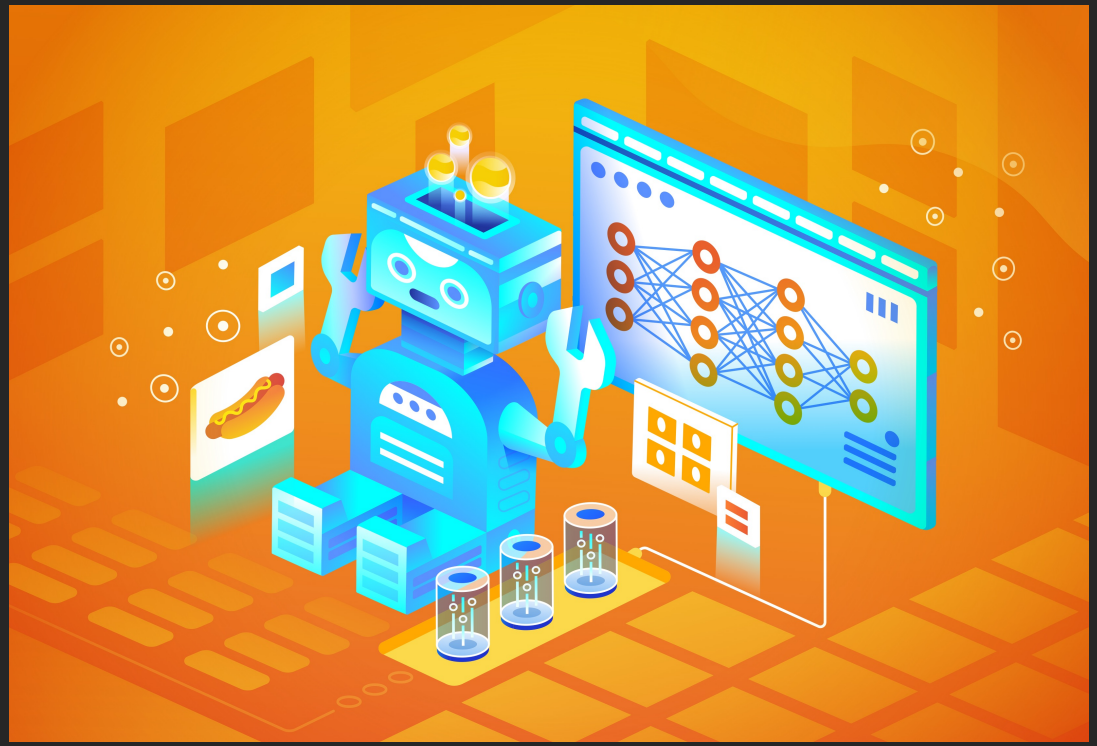


# AWS Certified Machine Learning Specialty

## Course papers

# AWS Certified Machine Learning - Specialty

Hello – Mike here. As you probably know, each course here at Linux Academy comes with a set of interactive diagrams. Well, this course is a little ...

This course, and its content, is different from many other courses – this reflects AWS MLS-C01, which is a very different kind of exam. While this course will still have plenty of hands-on lessons and labs, there is a whole load of ML/DL-specific terminology and concepts that need to be covered too.

I am including a set of "slides" in this PDF, which you will recognize from the lessons.

Also, as I like to think this course is just a little bit special, I have created a channel on the Linux Academy Community Slack. It would be great to catch up with you in there, and chat about how you think the course is going.

If you're not already a member of the community Slack you can join here: http://slack.linuxacademy.com/

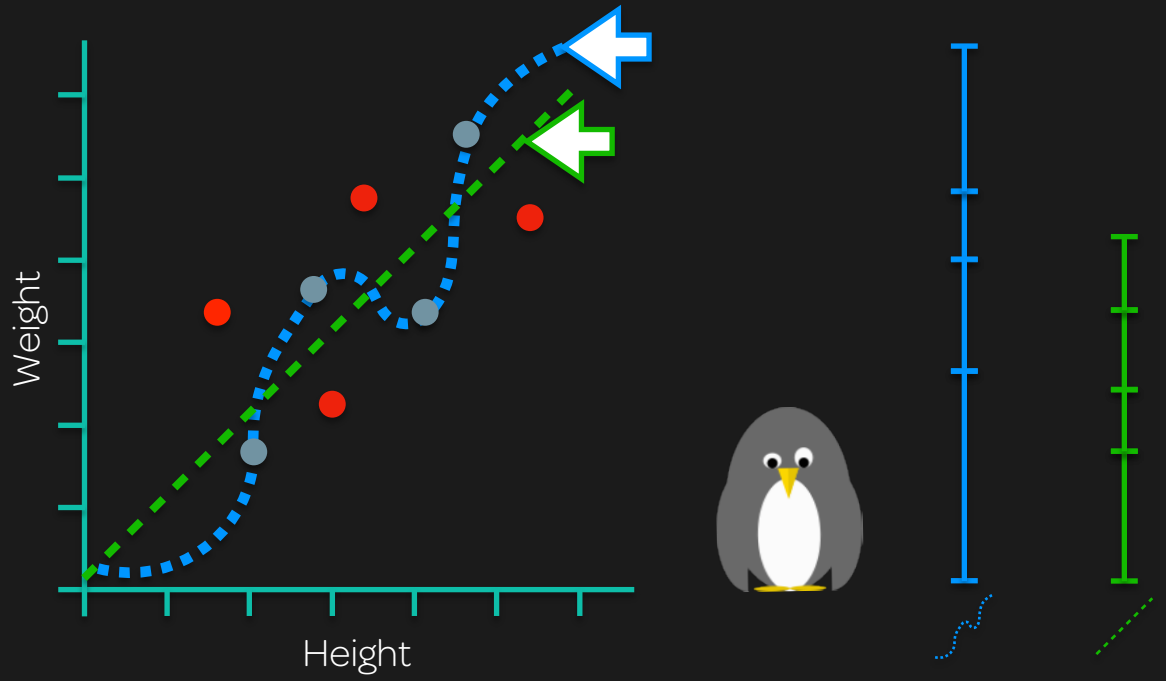
And then come and join in the course discussion here: #aws-mls-c01-2019

I hope you enjoy this course. Lets go...

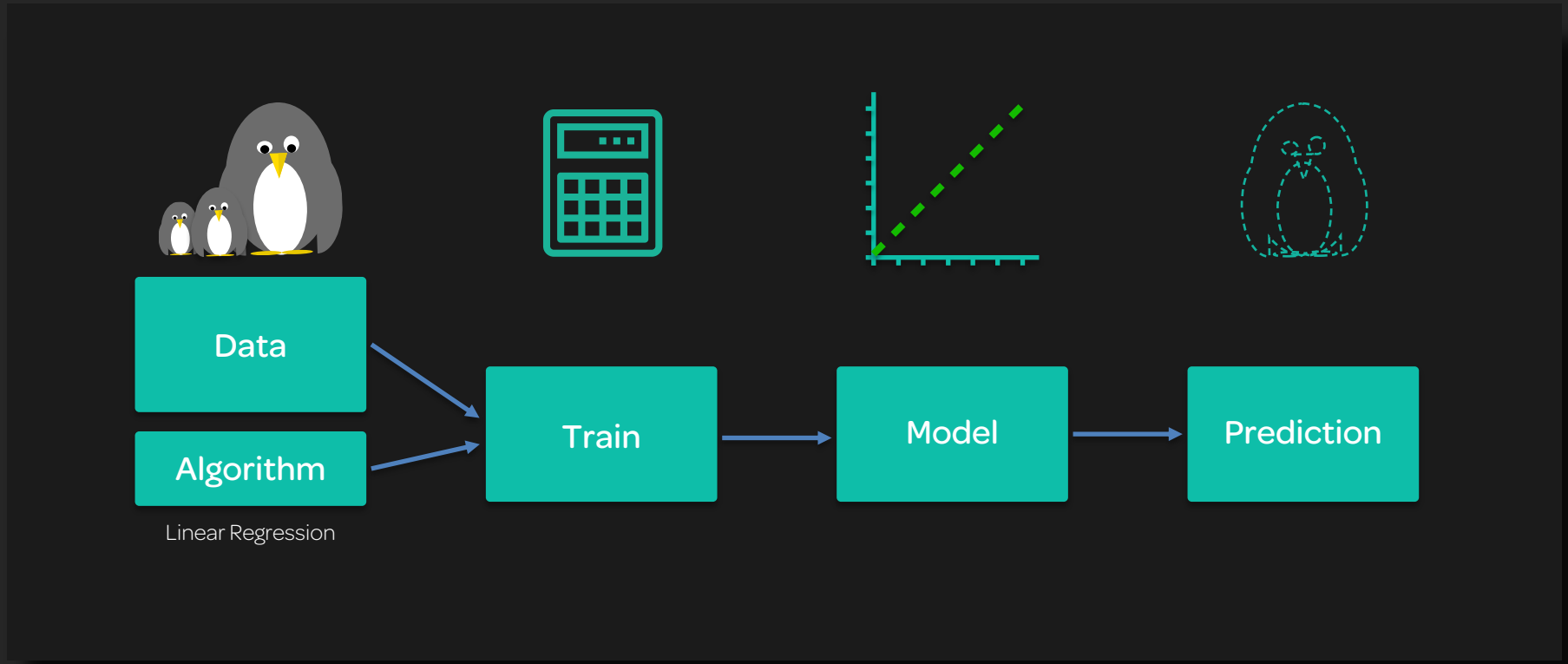

Name: Mike Chambers
Location: Brisbane, Australia
Born: Birmingham, England
Likes: Tea, Sparky the dog – oh, and AWS!

# Machine Learning

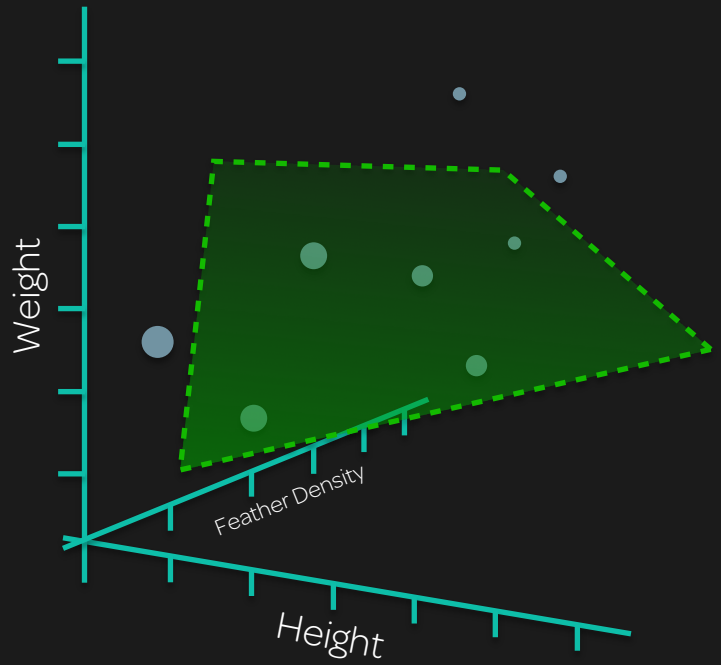
- The blue line is **overfit**, and fails to **generalize**.
- The green line is better at making **predictions**.
- The green line represents a machine learning **model** to predict the weight of penguins.



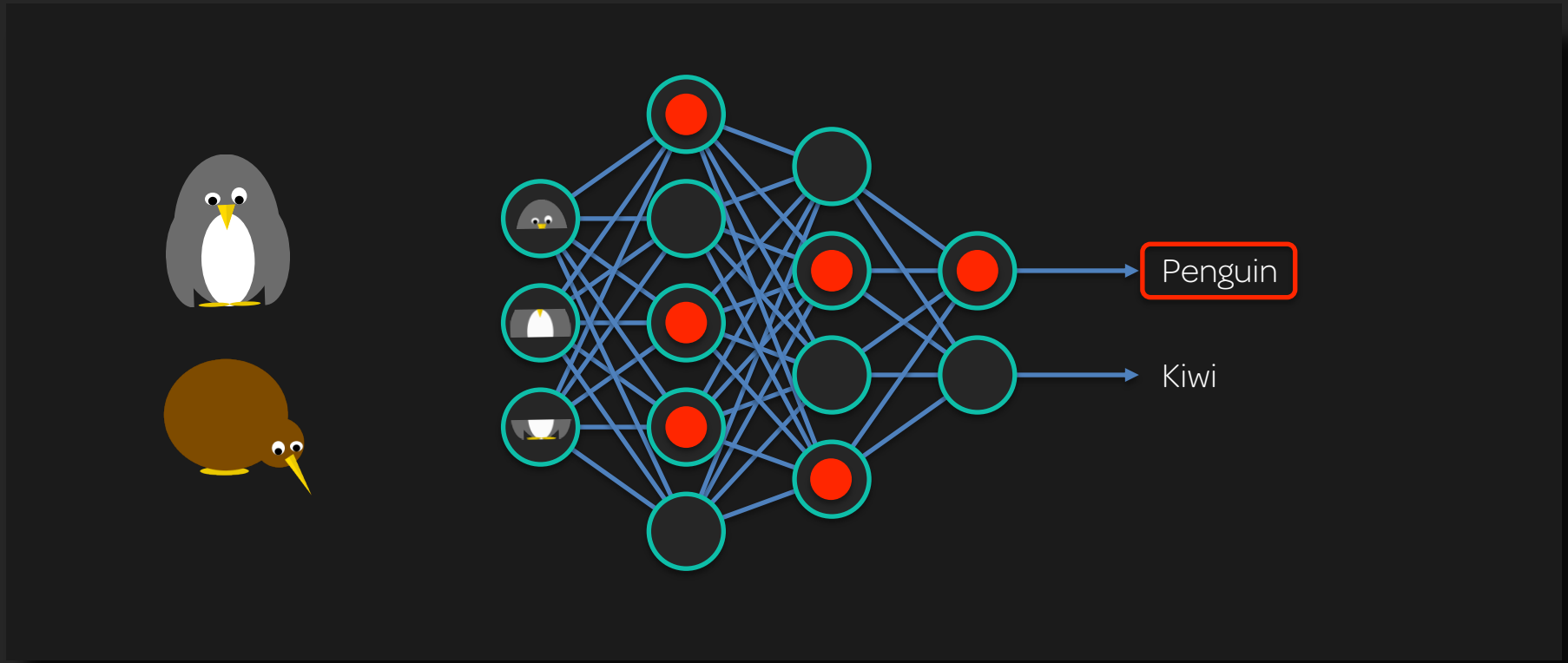
# Machine Learning



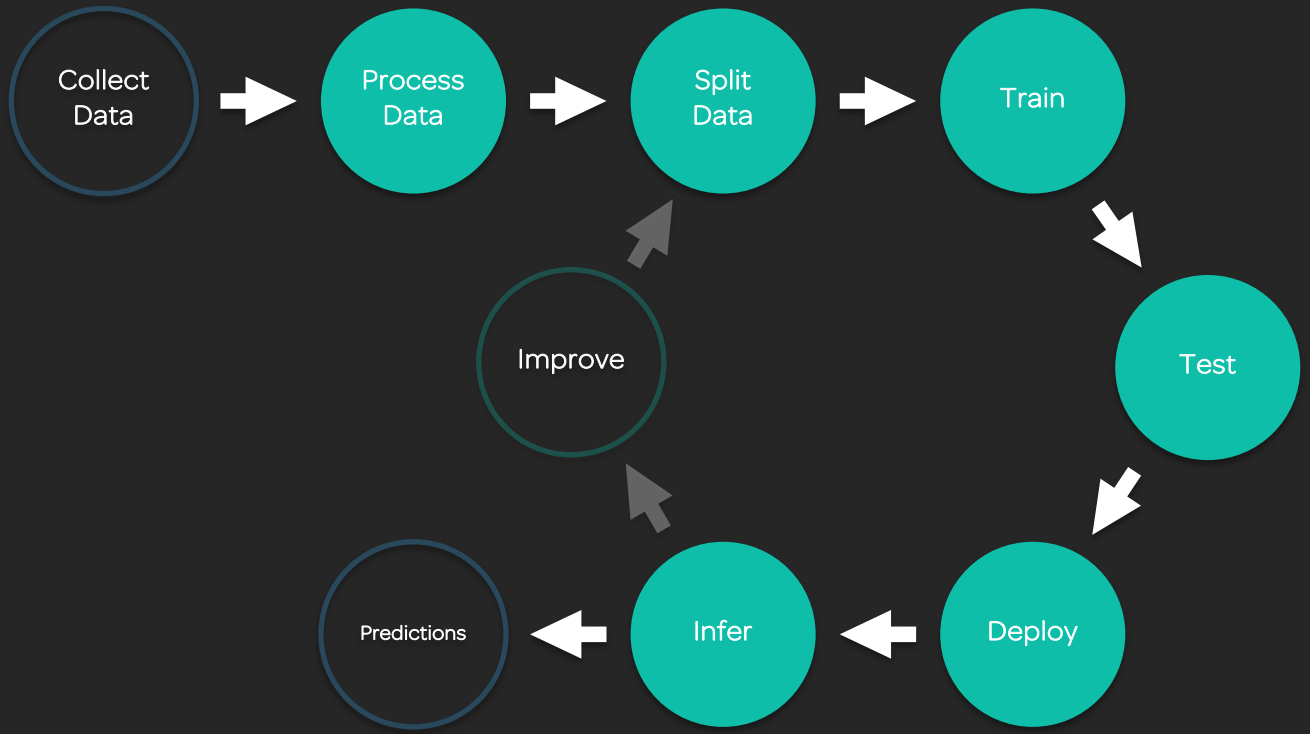
# Machine Learning



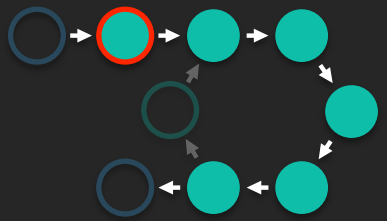

# Deep Learning



# Machine Learning Lifecycle



# Machine Learning Lifecycle

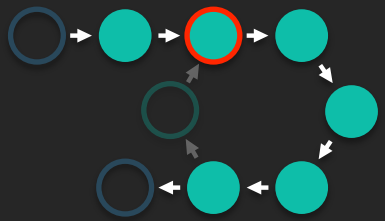
## Process Data

- Features and labels
- Feature engineering
- Feature reduction
- Encoding
- Formatting

Feature Label

| Age | Star Sign | Drinks Tea | Likes Penguins |
|---|---|---|---|
| 42 | 3 | No | Yes |
| 20 | 5 | Yes | No |
| 15 | 2 | Yes | No |
| 37 | 4 | Yes | Yes |

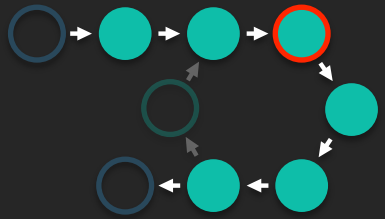
# Machine Learning Lifecycle

## Split Data


- Training (~80%)
- Validation (optional – see cross-validation)
- Testing (~20%)

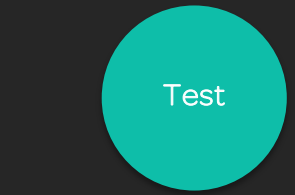
# Machine Learning Lifecycle

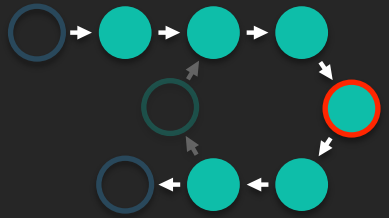



## Train

- Algorithm "sees" and is directly influenced by the training data
- Algorithm uses but is indirectly influenced from validation data
- Algorithm "does not see" testing data during training

# Machine Learning Lifecycle

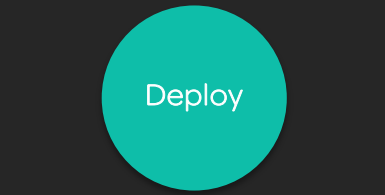


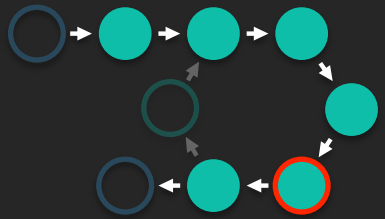



## Test

- Perform inference on testing data to see how well the model fits
- Overfit?

# Machine Learning Lifecycle

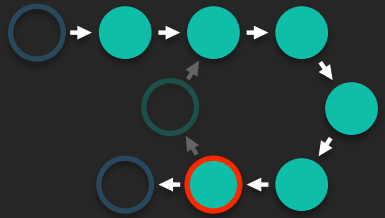


## Deploy

- Host model in execution environment according to the requirements
- Batch
- As a service
- Infrastructure (e.g., load balancing)

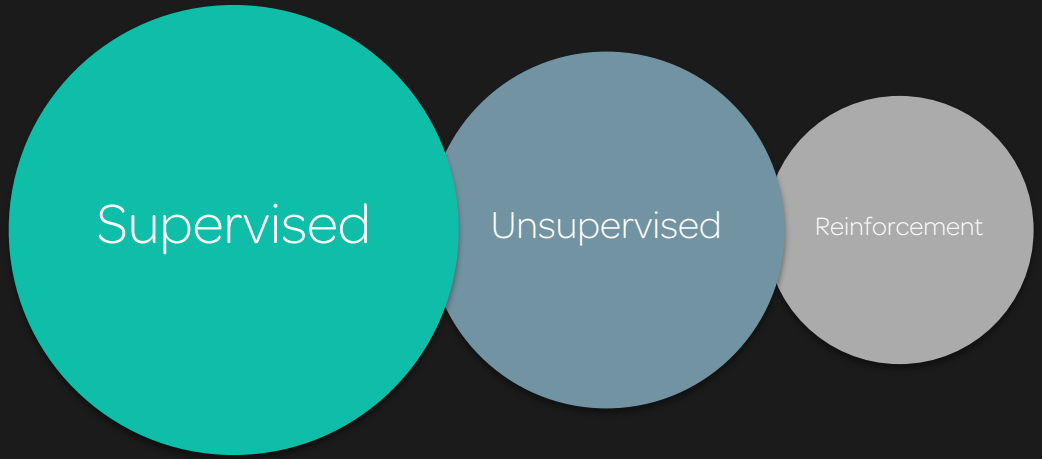
# Machine Learning Lifecycle

## Infer

- Use this model in production to make inference/predictions

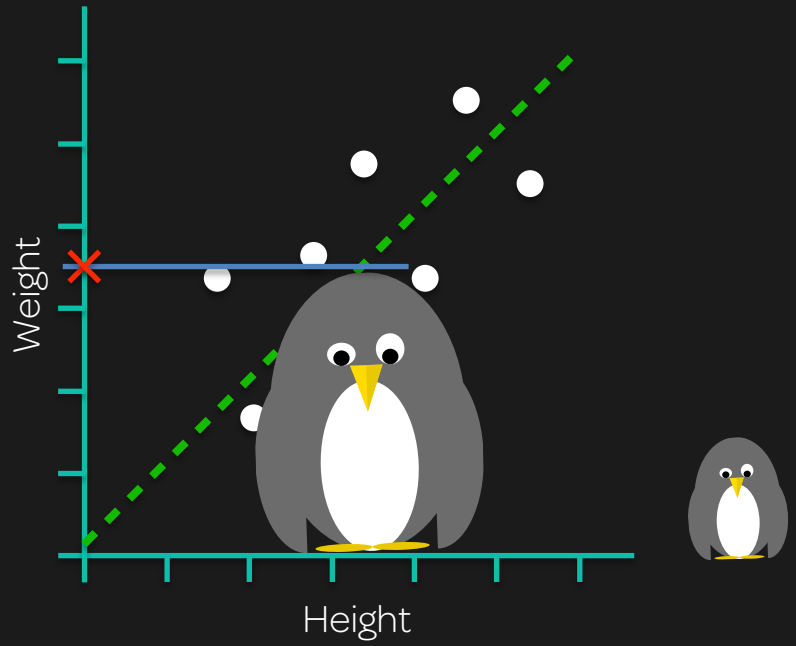
# Supervised, Unsupervised, and Reinforcement Learning



# Supervised, Unsupervised, and Reinforcement Learning

## Supervised Learning

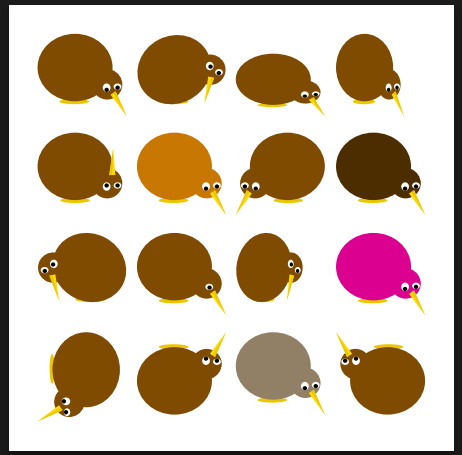
- Numeric data



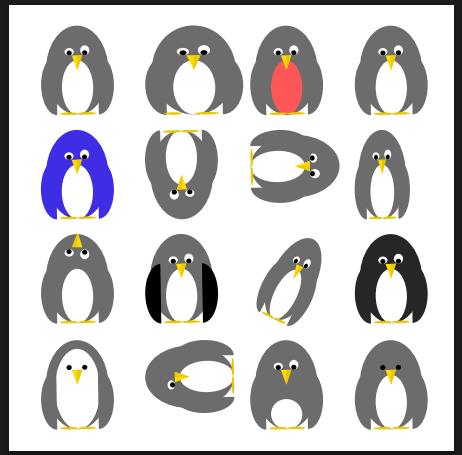
# Supervised, Unsupervised, and Reinforcement Learning

## Supervised Learning

- Classified data



Kiwi

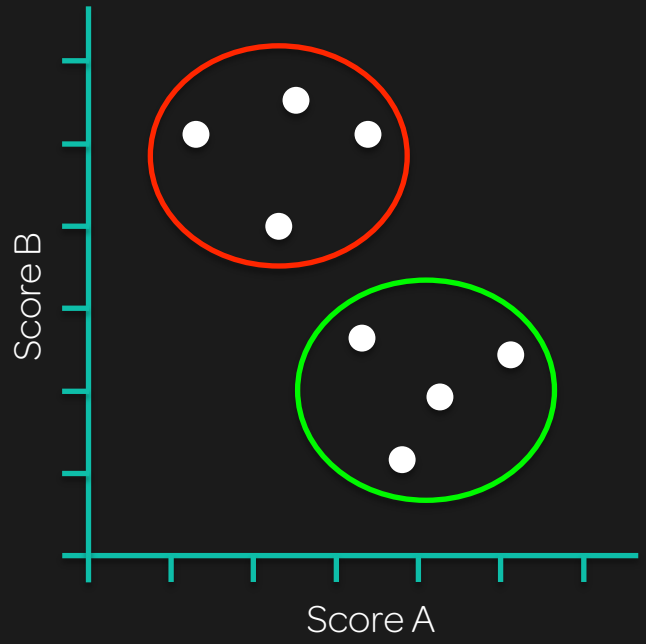


Penguin

This is a penguin.

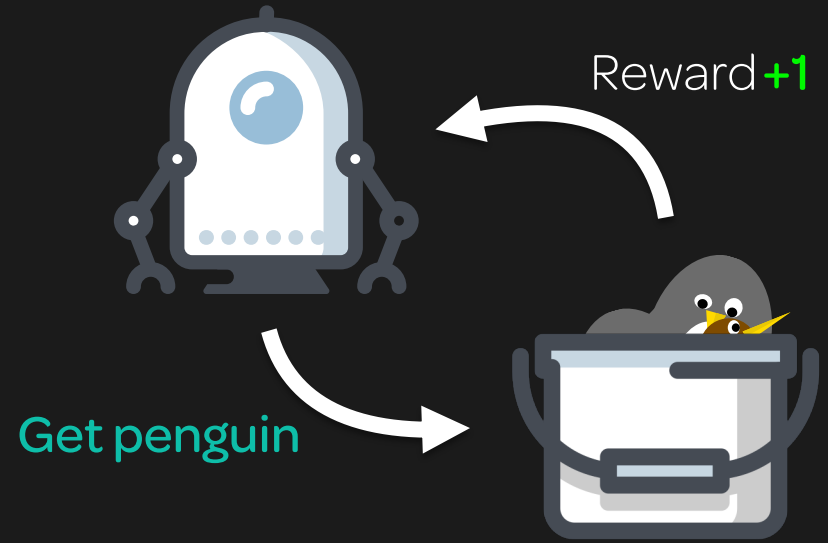
# Supervised, Unsupervised, and Reinforcement Learning

## Unsupervised Learning



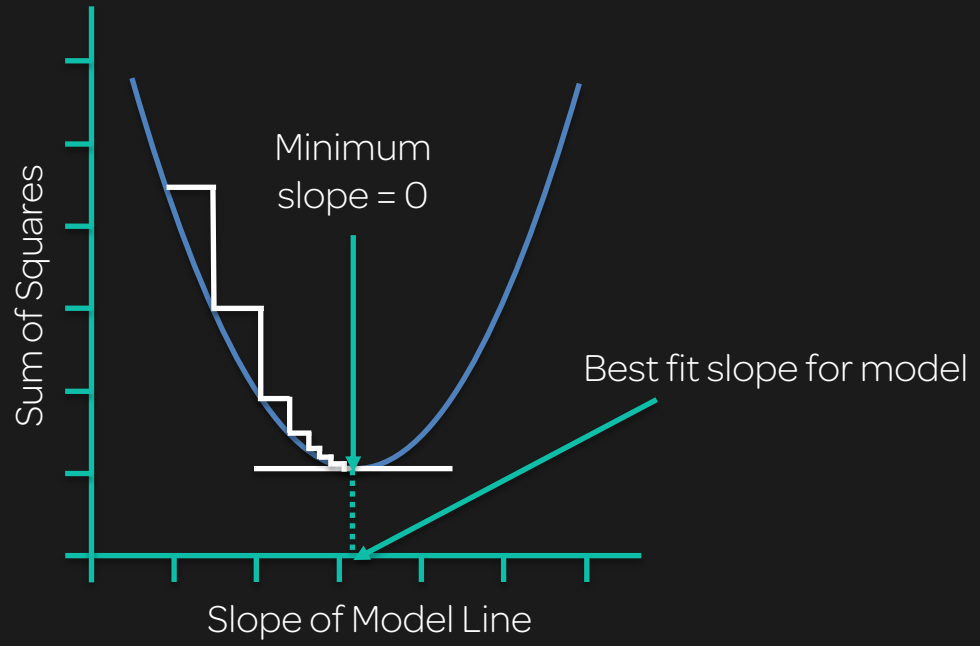
# Supervised, Unsupervised, and Reinforcement Learning

## Reinforcement Learning



# Optimization

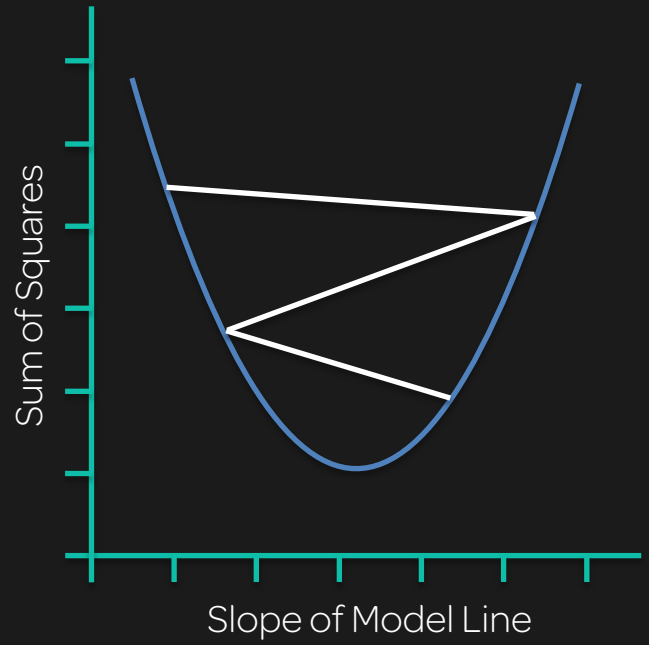
- Step size sets the **learning rate**.



