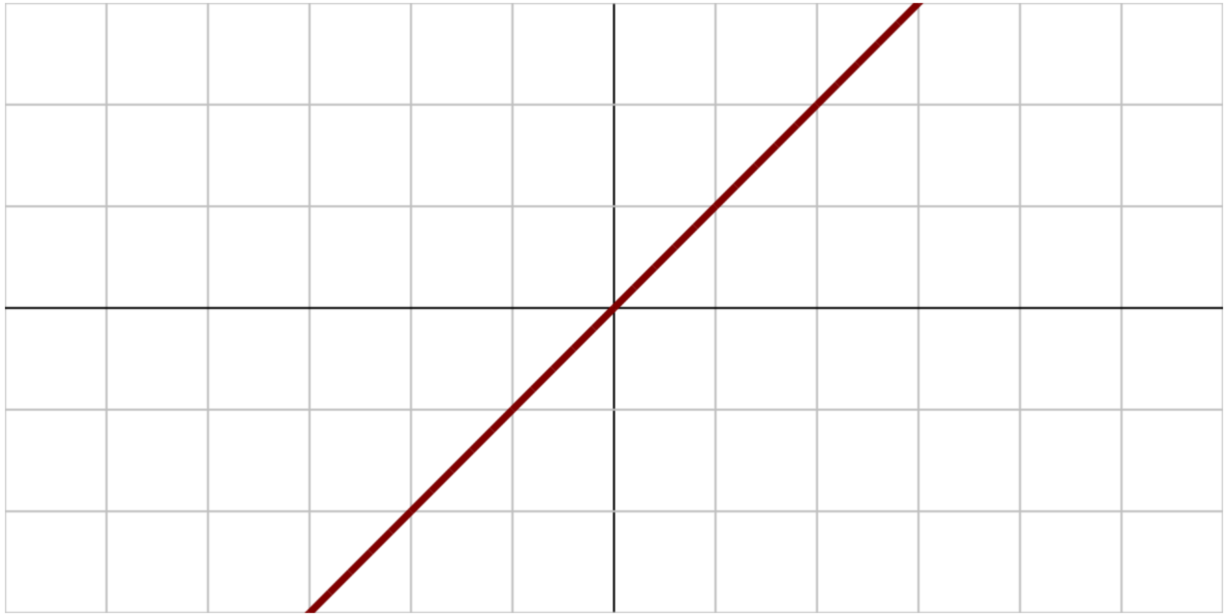
Activation Functions

- Define the output of a node / neuron given its input signals



Linear activation function

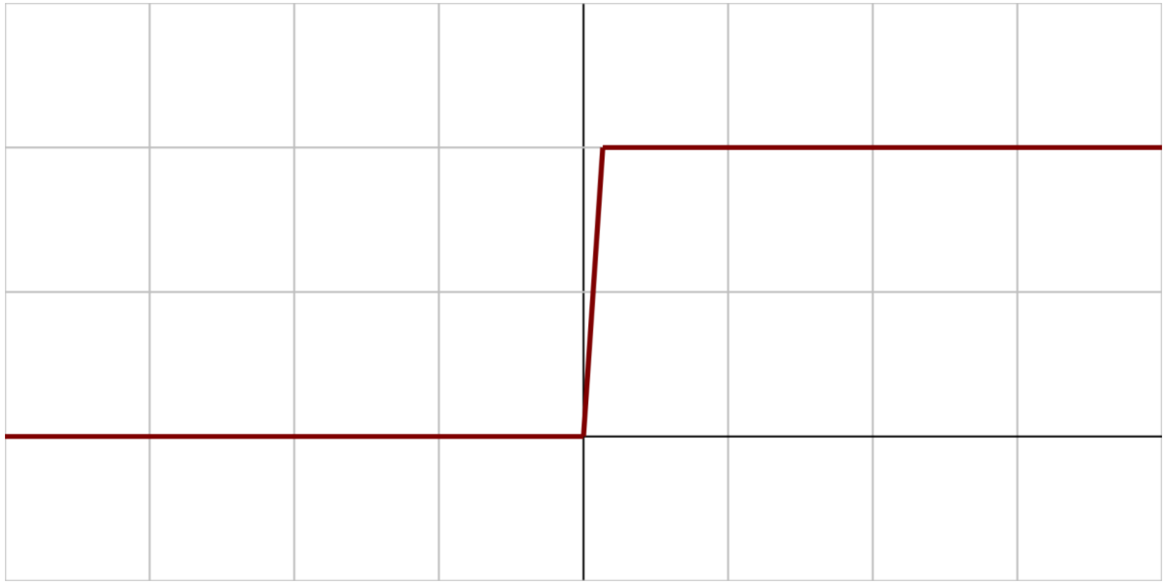
- It doesn't really *do* anything
- Can't do backpropagation



By Laughsinthestocks - Own work, CC BY-SA 4.0, https://commons.wikimedia.org/w/index.php?curid=44920411

Binary step function

- It's on or off
- Can't handle multiple classification – it's binary after all
- Vertical slopes don't work well with calculus!



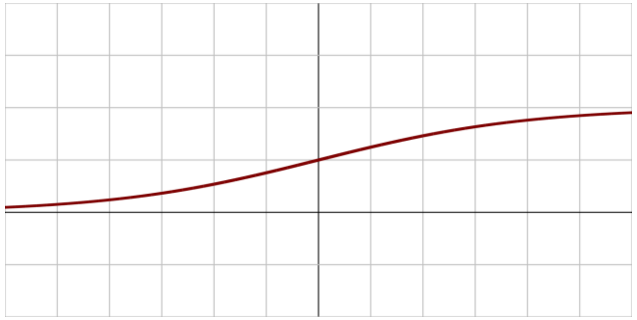
By Laughsinthestocks - Own work, CC BY-SA 4.0, https://commons.wikimedia.org/w/index.php?curid=44920435

Instead we need non-linear activation functions

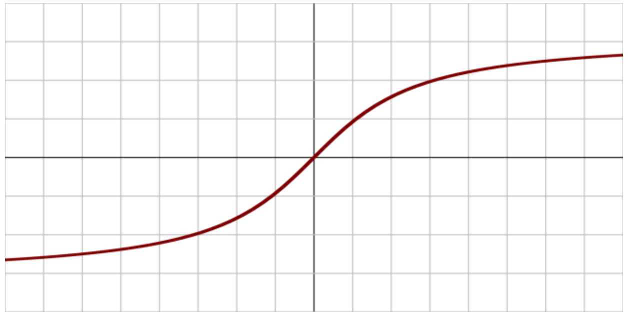
- These can create complex mappings between inputs and outputs
- Allow backpropagation (because they have a useful derivative)
- Allow for multiple layers (linear functions degenerate to a single layer)

Sigmoid / Logistic / TanH

- Nice & smooth
- Scales everything from 0-1 (Sigmoid / Logistic) or -1 to 1 (tanh / hyperbolic tangent)
- But: changes slowly for high or low values
  - The "Vanishing Gradient" problem
- Computationally expensive
- Tanh generally preferred over sigmoid



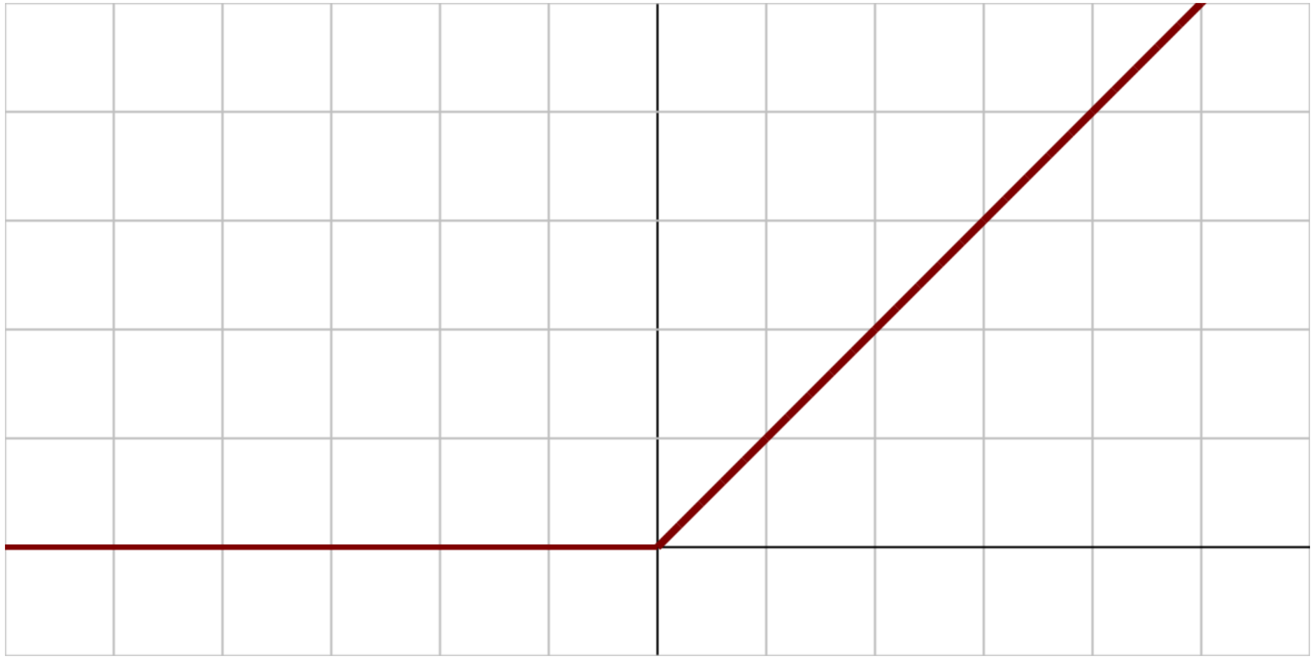
Sigmoid AKA Logistic



TanH AKA Hyperbolic Tangent

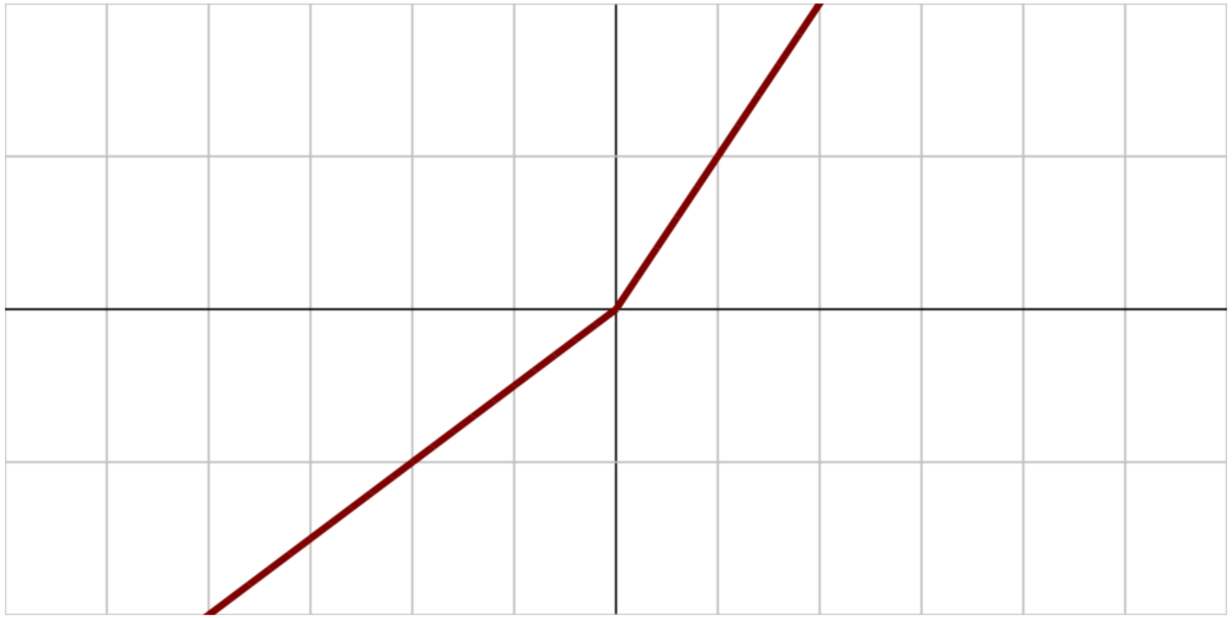
Rectified Linear Unit (ReLU)

- Now we're talking
- Very popular choice
- Easy & fast to compute
- But, when inputs are zero or negative, we have a linear function and all of its problems
  - The "Dying ReLU problem"



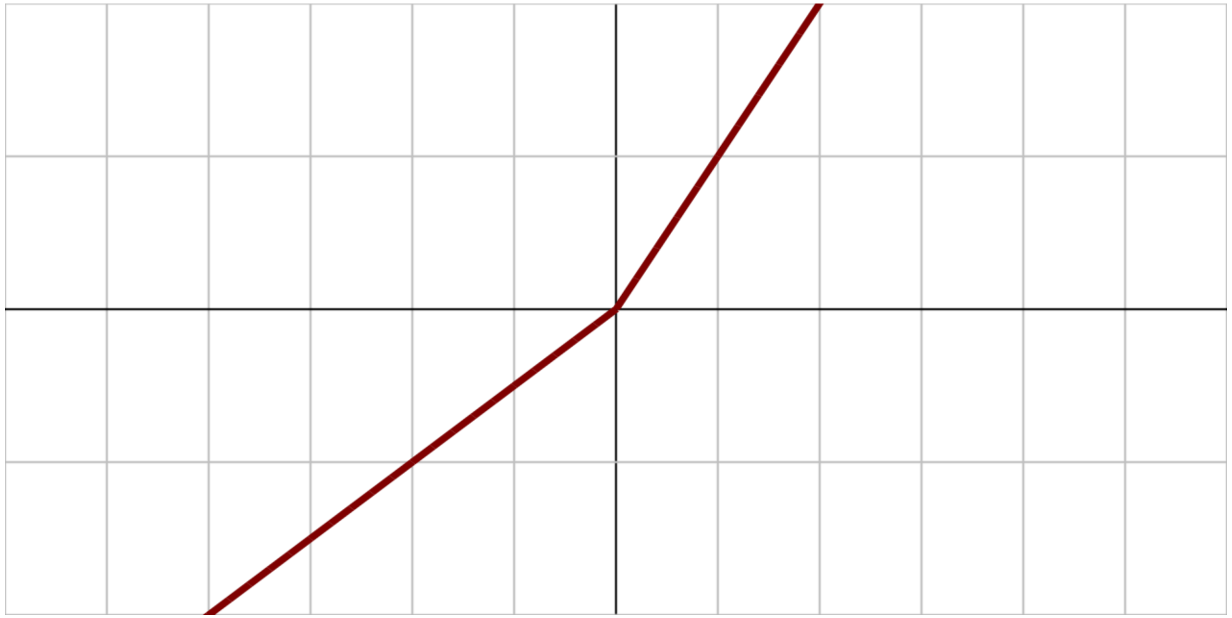
Leaky ReLU

- Solves "dying ReLU" by introducing a negative slope below 0 (usually not as steep as this)



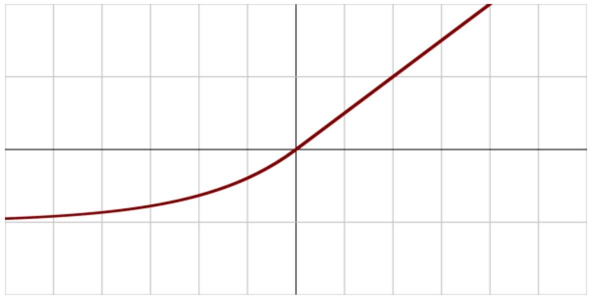
Parametric ReLU (PReLU)

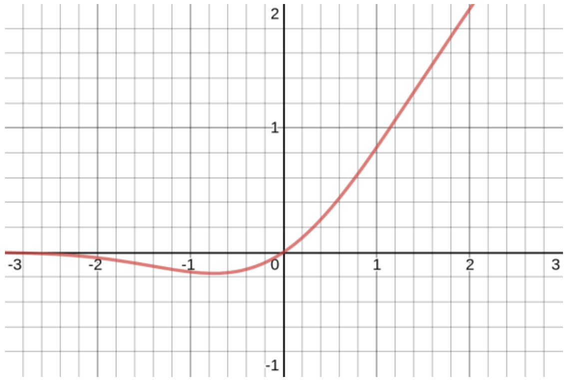
- ReLU, but the slope in the negative part is learned via backpropagation
- Complicated and YMMV



Other ReLU variants

- Exponential Linear Unit (ELU)
- Swish
  - From Google, performs really well
  - But it's from Google, not Amazon…
  - Mostly a benefit with very deep networks (40+ layers)
- Maxout
  - Outputs the max of the inputs
  - Technically ReLU is a special case of maxout
  - But doubles parameters that need to be trained, not often practical.

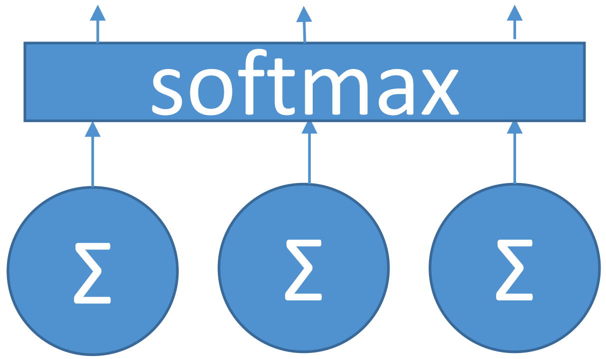




DataCumulus Sundog™ Education

Softmax

- Used on the final output layer of a multi-class classification problem
- Basically converts outputs to probabilities of each classification
- Can't produce more than one label for something (sigmoid can)
- Don't worry about the actual function for the exam, just know what it's used for.



Choosing an activation function

- For multiple classification, use softmax on the output layer
- RNN's do well with Tanh
- For everything else
  - Start with ReLU
  - If you need to do better, try Leaky ReLU
  - Last resort: PReLU, Maxout
  - Swish for really deep networks

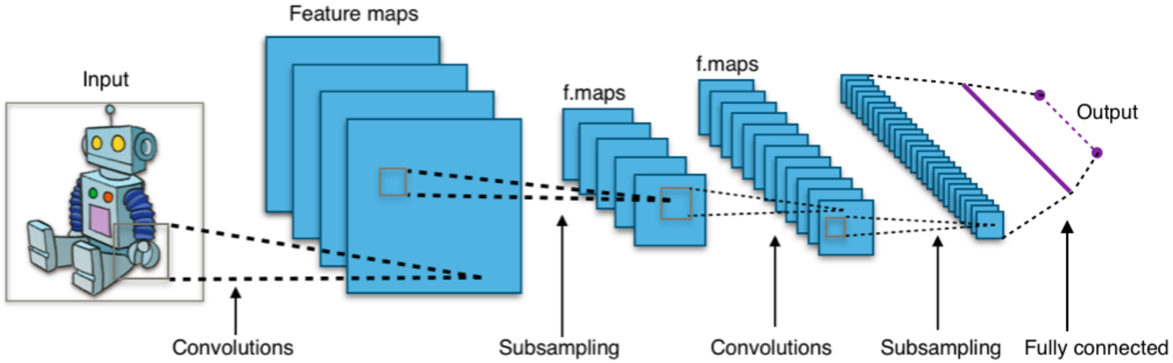
CNN's: what are they for?

- When you have data that doesn't neatly align into columns
  - Images that you want to find features within
  - Machine translation
  - Sentence classification
  - Sentiment analysis
- They can find features that aren't in a specific spot
  - Like a stop sign in a picture
  - Or words within a sentence
- They are "feature-location invariant"

DataCumulus Sundog™ Education

CNN's: how do they work?

- Inspired by the biology of the visual cortex
  - Local receptive fields are groups of neurons that only respond to a part of what your eyes see (subsampling)
  - They overlap each other to cover the entire visual field (convolutions)
  - They feed into higher layers that identify increasingly complex images
    - Some receptive fields identify horizontal lines, lines at different angles, etc. (filters)
    - These would feed into a layer that identifies shapes
    - Which might feed into a layer that identifies objects
  - For color images, extra layers for red, green, and blue



DataCumulus Sundog Education

How do we “know” that’s a stop sign?

- Individual local receptive fields scan the image looking for edges, and pick up the edges of the stop sign in a layer
- Those edges in turn get picked up by a higher level convolution that identifies the stop sign’s shape (and letters, too)
- This shape then gets matched against your pattern of what a stop sign looks like, also using the strong red signal coming from your red layers
- That information keeps getting processed upward until your foot hits the brake!
- A CNN works the same way

CNN's with Keras / Tensorflow

- Source data must be of appropriate dimensions
  - ie width x length x color channels
- Conv2D layer type does the actual convolution on a 2D image
  - Conv1D and Conv3D also available – doesn't have to be image data
- MaxPooling2D layers can be used to reduce a 2D layer down by taking the maximum value in a given block
- Flatten layers will convert the 2D layer to a 1D layer for passing into a flat hidden layer of neurons
- Typical usage:
  - Conv2D -> MaxPooling2D -> Dropout -> Flatten -> Dense -> Dropout -> Softmax

DataCumulus Sundog Education

CNN's are hard

- Very resource-intensive (CPU, GPU, and RAM)
- Lots of hyperparameters
  - Kernel sizes, many layers with different numbers of units, amount of pooling… in addition to the usual stuff like number of layers, choice of optimizer
- Getting the training data is often the hardest part! (As well as storing and accessing it)

Specialized CNN architectures

- Defines specific arrangement of layers, padding, and hyperparameters
- LeNet-5
  - Good for handwriting recognition
- AlexNet
  - Image classification, deeper than LeNet
- GoogLeNet
  - Even deeper, but with better performance
  - Introduces *inception modules* (groups of convolution layers)
- ResNet (Residual Network)
  - Even deeper – maintains performance via *skip connections*.

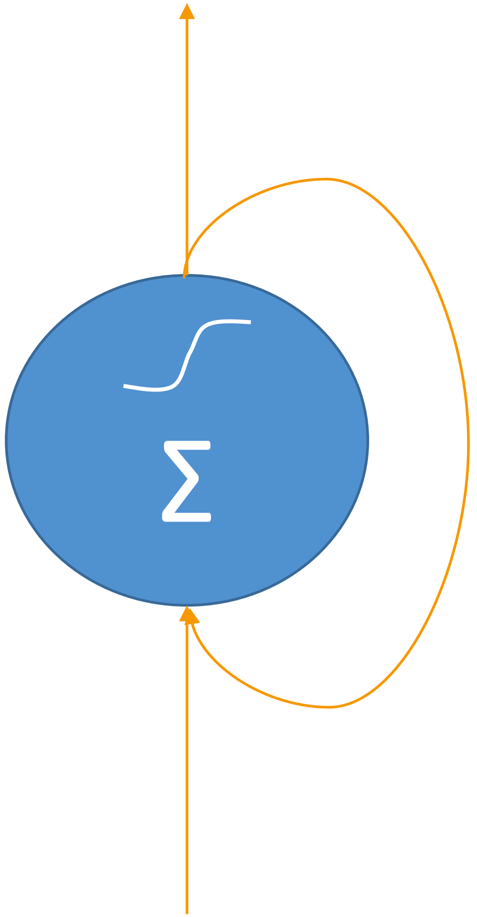
Recurrent Neural Networks

RNN's: what are they for?

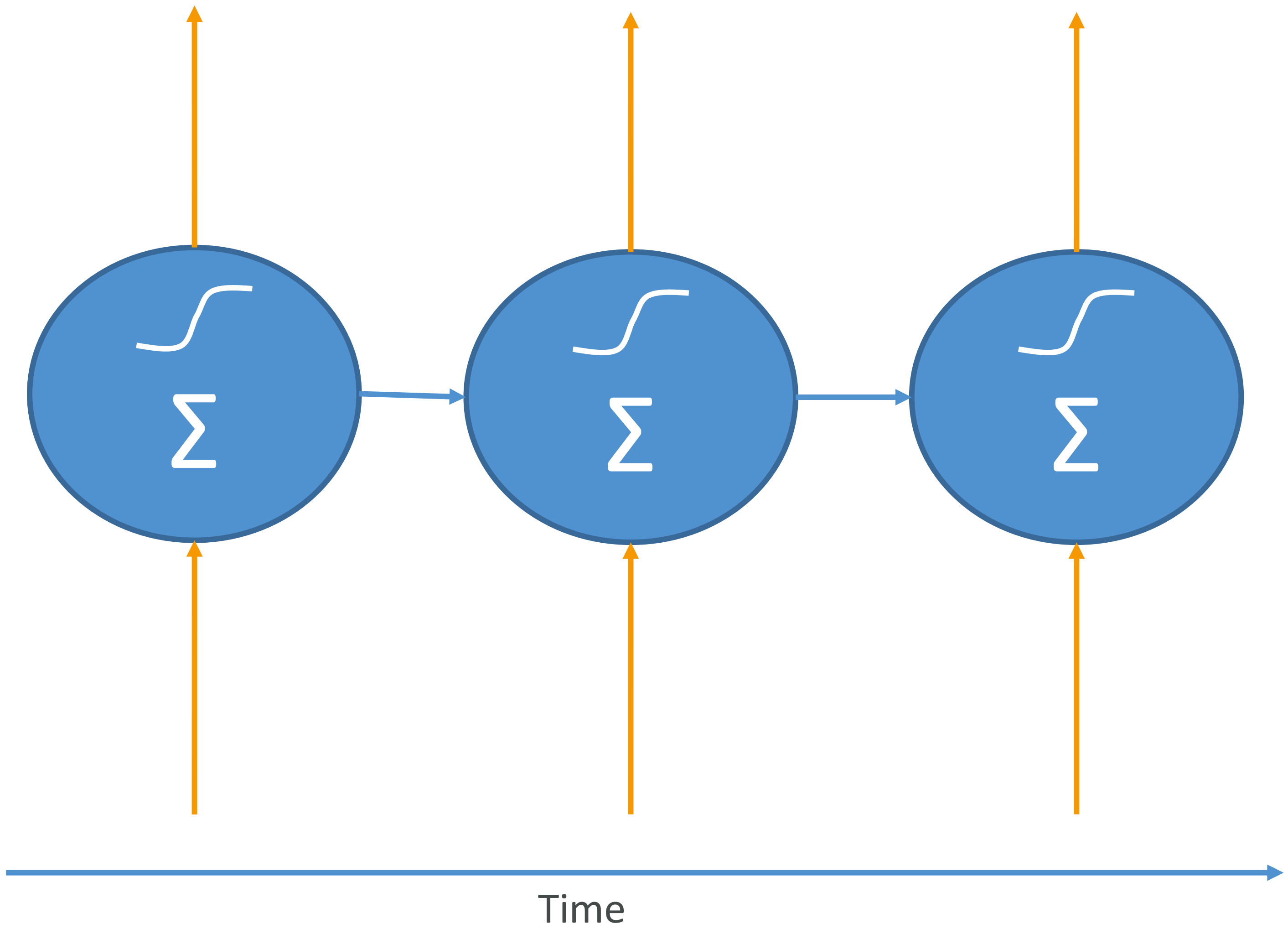
- Time-series data
  - When you want to predict future behavior based on past behavior
  - Web logs, sensor logs, stock trades
  - Where to drive your self-driving car based on past trajectories
- Data that consists of sequences of arbitrary length
  - Machine translation
  - Image captions
  - Machine-generated music

DataCumulus Sundog Education

A recurrent neuron

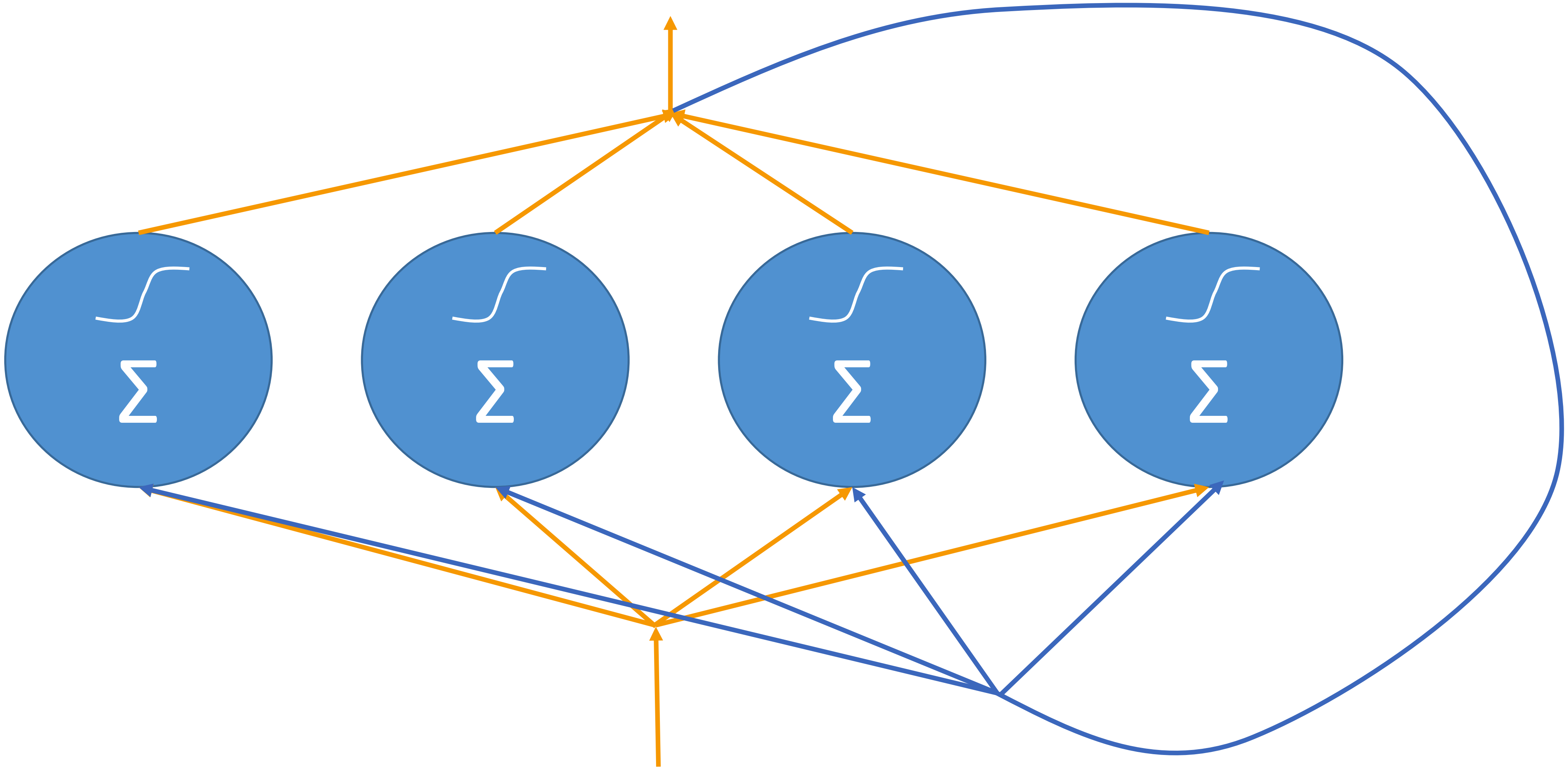


Another way to look at it



A "Memory Cell"

A layer of recurrent neurons



RNN topologies

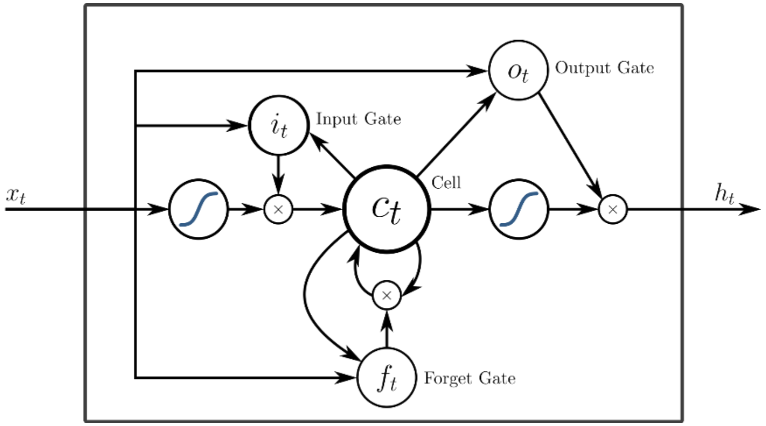
- Sequence to sequence
  - i.e., predict stock prices based on series of historical data
- Sequence to vector
  - i.e., words in a sentence to sentiment
- Vector to sequence
  - i.e., create captions from an image
- Encoder -> Decoder
  - Sequence -> vector -> sequence
  - i.e., machine translation

Training rnn's

- Backpropagation through time
  - Just like backpropagation on MLP's, but applied to each time step.
- All those time steps add up fast
  - Ends up looking like a really, really deep neural network.
  - Can limit backpropagation to a limited number of time steps (truncated backpropagation through time)

Training rnn's

- State from earlier time steps get diluted over time
  - This can be a problem, for example when learning sentence structures
- LSTM Cell
  - Long Short-Term Memory Cell
  - Maintains separate short-term and long-term states
- GRU Cell
  - Gated Recurrent Unit
  - Simplified LSTM Cell that performs about as well



DataCumulus
Sundog Education

Training rnn's

- It's really hard
  - Very sensitive to topologies, choice of hyperparameters
  - Very resource intensive
  - A wrong choice can lead to a RNN that doesn't converge at all.

Modern Natural Language Processing

- **Transformer** deep learning architectures are what's hot
  - Adopts mechanism of "self-attention"
    - Weighs significance of each part of the input data
    - Processes sequential data (like words, like an RNN), but processes entire input all at once.
    - The attention mechanism provides context, so no need to process one word at a time.
  - BERT, RoBERTa, T5, GPT-2 etc., DistilBERT
  - DistilBERT: uses knowledge distillation to reduce model size by 40%
- BERT: Bi-directional Encoder Representations from Transformers
- GPT: Generative Pre-trained Transformer

DataCumulus Sundog Education

Transfer Learning

- NLP models (and others) are too big and complex to build from scratch and re-train every time
  - The latest may have hundreds of billions of parameters!
- Model zoos such as **Hugging Face** offer pre-trained models to start from
  - Integrated with Sagemaker via Hugging Face Deep Learning Containers
- You can fine-tune these models for your own use cases

Transfer Learning: BERT example

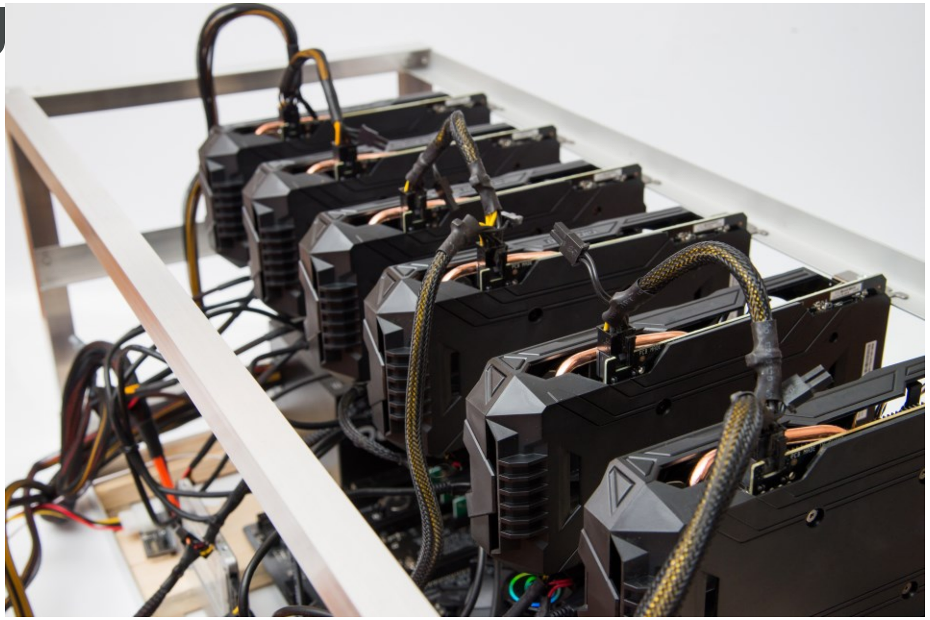
- Hugging Face offers a Deep Learning Container (DLC) for BERT
- It's pre-trained on BookCorpus and Wikipedia
- You can **fine-tune** BERT (or DistilBERT etc) with your own additional training data through **transfer learning**
  - Tokenize your own training data to be of the same format
  - Just start training it further with your data, with a low learning rate.

Transfer Learning approaches

- Continue training a pre-trained model **(fine-tuning)**
  - Use for fine-tuning a model that has way more training data than you'll ever have
  - Use a low learning rate to ensure you are just incrementally improving the model
- Add new trainable layers to the top of a frozen model
  - Learns to turn old features into predictions on new data
  - Can do both: add new layers, then fine tune as well
- Retrain from scratch
  - If you have large amounts of training data, and it's fundamentally different from what the model was pre-trained with
  - And you have the computing capacity for it!
- Use it as-is
  - When the model's training data is what you want already

Deep Learning on EC2 / EMR

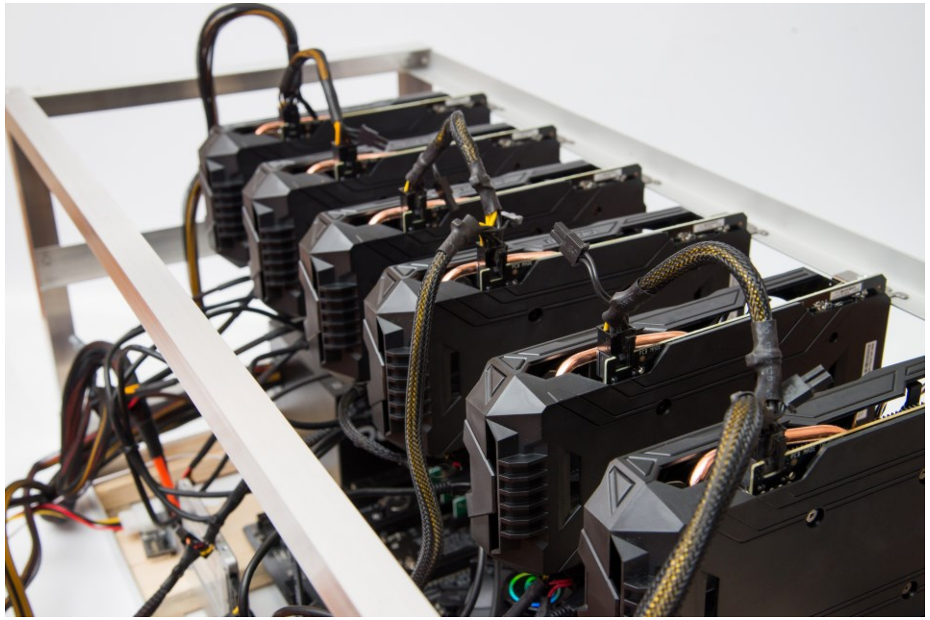
- EMR supports Apache MXNet and GPU instance types
- Appropriate instance types for deep learning:
  - P3: 8 Tesla V100 GPU's
  - P2: 16 K80 GPU's
  - G3: 4 M60 GPU's (all Nvidia chips)
  - G5g: AWS Graviton 2 processors / Nvidia T4G Tensor Core GPU's
    - Not (yet) available in EMR
    - Also used for Android game streaming
  - P4d – A100 "UltraClusters" for supercomputing
- Deep Learning AMI's



DataCumulus Sundog Education

Deep Learning on EC2 / EMR

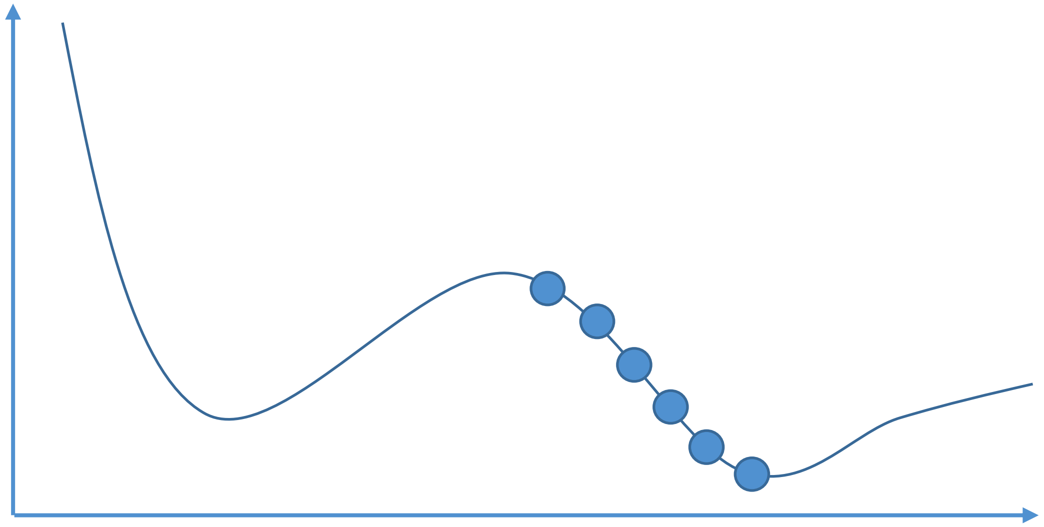
- Trn1 instances
  - "Powered by Trainium"
  - Optimized for training (50% savings)
  - 800 Gbps of Elastic Fabric Adapter (EFA) networking for fast clusters
- Trn1n instances
  - Even more bandwidth (1600 Gbps)
- Inf2 instances
  - "Powered by AWS Inferentia2"
  - Optimized for inference



Tuning Neural Networks

Learning Rate

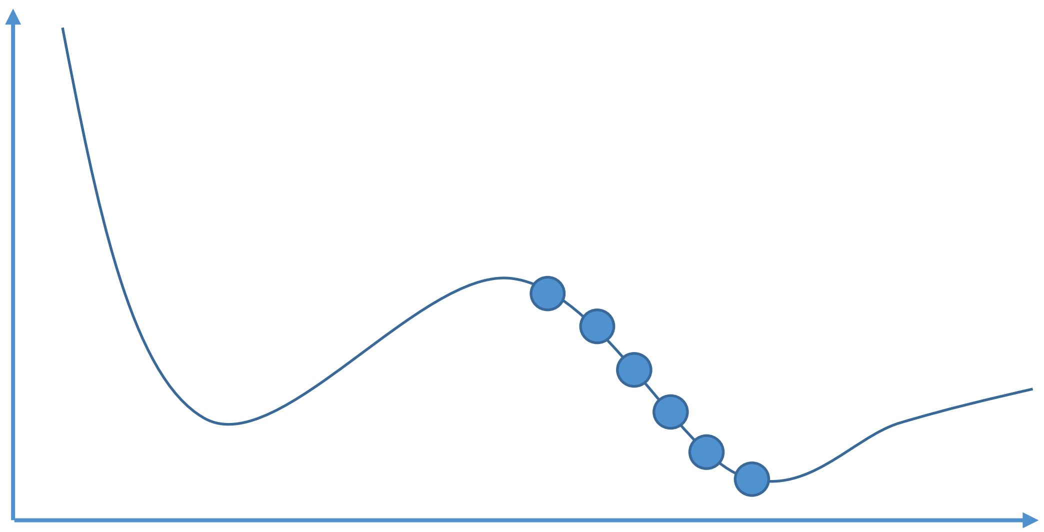
- Neural networks are trained by gradient descent (or similar means)
- We start at some random point, and sample different solutions (weights) seeking to minimize some cost function, over many *epochs*
- How far apart these samples are is the *learning rate*



DataCumulus Sundog Education

Effect of learning rate

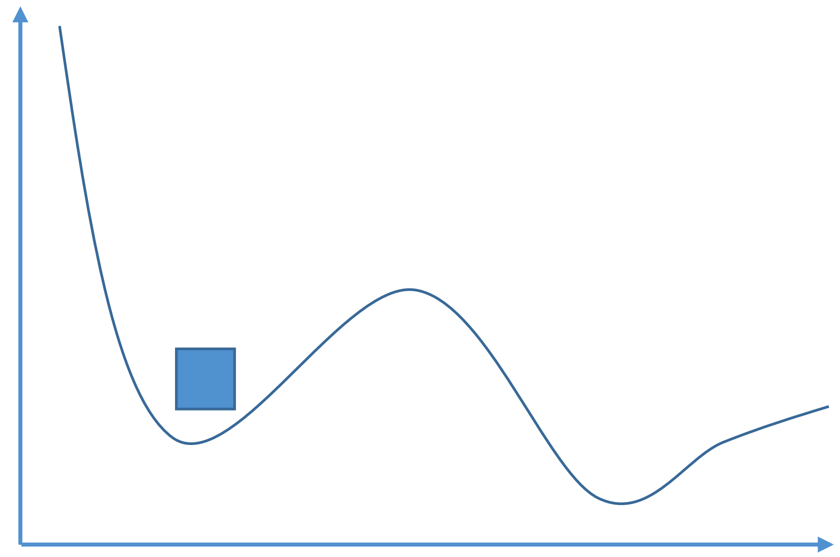
- Too high a learning rate means you might overshoot the optimal solution!
- Too small a learning rate will take too long to find the optimal solution
- Learning rate is an example of a *hyperparameter*



DataCumulus Sundog™ Education

Batch Size

- How many training samples are used within each batch of each epoch
- Somewhat counter-intuitively:
  - Smaller batch sizes can work their way out of “local minima” more easily
  - Batch sizes that are too large can end up getting stuck in the wrong solution
  - Random shuffling at each epoch can make this look like very inconsistent results from run to run



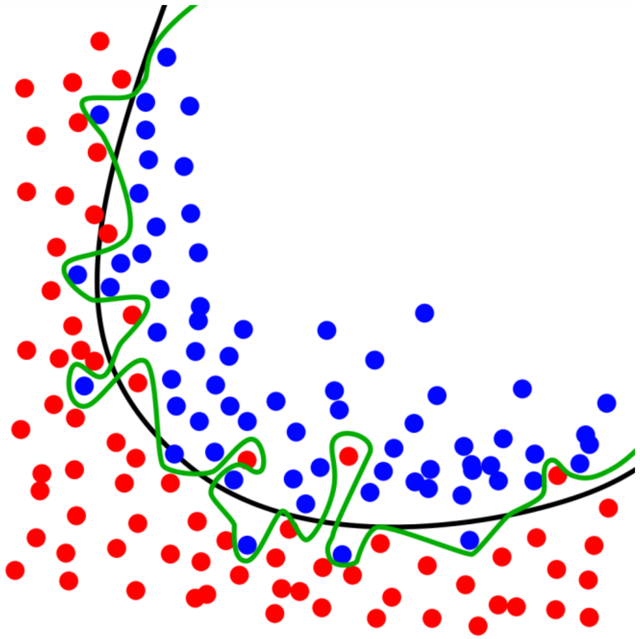
To Recap (this is important!)

- Small batch sizes tend to not get stuck in local minima
- Large batch sizes can converge on the wrong solution at random
- Large learning rates can overshoot the correct solution
- Small learning rates increase training time

Neural Network Regularization Techniques

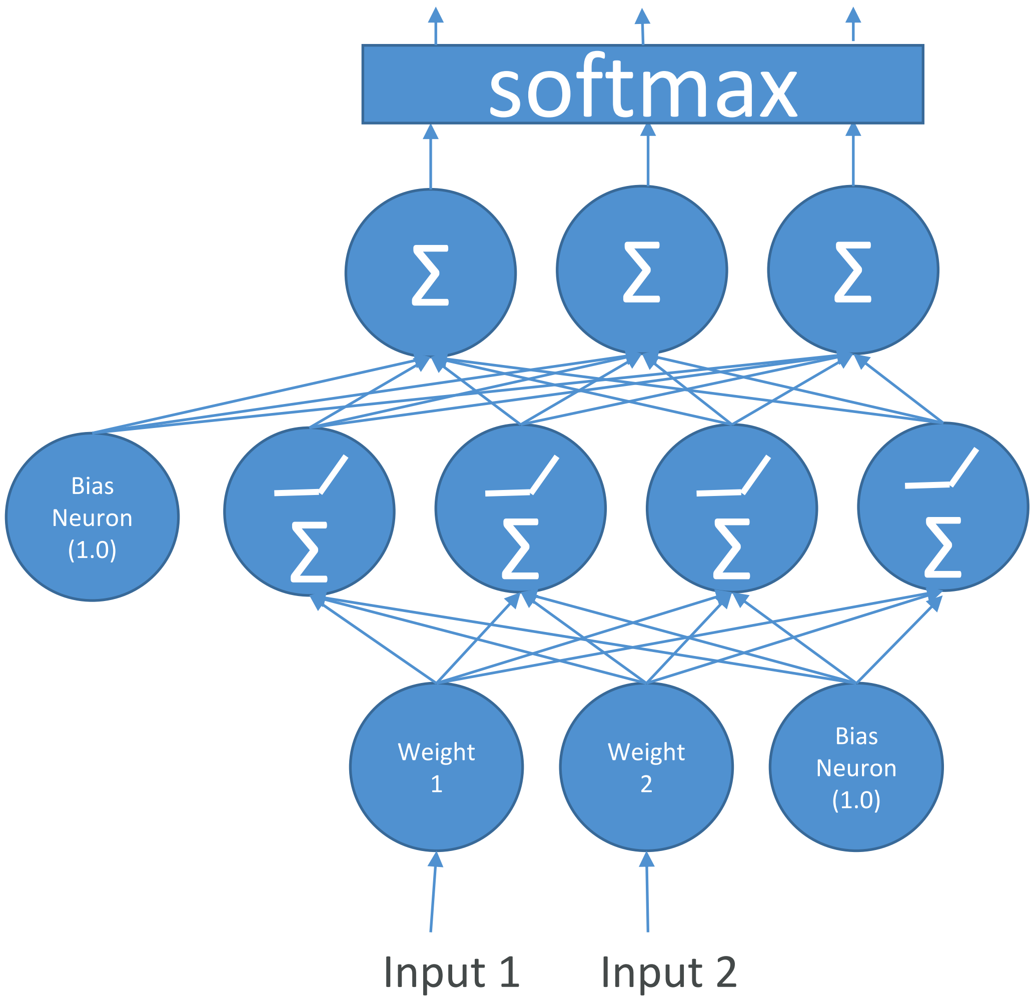
What is regularization?

- Preventing *overfitting*
  - Models that are good at making predictions on the data they were trained on, but not on new data it hasn't seen before
  - Overfitted models have learned patterns in the training data that don't generalize to the real world
  - Often seen as high accuracy on training data set, but lower accuracy on test or evaluation data set.
    - When training and evaluating a model, we use *training*, *evaluation*, and *testing* data sets.
- Regularization techniques are intended to prevent overfitting.

