Training and Evaluating the Model

Amber Israelsen
AUTHOR | DEVELOPER | TRAINER
www.amberisraelsen.com

Course Overview

Course Introduction	Identifying Opportunities for Machine Learning	Defining Machine Learning Problems	Fetching and Preparing Data
Training and Evaluating the Model	Deploying and Monitoring the Model	The AWS Machine Learning Stack	Next Steps

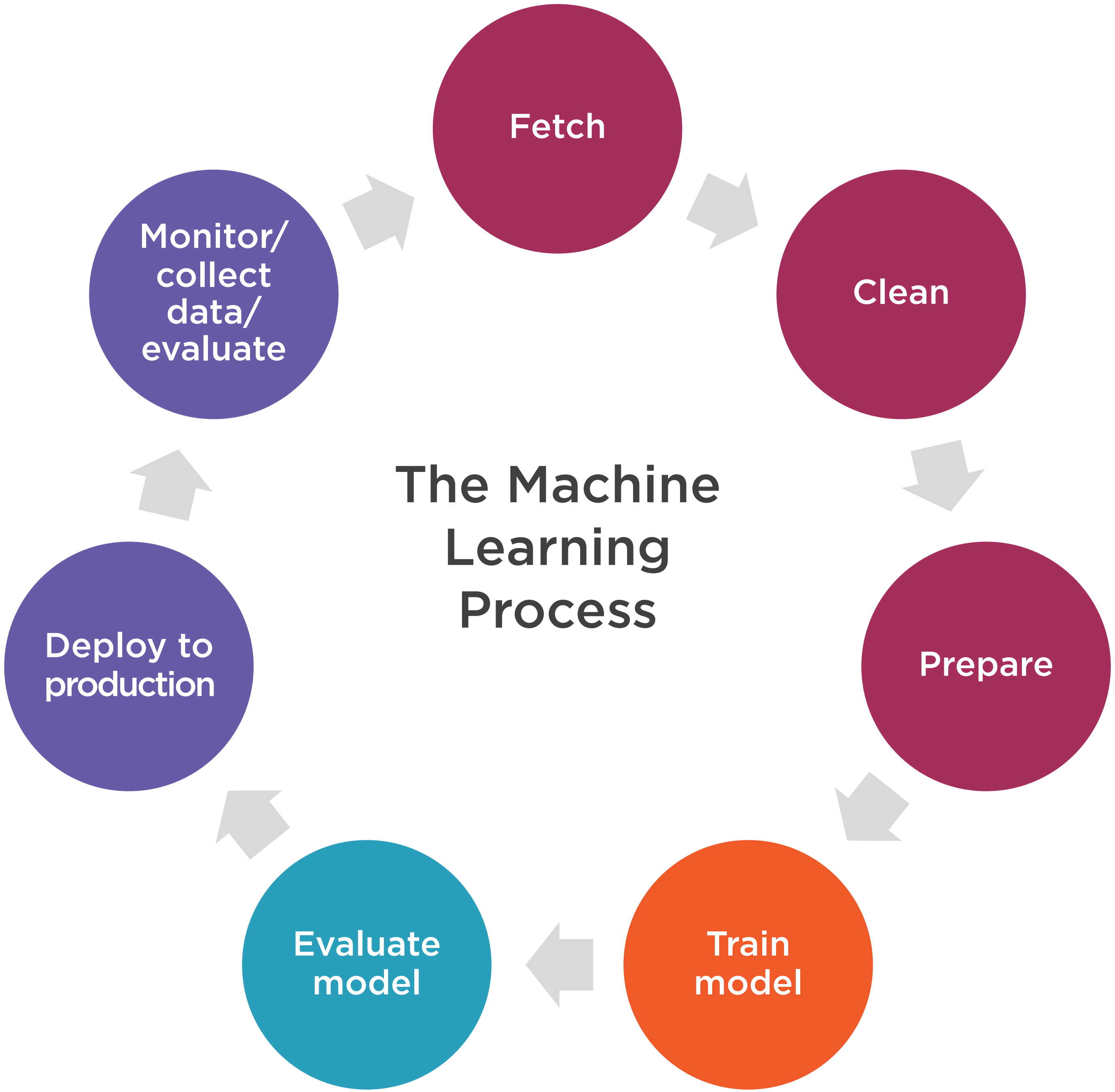
Module Overview

The machine learning process

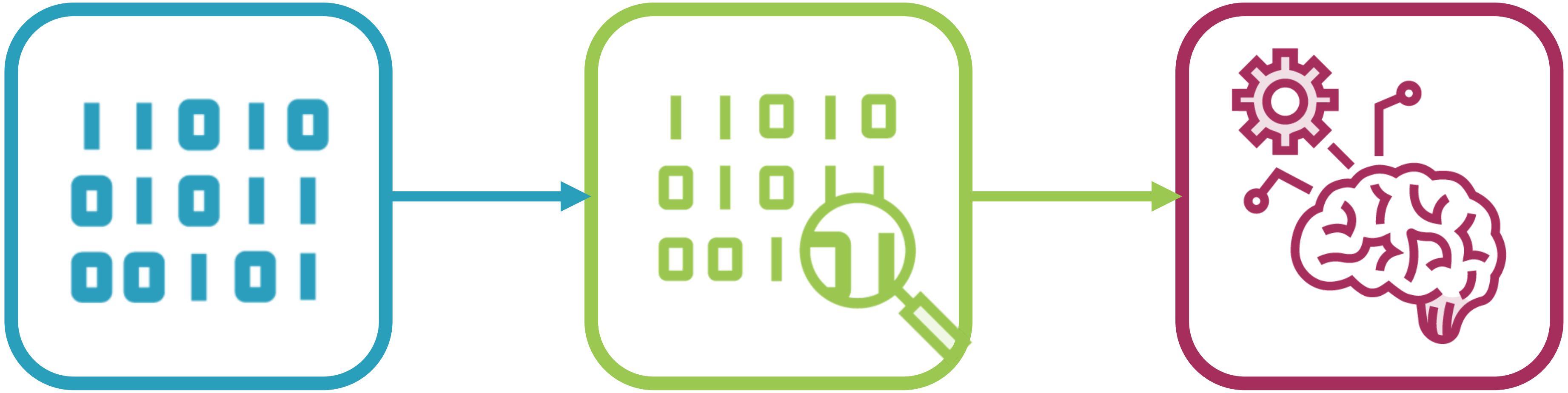
- Training the model
  - Machine learning algorithms
- Evaluating the model
  - Hyperparameter tuning