# Optimization

## Gradient Descent

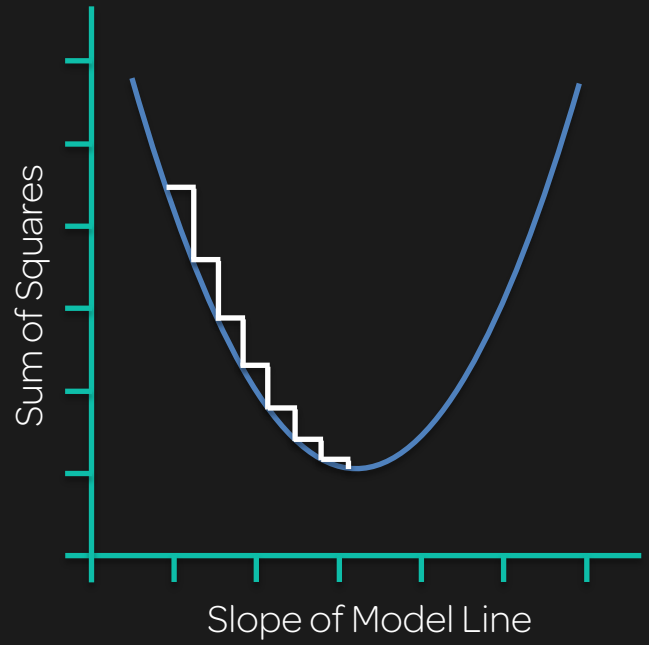
- If the step is **too large**, the minimum slope may be missed
- Less efficient



# Optimization

## Gradient Descent

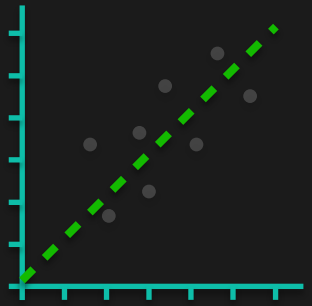
- If the step is **too small**, the process will take longer
- Less efficient



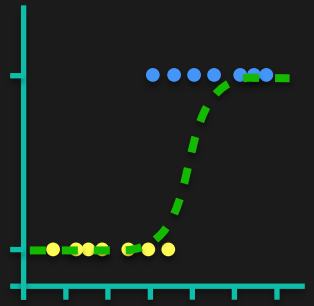
# Optimization

## Gradient Descent

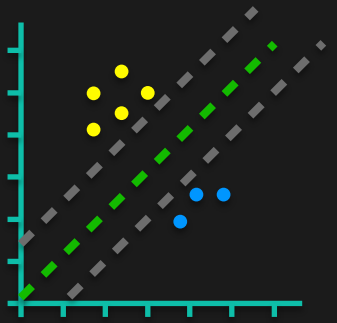
- **Gradient descent** is used to optimize many different types of machine learning algorithms.



Linear Regression



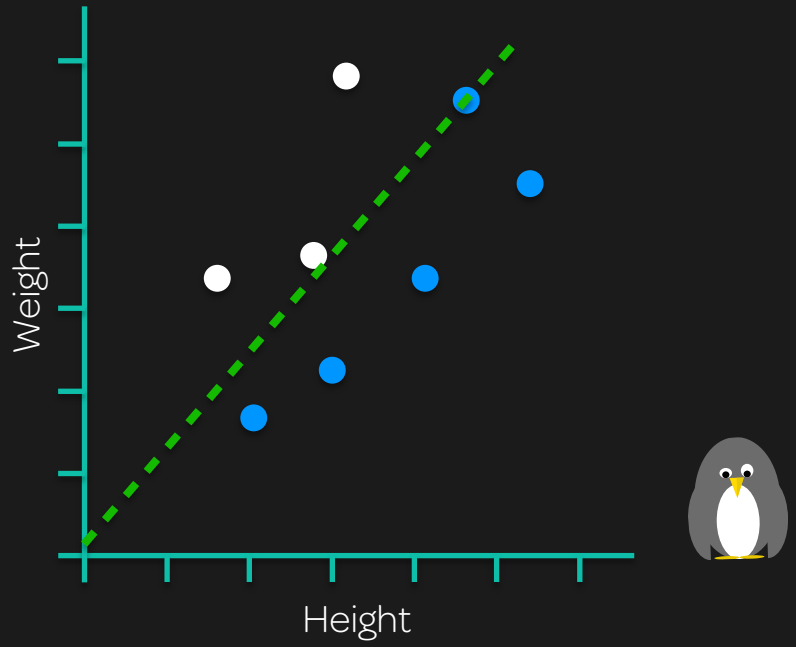
Logistical Regression



Support Vector Machines

# Regularization

- **Regularization** through **Regression**
  - L1 regularization (Lasso Regression)
  - L2 regularization (Ridge Regression)

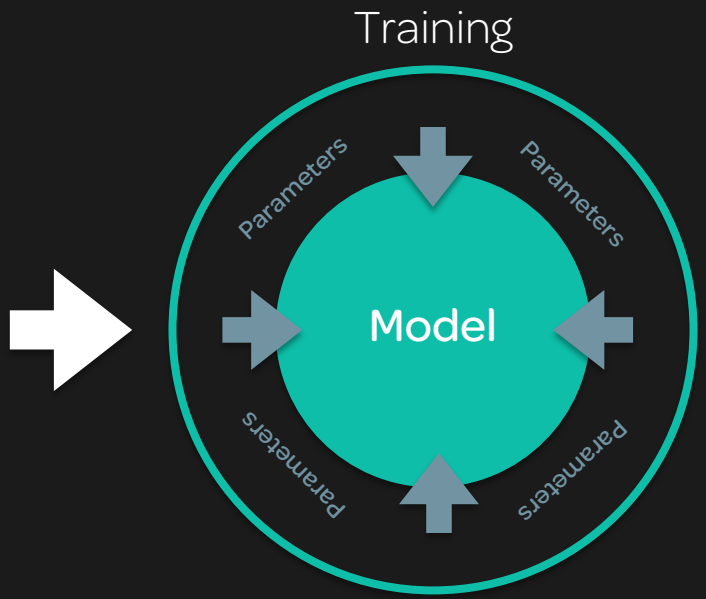


# Regularization

- We apply **regularization** when our model is **overfit**.
- Regularization is achieved through **regression**.
- Regression **desensitizes** our model to the data such that it **generalizes** better.

# Hyperparameters

Hyperparameters




- Learning rate
- Epochs
- Batch size

# Hyperparameters

## Learning Rate

- Determines the size of the step taken during gradient descent optimization
- Between 0 and 1

## Batch Size

- The number of samples used to train at any one time
- Could be all, one, or some of your data (batch, stochastic, or mini-batch)
- Often 32, 64, and 128
- Calculable from infrastructure

## Epochs

- The number of times that the algorithm will process the entire training data
- Each epoch contains one or more batches
- Each epoch should see the model get closer to the desired state
- Usually a high number: 10, 100, 1,000, and up

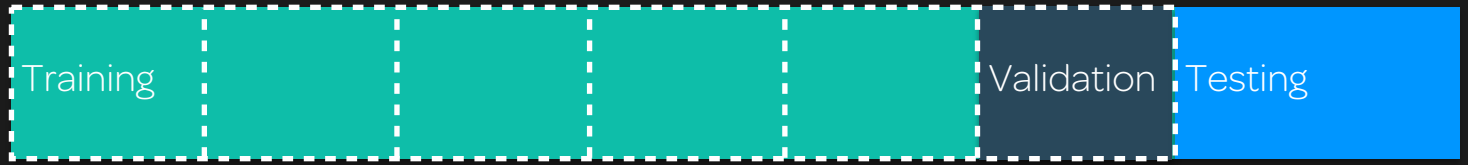
# Cross-Validation

- Is NOT “seen” by training process
- Indirectly influences the model


- Is “seen” by training process
- Directly influences the model

- Is NOT “seen” by training process
- Informs user of success

# Cross-Validation



# Feature Selection Tips:

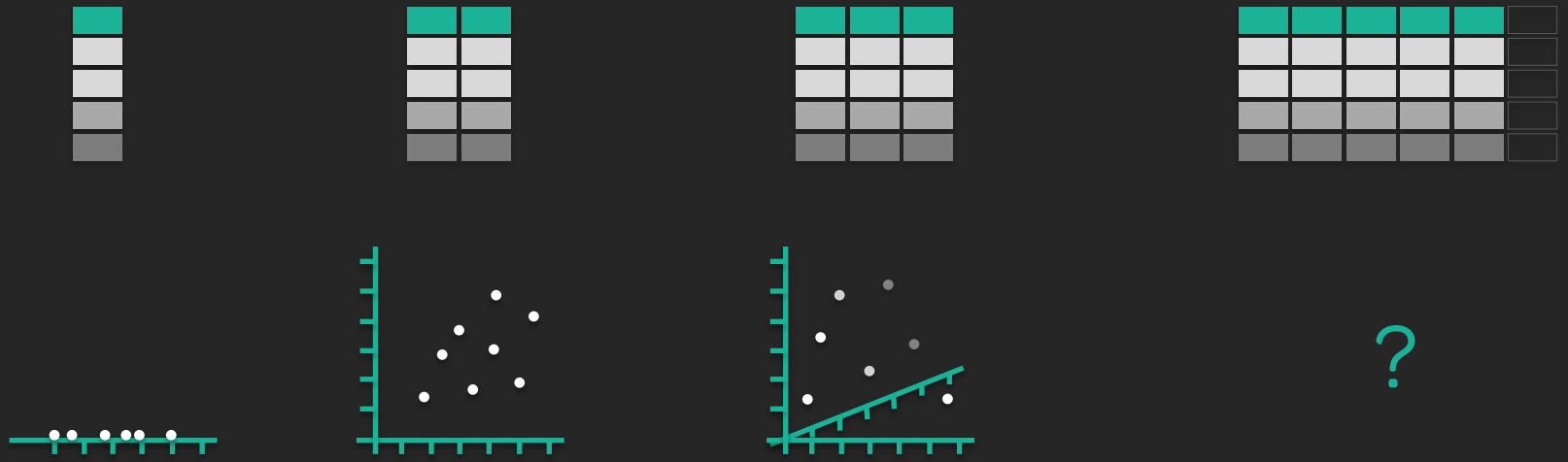
- Use domain knowledge to drop irrelevant features
- Drop features with very low correlation to the labeled data
- Drop features with very low variance
- Drop features with lots of missing data

# Feature Engineering Tips:

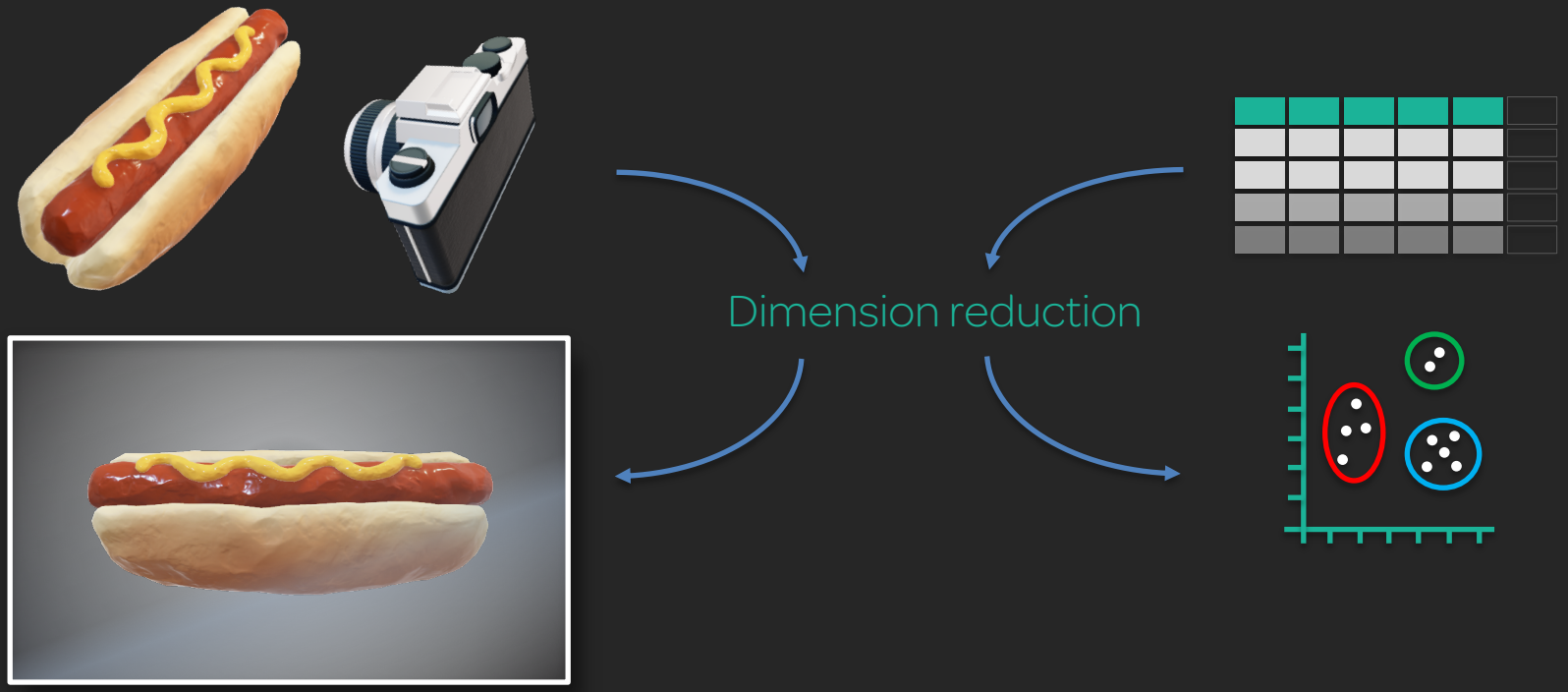
- Simplify features and remove irrelevant information
- Standardize your data ranges across features
- Transform the data to suit the model/problem

- Also see: *PCA, Label Encoding, and One Hot Encoding*

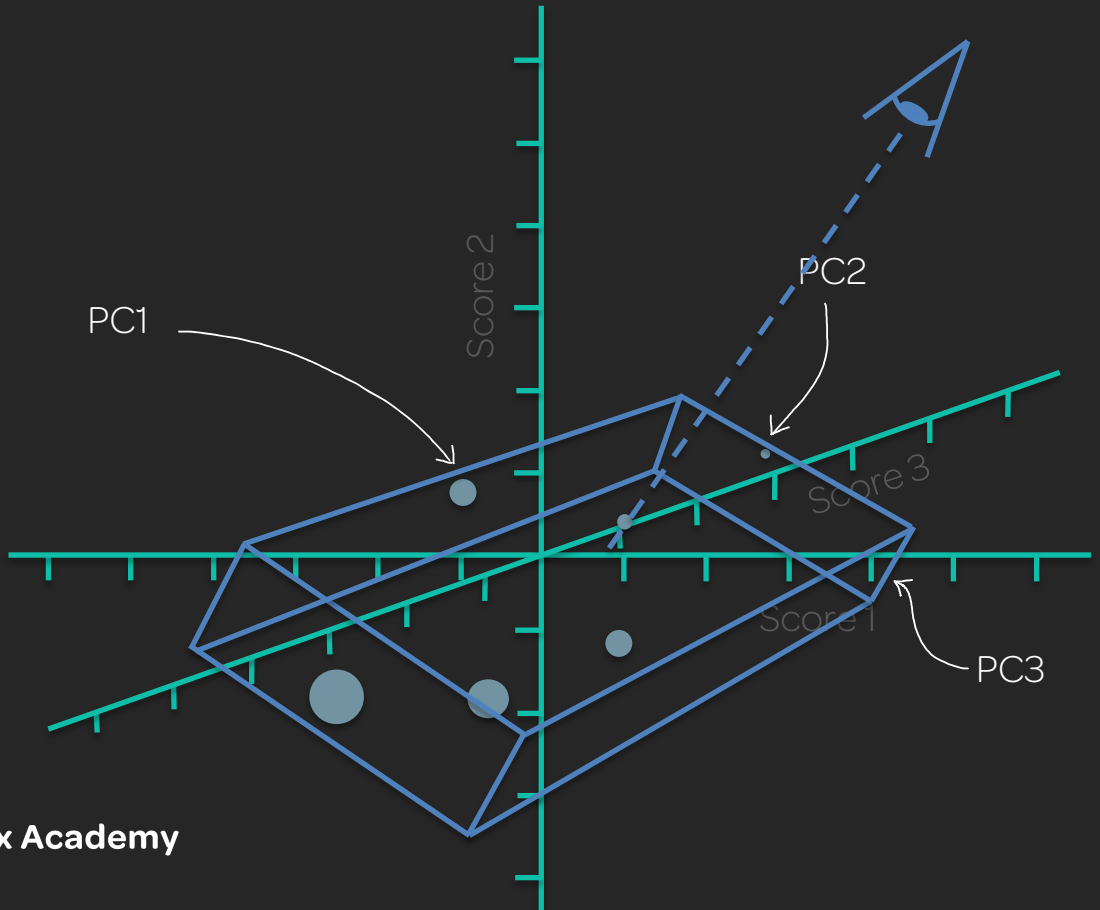
# Principal Component Analysis (PCA)



# Principal Component Analysis (PCA)



# Principal Component Analysis (PCA)



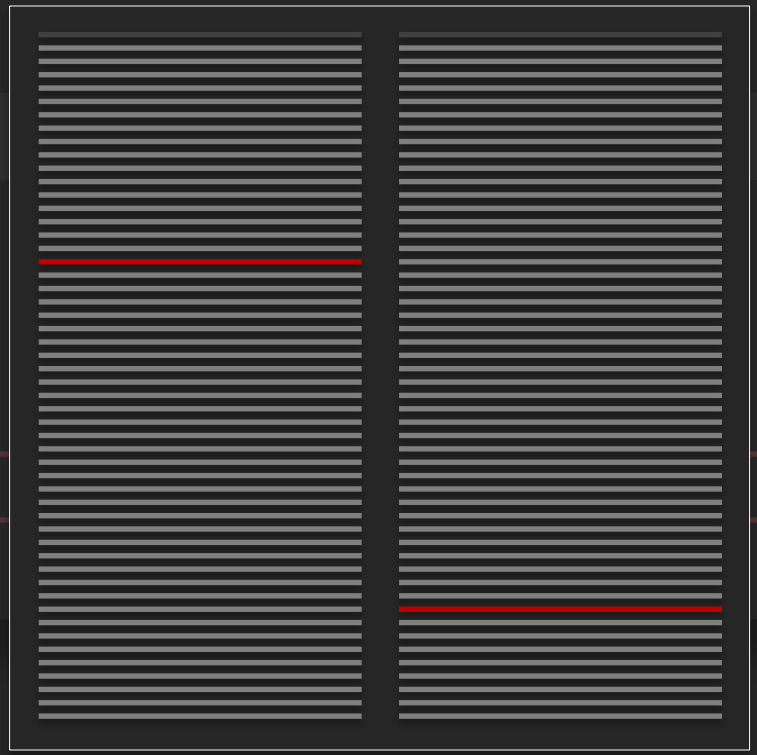
# Points to remember:

- PCA is an unsupervised ML model
- PCA is often used as a data preprocessing step
- There can be as many PC's as features or values
- PC1 and PC2 can be used to plot a 2D graph to show groups of features

# Missing and Unbalanced Data

## Unbalanced:

- Source more real data
- Oversample minority data
- Synthesize data
- Try different types of algorithm



# Label and One Hot Encoding

| COUNTRY |
| --- |
| Brazil |
| Australia |
| U.S.A. |
| U.K. |
| Australia |

| Brazil | Australia | U.S.A. | U.K. |
| --- | --- | --- | --- |
| 1 | 0 | 0 | 0 |
| 0 | 1 | 0 | 0 |
| 0 | 0 | 1 | 0 |
| 0 | 0 | 0 | 1 |
| 0 | 1 | 0 | 0 |

# Points to remember:

- Machine learning algorithms use numbers
- Use **label encoding** to replace string values
- Machine learning looks for patterns and relationships
- Use **one hot encoding** for categorical features

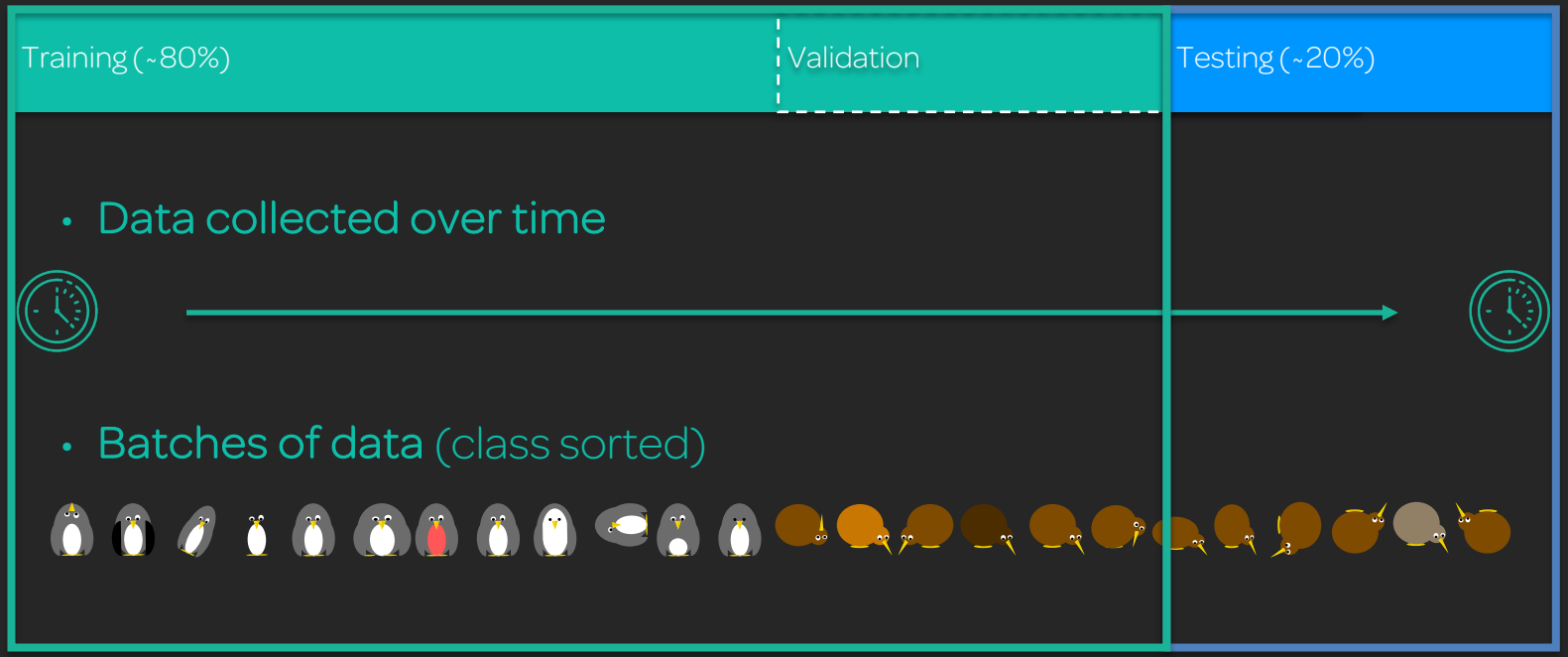
# Splitting Data


- Training
- Directly influences the model

- Validation
- Used to tune hyperparameters

- Testing
- Used to test the complete model

# Splitting Data

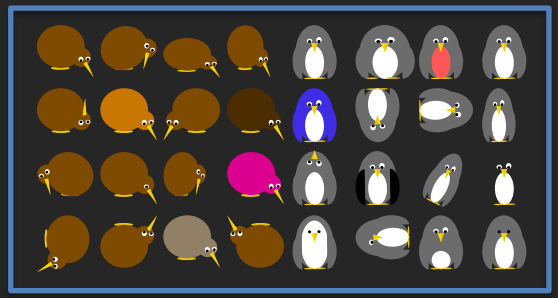


# Splitting Data

- Always randomize the data
- Even you're unaware of data clumping
- Some algorithms will shuffle data during training, but not between the training and test data sets.

# Record IO

- "Pipe mode" streams data (As opposed to "File mode")
- Faster training start times and better throughput
- Most Amazon SageMaker algorithms work best with RecordIO
  - Streams data directly from Amazon S3
  - Training instances don't need a local disk copy of data



RecordIO file

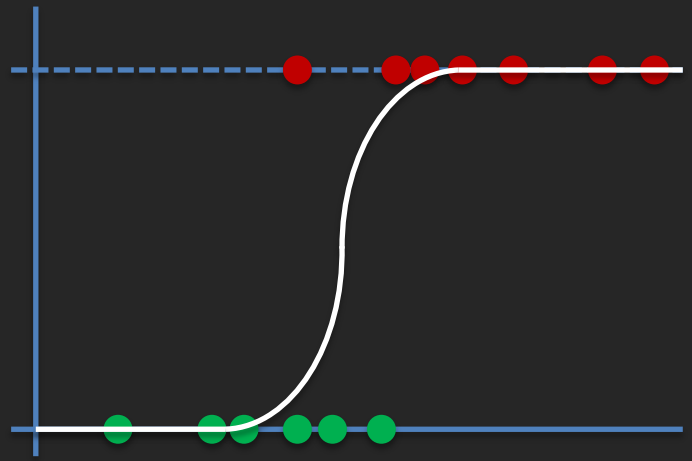
# Logistical Regression

## Type

- Supervised
- Unsupervised
- Reinforcement

## Example Inference

- Binary Yes or No
- Numeric 1, 2, 3
- Classification
- Other...



# Logistical Regression

## Type

- (●) Supervised
- ( ) Unsupervised
- ( ) Reinforcement

## Example Inference

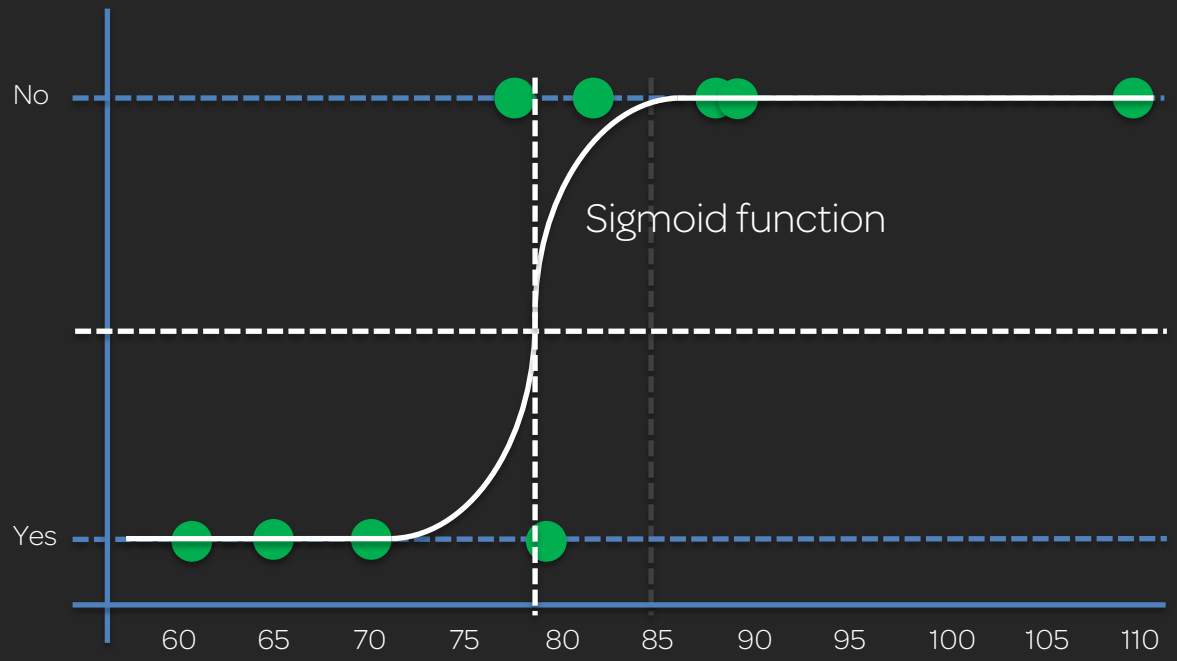
- (●) Binary Yes or No
- ( ) Numeric 1, 2, 3
- ( ) Classification
- ( ) Other...

## Use Cases

- Credit risk
- Medical conditions
- Person will perform action

# Logistical Regression

| Resting Heart Rate | Likes Cats |
|---|---|
| 70 | Yes |
| 88 | No |
| 65 | Yes |
| 89 | No |
| 78 | Yes |
| 61 | Yes |
| 77 | No |
| 110 | No |
| 82 | No |



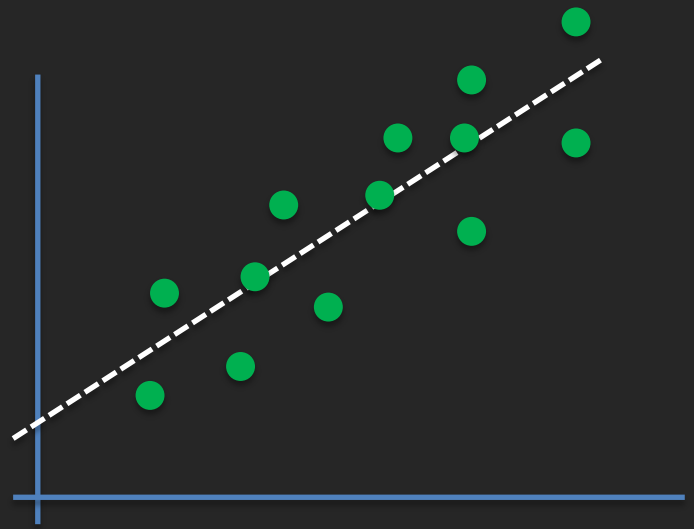
# Linear Regression

## Type

- (●) Supervised
- ( ) Unsupervised
- ( ) Reinforcement

## Example Inference

- ( ) Binary Yes or No
- (●) Numeric 1, 2, 3
- ( ) Classification
- ( ) Other...