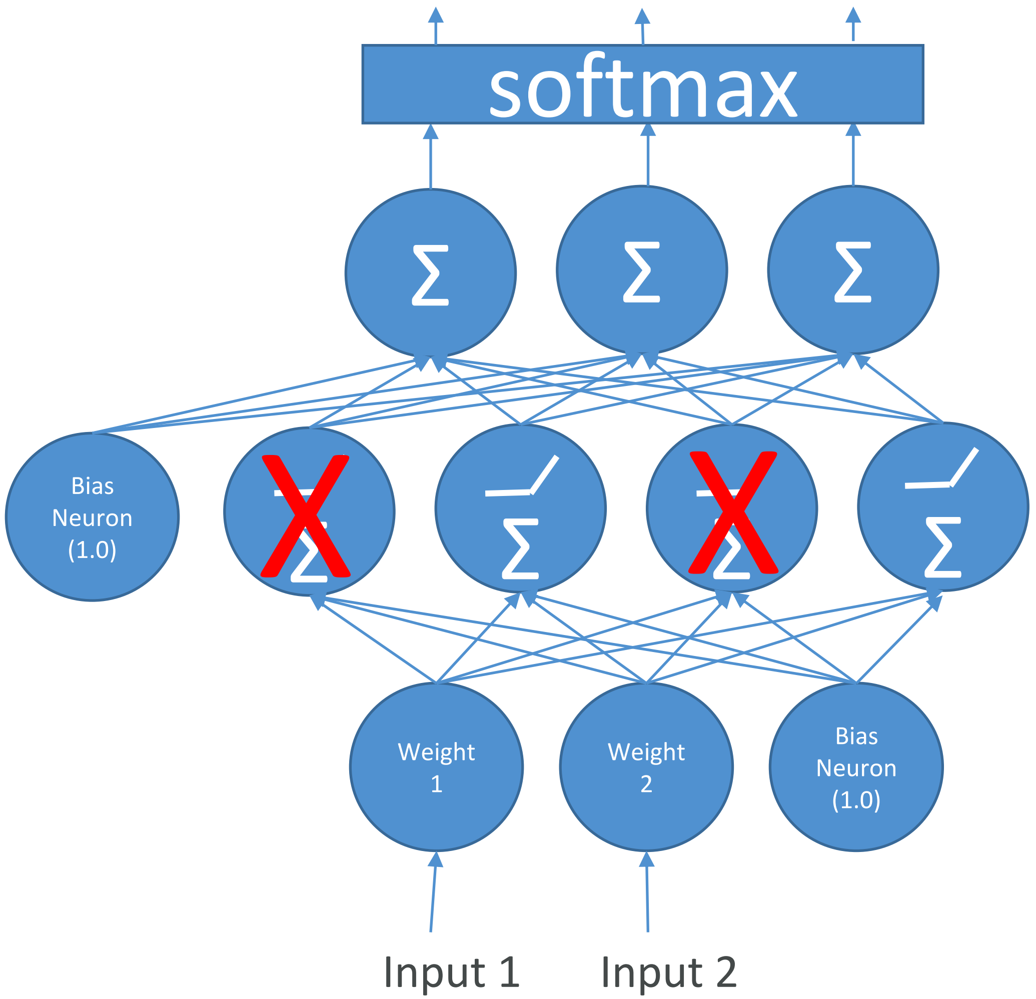


Chabacano [CC BY-SA 4.0 (https://creativecommons.org/licenses/by-sa/4.0)]

Too many layers? Too many neurons?



Dropout


softmax
Σ
Σ
Σ
Bias
Neuron
(1.0)
Σ
Σ
Σ
Σ
Weight
1
Weight
2
Bias
Neuron
(1.0)
Input 1
Input 2

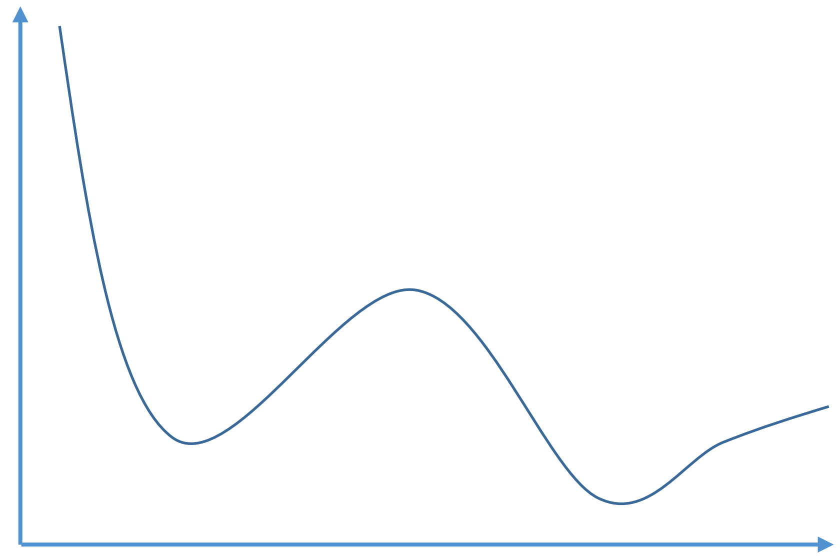
Early Stopping

```
Epoch 1/10
 - 4s - loss: 0.2406 - acc: 0.9302 - val_loss: 0.1437 - val_acc: 0.9557
Epoch 2/10
 - 2s - loss: 0.0971 - acc: 0.9712 - val_loss: 0.0900 - val_acc: 0.9725
Epoch 3/10
 - 2s - loss: 0.0653 - acc: 0.9803 - val_loss: 0.0725 - val_acc: 0.9786
Epoch 4/10
 - 2s - loss: 0.0471 - acc: 0.9860 - val_loss: 0.0689 - val_acc: 0.9795
Epoch 5/10
 - 2s - loss: 0.0367 - acc: 0.9890 - val_loss: 0.0675 - val_acc: 0.9808
Epoch 6/10
 - 2s - loss: 0.0266 - acc: 0.9919 - val_loss: 0.0680 - val_acc: 0.9796
Epoch 7/10
 - 2s - loss: 0.0208 - acc: 0.9937 - val_loss: 0.0678 - val_acc: 0.9811
Epoch 8/10
 - 2s - loss: 0.0157 - acc: 0.9953 - val_loss: 0.0719 - val_acc: 0.9810
Epoch 9/10
 - 2s - loss: 0.0130 - acc: 0.9960 - val_loss: 0.0707 - val_acc: 0.9825
Epoch 10/10
 - 2s - loss: 0.0097 - acc: 0.9972 - val_loss: 0.0807 - val_acc: 0.9805
```

Grief with Gradients

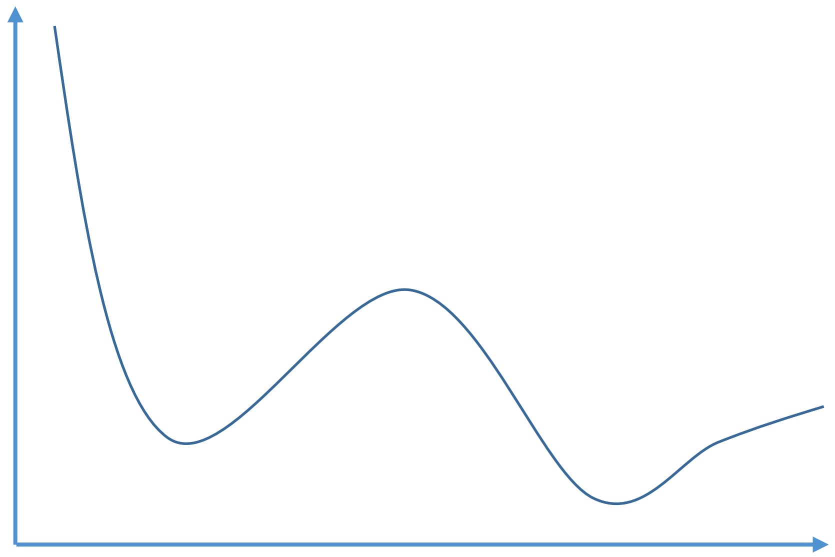
The Vanishing Gradient Problem

- When the slope of the learning curve approaches zero, things can get stuck
- We end up working with very small numbers that slow down training, or even introduce numerical errors
- Becomes a problem with deeper networks and RNN's as these "vanishing gradients" propagate to deeper layers
- Opposite problem: "exploding gradients"



Fixing the Vanishing Gradient Problem

- Multi-level heirarchy
  - Break up levels into their own sub-networks trained individually
- Long short-term memory (LSTM)
- Residual Networks
  - i.e., ResNet
  - Ensemble of shorter networks
- Better choice of activation function
  - ReLU is a good choice



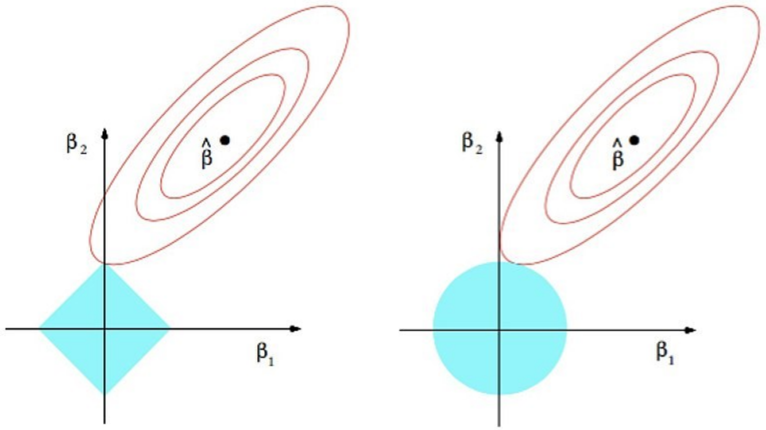
Gradient Checking

- A debugging technique
- Numerically check the derivatives computed during training
- Useful for validating code of neural network training
  - But you're probably not going to be writing this code…

L1 and L2 Regularization

L1 and L2 Regularization

- Preventing overfitting in ML in general
- A regularization term is added as weights are learned
- L1 term is the sum of the weights
  - $\lambda \sum_{i=1}^{k} |w_i|$
- L2 term is the sum of the square of the weights
  - $\lambda \sum_{i=1}^{k} w_i^2$
- Same idea can be applied to loss functions



What's the difference?

- L1: sum of weights
  - Performs *feature selection* – entire features go to 0
  - Computationally inefficient
  - Sparse output
- L2: sum of square of weights
  - All features remain considered, just weighted
  - Computationally efficient
  - Dense output

Why would you want L1?

- Feature selection can reduce dimensionality
  - Out of 100 features, maybe only 10 end up with non-zero coefficients!
  - The resulting sparsity can make up for its computational inefficiency
- But, if you think all of your features are important, L2 is probably a better choice.

Confusion Matrix

Sometimes accuracy doesn't tell the whole story

- A test for a rare disease can be 99.9% accurate by just guessing "no" all the time
- We need to understand true positives and true negative, as well as false positives and false negatives.
- A confusion matrix shows this.

Binary confusion matrix

	Actual YES	Actual NO
Predicted YES	TRUE POSITIVES	FALSE POSITIVES
Predicted NO	FALSE NEGATIVES	TRUE NEGATIVE

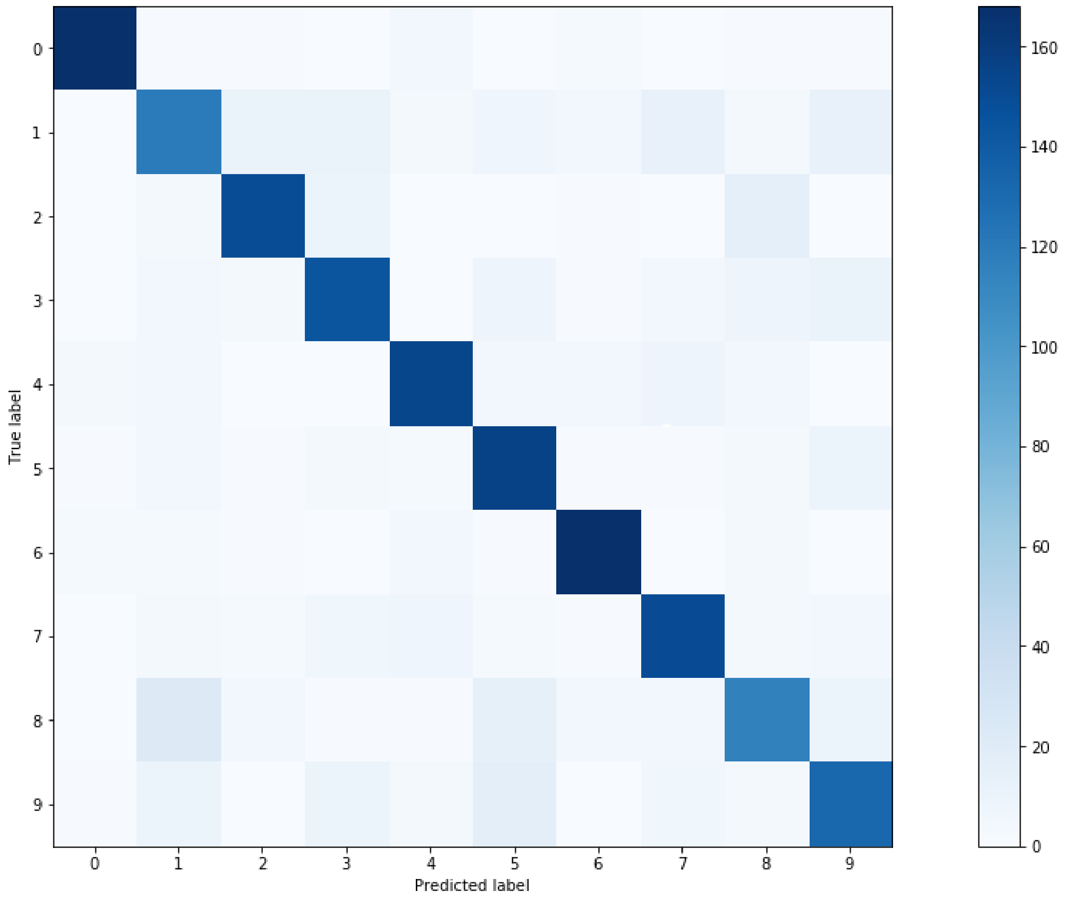
Image has cat?

	Actual cat	Actual not cat
Predicted cat	50	5
Predicted not cat	10	100

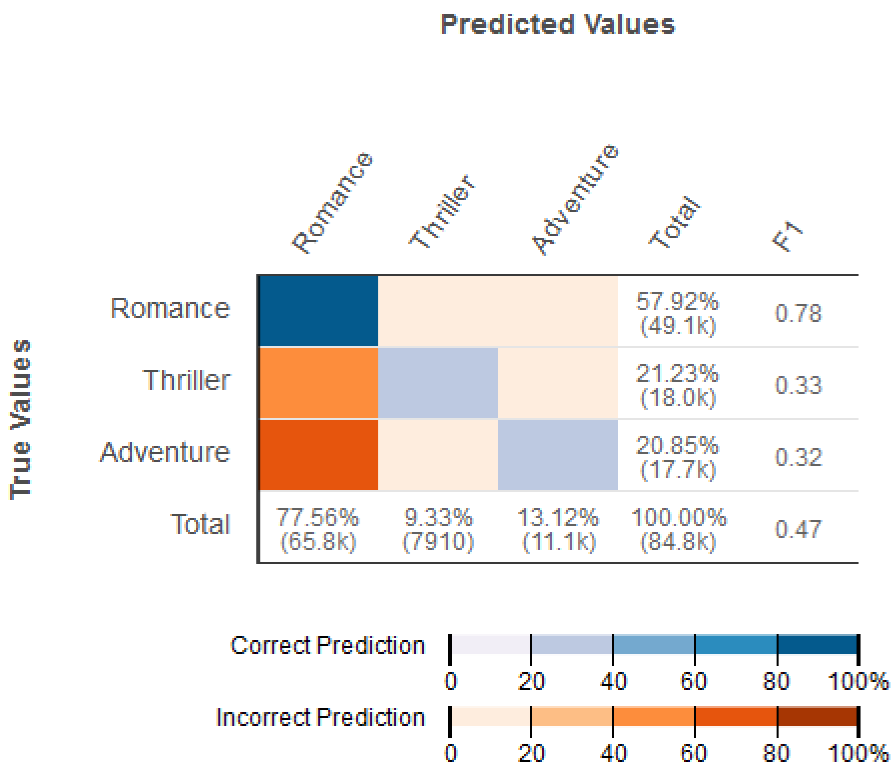
Another format

	Predicted NO	Predicted YES	
Actual NO	50	5	**55**
Actual YES	10	100	**110**
	60	**105**	

Multi-class confusion matrix + heat map



Example from the AWS docs



- # of correct and incorrect predictions per class (infer from colors of each cell)
- F1 scores per class
- True class frequencies: the "total" column
- Predicted class frequencies: the "total" row

DataCumulus
Sundog Education

Measuring your Models

Remember our friend the confusion matrix

	Actual YES	Actual NO
Predicted YES	TRUE POSITIVES	FALSE POSITIVES
Predicted NO	FALSE NEGATIVES	TRUE NEGATIVE

Recall

- $\frac{TRUE\ POSITIVES}{TRUE\ POSITIVES+FALSE\ NEGATIVES}$

- AKA Sensitivity, True Positive rate, Completeness
- Percent of positives rightly predicted
- Good choice of metric when you care a lot about false negatives
  - i.e., fraud detection

Recall example

	Actual fraud	Actual not fraud
Predicted fraud	5	20
Predicted not fraud	10	100

Recall = TP/(TP+FN)

Recall = 5/(5+10) = 5/15 = 1/3 = 33%

DataCumulus Sundog Education

Precision

- $\frac{TRUE\ POSITIVES}{TRUE\ POSITIVES + FALSE\ POSITIVES}$

- AKA Correct Positives
- Percent of relevant results
- Good choice of metric when you care a lot about false positives
  - i.e., medical screening, drug testing

DataCumulus Sundog Education

Precision example

	Actual fraud	Actual not fraud
Predicted fraud	5	20
Predicted not fraud	10	100

Precision = TP/(TP+FP)

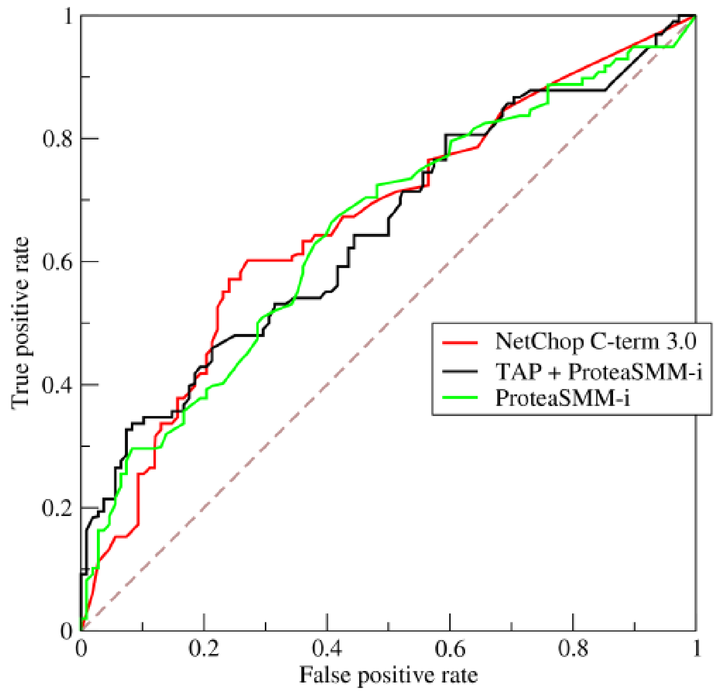
Precision = 5/(5+20) = 5/25 = 1/5 = 20%

Other metrics

- Specificity = $\frac{TN}{TN+FP}$ = "True negative rate"
- F1 Score
  - $\frac{2TP}{2TP+FP+FN}$
  - $2 \cdot \frac{Precision \cdot Recall}{Precision+Recall}$
  - Harmonic mean of precision and sensitivity
  - When you care about precision AND recall
- RMSE
  - Root mean squared error, exactly what it sounds like
  - Accuracy measurement
  - Only cares about right & wrong answers

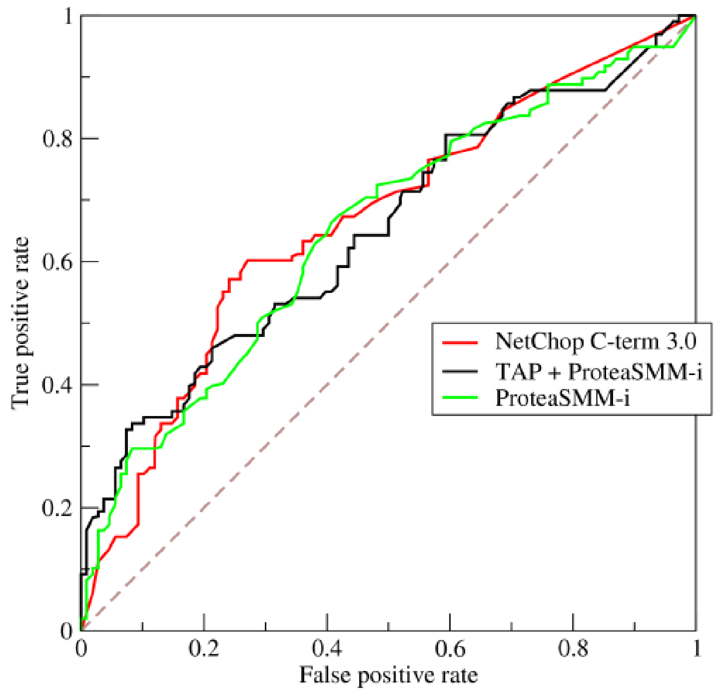
ROC Curve

- Receiver Operating Characteristic Curve
- Plot of true positive rate (recall) vs. false positive rate at various threshold settings.
- Points above the diagonal represent good classification (better than random)
- Ideal curve would just be a point in the upper-left corner
- The more it's "bent" toward the upper-left, the better



AUC

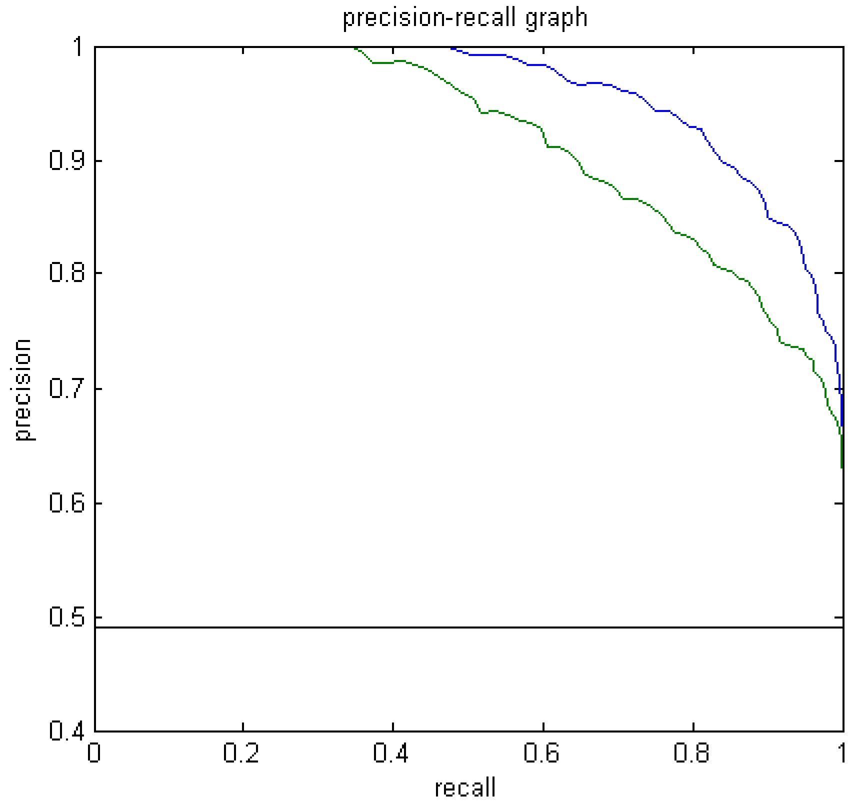
- The area under the ROC curve is… wait for it..
- Area Under the Curve (AUC)
- Equal to probability that a classifier will rank a randomly chosen positive instance higher than a randomly chosen negative one
- ROC AUC of 0.5 is a useless classifier, 1.0 is perfect
- Commonly used metric for comparing classifiers



DataCumulus Sundog Education

P-R Curve

- Precision / Recall curve
- Good = higher area under curve
- Similar to ROC curve
  - But better suited for information retrieval problems
  - ROC can result in very small values if you are searching large number of documents for a tiny number that are relevant

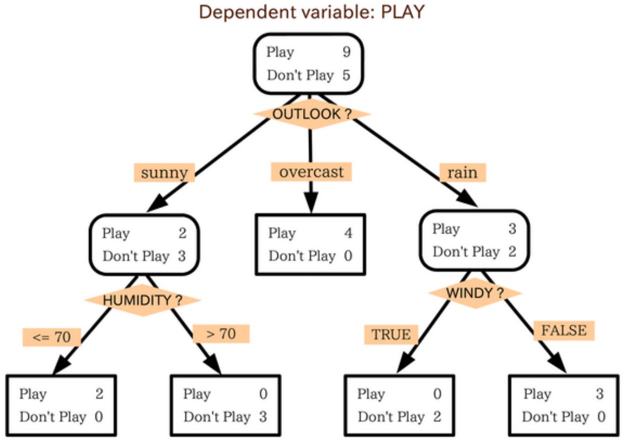


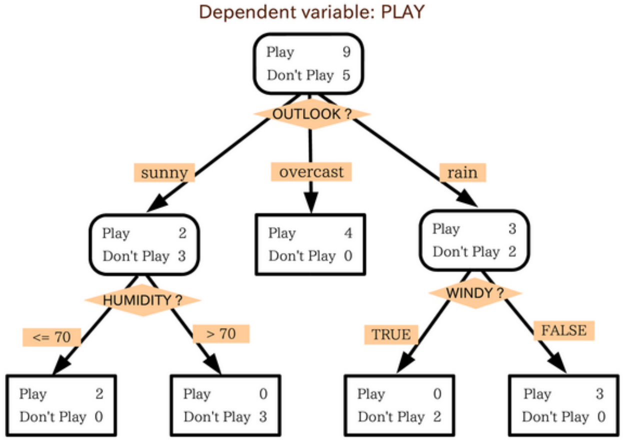
Ensemble Learning

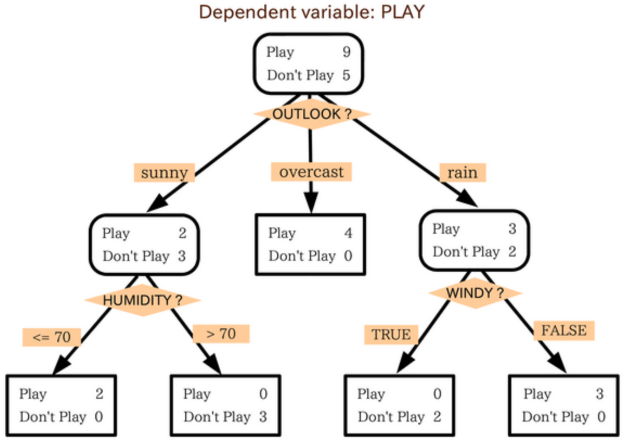
Bagging & Boosting

Ensemble methods

- Common example: random forest
  - Decision trees are prone to overfitting
  - So, make lots of decision trees and let them all vote on the result
  - This is a random forest
  - How do they differ?

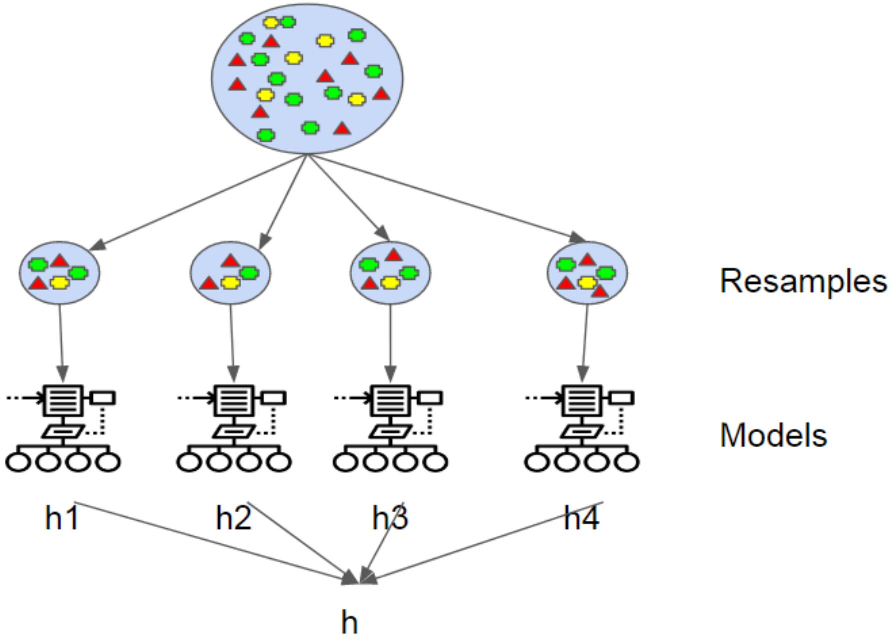






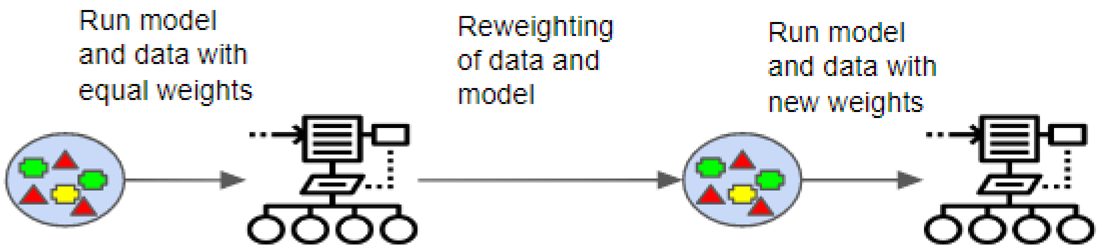
Bagging

- Generate N new training sets by **random sampling with replacement**
- Each resampled model can be trained in parallel



Boosting

- Observations are weighted
- Some will take part in new training sets more often
- Training is sequential; each classifier takes into account the previous one's success.



Bagging vs. Boosting

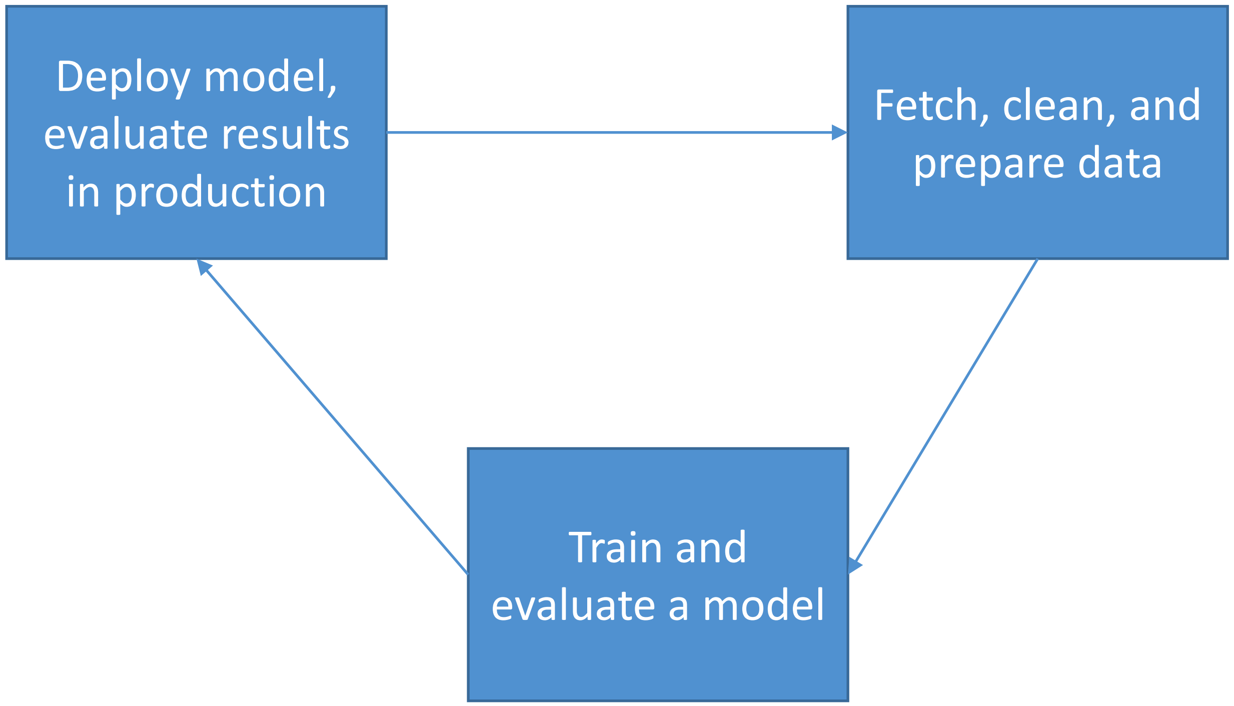
- XGBoost is the latest hotness
- Boosting generally yields better accuracy
- But bagging avoids overfitting
- Bagging is easier to parallelize
- So, depends on your goal

Modeling, Part 2: Amazon SageMaker

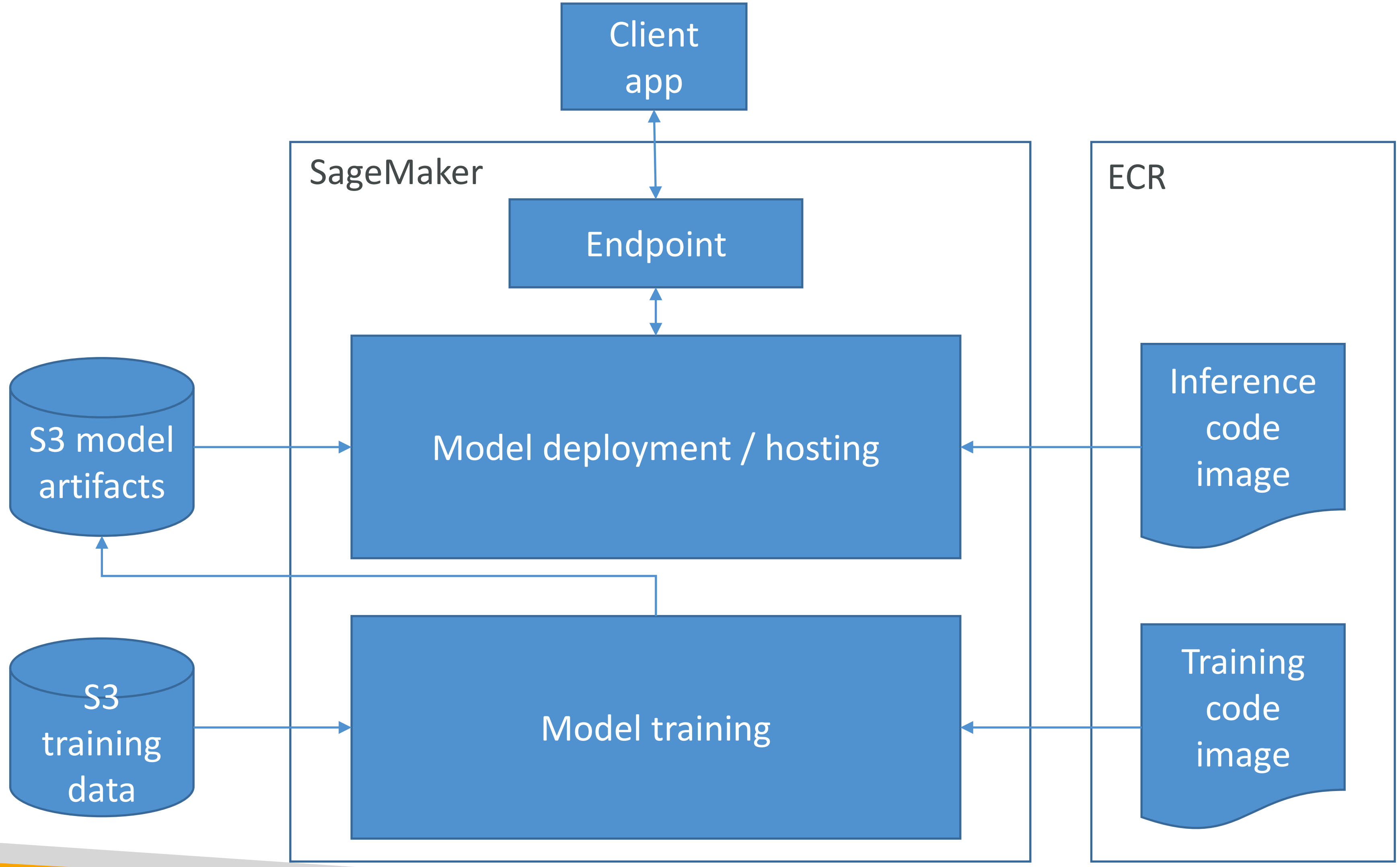
DataCumulus Sundog™ Education

Amazon SageMaker

SageMaker is built to handle the entire machine learning workflow.

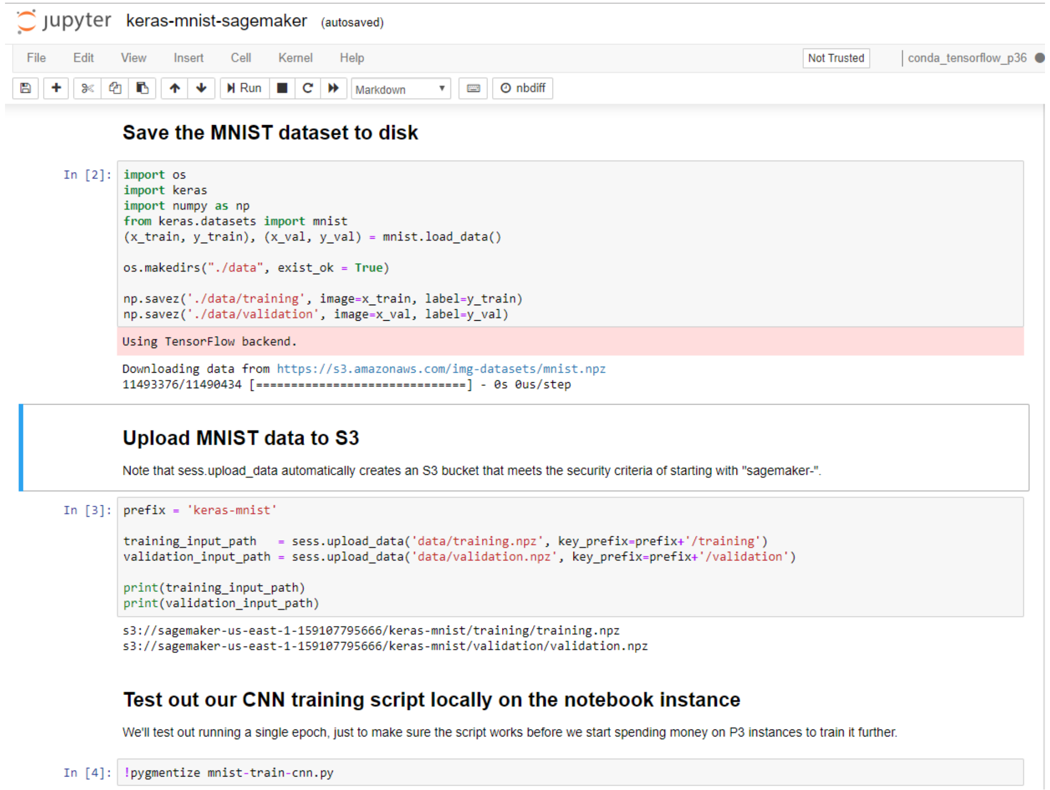


SageMaker Training & Deployment

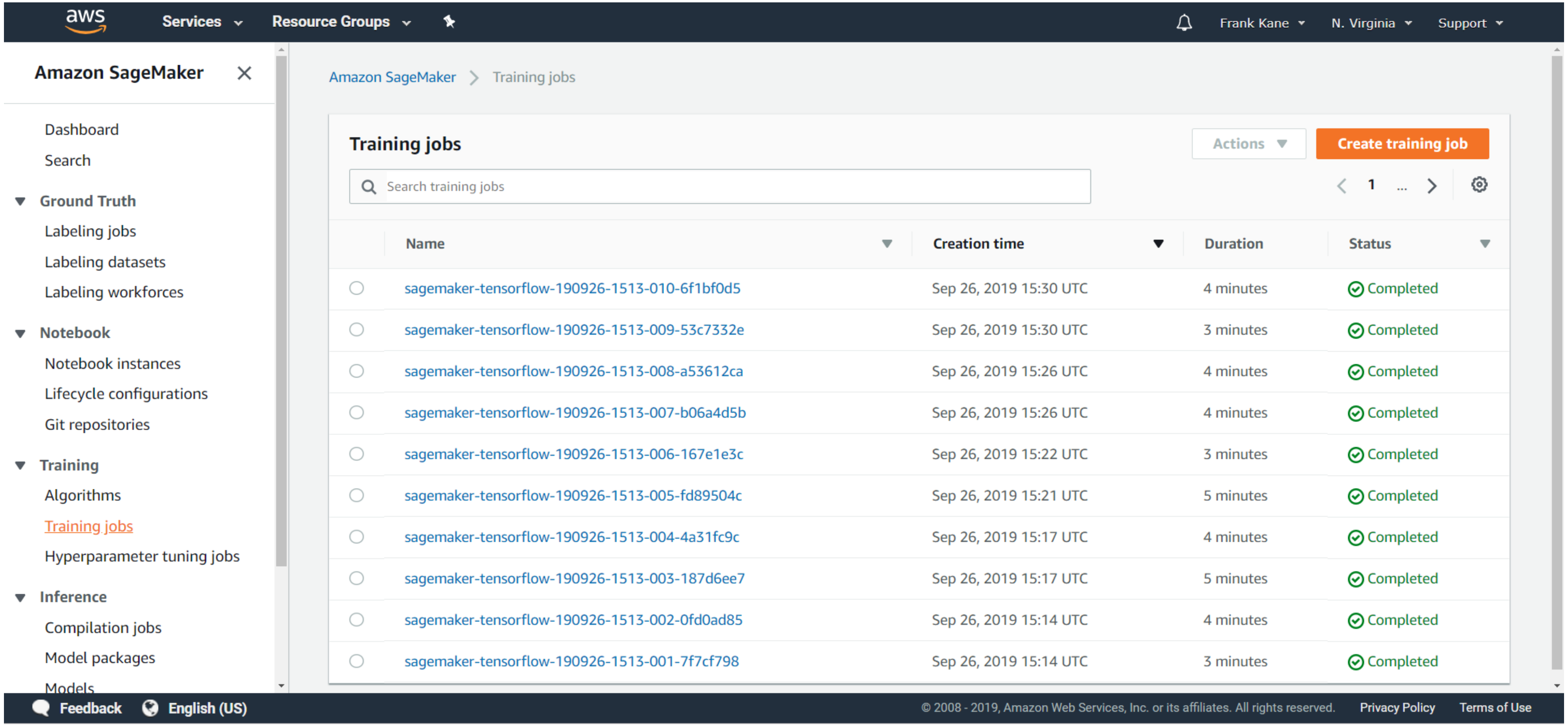


SageMaker Notebooks can direct the process

- Notebook Instances on EC2 are spun up from the console
  - S3 data access
  - Scikit_learn, Spark, Tensorflow
  - Wide variety of built-in models
  - Ability to spin up training instances
  - Abilty to deploy trained models for making predictions at scale



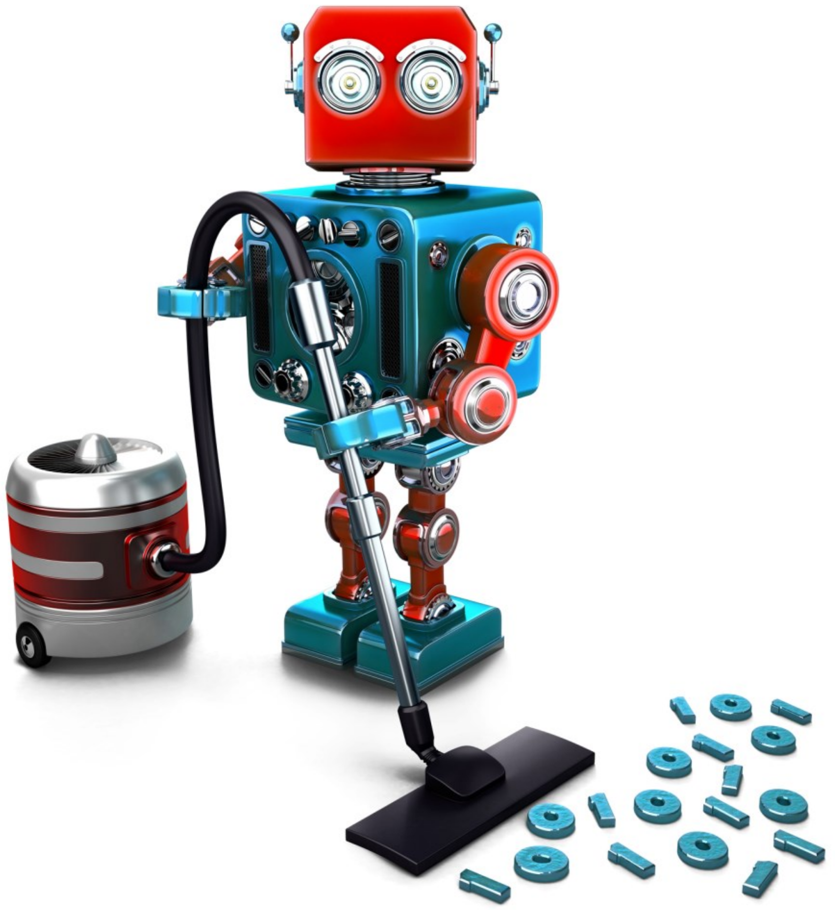
So can the SageMaker console



DataCumulus Sundog Education

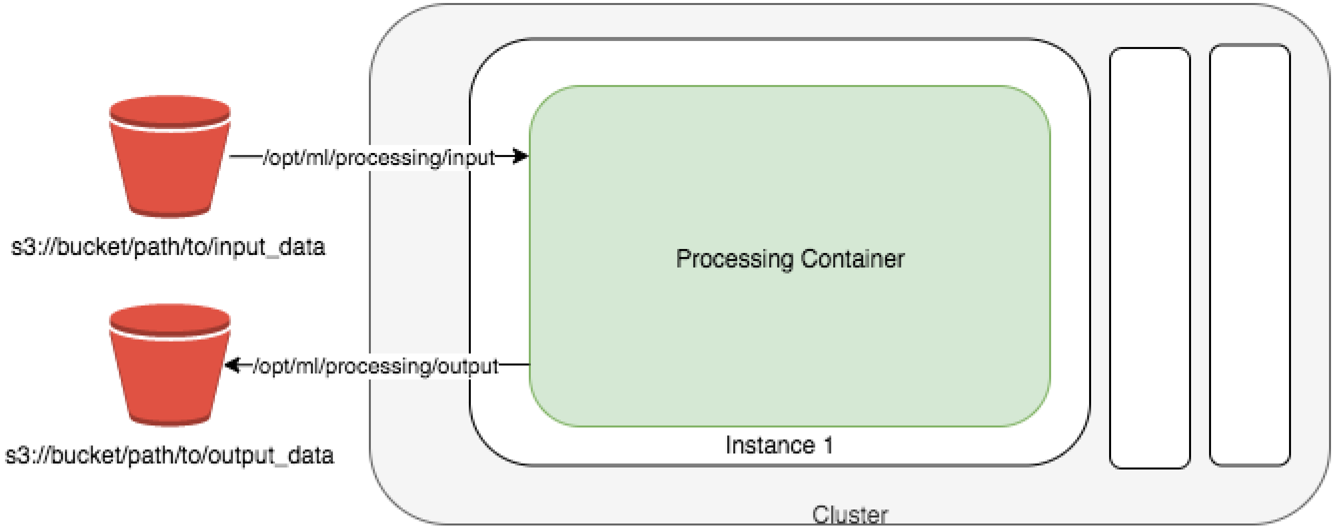
Data prep on SageMaker

- Data usually comes from S3
  - Ideal format varies with algorithm – often it is RecordIO / Protobuf
- Can also ingest from Athena, EMR, Redshift, and Amazon Keyspaces DB
- Apache Spark integrates with SageMaker
  - More on this later…
- Scikit_learn, numpy, pandas all at your disposal within a notebook



SageMaker Processing

- Processing jobs
  - Copy data from S3
  - Spin up a processing container
    - SageMaker built-in or user provided
  - Output processed data to S3



Training on SageMaker

- Create a training job
  - URL of S3 bucket with training data
  - ML compute resources
  - URL of S3 bucket for output
  - ECR path to training code
- Training options
  - Built-in training algorithms
  - Spark MLLib
  - Custom Python Tensorflow / MXNet code
  - PyTorch, Scikit-Learn, RLEstimator
  - XGBoost, Hugging Face, Chainer
  - Your own Docker image
  - Algorithm purchased from AWS marketplace

DataCumulus Sundog™ Education

Deploying Trained Models

- Save your trained model to S3
- Can deploy two ways:
  - Persistent endpoint for making individual predictions on demand
  - SageMaker Batch Transform to get predictions for an entire dataset
- Lots of cool options
  - Inference Pipelines for more complex processing
  - SageMaker Neo for deploying to edge devices
  - Elastic Inference for accelerating deep learning models
  - Automatic scaling (increase # of endpoints as needed)
  - Shadow Testing evaluates new models against currently deployed model to catch errors

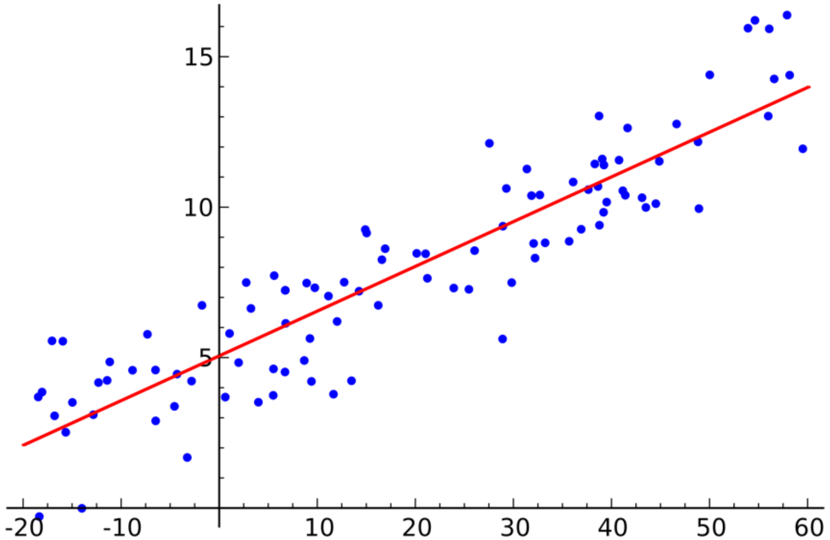
DataCumulus Sundog™ Education

SageMaker's Built-In Algorithms

Linear Learner

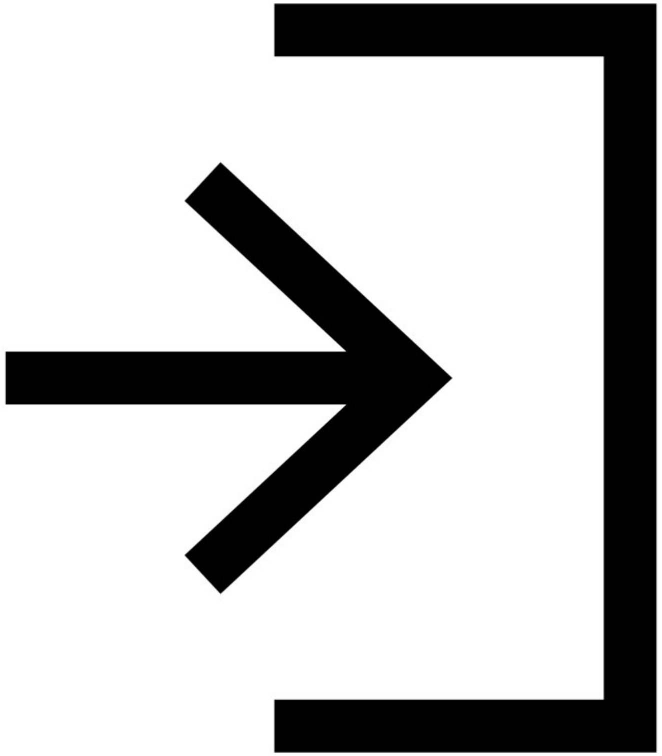
Linear Learner: What's it for?