Demo in SageMaker Studio

Training the Model


Fetch
Clean
Prepare
Train model
Evaluate model
Deploy to production
Monitor/ collect data/ evaluate
The Machine Learning Process

Algorithm vs. Model



Training data

Algorithm

Code that identifies patterns in the data

Model

The output from running the algorithm on data

Rules, numbers, data structures required to make predictions

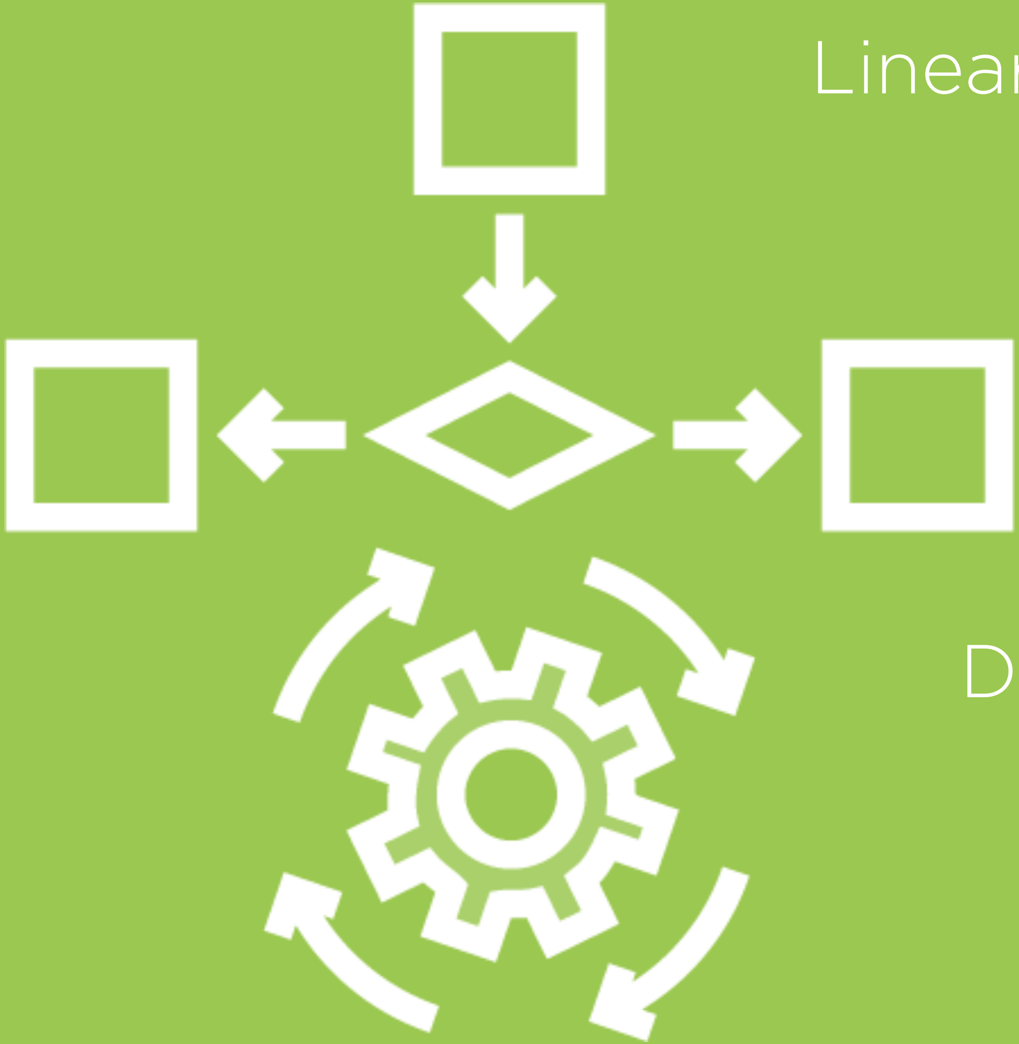


Image classification

K-means

Linear learner

Gradient boost

XGBoost

BlazingText

ALGORITHMS

Decision trees

Object2Vec

Sequence2sequence

Random forest

K-nearest neighbors

The Algorithm Depends on the Type of Problem

SUPERVISED

Learn from labeled data

Classification

BINARY

[yes/no]

[true/false]

[fraud, not fraud]

MULTICLASS

[cat, dog, horse]

[house, condo, townhome, apartment]

Regression

Uses continuous values

[predicting a stock price]

[predicting sales of a product]

UNSUPERVISED

Learn from finding hidden patterns in unlabeled data

Clustering

Groups data into clusters based on similar features

[after analyzing patient data, you find that certain groups respond better to treatment]

Anomaly Detection

Finds outliers in data

[suspicious network traffic]

[abnormal heart beat]

Built-in SageMaker Algorithms

Supervised Learning

- Linear learner
- XGBoost
- K-nearest neighbors

Unsupervised Learning

- K-means
- Principal component analysis (PCA)