# Linear Regression

## Type

- (●) Supervised
- ( ) Unsupervised
- ( ) Reinforcement

## Example Inference

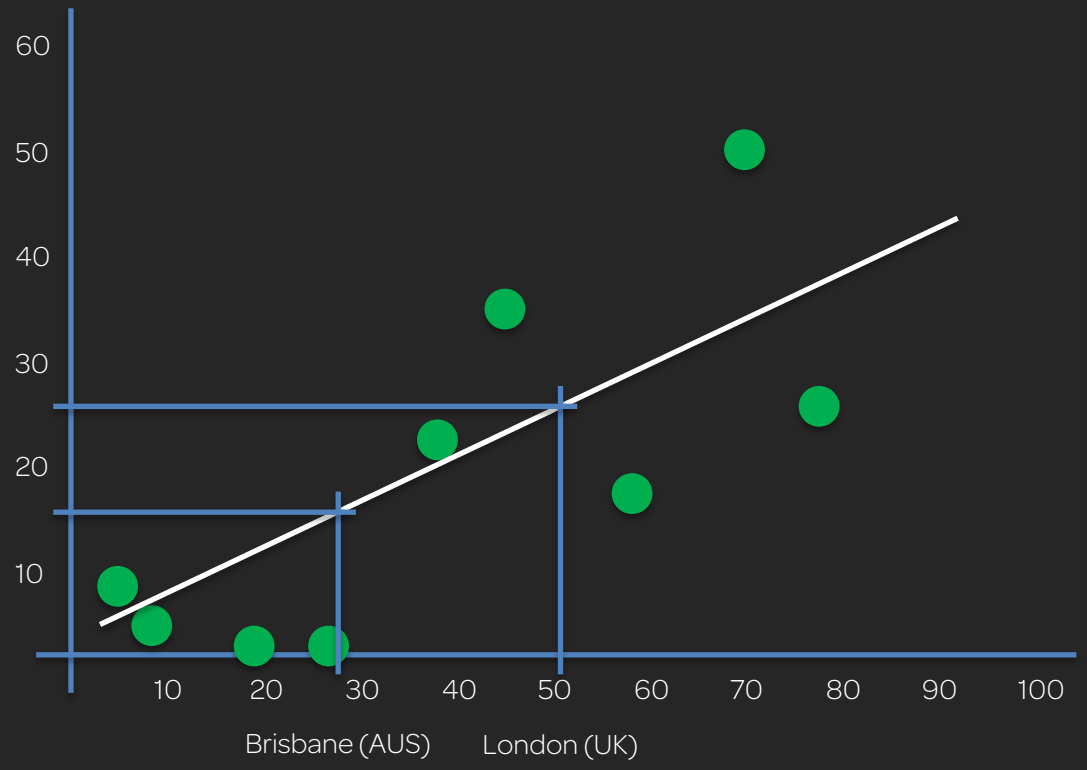
- ( ) Binary Yes or No
- (●) Numeric 1, 2, 3
- ( ) Classification
- ( ) Other...

## Use Cases


- Financial forecasting
- Marketing effectiveness
- Risk valuation

# Linear Regression

| Latitude | Tea consumed |
|---|---|
| 4 | 6 |
| 7 | 2 |
| 20 | 0 |
| 28 | 0 |
| 38 | 24 |
| 45 | 35 |
| 59 | 18 |
| 70 | 49 |
| 76 | 24 |



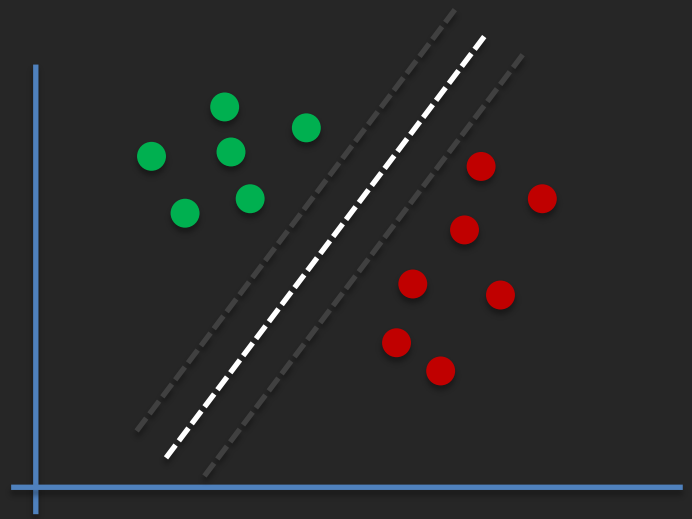
# Support Vector Machines

## Type

- (●) Supervised
- ( ) Unsupervised
- ( ) Reinforcement

## Example Inference

- ( ) Binary Yes or No
- ( ) Numeric 1, 2, 3
- (●) Classification
- ( ) Other...



# Support Vector Machines

## Type

- [x] Supervised
- [ ] Unsupervised
- [ ] Reinforcement

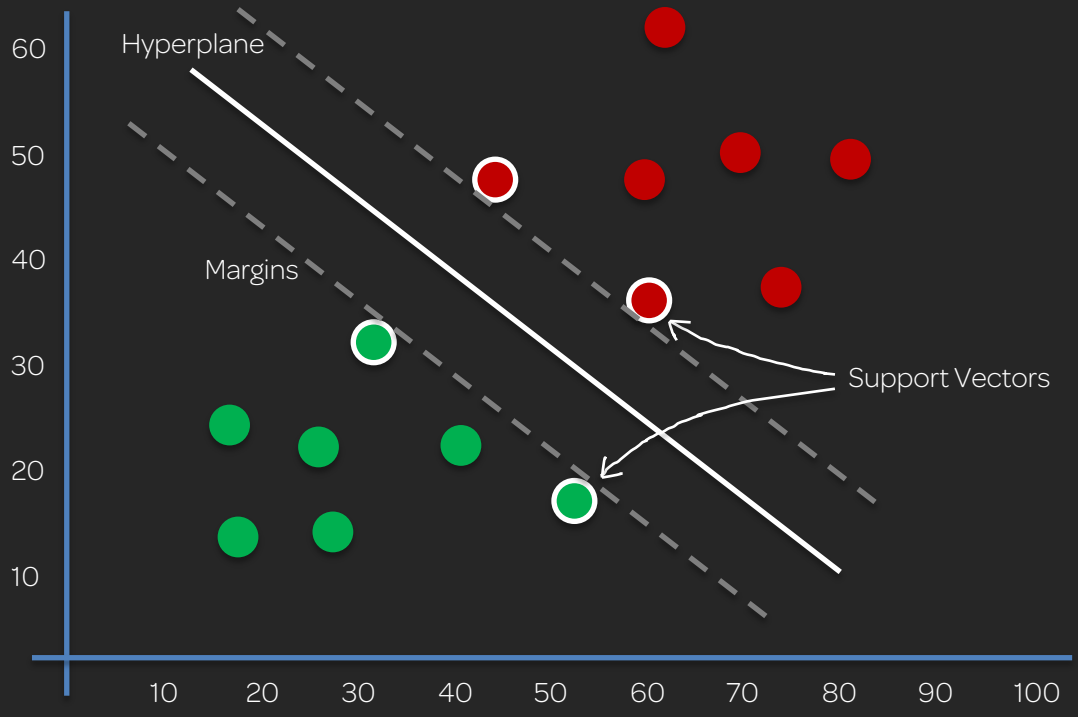
## Example Inference

- [ ] Binary Yes or No
- [ ] Numeric 1, 2, 3
- [x] Classification
- [ ] Other...

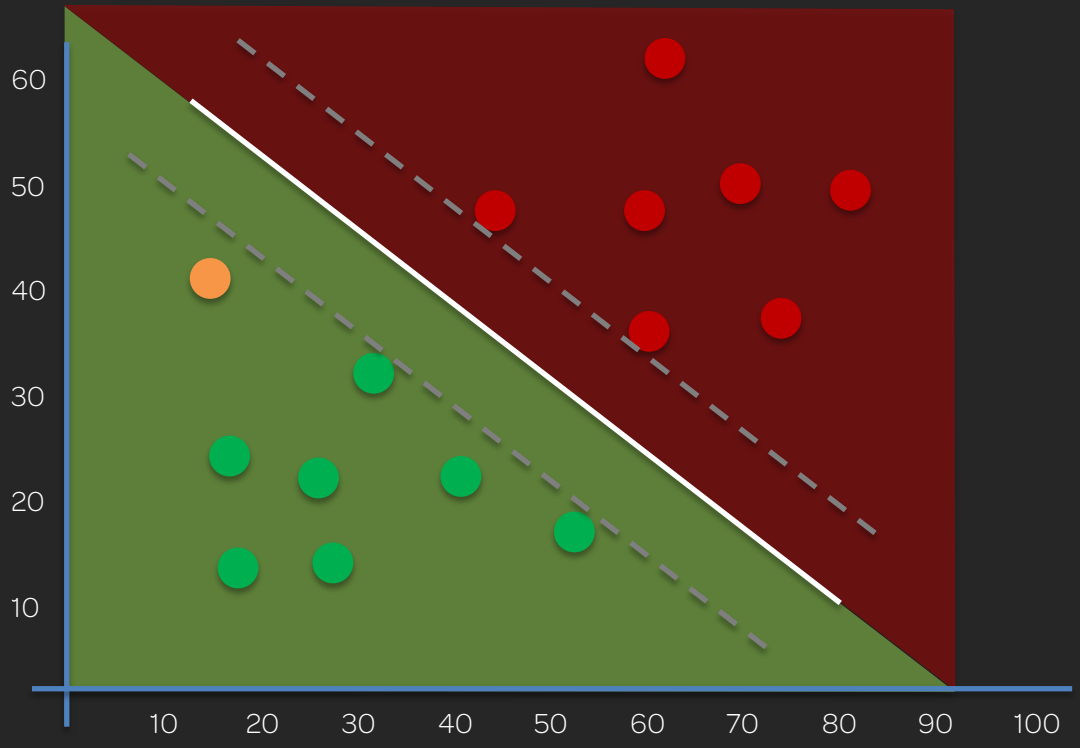
## Use Cases

- Customer classification
- Genomic identification

# Support Vector Machines



# Support Vector Machines



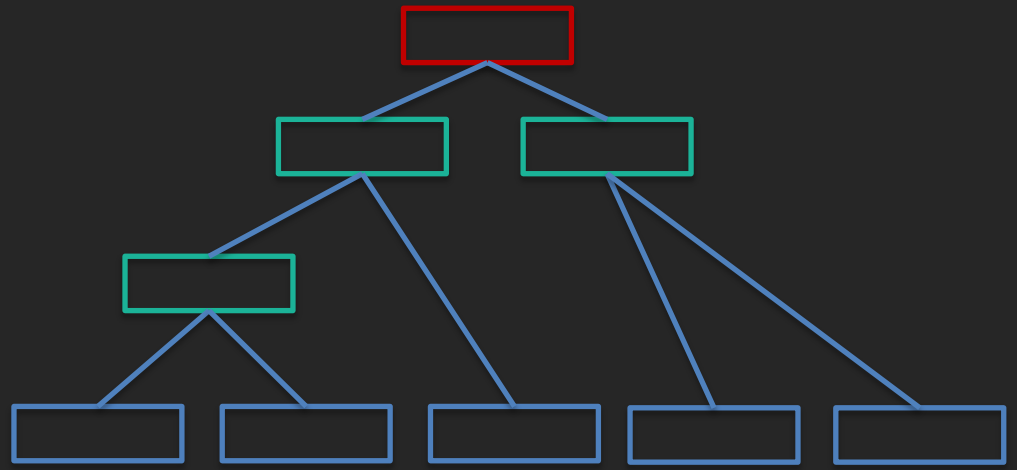
# Decision Trees

## Type

- Supervised
- Unsupervised
- Reinforcement

## Example Inference

- Binary Yes or No
- Numeric 1, 2, 3
- Classification
- Other...



# Decision Trees

## Type

- Supervised
- Unsupervised
- Reinforcement

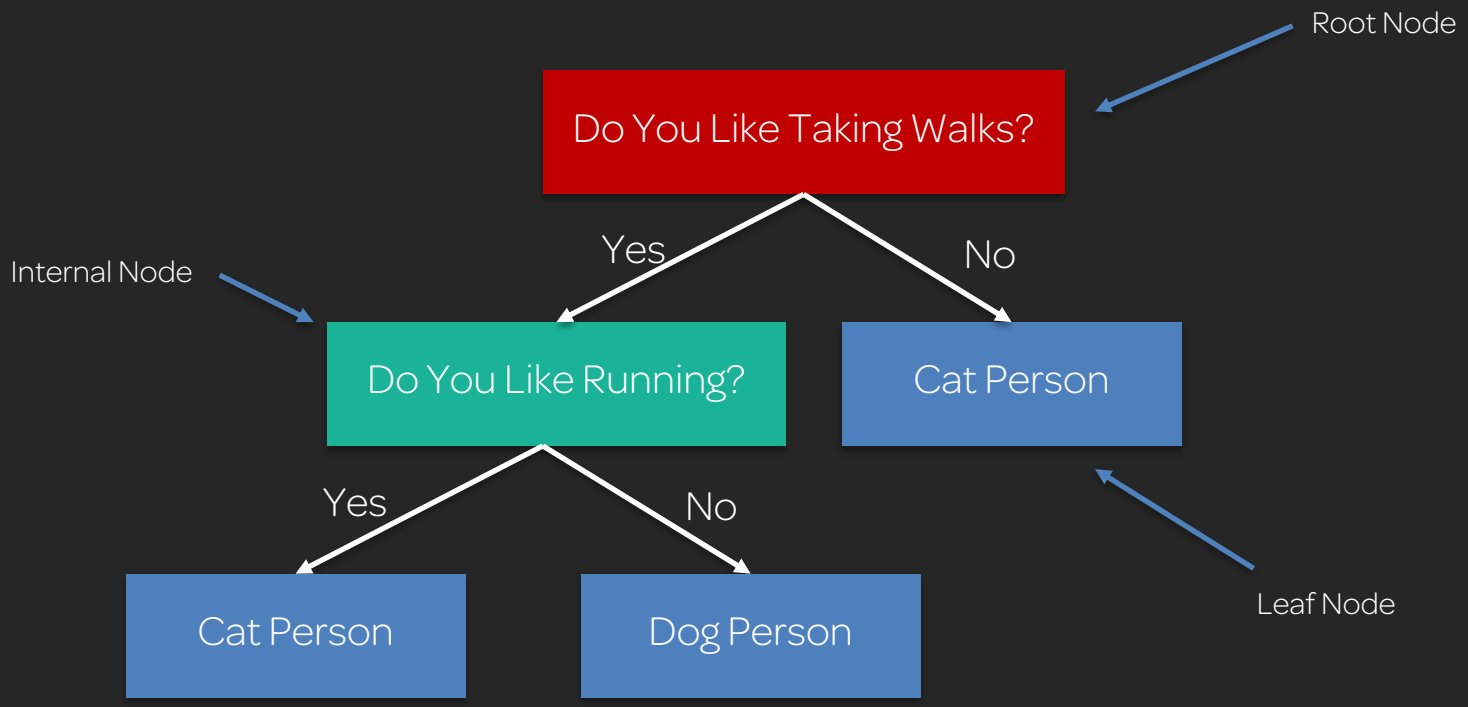
## Example Inference

- Binary Yes or No
- Numeric 1, 2, 3
- Classification
- Other...

## Use Cases

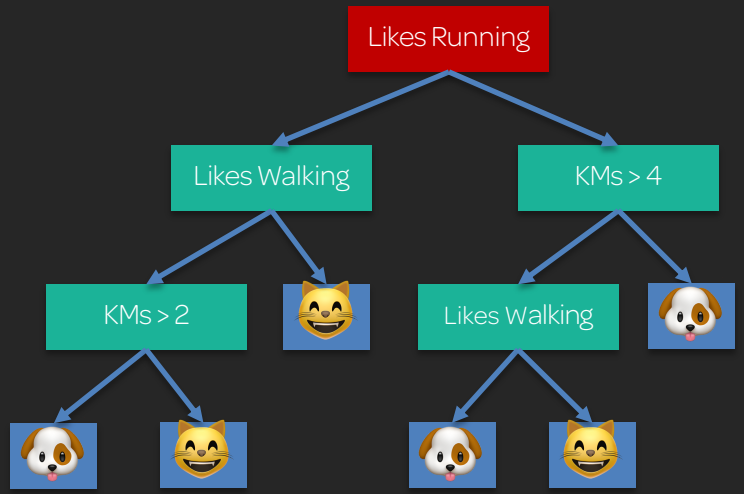
- Customer analysis
- Medical conditions

# Decision Trees



# Decision Trees

| Likes Walking | Likes Running | KM's Walked | Favourite Color | Type |
|---|---|---|---|---|
| No | Yes | 1 | Green | Dog |
| No | No | 2 | Blue | Cat |
| Yes | Yes | 1 | Red | Dog |
| Yes | No | 1 | Green | Cat |
| Yes | No | 3 | Green | Dog |
| Yes | Yes | 4 | Blue | Dog |
| No | No | 3 | Red | Cat |
| ... | ... | ... | ... | ... |



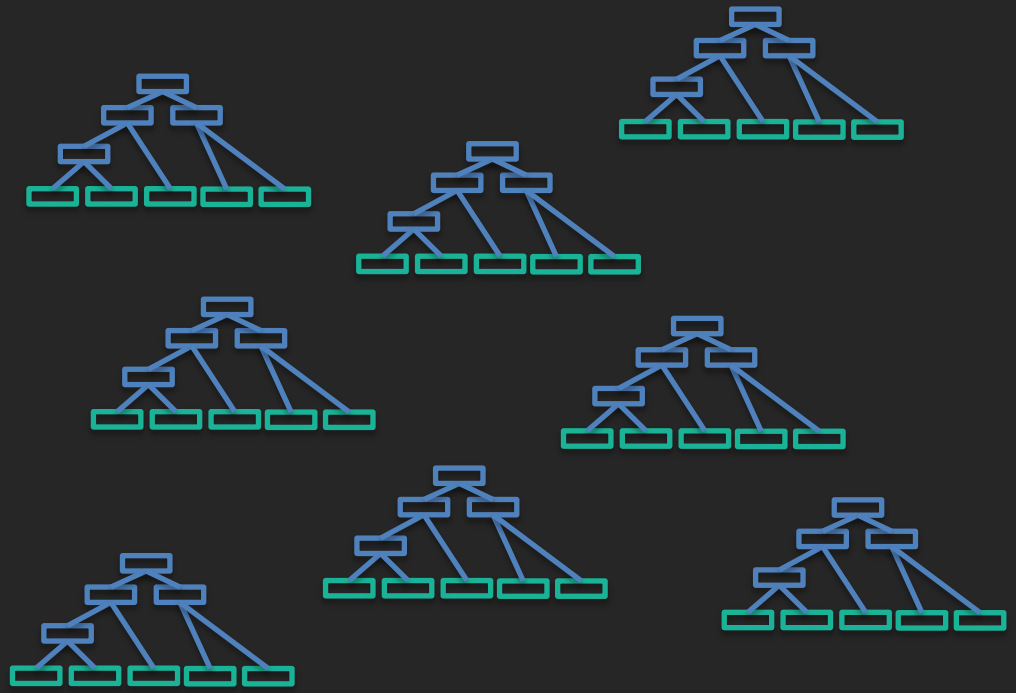
# Random Forests

## Type

- Supervised
- Unsupervised
- Reinforcement

## Example Inference

- Binary Yes or No
- Numeric 1, 2, 3
- Classification
- Other...



# Random Forests

## Type

- [x] Supervised
- [ ] Unsupervised
- [ ] Reinforcement

## Example Inference

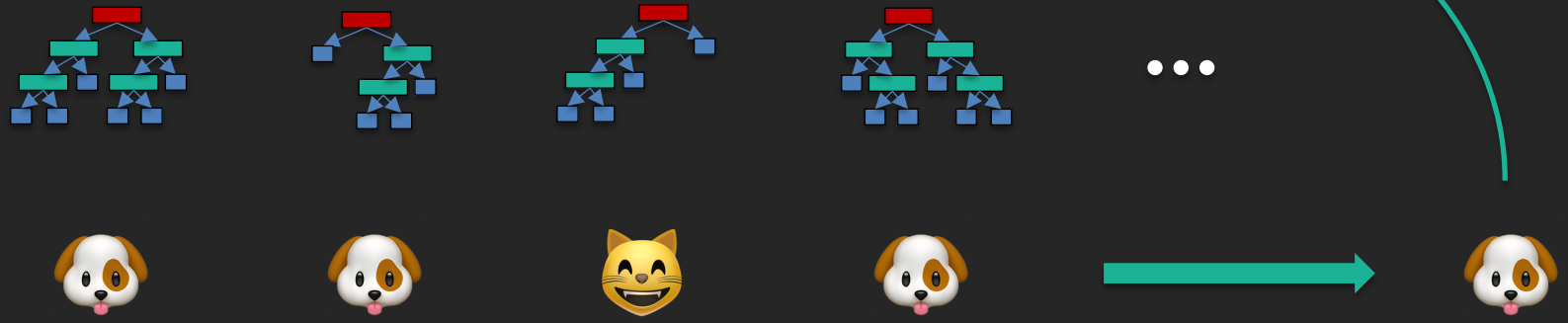
- [x] Binary Yes or No
- [ ] Numeric 1, 2, 3
- [x] Classification
- [ ] Other...

## Use Cases

- Customer analysis
- Medical conditions

# Random Forests

| Likes Walking | Likes Running | KM's Walked | Favourite Color | Type |
|---|---|---|---|---|
| Yes | No | 3 | Red | Dog |



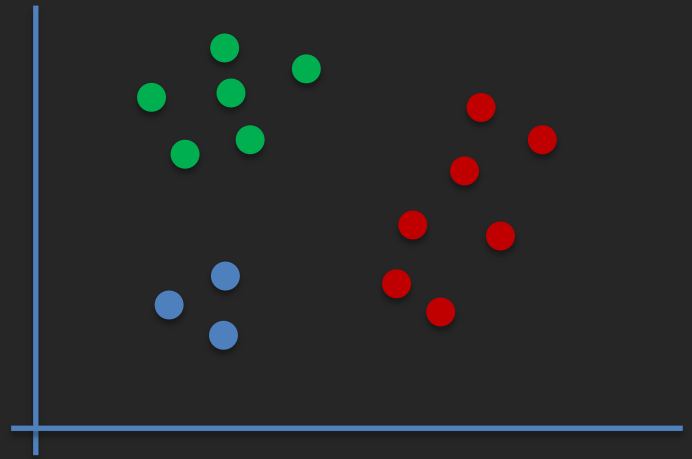
# K-Means

## Type

- Supervised
- Unsupervised
- Reinforcement

## Example Inference

- Binary Yes or No
- Numeric 1, 2, 3
- Classification
- Other...



# K-Means

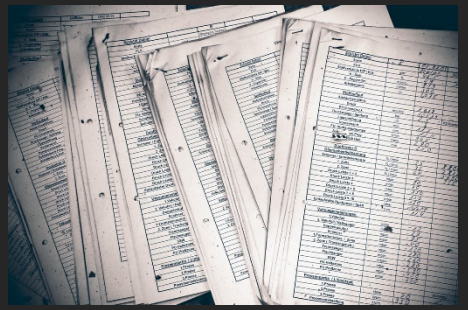
## Type

- ○ Supervised
- ● Unsupervised
- ○ Reinforcement

## Example Inference

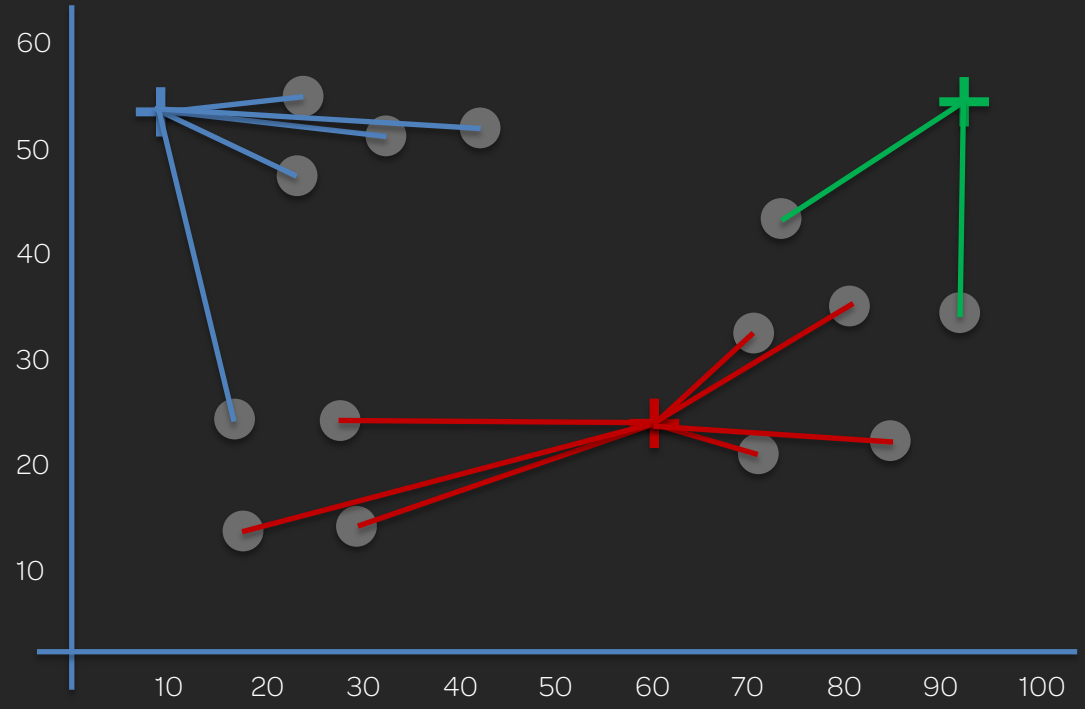
- ○ Binary Yes or No
- ○ Numeric 1, 2, 3
- ● Classification
- ○ Other...

## Use Cases

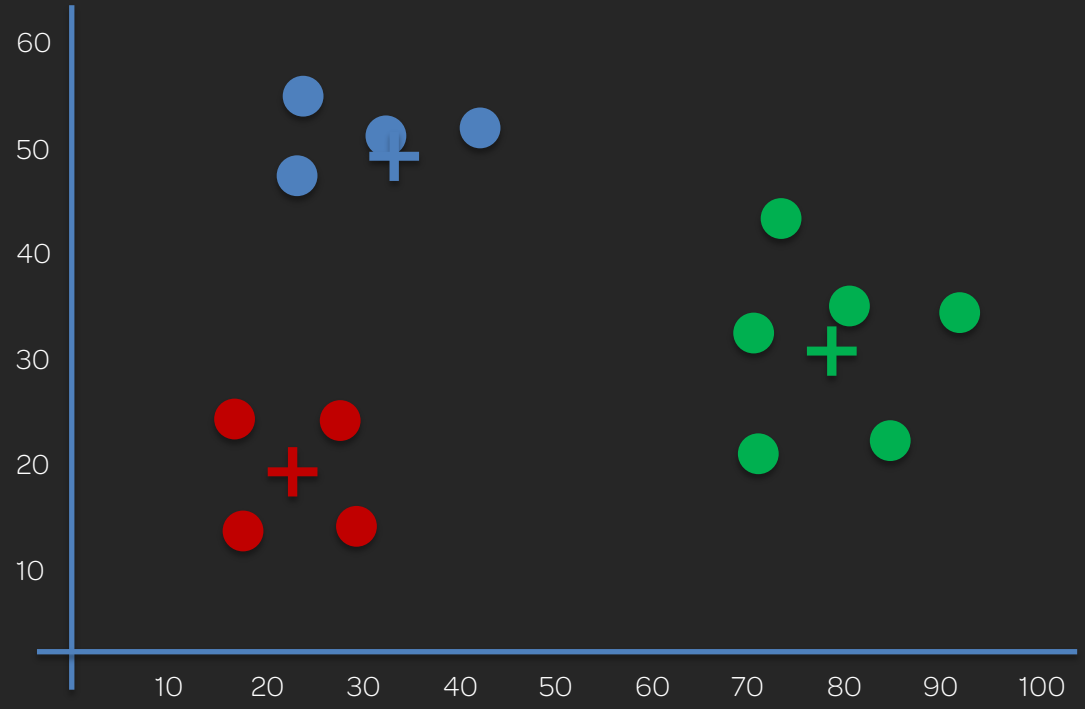


- Data exploration
- Customer categorization

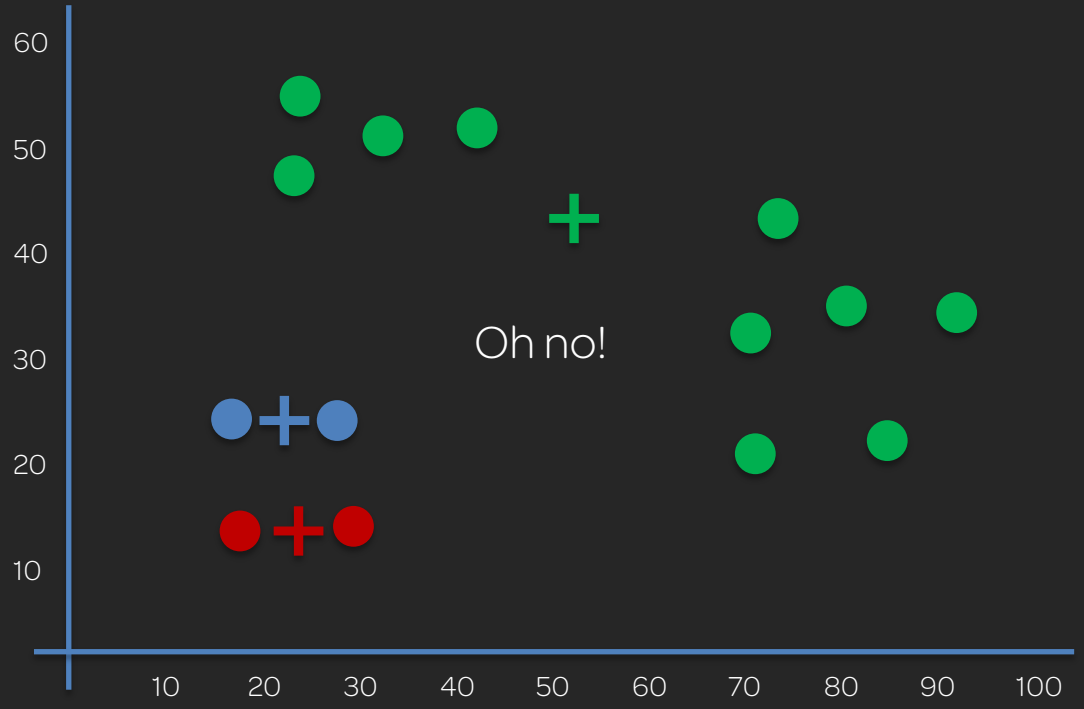
# K-Means



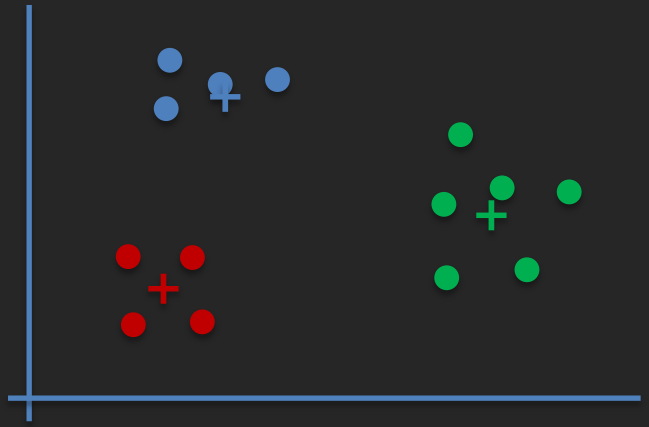
# K-Means



# K-Means



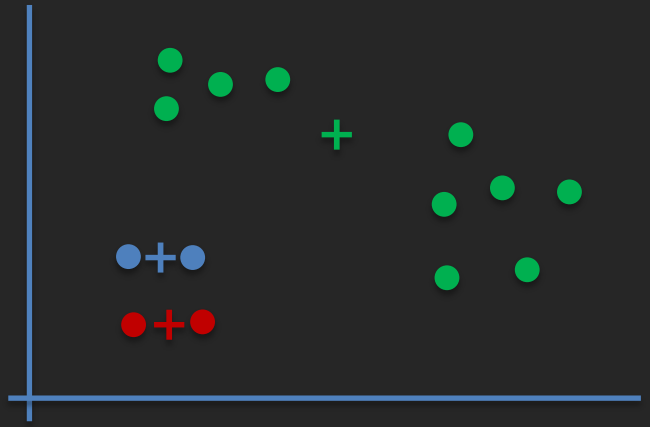
# K-Means



Total variation:

Winner

Less variation

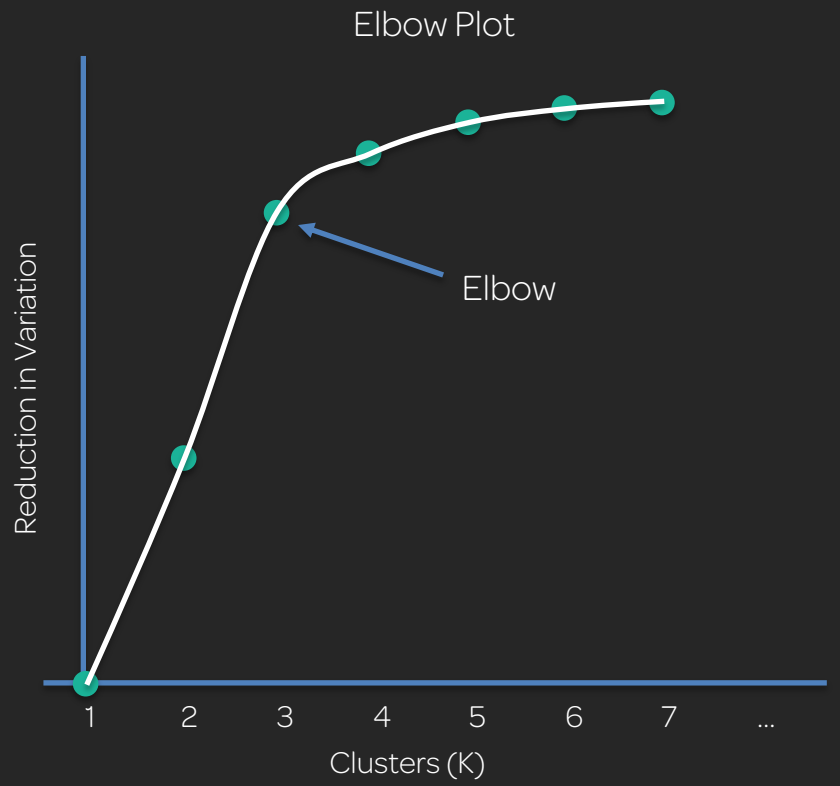


Total variation:

# K-Means

- One cluster results in maximum variation.
- Two clusters reduces the variation.
- Three clusters reduces the variation more.
- And so on...