- Linear regression
  - Fit a line to your training data
  - Predications based on that line
- Can handle both regression (numeric) predictions and classification predictions
  - For classification, a linear threshold function is used.
  - Can do binary or multi-class



Linear Learner: What training input does it expect?

- RecordIO-wrapped protobuf
  - Float32 data only!
- CSV
  - First column assumed to be the label
- File or Pipe mode both supported



Linear Learner: How is it used?

- Preprocessing
  - Training data must be *normalized* (so all features are weighted the same)
  - Linear Learner can do this for you automatically
  - Input data should be *shuffled*
- Training
  - Uses stochastic gradient descent
  - Choose an optimization algorithm (Adam, AdaGrad, SGD, etc)
  - Multiple models are optimized in parallel
  - Tune L1, L2 regularization
- Validation
  - Most optimal model is selected

Linear Learner: Important Hyperparameters

- Balance_multiclass_weights
  - Gives each class equal importance in loss functions
- Learning_rate, mini_batch_size
- L1
  - Regularization
- Wd
  - Weight decay (L2 regularization)
- target_precision
  - Use with binary_classifier_model_selection_criteria set to recall_at_target_precision
  - Holds precision at this value while maximizing recall
- target_recall
  - Use with binary_classifier_model_selection_criteria set to precision_at_target_recall
  - Holds recall at this value while maximizing precision

Linear Learner: Instance Types

- Training
  - Single or multi-machine CPU or GPU
  - Multi-GPU does not help

DataCumulus Sundog™ Education

Example

jupyter linear_learner_mnist (autosaved)

File Edit View Insert Cell Kernel Widgets Help Trusted conda_python3

Run Code nbdiff

Data inspection

Once the dataset is imported, it's typical as part of the machine learning process to inspect the data, understand the distributions, and determine what type(s) of preprocessing might be needed. You can perform those tasks right here in the notebook. As an example, let's go ahead and look at one of the digits that is part of the dataset.

In [3]:

```
%matplotlib inline
import matplotlib.pyplot as plt
plt.rcParams["figure.figsize"] = (2,10)


def show_digit(img, caption='', subplot=None):
    if subplot==None:
        _,(subplot)=plt.subplots(1,1)
    imgr=img.reshape((28,28))
    subplot.axis('off')
    subplot.imshow(imgr, cmap='gray')
    plt.title(caption)

show_digit(train_set[0][30], 'This is a {}'.format(train_set[1][30]))
```

This is a 3

Data conversion

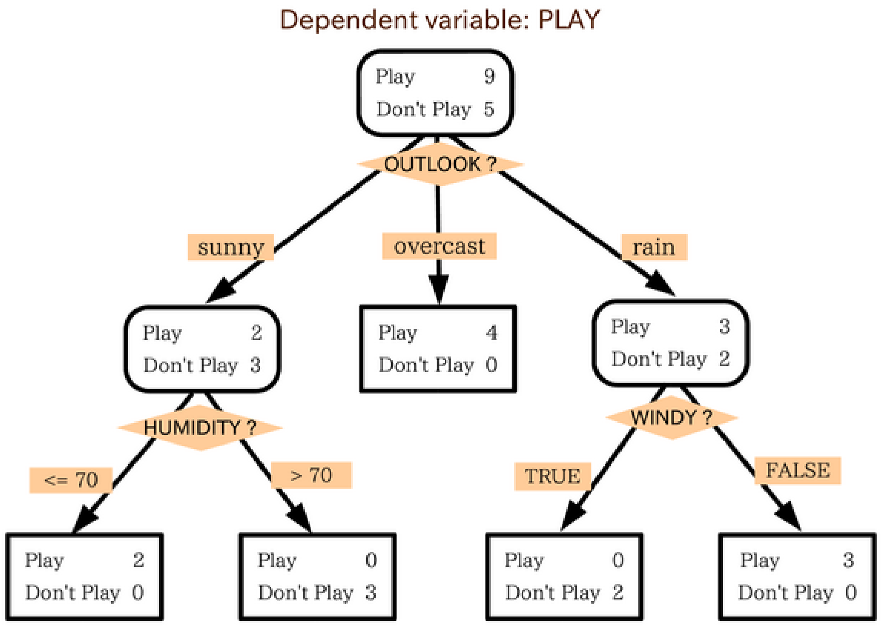
Since algorithms have particular input and output requirements, converting the dataset is also part of the process that a data scientist goes through prior to initiating training. In this particular case, the Amazon SageMaker implementation of Linear Learner takes recordIO-wrapped protobuf, where the data we have today is a pickle-ized numpy array on disk.

Most of the conversion effort is handled by the Amazon SageMaker Python SDK, imported as `sagemaker` below.

XGBoost

XGBoost: What's it for?

- eXtreme Gradient Boosting
  - Boosted group of decision trees
  - New trees made to correct the errors of previous trees
  - Uses gradient descent to minimize loss as new trees are added
- It's been winning a lot of Kaggle competitions
  - And it's fast, too
- Can be used for classification
- And also for regression
  - Using regression trees



XGBoost: What training input does it expect?

- XGBoost is weird, since it's not made for SageMaker. It's just open source XGBoost
- So, it takes CSV or libsvm input.
- AWS recently extended it to accept recordIO-protobuf and Parquet as well.

XGBoost: How is it used?

- Models are serialized/deserialized with Pickle
- Can use as a framework within notebooks
  - Sagemaker.xgboost
- Or as a built-in SageMaker algorithm

DataCumulus Sundog Education

XGBoost: Important Hyperparameters

- There are a lot of them. A few:
- Subsample
  - Prevents overfitting
- Eta
  - Step size shrinkage, prevents overfitting
- Gamma
  - Minimum loss reduction to create a partition; larger = more conservative
- Alpha
  - L1 regularization term; larger = more conservative
- Lambda
  - L2 regularization term; larger = more conservative

XGBoost: Important Hyperparameters

- eval_metric
  - Optimize on AUC, error, rmse…
  - For example, if you care about false positives more than accuracy, you might use AUC here
- scale_pos_weight
  - Adjusts balance of positive and negative weights
  - Helpful for unbalanced classes
  - Might set to sum(negative cases) / sum(positive cases)
- max_depth
  - Max depth of the tree
  - Too high and you may overfit

XGBoost: Instance Types

- Is memory-bound, not compute-bound
- So, **M5** is a good choice
- As of XGBoost 1.2, single-instance GPU training is available
  - For **example P2, P3**
  - Must set tree_method hyperparameter to gpu_hist
  - Trains more quickly and can be more cost effective.
- XGBoost 1.2-2
  - P2, P3, G4dn, G5
- XGBoost 1.5+: Distributed GPU training now available
  - Must set use_dask_gpu_training to true
  - Set distribution to fully_replicated in TrainingInput
  - Only works with csv or parquet input

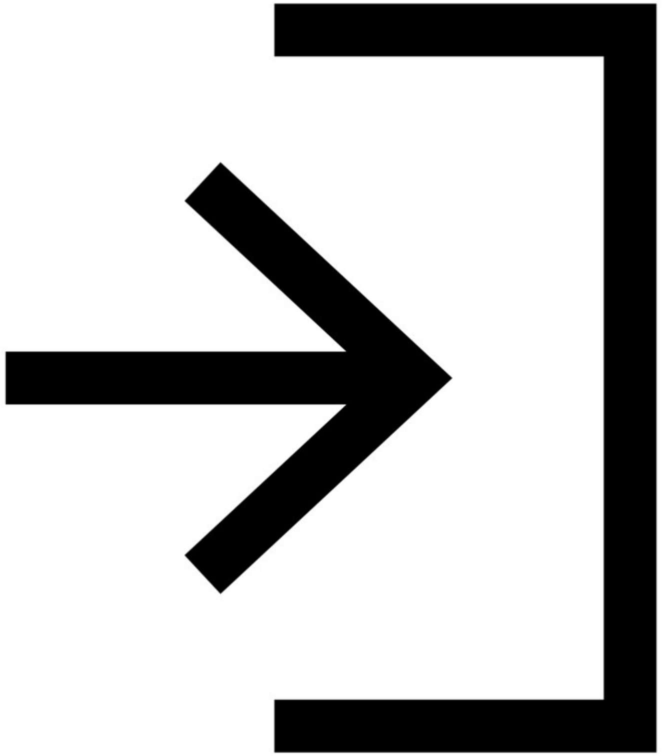
Seq2Seq

Seq2Seq: What's it for?

- Input is a sequence of tokens, output is a sequence of tokens
- Machine Translation
- Text summarization
- Speech to text
- Implemented with RNN's and CNN's with attention

Seq2Seq: What training input does it expect?

- RecordIO-Protobuf
  - Tokens must be integers (this is unusual, since most algorithms want floating point data.)
- Start with tokenized text files
- Convert to protobuf using sample code
  - Packs into integer tensors with vocabulary files
  - A lot like the TF/IDF lab we did earlier.
- Must provide training data, validation data, and vocabulary files.



Seq2Seq: How is it used?

- Training for machine translation can take days, even on SageMaker
- Pre-trained models are available
  - See the example notebook
- Public training datasets are available for specific translation tasks

Seq2Seq: Important Hyperparameters

- Batch_size
- Optimizer_type (adam, sgd, rmsprop)
- Learning_rate
- Num_layers_encoder
- Num_layers_decoder
- Can optimize on:
  - Accuracy
    - Vs. provided validation dataset
  - BLEU score
    - Compares against multiple reference translations
  - Perplexity
    - Cross-entropy

Seq2Seq: Instance Types

- Can only use GPU instance types (P3 for example)
- Can only use a single machine for training
  - But can use multi-GPU's on one machine

DeepAR

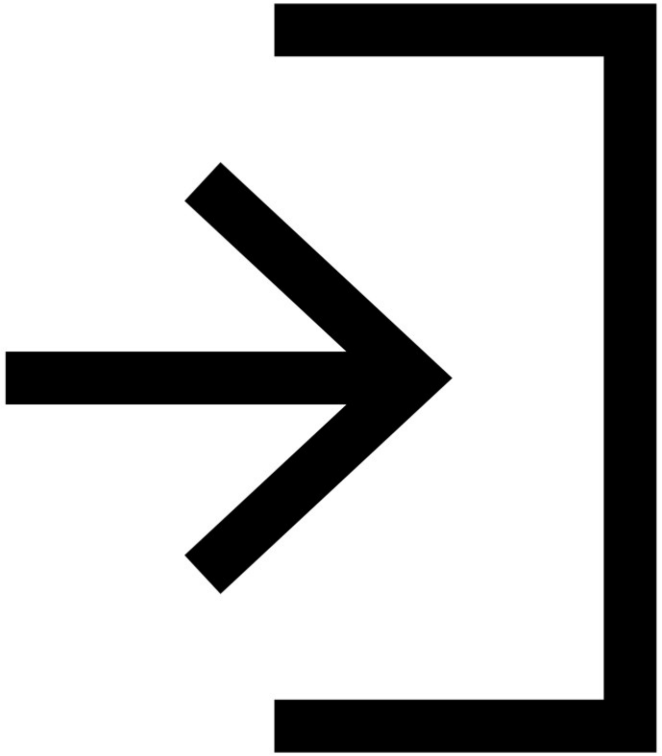
DeepAR: What's it for?

- Forecasting one-dimensional time series data
- Uses RNN's
- Allows you to train the same model over several related time series
- Finds frequencies and seasonality

DataCumulus Sundog™ Education

DeepAR: What training input does it expect?

- JSON lines format
  - Gzip or Parquet
- Each record must contain:
  - Start: the starting time stamp
  - Target: the time series values
- Each record can contain:
  - Dynamic_feat: dynamic features (such as, was a promotion applied to a product in a time series of product purchases)
  - Cat: categorical features



```
{"start": "2009-11-01 00:00:00", "target": [4.3, "NaN", 5.1, ...], "cat": [0, 1], "dynamic_feat": [[1.1, 1.2, 0.5, ...]]}
{"start": "2012-01-30 00:00:00", "target": [1.0, -5.0, ...], "cat": [2, 3], "dynamic_feat": [[1.1, 2.05, ...]]}
{"start": "1999-01-30 00:00:00", "target": [2.0, 1.0], "cat": [1, 4], "dynamic_feat": [[1.3, 0.4]]}
```

DeepAR: How is it used?

- Always include entire time series for training, testing, and inference
- Use entire dataset as training set, remove last time points for testing. Evaluate on withheld values.
- Don’t use very large values for prediction length (> 400)
- Train on many time series and not just one when possible

DeepAR: Important Hyperparameters

- Context_length
  - Number of time points the model sees before making a prediction
  - Can be smaller than seasonalities; the model will lag one year anyhow.
- Epochs
- mini_batch_size
- Learning_rate
- Num_cells

DataCumulus Sundog Education

DeepAR: Instance Types

- Can use CPU or GPU
- Single or multi machine
- Start with CPU (ml.c4.2xlarge, ml.c4.4xlarge)
- Move up to GPU if necessary
  - Only helps with larger models
  - Or with large mini-batch sizes (>512)
- CPU-only for inference
- May need larger instances for tuning

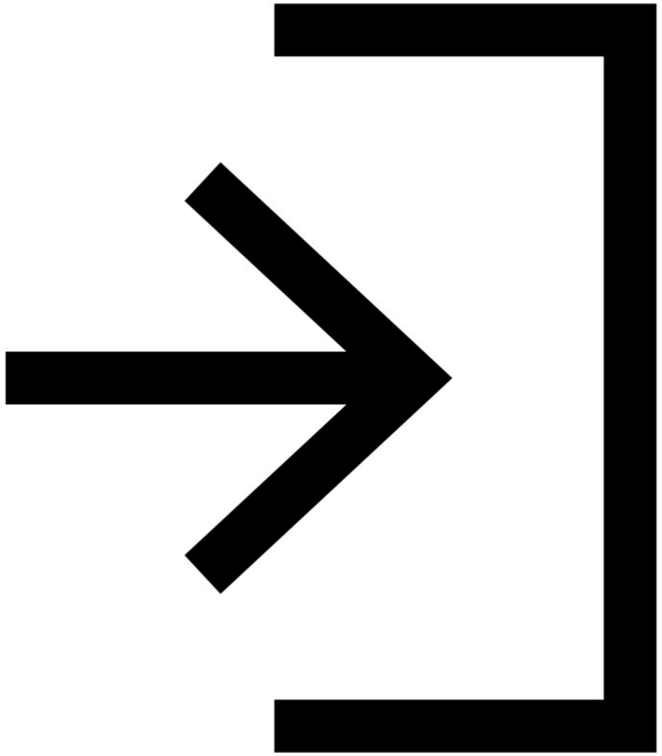
BlazingText

BlazingText: What's it for?

- Text classification
  - Predict labels for a sentence
  - Useful in web searches, information retrieval
  - Supervised
- Word2vec
  - Creates a vector representation of words
  - Semantically similar words are represented by vectors close to each other
  - This is called a *word embedding*
  - It is useful for NLP, but is not an NLP algorithm in itself!
    - Used in machine translation, sentiment analysis
  - Remember it only works on individual words, not sentences or documents

BlazingText: What training input does it expect?

- For supervised mode (text classification):
  - One sentence per line
  - First "word" in the sentence is the string __label__ followed by the label
- Also, "augmented manifest text format"
- Word2vec just wants a text file with one training sentence per line.



__label__4 linux ready for prime time , intel says , despite all the linux hype , the open-source movement has yet to make a huge splash in the desktop market . that may be about to change , thanks to chipmaking giant intel corp .

__label__2 bowled by the slower one again , kolkata , november 14 the past caught up with sourav ganguly as the indian skippers return to international cricket was short lived .

{"source":"linux ready for prime time , intel says , despite all the linux hype", "label":1}
{"source":"bowled by the slower one again , kolkata , november 14 the past caught up with sourav ganguly", "label":2}

BlazingText: How is it used?

- Word2vec has multiple modes
  - Cbow (Continuous Bag of Words)
  - Skip-gram
  - Batch skip-gram
    - Distributed computation over many CPU nodes

BlazingText: Important Hyperparameters

- Word2vec:
  - Mode (batch_skipgram, skipgram, cbow)
  - Learning_rate
  - Window_size
  - Vector_dim
  - Negative_samples
- Text classification:
  - Epochs
  - Learning_rate
  - Word_ngrams
  - Vector_dim

BlazingText: Instance Types

- For cbow and skipgram, recommend a single ml.p3.2xlarge
  - Any single CPU or single GPU instance will work
- For batch_skipgram, can use single or multiple CPU instances
- For text classification, C5 recommended if less than 2GB training data. For larger data sets, use a single GPU instance (ml.p2.xlarge or ml.p3.2xlarge)

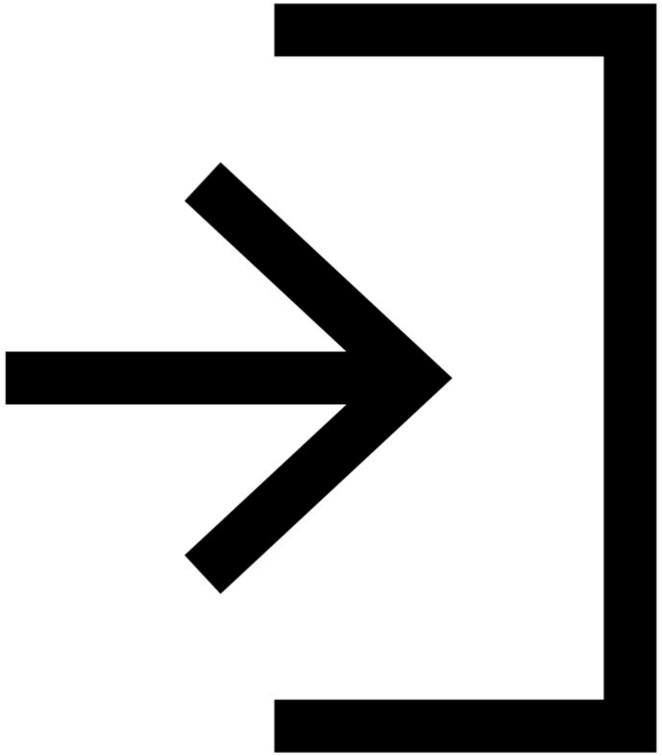
Object2Vec

Object2Vec: What's it for?

- Remember word2vec from Blazing Text? It's like that, but arbitrary objects
- It creates low-dimensional dense embeddings of high-dimensional objects
- It is basically word2vec, generalized to handle things other than words.
- Compute nearest neighbors of objects
- Visualize clusters
- Genre prediction
- Recommendations (similar items or users)

Object2Vec: What training input does it expect?

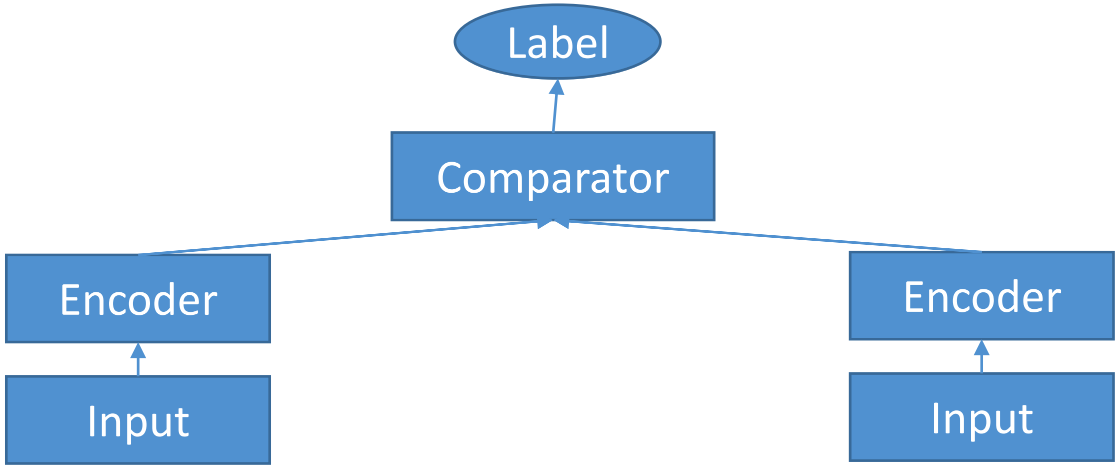
- Data must be tokenized into integers
- Training data consists of pairs of tokens and/or sequences of tokens
  - Sentence – sentence
  - Labels-sequence (genre to description?)
  - Customer-customer
  - Product-product
  - User-item



{"label": 0, "in0": [6, 17, 606, 19, 53, 67, 52, 12, 5, 10, 15, 10178, 7, 33, 652, 80, 15, 69, 821, 4], "in1": [16, 21, 13, 45, 14, 9, 80, 59, 164, 4]}
{"label": 1, "in0": [22, 1016, 32, 13, 25, 11, 5, 64, 573, 45, 5, 80, 15, 67, 21, 7, 9, 107, 4], "in1": [22, 32, 13, 25, 1016, 573, 3252, 4]}
{"label": 1, "in0": [774, 14, 21, 206], "in1": [21, 366, 125]}

Object2Vec: How is it used?

- Process data into JSON Lines and shuffle it
- Train with two input channels, two encoders, and a comparator
- Encoder choices:
  - Average-pooled embeddings
  - CNN's
  - Bidirectional LSTM
- Comparator is followed by a feed-forward neural network



DataCumulus Sundog Education

Object2Vec: Important Hyperparameters

- The usual deep learning ones…
  - Dropout, early stopping, epochs, learning rate, batch size, layers, activation function, optimizer, weight decay
- Enc1_network, enc2_network
  - Choose hcnn, bilstm, pooled_embedding

DataCumulus Sundog Education

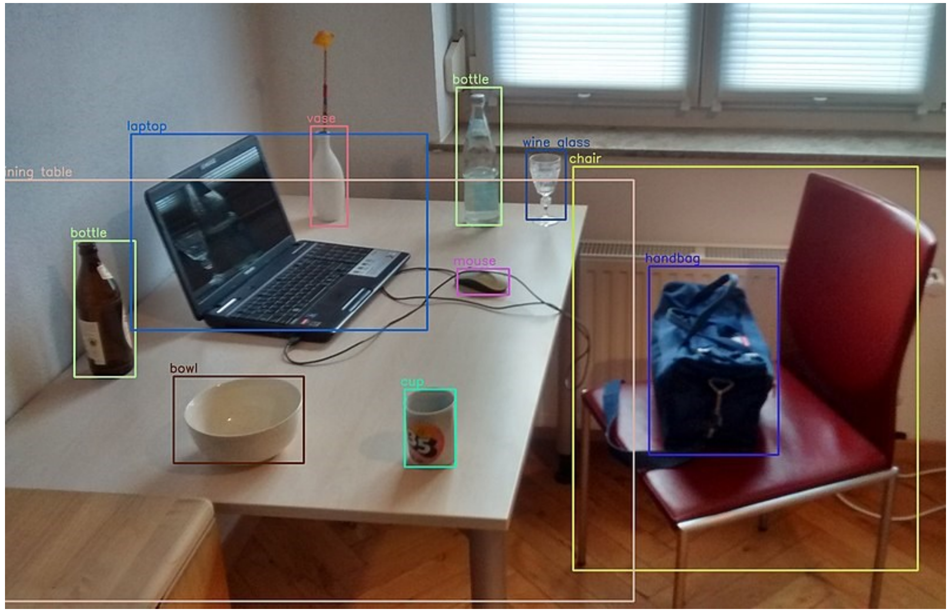
Object2Vec: Instance Types

- Can only train on a single machine (CPU or GPU, multi-GPU OK)
  - ml.m5.2xlarge
  - ml.p2.xlarge
  - If needed, go up to ml.m5.4xlarge or ml.m5.12xlarge
  - GPU options: P2, P3, G4dn, G5
- Inference: use ml.p3.2xlarge
  - Use INFERENCE_PREFERRED_MODE environment variable to optimize for encoder embeddings rather than classification or regression.

Object Detection

Object Detection: What's it for?

- Identify all objects in an image with bounding boxes
- Detects and classifies objects with a single deep neural network
- Classes are accompanied by confidence scores
- Can train from scratch, or use pre-trained models based on ImageNet



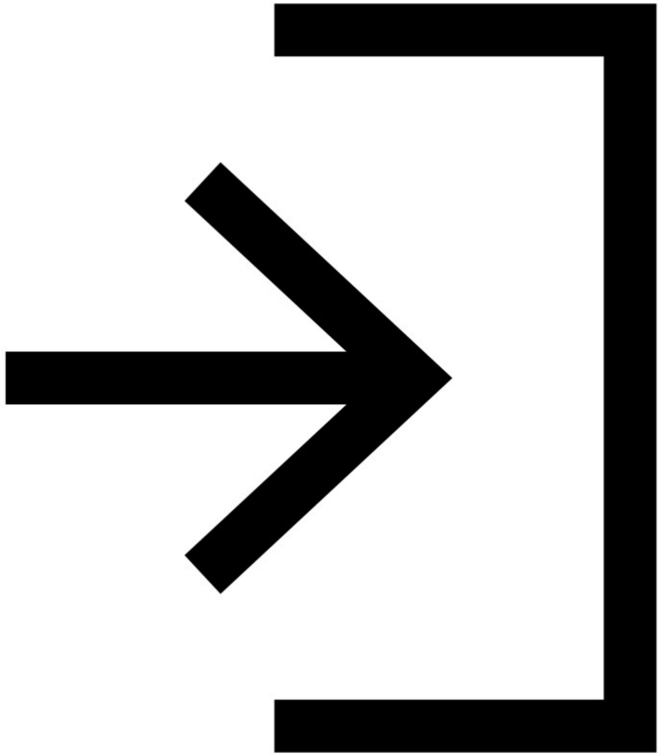
Object Detection: How is it used?

- Two variants: MXNet and Tensorflow
- Takes an image as input, outputs all instances of objects in the image with categories and confidence scores
- MXNet
  - Uses a CNN with the Single Shot multibox Detector (SSD) algorithm
    - The base CNN can be VGG-16 or ResNet-50
  - Transfer learning mode / incremental training
    - Use a pre-trained model for the base network weights instead of random initial weights
  - Uses flip, rescale, and jitter internally to avoid overfitting
- Tensorflow
  - Uses ResNet, EfficientNet, MobileNet models from the TensorFlow Model Garden

Object Detection: What training input does it expect?

- MXNet: RecordIO or image format (jpg or png)
- With image format, supply a JSON file for annotation data for each image

```
{
  "file": "your_image_directory/sample_image1.jpg",
  "image_size": [
    {
      "width": 500,
      "height": 400,
      "depth": 3
    }
  ],
  "annotations": [
    {
      "class_id": 0,
      "left": 111,
      "top": 134,
      "width": 61,
      "height": 128
    },
],
  "categories": [
    {
      "class_id": 0,
      "name": "dog"
    },
]
}
```



DataCumulus Sundog Education

Object Detection: Important Hyperparameters

- Mini_batch_size
- Learning_rate
- Optimizer
  - Sgd, adam, rmsprop, adadelta

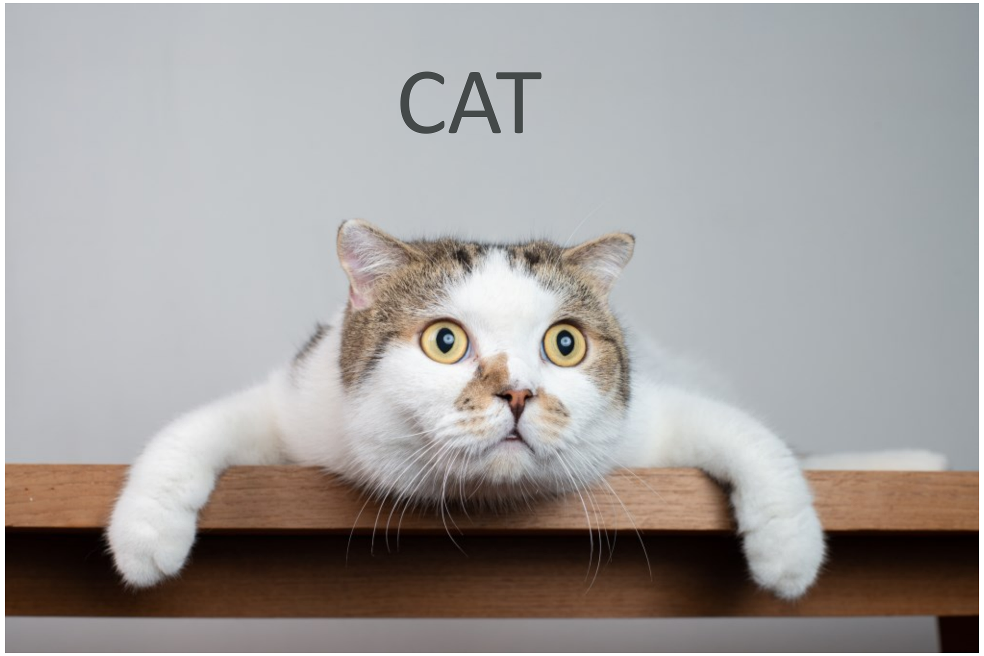
Object Detection: Instance Types

- Use GPU instances for training (multi-GPU and multi-machine OK)
  - ml.p2.xlarge, ml.p2.16xlarge, ml.p3.2xlarge, ml.p3.16xlarge, G4dn, G5
- Use CPU or GPU for inference
  - M5, P2, P3, G4dn all OK

Image Classification

Image Classification: What's it for?

- Assign one or more labels to an image
- Doesn't tell you where objects are, just what objects are in the image



DataCumulus Sundog™ Education

Image Classification: How is it used?

- Separate algorithms for MXNet and Tensorflow
- MXNet:
  - Full training mode
    - Network initialized with random weights
  - Transfer learning mode
    - Initialized with pre-trained weights
    - The top fully-connected layer is initialized with random weights
    - Network is fine-tuned with new training data
  - Default image size is 3-channel 224x224 (ImageNet's dataset)
- Tensorflow: Uses various Tensorflow Hub models (MobileNet, Inception, ResNet, EfficientNet)
  - Top classification layer is available for fine tuning or further training

DataCumulus Sundog Education

Image Classification: Important Hyperparameters

- The usual suspects for deep learning
  - Batch size, learning rate, optimizer
- Optimizer-specific parameters
  - Weight decay, beta 1, beta 2, eps, gamma
  - Slightly different between MXNet and Tensorflow versions

DataCumulus Sundog Education

Image Classification: Instance Types

- GPU instances for training (ml.p2, p3, g4dn, g5) Multi-GPU and multi-machine OK.
- CPU or GPU for inference (m5, p2, p3, g4dn, g5)

Semantic Segmentation

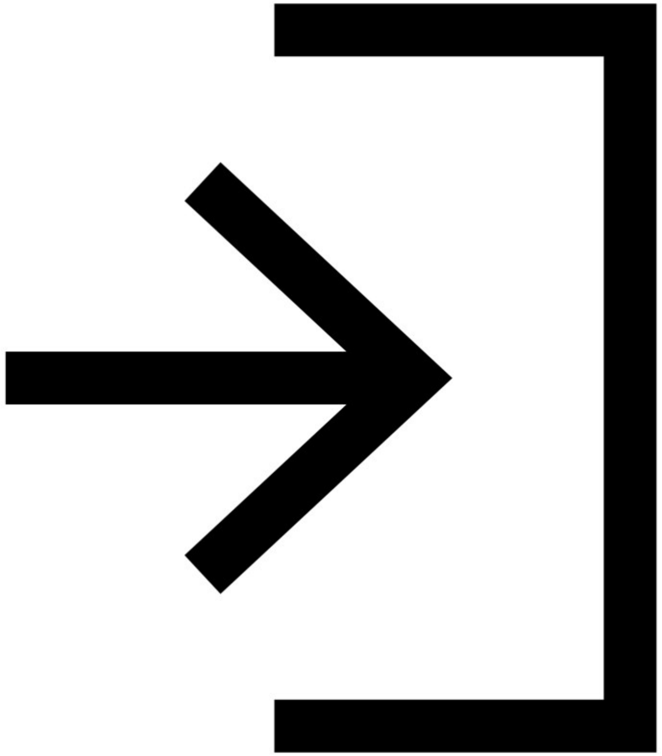
Semantic Segmentation: What's it for?

- Pixel-level object classification
- Different from image classification – that assigns labels to whole images
- Different from object detection – that assigns labels to bounding boxes
- Useful for self-driving vehicles, medical imaging diagnostics, robot sensing
- Produces a *segmentation mask*

DataCumulus Sundog™ Education

Semantic Segmentation: What training input does it expect?

- JPG Images and PNG annotations
- For both training and validation
- Label maps to describe annotations
- Augmented manifest image format supported for Pipe mode.
- JPG images accepted for inference



Semantic Segmentation: How is it used?

- Built on MXNet Gluon and Gluon CV
- Choice of 3 algorithms:
  - Fully-Convolutional Network (FCN)
  - Pyramid Scene Parsing (PSP)
  - DeepLabV3
- Choice of backbones:
  - ResNet50
  - ResNet101
  - Both trained on ImageNet
- Incremental training, or training from scratch, supported too

Semantic Segmentation: Important Hyperparameters

- Epochs, learning rate, batch size, optimizer, etc
- Algorithm
- Backbone

DataCumulus Sundog™ Education

Image Classification: Instance Types

- GPU instances for training (ml.p2, p3, g4dn, g5) Multi-GPU and multi-machine OK.
- CPU or GPU for inference (m5, p2, p3, g4dn, g5)

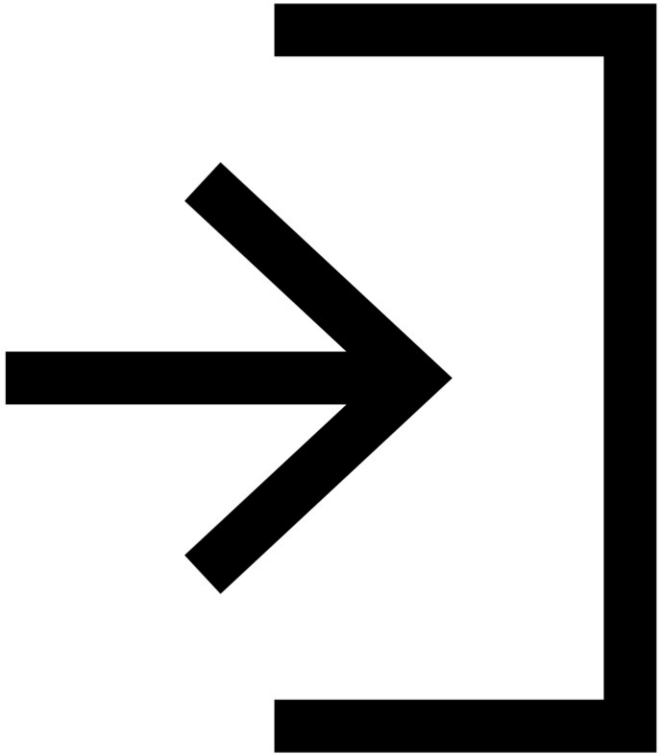
Random Cut Forest

Random Cut Forest: What's it for?

- Anomaly detection
- Unsupervised
- Detect unexpected spikes in time series data
- Breaks in periodicity
- Unclassifiable data points
- Assigns an anomaly score to each data point
- Based on an algorithm developed by Amazon that they seem to be very proud of!

Random Cut Forest: What training input does it expect?

- RecordIO-protobuf or CSV
- Can use File or Pipe mode on either
- Optional test channel for computing accuracy, precision, recall, and F1 on labeled data (anomaly or not)



Random Cut Forest: How is it used?

- Creates a forest of trees where each tree is a partition of the training data; looks at expected change in complexity of the tree as a result of adding a point into it
- Data is sampled randomly
- Then trained
- RCF shows up in Kinesis Analytics as well; it can work on streaming data too.

Random Cut Forest: Important Hyperparameters

- Num_trees
  - Increasing reduces noise
- Num_samples_per_tree
  - Should be chosen such that 1/num_samples_per_tree approximates the ratio of anomalous to normal data

DataCumulus
Sundog Education

Random Cut Forest: Instance Types

- Does not take advantage of GPUs
- Use M4, C4, or C5 for training
- ml.c5.xl for inference

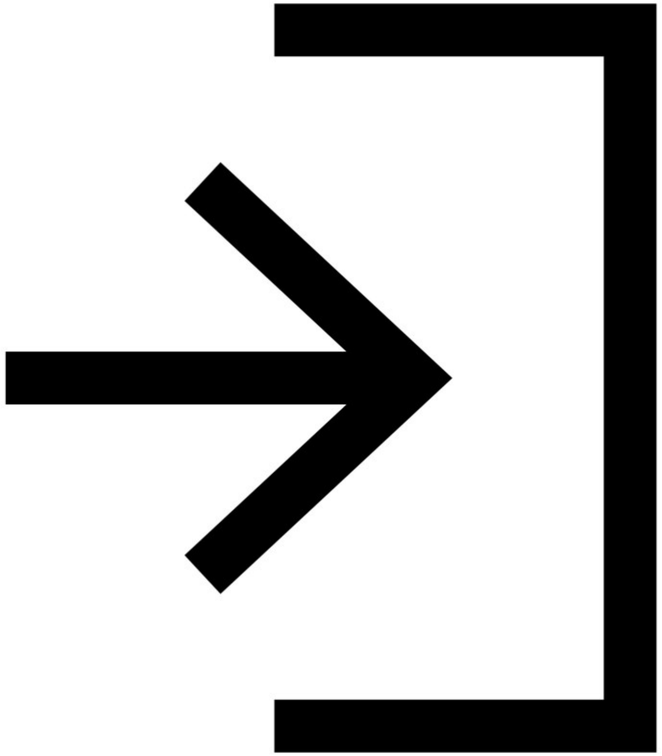
Neural Topic Model

Neural Topic Model: What's it for?

- Organize documents into topics
- Classify or summarize documents based on topics
- It's not just TF/IDF
  - "bike", "car", "train", "mileage", and "speed" might classify a document as "transportation" for example (although it wouldn't know to call it that)
- Unsupervised
  - Algorithm is "Neural Variational Inference"

Neural Topic Model: What training input does it expect?

- Four data channels
  - “train” is required
  - “validation”, “test”, and “auxiliary” optional
- recordIO-protobuf or CSV
- Words must be tokenized into integers
  - Every document must contain a count for every word in the vocabulary in CSV
  - The “auxiliary” channel is for the vocabulary
- File or pipe mode



Neural Topic Model: How is it used?

- You define how many topics you want
- These topics are a latent representation based on top ranking words
- One of two topic modeling algorithms in SageMaker – you can try them both!

Neural Topic Model: Important Hyperparameters

- Lowering mini_batch_size and learning_rate can reduce validation loss
  - At expense of training time
- Num_topics

Neural Topic Model: Instance Types

- GPU or CPU
  - GPU recommended for training
  - CPU OK for inference
  - CPU is cheaper

DataCumulus Sundog™ Education

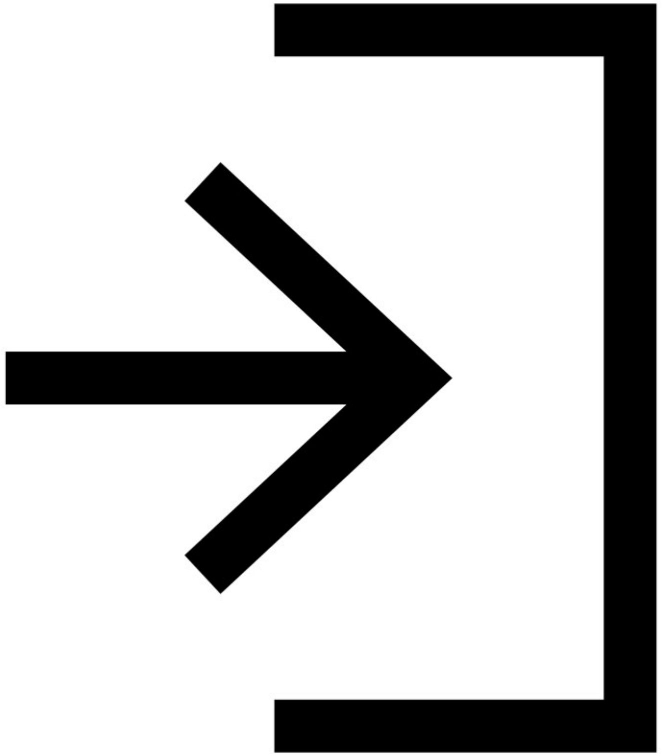
LDA

LDA: What's it for?

- Latent Dirichlet Allocation
- Another topic modeling algorithm
  - Not deep learning
- Unsupervised
  - The topics themselves are unlabeled; they are just groupings of documents with a shared subset of words
- Can be used for things other than words
  - Cluster customers based on purchases
  - Harmonic analysis in music

LDA: What training input does it expect?

- Train channel, optional test channel
- recordIO-protobuf or CSV
- Each document has counts for every word in vocabulary (in CSV format)
- Pipe mode only supported with recordIO



DataCumulus Sundog Education

LDA: How is it used?

- Unsupervised; generates however many topics you specify
- Optional test channel can be used for scoring results
  - Per-word log likelihood
- Functionally similar to NTM, but CPU-based
  - Therefore maybe cheaper / more efficient

LDA: Important Hyperparameters

- Num_topics
- Alpha0
  - Initial guess for concentration parameter
  - Smaller values generate sparse topic mixtures
  - Larger values (>1.0) produce uniform mixtures

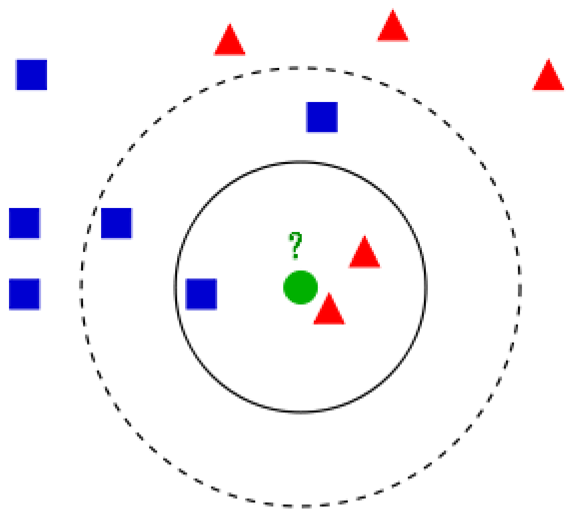
LDA: Instance Types

- Single-instance CPU training

KNN

KNN: What's it for?

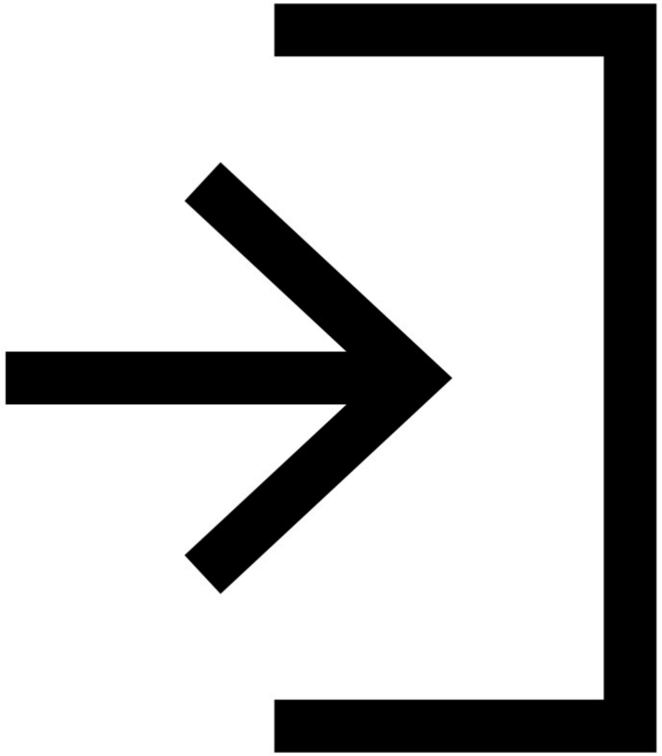
- K-Nearest-Neighbors
- Simple classification or regression algorithm
- Classification
  - Find the K closest points to a sample point and return the most frequent label
- Regression
  - Find the K closest points to a sample point and return the average value



DataCumulus Sundog™ Education

KNN: What training input does it expect?

- Train channel contains your data
- Test channel emits accuracy or MSE
- recordIO-protobuf or CSV training
  - First column is label
- File or pipe mode on either



KNN: How is it used?

- Data is first sampled
- SageMaker includes a dimensionality reduction stage
  - Avoid sparse data (“curse of dimensionality”)
  - At cost of noise / accuracy
  - “sign” or “fjlt” methods
- Build an index for looking up neighbors
- Serialize the model
- Query the model for a given K

KNN: Important Hyperparameters

- K!
- Sample_size

KNN: Instance Types

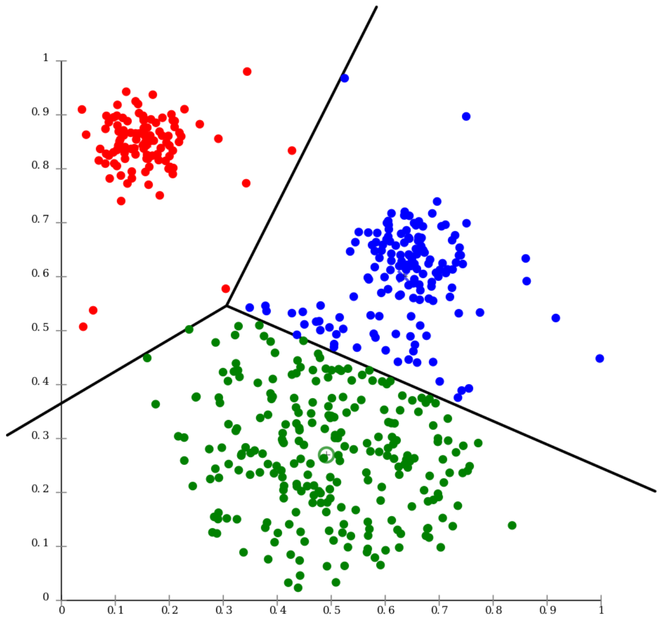
- Training on CPU or GPU
  - Ml.m5.2xlarge
  - Ml.p2.xlarge
- Inference
  - CPU for lower latency
  - GPU for higher throughput on large batches

DataCumulus Sundog Education

K-Means

K-Means: What's it for?

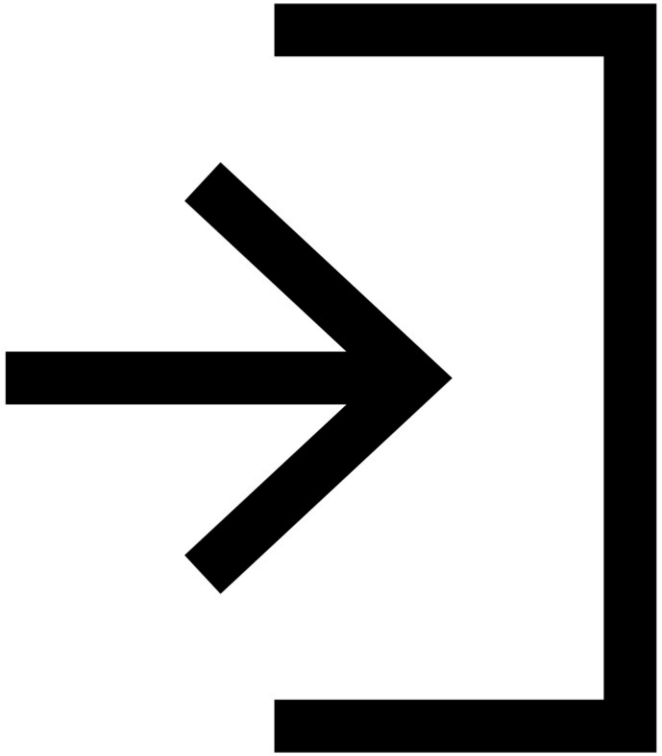
- Unsupervised clustering
- Divide data into K groups, where members of a group are as similar as possible to each other
  - You define what "similar" means
  - Measured by Euclidean distance
- Web-scale K-Means clustering



Chire [CC BY-SA 3.0 (https://creativecommons.org/licenses/by-sa/3.0)]

K-Means: What training input does it expect?

- Train channel, optional test
  - Train ShardedByS3Key, test FullyReplicate
- recordIO-protobuf or CSV
- File or Pipe on either



K-Means: How is it used?