Linear Learner Algorithm

$y = 3x + 7$

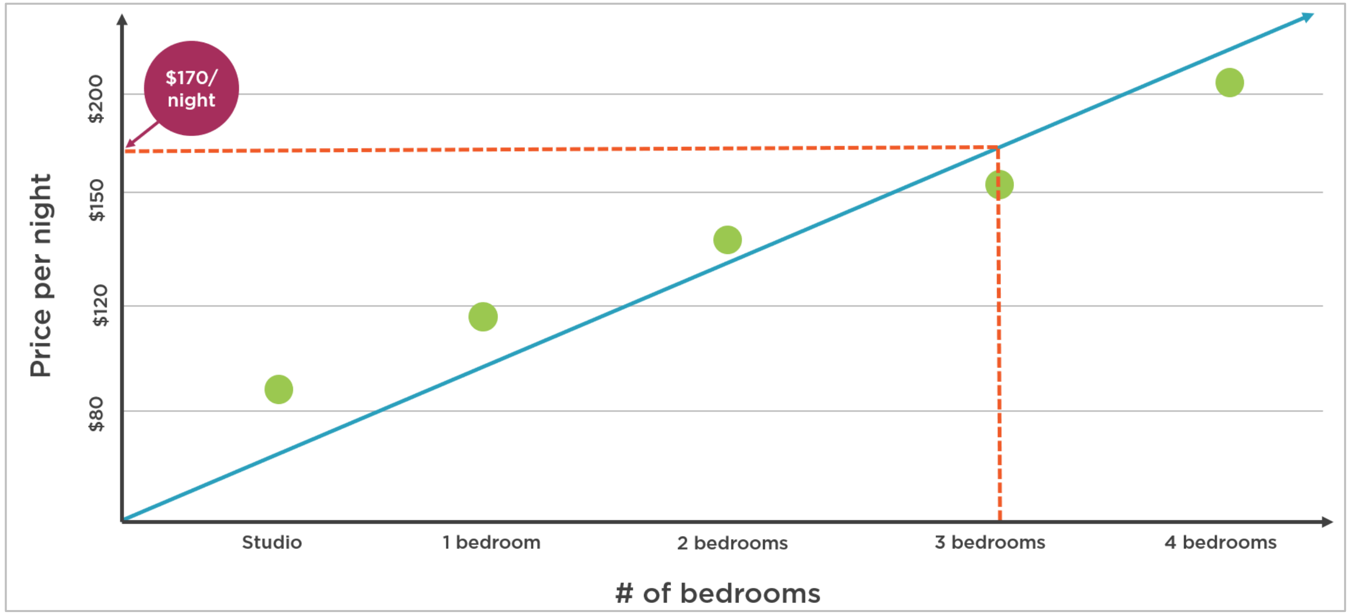
x	y
0	7
1	10
2	13
3	?

Used for classification and regression problems

Classification

- For a given value x, what is y?
- Based on past shopping habits, will this customer buy this product? (yes/no)

Linear Learner Algorithm



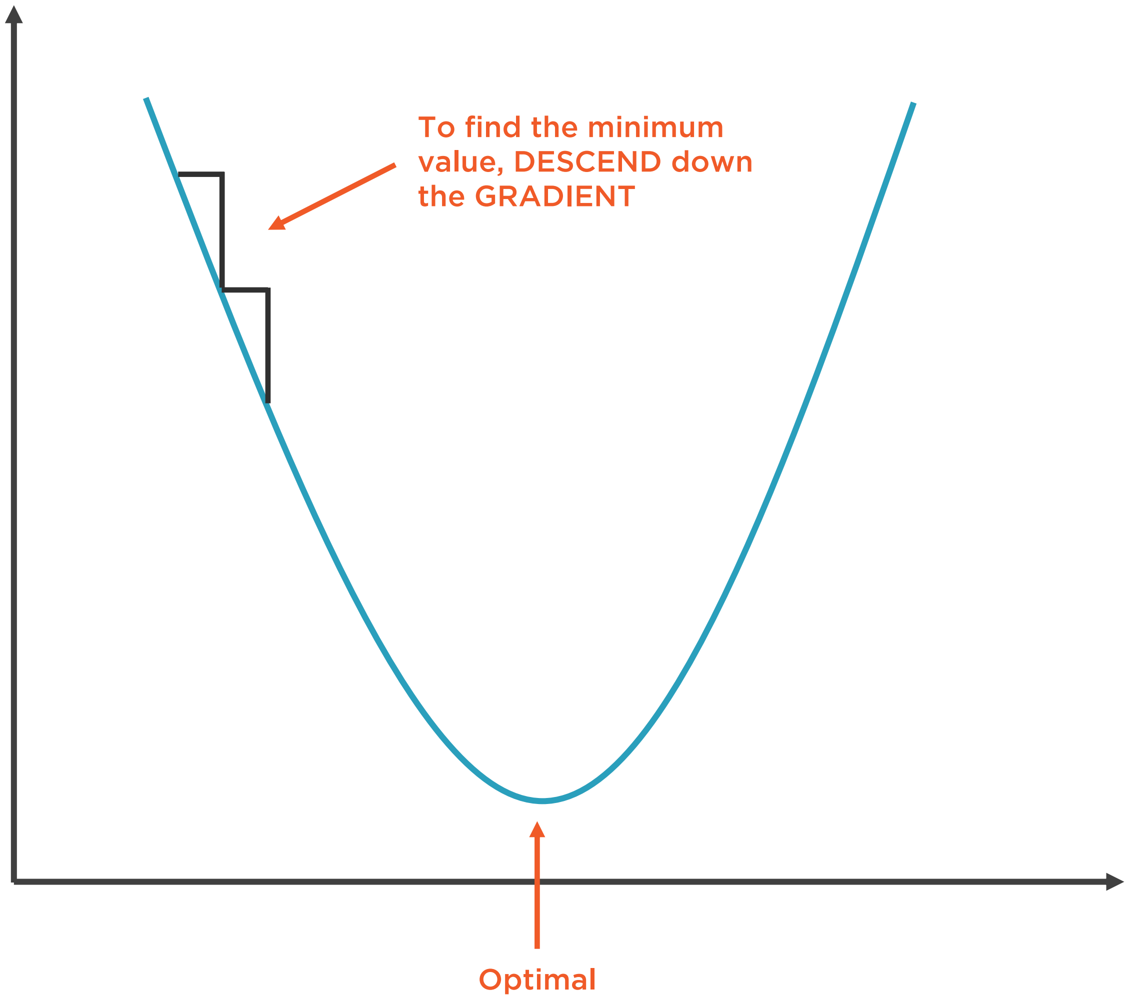
Used for classification and regression problems