- The variation doesn't change much after three clusters.



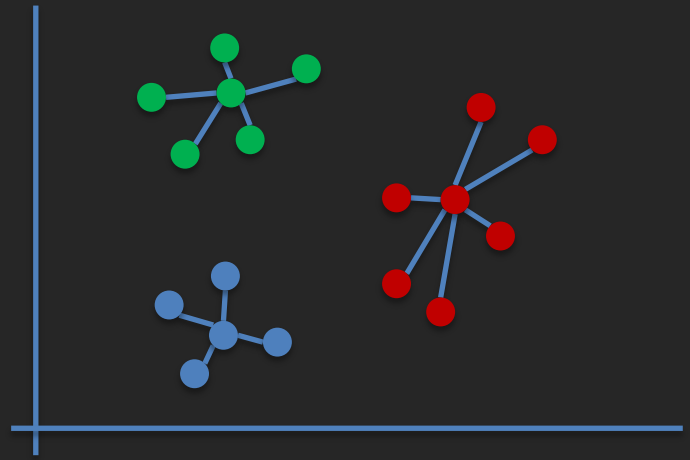
# K-Nearest Neighbor

## Type

- [x] Supervised
- [ ] Unsupervised
- [ ] Reinforcement

## Example Inference

- [ ] Binary Yes or No
- [ ] Numeric 1, 2, 3
- [x] Classification
- [ ] Other...



# K-Nearest Neighbor

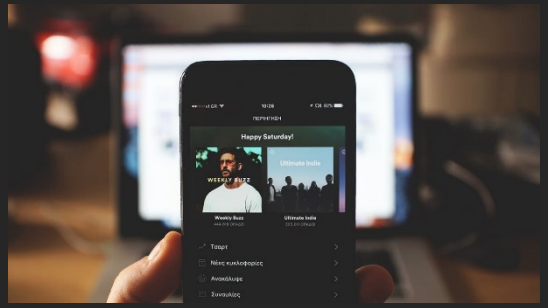
## Type

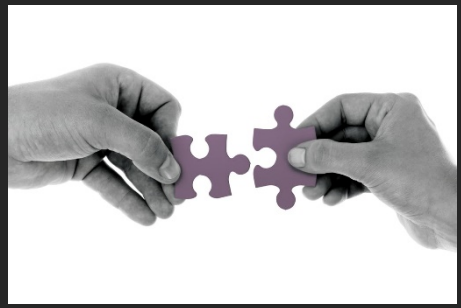
- (●) Supervised
- ( ) Unsupervised
- ( ) Reinforcement

## Example Inference

- ( ) Binary Yes or No
- ( ) Numeric 1, 2, 3
- (●) Classification
- ( ) Other...

## Use Cases



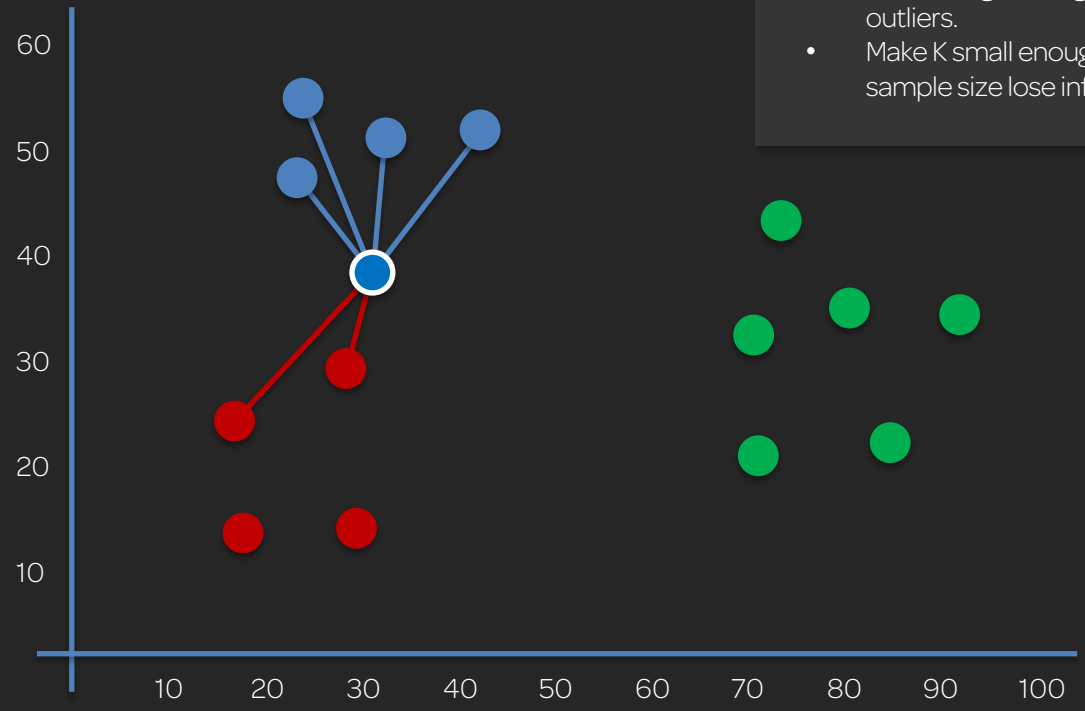



- Recommendation engine
- Similar articles and objects

# K-Nearest Neighbor

How to determine the value to use for K:

- Make K large enough to reduce the influence of outliers.
- Make K small enough that classes with a small sample size lose influence.



# Latent Dirichlet Allocation (LDA)

## Type

- ○ Supervised
- ● Unsupervised
- ○ Reinforcement

## Example Inference

- ○ Binary Yes or No
- ○ Numeric 1, 2, 3
- ● Classification
- ● Other...

# Latent Dirichlet Allocation (LDA)

## Type

- Supervised
- Unsupervised
- Reinforcement

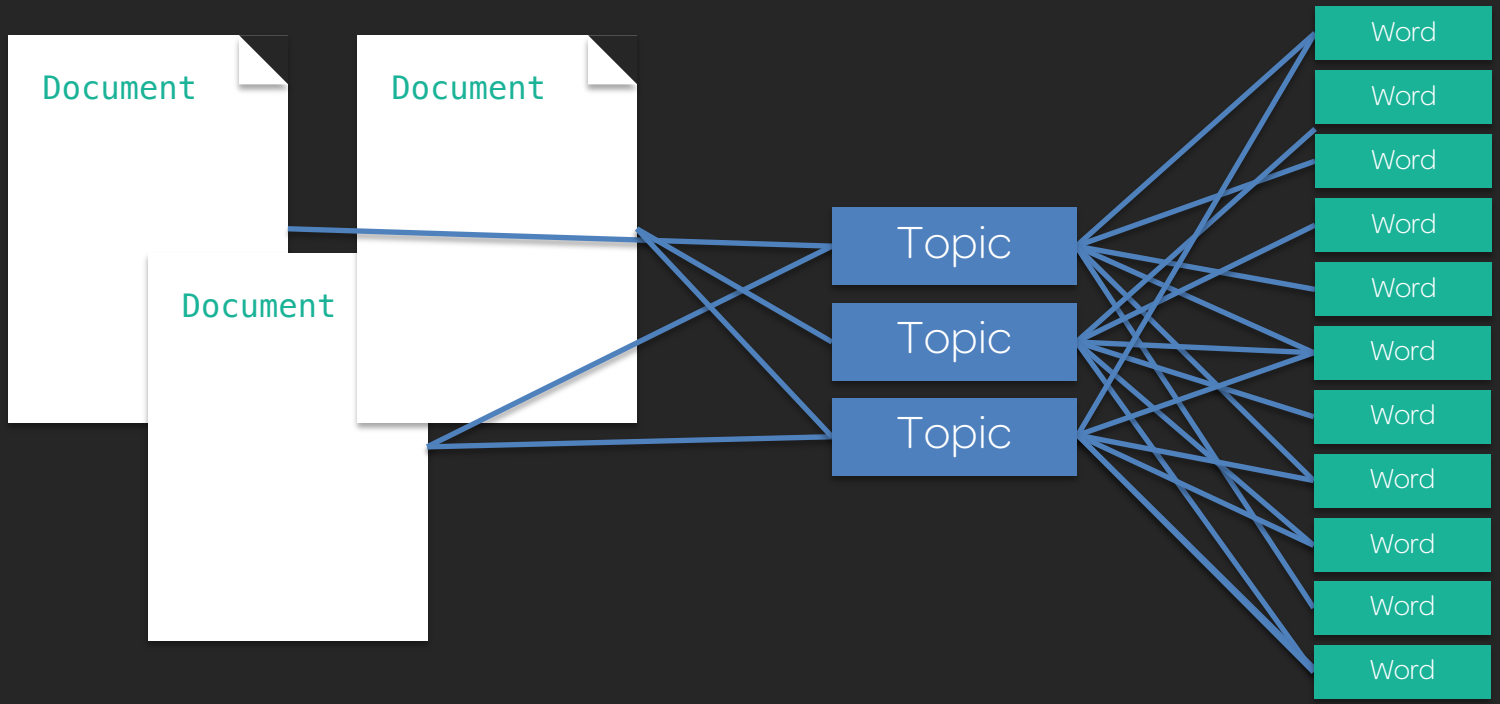
## Example Inference

- Binary Yes or No
- Numeric 1, 2, 3
- Classification
- Other...

## Use Cases


- Topic discovery
- Sentiment analysis
- Automated document tagging

# Latent Dirichlet Allocation (LDA)



# Latent Dirichlet Allocation (LDA)

- Reassign the words to topics.

  51 * 24 = 1224
  43 * 35 = 1505
  23 * 132 = **3036**

- Iterate over all the words in all the documents N times.

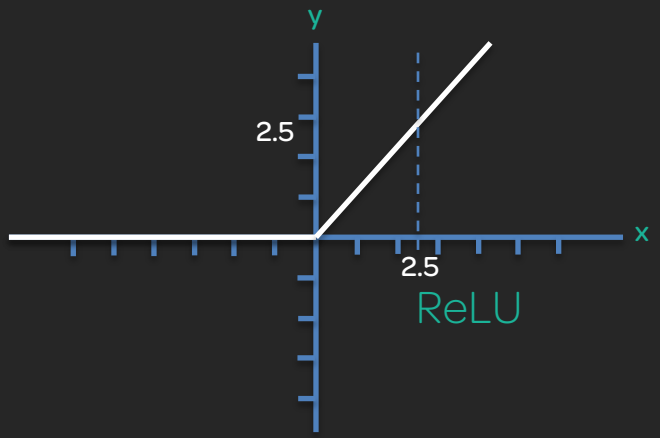
| | Topic |
|---|---|
| Word | 3 |
| Word | 2 |
| Word | 3 |
| Word | 1 |
| Word | 2 |
| Word | 3 |
| Word | 2 |
| Word | 3 |
| Word | 1 |
| Word | 2 |
| Word | 2 |
| Word | 3 |
| Word | 1 |
| ... | 1 |

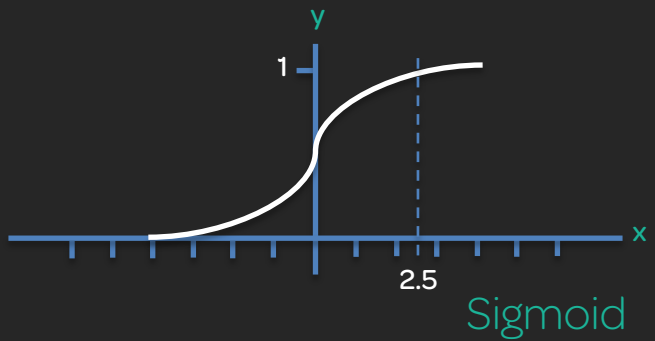
| Word | Topic 1 | Topic 2 | Topic 3 |
|---|---|---|---|
| machine learning | 22 | 33 | 43 |
| fun run | 32 | 34 | 23 |
| deep learning | 44 | 23 | 34 |
| python | 51 | 43 | 23 |
| storage | 33 | 64 | 54 |
| artificial intelligence | 45 | 33 | 23 |

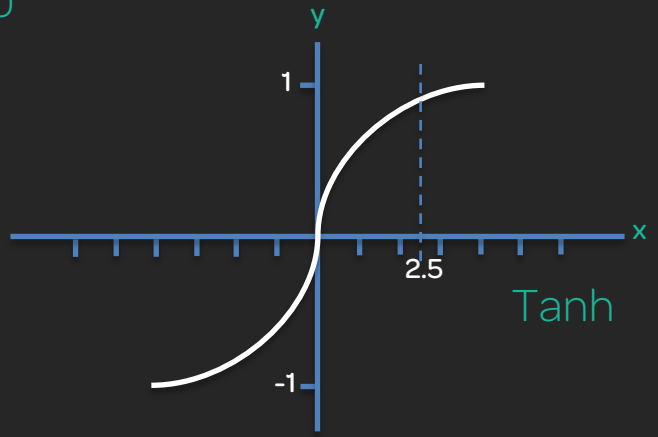
| Document | Topic 1 | Topic 2 | Topic 3 |
|---|---|---|---|
| Storage | 123 | 23 | 34 |
| Machine Learning | 43 | 143 | 45 |
| Lambda | 24 | 35 | 132 |

# Neural Networks

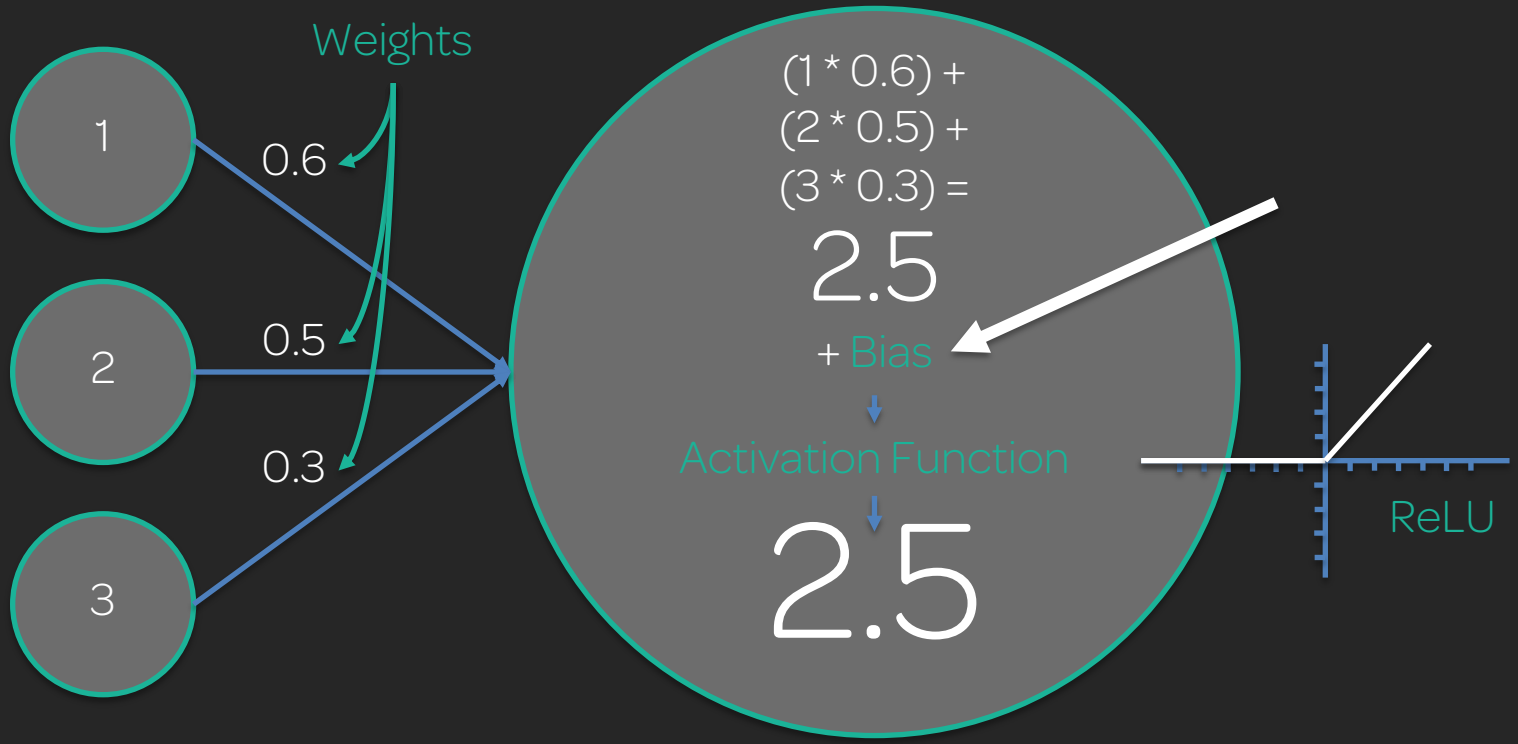
## Activation Functions:



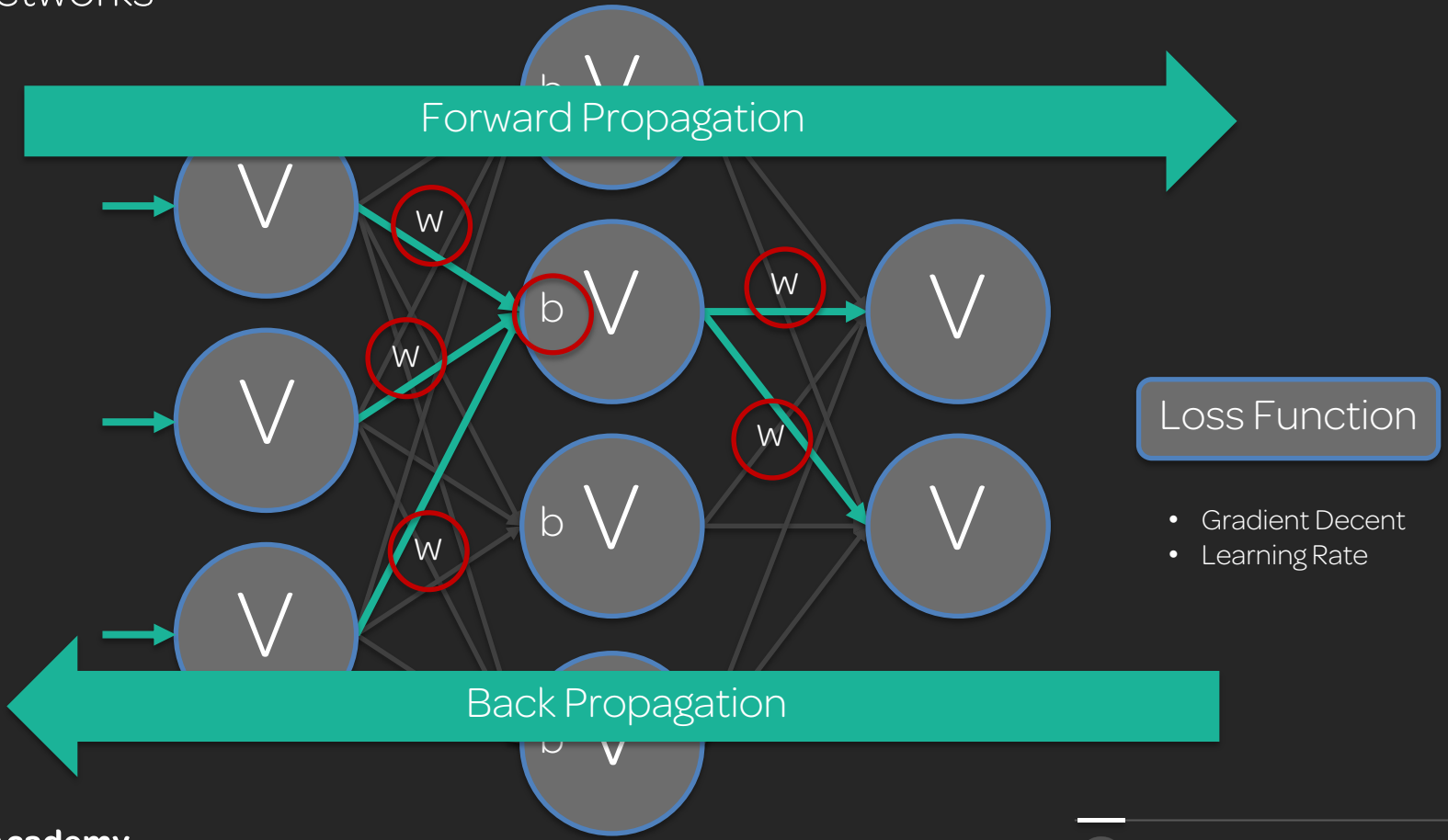



# Neural Networks



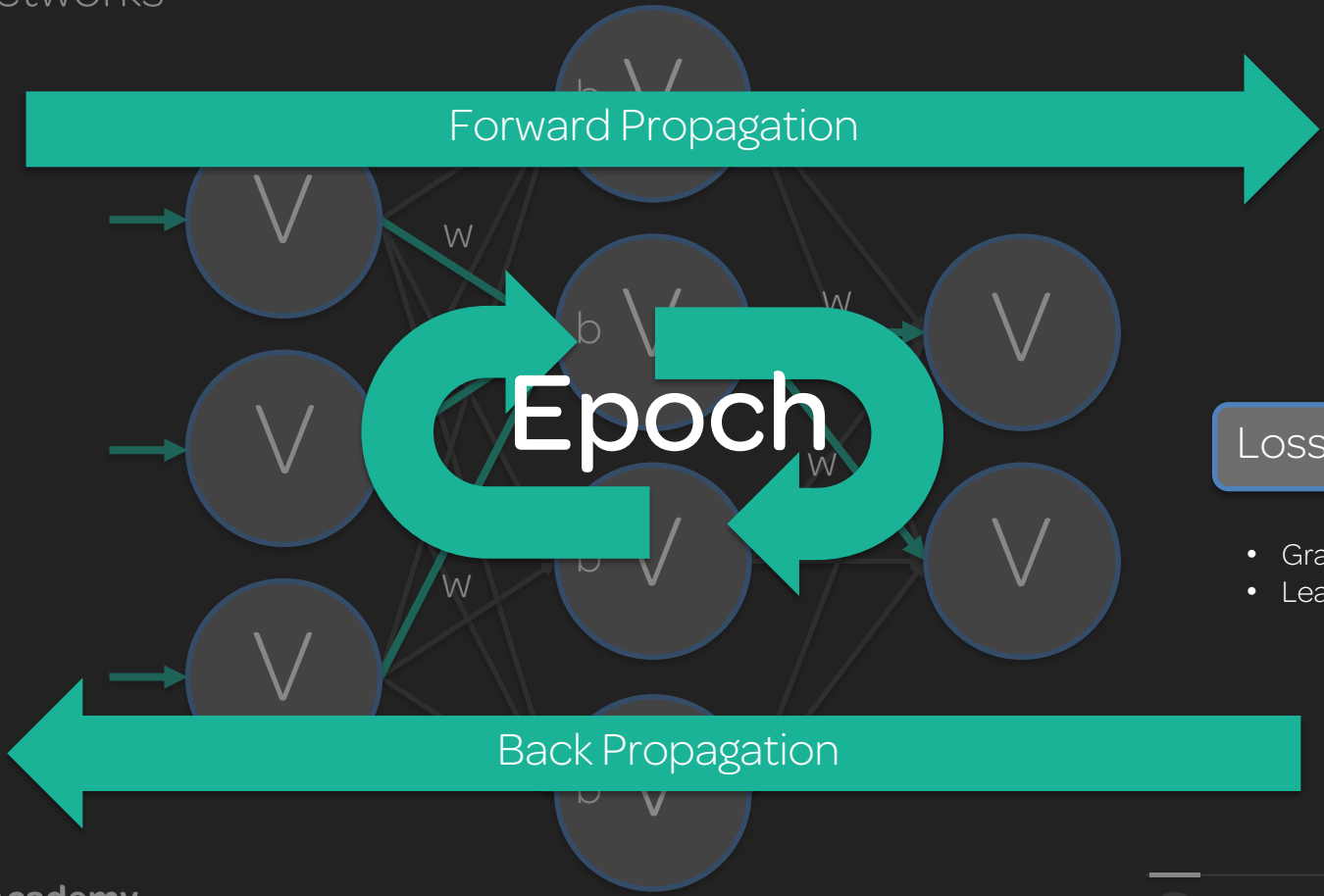
# Neural Networks




Loss Function

- Gradient Decent
- Learning Rate

# Neural Networks




Loss Function

- Gradient Decent
- Learning Rate

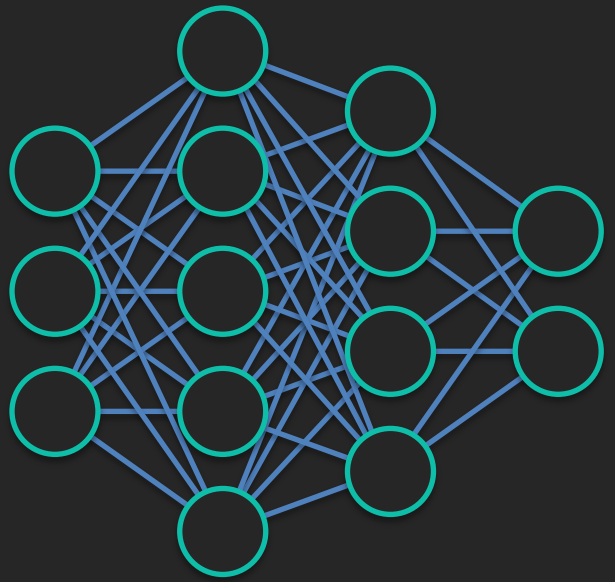
# Convolutional Neural Networks (CNN)

## Type

- (●) Supervised
- ( ) Unsupervised
- ( ) Reinforcement

## Example Inference

- ( ) Binary Yes or No
- ( ) Numeric 1, 2, 3
- (●) Classification
- ( ) Other...