- Every observation mapped to n-dimensional space (n = number of features)
- Works to optimize the center of K clusters
  - "extra cluster centers" may be specified to improve accuracy (which end up getting reduced to k)
  - K = k*x
- Algorithm:
  - Determine initial cluster centers
    - Random or k-means++ approach
    - K-means++ tries to make initial clusters far apart
  - Iterate over training data and calculate cluster centers
  - Reduce clusters from K to k
    - Using Lloyd's method with kmeans++

K-Means: Important Hyperparameters

- K!
  - Choosing K is tricky
  - Plot within-cluster sum of squares as function of K
  - Use “elbow method”
  - Basically optimize for tightness of clusters
- Mini_batch_size
- Extra_center_factor
- Init_method

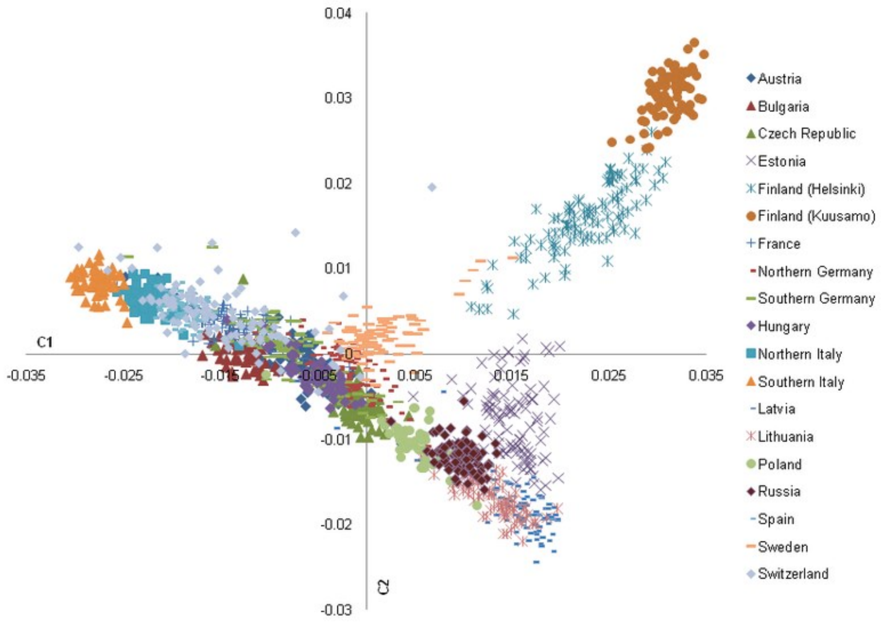
K-Means: Instance Types

- CPU or GPU, but CPU recommended
  - Only one GPU per instance used on GPU
  - So use ml.g4dn.xlarge if you're going to use GPU
  - p2, p3, g4dn, and g4 supported

PCA

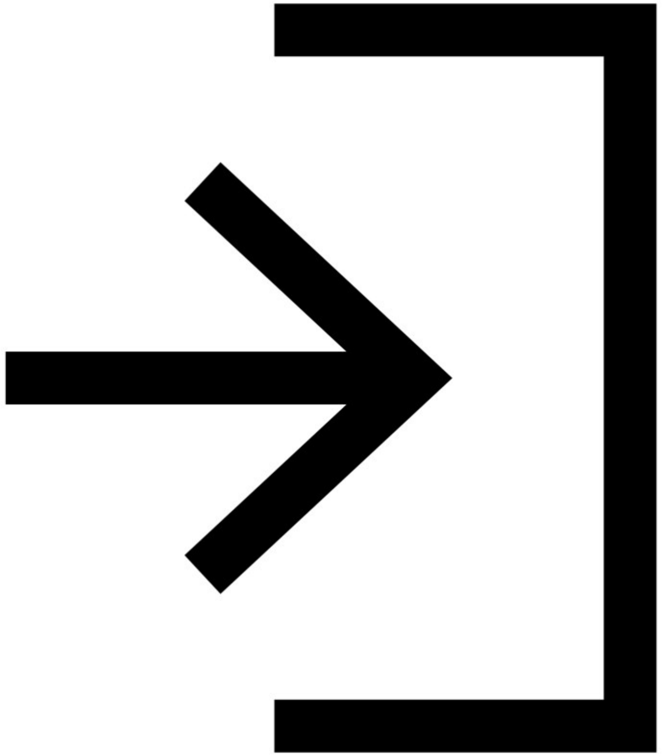
PCA: What's it for?

- Principal Component Analysis
- Dimensionality reduction
  - Project higher-dimensional data (lots of features) into lower-dimensional (like a 2D plot) while minimizing loss of information
  - The reduced dimensions are called components
    - First component has largest possible variability
    - Second component has the next largest…
- Unsupervised



PCA: What training input does it expect?

- recordIO-protobuf or CSV
- File or Pipe on either



PCA: How is it used?

- Covariance matrix is created, then singular value decomposition (SVD)
- Two modes
  - Regular
    - For sparse data and moderate number of observations and features
  - Randomized
    - For large number of observations and features
    - Uses approximation algorithm

PCA: Important Hyperparameters

- Algorithm_mode
- Subtract_mean
  - Unbias data

PCA: Instance Types

- GPU or CPU
  - It depends “on the specifics of the input data”

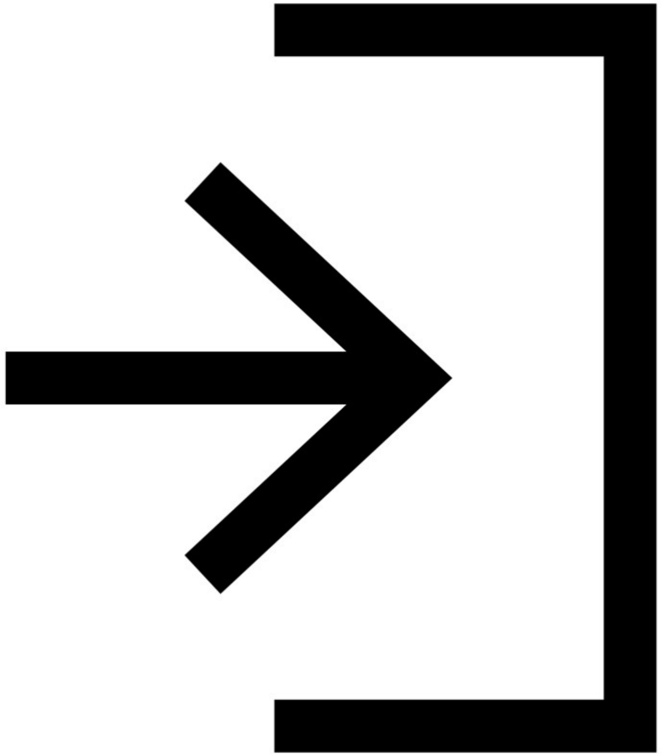
Factorization Machines

Factorization Machines: What's it for?

- Dealing with sparse data
  - Click prediction
  - Item recommendations
  - Since an individual user doesn't interact with most pages / products the data is sparse
- Supervised
  - Classification or regression
- Limited to pair-wise interactions
  - User -> item for example

Factorization Machines: What training input does it expect?

- recordIO-protobuf with Float32
  - Sparse data means CSV isn't practical



DataCumulus Sundog™ Education

Factorization Machines: How is it used?

- Finds factors we can use to predict a classification (click or not? Purchase or not?) or value (predicted rating?) given a matrix representing some pair of things (users & items?)
- Usually used in the context of recommender systems

Factorization Machines: Important Hyperparameters

- Initialization methods for bias, factors, and linear terms
  - Uniform, normal, or constant
  - Can tune properties of each method

DataCumulus Sundog Education

Factorization Machines: Instance Types

- CPU or GPU
  - CPU recommended
  - GPU only works with dense data

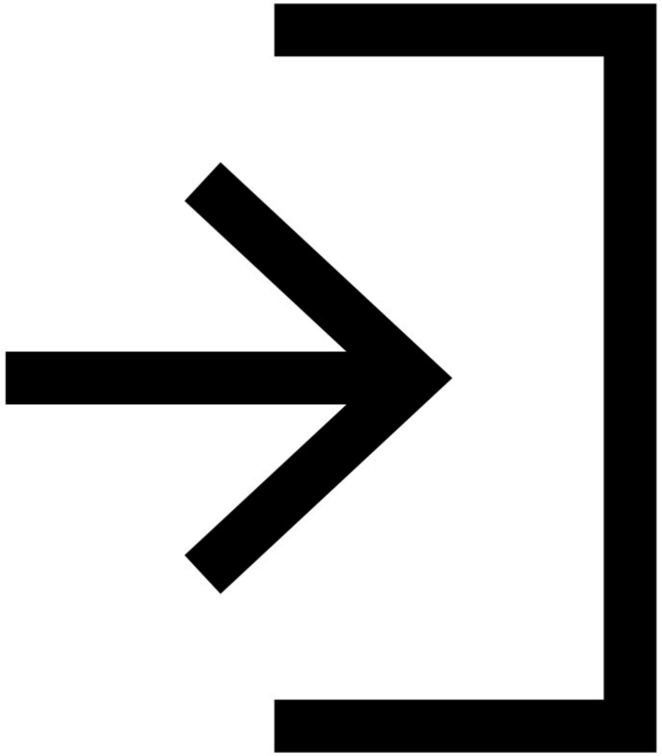
IP Insights

IP Insights: What's it for?

- Unsupervised learning of IP address usage patterns
- Identifies suspicious behavior from IP addresses
  - Identify logins from anomalous IP's
  - Identify accounts creating resources from anomalous IP's

IP Insights: What training input does it expect?

- User names, account ID's can be fed in directly; no need to pre-process
- Training channel, optional validation (computes AUC score)
- CSV only
  - Entity, IP



IP Insights: How is it used?

- Uses a neural network to learn latent vector representations of entities and IP addresses.
- Entities are hashed and embedded
  - Need sufficiently large hash size
- Automatically generates negative samples during training by randomly pairing entities and IP's

IP Insights: Important Hyperparameters

- Num_entity_vectors
  - Hash size
  - Set to twice the number of unique entity identifiers
- Vector_dim
  - Size of embedding vectors
  - Scales model size
  - Too large results in overfitting
- Epochs, learning rate, batch size, etc.

DataCumulus
Sundog Education

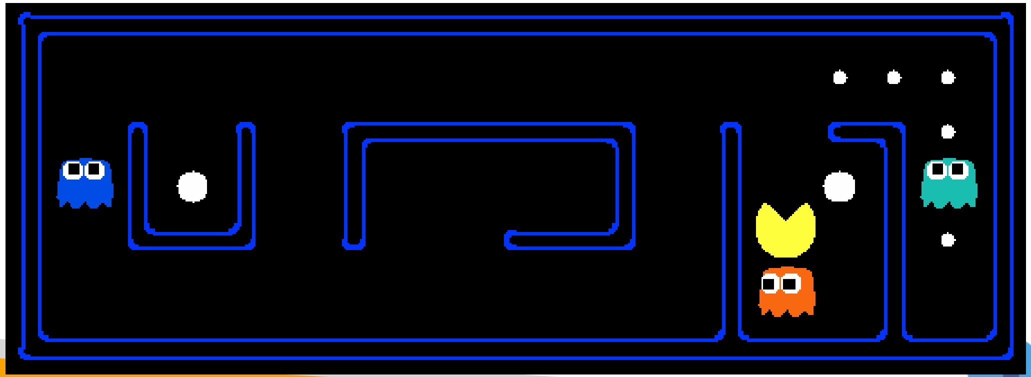
IP Insights: Instance Types

- CPU or GPU
  - GPU recommended
  - Ml.p3.2xlarge or higher
  - Can use multiple GPU's
  - Size of CPU instance depends on vector_dim and num_entity_vectors

Reinforcement Learning

Reinforcement Learning

- You have some sort of agent that "explores" some space
- As it goes, it learns the value of different state changes in different conditions
- Those values inform subsequent behavior of the agent
- Examples: Pac-Man, Cat & Mouse game (game AI)
  - Supply chain management
  - HVAC systems
  - Industrial robotics
  - Dialog systems
  - Autonomous vehicles
- Yields fast on-line performance once the space has been explored



DataCumulus Sundog™ Education

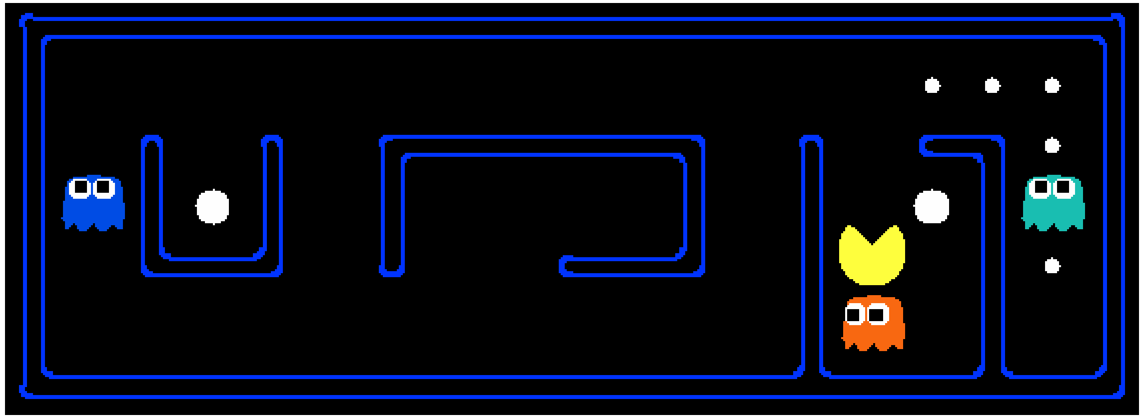
Q-Learning

- A specific implementation of reinforcement learning
- You have:
  - A set of environmental states *s*
  - A set of possible actions in those states *a*
  - A value of each state/action Q
- Start off with Q values of 0
- Explore the space
- As bad things happen after a given state/action, reduce its Q
- As rewards happen after a given state/action, increase its Q

Q(s, a) += alpha * (reward(s,a) + max(Q(s') - Q(s,a)) where s is the previous state, a is the previous action,s' is the current state, and alpha is the discount factor (set to .5 here).

Q-Learning

- What are some state/actions here?
  - Pac-man has a wall to the West
  - Pac-man dies if he moves one step South
  - Pac-man just continues to live if going North or East
- You can “look ahead” more than one step by using a discount factor when computing Q (here s is previous state, s’ is current state)
  - Q(s,a) += discount * (reward(s,a) + max(Q(s’)) – Q(s,a))



The exploration problem

- How do we efficiently explore all of the possible states?
  - Simple approach: always choose the action for a given state with the highest Q. If there's a tie, choose at random
    - But that's really inefficient, and you might miss a lot of paths that way
  - Better way: introduce an epsilon term
    - If a random number is less than epsilon, don't follow the highest Q, but choose at random
    - That way, exploration never totally stops
    - Choosing epsilon can be tricky

DataCumulus Sundog™ Education

Fancy Words

- Markov Decision Process
  - From Wikipedia: **Markov decision processes (MDPs)** provide a mathematical framework for modeling decision making in situations where outcomes are partly random and partly under the control of a decision maker.
  - Sound familiar? MDP's are just a way to describe what we just did using mathematical notation.
  - States are still described as s and s'
  - State transition functions are described as $P_a(s, s')$
  - Our "Q" values are described as a reward function $R_a(s, s')$
- Even fancier words! An MDP is a *discrete time stochastic control process.*

So to recap

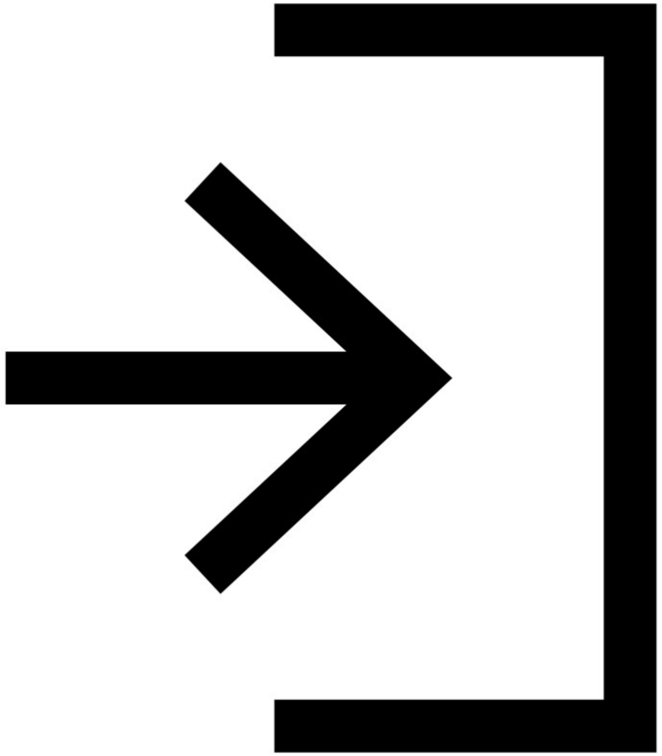
- You can make an intelligent Pac-Man in a few steps:
  - Have it semi-randomly explore different choices of movement (actions) given different conditions (states)
  - Keep track of the reward or penalty associated with each choice for a given state/action (Q)
  - Use those stored Q values to inform its future choices
- Pretty simple concept. But hey, now you can say you understand reinforcement learning, Q-learning, Markov Decision Processes, and Dynamic Programming!

Reinforcement Learning in SageMaker

- Uses a deep learning framework with Tensorflow and MXNet
- Supports Intel Coach and Ray Rllib toolkits.
- Custom, open-source, or commercial environments supported.
  - MATLAB, Simulink
  - EnergyPlus, RoboSchool, PyBullet
  - Amazon Sumerian, AWS RoboMaker

Distributed Training with SageMaker RL

- Can distribute training and/or environment rollout
- Multi-core and multi-instance



Reinforcement Learning: Key Terms

- Environment
  - The layout of the board / maze / etc
- State
  - Where the player / pieces are
- Action
  - Move in a given direction, etc
- Reward
  - Value associated with the action from that state
- Observation
  - i.e., surroundings in a maze, state of chess board

Reinforcement Learning: Hyperparameter Tuning

- Parameters of your choosing may be abstracted
- Hyperparameter tuning in SageMaker can then optimize them

Reinforcement Learning: Instance Types

- No specific guidance given in developer guide
- But, it's deep learning – so GPU's are helpful
- And we know it supports multiple instances and cores

Automatic Model Tuning

With SageMaker

DataCumulus Sundog™ Education

Hyperparameter tuning

- How do you know the best values of learning rate, batch size, depth, etc?
- Often you have to experiment
- Problem blows up quickly when you have many different hyperparameters; need to try every combination of every possible value somehow, train a model, and evaluate it every time

DataCumulus Sundog™ Education

Automatic Model Tuning

- Define the hyperparameters you care about and the ranges you want to try, and the metrics you are optimizing for
- SageMaker spins up a "HyperParameter Tuning Job" that trains as many combinations as you'll allow
  - Training instances are spun up as needed, potentially a lot of them
- The set of hyperparameters producing the best results can then be deployed as a model
- **It learns as it goes**, so it doesn't have to try every possible combination

Automatic Model Tuning: Best Practices

- Don’t optimize too many hyperparameters at once
- Limit your ranges to as small a range as possible
- Use logarithmic scales when appropriate
- Don’t run too many training jobs concurrently
  - This limits how well the process can learn as it goes
- Make sure training jobs running on multiple instances report the correct objective metric in the end

DataCumulus Sundog Education

SageMaker and Spark

Integrating SageMaker and Spark

- Pre-process data as normal with Spark
  - Generate DataFrames
- Use sagemaker-spark library
- SageMakerEstimator
  - KMeans, PCA, XGBoost
- SageMakerModel
- Notebooks can use the SparkMagic (PySpark) kernel

Create And Invoke Model

The IAM role specified in `iam_role` is passed to the containers SageMaker uses for model hosting allowing them to do things like publish CloudWatch metrics and download data from S3. If you are unsure of which policies to add to this role try adding the managed `AmazonSageMakerFullAccess` and scoping down permissions from there if needed.

Takes ~10-20 minutes

In [ ]:
```
from sagemaker_pyspark import IAMRole
from sagemaker_pyspark.algorithms import XGBoostSageMakerEstimator

iam_role = "name_of_your_iam_role"

xgboost_estimator = XGBoostSageMakerEstimator(
    trainingInstanceType="ml.m4.xlarge",
    trainingInstanceCount=1,
    endpointInstanceType="ml.m4.xlarge",
    endpointInitialInstanceCount=1,
    sagemakerRole=IAMRole(iam_role))

xgboost_estimator.setNumRound(25) # Set number of trees to use
xgboost_estimator.setNumClasses(10) # MNIST contains digits 0-9
xgboost_estimator.setObjective('multi:softmax') # Set XGBoost objective to multi-class classification w/ S
oftMax

xgboost_model = xgboost_estimator.fit(training_data)

transformed_data = xgboost_model.transform(test_data.limit(5)) # Score first 5 rows of test data
transformed_data.show()
```

Create And Invoke Model From An Existing Endpoint

In the last step we saw how you can create and train a model then invoke it from the model object. Here we create the model object from an existing SageMaker endpoint and use invoke it for scoring on the same test data.

In [8]:
```
from sagemaker_pyspark import SageMakerModel, EndpointCreationPolicy
from sagemaker_pyspark.transformation.serializers import LibSVMRequestRowSerializer
from sagemaker_pyspark.transformation.deserializers import XGBoostCSVRowDeserializer

my_endpoint = xgboost_model.endpointName # Get endpoint name of model created in previous step

xgboost_model = SageMakerModel(
    endpointInstanceType=None,
    endpointInitialInstanceCount=None,
    requestRowSerializer=LibSVMRequestRowSerializer(),
    responseRowDeserializer=XGBoostCSVRowDeserializer(),
    existingEndpointName=my_endpoint,
    endpointCreationPolicy=EndpointCreationPolicy.DO_NOT_CREATE
)
```


DataCumulus / Sundog Education

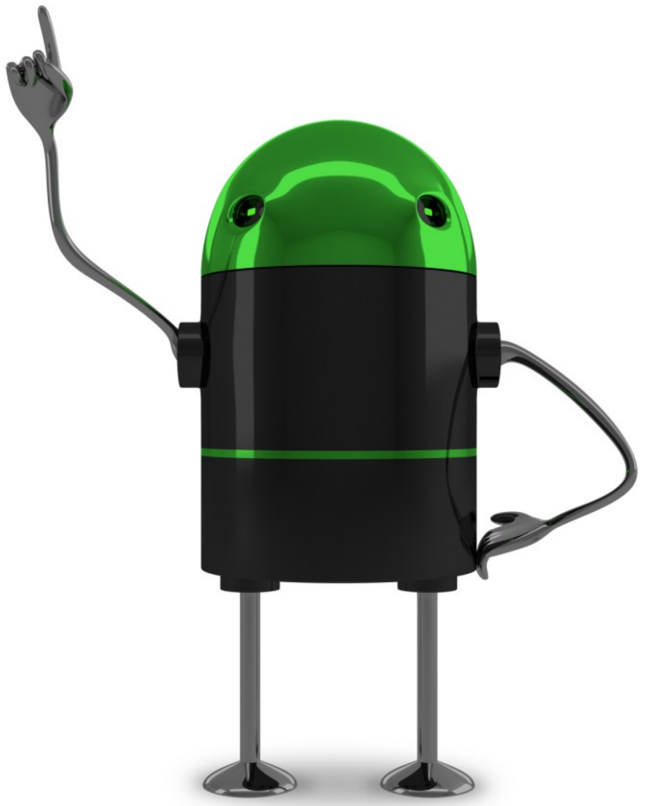
Integrating SageMaker and Spark

- Connect notebook to a remote EMR cluster running Spark (or use Zeppelin)
- Training dataframe should have:
  - A features column that is a vector of Doubles
  - An optional labels column of Doubles
- Call fit on your SageMakerEstimator to get a SageMakerModel
- Call transform on the SageMakerModel to make inferences
- Works with Spark Pipelines as well.

```
val estimator = new KMeansSageMakerEstimator(
 sagemakerRole = IAMRole(roleArn),
 trainingInstanceType = "ml.p2.xlarge",
 trainingInstanceCount = 1,
 endpointInstanceType = "ml.c4.xlarge",
 endpointInitialInstanceCount = 1)
 .setK(10).setFeatureDim(784)
 // train
 val model = estimator.fit(trainingData)
 val transformedData = model.transform(testData)
 transformedData.show
```

Why bother?

- Allows you to combine pre-processing big data in Spark with training and inference in SageMaker.
- EMR and SageMaker are now very tightly integrated.



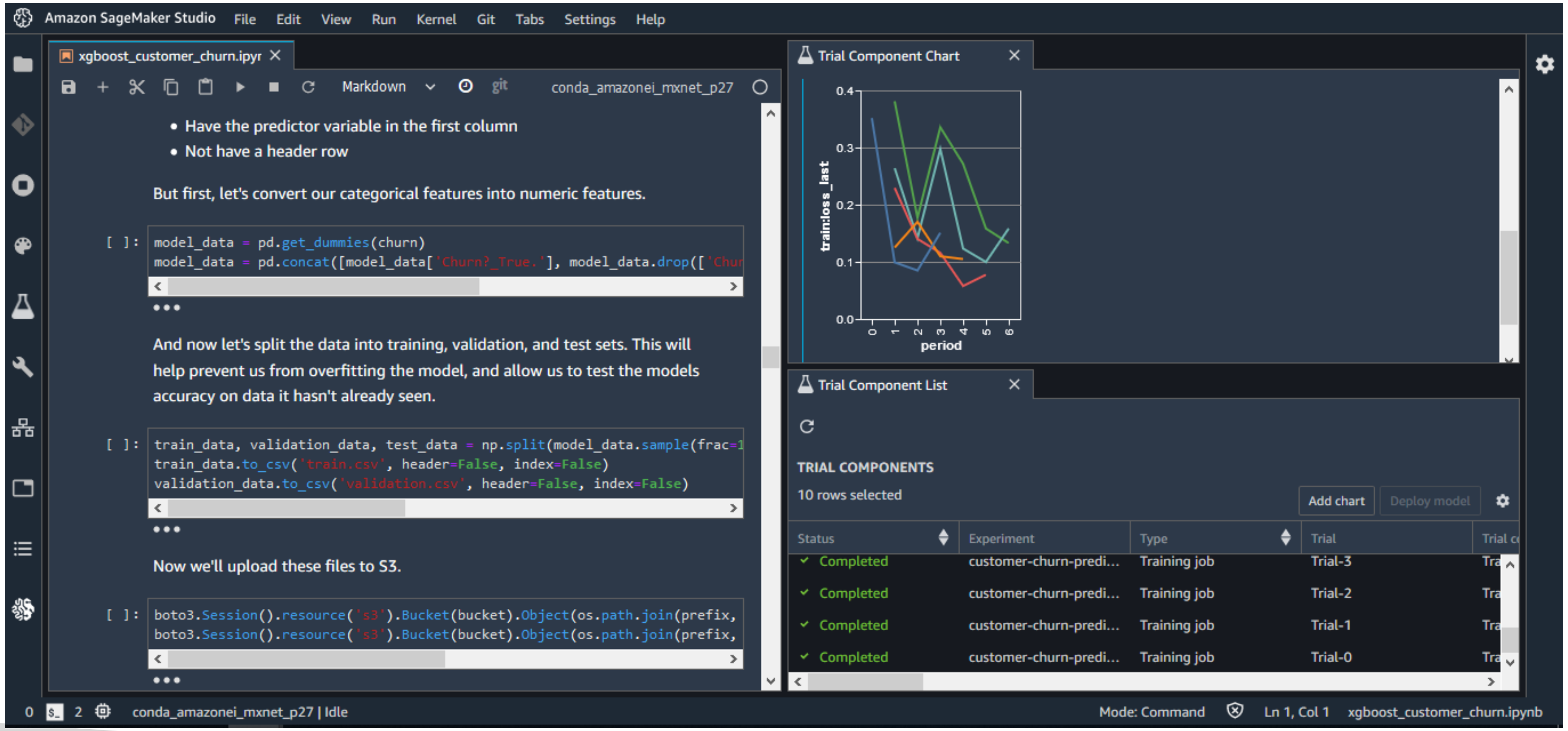
DataCumulus Sundog Education

Modern SageMaker

The SageMaker Studio era

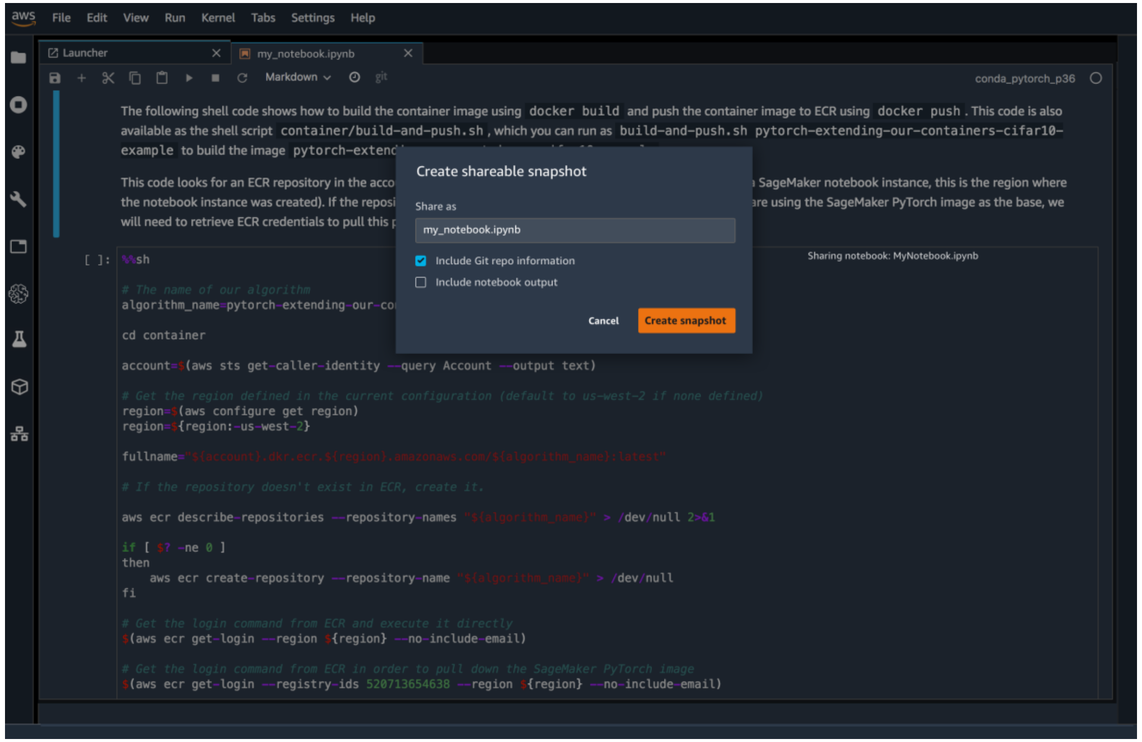
SageMaker Studio

- Visual IDE for machine learning!
- Integrates many of the features we're about to cover.



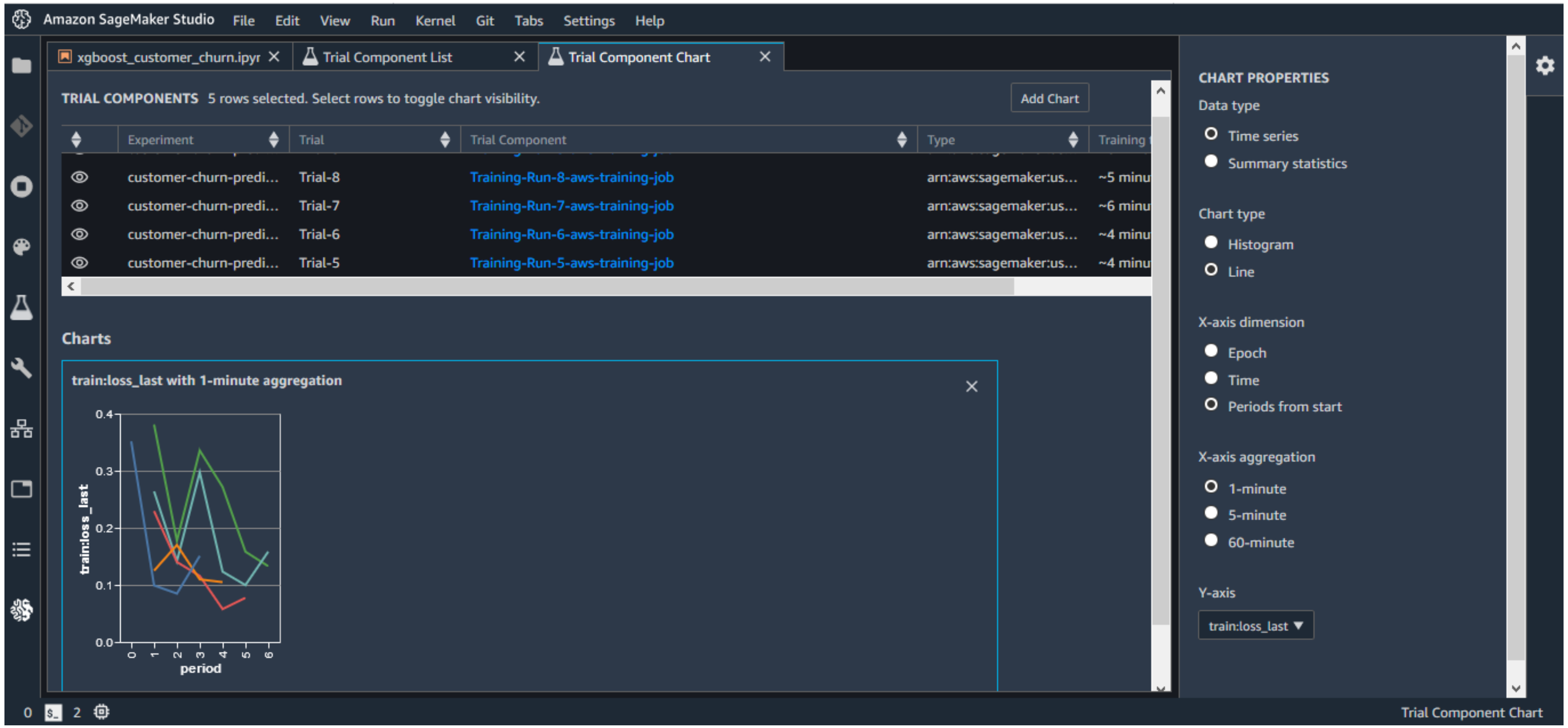
SageMaker Notebooks

- Create and share Jupyter notebooks with SageMaker Studio
- Switch between hardware configurations (no infrastructure to manage)



SageMaker Experiments

- Organize, capture, compare, and search your ML jobs



SageMaker Debugger

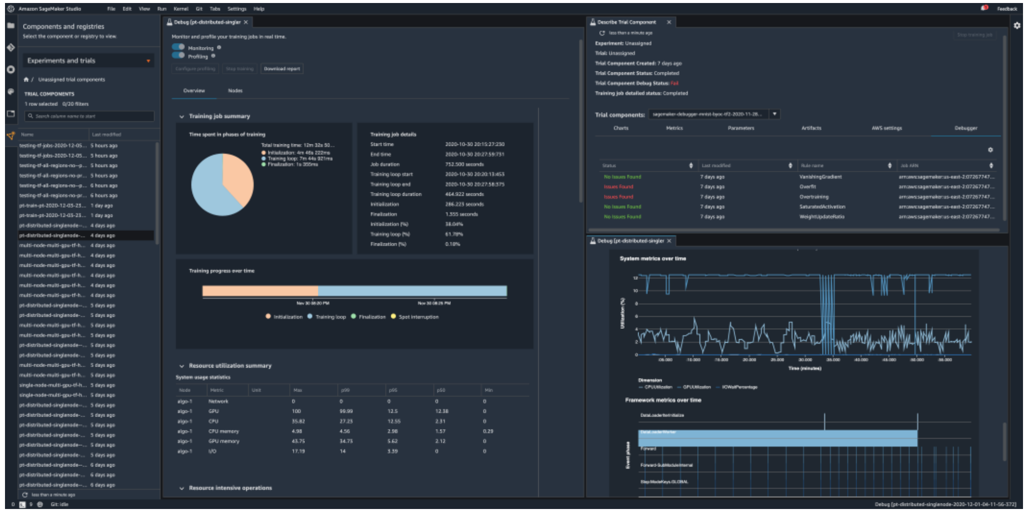
- Saves internal model state at periodical intervals
  - Gradients / tensors over time as a model is trained
  - Define rules for detecting unwanted conditions while training
  - A debug job is run for each rule you configure
  - Logs & fires a CloudWatch event when the rule is hit
- SageMaker Studio Debugger dashboards
- Auto-generated training reports
- Built-in rules:
  - Monitor system bottlenecks
  - Profile model framework operations
  - Debug model parameters

SageMaker Debugger

- Supported Frameworks & Algorithms:
  - Tensorflow
  - PyTorch
  - MXNet
  - XGBoost
  - SageMaker generic estimator (for use with custom training containers)
- Debugger API's available in GitHub
  - Construct hooks & rules for CreateTrainingJob and DescribeTrainingJob API's
  - SMDebug client library lets you register hooks for accessing training data

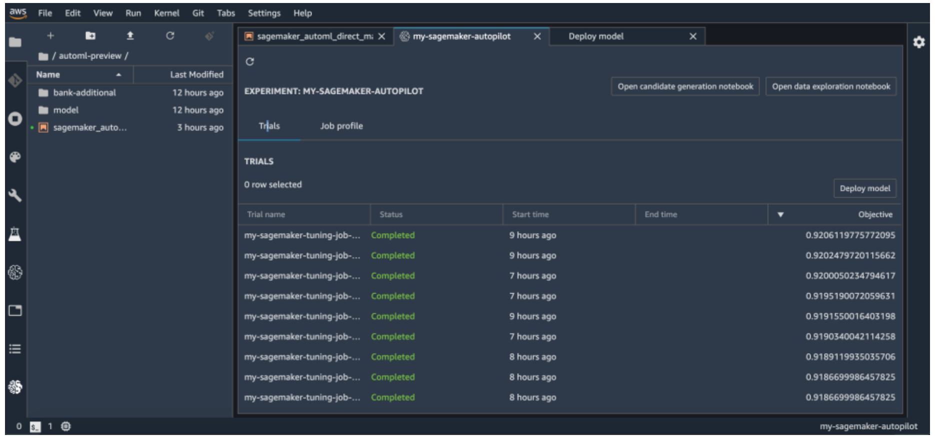
Even Newer SageMaker Debugger Features

- SageMaker Debugger Insights Dashboard
- Debugger ProfilerRule
  - ProfilerReport
  - Hardware system metrics (CPUBottlenck, GPUMemoryIncrease, etc)
  - Framework Metrics (MaxInitializationTime, OverallFrameworkMetrics, StepOutlier)
- Built-in actions to receive notifications or stop training
  - StopTraining(), Email(), or SMS()
  - In response to Debugger Rules
  - Sends notifications via SNS
- Profiling system resource usage and training



SageMaker Autopilot

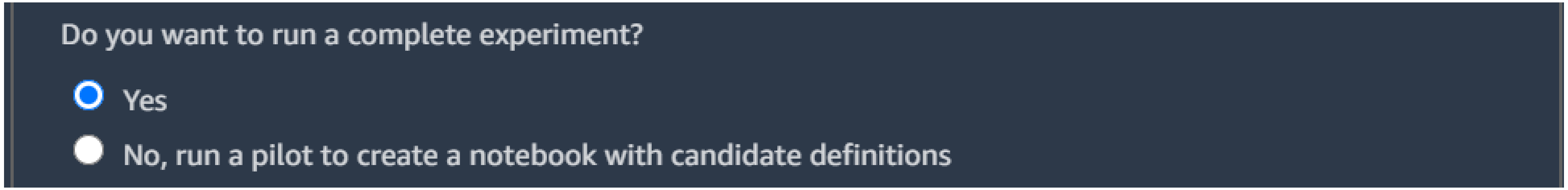
- Automates:
  - Algorithm selection
  - Data preprocessing
  - Model tuning
  - All infrastructure
- It does all the trial & error for you
- More broadly this is called AutoML



DataCumulus Sundog Education

SageMaker Autopilot workflow

- Load data from S3 for training
- Select your target column for prediction
- Automatic model creation
- Model notebook is available for visibility & control
- Model leaderboard
  - Ranked list of recommended models
  - You can pick one
- Deploy & monitor the model, refine via notebook if needed



SageMaker Autopilot

- Can add in human guidance
- With or without code in SageMaker Studio or AWS SDK's
- Problem types:
  - Binary classification
  - Multiclass classification
  - Regression
- Algorithm Types:
  - Linear Learner
  - XGBoost
  - Deep Learning (MLP's)
  - Ensemble mode
- Data must be tabular CSV or Parquet

Autopilot Training Modes

- HPO (Hyperparameter optimization)
  - Selects algorithms most relevant to your dataset
    - Linear Learner, XGBoost, or Deep Learning
  - Selects best range of hyperparameters to tune your models
    - Runs up to 100 trials to find optimal hyperparameters in the range
  - Bayesian optimization used if dataset < 100MB
  - Multi-fidelity optimization if > 100MB
    - Early stopping if a trial is performing poorly
- Ensembling
  - Trains several base models using AutoGluon library
    - Wider range of models, including more tree-based and neural network algorithms
  - Runs 10 trials with different model and parameter settings
  - Models are combined with a stacking ensemble method
- Auto
  - HPO if > 100MB
  - Ensembling if < 100MB
  - Autopilot needs to be able to read the size of your dataset, or will default to HPO
    - S3 bucket hidden inside a VPC
    - S3DataType is ManifestFile
    - S3Uri contains more than 1000 items

DataCumulus Sundog Education

Autopilot Explainability

- Integrates with SageMaker Clarify
- Transparency on how models arrive at predictions
- Feature attribution
  - Uses SHAP Baselines / Shapley Values
  - Research from cooperative game theory
  - Assigns each feature an importance value for a given prediction

A Unified Approach to Interpreting Model Predictions

Scott M. Lundberg
Paul G. Allen School of Computer Science
University of Washington
Seattle, WA 98105
slund1@cs.washington.edu

Su-In Lee
Paul G. Allen School of Computer Science
Department of Genome Sciences
University of Washington
Seattle, WA 98105
suinlee@cs.washington.edu

Abstract

Understanding why a model makes a certain prediction can be as crucial as the prediction's accuracy in many applications. However, the highest accuracy for large modern datasets is often achieved by complex models that even experts struggle to interpret, such as ensemble or deep learning models, creating a tension between *accuracy* and *interpretability*. In response, various methods have recently been proposed to help users interpret the predictions of complex models, but it is often unclear how these methods are related and when one method is preferable over another. To address this problem, we present a unified framework for interpreting predictions, SHAP (SHapley Additive exPlanations). SHAP assigns each feature an importance value for a particular prediction. Its novel components include: (1) the identification of a new class of additive feature importance measures, and (2) theoretical results showing there is a unique solution in this class with a set of desirable properties. The new class unifies six existing methods, notable because several recent methods in the class lack the proposed desirable properties. Based on insights from this unification, we present new methods that show improved computational performance and/or better consistency with human intuition than previous approaches.

1 Introduction

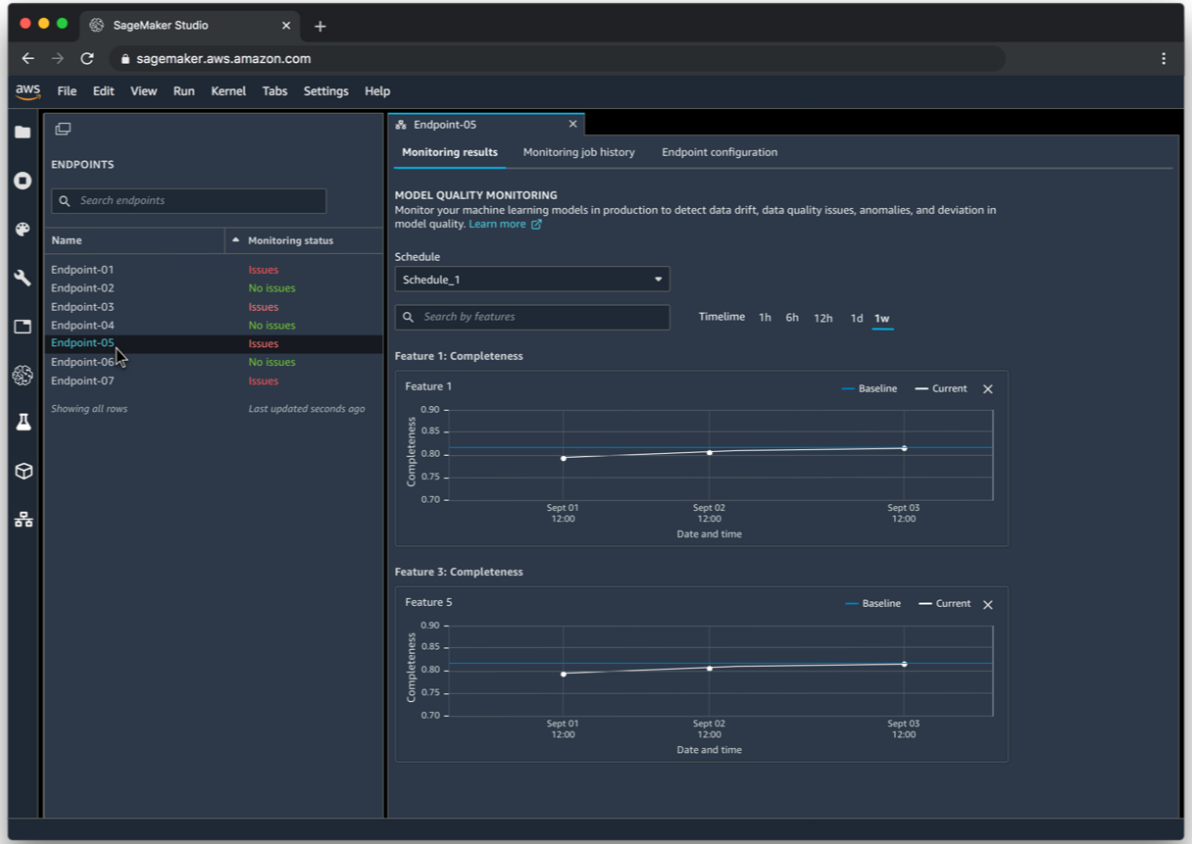
The ability to correctly interpret a prediction model's output is extremely important. It engenders appropriate user trust, provides insight into how a model may be improved, and supports understanding of the process being modeled. In some applications, simple models (e.g., linear models) are often preferred for their ease of interpretation, even if they may be less accurate than complex ones. However, the growing availability of big data has increased the benefits of using complex models, so bringing to the forefront the trade-off between accuracy and interpretability of a model's output. A wide variety of different methods have been recently proposed to address this issue [5, 8, 9, 3, 4, 1]. But an understanding of how these methods relate and when one method is preferable to another is still lacking.