Regression

- For a given value x, predict y
- Based on the number of rooms in this house, predict the rental rate for this house (a continuous value)

An implementation of stochastic gradient descent (SGD)

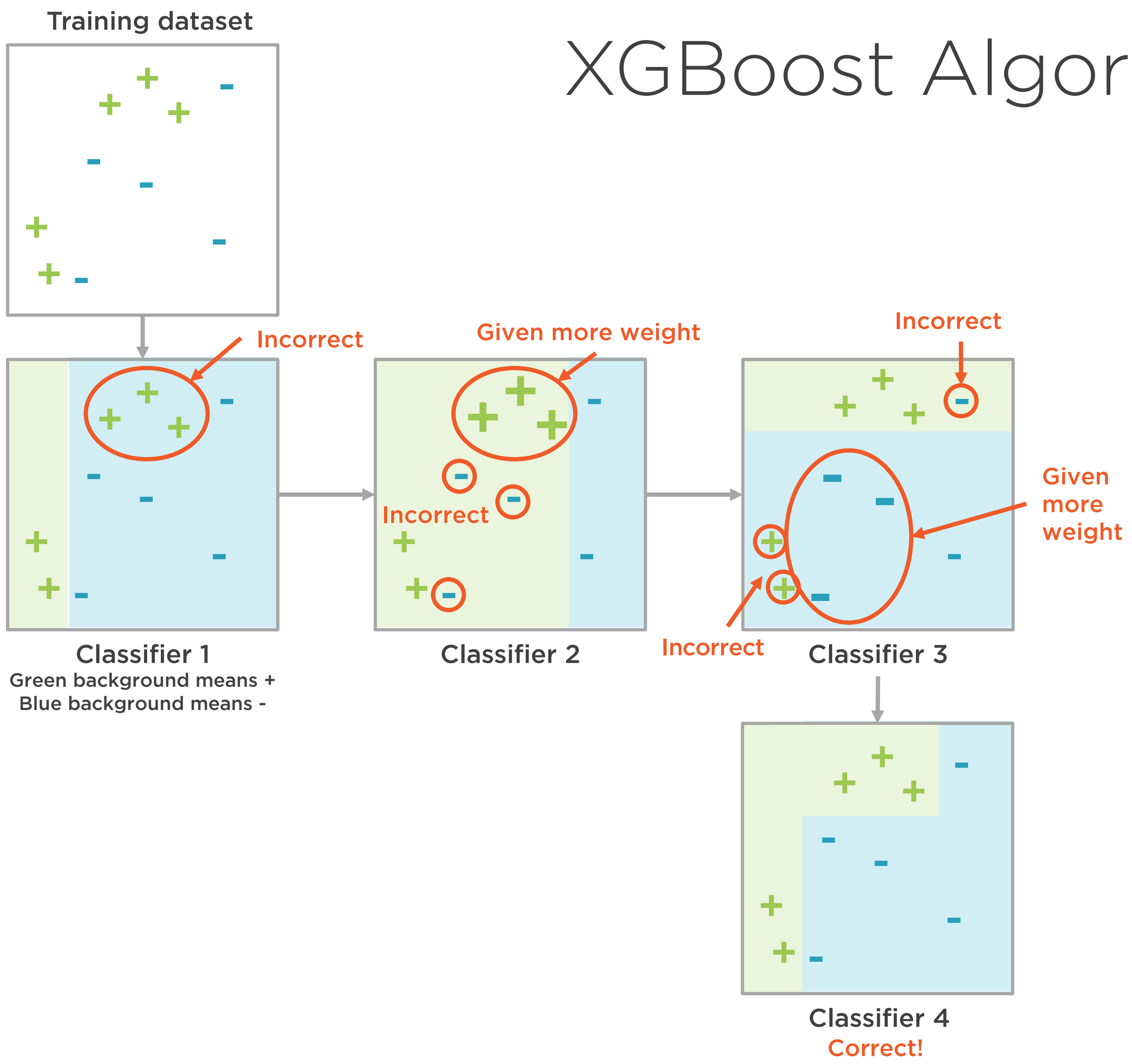
Intuition for Gradient Descent



Loss (cost) function tells us how good our predictions are

- How far away are predicted values from actual values?
- What to minimize the cost/errors

XGBoost Algorithm



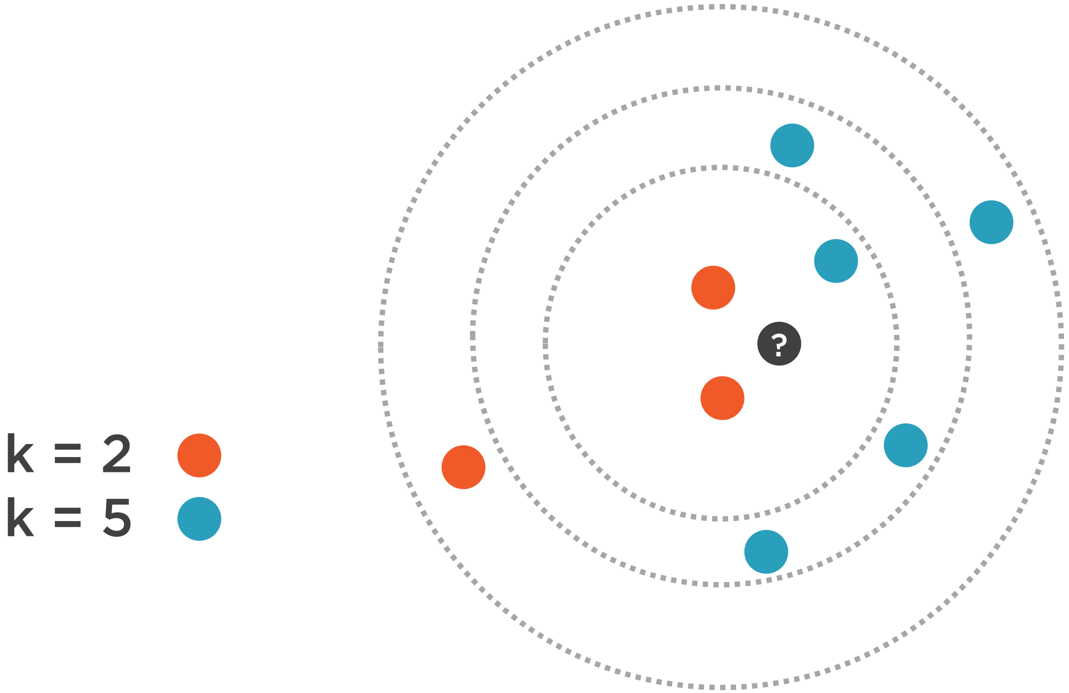
Used for classification and regression problems