# Convolutional Neural Networks (CNN)

## Type

- [x] Supervised
- [ ] Unsupervised
- [ ] Reinforcement

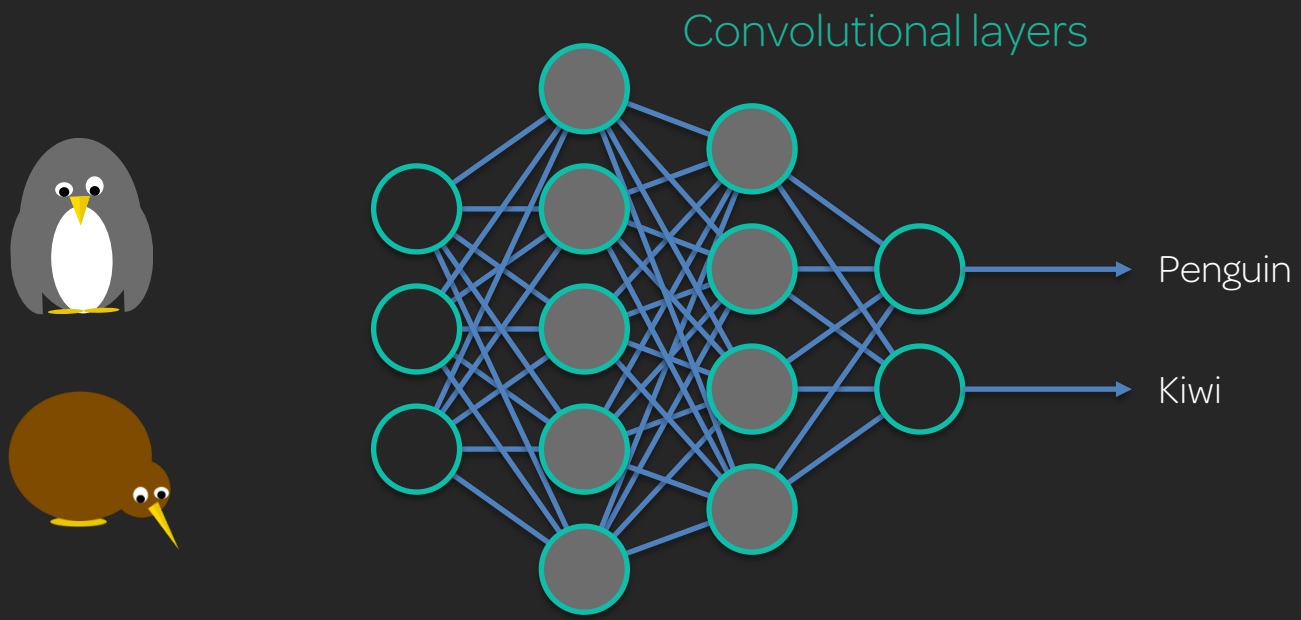
## Example Inference

- [ ] Binary Yes or No
- [ ] Numeric 1, 2, 3
- [x] Classification
- [ ] Other...

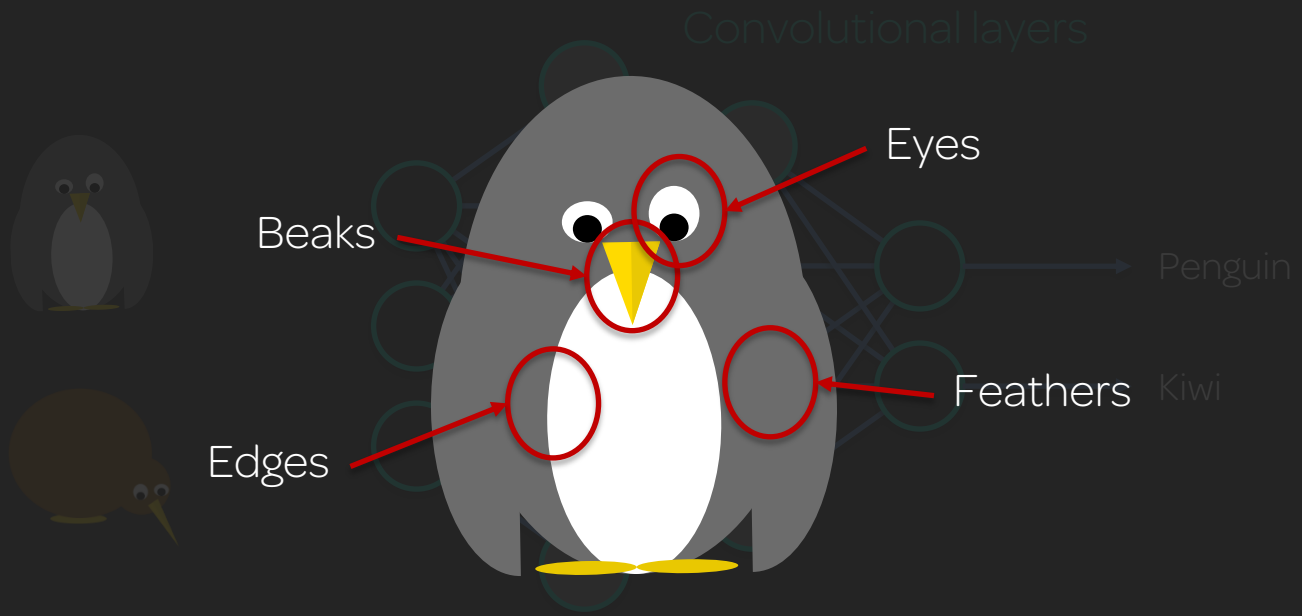
## Use Cases


- Image classification
- Spatial analysis
- Hotdog or not-hotdog

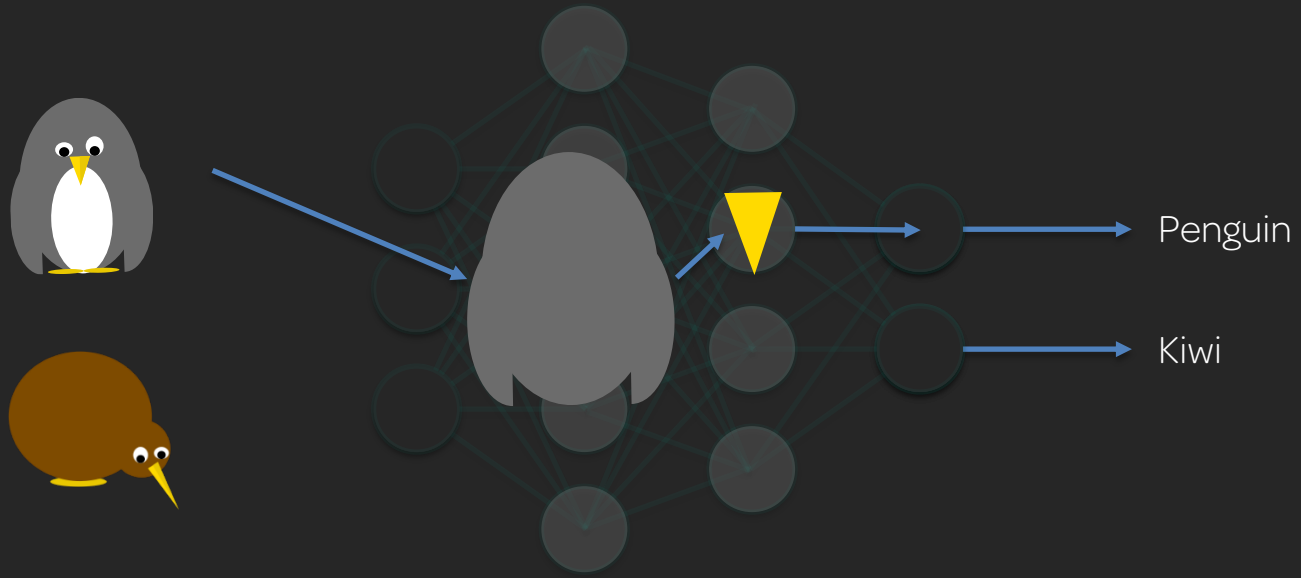
# Convolutional Neural Networks (CNN)



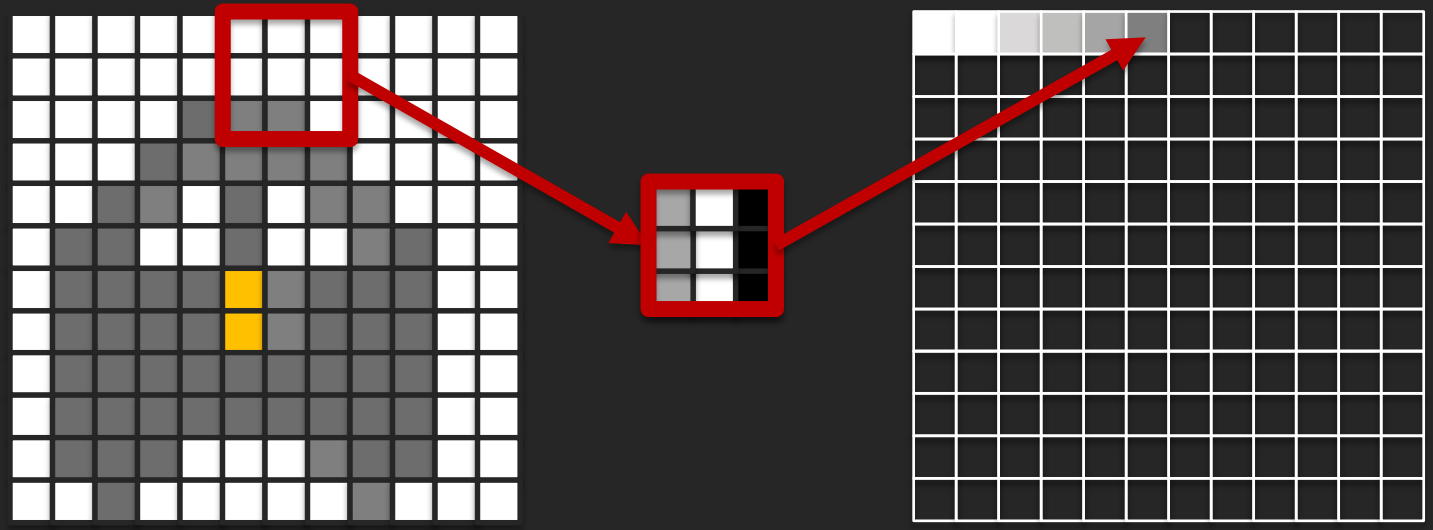
# Convolutional Neural Networks (CNN)



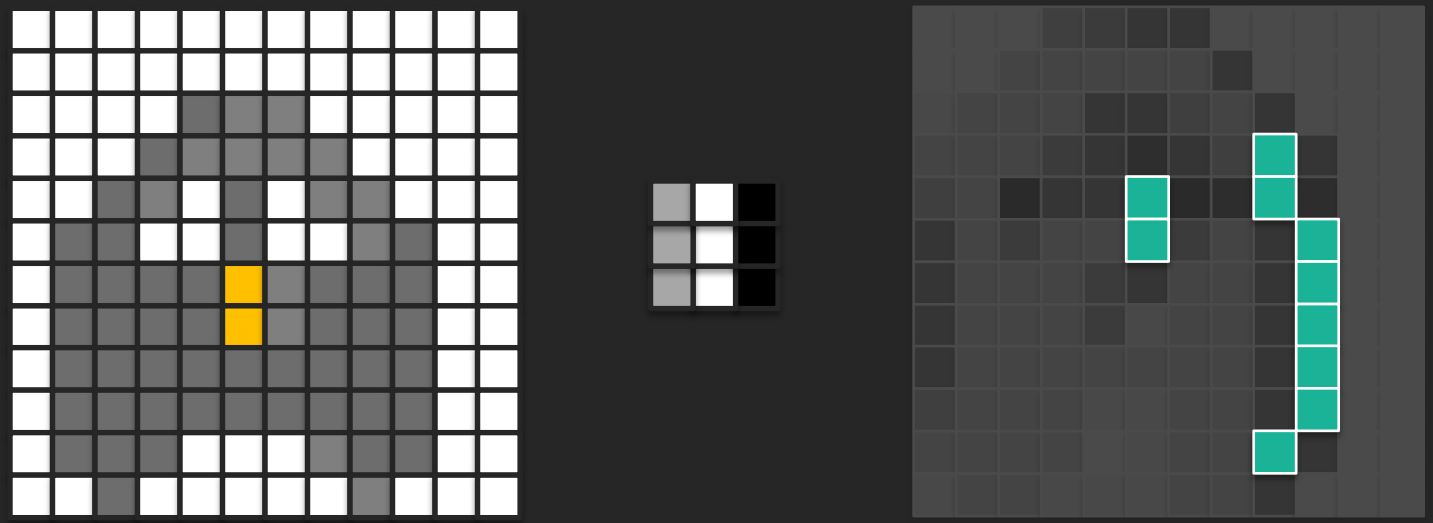
# Convolutional Neural Networks (CNN)



# Convolutional Neural Networks (CNN)



# Convolutional Neural Networks (CNN)



# Convolutional Neural Networks (CNN)

Filters:

Pre-trained edge detection
(transfer learning)

Learned

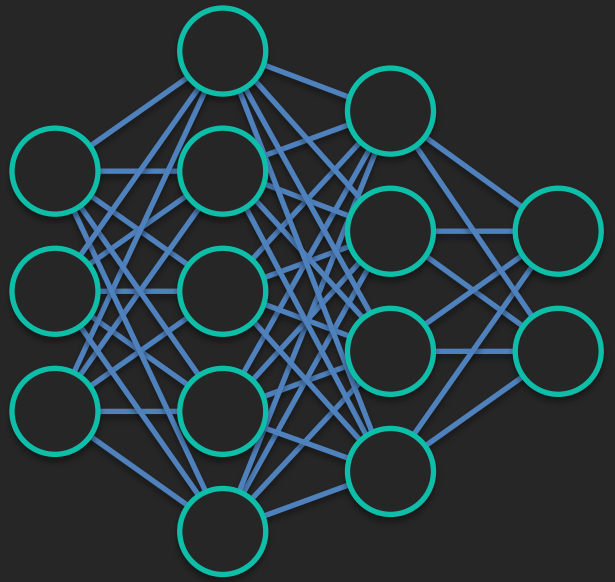
# Recurrent Neural Networks (RNN)

## Type

- [x] Supervised
- [ ] Unsupervised
- [ ] Reinforcement

## Example Inference

- [ ] Binary Yes or No
- [ ] Numeric 1, 2, 3
- [ ] Classification
- [x] Other...



# Recurrent Neural Networks (RNN)

## Type

- Supervised
- Unsupervised
- Reinforcement

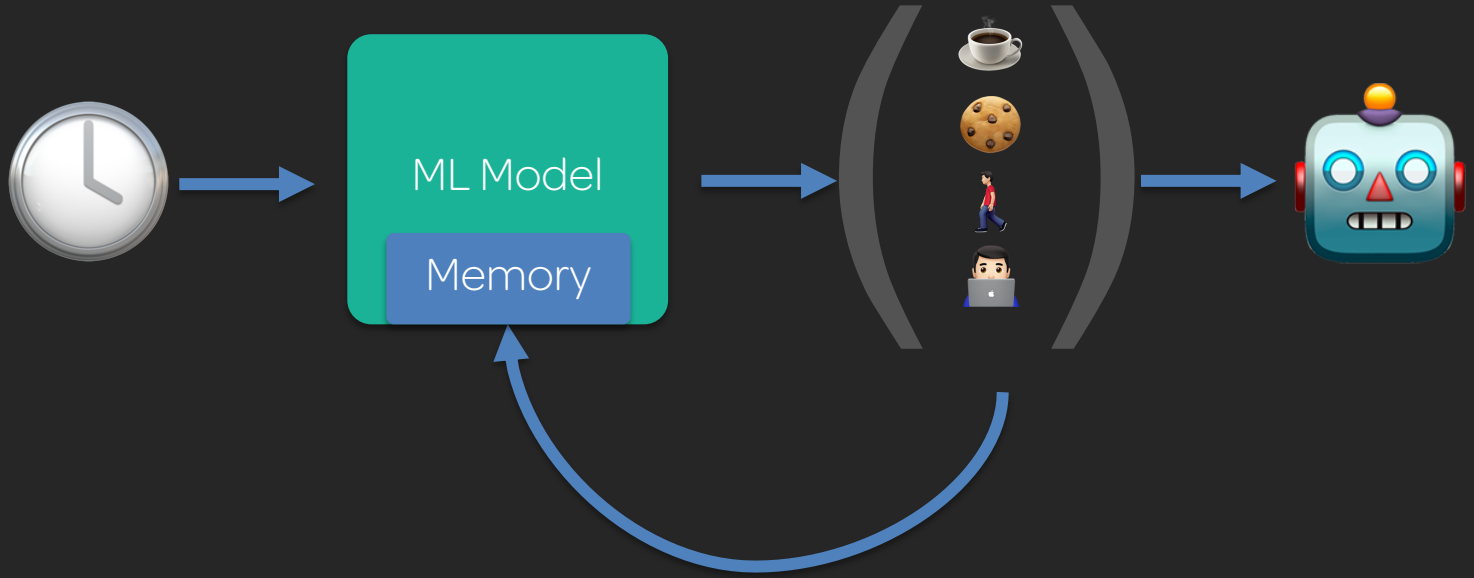
## Example Inference

- Binary Yes or No
- Numeric 1, 2, 3
- Classification
- Other...

## Use Cases

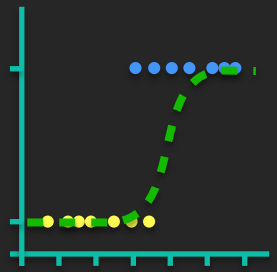

- Stock predictions
- Time series data
- Voice recognition (seq to seq)

# Recurrent Neural Networks (RNN)




Recurrent neural networks (RNN) can remember a bit.
(Long short-term memory (LSTM) can remember a lot.)

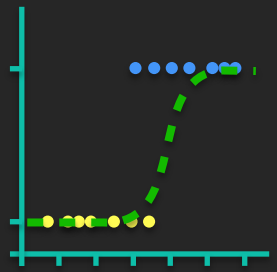
# Confusion Matrix



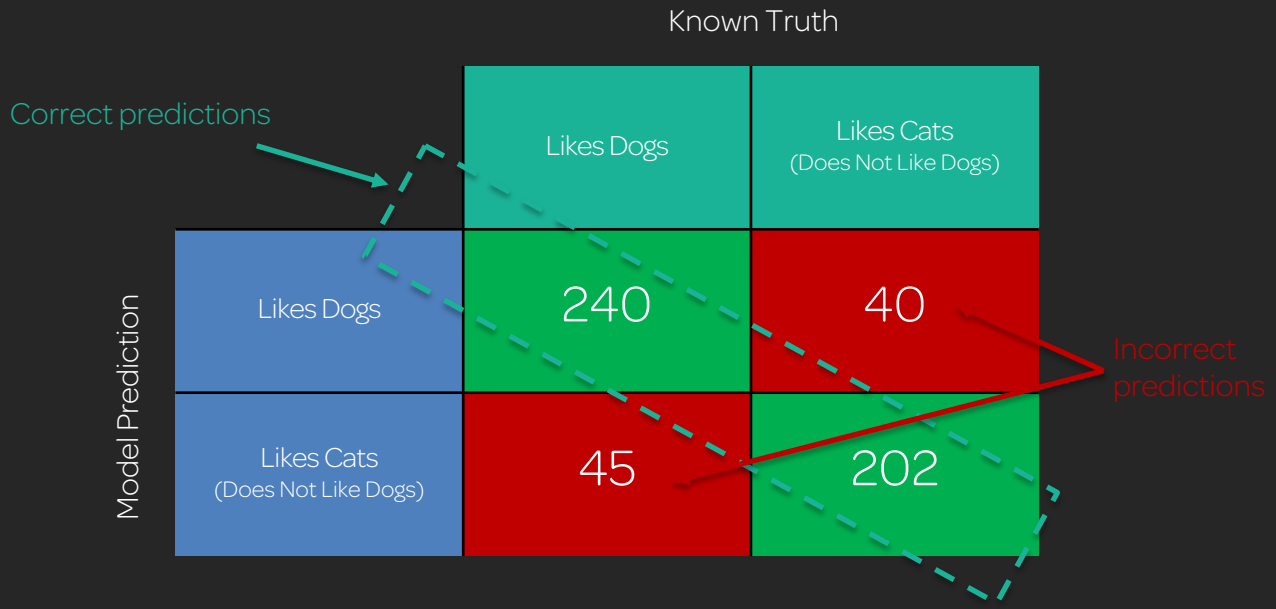
Logistical Regression

| Model Prediction \ Known Truth | Likes Dogs | Likes Cats (Does Not Like Dogs) |
|---|---|---|
| Likes Dogs | TRUE POSITIVES | FALSE POSITIVES |
| Likes Cats (Does Not Like Dogs) | FALSE NEGATIVES | TRUE NEGATIVES |

# Confusion Matrix



Logistical Regression



# Sensitivity and Specificity

## Sensitivity

True Positive Rate (TPR)
Recall

The number of correct positives out of the **actual positive** results.

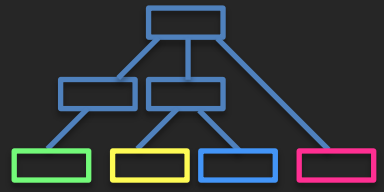
## Specificity

True Negative Rate (TNR)

The number of correct positives out of the **predicted positive** results.

# Sensitivity and Specificity

$$\text{Sensitivity} = \frac{\text{True Positives}}{\text{True Positives} + \text{False Negatives}}$$



Decision Trees

$$\text{Specificity} = \frac{\text{True Negatives}}{\text{True Negatives} + \text{False Positives}}$$

Known Truth

| Model Prediction | Likes Dogs | Likes Cats (Does Not Like Dogs) |
|---|---|---|
| Likes Dogs | 235 | 45 |
| Likes Cats (Does Not Like Dogs) | 37 | 210 |

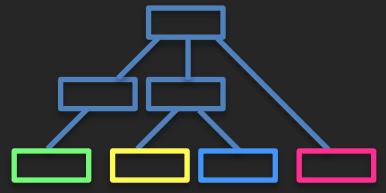
# Sensitivity and Specificity

$$\text{Sensitivity} = \frac{235}{235 + 37}$$

$$= 0.864$$

$$\text{Specificity} = \frac{210}{210 + 45}$$

$$= 0.824$$



Decision Trees

| Model Prediction \ Known Truth | Likes Dogs | Likes Cats (Does Not Like Dogs) |
|---|---|---|
| Likes Dogs | 235 | 45 |
| Likes Cats (Does Not Like Dogs) | 37 | 210 |

# Sensitivity

## Meet Rob

Rob works in a bank.
Keeping money safe is his priority.
He is SENSITIVE. Just look at him. ☺

## Rob's Problem

Rob's bank wants to implement a ML model to detect fraudulent activity.

Catching fraud is more important than falsely identifying fraud.

Or

False positives are acceptable as long as ALL positives are found.

## Rob's Solution

Rob implements an ML model with high sensitivity.

A model with sensitivity at or as close to 1 as possible will catch all fraud, but may catch non-fraud activity too.

Rob's managers are happy, and they give Rob a hug.

# Specificity

## Meet Sam

Sam is a smart kid.
Sam likes to watch videos online and is super specific about which ones and when.

## Our Problem

We want to create an ML model to classify video content so Sam never watches inappropriate grown-up videos.

All content flagged as being suitable for kids MUST be suitable for kids.

Or

False positives are unacceptable; it's better to have false negatives.

## Our Solution

We implement an ML model with high specificity. When the model has a specificity of 1, all the content flagged as appropriate for kids will be correct.

Some content suitable for kids will be incorrectly flagged as inappropriate, but that's ok.

Sam watches her iPad until dinner.

# Accuracy and Precision

## Accuracy

The proportion of all predictions that were correctly identified.

A model that produces completely correct predictions has an accuracy of 1.0.
(But watch out for overfitting!)

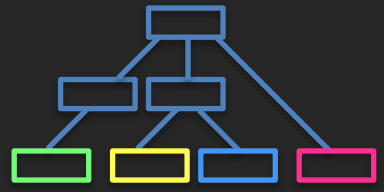
## Precision

The proportion of actual positives that were correctly identified.

A model that produces no false positives has a precision of 1.0.

# Accuracy and Precision

$$\text{Accuracy} = \frac{\text{True Positives} + \text{True Negatives}}{\text{Total}}$$



Decision Trees

$$\text{Precision} = \frac{\text{True Positives}}{\text{True Positives} + \text{False Positives}}$$

| Model Prediction \ Known Truth | Likes Dogs | Likes Cats (Does Not Like Dogs) |
|---|---|---|
| Likes Dogs | 235 | 45 |
| Likes Cats (Does Not Like Dogs) | 37 | 210 |

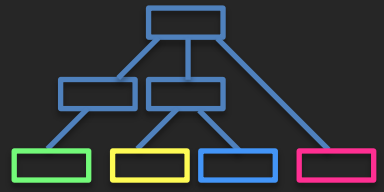
# Accuracy and Precision

$$\text{Accuracy} = \frac{235 + 210}{235 + 45 + 37 + 210}$$

$$= 0.844$$

$$\text{Precision} = \frac{235}{235 + 45}$$

$$= 0.839$$

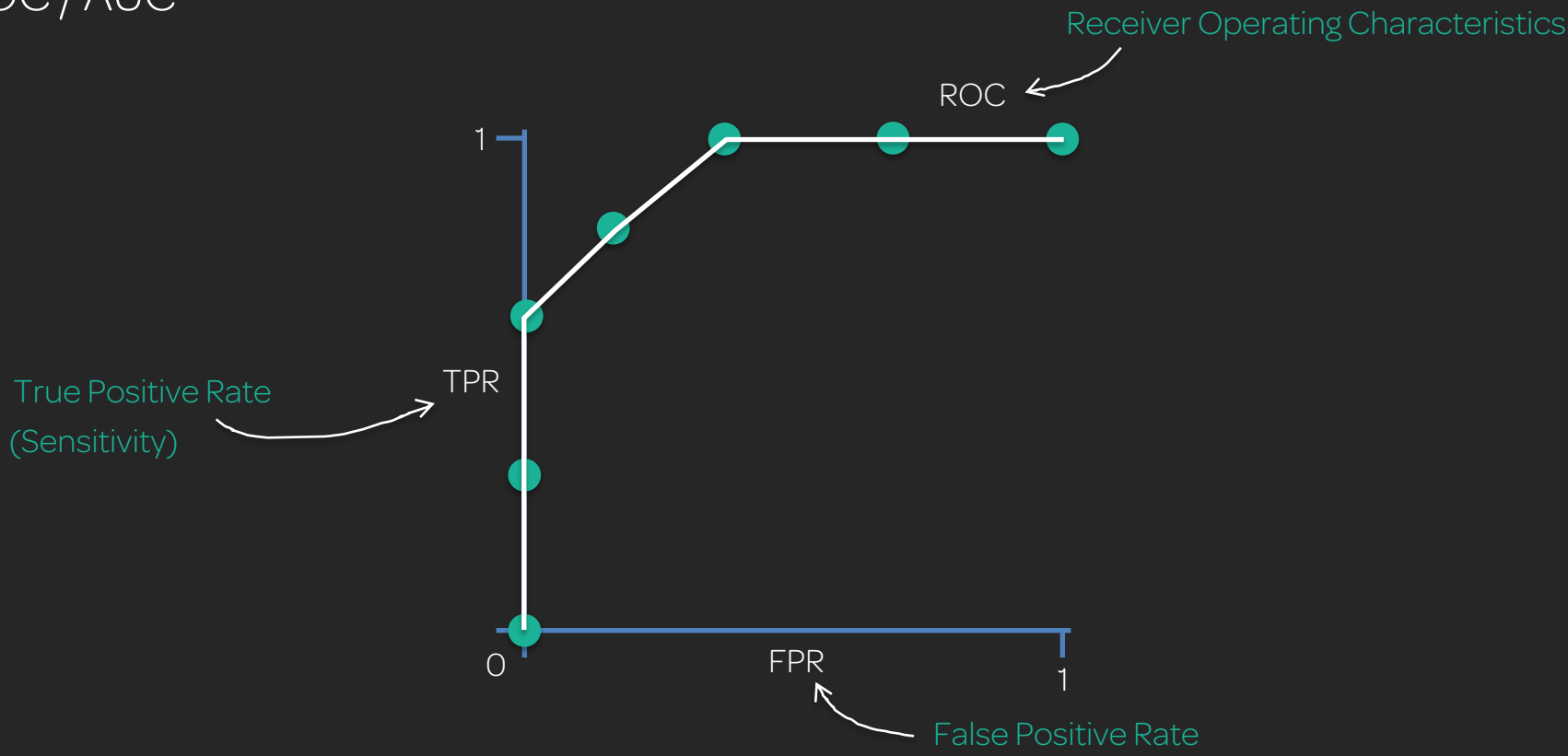


Decision Trees

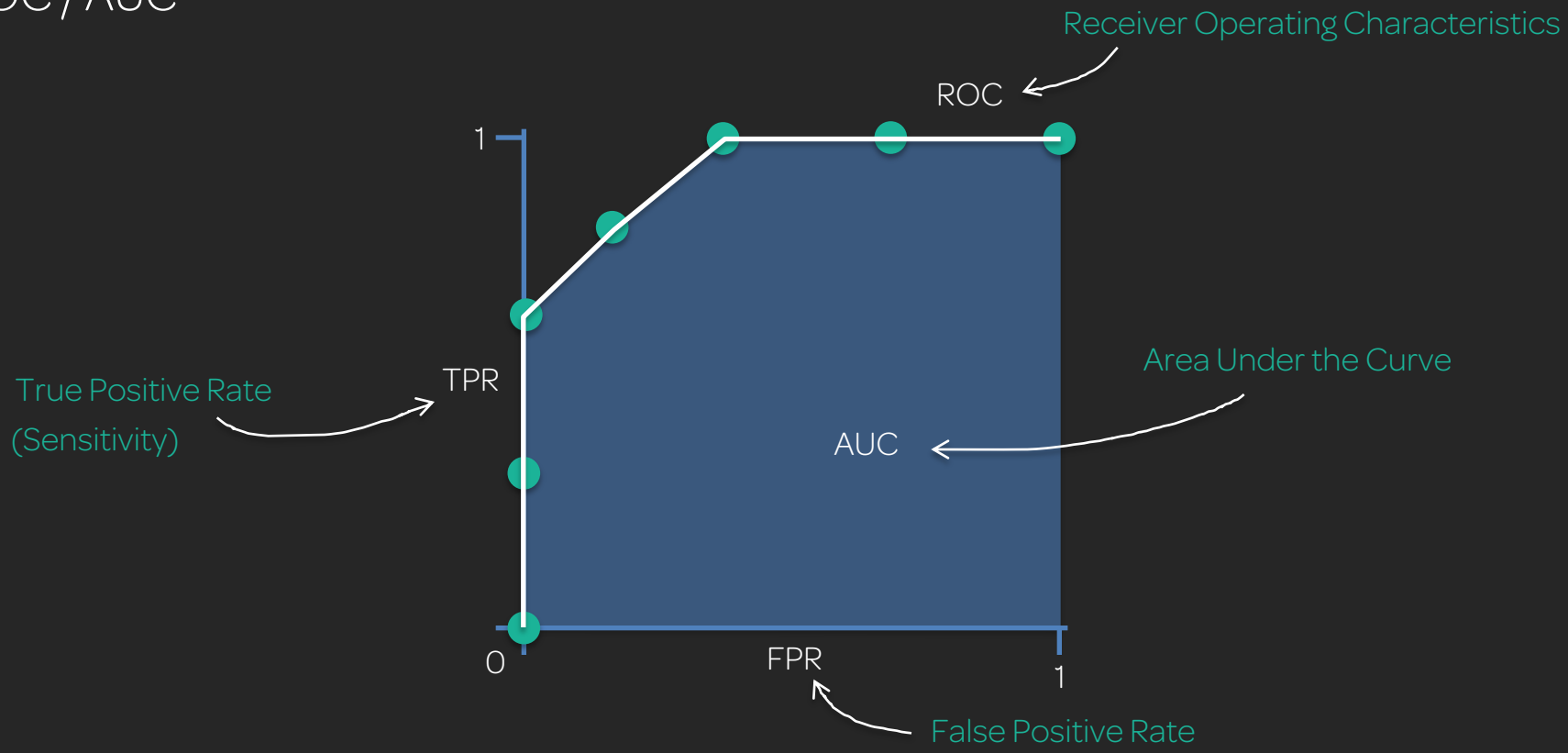
Known Truth

| Model Prediction | Likes Dogs | Likes Cats (Does Not Like Dogs) |
|---|---|---|
| Likes Dogs | 235 | 45 |
| Likes Cats (Does Not Like Dogs) | 37 | 210 |