Here, we present a novel unified approach to interpreting model predictions.[1] Our approach leads to three potentially surprising results that bring clarity to the growing space of methods:

DataCumulus Sundog Education

SageMaker Model Monitor

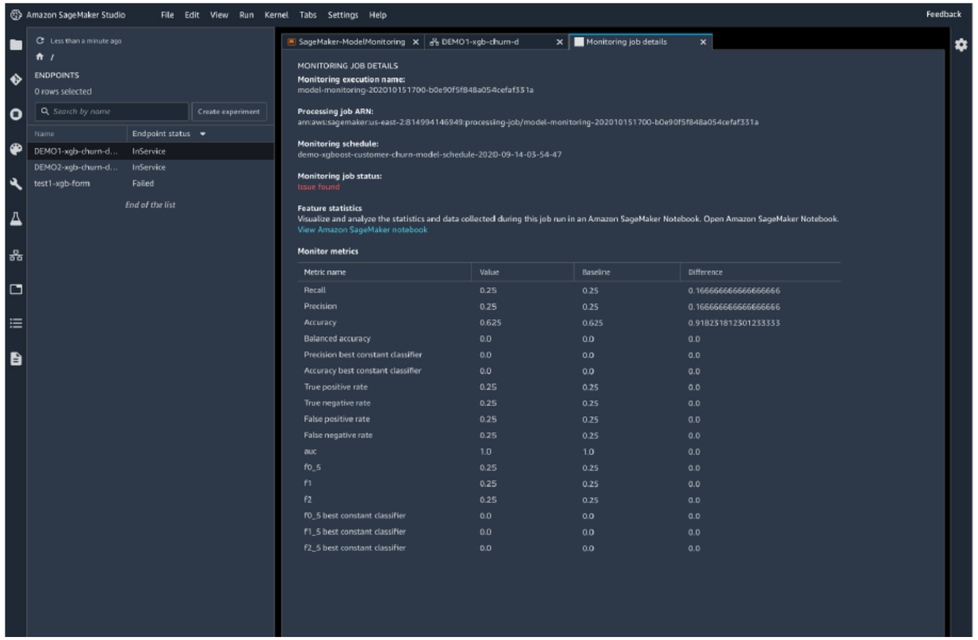
- Get alerts on quality deviations on your deployed models (via CloudWatch)
- Visualize data drift
  - Example: loan model starts giving people more credit due to drifting or missing input features
- Detect anomalies & outliers
- Detect new features
- No code needed



DataCumulus Sundog Education

SageMaker Model Monitor + Clarify

- Integrates with SageMaker Clarify
  - SageMaker Clarify detects potential bias
  - i.e., imbalances across different groups / ages / income brackets
  - With ModelMonitor, you can monitor for bias and be alerted to new potential bias via CloudWatch
  - SageMaker Clarify also helps explain model behavior
    - Understand which features contribute the most to your predictions

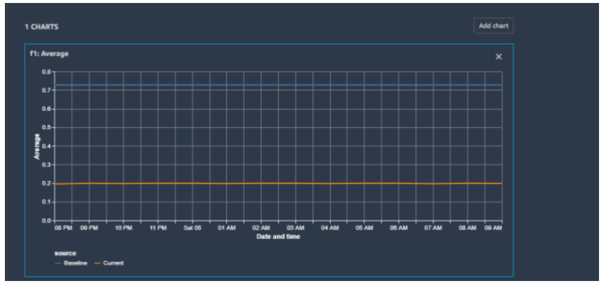


Pre-training Bias Metrics in Clarify

- Class Imbalance (CI)
  - One facet (demographic group) has fewer training values than another
- Difference in Proportions of Labels (DPL)
  - Imbalance of positive outcomes between facet values
- Kullback-Leibler Divergence (KL), Jensen-Shannon Divergence(JS)
  - How much outcome distributions of facets diverge
- Lp-norm (LP)
  - P-norm difference between distributions of outcomes from facets
- Total Variation Distance (TVD)
  - L1-norm difference between distributions of outcomes from facets
- Kolmogorov-Smirnov (KS)
  - Maximum divergence between outcomes in distributions from facets
- Conditional Demographic Disparity (CDD)
  - Disparity of outcomes between facets as a whole, and by subgroups

SageMaker Model Monitor

- Data is stored in S3 and secured
- Monitoring jobs are scheduled via a Monitoring Schedule
- Metrics are emitted to CloudWatch
  - CloudWatch notifications can be used to trigger alarms
  - You'd then take corrective action (retrain the model, audit the data)
- Integrates with Tensorboard, QuickSight, Tableau
  - Or just visualize within SageMaker Studio



DataCumulus Sundog Education

SageMaker Model Monitor

- Monitoring Types:
  - Drift in data quality
    - Relative to a baseline you create
    - “Quality” is just statistical properties of the features
  - Drift in model quality (accuracy, etc)
    - Works the same way with a model quality baseline
    - Can integrate with Ground Truth labels
  - Bias drift
  - Feature attribution drift
    - Based on Normalized Discounted Cumulative Gain (NDCG) score
    - This compares feature ranking of training vs. live data

Deployment Safeguards

- Deployment Guardrails
  - For asynchronous or real-time inference endpoints
  - Controls shifting traffic to new models
    - "Blue/Green Deployments"
      - All at once: shift everything, monitor, terminate blue fleet
      - Canary: shift a small portion of traffic and monitor
      - Linear: Shift traffic in linearly spaced steps
  - Auto-rollbacks
- Shadow Tests
  - Compare performance of shadow variant to production
  - You monitor in SageMaker console and decide when to promote it

DataCumulus Sundog Education

Putting them together

SageMaker Studio

SageMaker Autopilot

SageMaker Model Monitor

SageMaker Notebooks

SageMaker Experiments

SageMaker Debugger

Automatic Model Tuning

SageMaker: More Features

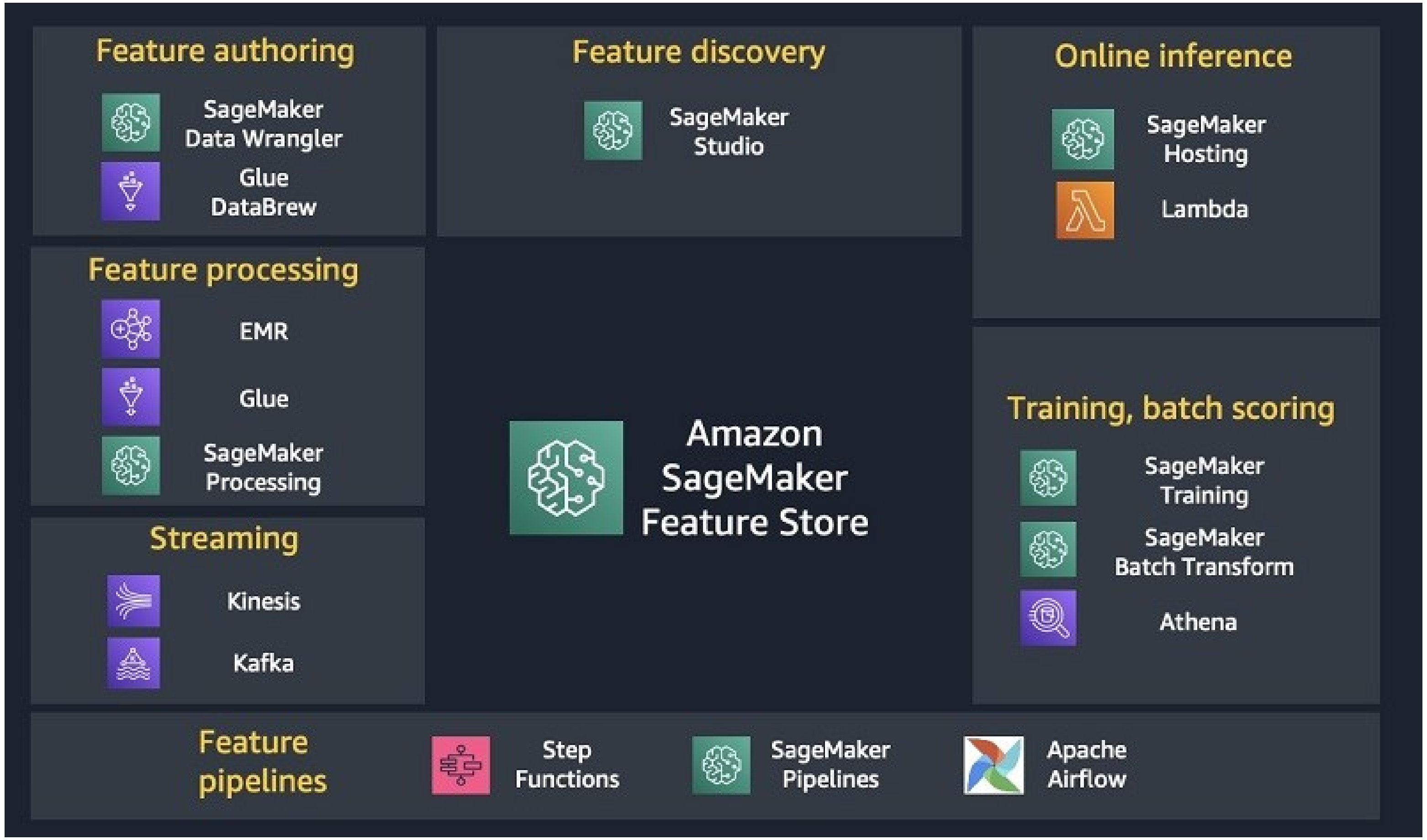
- SageMaker JumpStart
  - One-click models and algorithms from model zoos
  - Over 150 open source models in NLP, object detections, image classification, etc.
- SageMaker Data Wrangler
  - Import / transform / analyze / export data within SageMaker Studio
- SageMaker Feature Store
  - Find, discover, and share features in Studio
  - Online (low latency) or offline (for training or batch inference) modes
  - Features organized into Feature Groups
- SageMaker Edge Manager
  - Software agent for edge devices
  - Model optimized with SageMaker Neo
  - Collects and samples data for monitoring, labeling, retraining
- Asynchronous Inference endpoints

SageMaker Feature Store

- A "feature" is just a property used to train a machine learning model.
  - Like, you might predict someone's political party based on "features" such as their address, income, age, etc.
- Machine learning models require fast, secure access to feature data for training.
- It's also a challenge to keep it organized and share features across different models.

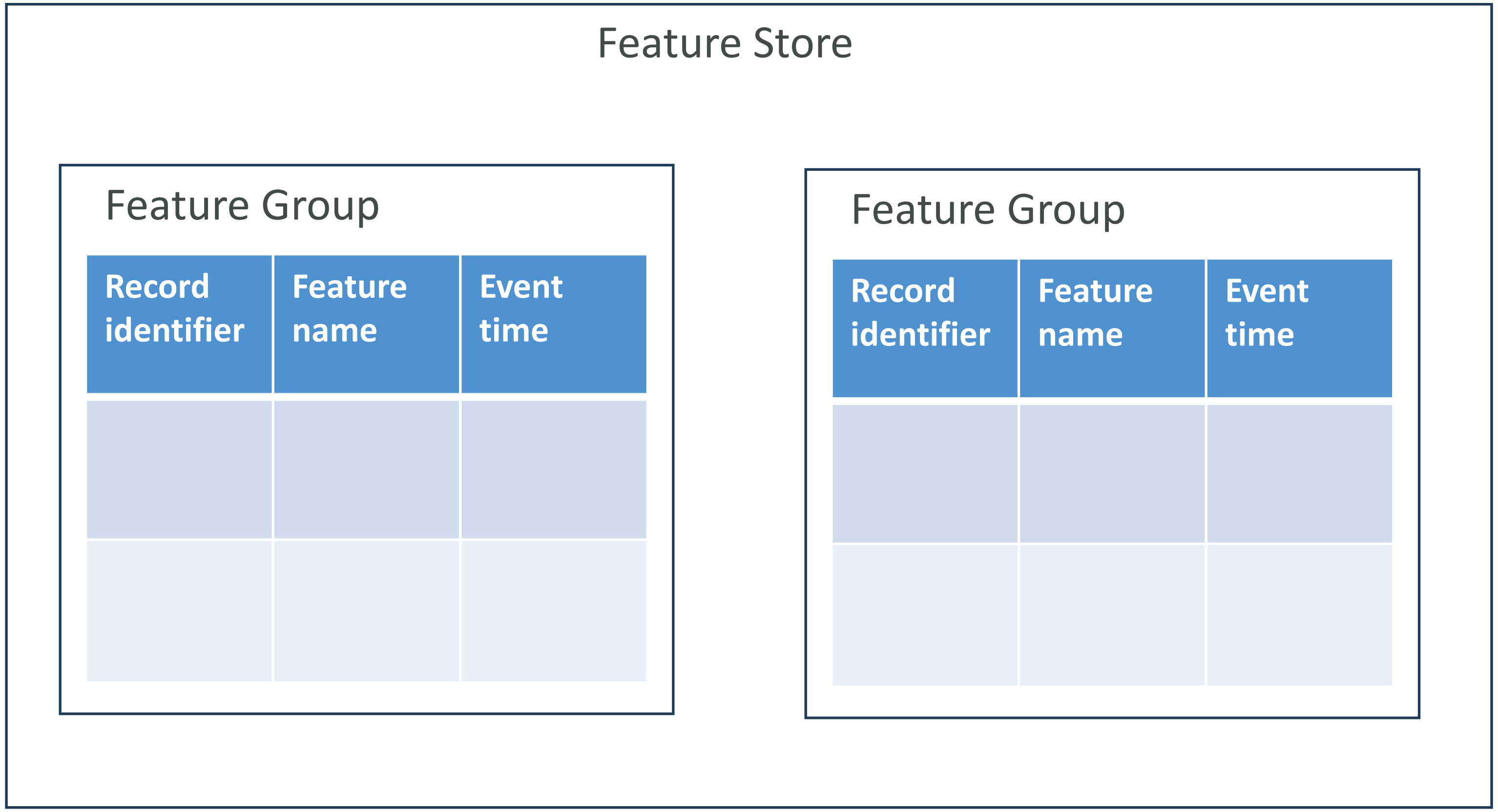
Amazon SageMaker Feature Store

Where the features come from is up to you.

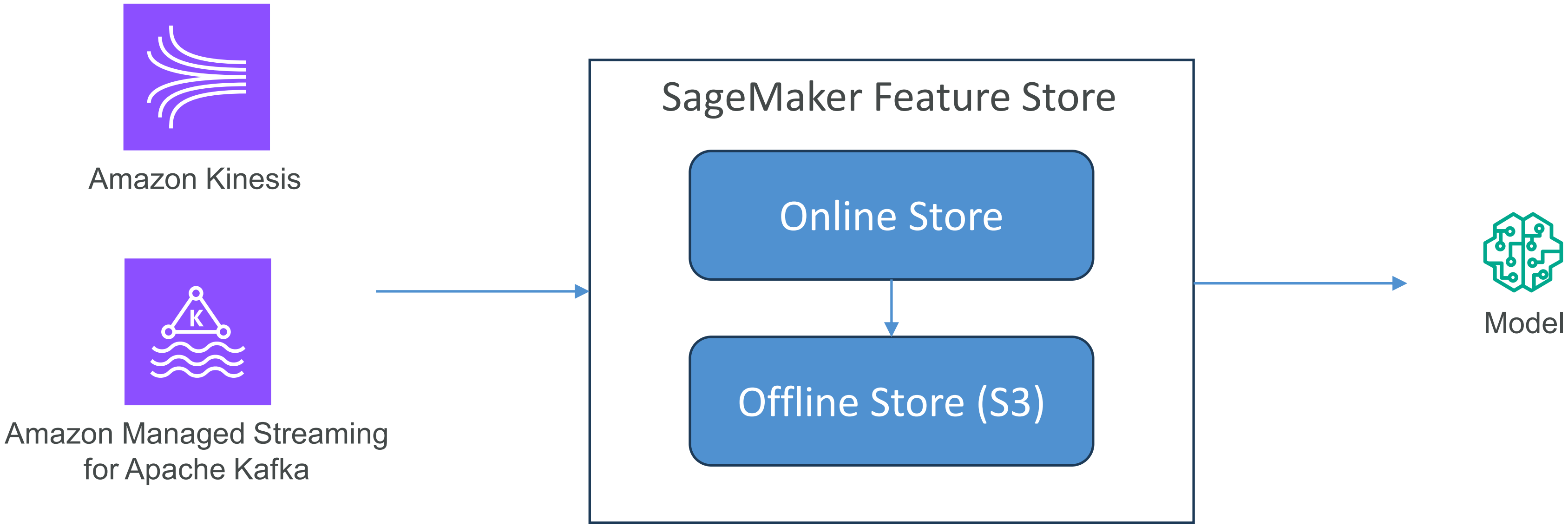


AWS

How SageMaker Feature Store Organizes Your Data



Data Ingestion (streaming or batch)



Spark, Data Wrangler, others

STREAMING access via PutRecord / GetRecord API's

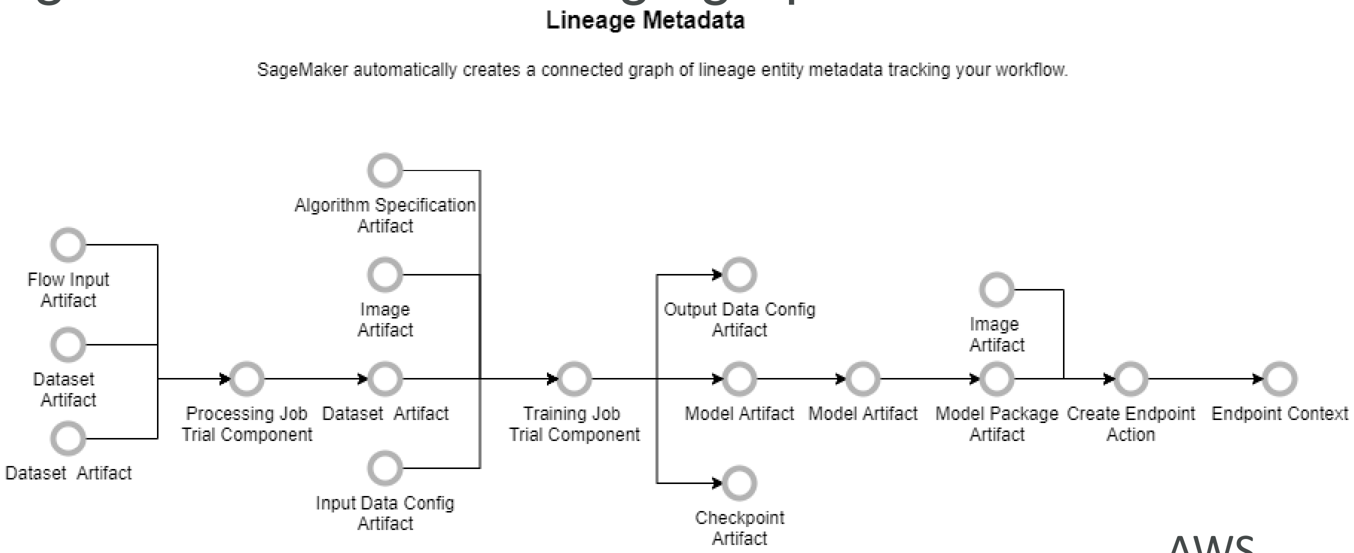
BATCH access via the offline S3 store (use with anything that hits S3, like Athena, Data Wrangler. Automatically creates a Glue Data Catalog for you.)

SageMaker Feature Store Security

- Encrypted at rest and in transit
- Works with KMS customer master keys
- Fine-grained access control with IAM
- May also be secured with AWS PrivateLink

SageMaker ML Lineage Tracking

- Creates & stores your ML workflow (MLOps)
- Keep a running history of your models
- Tracking for auditing and compliance
- Automatically or manually-created tracking entities
- Integrates with AWS Resource Access Manager for cross-account lineage
- Sample SageMaker-created lineage graph:



AWS

Lineage Tracking Entities

- Trial component (processing jobs, training jobs, transform jobs)
- Trial (a model composed of trial components)
- Experiment (a group of Trials for a given use case)
- Context (logical grouping of entities)
- Action (workflow step, model deployment
- Artifact (Object or data, such as an S3 bucket or an image in ECR)
- Association (connects entities together) – has optional AssociationType:
  - ContributedTo
  - AssociatedWith
  - DerivedFrom
  - Produced
  - SameAs

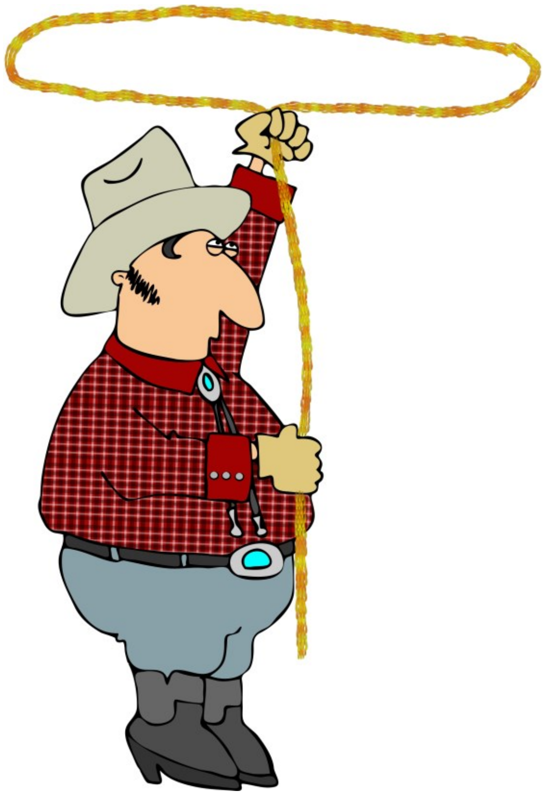
DataCumulus Sundog™ Education

Querying Lineage Entities

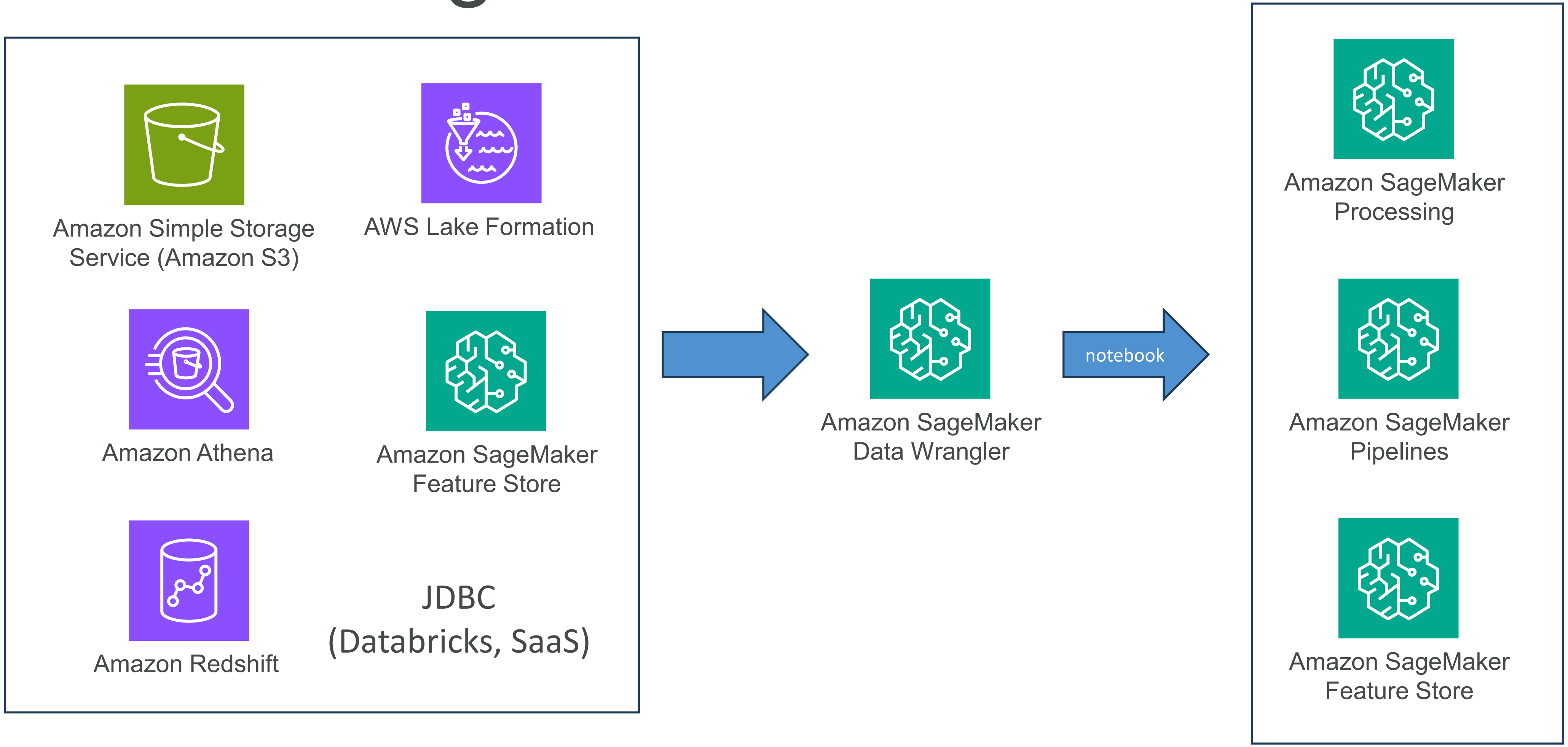
- Use the LineageQuery API from Python
  - Part of the Amazon SageMaker SDK for Python
- Do things like find all models / endpoints / etc. that use a given artifact
- Produce a visualization
  - Requires external Visualizer helper class

SageMaker Data Wrangler

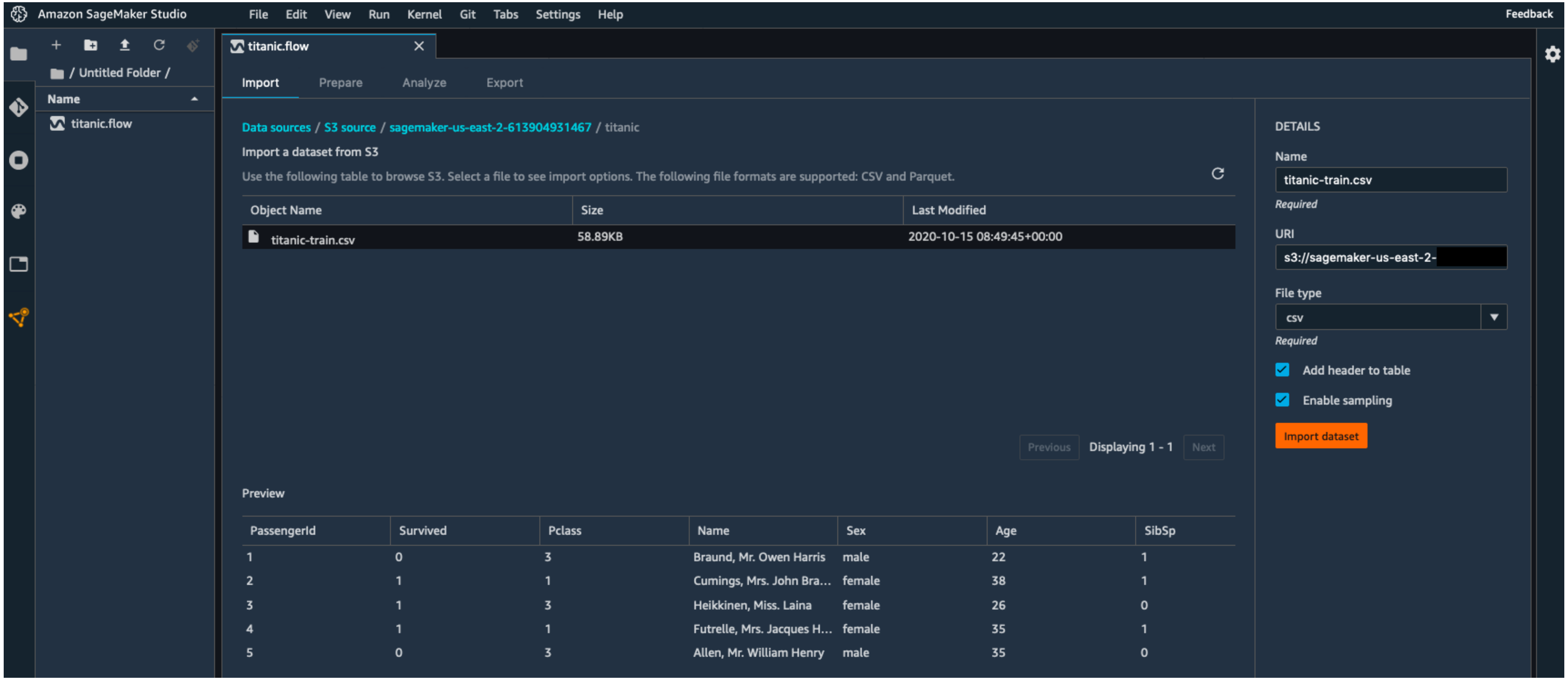
- Visual interface (in SageMaker Studio) to prepare data for machine learning
- Import data
- Visualize data
- Transform data (300+ transformations to choose from)
    - Or integrate your own custom xforms with pandas, PySpark, PySpark SQL
- "Quick Model" to train your model with your data and measure its results



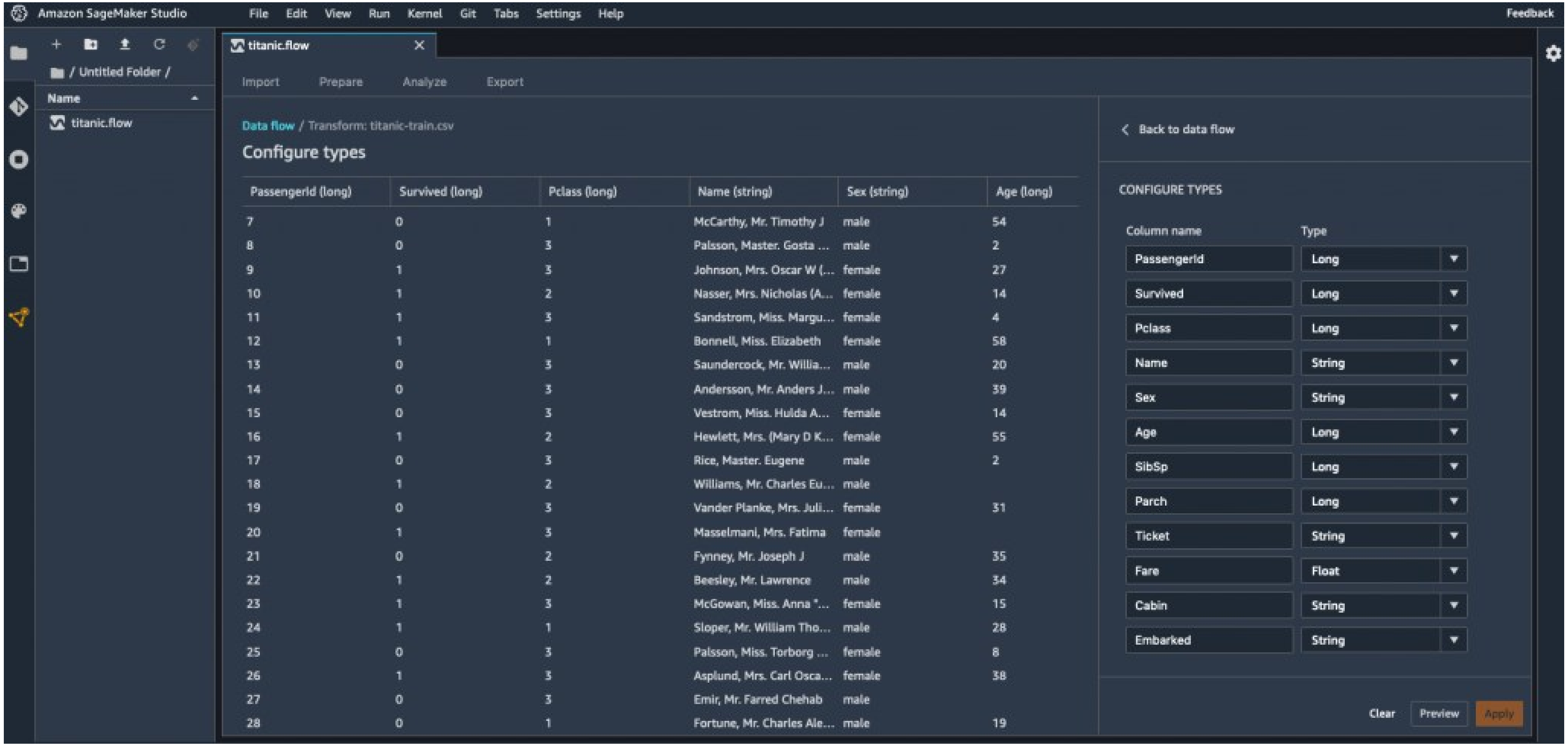
Data Wrangler sources



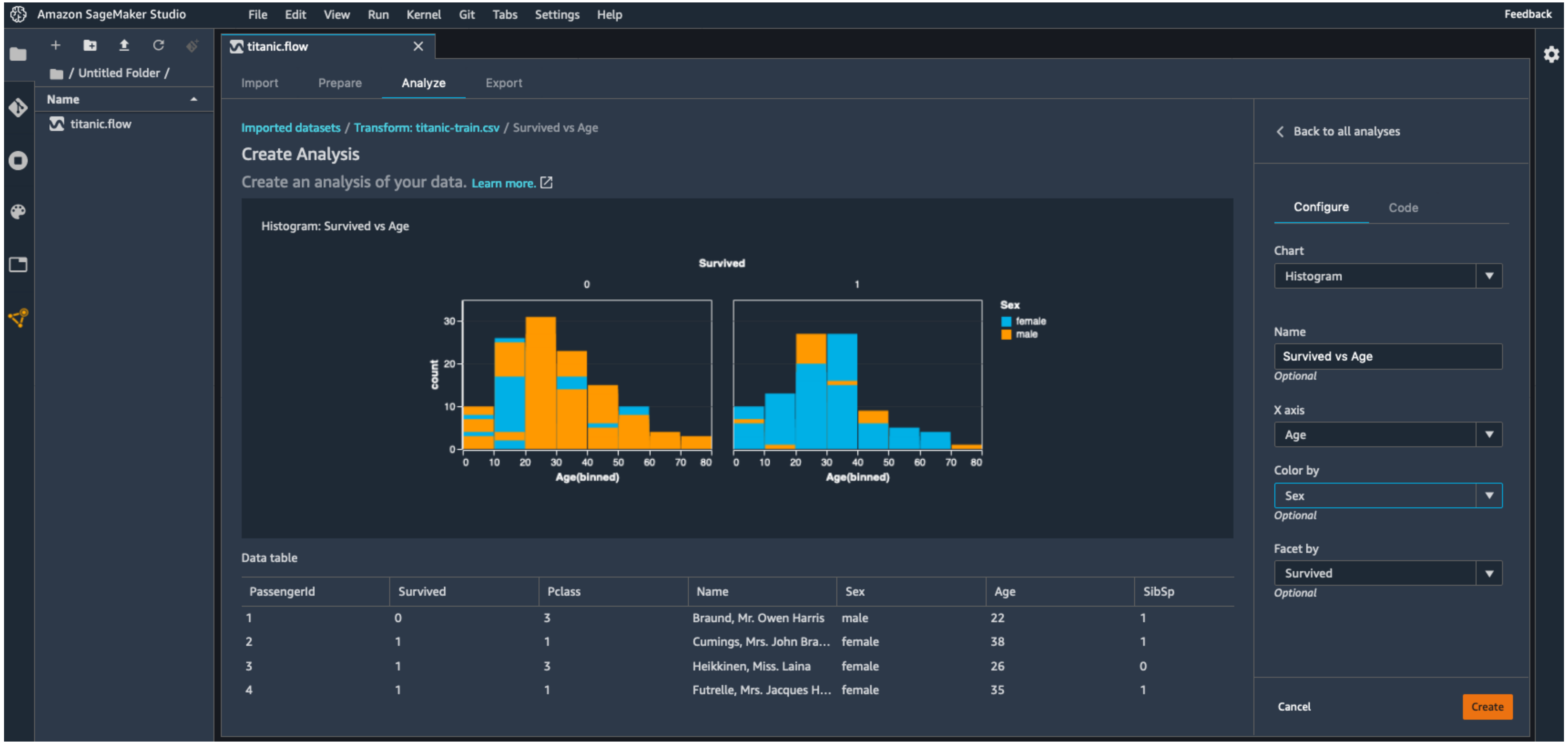
Data Wrangler: Import Data



Data Wrangler: Preview Data

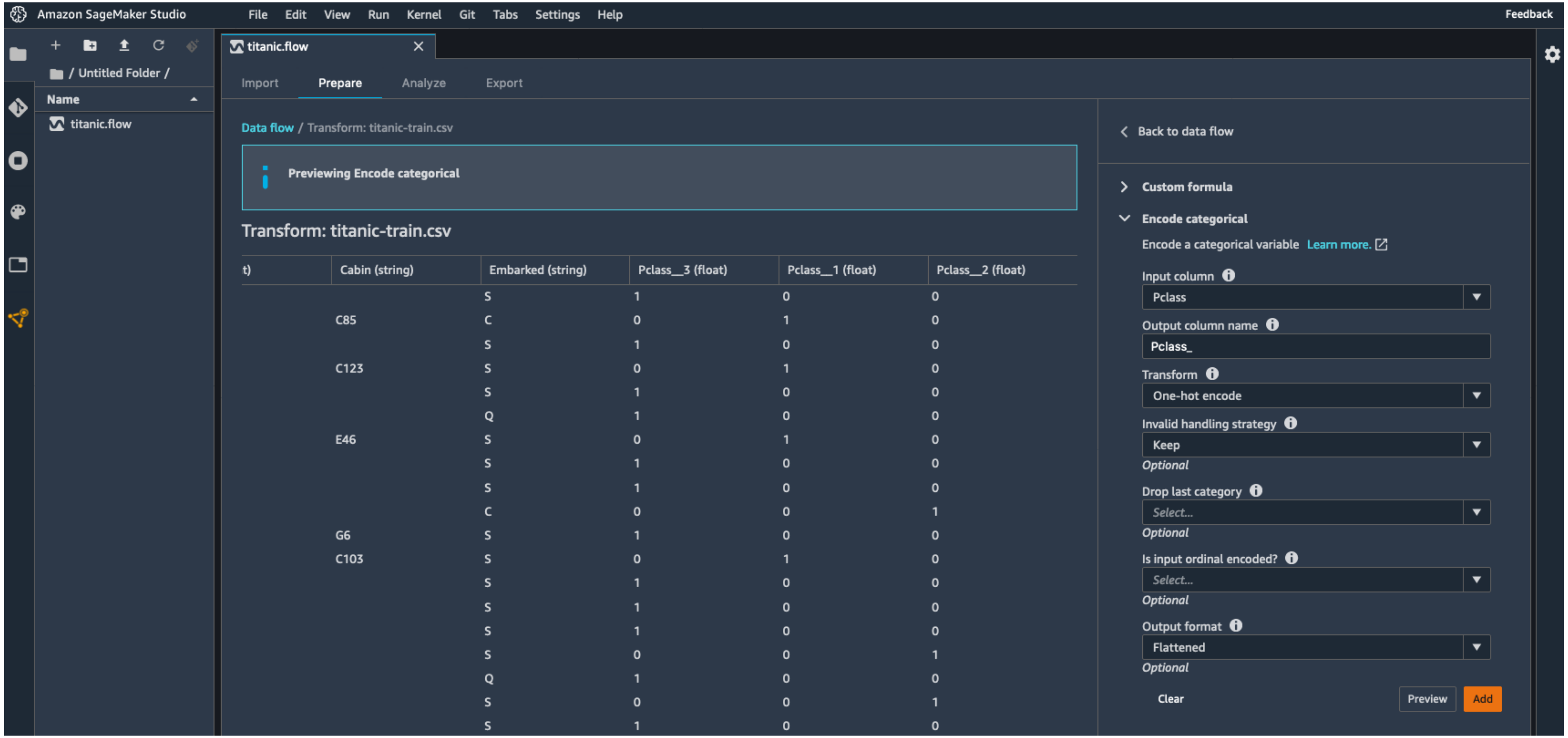


Data Wrangler: Visualize Data



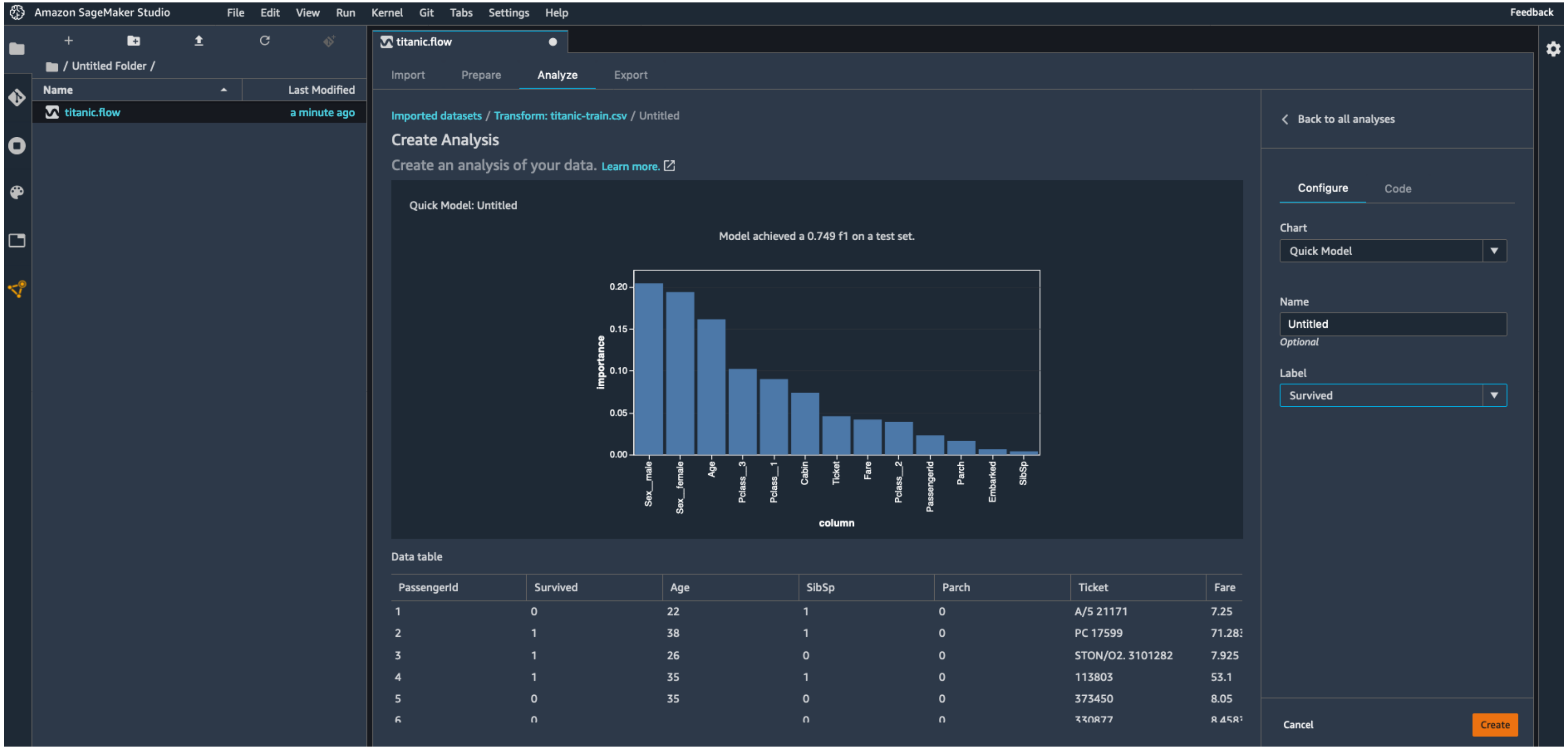
PassengerId	Survived	Pclass	Name	Sex	Age	SibSp
1	0	3	Braund, Mr. Owen Harris	male	22	1
2	1	1	Cumings, Mrs. John Bra...	female	38	1
3	1	3	Heikkinen, Miss. Laina	female	26	0
4	1	1	Futrelle, Mrs. Jacques H...	female	35	1

Data Wrangler: Transform Data



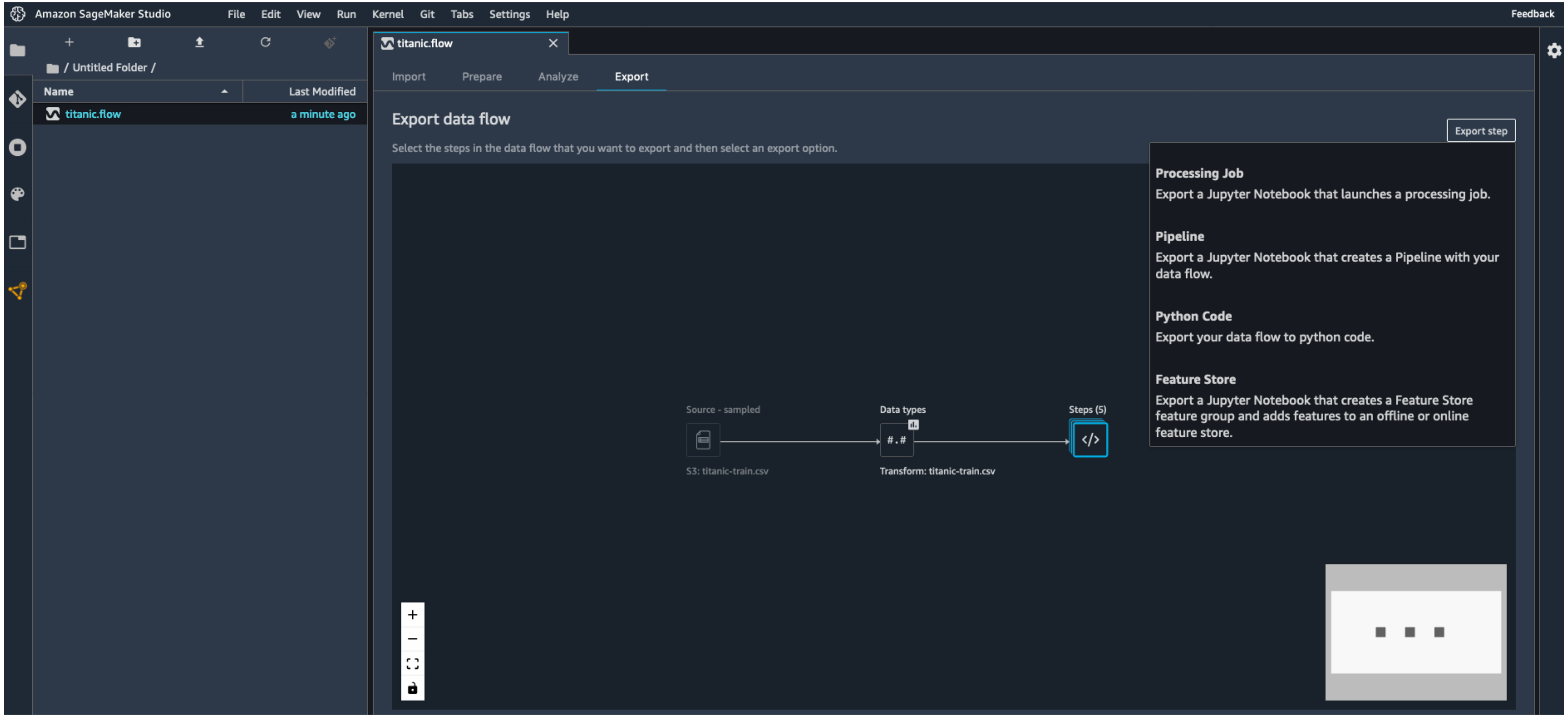
DataCumulus Sundog Education

Data Wrangler: Quick Model



DataCumulus
Sundog Education

Data Wrangler: Export Data Flow



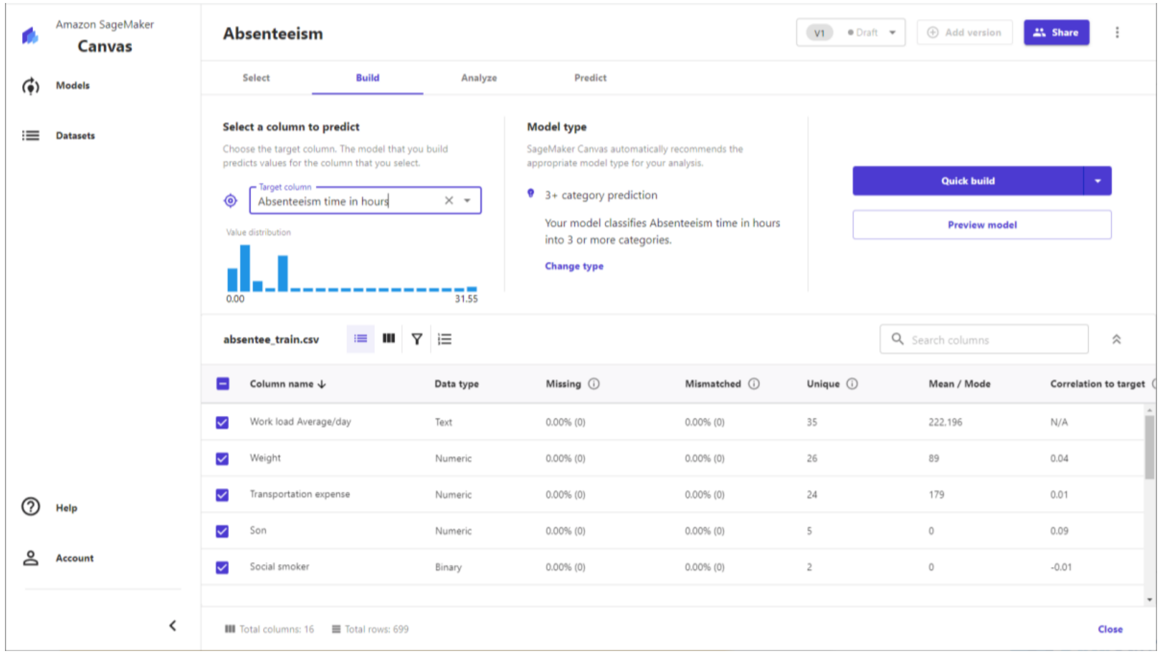
DataCumulus
Sundog Education

Data Wrangler Troubleshooting

- Make sure your Studio user has appropriate IAM roles
- Make sure permissions on your data sources allow Data Wrangler access
  - Add AmazonSageMakerFullAccess policy
- EC2 instance limit
  - If you get "The following instance type is not available…" errors
  - May need to request a quota increase
  - Service Quotas / Amazon SageMaker / Studio KernelGateway Apps running on ml.m5.4xlarge instance

SageMaker Canvas

- No-code machine learning for business analysts
- Upload csv data (csv only for now), select a column to predict, build it, and make predictions
- Can also join datasets
- Classification or regression
- Automatic data cleaning
  - Missing values
  - Outliers
  - Duplicates
- Share models & datasets with SageMaker Studio



DataCumulus Sundog Education

SageMaker Canvas Demo

SageMaker Canvas: The Finer Points

- Local file uploading must be configured "by your IT administrator."
  - Set up an S3 bucket with appropriate CORS permissions
- Can integrate with Okta SSO
- Canvas lives within a SageMaker Domain that must be manually updated
- Import from Redshift can be set up
- Time series forecasting must be enabled via IAM
- Can run within a VPC
- Pricing is $1.90/hr plus a charge based on number of training cells in a model

SageMaker Training Compiler

- Integrated into AWS Deep Learning Containers (DLCs)
  - Can't bring your own container
- Compile & optimize training jobs on GPU instances
- Can accelerate training up to 50%
- Converts models into hardware-optimized instructions
- Tested with Hugging Face transformers library, or bring your own model
- Incompatible with SageMaker distributed training libraries
- Best practices:
  - Ensure GPU instances are used (ml.p3, ml.p4)
  - PyTorch models must use PyTorch/XLA's model save function
  - Enable debug flag in compiler_config parameter to enable debugging

SageMaker Training Compiler

Automatically optimizes training job

DataCumulus Sundog Education

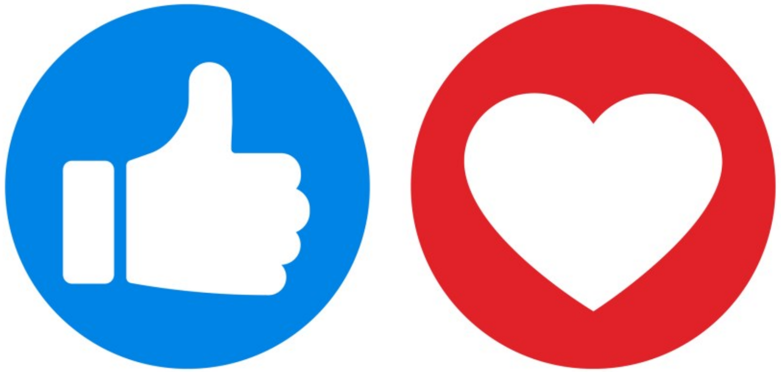
Modeling, Part 3: High-Level ML Services

Higher-Level AI/ML Services

Amazon Comprehend

Amazon Comprehend

- Natural Language Processing and Text Analytics
- Input social media, emails, web pages, documents, transcripts, medical records (Comprehend Medical)
- Extract key phrases, entities, sentiment, language, syntax, topics, and document classifications
- Events detection
- PII Identification & Redaction
- Targeted sentiment (for specific entities)
- Can train on your own data



Entities

Entities | Key phrases | Language | Sentiment | Syntax

Analyzed text

Amazon.com, Inc. is located in Seattle, WA and was founded July 5th, 1994 by Jeff Bezos, allowing customers to buy everything from books to blenders. Seattle is north of Portland and south of Vancouver, BC. Other notable Seattle - based companies are Starbucks and Boeing.

Results

Search

1

Entity	Category	Confidence
Amazon.com, Inc	Organization	0.90
Seattle, WA	Location	0.89
July 5th, 1994	Date	0.99+
Jeff Bezos	Person	0.99+
Seattle	Location	0.97

Key Phrases

Entities | **Key phrases** | Language | Sentiment | Syntax

Analyzed text

Amazon.com, Inc. is located in Seattle, WA and was founded July 5th, 1994 by Jeff Bezos, allowing customers to buy everything from books to blenders. Seattle is north of Portland and south of Vancouver, BC. Other notable Seattle - based companies are Starbucks and Boeing.

Results

Search 1 2

Key phrases	Confidence
Amazon.com	0.86
Seattle, WA	0.95
July 5th, 1994	0.91
Jeff Bezos	0.99+
customers	0.99+
books	0.99+
blenders	0.98

DataCumulus Sundog™ Education

Language

Insights Info

Entities | Key phrases | **Language** | Sentiment | Syntax

Analyzed text

Amazon.com, Inc. is located in Seattle, WA and was founded July 5th, 1994 by Jeff Bezos, allowing customers to buy everything from books to blenders. Seattle is north of Portland and south of Vancouver, BC. Other notable Seattle - based companies are Starbucks and Boeing.