An implementation of gradient boosted trees (eXtreme Gradient Boosting)

Designed for speed and performance, and very popular today

Predicts a target variable by attempting to correct the mistakes of models before it

K-nearest Neighbors Algorithm



Used for classification and regression problems, also called k-NN

Makes predictions based on the k points closest to the sample point

Used for concept search, image classification and recommendation systems

Built-in SageMaker Algorithms

Supervised Learning

Linear learner

XGBoost

K-nearest neighbors

Unsupervised Learning

K-means

Principal component analysis (PCA)

K-means Algorithm

TASK

- Organize a pile of books
- We have three bookshelves (groups); k = 3
- They should be grouped by genre (similarity attribute)

Used for clustering (unsupervised) problems

Groups data based on similarity

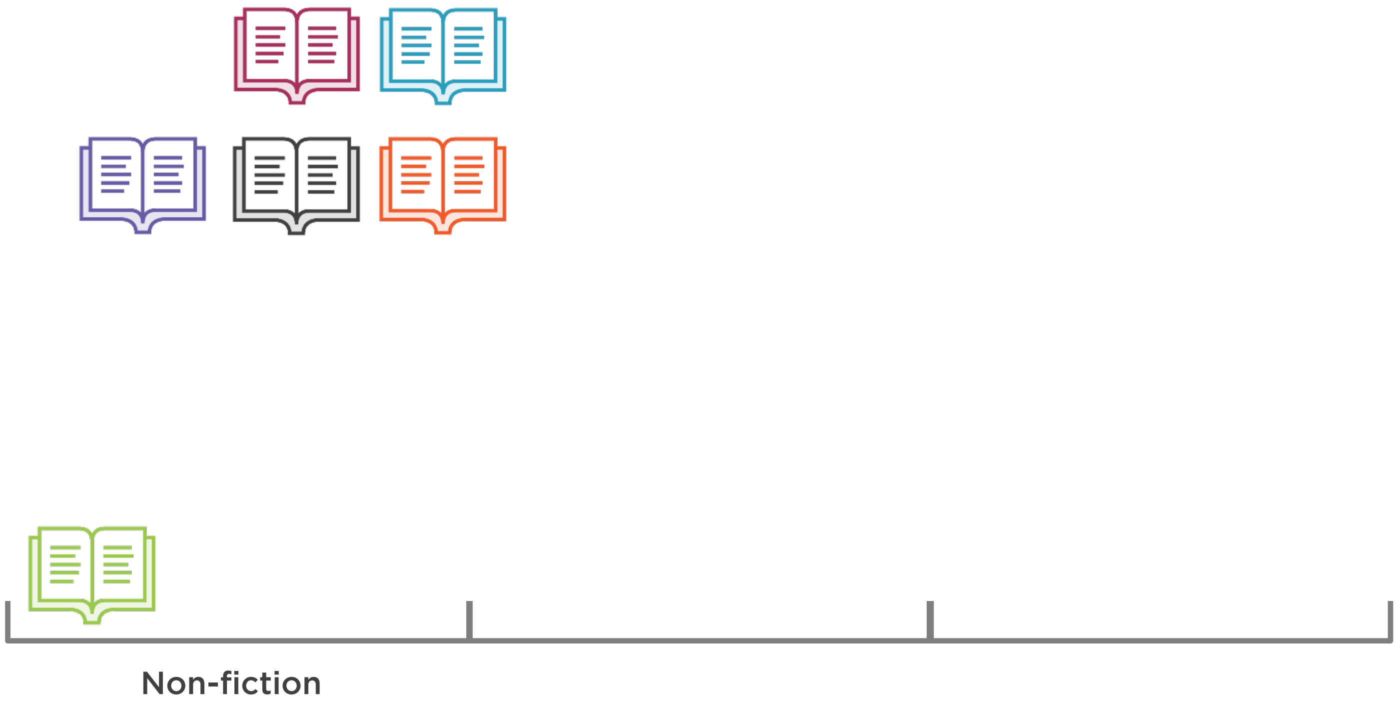
The number of groups is “k”

You must supply attribute that indicates similarity

K-means Algorithm

TASK

- Organize a pile of books
- We have three bookshelves (groups); k = 3
- They should be grouped by genre (similarity attribute)