# ROC / AUC


Receiver Operating Characteristics
ROC
1
TPR
True Positive Rate
(Sensitivity)
0
FPR
1
False Positive Rate

# ROC / AUC



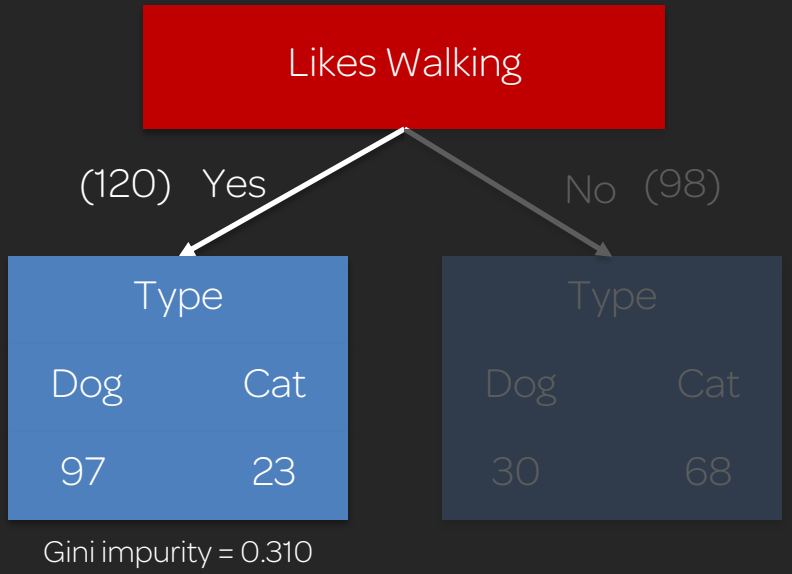

(Area Under the Receiver Operating Characteristic)

# Gini Impurity

Gini impurity = 1 – (probability of dog)$^2$ - (probability of cat)$^2$

$$= 1 - \left(\frac{97}{120}\right)^2 - \left(\frac{23}{120}\right)^2$$

$$= 0.310$$

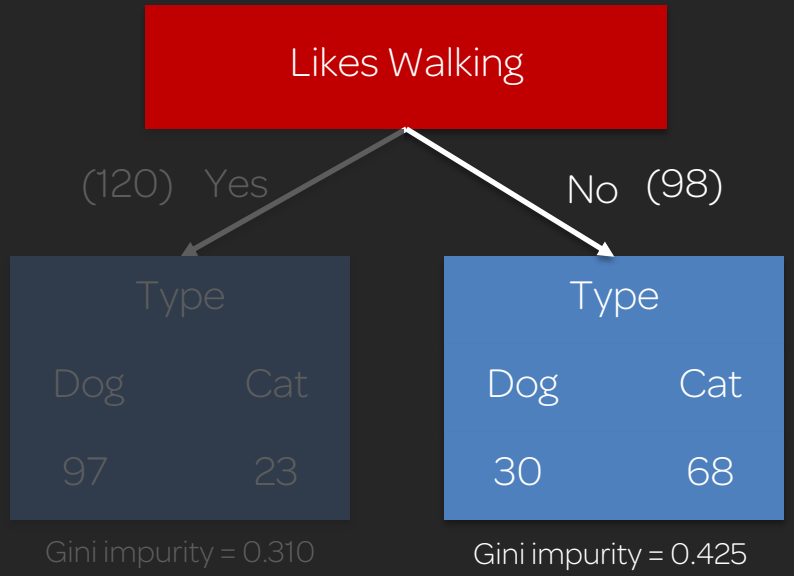


# Gini Impurity

Gini impurity = 1 – (probability of dog)$^2$ - (probability of cat)$^2$

$$= 1 - \left(\frac{30}{98}\right)^2 - \left(\frac{68}{98}\right)^2$$

$$= 0.425$$



# Gini Impurity

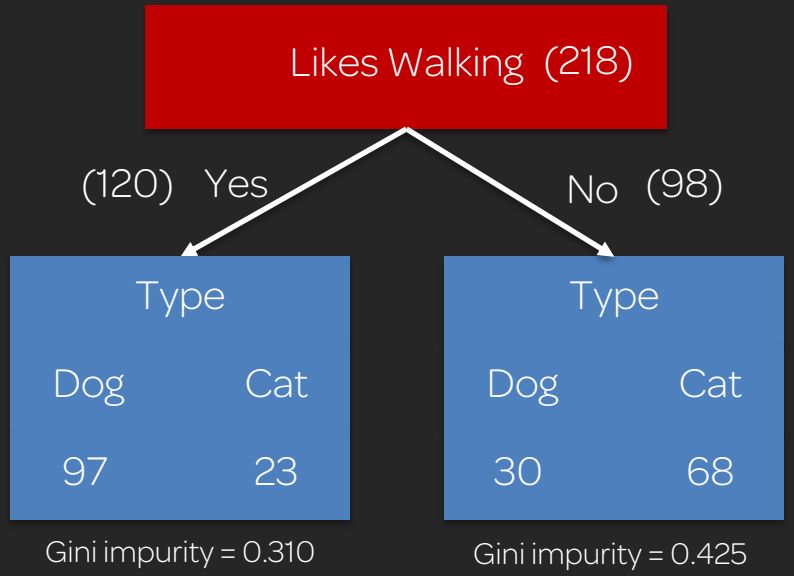
Weighted average

$$= \left(\frac{\text{Yes likes walking}}{\text{All people}}\right) \times \text{Gini Impurity}$$

$$+ \left(\frac{\text{No likes walking}}{\text{All people}}\right) \times \text{Gini Impurity}$$

$$= \left(\frac{120}{218}\right) \times 0.310$$

$$+ \left(\frac{98}{218}\right) \times 0.425$$



# Gini Impurity

| Likes Walking | Likes Running | Favorite Color | Type |
|---|---|---|---|
| No | Yes | Green | Dog |
| No | No | Blue | Cat |
| Yes | Yes | Red | Dog |
| Yes | No | Green | Cat |
| Yes | No | Green | Dog |
| Yes | Yes | Blue | Dog |
| No | No | Red | Cat |
| ... | ... | ... | ... |

Gini Impurity Values:

| Feature | GI |
|---|---|
| Likes Walking | 0.362 |
| Likes Running | 0.384 |
| Favorite Color | 0.371 |

Likes Walking has the lowest weighted Gini impurity, so it best separates people who like dogs over cats.

We will use Likes Walking as our root node.

# F1 Score

F1 Score = Combination of Recall and Precision

$$= \frac{2}{\frac{1}{\text{Recall}} + \frac{1}{\text{Precision}}}$$

$$= \frac{\text{Recall} \times \text{Precision}}{\text{Recall} + \text{Precision}} \times 2$$

# F1 Score

## Sensitivity / Recall

The number of correct positives out of the **actual positive** results.

$$\frac{\text{True Positives}}{\text{True Positives + False Negatives}}$$

## Precision

The proportion of actual positives that were correctly identified.

$$\frac{\text{True Positives}}{\text{True Positives + False Positives}}$$

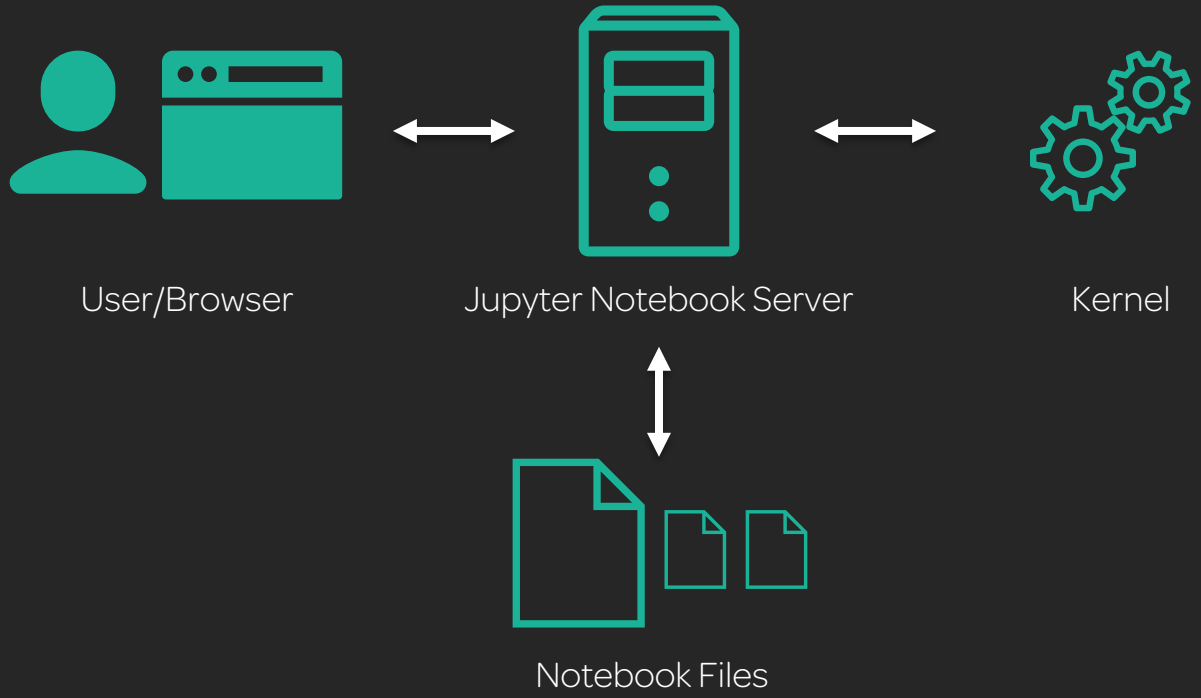
# Jupyter Notebooks


- Interactive development environment (IDE).
- Web-based.

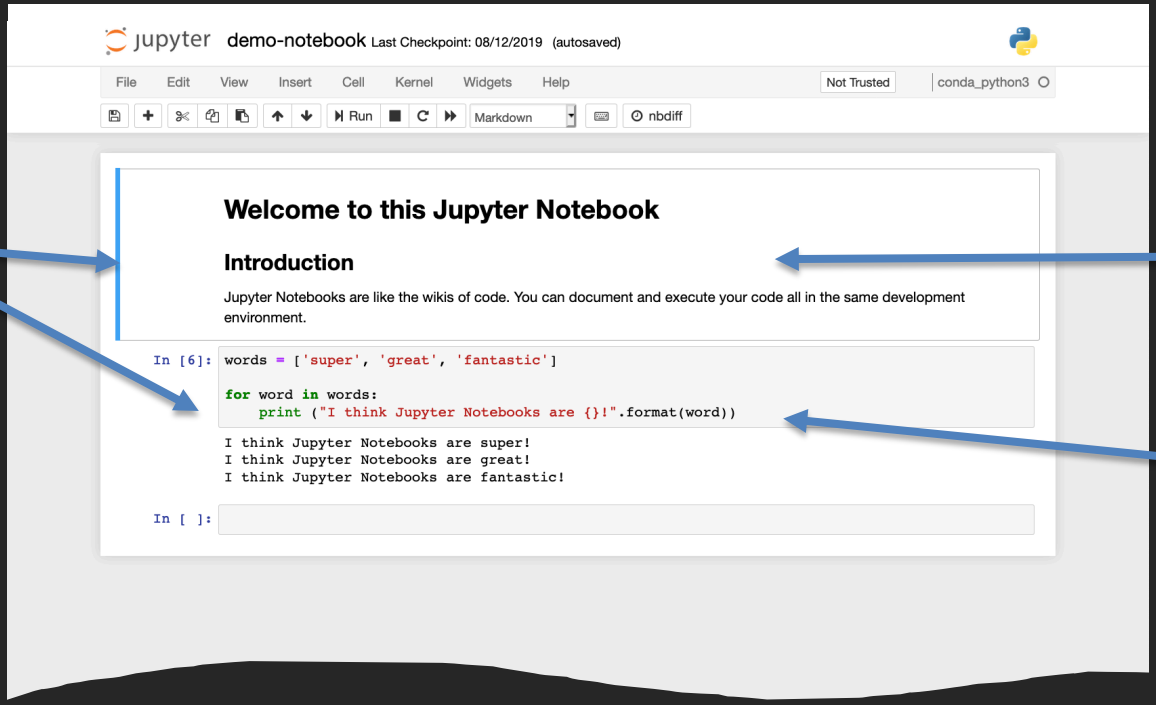
- Think wiki for code.
- Combines executable code with documentation.
- Designed to be shared.

- Used extensively for data science, ML, and DL.
- Supports many languages, including Python.

# Jupyter Notebooks



# Jupyter Notebooks

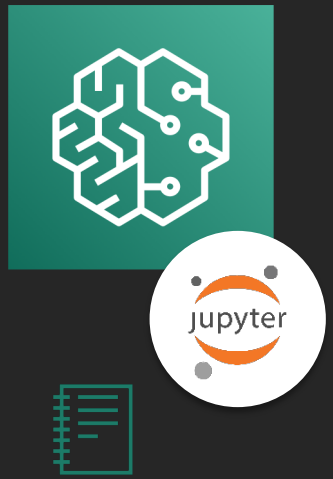



Cells

Rich Text (Markdown)

Executable code

# Jupyter Notebooks




## Amazon SageMaker

- The home of Jupyter Notebooks in AWS.
- Fully-managed notebook instances.
- Multiple kernel options.
- Used by AWS to demo SageMaker algorithms.

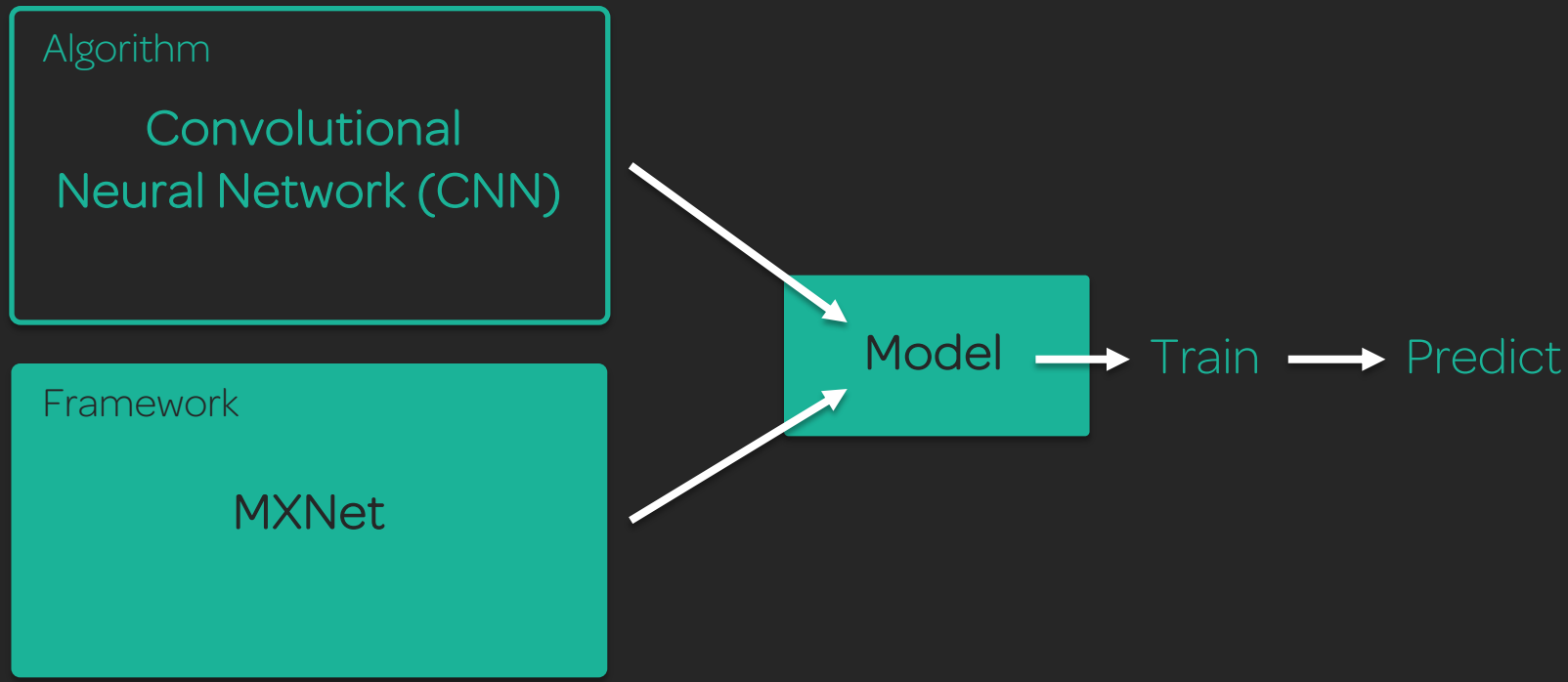
- For more on this, see the AWS SageMaker section of this course.

# ML and DL Frameworks






# ML and DL Frameworks



# Simple Storage Services - S3

## General

- Cost effective storage for large amounts of data.
- Structured or unstructured.
- Data lake...

# Simple Storage Services - S3

## Data Lake

- Destination for all data sets.
- Structured data
  - CSV
  - JSON
- Unstructured data
  - Text files
  - Images

# Simple Storage Services - S3

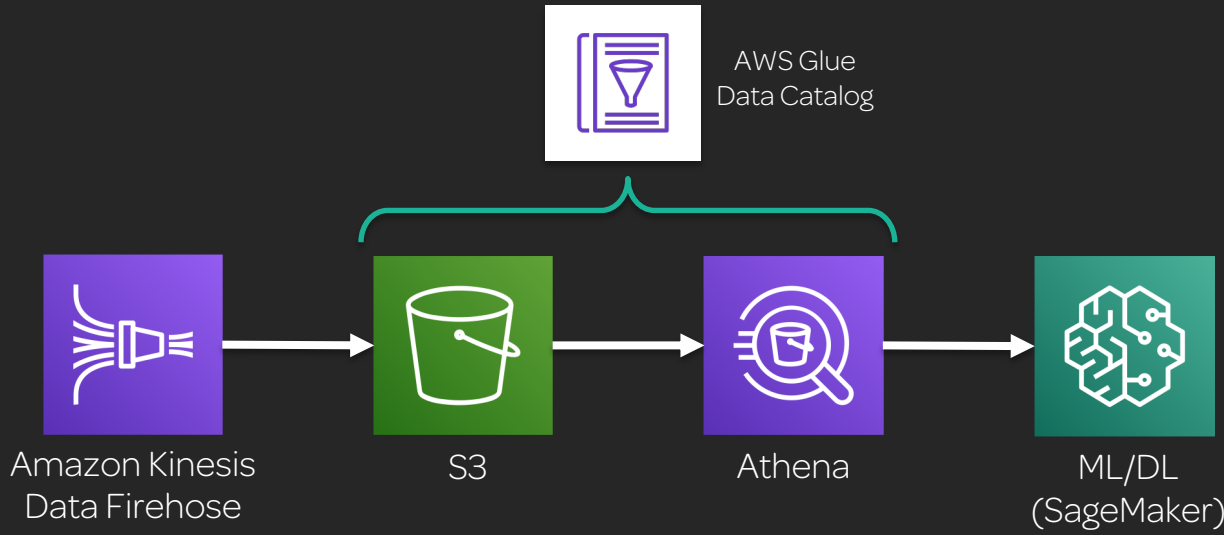
## Data Lake Advantages

- Add data from many sources.
- Define the data schema at the time of analysis.
- Much lower cost than data warehouse solutions.
- Tolerant of low-quality data.

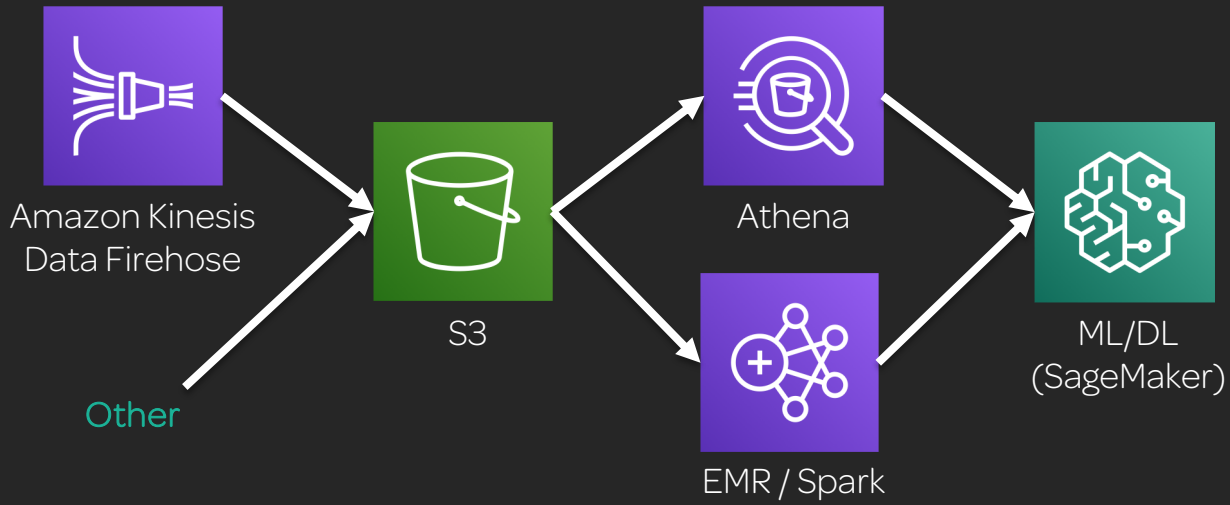
## Date Lake Disadvantages

- Unsuitable for transactional systems.
- Needs cataloguing before analysis.

# Simple Storage Services - S3



# Simple Storage Services - S3



# Simple Storage Services - S3

## Security

- IAM Users and Roles
- Bucket policy

## Encryption

- S3 SSE
- S3 KMS

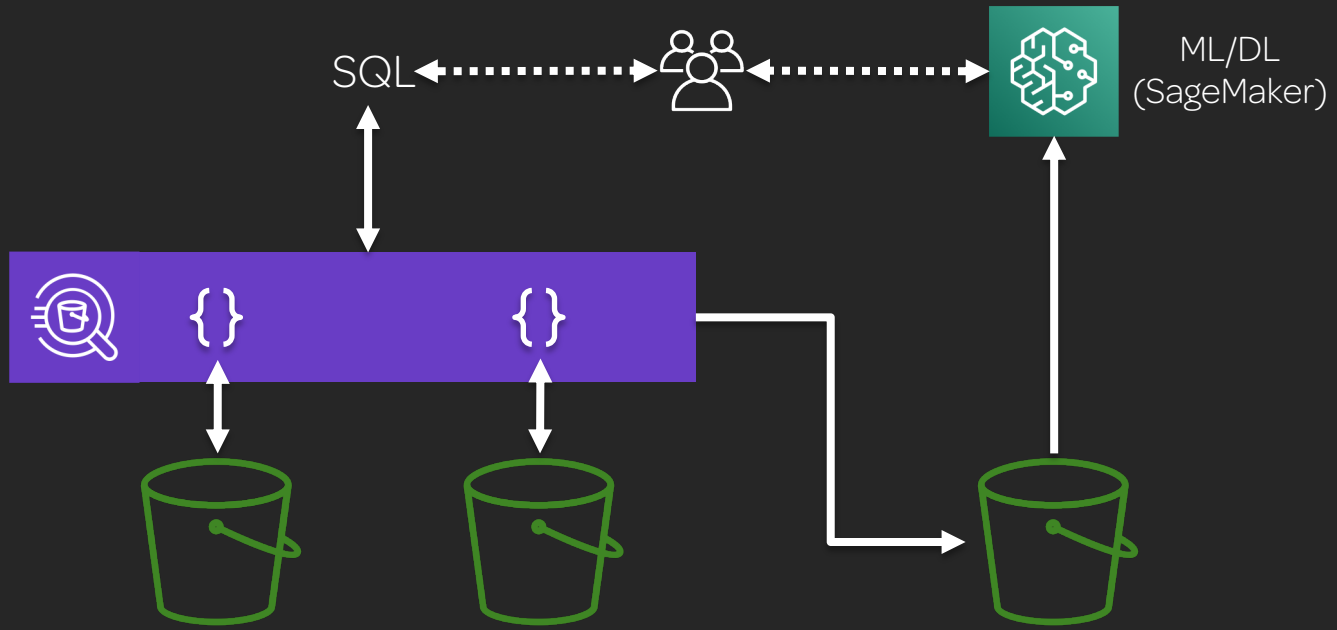
# AWS Glue

- Creates catalogues of data (schemas).
- Performs ETL.
- Some limited ML capabilities.

# Amazon Athena

- Query S3 data with SQL.
- Source data from multiple S3 locations.
- Save outputs to S3.
- Use for data pre-processing ahead of ML.

# Amazon Athena



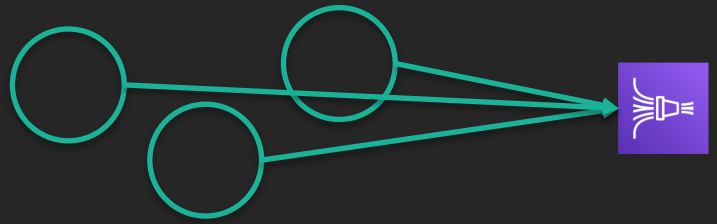
# Amazon QuickSight

- Business Intelligence (BI) tool.
- Visualise data from many sources.
  - Dashboards
  - Email reports
  - Embedded reports

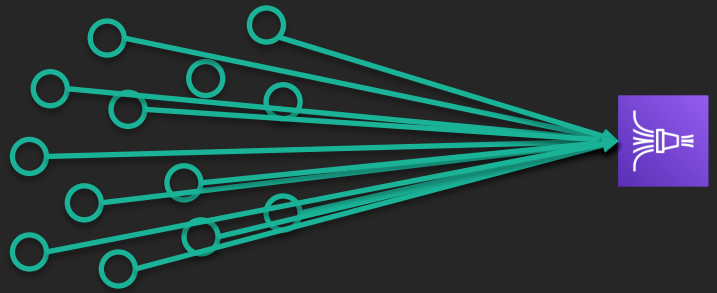
- End-user targeted.

# Amazon Kinesis

- Ingest large scale data.
- Lots of data from a few sources (video).



- Small amount of data from many sources (IoT).



# Amazon Kinesis

Amazon Kinesis Video Streams

Securely stream video from connected devices to AWS for analytics, machine learning (ML), playback, and other processing.

Amazon Kinesis Data Streams

General endpoint for ingesting large amounts of data for processing by:

- Kinesis Data Analytics
- Spark on EMR
- Amazon EC2
- Lambda

Amazon Kinesis Data Firehose

Simple endpoint to stream data into:

- S3
- Redshift
- Elasticsearch
- Splunk

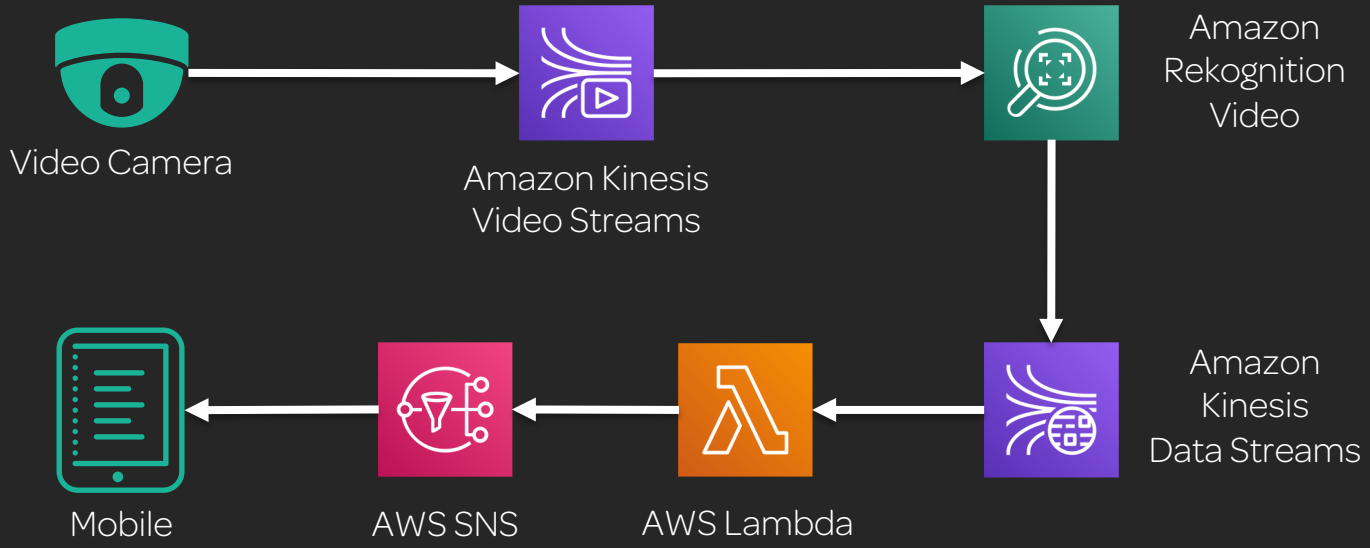
Amazon Kinesis Data Analytics

Process streaming data from Kinesis Streams or Firehose at scale using:

- SQL
- Java libraries

# Amazon Kinesis

# Amazon Kinesis



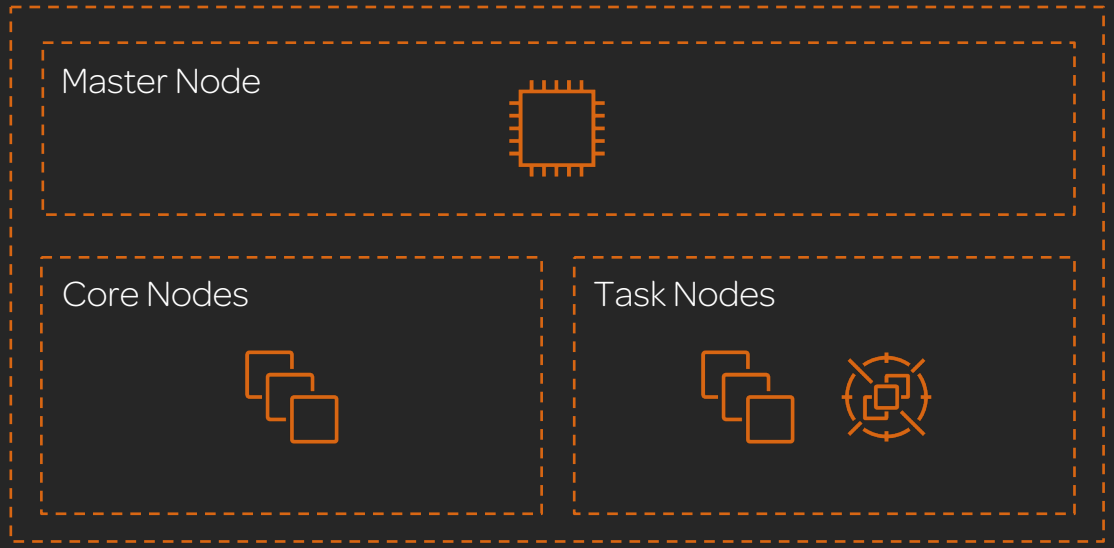
# Amazon EMR with Spark

## Amazon EMR (Elastic Map Reduce)

- Managed service for hosting massively parallel compute tasks.
- Integrates with storage service S3.
- Petabyte scale.
- Uses 'big data' tools:
  - Spark
  - Hadoop
  - HBase
  - ...