▼ Results

Language

English, en
0.99 confidence

▶ Application integration

DataCumulus Sundog™ Education

Sentiment

Entities | Key phrases | Language | **Sentiment** | Syntax

Analyzed text

Amazon.com, Inc. is located in Seattle, WA and was founded July 5th, 1994 by Jeff Bezos, allowing customers to buy everything from books to blenders. Seattle is north of Portland and south of Vancouver, BC. Other notable Seattle - based companies are Starbucks and Boeing.

▼ **Results**

Sentiment

Neutral	Positive	Negative	Mixed
0.99 confidence	0.00 confidence	0.00 confidence	0.00 confidence

▶ **Application integration**

DataCumulus Sundog™ Education

Syntax

Amazon.com, Inc. is located in Seattle, WA and was founded July 5th, 1994 by Jeff Bezos, allowing customers to buy everything from books to blenders. Seattle is north of Portland and south of Vancouver, BC. Other notable Seattle - based companies are Starbucks and Boeing.

Results

Search

1 2 3 4 5 6

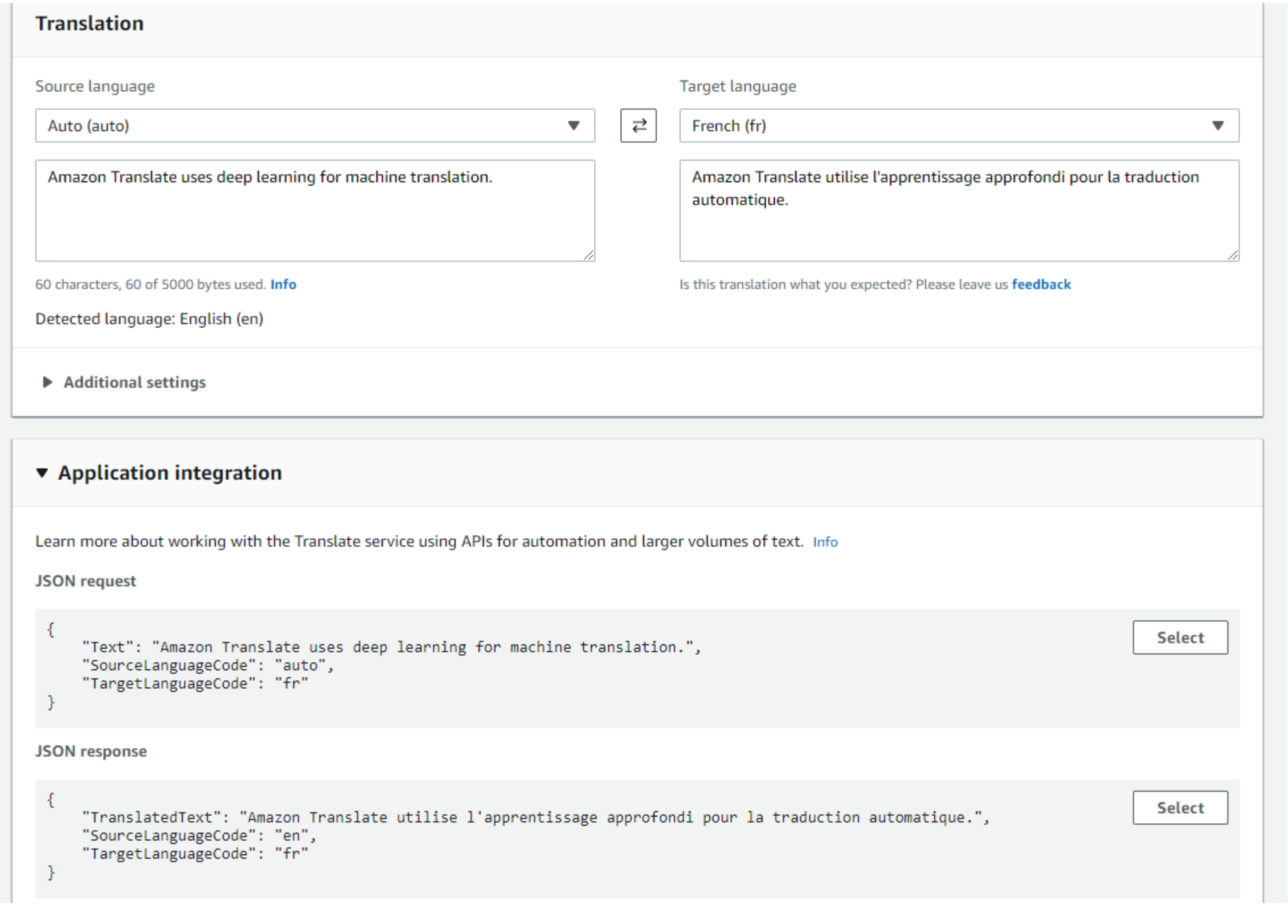
Word	Part of speech	Confidence
Amazon.com	Proper noun	0.94
,	Punctuation	0.99+
Inc.	Proper noun	0.99+
is	Auxiliary verb	0.98
located	Verb	0.99+
in	Adposition	0.99+
Seattle	Proper noun	0.99+
,	Punctuation	0.99+

Amazon Translate

Amazon Translate

- Uses deep learning for translation
- Supports custom terminology
  - In CSV or TMX format
  - Appropriate for proper names, brand names, etc.

Amazon Translate



Amazon Transcribe

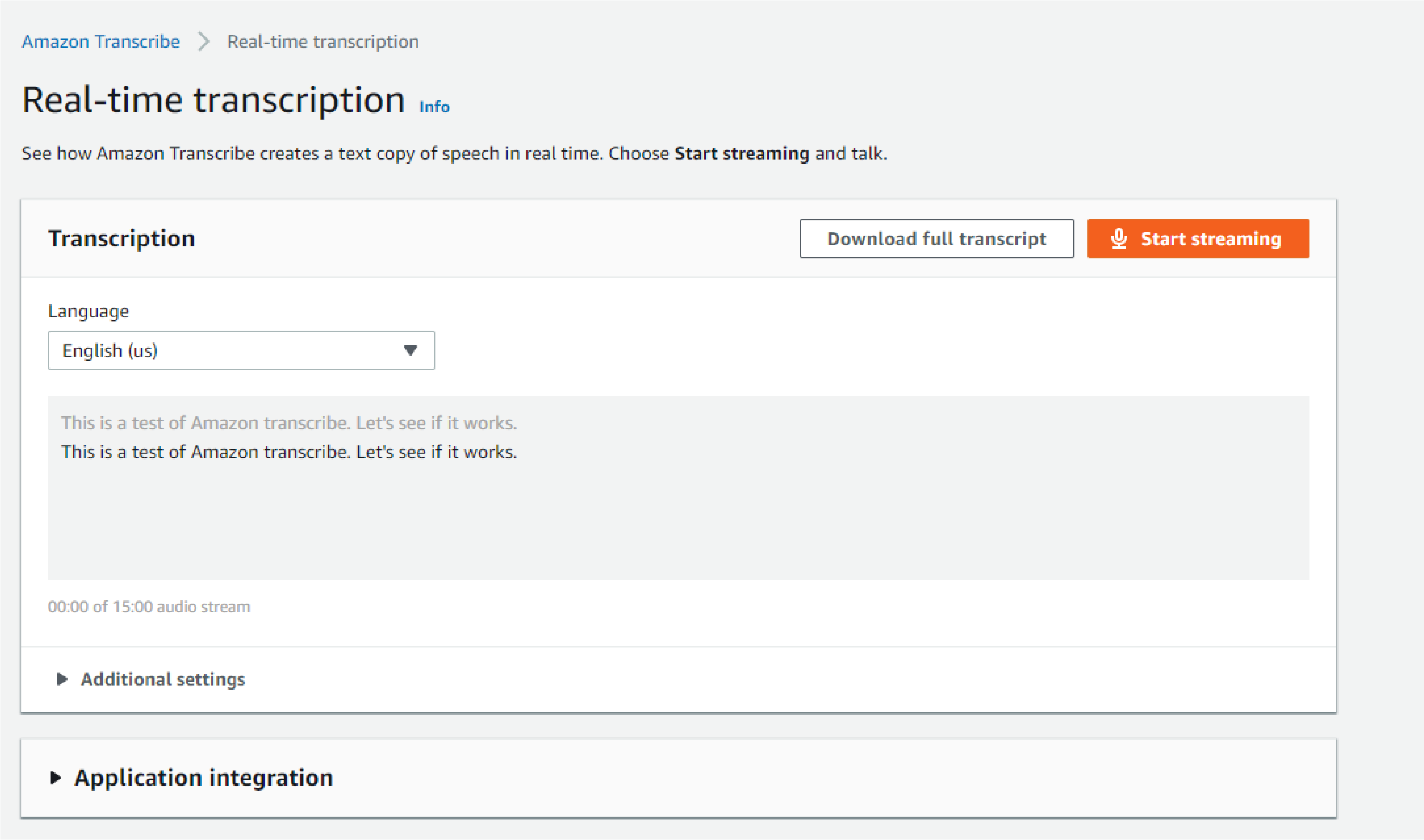
Amazon Transcribe

- Speech to text
  - Input in FLAC, MP3, MP4, or WAV, in a specified language
  - Streaming audio supported (HTTP/2 or WebSocket)
    - French, English, Spanish only
- Speaker Identificiation
  - Specify number of speakers
- Channel Identification
  - i.e., two callers could be transcribed separately
  - Merging based on timing of "utterances"
- Automatic Language Identification
  - You don't have to specify a language; it can detect the dominant one spoken.
- Custom Vocabularies
  - Vocabulary Lists (just a list of special words – names, acronyms)
  - Vocabulary Tables (can include "SoundsLike", "IPA", and "DisplayAs")

Amazon Transcribe Use Cases

- Call Analytics
  - Trained specifically for customer service & sales calls
  - Real-time transcriptions & insights
  - Sentiment, talk speed, interruptions, look for specific phrases
    - Like "cancel my subscription"
- Medical
  - Trained on medical terminology
  - HIPAA-eligible
- Subtitling
  - Live subtitle output

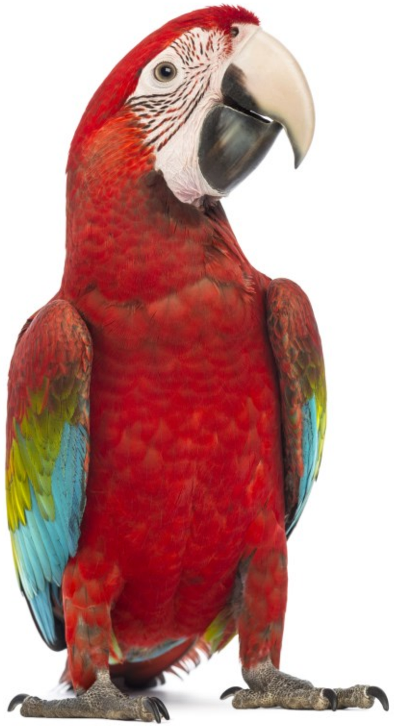
Amazon Transcribe



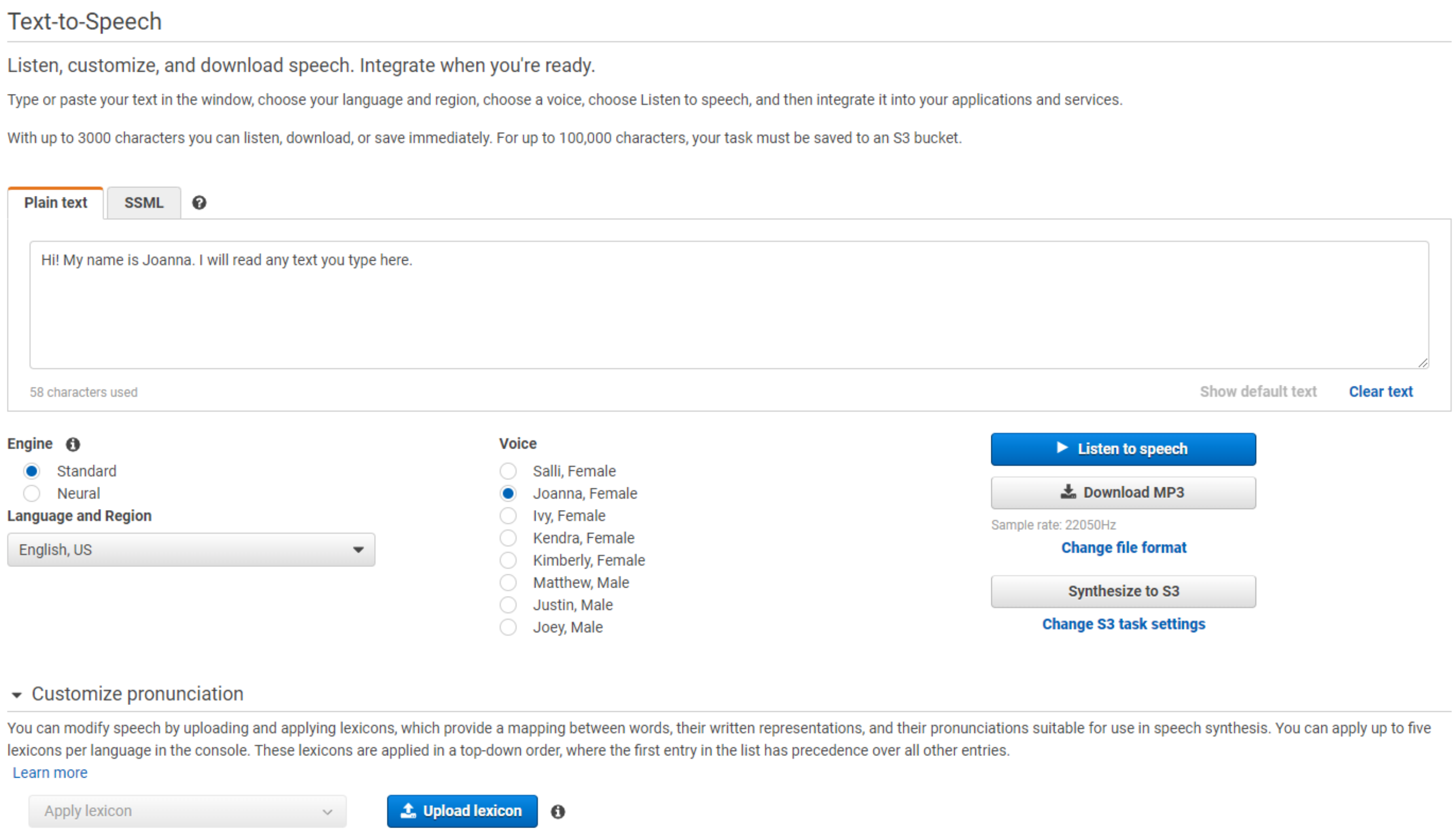
Amazon Polly

Amazon Polly

- Neural Text-To-Speech, many voices & languages
- Lexicons
  - Customize pronunciation of specific words & phrases
  - Example: “World Wide Web Consortium” instead of “W3C”
- SSML
  - Alternative to plain text
  - Speech Synthesis Markup Language
  - Gives control over emphasis, pronunciation, breathing, whispering, speech rate, pitch, pauses.
- Speech Marks
  - Can encode when sentence / word starts and ends in the audio stream
  - Useful for lip-synching animation



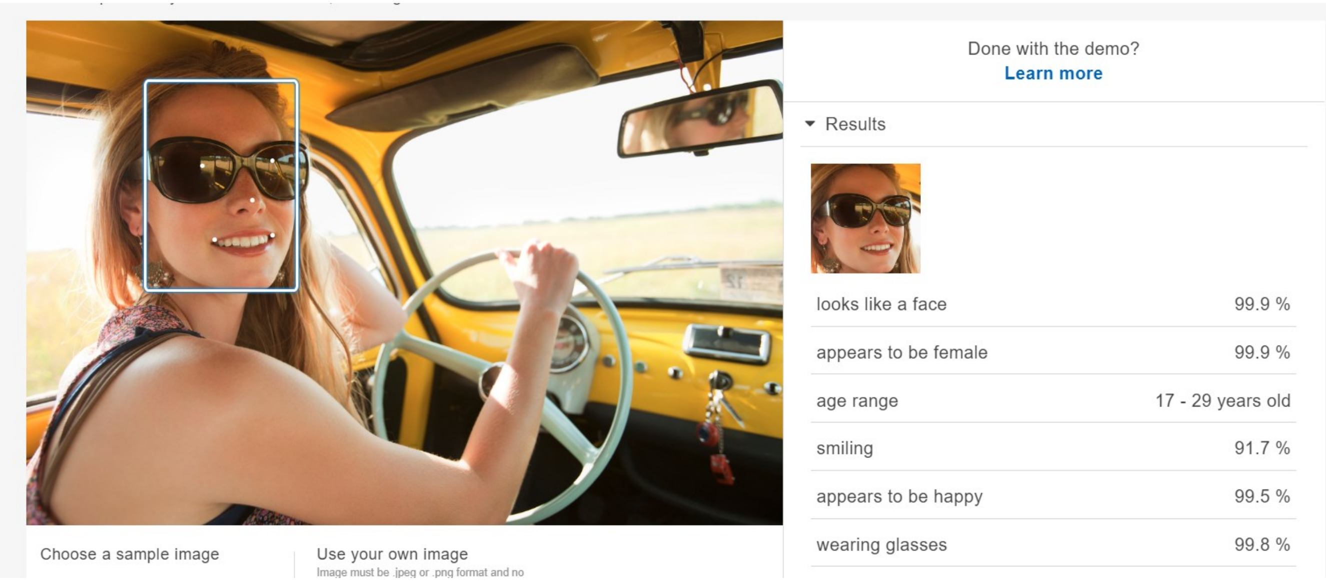
Amazon Polly



Rekognition

Rekognition

- Computer vision
- Object and scene detection
    - Can use your own face collection
- Image moderation
- Facial analysis
- Celebrity recognition
- Face comparison
- Text in image
- Video analysis
    - Objects / people / celebrities marked on timeline
    - People Pathing
- Image and video libraries

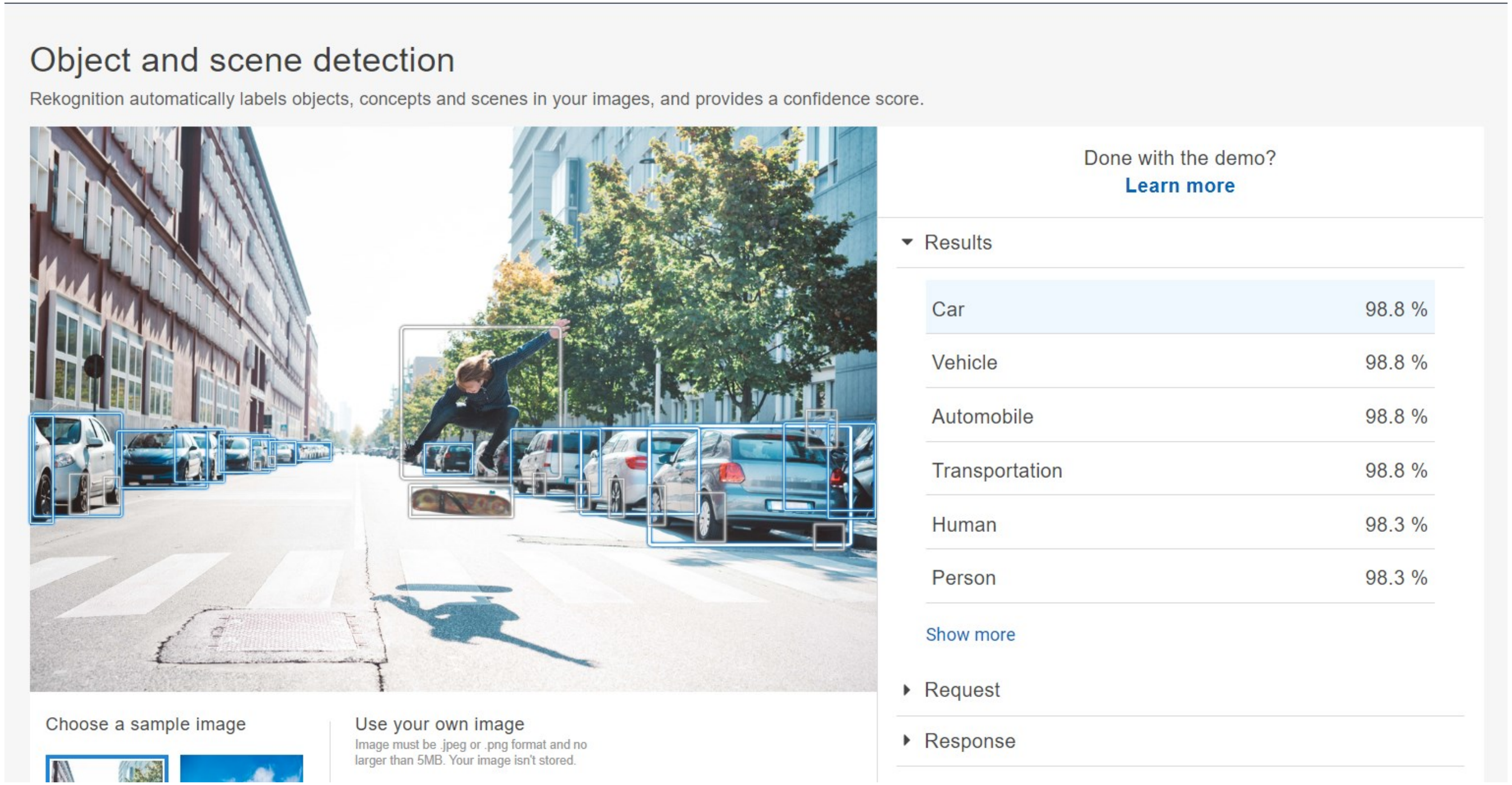


Rekognition: The Nitty Gritty

- Images come from S3, or provide image bytes as part of request
  - S3 will be faster if the image is already there
- Facial recognition depends on good lighting, angle, visibility of eyes, resolution
- Video must come from Kinesis Video Streams
  - H.264 encoded
  - 5-30 FPS
  - Favor resolution over framerate
- Can use with Lambda to trigger image analysis upon upload

DataCumulus Sundog Education

Rekognition



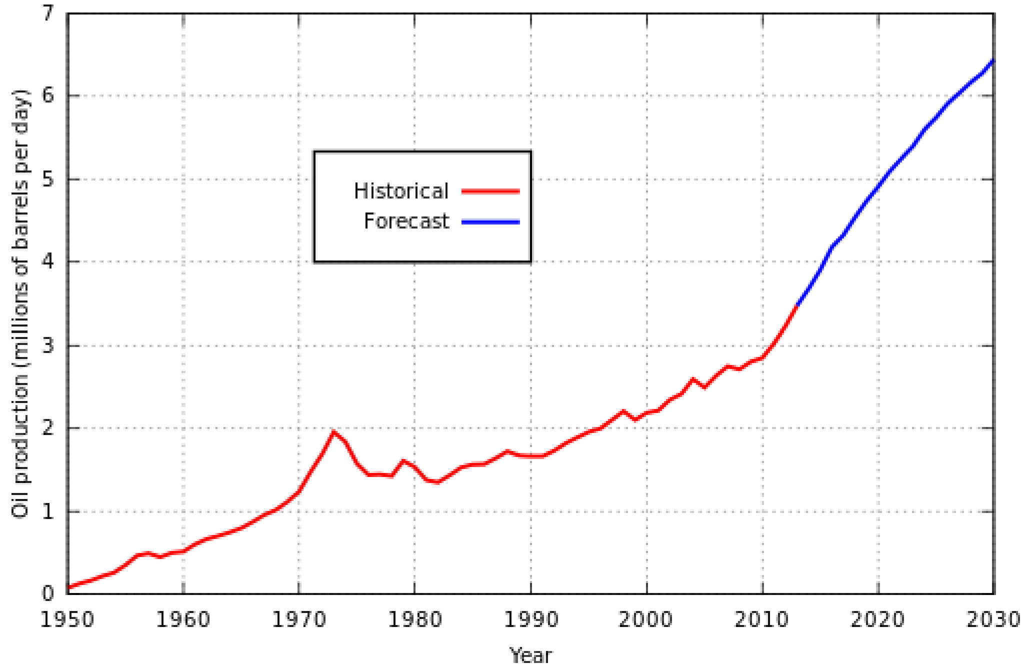
New in 2020: Rekognition Custom Labels

- Train with a small set of labeled images
- Use your own labels for unique items
- Example: the NFL (National Football League in the US) uses custom labels to identify team logos, pylons, and foam fingers in images.

Amazon Forecast

Amazon Forecast

- Fully-managed service to deliver highly accurate forecasts with ML
- "AutoML" chooses best model for your time series data
    - ARIMA, DeepAR, ETS, NPTS, CNN-QR Prophet
- Works with any time series
    - Price, promotions, economic performance, etc.
    - Can combine with associated data to find relationships
- Inventory planning, financial planning, resource planning
- Based on "dataset groups," "predictors," and "forecasts."



Ain92, using modified code by David "RockyMtnGuy" Moe [CC BY-SA 3.0 (https://creativecommons.org/licenses/by-sa/3.0)]

DataCumulus
Sundog Education

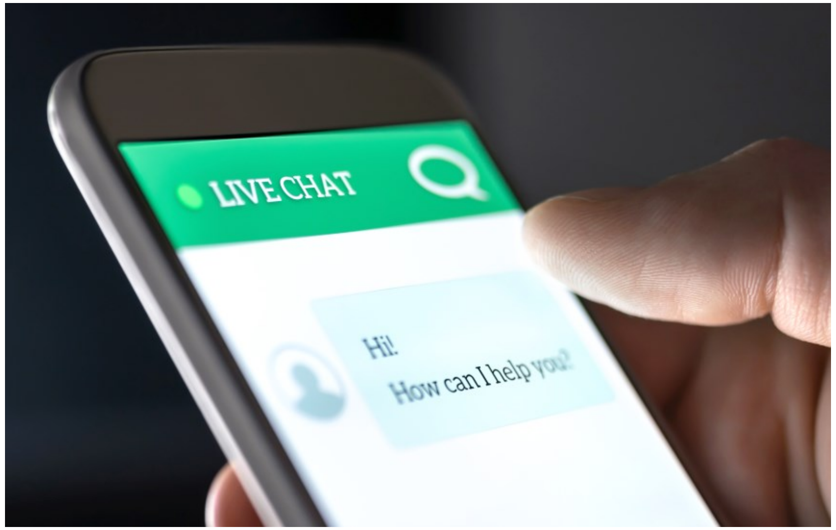
More on Forecast algorithms

- CNN-QR
  - Convolutional Neural Network – Quantile Regression
  - Best for large datasets with hundreds of time series
  - Accepts related historical time series data & metadata
- DeepAR+
  - Recurrent Neural Network
  - Best for large datasets
  - Accepts related forward-looking time series & metadata
- Prophet
  - Additive model with non-linear trends and seasonality
- NPTS
  - Non-Parametric Time Series
  - Good for sparse data. Has variants for seasonal / climatological forecasts
- ARIMA
  - Autoregressive Integrated Moving Average
  - Commonly used for simple datasets (<100 time series)
- ETS
  - Exponential Smoothing
  - Commonly used for simple datasets (<100 time series)

Amazon Lex

Amazon Lex

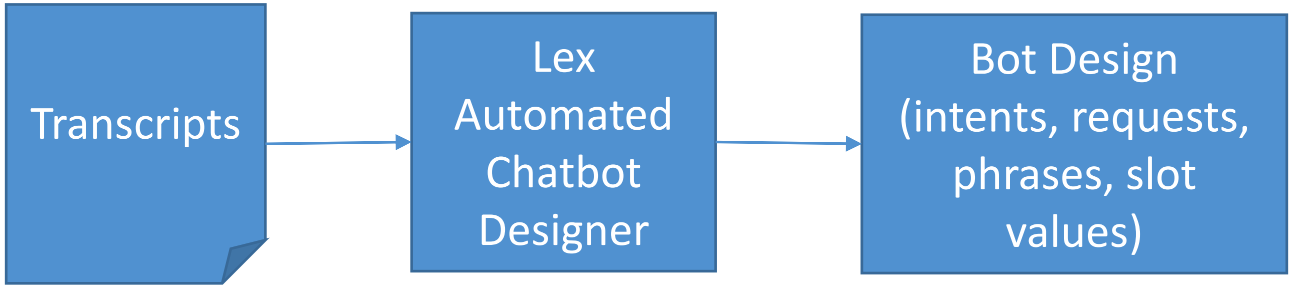
- Billed as the inner workings of Alexa
- Natural-language chatbot engine
- A Bot is built around Intents
  - Utterances invoke intents ("I want to order a pizza")
  - Lambda functions are invoked to fulfill the intent
  - Slots specify extra information needed by the intent
    - Pizza size, toppings, crust type, when to deliver, etc.
- Can deploy to AWS Mobile SDK, Facebook Messenger, Slack, and Twilio



DataCumulus Sundog Education

Amazon Lex Automated Chatbot Designer

- You provide existing conversation transcripts
- Lex applies NLP & deep learning, removing overlaps & ambiguity
- Intents, user requests, phrases, values for slots are extracted
- Ensures intents are well defined and separated
- Integrates with Amazon Connect transcripts



Amazon Personalize

Amazon Personalize

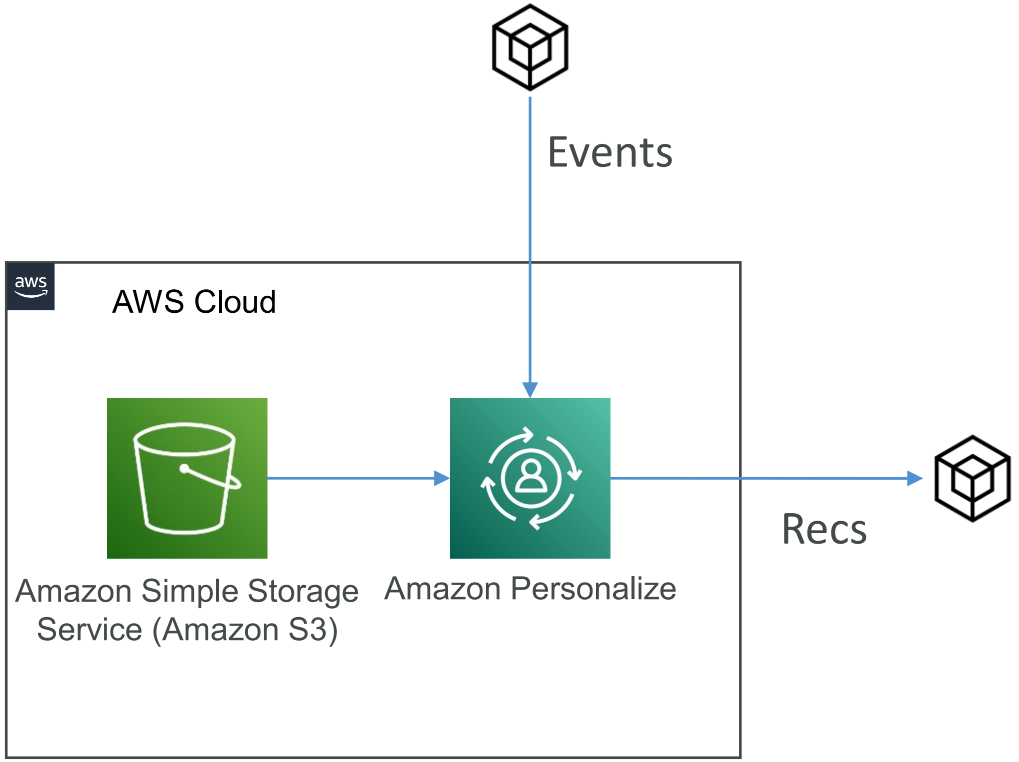
- Fully-managed recommender engine
  - Same one Amazon uses
- API access
  - Feed in data (purchases, ratings, impressions, cart adds, catalog, user demographics etc.) via S3 or API integration
  - You provide an explicit schema in Avro format
  - Javascript or SDK
  - GetRecommendations
    - Recommended products, content, etc.
    - Similar items
  - GetPersonalizedRanking
    - Rank a list of items provided
    - Allows editorial control / curation
- Console and CLI too

Amazon Personalize

DataCumulus Sundog Education

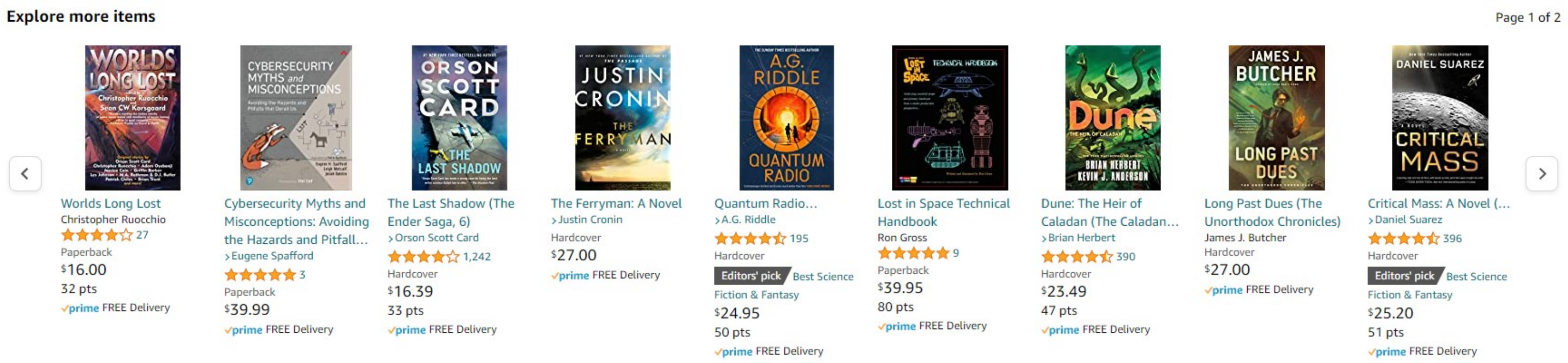
Amazon Personalize Features

- Real-time or batch recommendations
- Recommendations for new users and new items (the cold start problem)
- Contextual recommendations
  - Device type, time, etc.
- Similar items
- Unstructured text input
- Intelligent user segmentation
  - For marketing campaigns



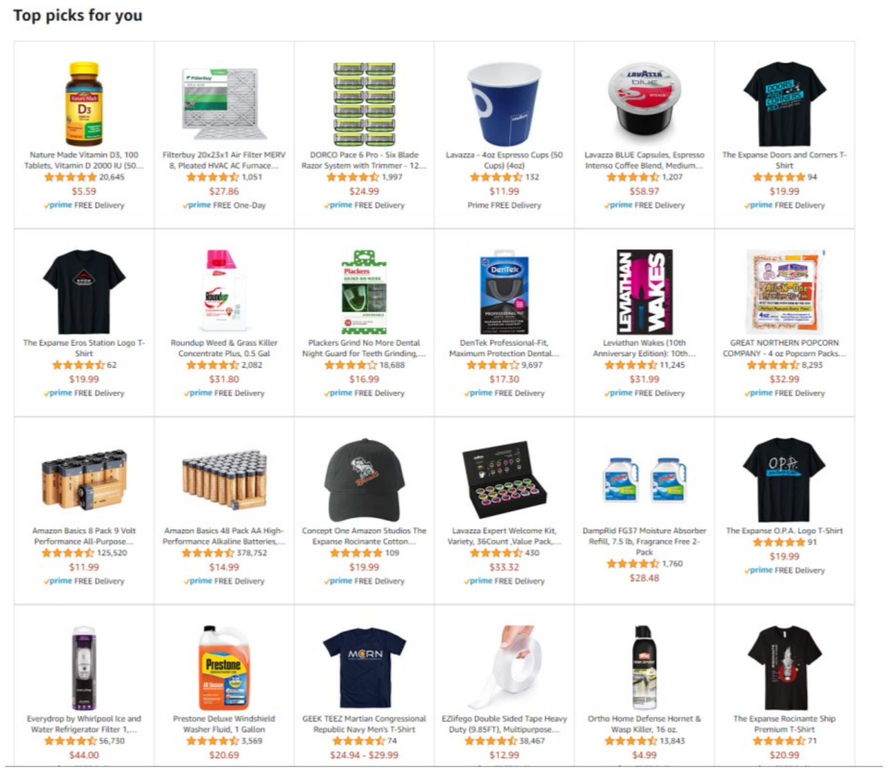
Amazon Personalize Features

- Business rules and filters
  - Filter out recently purchased items
  - Highlight premium content
  - Ensure a certain percentage of results are of some category
- Promotions
  - Inject promoted content into recommendations
  - Can find most relevant promoted content
- Trending Now
- Personalized Rankings
  - Search results
  - Promotions
  - Curated lists



Amazon Personalize Terminology

- Datasets
  - Users, Items, Interactions
- Recipes
  - USER_PERSONALIZATION
  - PERSONALIZED_RANKING
  - RELATED_ITEMS
  - USER_SEGMENTATION
- Solutions
  - Trains the model
  - Optimizes for relevance as well as your additional objectives
    - Video length, price, etc. – must be numeric
  - Hyperparameter Optimization (HPO)
- Campaigns
  - Deploys your "solution version"
  - Deploys capacity for generating real-time recommendations



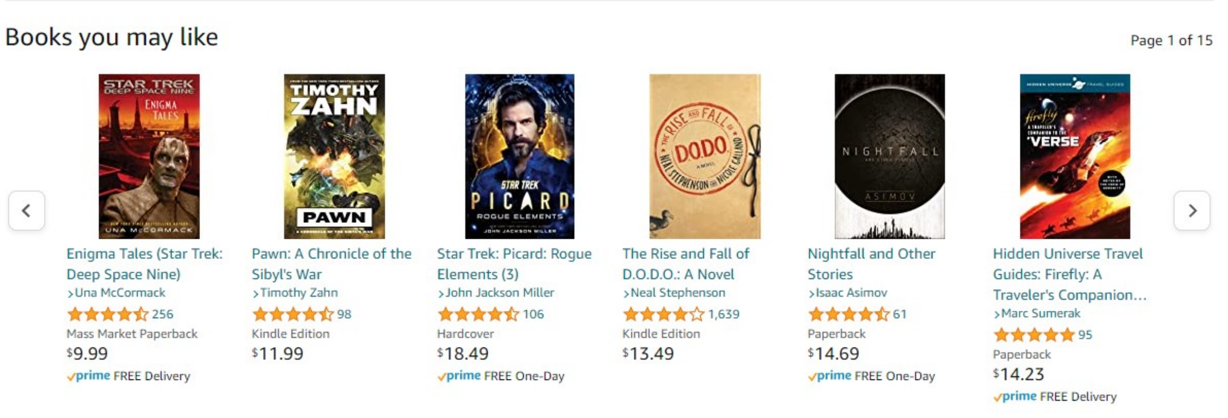
DataCumulus Sundog Education

Amazon Personalize Hyperparameters

- User-Personalization, Personalized-Ranking
  - `hidden_dimension` (HPO)
  - `bptt` (back-propagation through time - RNN)
  - `recency_mask` (weights recent events)
  - `min/max_user_history_length_percentile` (filter out robots)
  - `exploration_weight` 0-1, controls relevance
  - `exploration_item_age_cut_off` – how far back in time you go
- Similar-items
  - `item_id_hidden_dim` (HPO)
  - `item_metadata_hidden_dim` (HPO with min & max range specified)

Maintaining Relevance

- Keep your datasets current
  - Incremental data import
- Use PutEvents operation to feed in real-time user behavior
- Retrain the model
  - They call this a new *solution version*
  - Updates every 2 hours by default
  - Should do a full retrain (trainingMode=FULL) weekly



Amazon Personalize Security

- Data not shared across accounts
- Data may be encrypted with KMS
- Data may be encrypted at rest in your region (SSE-S3)
- Data in transit between your account and Amazon's internal systems encrypted with TLS 1.2
- Access control via IAM
- Data in S3 must have appropriate bucket policy for Amazon Personalize to process it
- Monitoring & logging via CloudWatch and CloudTrail

Amazon Personalize Pricing

- Data ingestion: per-GB
- Training: per training-hour
- Inference: per TPS-hour
- Batch recommendations: per user or per item

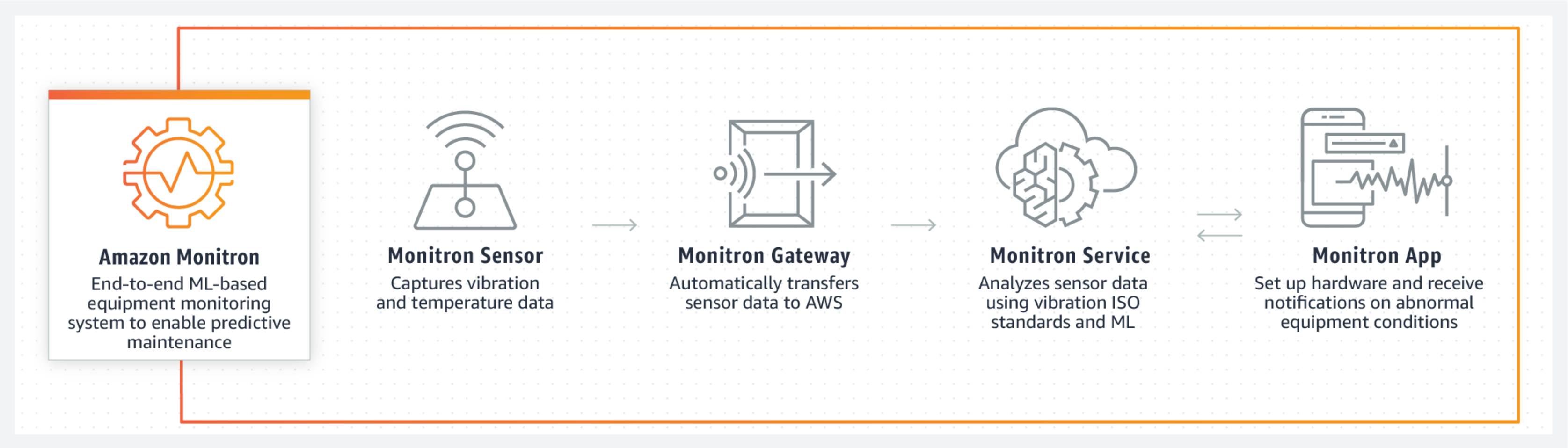
Other ML Services

The Best of the Rest

- Amazon Textract
  - OCR with forms, fields, tables support
- AWS DeepRacer
  - Reinforcement learning powered 1/18-scale race car
- DeepLens
  - Deep learning-enabled video camera
  - Integrated with Rekognition, SageMaker, Polly, Tensorflow, MXNet, Caffe

Industrial Applications

- Amazon Lookout
  - Equipment, metrics, vision
  - Detects abnormalities from sensor data automatically to detect equipment issues
  - Monitors metrics from S3, RDS, Redshift, 3rd party SaaS apps
  - Vision uses computer vision to detect defects in silicon wafers, circuit boards, etc.
- Amazon Monitron
  - End to end system for monitoring industrial equipment & predictive maintenance



TorchServe and Neuron

- TorchServe
  - Model serving framework for PyTorch
  - Part of the PyTorch open source project from Facebook (Meta?)
- AWS Neuron
  - SDK for ML inference specifically on AWS Inferentia chips
  - EC2 Inf1 instance type
  - Integrated with SageMaker or whatever else you want (deep learning AMI's, containers, Tensorflow, PyTorch, MXNet)

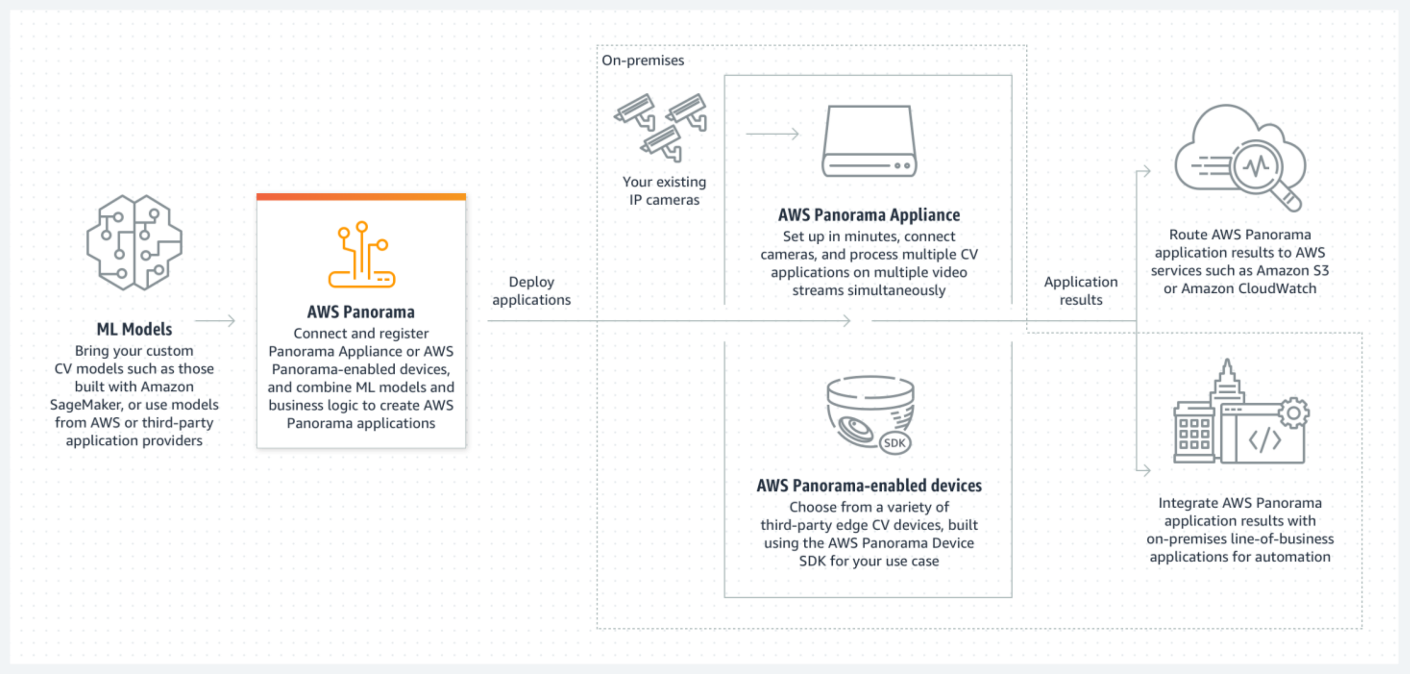
TorchServe

AWS Neuron

AWS Panorama

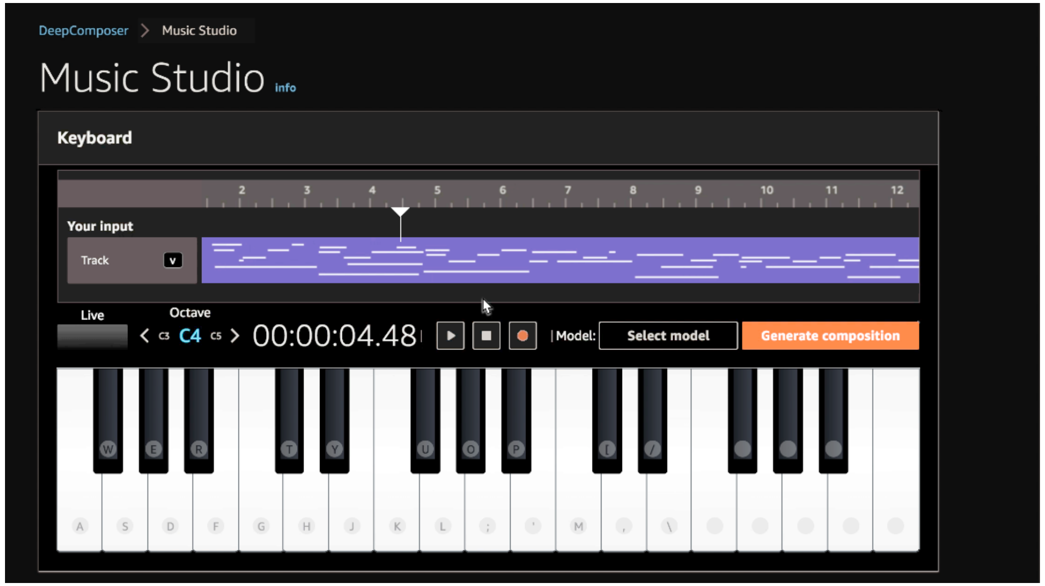
- Computer Vision at the edge
- Brings computer vision to your existing IP cameras

AWS Panorama



AWS DeepComposer

- AI-powered keyboard
- Composes a melody into an entire song
- For educational purposes

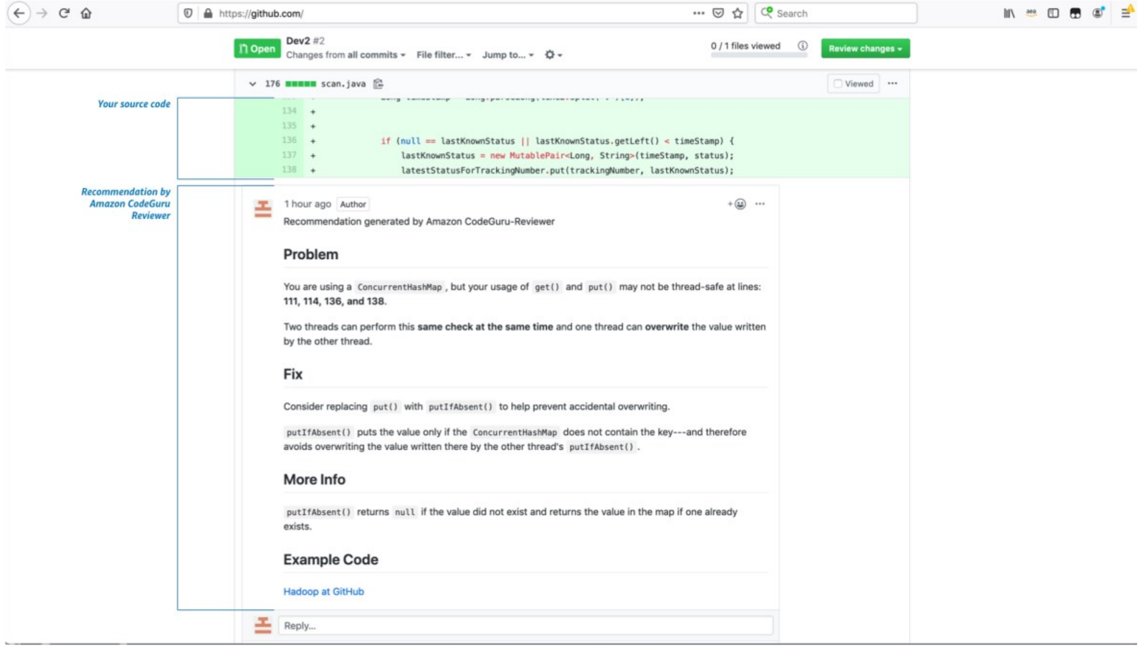


Amazon Fraud Detector

- Upload your own historical fraud data
- Builds custom models from a template you choose
- Exposes an API for your online application
- Assess risk from:
  - New accounts
  - Guest checkout
  - “Try before you buy” abuse
  - Online payments