Used for clustering (unsupervised) problems

Groups data based on similarity

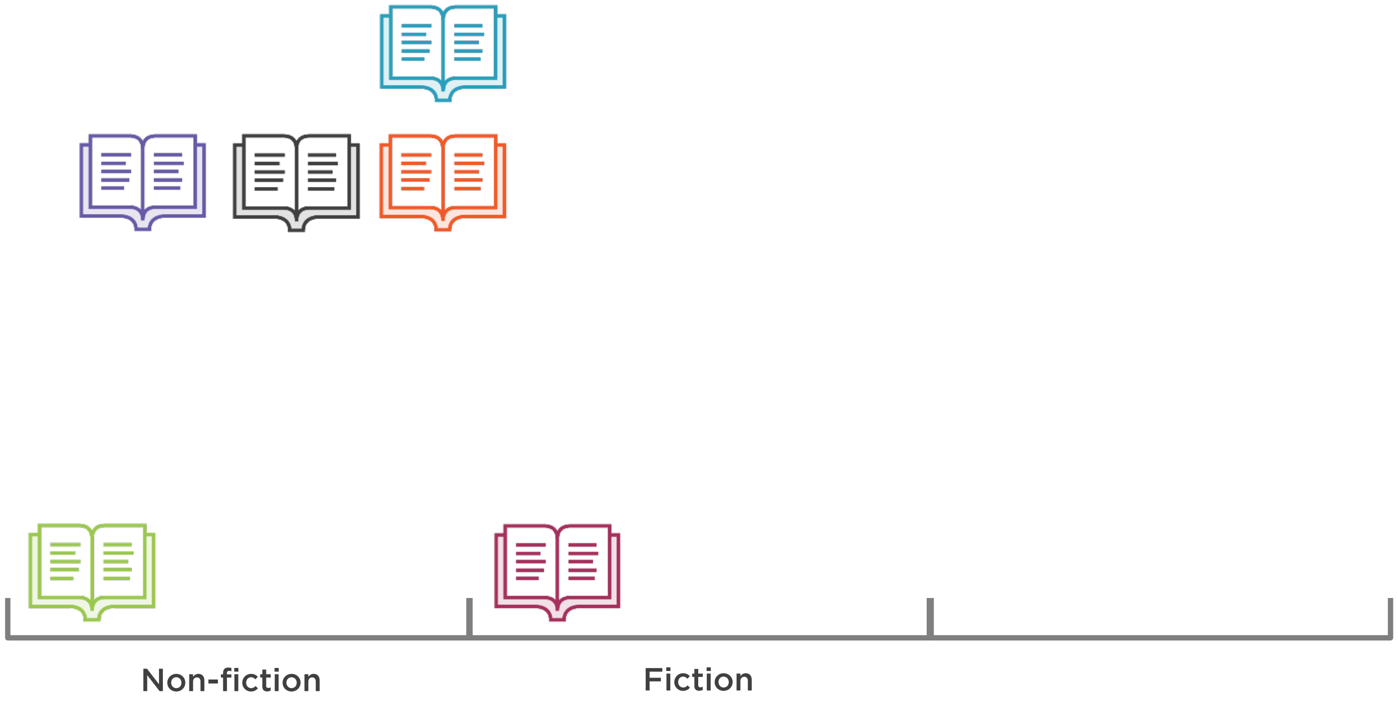
The number of groups is “k”

You must supply attribute that indicates similarity

K-means Algorithm

TASK

- Organize a pile of books
- We have three bookshelves (groups); k = 3
- They should be grouped by genre (similarity attribute)



Used for clustering (unsupervised) problems

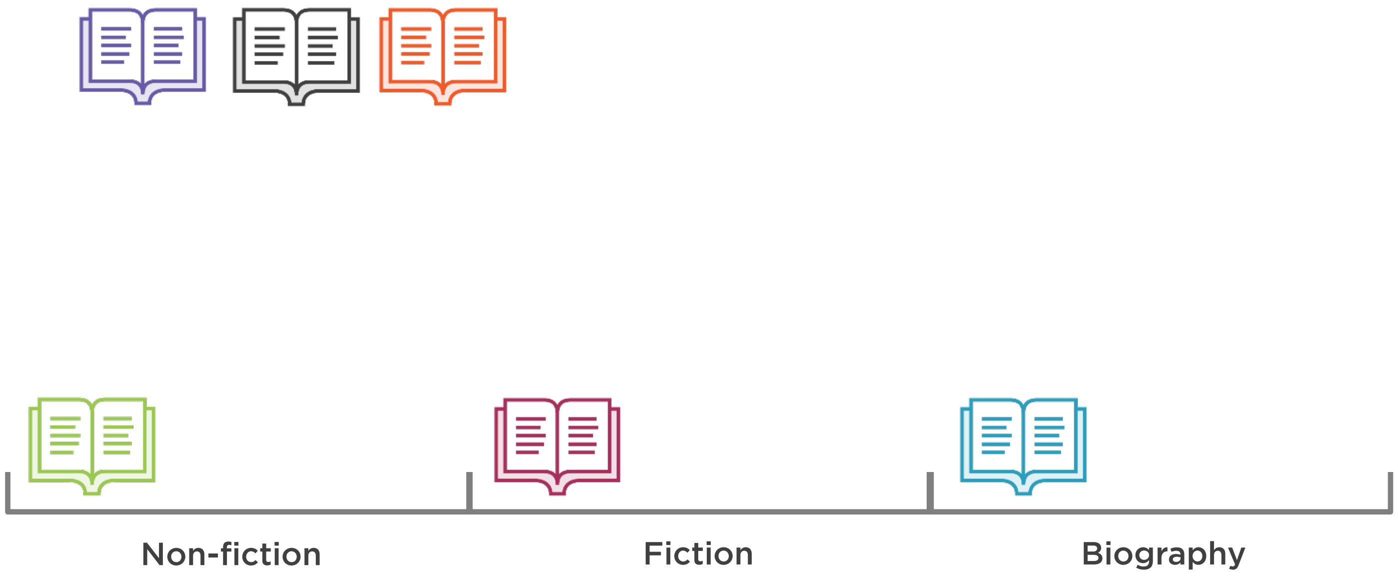
Groups data based on similarity

The number of groups is “k”

You must supply attribute that indicates similarity

K-means Algorithm

TASK
- Organize a pile of books
- We have three bookshelves (groups); k = 3
- They should be grouped by genre (similarity attribute)



Used for clustering (unsupervised) problems

Groups data based on similarity

The number of groups is “k”

You must supply attribute that indicates similarity

K-means Algorithm

TASK

- Organize a pile of books
- We have three bookshelves (groups); k = 3
- They should be grouped by genre (similarity attribute)