# Amazon EMR with Spark

## Amazon EMR (Elastic Map Reduce)



# Amazon EMR with Spark


## Apache Spark

- Fast analytics engine.
- Massively parallel compute tasks.
- Deployed over clusters of resources.

- Variations of Spark run on:
  - Amazon EMR
  - Amazon SageMaker

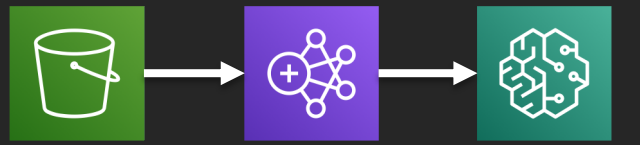
- And Spark ML runs on EMR too.

# Amazon EMR with Spark

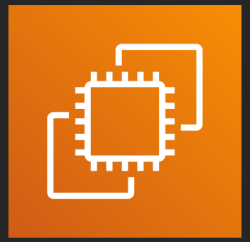
+


## Machine Learning

- Integrates into Amazon SageMaker.
- Performs massive ETL of data into SageMaker.



# AWS EC2 for ML

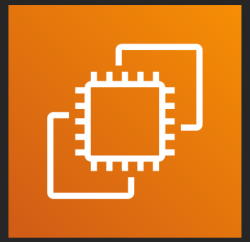


## EC2 Instance Types

- AWS EC2 instance types targeted at ML tasks:
  - Compute Optimized
  - Accelerated Computing (GPU)

- The `ml.*` instances are not available outside of SageMaker

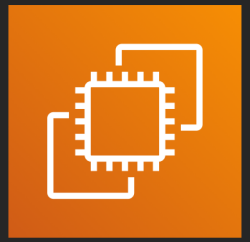
# AWS EC2 for ML



## Amazon Machine Images (AMIs)

- Conda-based Deep Learning AMIs:
  - Libraries:
    - TensorFlow
    - Keras
    - MXNet
    - Gluon
    - PyTorch
    - ...
  - GPU acceleration:
    - CUDA 8 and 9
    - cuDNN 6 and 7
    - NCCL 2.0.5 libraries
    - NVidia Driver 384.81

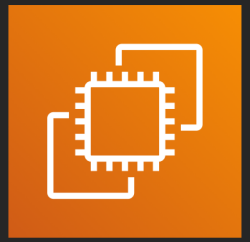
# AWS EC2 for ML



Amazon Machine Images (AMIs)

- Deep Learning Base AMIs:
  - Libraries:
    - 'none'
  - GPU acceleration:
    - CUDA 8 and 9
    - cuDNN 6 and 7
    - NCCL 2.0.5 libraries
    - NVidia Driver 384.81
    - ...

# AWS EC2 for ML



## EC2 Instance Type Limits

- Brand new account = no ML for you!
- Service limit increases take days.

# AWS Machine Learning (the Service)


Thank you for your interest in Amazon Machine Learning. AmazonML is no longer available to new customers. Please consider using Amazon SageMaker. Click here to go to the SageMaker home page.

- AWS ML was the forerunner to SageMaker services.
- It is no longer available; only existing projects are supported.
- It is known to be mentioned in the exam, but its not understood to ever be the correct answer.

# Amazon Rekognition

- Image and video analysis
- Pre-trained deep learning
- Simple API

- Image moderation
- Facial analysis
- Celebrity recognition
- Face comparison
- Text in image

# Amazon Rekognition

## Use Cases

- Create a filter to prevent inappropriate images being sent via a messaging platform. This can include nudity or offensive text.
- Enhance metadata catalog of an image library to include the number of people in each image.
- Scan an image library to detect instances of famous people.

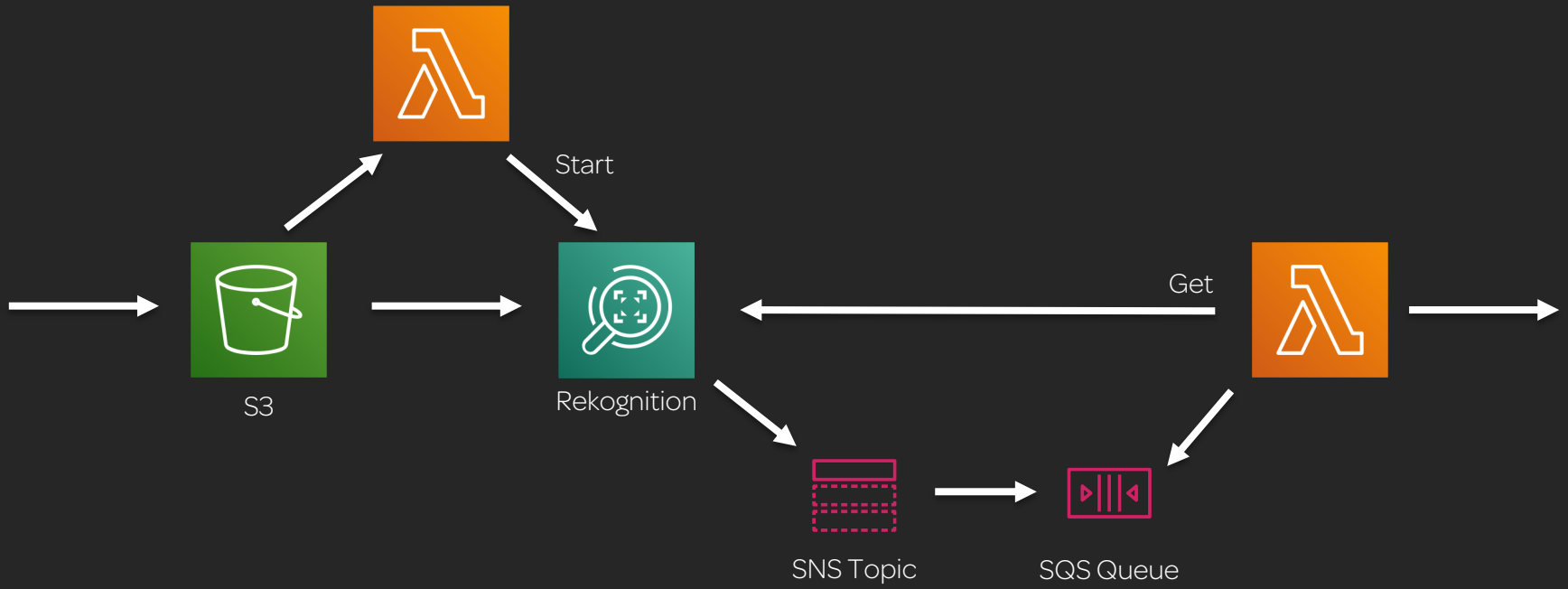
# Amazon Rekognition Video

- Image and video analysis
- Pre-trained deep learning
- Simple API

- Stored videos
- Streaming videos

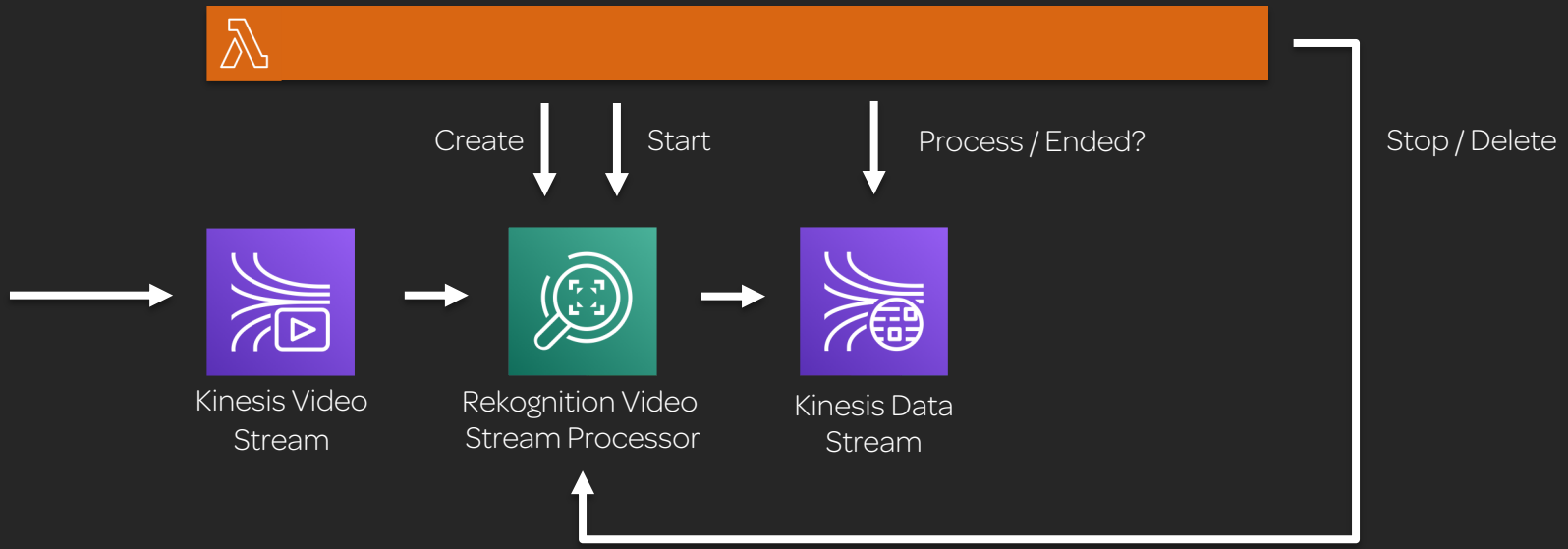
# Amazon Rekognition Video

## Example architecture for stored video.



# Amazon Rekognition Video

## Example architecture for streaming video.



# Amazon Rekognition Video

## Use Cases

- Detect people of interest in a live video stream for a public safety application.
- Create a metadata catalog for stock video footage library.
- Detect offensive content within videos uploaded to a social media platform.

# Amazon Polly

- Text to speech (TTS)
- Deep learning powered
- Simple API

- Language
- Female or male
- Custom lexicons

# Amazon Polly

## Use Cases

- Create accessibility tools to “read” web content.
- Provide automatically generated announcements via a public address (PA) system.
- Create an automated voice response (AVR) solution for a telephony system (including Amazon Connect).

# Amazon Transcribe

- Automatic speech recognition (ASR)
- Pre-trained deep learning
- Simple API

- Language
- Custom vocabulary

# Amazon Transcribe

## Use Cases

- Create a call center monitoring solution that integrates with other services to analyze caller sentiment.
- Create a solution to enable text search of media with spoken words.
- Provide a closed captioning solution for online video training.

# Amazon Translate

- Text translation
- Pre-trained deep learning
- Simple API

- Batch or real-time
- Languages
- Custom terminology

# Amazon Translate

## Use Cases

- Enhance an online customer chat application to translate conversations in real-time.
- Batch translate documents within a multilingual company.
- Create a news publishing solution to convert posted stories to multiple languages.

# Amazon Comprehend

- Text analysis
- Natural language processing (NLP)
- Pre-trained deep learning
- Simple API

# Amazon Comprehend

- Keyphrase extraction
- Sentiment analysis
- Syntax analysis
- Entity recognition
- Medical Named Entity and Relationship Extraction (NERe)
- Custom entities
- Language detection
- Custom classification
- Topic modeling
- Multiple language support

# Amazon Comprehend

## Use Cases

- Perform customer sentiment analysis on inbound messages to support the system.
- Create a system to label unstructured (clinical) data to assist in research and analysis.
- Determine the topics from transcribed audio recordings of company meetings.

# Amazon Lex

- Conversation interface service
- Think “Alexa” or “chatbots”
- Voice enabled or text

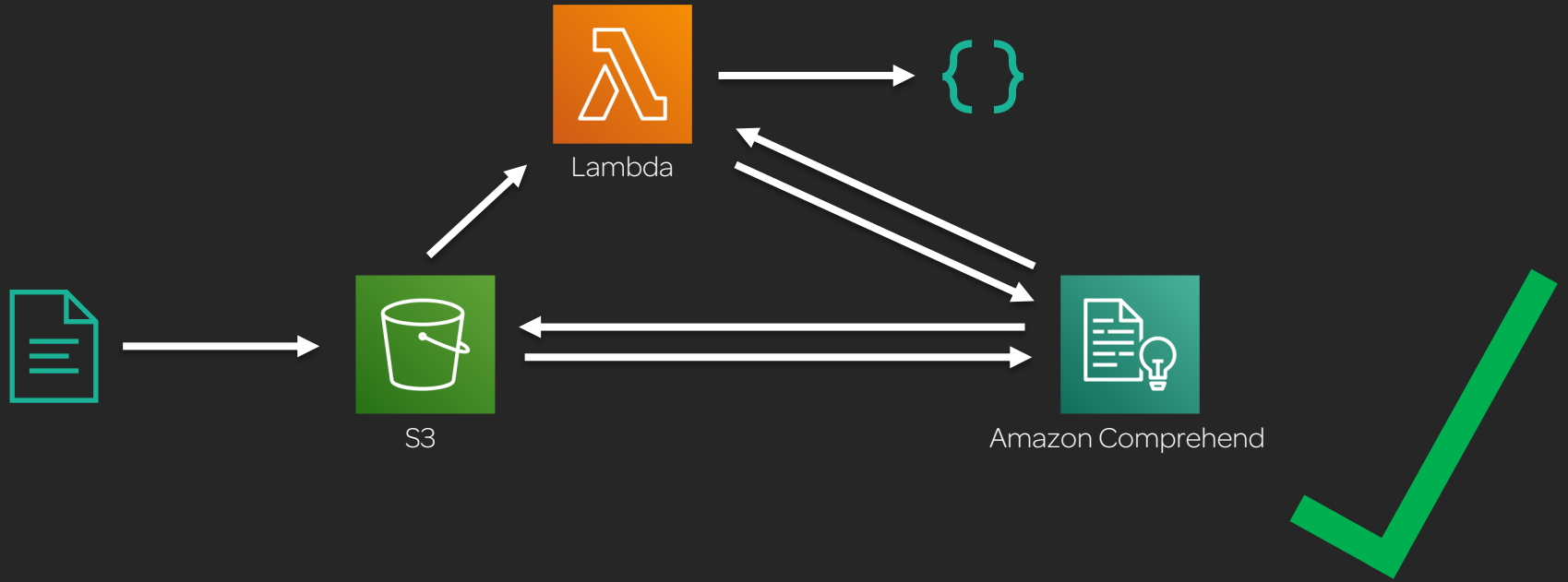
- Automatic speech recognition (ASR)
- Natural language understanding (NLU)

# Amazon Lex

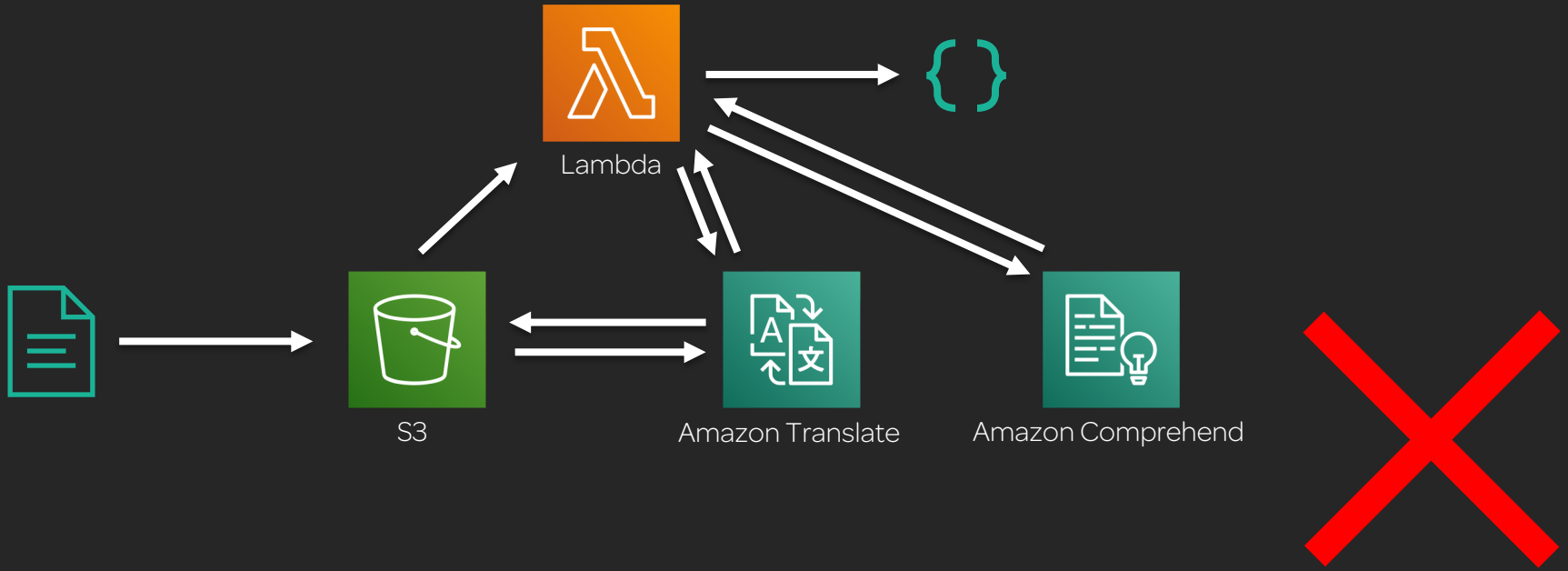
## Use Cases

- Create a chatbot that triages customer support requests directly on the product page of a website.
- Create an automated receptionist that directs people as they enter a building.
- Provide an interactive voice interface to [...] application.