Amazon CodeGuru

- Automated code reviews!
- Finds lines of code that hurt performance
- Resource leaks, race conditions
- Fix security vulnerabilities
- Offers specific recommendations
- Powered by ML
- Supports Java and Python



DataCumulus Sundog™ Education

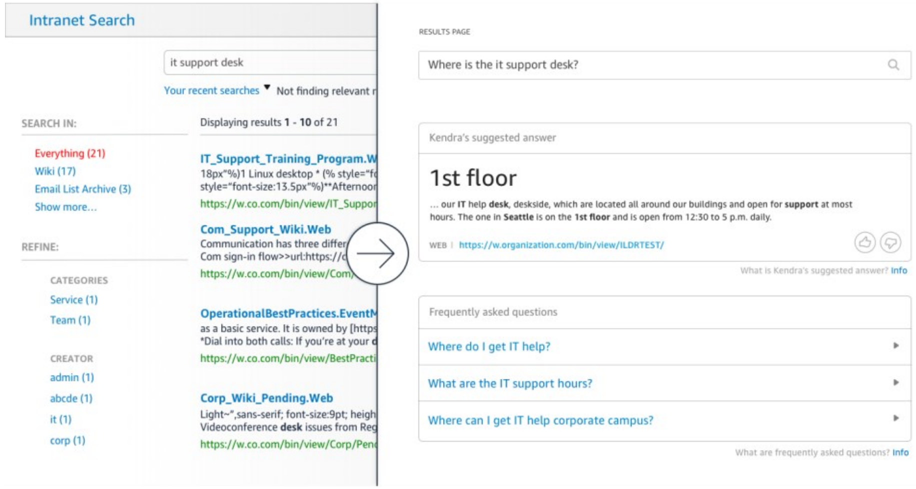
Contact Lens for Amazon Connect

- For customer support call centers
- Ingests audio data from recorded calls
- Allows search on calls / chats
- Sentiment analysis
- Find "utterances" that correlate with successful calls
- Categorize calls automatically
- Measure talk speed and interruptions
- Theme detection: discovers emerging issues

DataCumulus Sundog Education

Amazon Kendra

- Enterprise search with natural language
- For example, “Where is the IT support desk?” “How do I connect to my VPN?”
- Combines data from file systems, SharePoint, intranet, sharing services (JDBC, S3) into one searchable repository
- ML-powered (of course) – uses thumbs up / down feedback
- Relevance tuning – boost strength of document freshness, view counts, etc.
- Alexa’s sister? I don’t know, but that’s one way to remember it ☺



Amazon Augmented AI (A2I)

- Human review of ML predictions
- Builds workflows for reviewing low-confidence predictions
- Access the Mechanical Turk workforce or vendors
- Integrated into Amazon Textract and Rekognition
- Integrates with SageMaker
- Very similar to Ground Truth

Modeling, Part 4: Wrapping up & Lab Activity

Putting Them Together

Putting the blocks together

- Build your own Alexa!
  - Transcribe -> Lex -> Polly
- Make a universal translator!
  - Transcribe -> Translate -> Polly
- Build a Jeff Bezos detector!
  - DeepLens -> Rekognition
- Are people on the phone happy?
  - Transcribe -> Comprehend

ML Implementation and Operations

ML Implementations and Operations

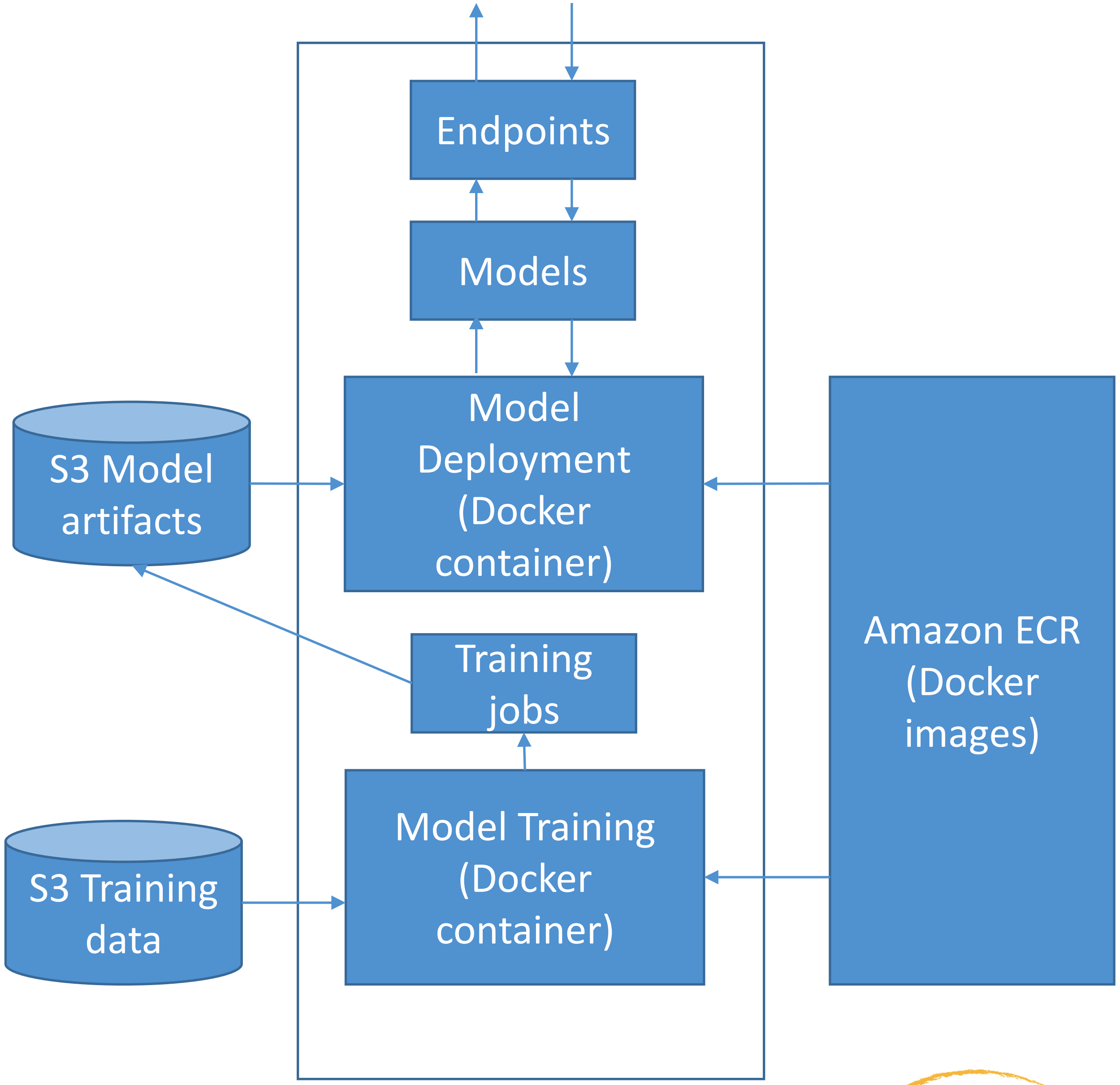
SageMaker and Docker Containers

SageMaker + Docker

- All models in SageMaker are hosted in Docker containers
  - Pre-built deep learning
  - Pre-built scikit-learn and Spark ML
  - Pre-built Tensorflow, MXNet, Chainer, PyTorch
    - Distributed training via Horovod or Parameter Servers
  - Your own training and inference code! Or extend a pre-built image.
- This allows you to use any script or algorithm within SageMaker, regardless of runtime or language
  - Containers are isolated, and contain all dependencies and resources needed to run

Using Docker

- Docker containers are created from *images*
- Images are built from a *Dockerfile*
- Images are saved in a *repository*
  - Amazon Elastic Container Registry



Amazon SageMaker Containers

- Library for making containers compatible with SageMaker
  - `RUN pip install sagemaker-containers` in your Dockerfile

Structure of a training container

```
/opt/ml
├── input
│   ├── config
│   │   ├── hyperparameters.json
│   │   └── resourceConfig.json
│   └── data
│       └── <channel_name>
│           └── <input data>
├── model
│
├── code
│   └── <script files>
│
└── output
└── failure
```

Structure of a Deployment Container

```
/opt/ml
└── model
    └── <model files>
```

Structure of your Docker image

- WORKDIR
  - nginx.conf
  - predictor.py
  - serve/
  - train/
  - wsgi.py

DataCumulus Sundog™ Education

Assembling it all in a Dockerfile

```
FROM tensorflow/tensorflow:2.0.0a0

RUN pip3 install sagemaker-training

# Copies the training code inside the
container
COPY train.py /opt/ml/code/train.py

# Defines train.py as script entrypoint
ENV SAGEMAKER_PROGRAM train.py
```

Environment variables

- SAGEMAKER_PROGRAM
  - Run a script inside /opt/ml/code
- SAGEMAKER_TRAINING_MODULE
- SAGEMAKER_SERVICE_MODULE
- SM_MODEL_DIR
- SM_CHANNELS / SM_CHANNEL_*
- SM_HPS / SM_HP_*
- SM_USER_ARGS
- …and many more

Using your own image

- cd dockerfile
- !docker build -t foo .
- from sagemaker.estimator import Estimator

```
estimator = Estimator(image_name='foo',
                      role='SageMakerRole',
                      train_instance_count=1,
                      train_instance_type='local')

estimator.fit()
```


DataCumulus Sundog Education

Production Variants

- You can test out multiple models on live traffic using Production Variants
  - Variant Weights tell SageMaker how to distribute traffic among them
  - So, you could roll out a new iteration of your model at say 10% variant weight
  - Once you're confident in its performance, ramp it up to 100%
- This lets you do A/B tests, and to validate performance in real-world settings
  - Offline validation isn't always useful
- Shadow Variants
- Deployment Guardrails

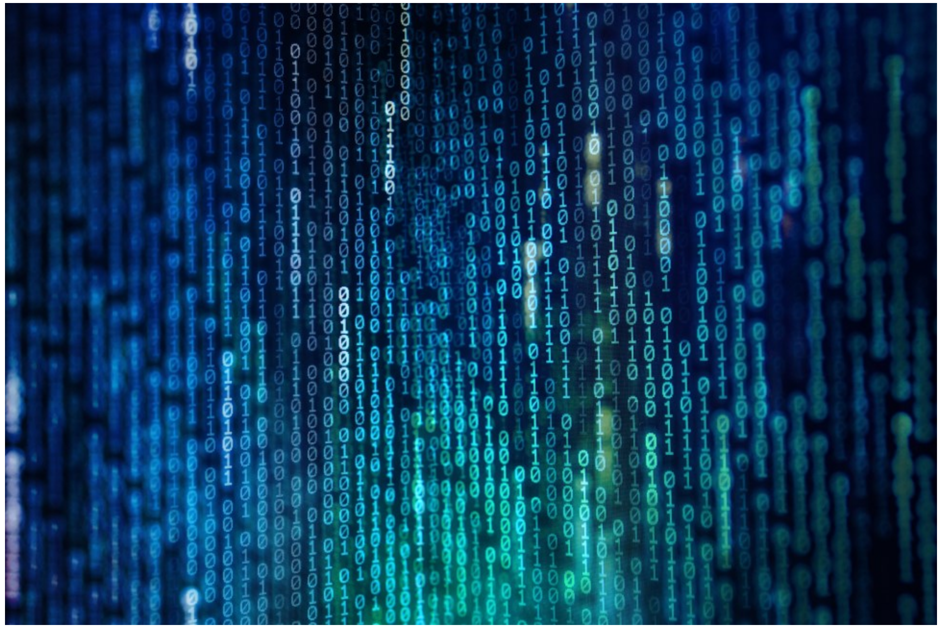
SageMaker on the Edge

DataCumulus

Sundog™
Education

SageMaker Neo

- Train once, run anywhere
  - Edge devices
    - ARM, Intel, Nvidia processors
    - Embedded in whatever – your car?
- Optimizes code for specific devices
  - Tensorflow, MXNet, PyTorch, ONNX, XGBoost, DarkNet, Keras
- Consists of a **compiler** and a **runtime**



Neo + AWS IoT Greengrass

- Neo-compiled models can be deployed to an HTTPS endpoint
  - Hosted on C5, M5, M4, P3, or P2 instances
  - Must be same instance type used for compilation
- OR! You can deploy to IoT Greengrass
  - This is how you get the model to an actual edge device
  - Inference at the edge with local data, using model trained in the cloud
  - Uses Lambda inference applications

SageMaker Security

DataCumulus Sundog™ Education

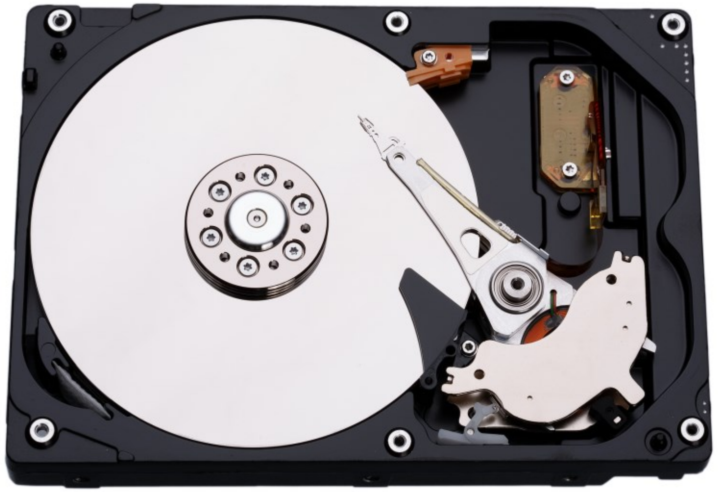
General AWS Security

- Use Identity and Access Management (IAM)
  - Set up user accounts with only the permissions they need
- Use MFA
- Use SSL/TLS when connecting to anything
- Use CloudTrail to log API and user activity
- Use encryption
- Be careful with PII

DataCumulus Sundog Education

Protecting your Data at Rest in SageMaker

- AWS Key Management Service (KMS)
  - Accepted by notebooks and all SageMaker jobs
    - Training, tuning, batch transform, endpoints
    - Notebooks and everything under /opt/ml/ and /tmp can be encrypted with a KMS key
- S3
  - Can use encrypted S3 buckets for training data and hosting models
  - S3 can also use KMS



Protecting Data in Transit in SageMaker

- All traffic supports TLS / SSL
- IAM roles are assigned to SageMaker to give it permissions to access resources
- Inter-node training communication may be optionally encrypted
  - Can increase training time and cost with deep learning
  - AKA inter-container traffic encryption
  - Enabled via console or API when setting up a training or tuning job

DataCumulus Sundog Education

SageMaker + VPC

- Training jobs run in a Virtual Private Cloud (VPC)
- You can use a private VPC for even more security
  - You'll need to set up S3 VPC endpoints
  - Custom endpoint policies and S3 bucket policies can keep this secure
- Notebooks are Internet-enabled by default
  - This can be a security hole
  - **If disabled, your VPC needs an interface endpoint (PrivateLink) or NAT Gateway, and allow outbound connections, for training and hosting to work**
- Training and Inference Containers are also Internet-enabled by default
  - Network isolation is an option, but this also prevents S3 access

SageMaker + IAM

- User permissions for:
  - CreateTrainingJob
  - CreateModel
  - CreateEndpointConfig
  - CreateTransformJob
  - CreateHyperParameterTuningJob
  - CreateNotebookInstance
  - UpdateNotebookInstance
- Predefined policies:
  - AmazonSageMakerReadOnly
  - AmazonSageMakerFullAccess
  - AdministratorAccess
  - DataScientist

SageMaker Logging and Monitoring

- CloudWatch can log, monitor and alarm on:
  - Invocations and latency of endpoints
  - Health of instance nodes (CPU, memory, etc)
  - Ground Truth (active workers, how much they are doing)
- CloudTrail records actions from users, roles, and services within SageMaker
  - Log files delivered to S3 for auditing

Managing SageMaker Resources

Choosing your instance types

- We covered this under "modeling", even though it's an operations concern
- In general, algorithms that rely on deep learning will benefit from GPU instances (P3, g4dn) for training
- Inference is usually less demanding and you can often get away with compute instances there (C5)
- GPU instances can be really pricey

Managed Spot Training

- Can use EC2 Spot instances for training
  - Save up to 90% over on-demand instances
- Spot instances can be interrupted!
  - Use checkpoints to S3 so training can resume
- Can increase training time as you need to wait for spot instances to become available

Elastic Inference

- Accelerates deep learning inference
  - At fraction of cost of using a GPU instance for inference
- EI accelerators may be added alongside a CPU instance
  - ml.eia1.medium / large / xlarge
- EI accelerators may also be applied to notebooks
- Works with Tensorflow, PyTorch, and MXNet pre-built containers
  - ONNX may be used to export models to MXNet
- Works with custom containers built with EI-enabled Tensorflow, PyTorch, or MXNet
- Works with Image Classification and Object Detection built-in algorithms

DataCumulus Sundog Education

Automatic Scaling

- You set up a scaling policy to define target metrics, min/max capacity, cooldown periods
- Works with CloudWatch
- Dynamically adjusts number of instances for a production variant
- Load test your configuration before using it!

Serverless Inference

- Introduced for 2022
- Specify your container, memory requirement, concurrency requirements
- Underlying capacity is automatically provisioned and scaled
- Good for infrequent or unpredictable traffic; will scale down to zero when there are no requests.
- Charged based on usage
- Monitor via CloudWatch
  - ModelSetupTime, Invocations, MemoryUtilization

Amazon SageMaker Serverless Inference

Fully managed serverless endpoints for machine learning inference with pay-per-use pricing

Amazon SageMaker Inference Recommender

- Recommends best instance type & configuration for your models
- Automates load testing model tuning
- Deploys to optimal inference endpoint
- How it works:
    - Register your model to the model registry
    - Benchmark different endpoint configurations
    - Collect & visualize metrics to decide on instance types
    - Existing models from zoos may have benchmarks already
- Instance Recommendations
    - Runs load tests on recommended instance types
    - Takes about 45 minutes
- Endpoint Recommendations
    - Custom load test
    - You specify instances, traffic patterns, latency requirements, throughput requirements
    - Takes about 2 hours


```
'InferenceRecommendations': [{
        'Metrics': {
          'CostPerHour': 0.20399999618530273,
          'CostPerInference': 5.246913588052848e-06,
          'MaximumInvocations': 648,
          'ModelLatency': 263596
          },
        'EndpointConfiguration': {
          'EndpointName': 'endpoint-name',
          'VariantName': 'variant-name',
          'InstanceType': 'ml.c5.xlarge',
          'InitialInstanceCount': 1
          },
        'ModelConfiguration': {
          'Compiled': False,
          'EnvironmentParameters': []
          }
      },
```


DataCumulus Sundog™ Education

SageMaker and Availability Zones

- SageMaker automatically attempts to distribute instances across availability zones
- But you need more than one instance for this to work!
- Deploy multiple instances for each production endpoint
- Configure VPC's with at least two subnets, each in a different AZ

MLOps with SageMaker

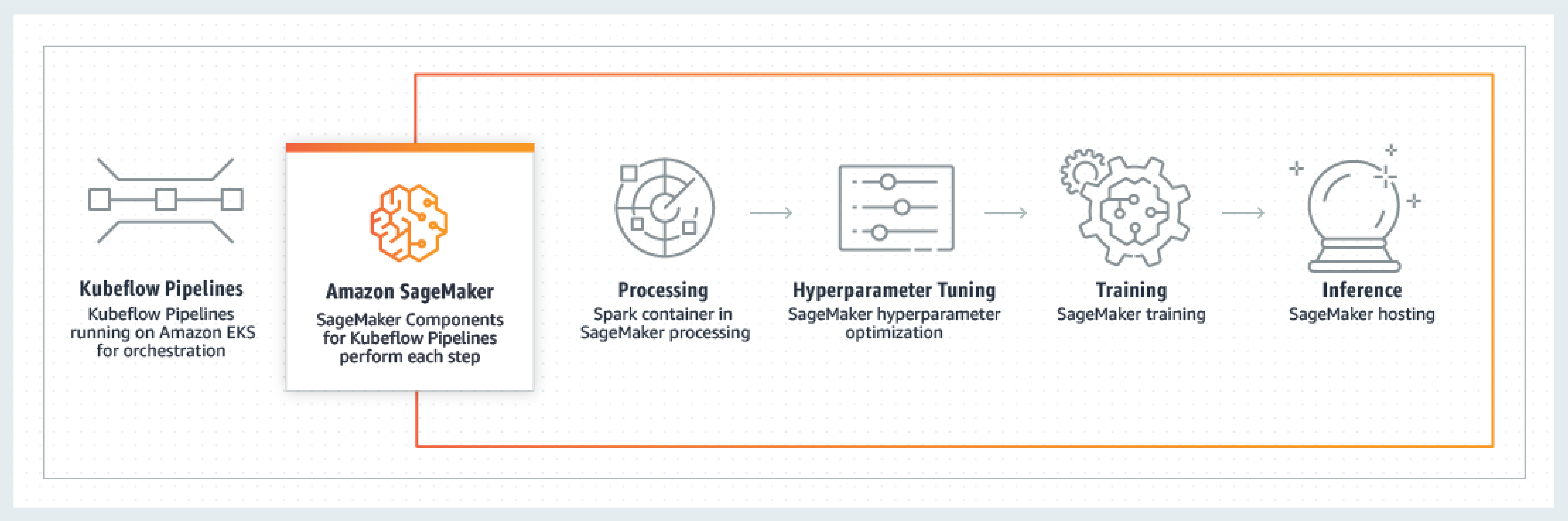
MLOps with SageMaker and Kubernetes

- Integrates SageMaker with Kubernetes-based ML infrastructure
- Amazon SageMaker Operators for Kubernetes
- Components for Kubeflow Pipelines
- Enables hybrid ML workflows (on-prem + cloud)
- Enables integration of existing ML platforms built on Kubernetes / Kubeflow

SageMaker Operators for Kubernetes

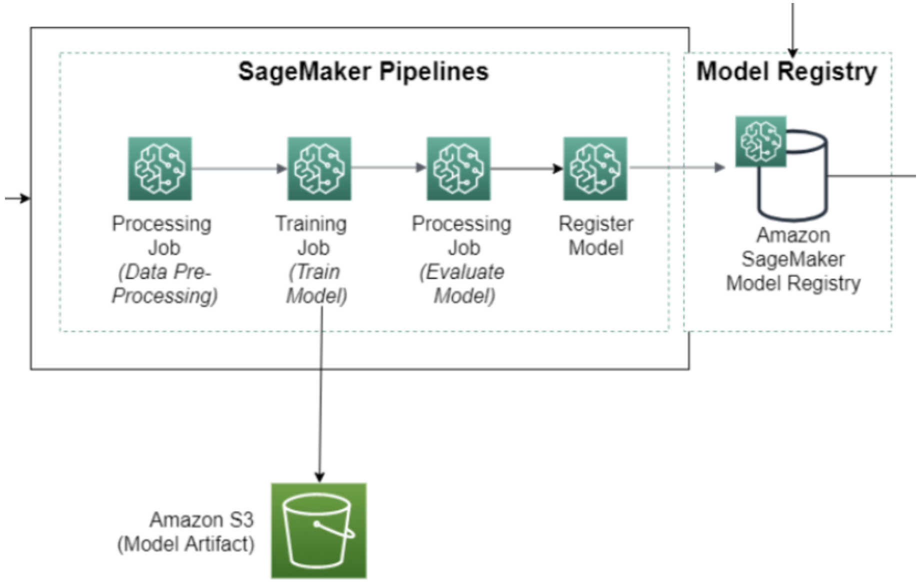
DataCumulus Sundog Education

SageMaker Components for Kubeflow Pipelines



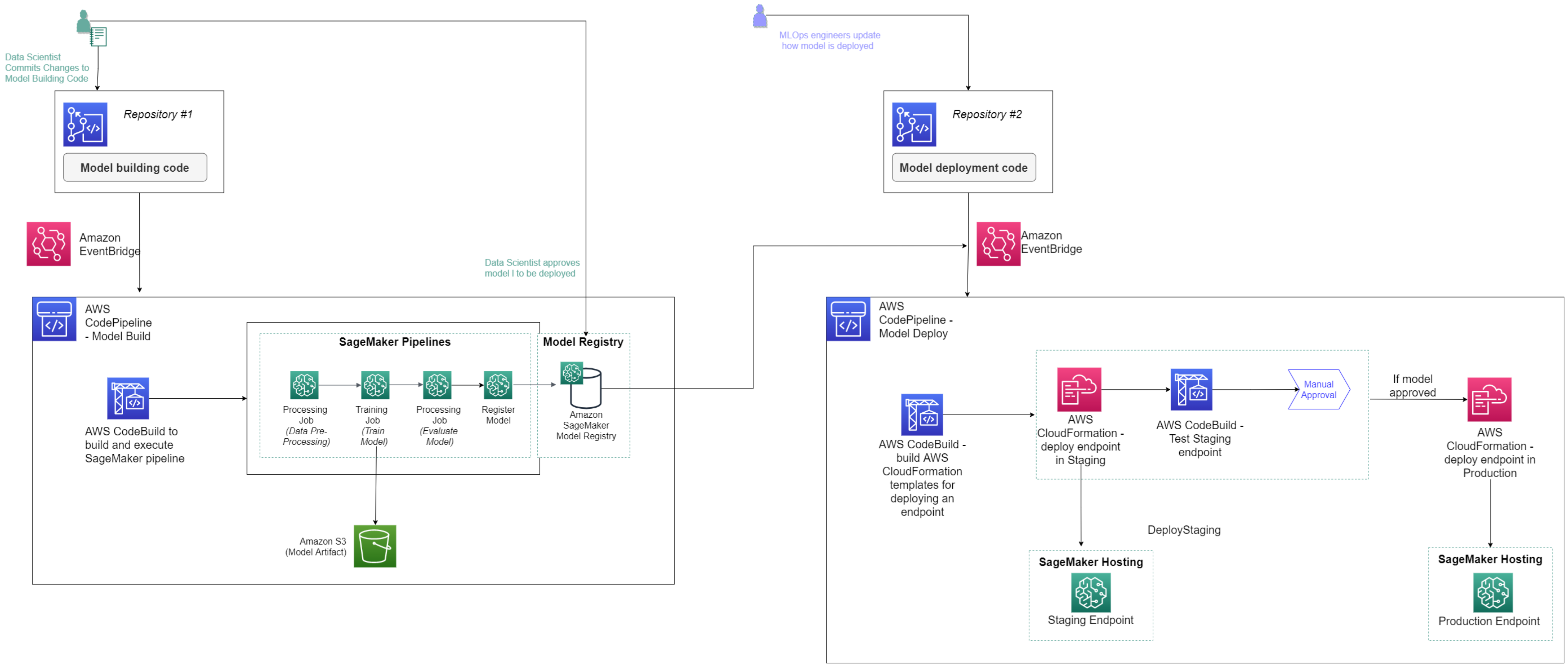
SageMaker Projects

- SageMaker Studio's native MLOps solution with CI/CD
  - Build images
  - Prep data, feature engineering
  - Train models
  - Evaluate models
  - Deploy models
  - Monitor & update models
- Uses code repositories for building & deploying ML solutions
- Uses SageMaker Pipelines defining steps



DataCumulus Sundog Education

SageMaker Projects



Inference Pipelines

Inference Pipelines

- Linear sequence of 2-15 containers
- Any combination of pre-trained built-in algorithms or your own algorithms in Docker containers
- Combine pre-processing, predictions, post-processing
- Spark ML and scikit-learn containers OK
  - Spark ML can be run with Glue or EMR
  - Serialized into MLeap format
- Can handle both real-time inference and batch transforms

Transformers and Generative AI

Large language models, transformers, GPT, and AWS support

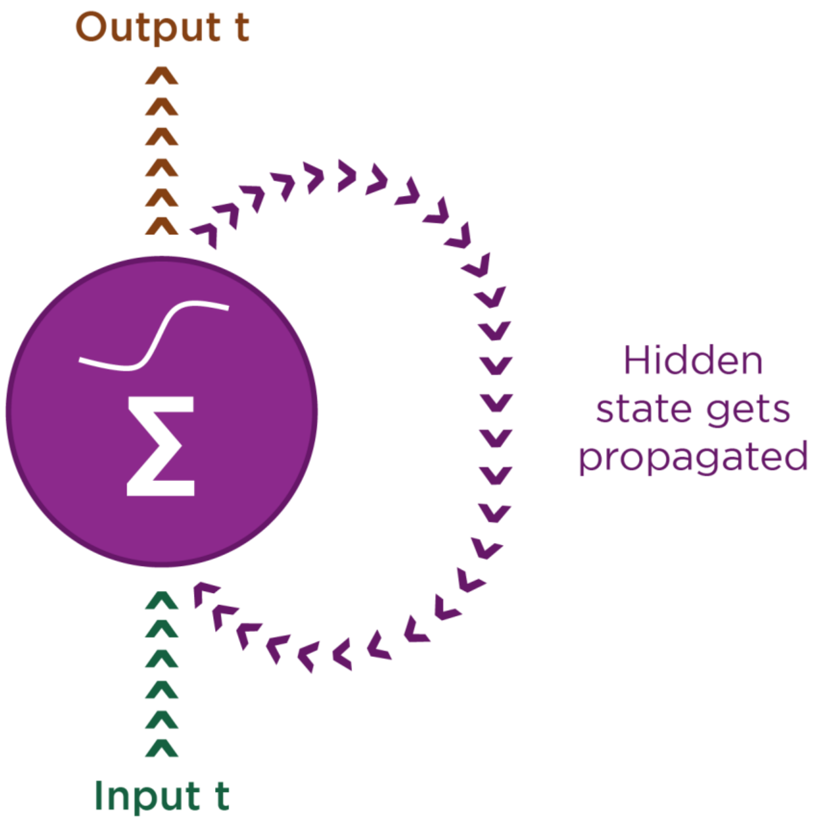
DataCumulus Sundog™ Education

The Transformer Architecture

And how self-attention works its magic

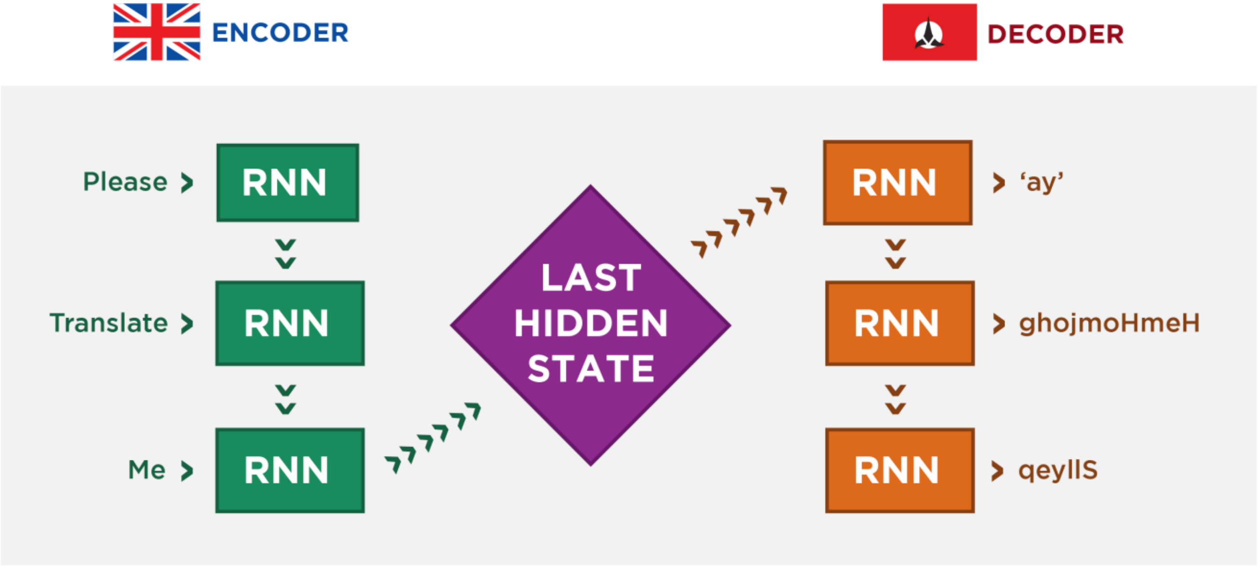
The Evolution of Transformers

- RNN's, LSTMs
- Introduced a feedback loop for propagating information forward
- Useful for modeling sequential things
  - Time series
  - Language! A sequence of words (or tokens)



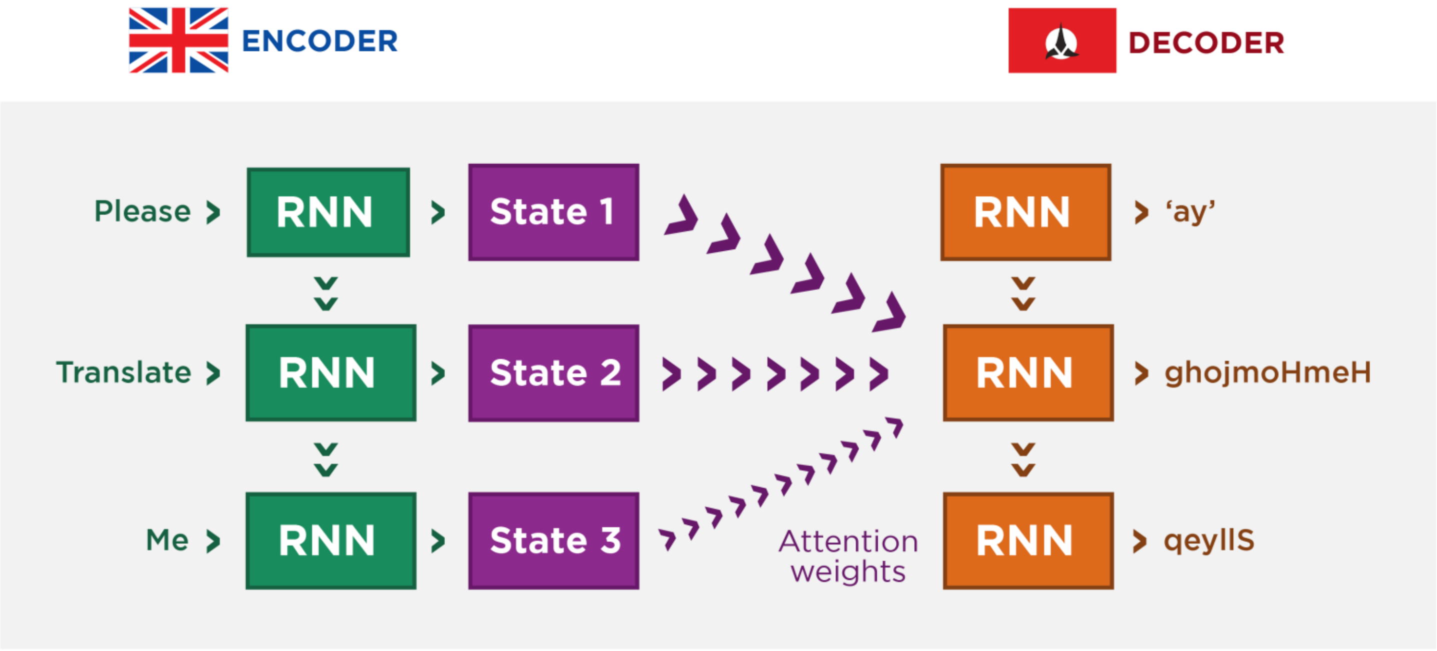
The Evolution of Transformers

- Machine translation
- Encoder / Decoder architecture
- Encoders and Decoders are RNN's
- But, the one vector tying them together creates an information bottleneck
  - Information from the start of the sequence may be lost



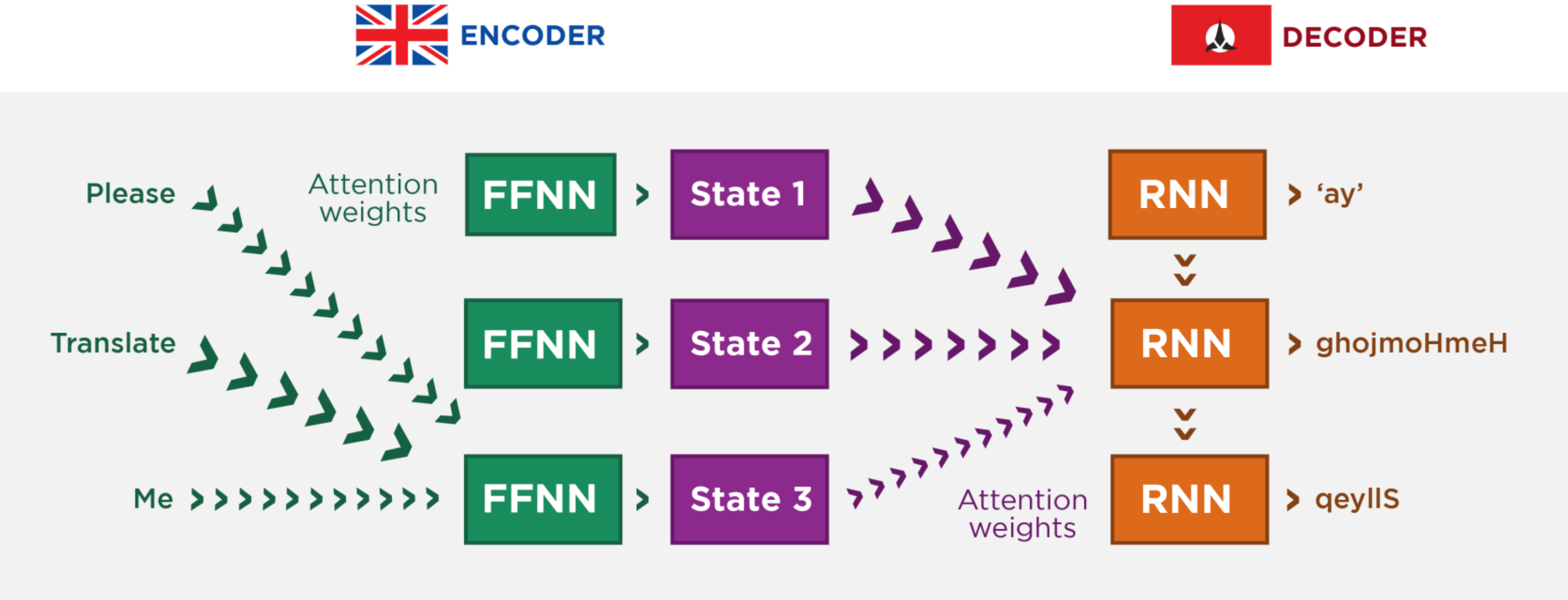
“Attention is all you need”

- A hidden state for each step (token)
- Deals better with differences in word order
- Starts to have a concept of relationships between words
- But RNN’s are still sequential in nature, can’t parallelize it



Transformers

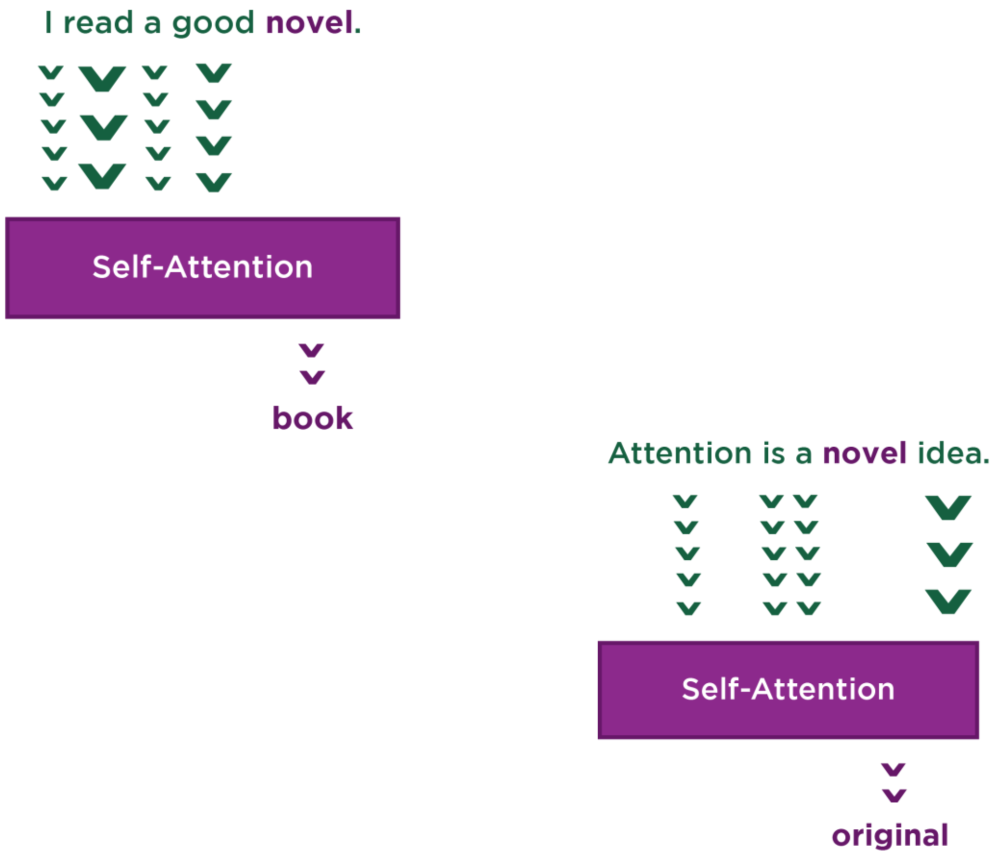
- Ditch RNN’s for feed-forward neural networks (FFNN’s)
- Use “self-attention”
- This makes it parallelizable (so can train on much more data)



DataCumulus Sundog Education

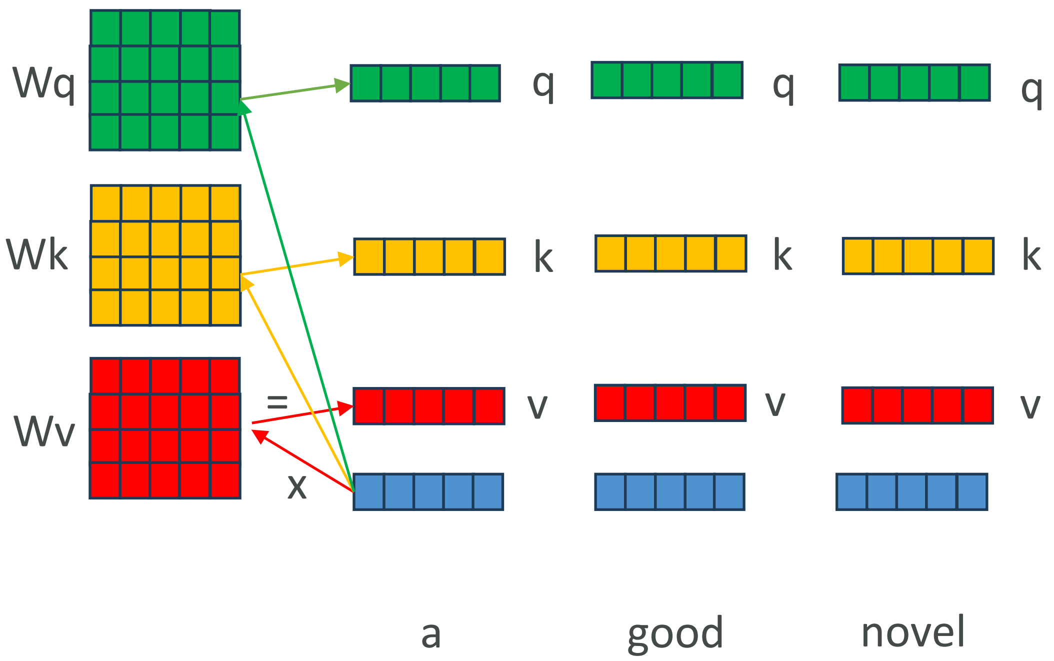
Self-Attention (in more depth)

- Each encoder or decoder has a list of embeddings (vectors) for each token
- Self-attention produces a *weighted average* of all token embeddings. The magic is in computing the attention weights.
- This results in tokens being tied to other tokens that are important for its context, and a new embedding that captures its "meaning" in context.



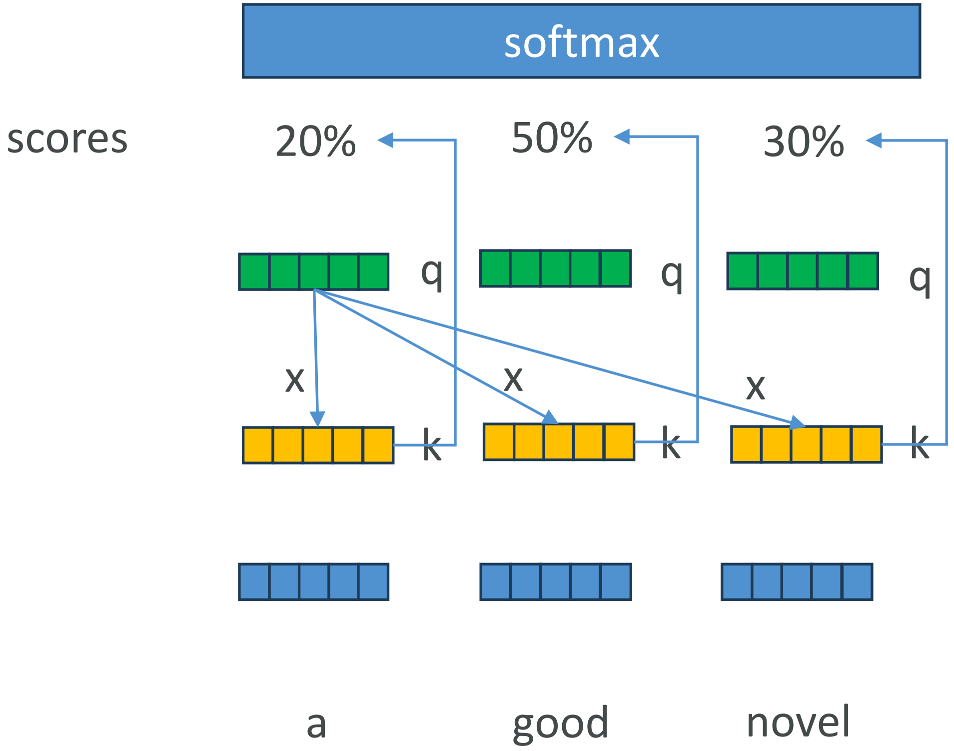
Self-Attention (in more depth)

- Three matrices of weights are learned through back-propagation
  - Query (Wq)
  - Key (Wk)
  - Value (Wv)
- Every token gets a query (q), key (k), and value (v) vector by multiplying its embedding against these matrices



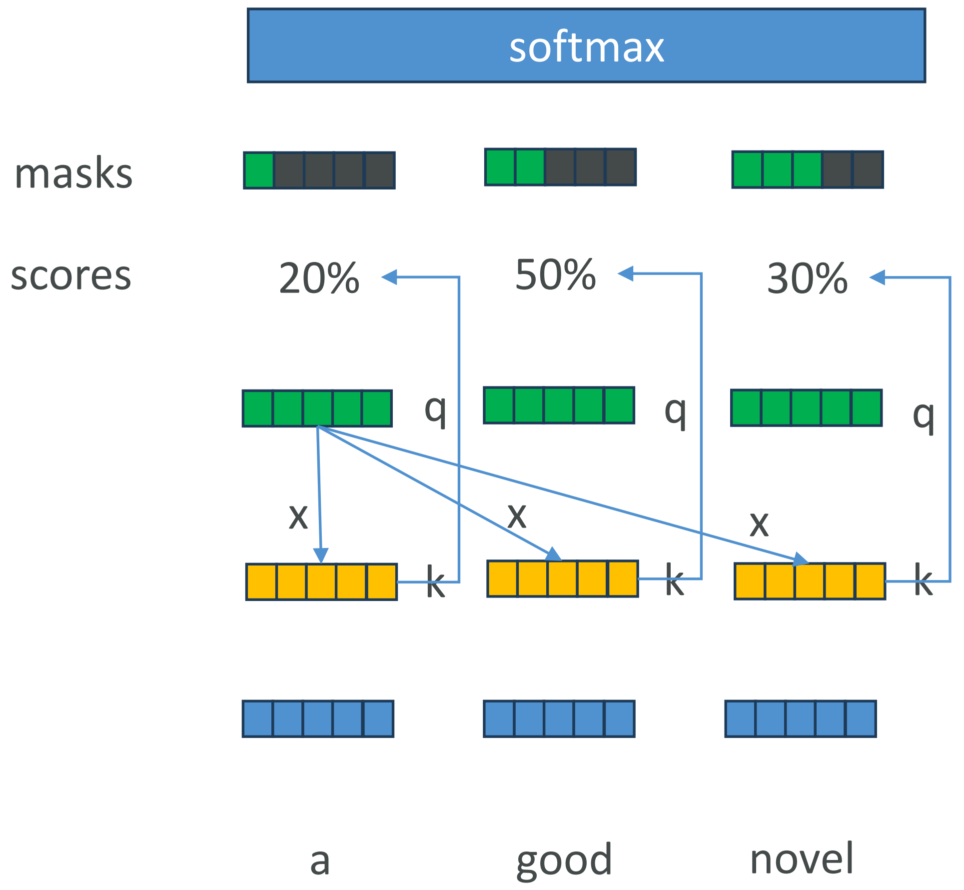
Self-Attention (in more depth)

- Compute a *score* for each token by multiplying (dot product) its query with each key
- "Scaled dot-product attention"
- Dot product is just one *similarity function* we can use.
- In practice, softmax is then applied to the scores to normalize them.