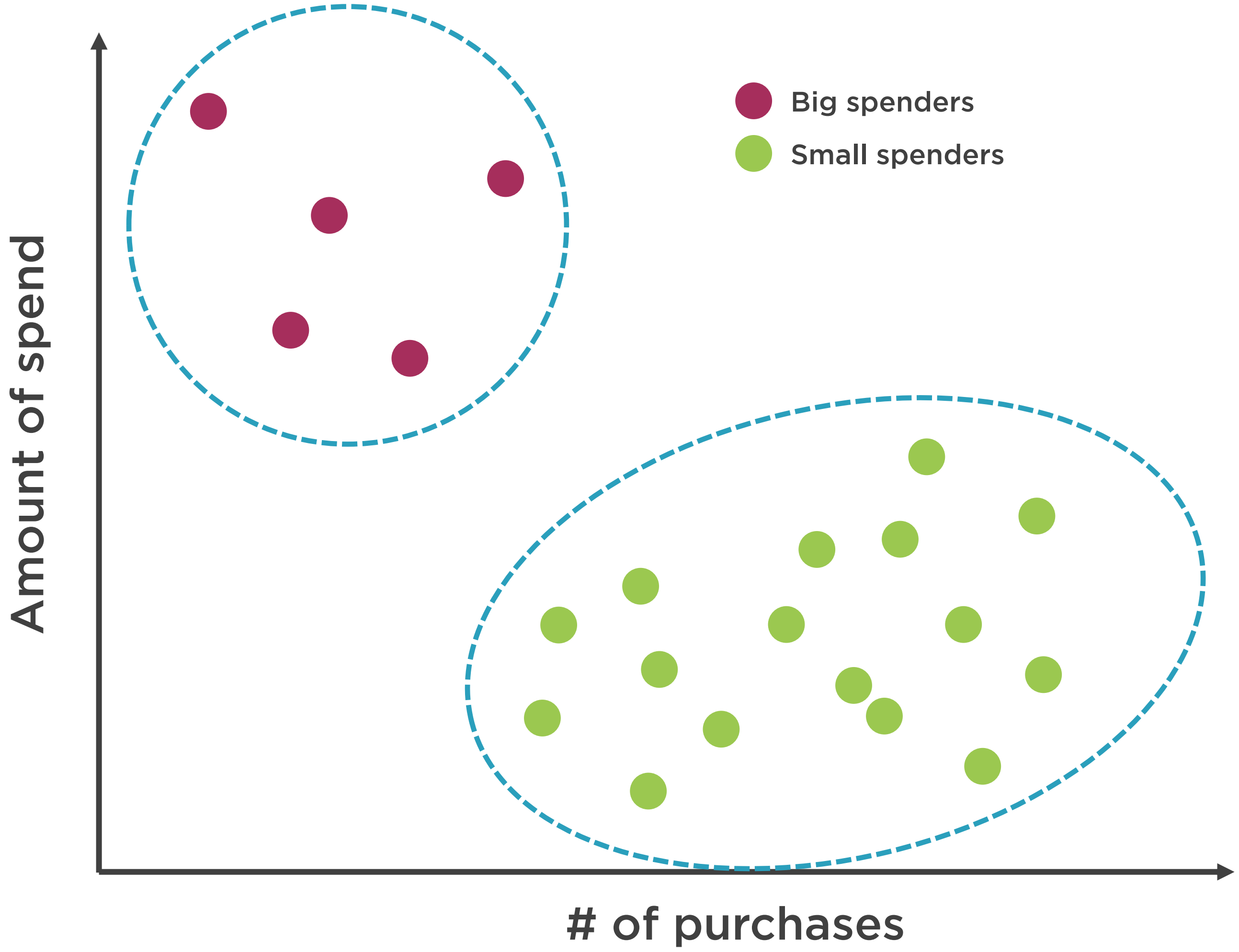
Used for **clustering (unsupervised)** problems

Groups data based on similarity

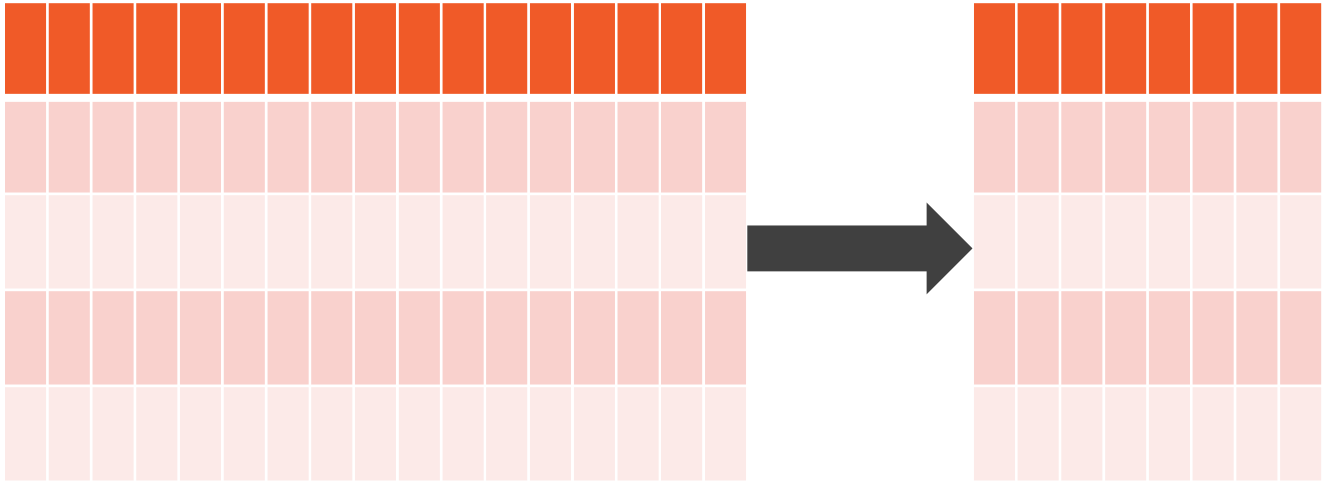
The number of groups is “k”

You must supply attribute that indicates similarity

What Are My Customer Segments?