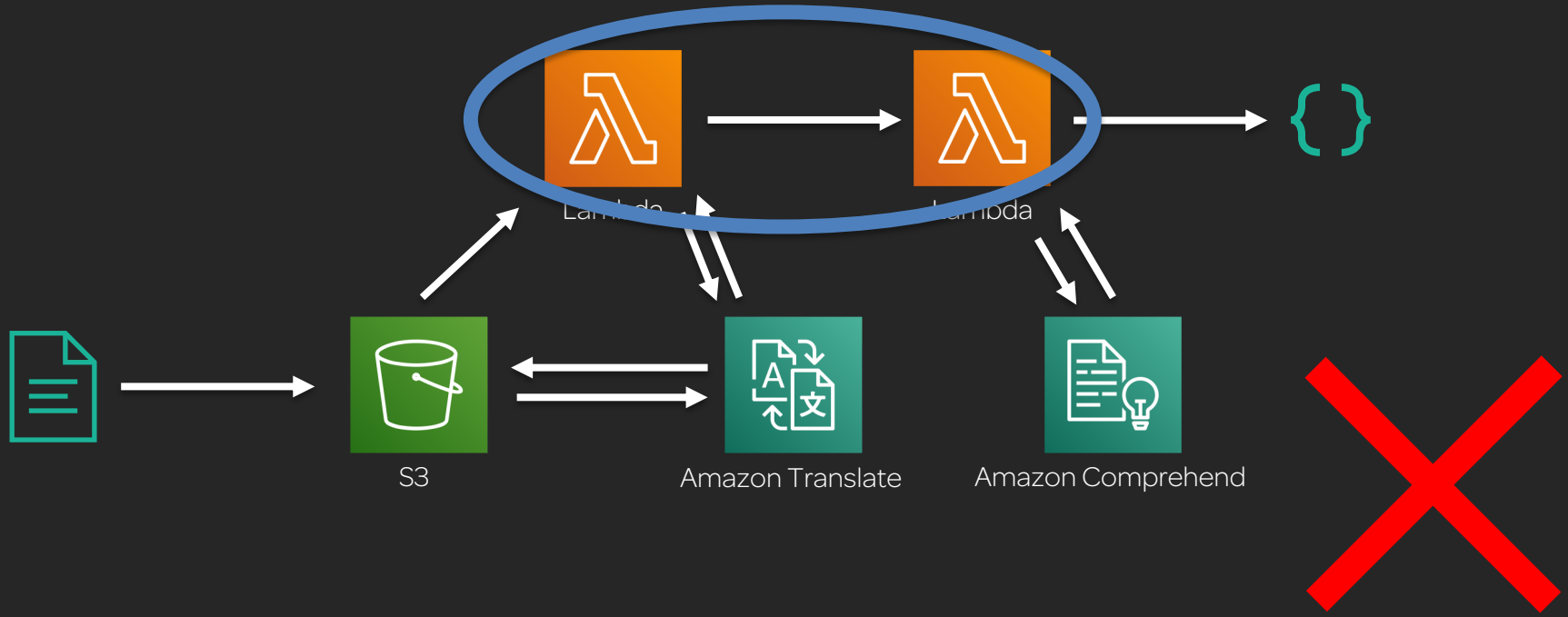
# Amazon Service Chaining



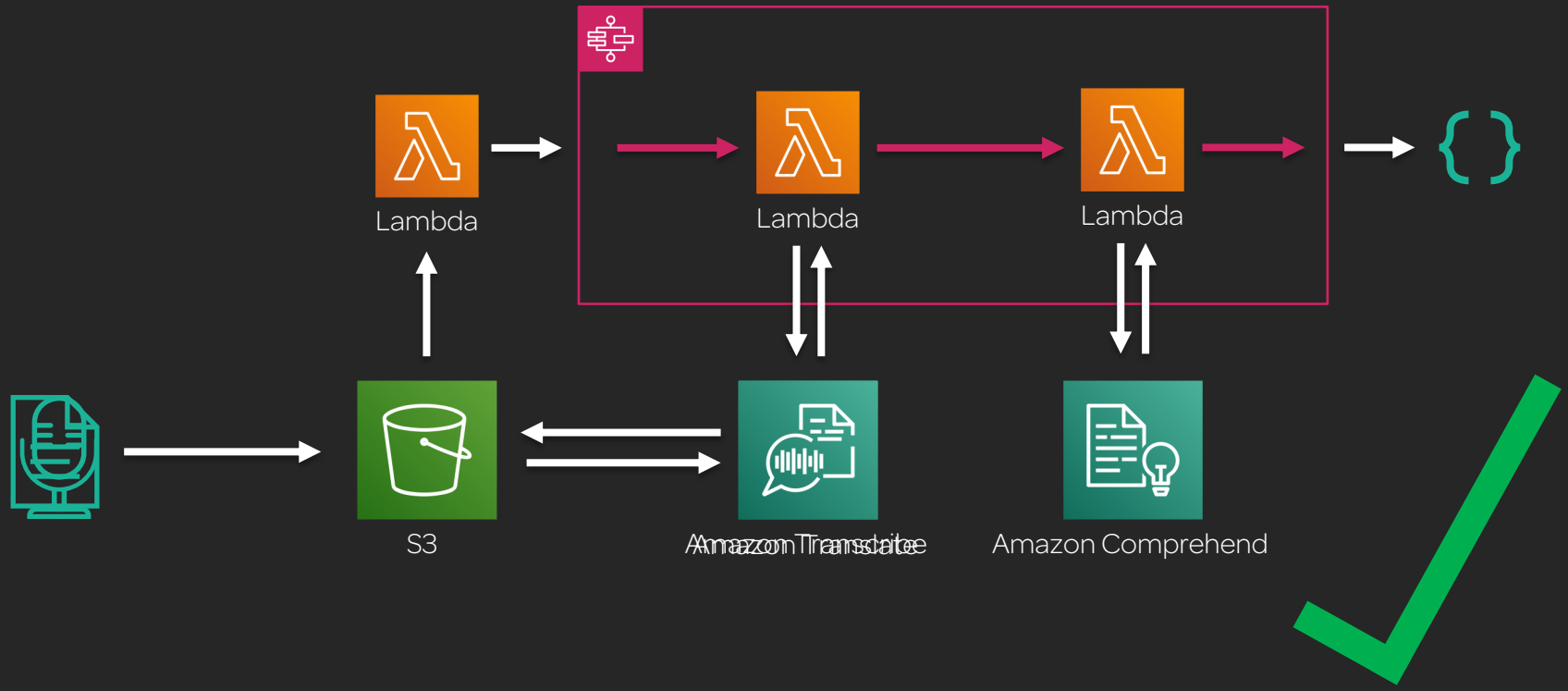
# Amazon Service Chaining



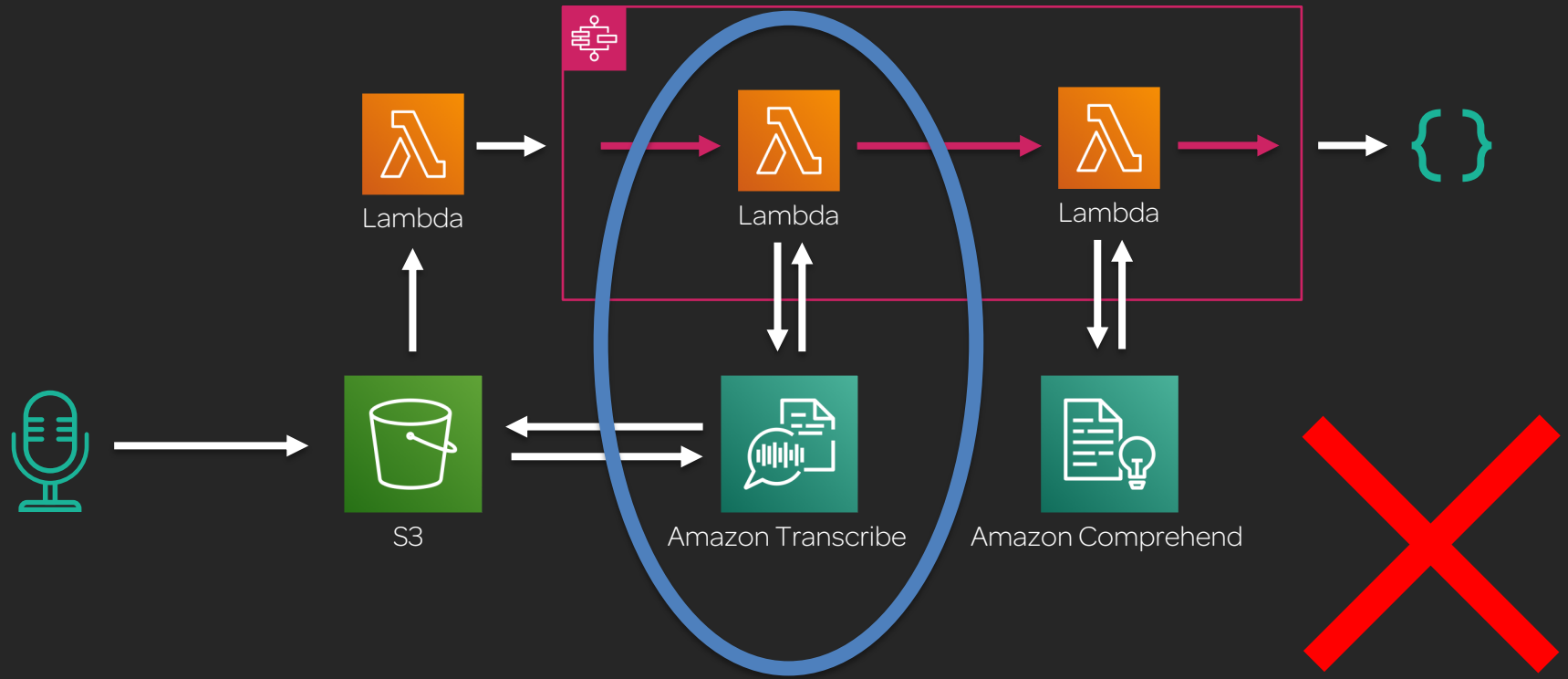
# Amazon Service Chaining



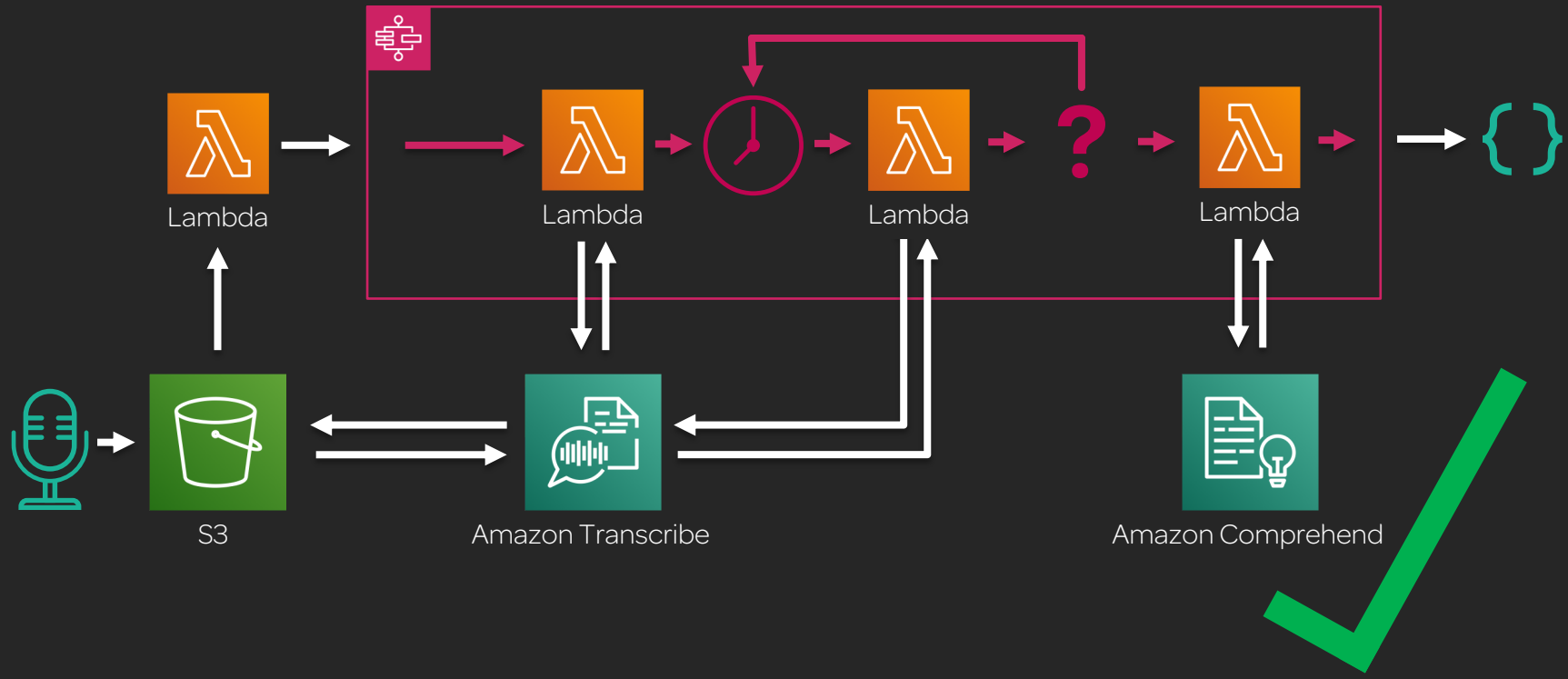
# Amazon Service Chaining



# Amazon Service Chaining



# Amazon Service Chaining



# Amazon Service Chaining

Amazon Translate

Amazon Polly

Lambda

AWS Step Functions

Amazon Transcribe

S3

Amazon Rekognition

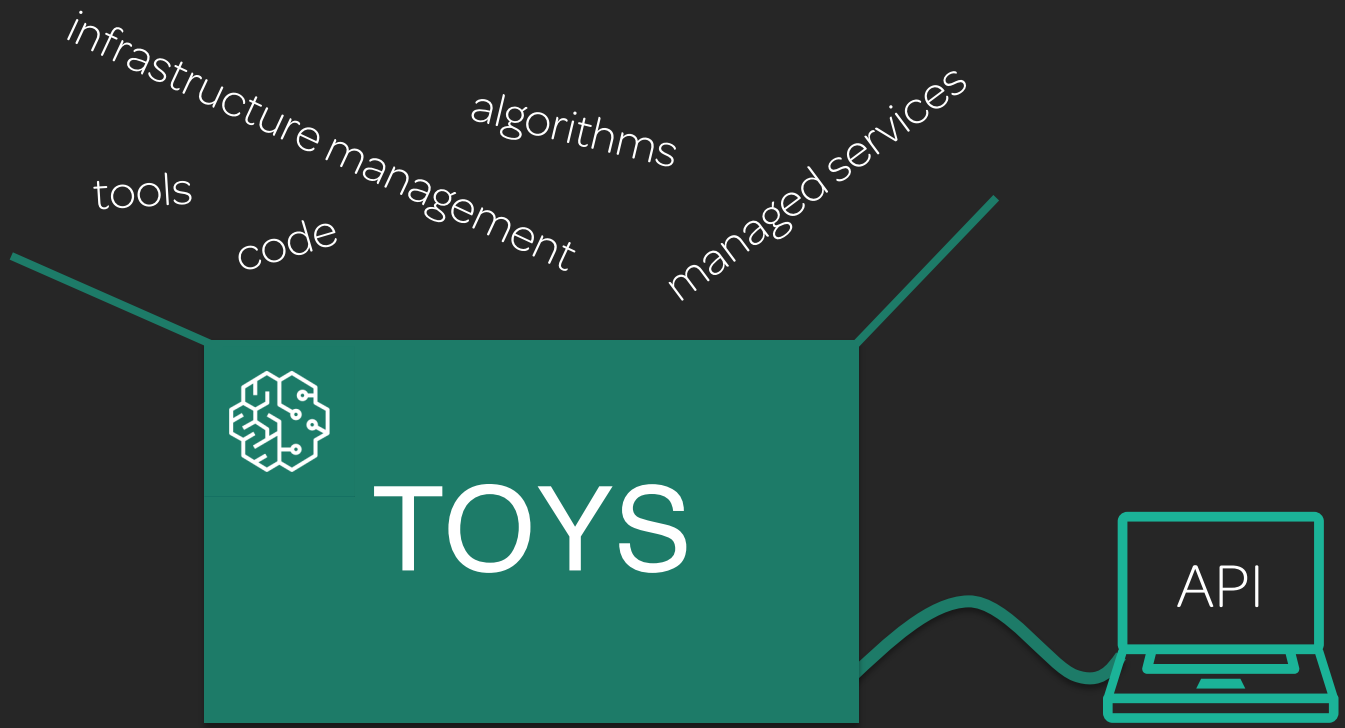
Amazon Comprehend

# What is Amazon SageMaker

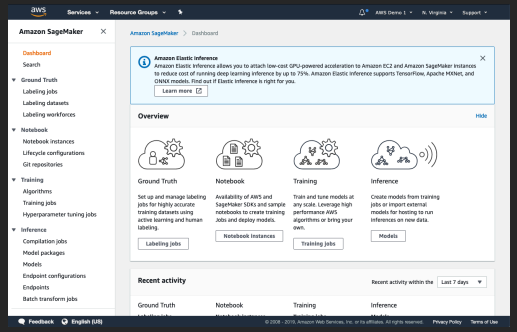
"Amazon SageMaker provides every developer and data scientist with the ability to build, train, and deploy machine learning models quickly. Amazon SageMaker is a fully-managed service that covers the entire machine learning workflow to label and prepare your data, choose an algorithm, train the model, tune and optimize it for deployment, make predictions, and take action."

- Amazon Web Services

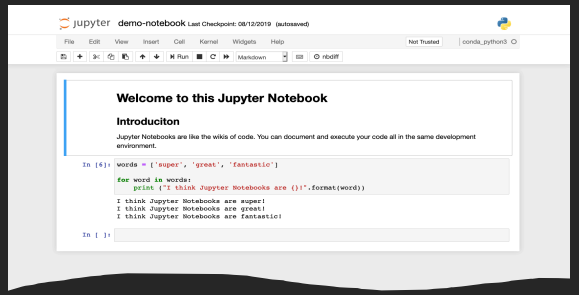
# What is Amazon SageMaker



# Control – Console, SDK, and Notebooks




AWS Console


Jupyter Notebooks

# The Three Stages of SageMaker

## Build

- Preprocessing
- Ground Truth
- Notebooks

## Train

- Built-in algorithms
- Hyperparameter tuning
- Notebooks
- Infrastructure

## Deploy

- Realtime
- Batch
- Notebooks
- Infrastructure
- Neo

# Data Preprocessing

Visualize

Engineer

Explore

Synthesize

Convert

Structure

Split

# Data Preprocessing

## SageMaker Components

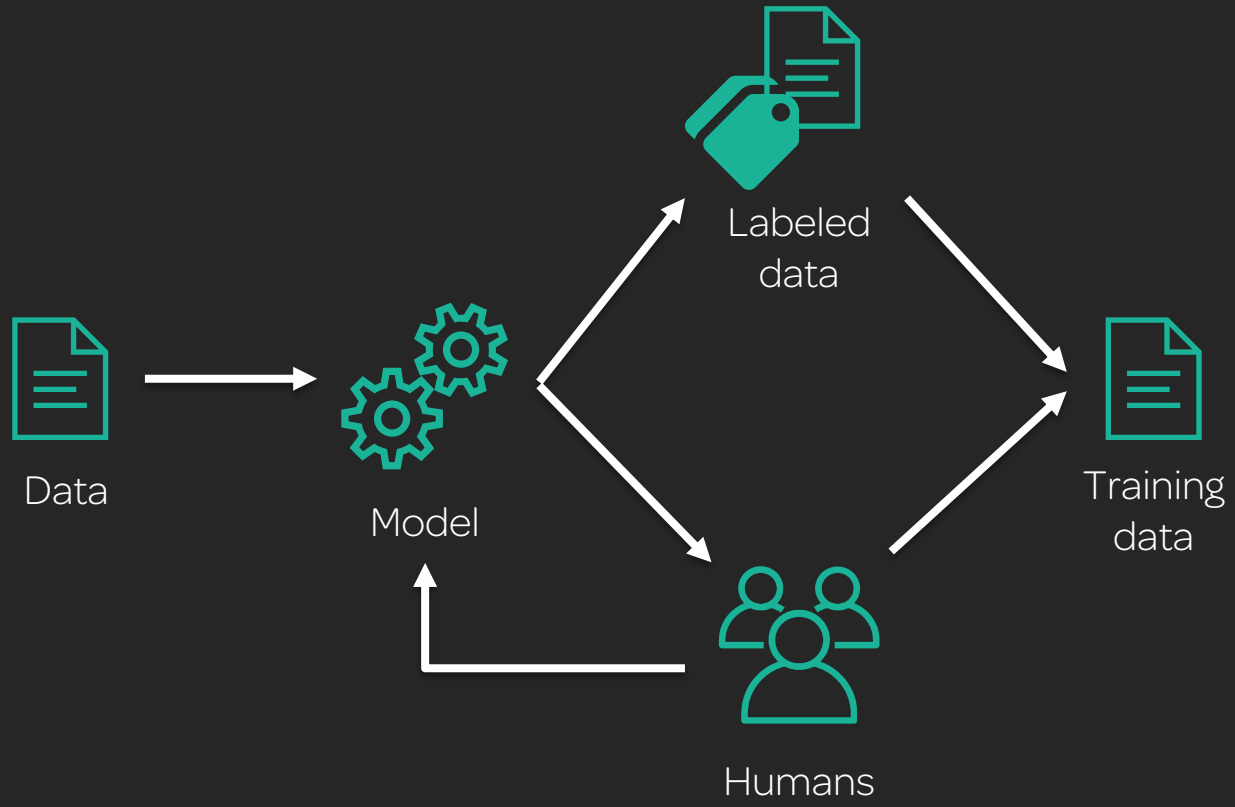
- SageMaker Notebooks
- SageMaker Algorithms

# Amazon Ground Truth

"Build highly accurate training datasets using machine learning and reduce data labeling costs by up to 70%."

- Amazon Web Services

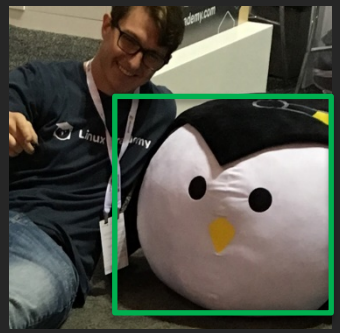
# Amazon Ground Truth

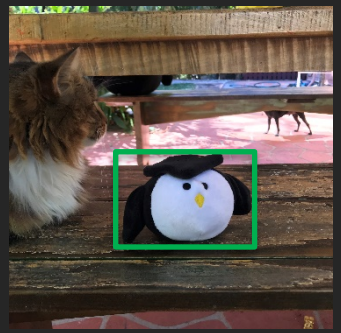


# Amazon Ground Truth

## Features

- Reduce data labeling costs by up to 70%
- Work with public and private human labelers
- Define work instructions:





# SageMaker Algorithms

## Sources

- SageMaker built-in algorithms
- AWS Marketplace
- Custom

# SageMaker Algorithms

## SageMaker Built-In Algorithms

- BlazingText
- Image classification algorithm
- K-means algorithm
- K-nearest neighbors (k-NN) algorithm
- Latent Dirichlet allocation (LDA) algorithm
- Linear learner algorithm
- Object2Vec algorithm
- Principal Component Analysis (PCA) algorithm
- Random Cut Forest (RCF) algorithm
- Sequence-to-Sequence algorithm
- XGBoost algorithm

# SageMaker Algorithms

Think: Amazon Comprehend

## BlazingText

### Type:

- Word2vec / Text classification

### Use cases:

- Natural language processing (NLP)
- Sentiment analysis
- Named Entity Recognition
- Machine translation

# SageMaker Algorithms

Think: Amazon Rekognition

## Image Classification Algorithm

### Type:

- Convolutional Neural Network (CNN)

### Use cases:

- Image recognition

# SageMaker Algorithms

## K-Means Algorithm

### Type:

- K-means
- Based off web-scale k-means clustering algorithm

### Use cases:

- Find discrete groupings within data

# SageMaker Algorithms

Think: Amazon Comprehend

## Latent Dirichlet Allocation (LDA) Algorithm

### Type:

- Latent Dirichlet allocation (LDA) algorithm

### Use cases:

- Text analysis
- Topic discovery

# SageMaker Algorithms

## Principal Component Analysis (PCA) Algorithm

### Type:

- Principal Component Analysis (PCA) algorithm

### Use cases:

- Reduce the dimensionality (i.e. the number of features)

# SageMaker Algorithms

## XGBoost Algorithm

### Type:

- eXtreme Gradient Boosting
- Gradient boosted trees algorithm

### Use cases:

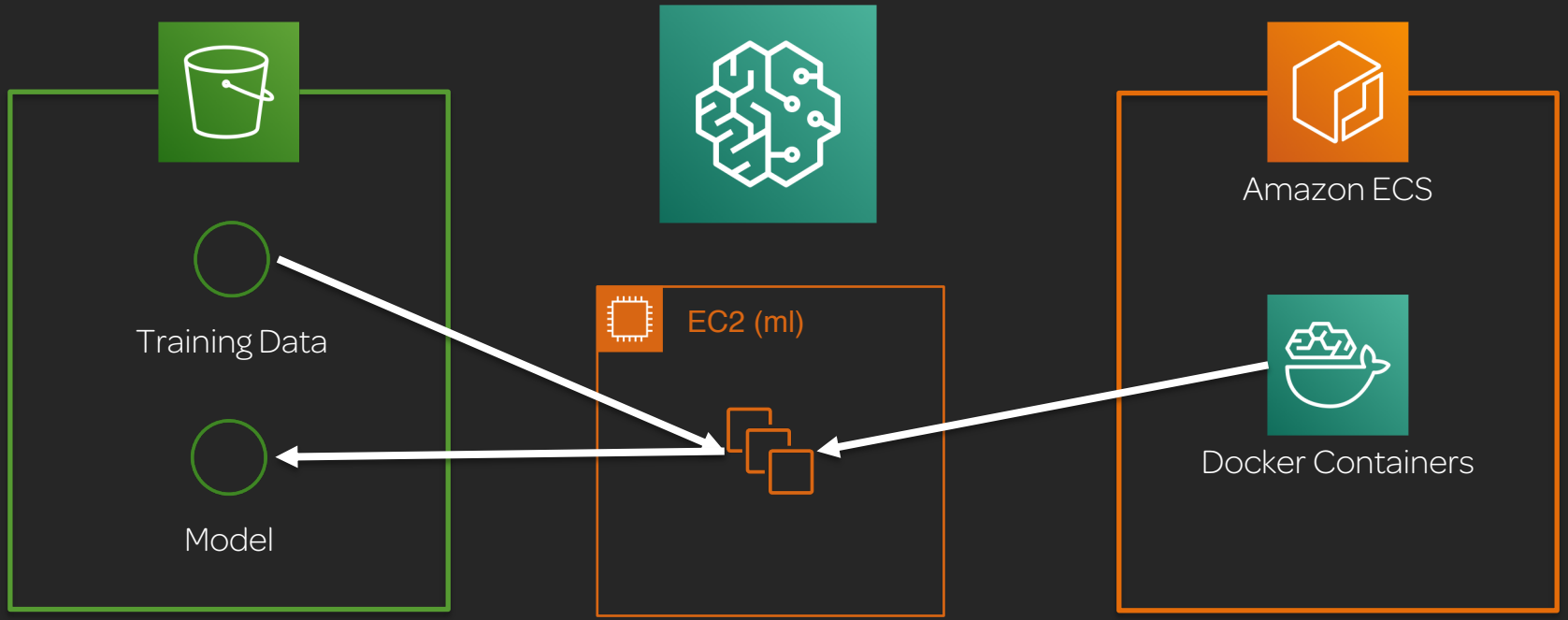
- Making predictions from tabular data

# SageMaker Algorithms

## Documentation Link

Amazon SageMaker built-in algorithms:
https://docs.aws.amazon.com/sagemaker/latest/dg/algos.html

# SageMaker Built-In Algorithms - Architecture



# SageMaker Built-In Algorithms - Architecture

## ML Docker Containers

- SageMaker Built-In Algorithms
- AWS Deep Learning Containers
- AWS Marketplace
- Custom (build your own)...

# SageMaker Built-In Algorithms - Architecture

## Custom ML Docker Containers (Training)

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

- "Amazon SageMaker Containers" tools to create containers.

# SageMaker Built-In Algorithms - Architecture

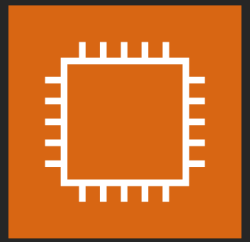
Amazon Simple Storage Service

Amazon Elastic File System


Amazon FSx for Lustre

Parameter: Channel
(e.g: `train`, `validation`, `train_lst`, `validation_lst`, `model`)

# SageMaker Built-In Algorithms - Architecture

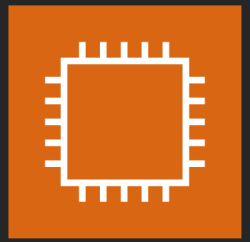


## EC2 Instances

- AWS recommend:
  - ml.m4.xlarge, ml.m4.4xlarge, and ml.m4.10xlarge
  - ml.c4.xlarge, ml.c4.2xlarge, and ml.c4.8xlarge
  - ml.p2.xlarge, ml.p2.8xlarge, and ml.p2.16xlarge


- Some algorithms only support GPUs.
- GPU instances are more expensive, but faster.

# SageMaker Built-In Algorithms - Architecture



## Managed Spot Training

- "Managed spot training can optimize the cost of training models up to 90% over on-demand instances."
- Checkpoints (snapshot of model state in S3)

# Hyperparameter Tuning

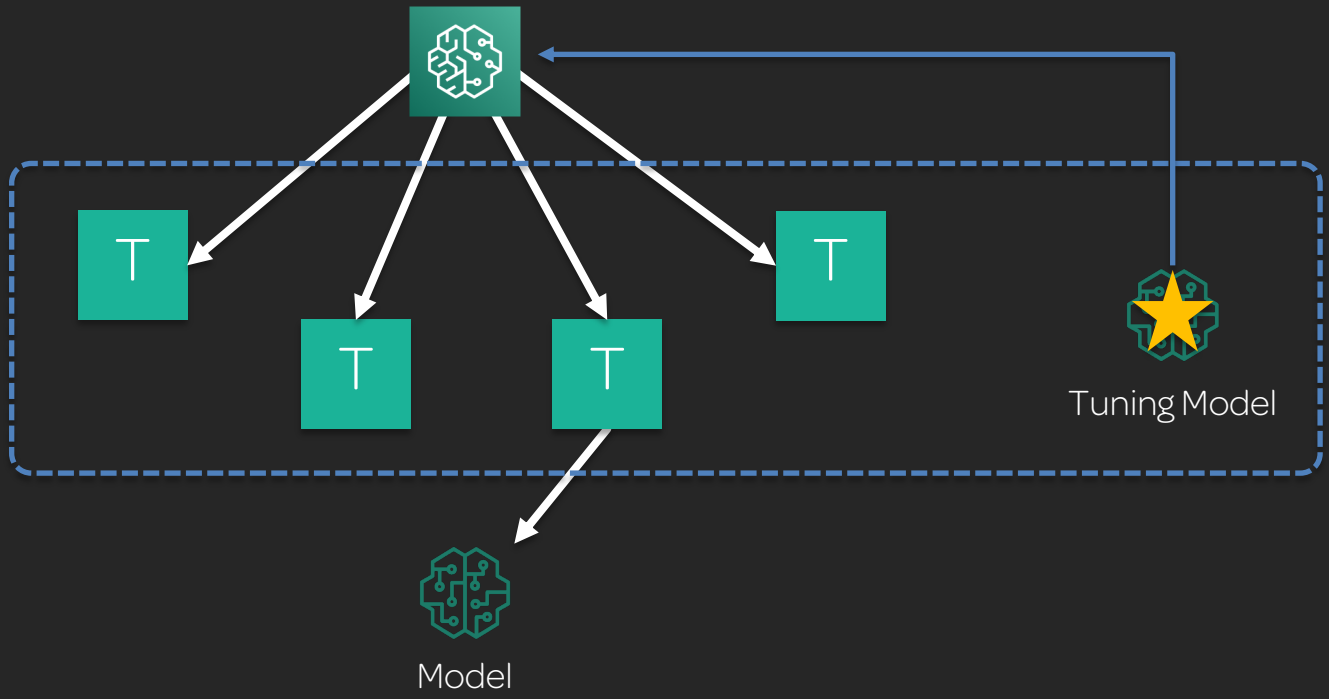
Amazon SageMaker Automatic Model Tuning

1. Choose an algorithm.
   XGBoost algorithm

2. Set ranges of hyperparameters.
   e.g. : `max_depth` – from 3 to 9

3. Choose the metric to measure.
   AUC (area under the curve) - maximize

# Hyperparameter Tuning



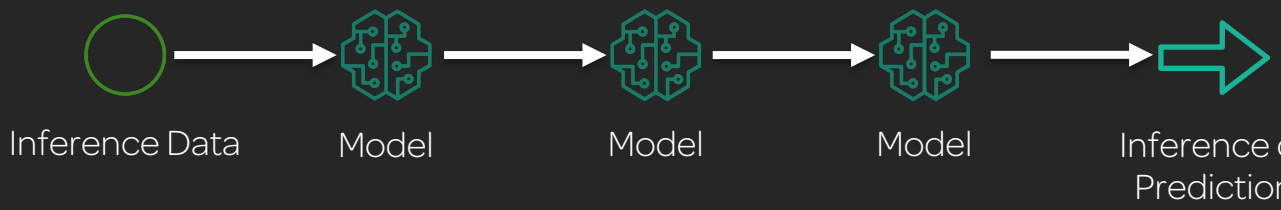
# Hyperparameter Tuning

## Amazon SageMaker Automatic Model Tuning

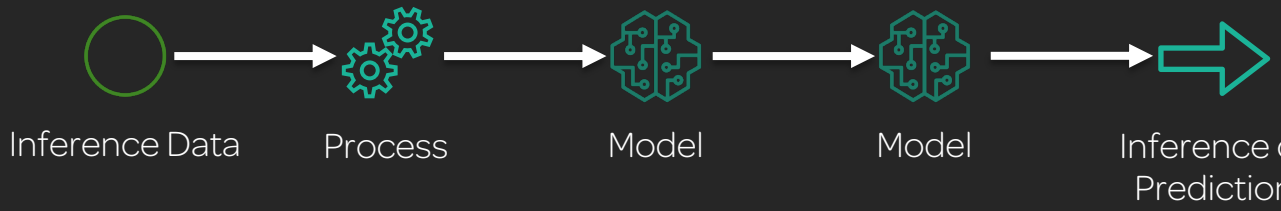
- Works with:
  - AWS built-in algorithms.
  - Custom algorithms.
  - SageMaker pre-built containers.

- Limits:
  - There are limits.
  - Be mindful of the EC2 resource limits.

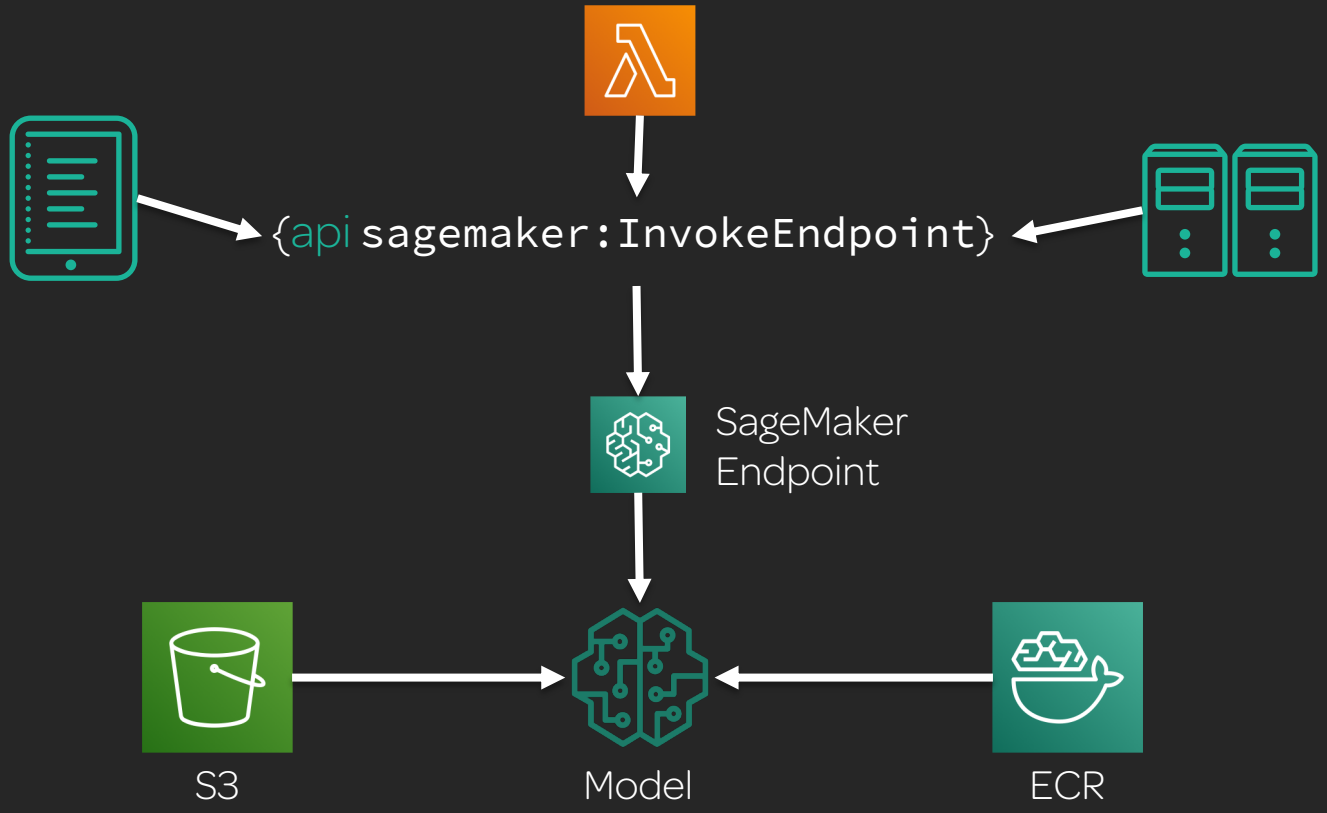
# Inference Pipelines



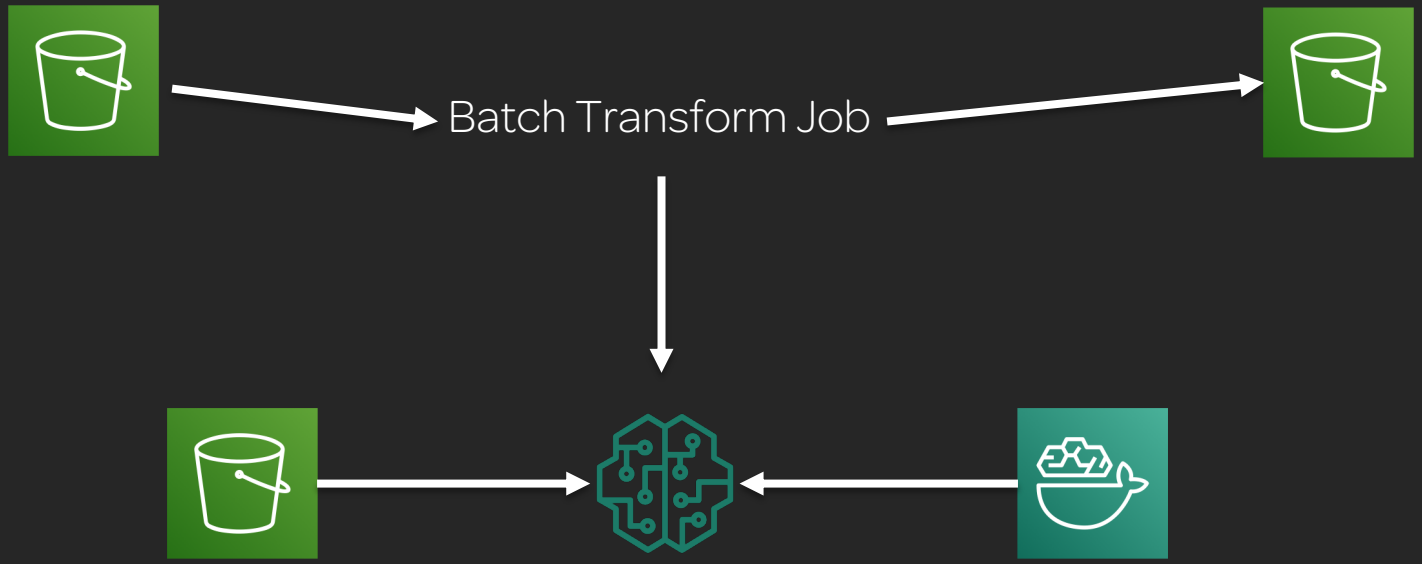
# Inference Pipelines



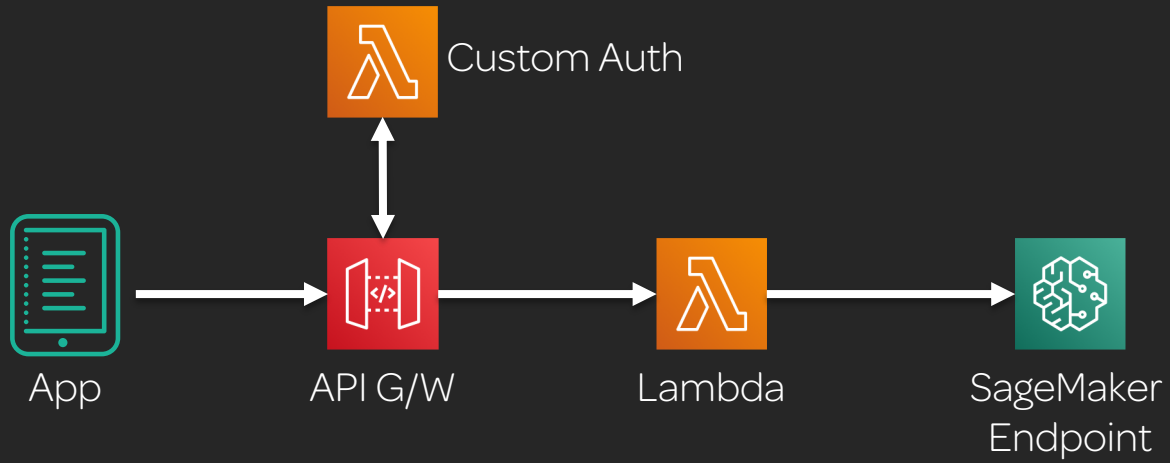
# Real-Time Inference



# Batch Inference



# Accessing Inference from Apps



# Secure SageMaker Notebooks

## Security Controls

- IAM Policy:
  - `sagemaker:CreatePresignedNotebookInstanceUrl`
- Notebook root access.
- SageMaker instance profiles.
- SageMaker does not support resource-based policies.

# SageMaker and the VPC

## VPC Security Controls

- SageMaker hosts models in a pubic VPC by default.
  - Create a private VPC.
- Models and data stored in S3 (public Internet).
  - Create an Amazon S3 VPC endpoint.
  - Use a custom endpoint policy for S3.
  - Encrypt the data within S3.