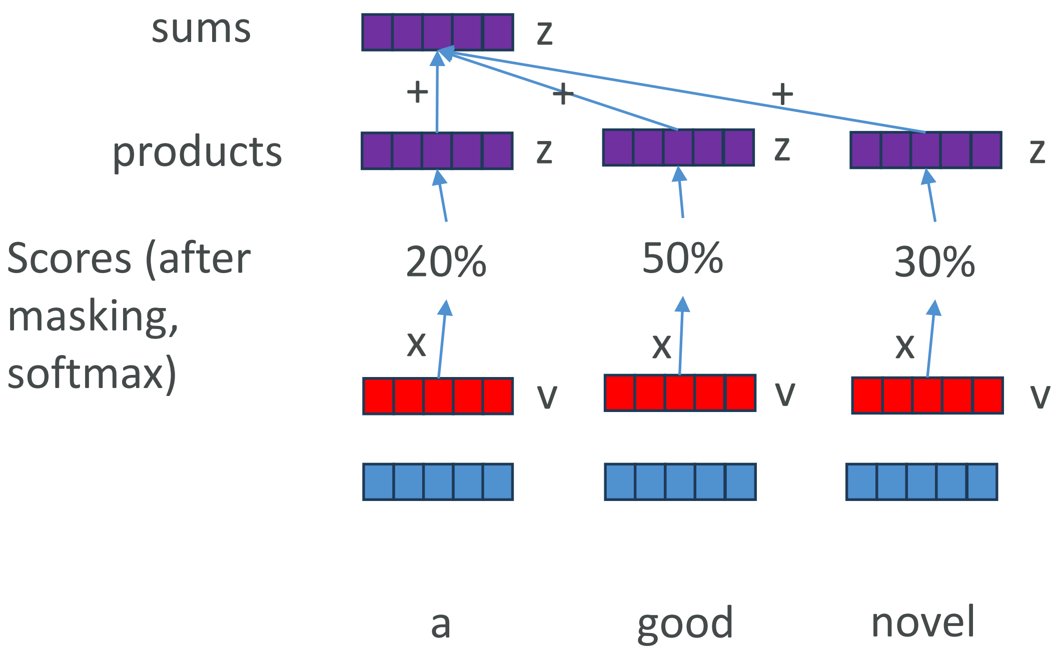
Masked Self-Attention

- A mask can be applied to prevent tokens from "peeking" into future tokens (words)
- GPT does this, but BERT does something else (masked language modeling)
- In this example, "good" wouldn't be affected by "novel", but "novel" could by affected by "good".
- This is just the concept… actual implementation details will vary.



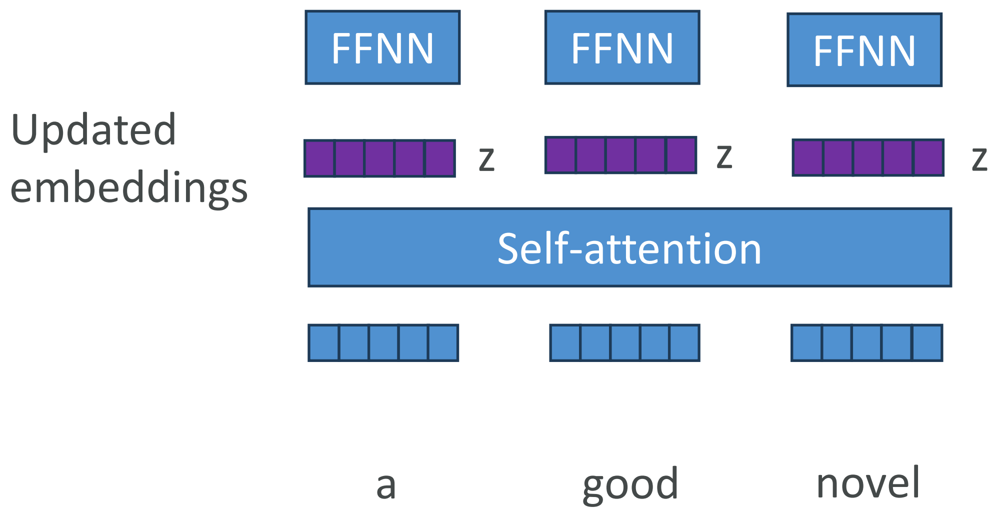
Self-Attention (in more depth)

- Now we sum.
- Multiply values by scores, and sum them up.



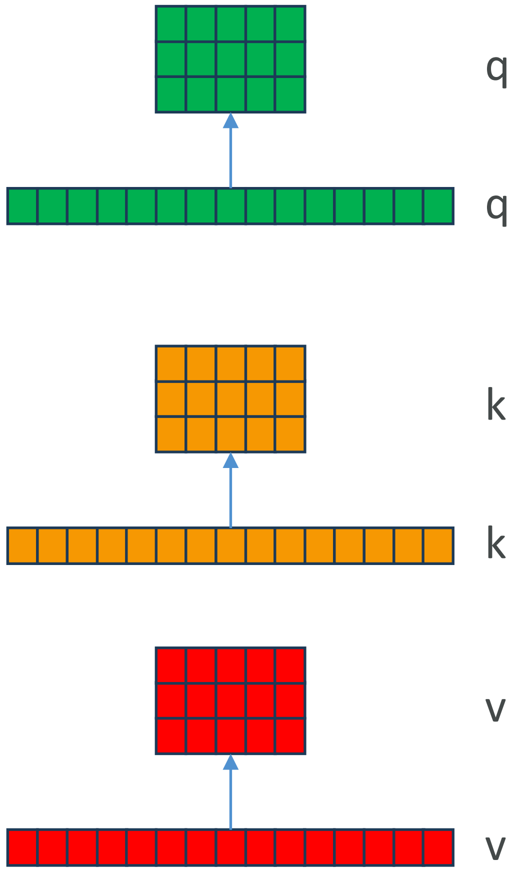
Self-Attention (in more depth)

- Repeat entire process for each token (in parallel)
- Now we have our updated embeddings for each token!
- These weight each token embedding as it’s passed into the feed-forward neural network.



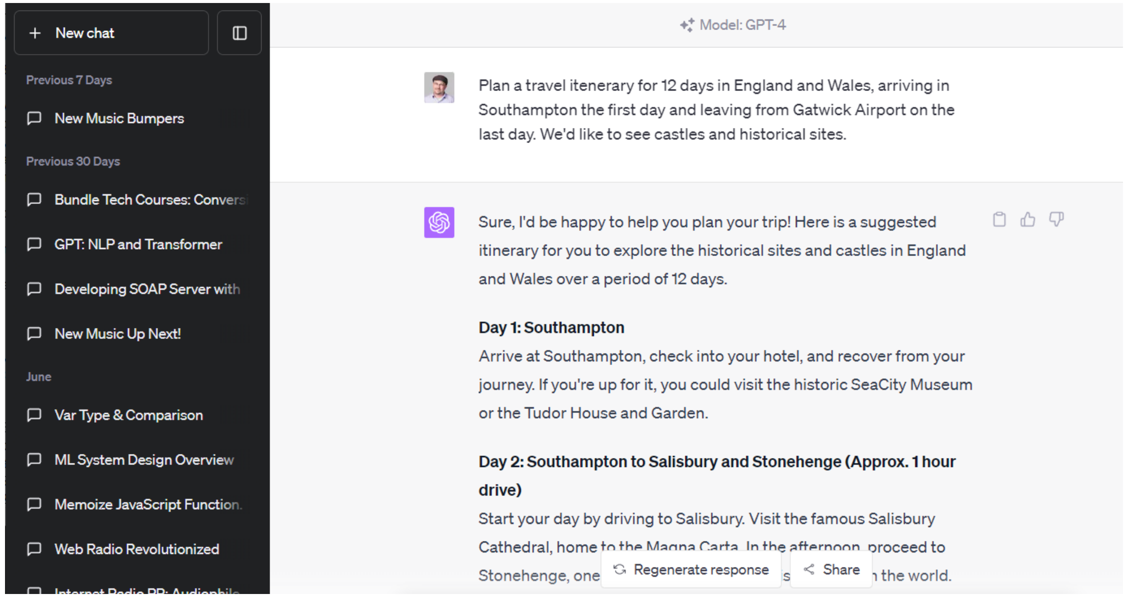
Multi-Headed Self-Attention

- The q, k, and v vectors are reshaped into matrices
- Then each row of the matrix can be processed in parallel
- The number of rows are the number of "heads"



Applications of Transformers

- Chat!
- Question answering
- Text classification
  - i.e., sentiment analysis
- Named entity recognition
- Summarization
- Translation
- Code generation
- Text generation
  - i.e., automated customer service

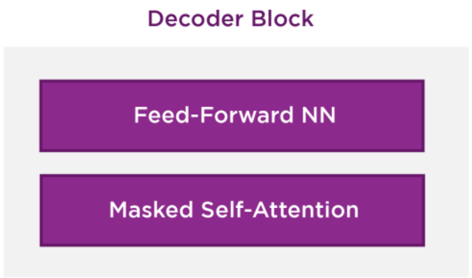


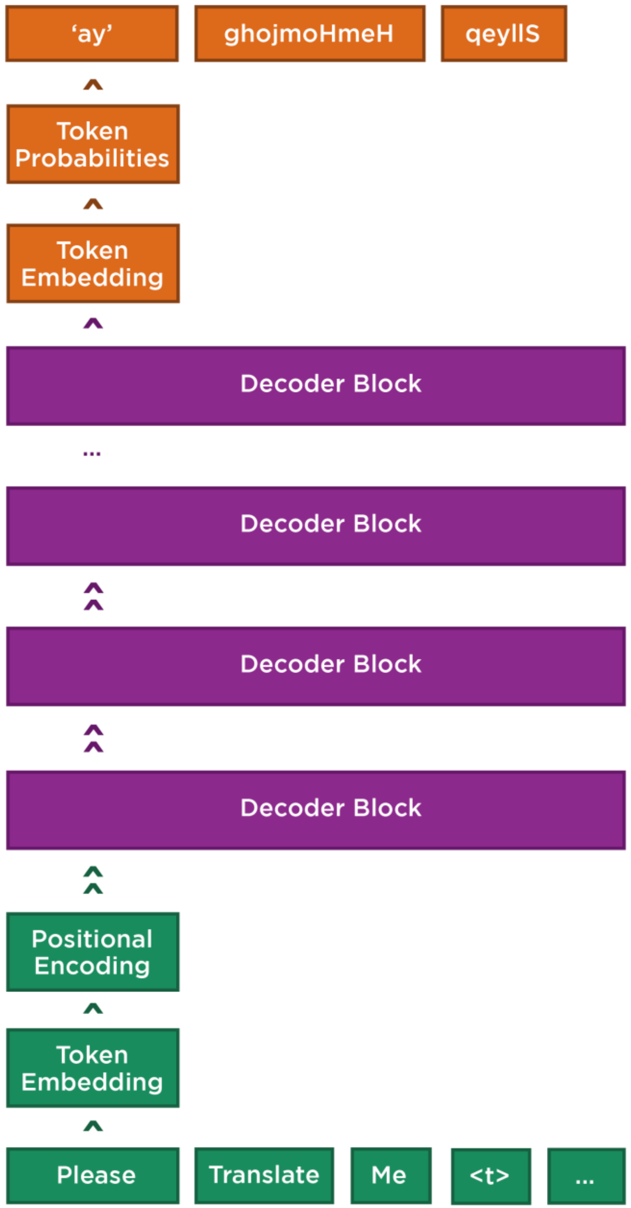
From Transformers to GPT

Generative Pre-Trained Transformers

GPT

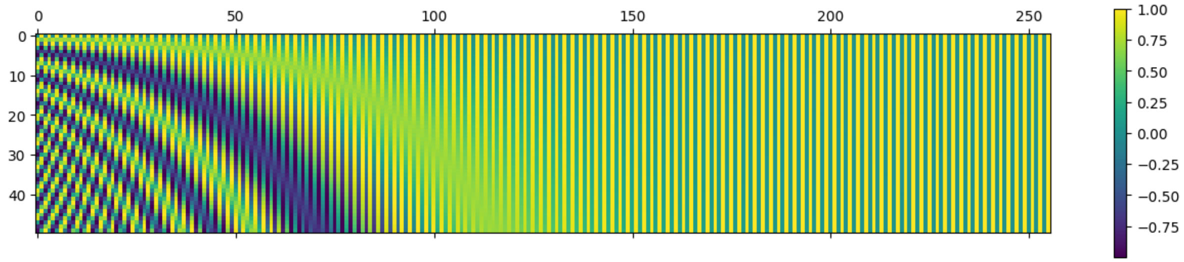
- Generative Pre-Trained Transformer (GPT-2 in this example)
- Decoder-only – stacks of decoder blocks
  - Each consisting of a masked self-attention layer, and a feed-forward neural network
  - As an aside, BERT consists only of encoders. T5 is an example of a model that uses both encoders and decoders.
- No concept of input, all it does is generate the next token over and over
  - Using attention to maintain relationships to previous words / tokens
  - You “prompt” it with the tokens of your question or whatever
  - It then keeps on generating given the previous tokens
- Getting rid of the idea of inputs and outputs is what allows us to train it on unlabeled piles of text
  - It’s “learning a language” rather than optimizing for some specific task
- Hundreds of billions of parameters





GPT: Input processing

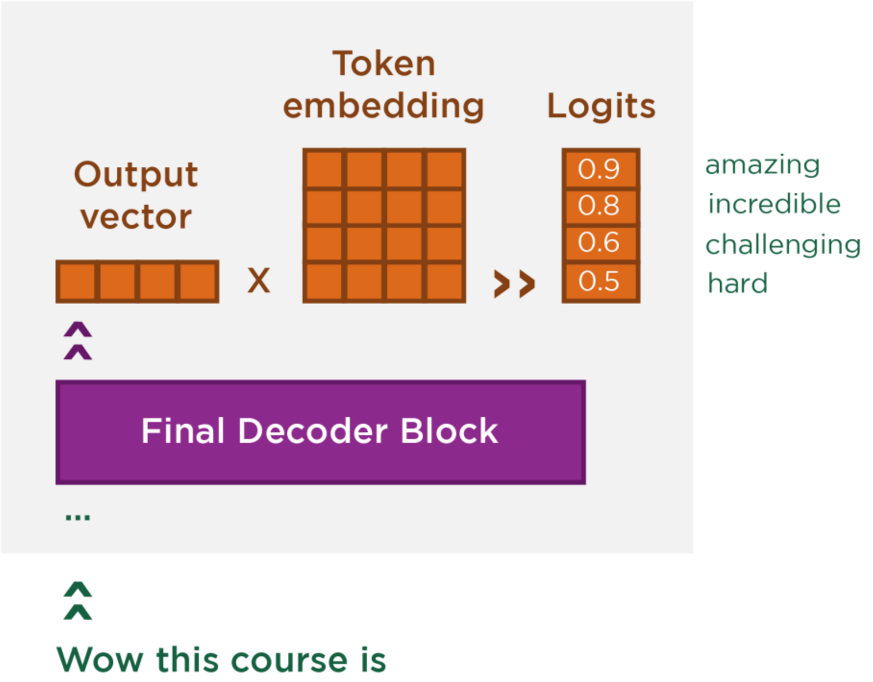
- Tokenization, token encoding
- Token embedding
  - Captures semantic relationships between tokens, token similarities
- Positional encoding
  - Captures the position of the token in the input relative to other nearby tokens
  - Uses an interleaved sinusoidal function so it works on any length



DataCumulus
Sundog™ Education

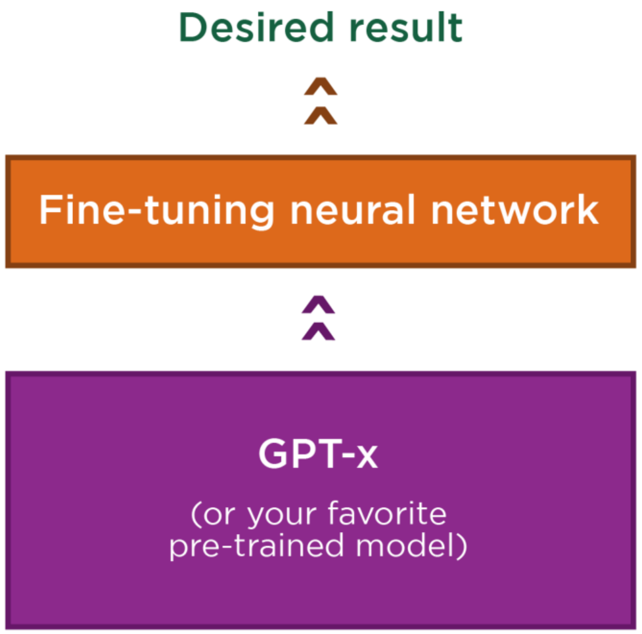
GPT: Output processing

- The stack of decoders outputs a vector at the end
- Multiply this with the token embeddings
- This gives you probabilities (logits) of each token being the right next token (word) in the sequence
- You can randomize things a bit here ("temperature") instead of always picking the highest probability



Transfer Learning (Fine Tuning) with Transformers

- Add additional training data through the whole thing
- Freeze specific layers, re-train others
  - Train a new tokenizer to learn a new language
- Add a layer on top of the pre-trained model
  - Just a few may be all that's needed!
  - Provide examples of prompts and desired completions
    - "How's the weather?" -> "What's it to you, bucko?"
  - Adapt it to classification or other tasks
    - "Wow, I love this course!" -> "Positive"



Generative AI in AWS

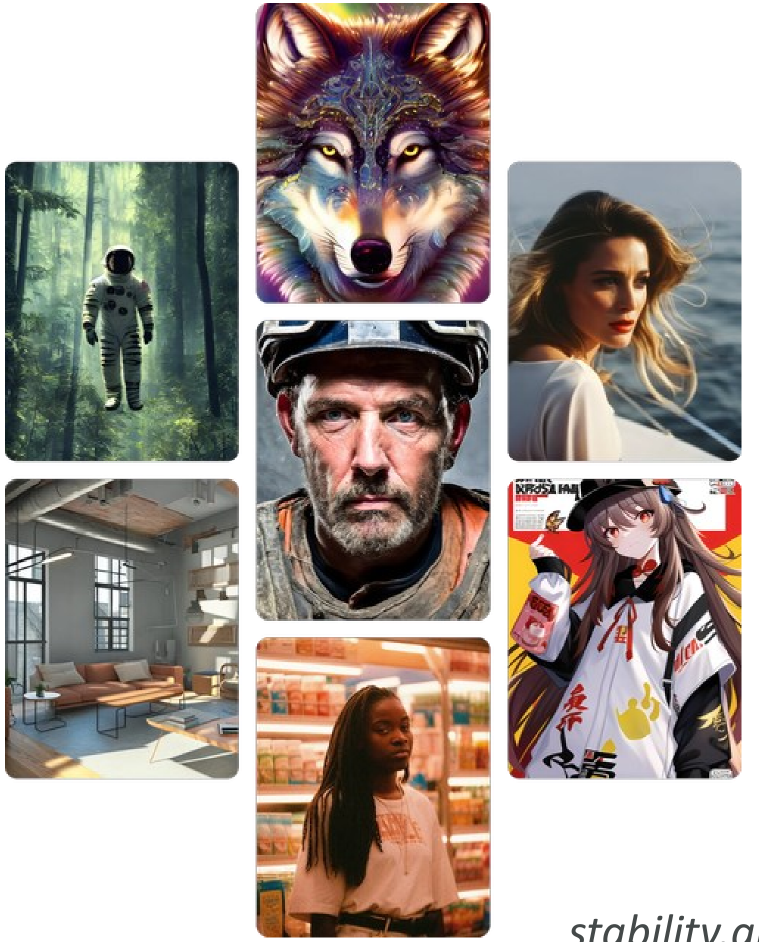
Foundation Models

- The giant, pre-trained transformer models we are fine tuning for specific tasks, or applying to new applications
- GPT-n (OpenAI, Microsoft)
- BERT (Google)
- DALL-E (OpenAI, Microsoft)
- LLaMa (Meta)
- Segment Anything (Meta)
- Where's Amazon?

DataCumulus Sundog Education

AWS Foundation Models

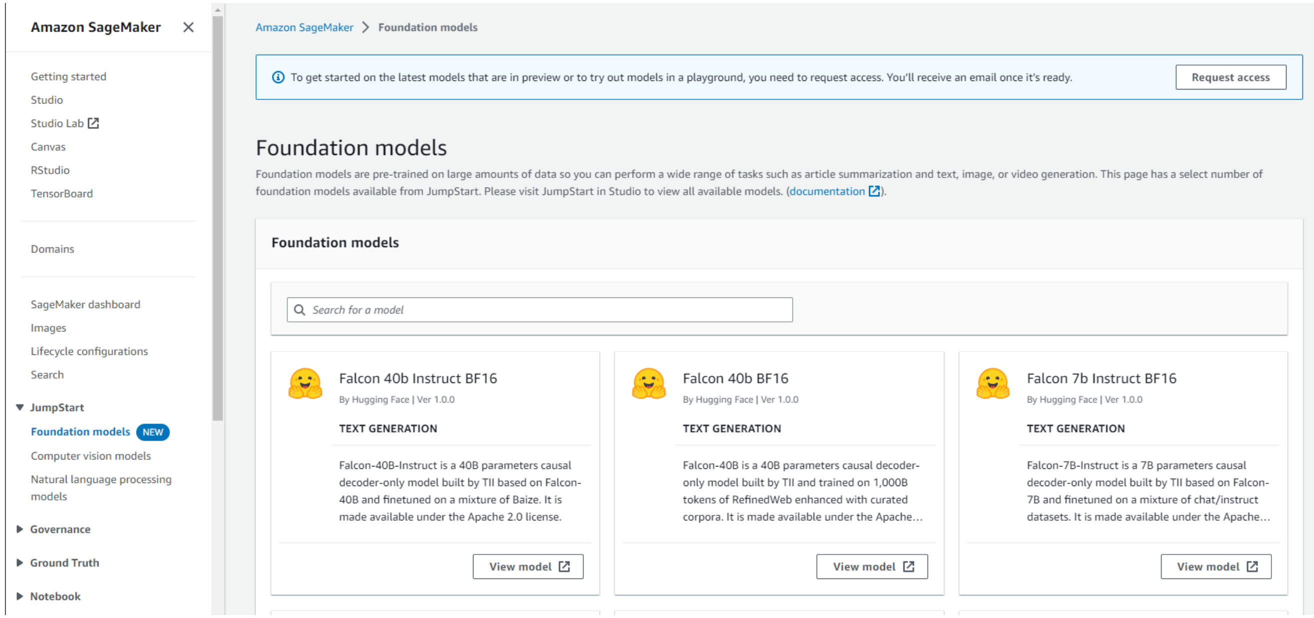
- Jurassic-2 (AI21labs)
  - Multilingual LLMs for text generation
  - Spanish, French, German, Portuguese, Italian, Dutch
- Claude (Anthropic)
  - LLM’s for conversations
  - Question answering
  - Workflow automation
- Stable Diffusion (stability.ai)
  - Image, art, logo, design generation
- Amazon Titan
  - Text summarization
  - Text generation
  - Q&A
  - Embeddings
    - Personalization
    - Search



stability.ai

Amazon SageMaker Jumpstart with Generative AI

- SageMaker Studio has a “JumpStart” feature
- Lets you quickly open up a notebook with a given model loaded up and ready to go
- Current foundation models
  - Hugging Face models (text generation)
    - Falcon, Flan, BloomZ, GPT-J
  - Stabile Diffusion (image generation)
  - Amazon Alexa (encoder/decoder multilingual LLM)



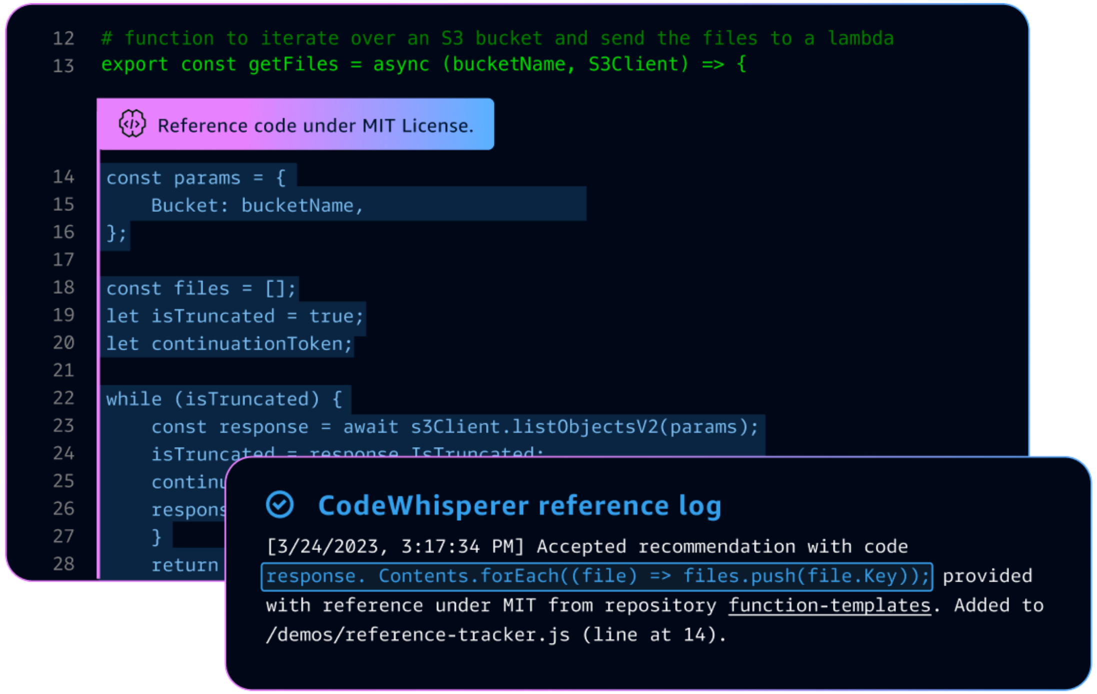
Amazon Bedrock

- An API for Foundation Models
- Serverless
- Fine-tuning API
  - Provide labeled examples in S3
  - As few as 20
  - Your data is only used in your copy of the FM
  - Your data is encrypted and does not leave your VPC
- Integrates with SageMaker

DataCumulus Sundog™ Education

Amazon CodeWhisperer

- An "AI coding companion"
  - Java, JavaScript, Python, TypeScript, C#
- Real-time code suggestions
  - Write a comment of what you want
  - It suggests blocks of code into your IDE
  - Based on LLM's trained on billions of lines of code
    - Amazon's code and open source code
- Security scans
  - Analyzes code for vulnerabilities
  - Java, JavaScript, Python
- Reference tracker
  - Flags suggestions that are similar to open source code
  - Provides annotations for proper attribution

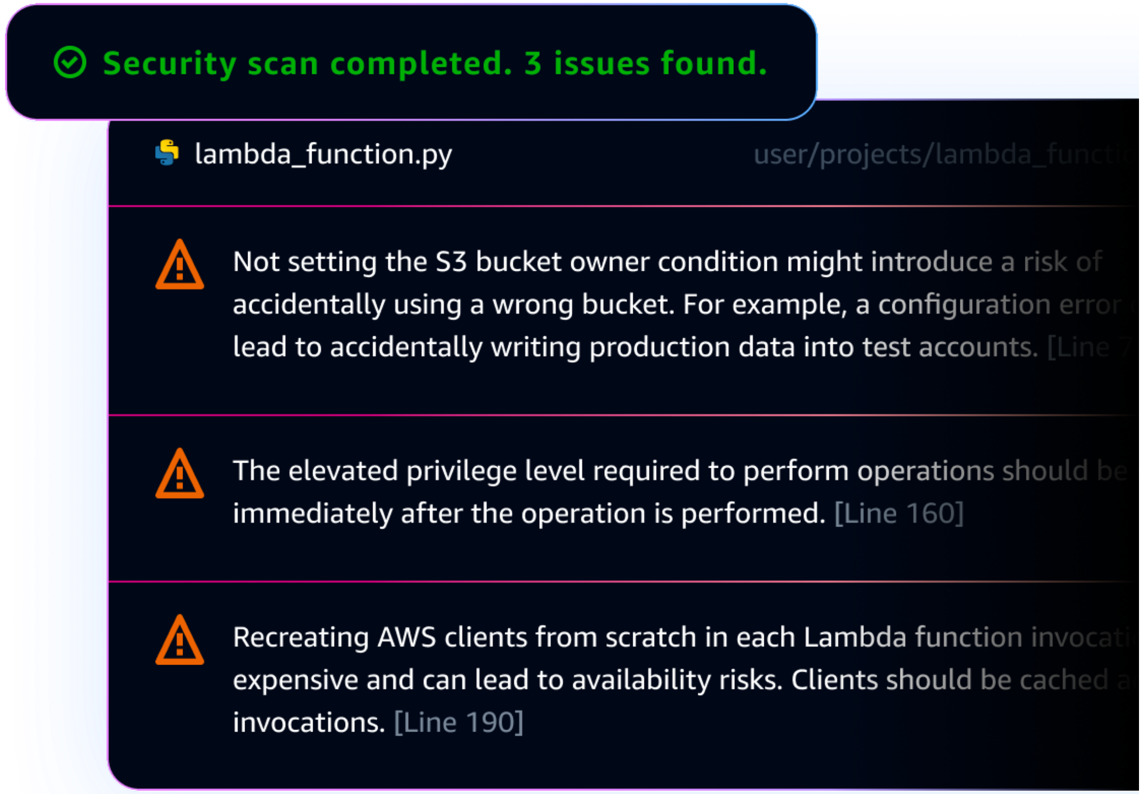


Amazon CodeWhisperer

- Bias avoidance
  - Filters out code suggestions that might be biased or unfair
- AWS service integration
  - Can suggest code for interfacing with AWS API's
    - EC2
    - Lambda
    - S3
- Security
  - All content transmitted with TLS
  - Encrypted in transit
  - Encrypted at rest
  - However – Amazon is allowed to mine your data for individual plans

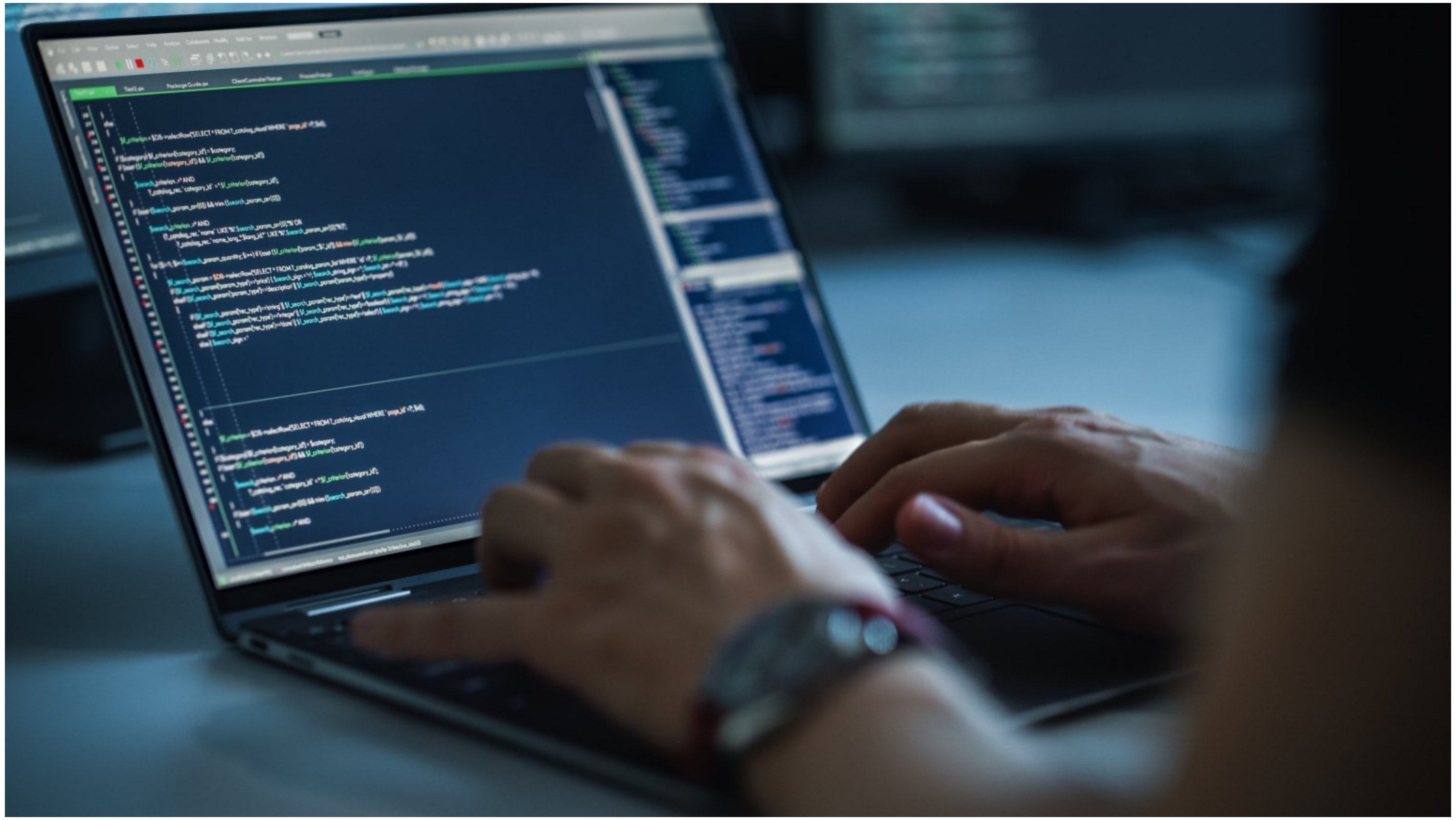
Amazon CodeWhisperer Pricing

- Individual Tier
  - Free to use
  - Up to 50 security scans / month
- Professional Tier
  - Manage which developers in an organization has access
  - Organizational policy management
    - i.e., can you use open source code?
  - Authenticated with IAM Identity Center
  - Up to 500/user/month security scans
  - $19 / user / month



Amazon CodeWhisperer: Supported IDE's

- Visual Studio Code
- JetBrains
  - IntelliJ
  - PyCharm
  - Clion
  - GoLand
  - WebStorm
  - PhpStorm
  - RubyMine
  - DataGrip
- JupyterLab
- SageMaker Studio
- AWS Lambda console
- AWS Cloud9



DataCumulus Sundog Education

Exam Preparation

More Prep Resources

DataCumulus

Sundog™
Education

Practice Exams (including Amazon's)

AWS Certified Machine Learning Specialty Practice Exam

65 questions | 3 hours | 75% correct required to pass

Simulate the real AWS Certified Machine Learning Specialty exam experience. You'll encounter the same number of questions, the same breakdown of topics, and a similar format to the real thing. Your ability to pass or fail this exam should be a good indicator of your readiness for the real thing.

When taking the real exam, some questions might sound similar to some of these practice questions, since they cover the same material. But they are not the same. If you see a question on the real exam that looks familiar, don't automatically select the same answer that you remember from this practice exam. Odds are the question is actually a little different, and has a different answer.

The actual exam is timed and you will have 3 hours, with no breaks, to complete it. You should find that 3 hours is more than sufficient, so take your time and read each question and possible answer carefully.

Instructions:

- You can pause the test at any time and resume later.
- You can retake the test as many times as you would like.
- The progress bar at the top of the screen will show your progress as well as the time remaining in the

Skip test Begin test >

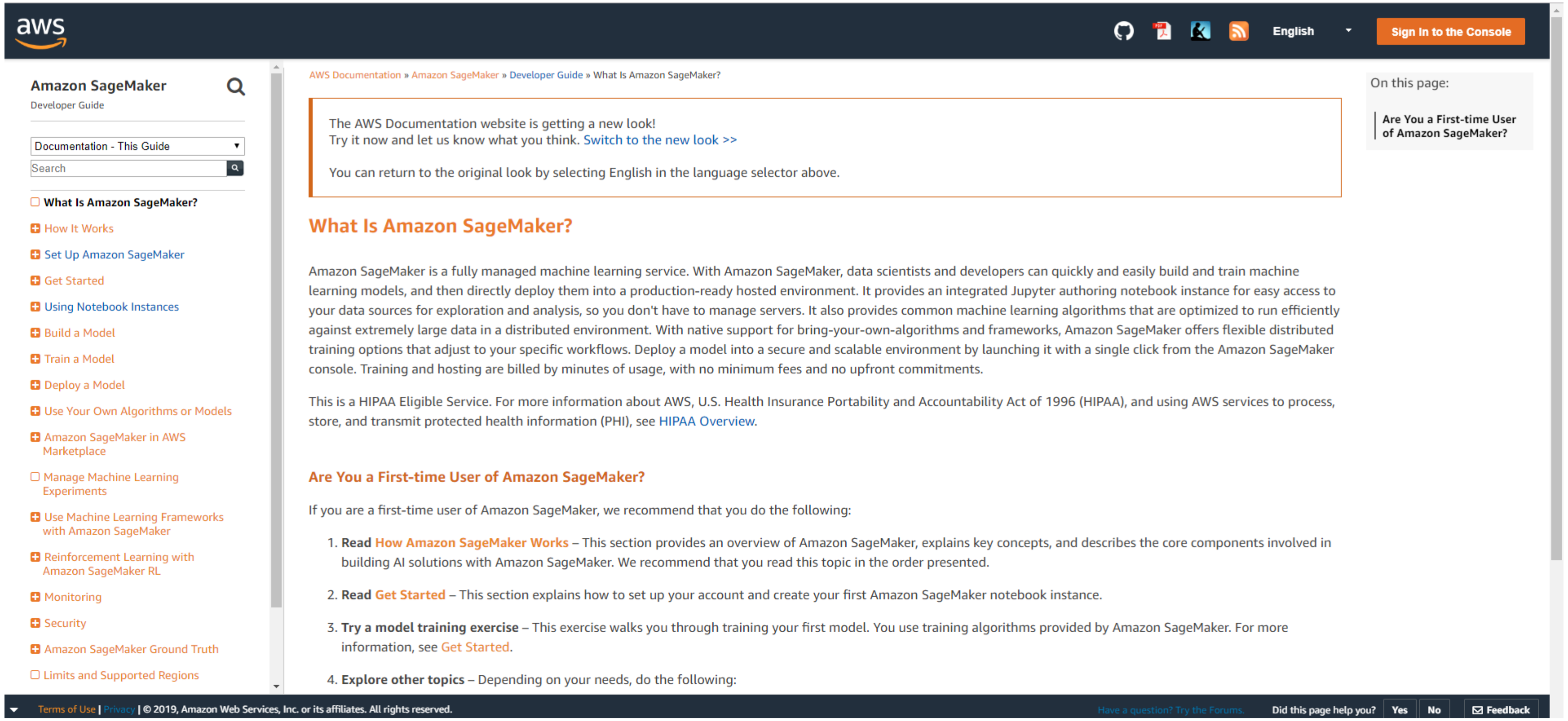
Bookmarks Announcements

About this course

Test your readiness for the newest, toughest AWS certification (MLS-C01) with a full-length, realistic practice exam.

SageMaker Developer Guide

https://docs.aws.amazon.com/sagemaker/latest/dg/



DataCumulus
Sundog™ Education

Amazon's Exam Guide

https://aws.amazon.com/certification/certified-machine-learning-specialty/

aws training and certification

AWS Certified Machine Learning–Specialty
(MLS-C01) Exam Guide

Introduction

The AWS Certified Machine Learning—Specialty (MLS-C01) exam is intended for individuals who perform a Development or Data Science role. This exam validates an examinee's ability to build, train, tune, and deploy machine learning (ML) models using the AWS Cloud.

It validates an examinee's ability to design, implement, deploy, and maintain ML solutions for given business problems. It will validate the candidate's ability to:

- Select and justify the appropriate ML approach for a given business problem.
- Identify appropriate AWS services to implement ML solutions.
- Design and implement scalable, cost-optimized, reliable, and secure ML solutions.

Recommended AWS Knowledge

The successful candidate likely has one to two years of hands-on experience developing, architecting, or running ML/deep learning workloads on the AWS Cloud, along with:

- The ability to express the intuition behind basic ML algorithms
- Experience performing basic hyperparameter optimization
- Experience with ML and deep learning frameworks
- The ability to follow model-training best practices
- The ability to follow deployment and operational best practices

Exam Preparation

These training courses and materials may be helpful for examination preparation:

AWS Training (aws.amazon.com/training)

- Big Data on AWS: 3-day instructor-led live course
- Deep Learning on AWS: 1-day instructor-led live course

Other Materials

- Machine Learning on AWS web-page
- Amazon Machine Learning Concepts document
- Splitting Your Data document
- Types of Machine Learning Models document
- Data Transformations document
- Containerized Machine Learning on AWS video
- AWS re:Invent Machine Learning talk video

DataCumulus Sundog Education

Amazon's Sample Questions

aws training and certification

Machine Learning - Specialty (MLS-C01)
Sample Exam Questions

1) **A Machine Learning team has several large CSV datasets in Amazon S3. Historically, models built with the Amazon SageMaker Linear Learner algorithm have taken hours to train on similar-sized datasets. The team's leaders need to accelerate the training process.**

   What can a Machine Learning Specialist do to address this concern?

   A. Use Amazon SageMaker Pipe mode.
   B. Use Amazon Machine Learning to train the models.
   C. Use Amazon Kinesis to stream the data to Amazon SageMaker.
   D. Use AWS Glue to transform the CSV dataset to the JSON format.

2) **A term frequency–inverse document frequency (tf–idf) matrix using both unigrams and bigrams is built from a text corpus consisting of the following two sentences:**

   1. Please call the number below.
   2. Please do not call us.

   What are the dimensions of the tf–idf matrix?

   A. (2, 16)
   B. (2, 8)
   C. (2, 10)
   D. (8, 10)

3) **A company is setting up a system to manage all of the datasets it stores in Amazon S3. The company would like to automate running transformation jobs on the data and maintaining a catalog of the metadata concerning the datasets. The solution should require the least amount of setup and maintenance.**

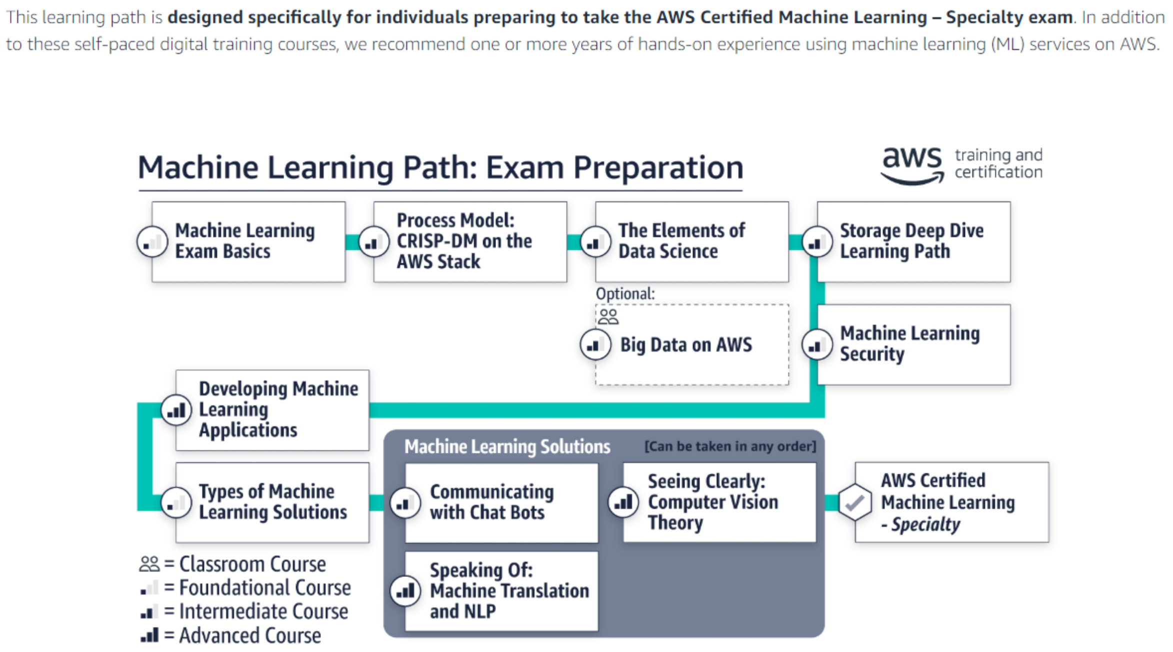
   Which solution will allow the company to achieve its goals?

   A. Create an Amazon EMR cluster with Apache Hive installed. Then, create a Hive metastore and a script to run transformation jobs on a schedule.
   B. Create an AWS Glue crawler to populate the AWS Glue Data Catalog. Then, author an AWS Glue ETL job, and set up a schedule for data transformation jobs.
   C. Create an Amazon EMR cluster with Apache Spark installed. Then, create an Apache Hive metastore and a script to run transformation jobs on a schedule.
   D. Create an AWS Data Pipeline that transforms the data. Then, create an Apache Hive metastore and a

DataCumulus Sundog™ Education

Amazon’s learning path

- https://aws.amazon.com/training/learning-paths/machine-learning/exam-preparation/



Consider taking the Big Data / Data Analytics Exam first

What to Expect

Sitting for the exam

- 3 hours! (170 minutes technically)
  - But you'll probably only need a little over 2.
  - 65 questions
  - TAKE THE TIME TO FULLY UNDERSTAND THEM
- Use the flag capability
  - You'll have time to go back and double check things.
- You can't bring anything into the room
  - You'll be given note paper or dry-erase board, earplugs, and that's it.
- Schedule smartly
  - As soon as possible after you've mastered the practice exams
  - Choose a time of day that matches your highest energy level – you need stamina
  - Remember different testing providers may have different availability

Prepare

- Use the bathroom before you head in
- Eat something (but not right beforehand)
- Get a good night's sleep
- Review things you need to memorize before going in
  - Recall, Precision, F1
  - Regularization techniques
- Be alert
  - Whatever works for you…
- Make sure you're ready
  - You've got $300 on the line
  - Take the practice exams! Repeatedly!

Strategies

- Don't rush
  - For each question, take the time to understand:
    - What you are optimizing for
    - Requirements
    - The system as a whole
- Your pace should be about 2 – 2 ½ minutes per question
  - If you're taking longer, make your best guess, flag it and come back to it later
- Use the process of elimination
  - Even if you don't know the correct answer, you can often eliminate several.
- Keep calm
  - You're not going to get 100%. You don't have to.

AWS Certification Paths – Architecture

Architecture

Solutions Architect

Design, develop, and manage cloud infrastructure and assets, work with DevOps to migrate applications to the cloud

Dive Deep

Architecture

Application Architect

Design significant aspects of application architecture including user interface, middleware, and infrastructure, and ensure enterprise-wide scalable, reliable, and manageable systems

Dive Deep

https://d1.awsstatic.com/training-and-certification/docs/AWS_certification_paths.pdf

AWS Certification Paths – Operations

Operations

Systems Administrator

Install, upgrade, and maintain computer components and software, and integrate automation processes

Dive Deep

Operations

Cloud Engineer

Implement and operate an organization's networked computing infrastructure and Implement security systems to maintain data safety

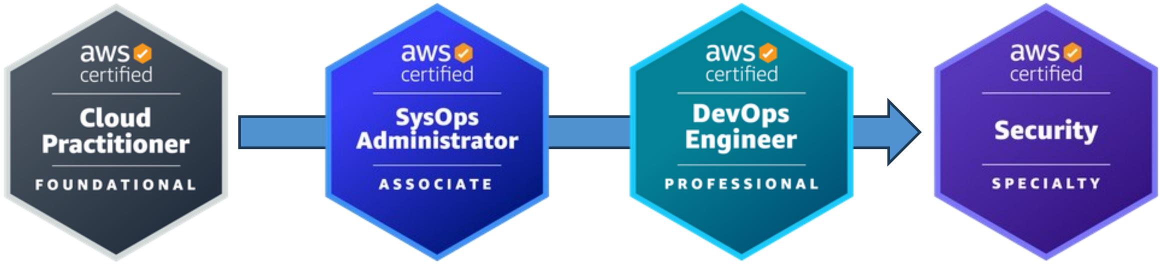
Dive Deep

AWS Certification Paths – DevOps

DevOps
Test Engineer
Embed testing and quality best practices for software development from design to release, throughout the product life cycle

DevOps
Cloud DevOps Engineer
Design, deployment, and operations of large-scale global hybrid cloud computing environment, advocating for end-to-end automated CI/CD DevOps pipelines

DevOps
DevSecOps Engineer
Accelerate enterprise cloud adoption while enabling rapid and stable delivery of capabilities using CI/CD principles, methodologies, and technologies



DataCumulus
Sundog™ Education

AWS Certification Paths – Security

Security
Cloud Security Engineer
Design computer security architecture and develop detailed cyber security designs. Develop, execute, and track performance of security measures to protect information

Dive Deep

Security
Cloud Security Architect
Design and implement enterprise cloud solutions applying governance to identify, communicate, and minimize business and technical risks

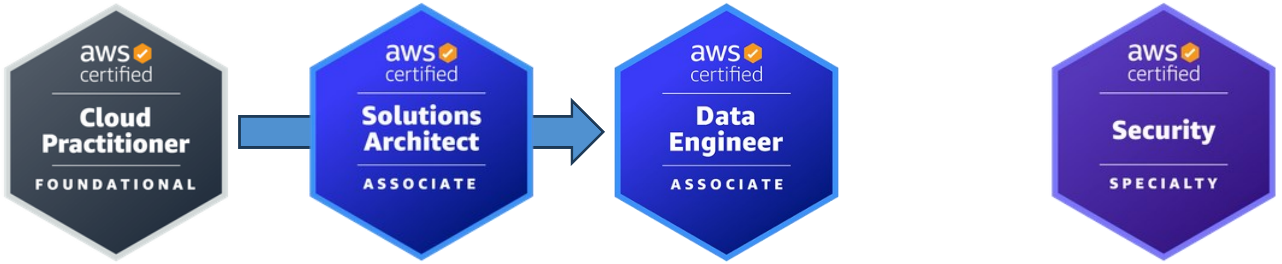
Dive Deep

AWS Certification Paths – Data Analytics & Development

Data Analytics

Cloud Data Engineer

Automate collection and processing
of structured/semi-structured data
and monitor data pipeline performance



Dive Deep

Development

Software Development Engineer

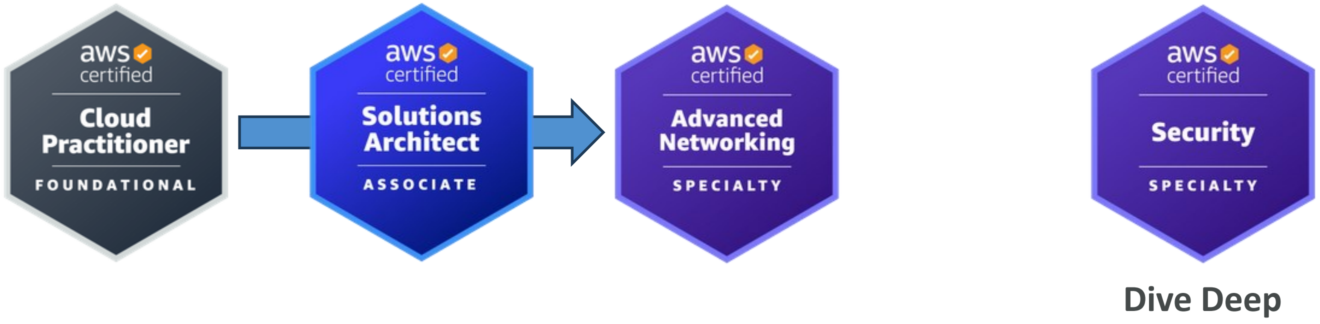
Develop, construct, and maintain
software across platforms and devices

AWS Certification Paths – Networking & AI/ML

Networking

Network Engineer

Design and implement computer and information networks, such as local area networks (LAN), wide area networks (WAN), intranets, extranets, etc.



AI/ML

Machine Learning Engineer

Research, build, and design artificial intelligence (AI) systems to automate predictive models, and design machine learning systems, models, and schemes

Congratulations