Principal Component Analysis (PCA)

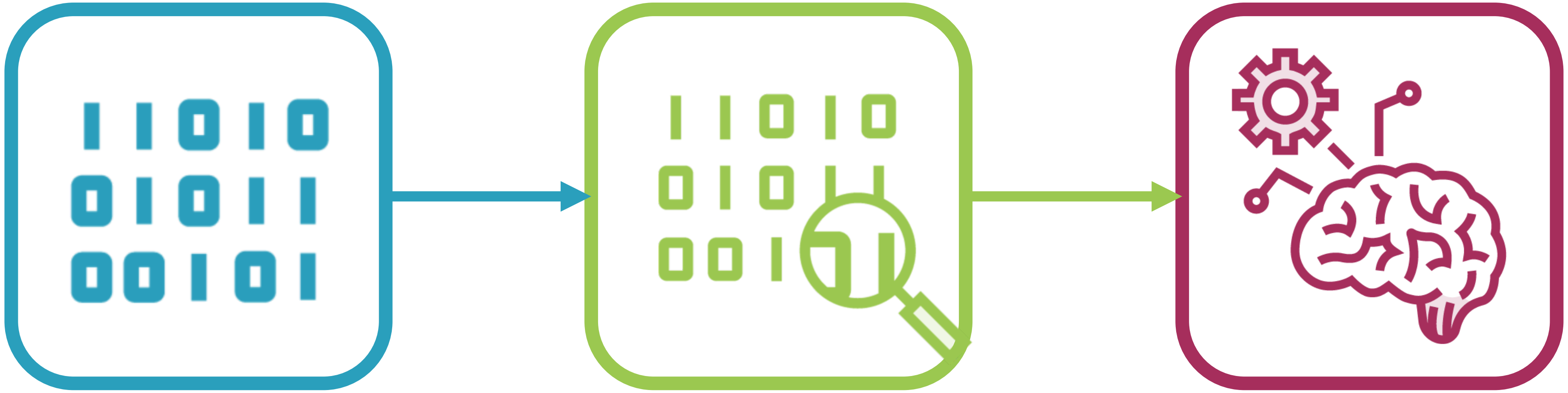


Used for unsupervised problems, to solve “the curse of dimensionality”

Identifies important relationships

Quantifies the importance of those relationships so we can keep the most important and drop others

Algorithm vs. Model



Training data

Algorithm

Code that identifies patterns in the data

Model

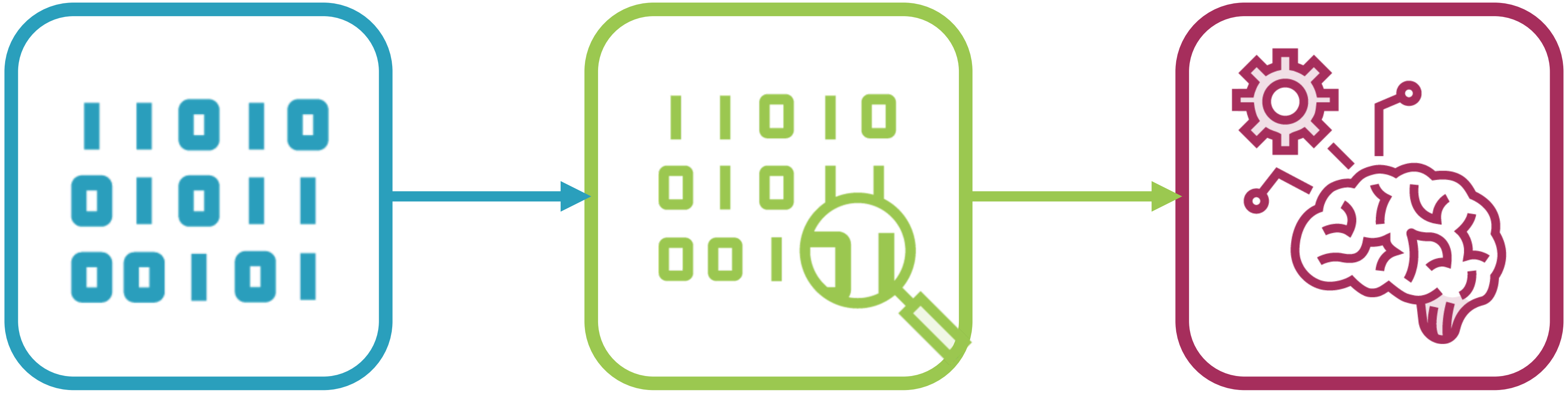
The output from running the algorithm on data

Rules, numbers, data structures required to make predictions

Splitting Data

70%	20%	10%
Train	Validation	Test

Algorithm vs. Model



Training data

Algorithm

Code that identifies patterns in the data

Model

The output from running the algorithm on data

Rules, numbers, data structures required to make predictions

Create the Training Job in SageMaker

#1

S3

Specify URLs for:

S3 bucket containing the training data

S3 bucket to store the model output/artifacts

#2

EC2

Specify the algorithm and compute instances to use for the training

#3

ECR

Specify the path to the code stored in Elastic Container Registry

Create training job

When you create a training job, Amazon SageMaker sets up the distributed compute cluster, performs the training, and deletes the cluster when training has completed. The resulting model artifacts are stored in the location you specified when you created the training job. Learn more

Job settings

Job name

Maximum of 63 alphanumeric characters. Can include hyphens (-), but not spaces. Must be unique within your account in an AWS Region.

IAM role

Amazon SageMaker requires permissions to call other services on your behalf. Choose a role or let us create a role that has the **AmazonSageMakerFullAccess** IAM policy attached.

AmazonSageMaker-ExecutionRole-20200502T174777

Algorithm options

Use an Amazon SageMaker built-in algorithm, your own algorithm, or a third-party algorithm from AWS Marketplace.

Algorithm source

- (•) Amazon SageMaker built-in algorithm Learn more
- ( ) Your own algorithm resource
- ( ) Your own algorithm container in ECR Learn more
- ( ) An algorithm subscription from AWS Marketplace

Choose an algorithm

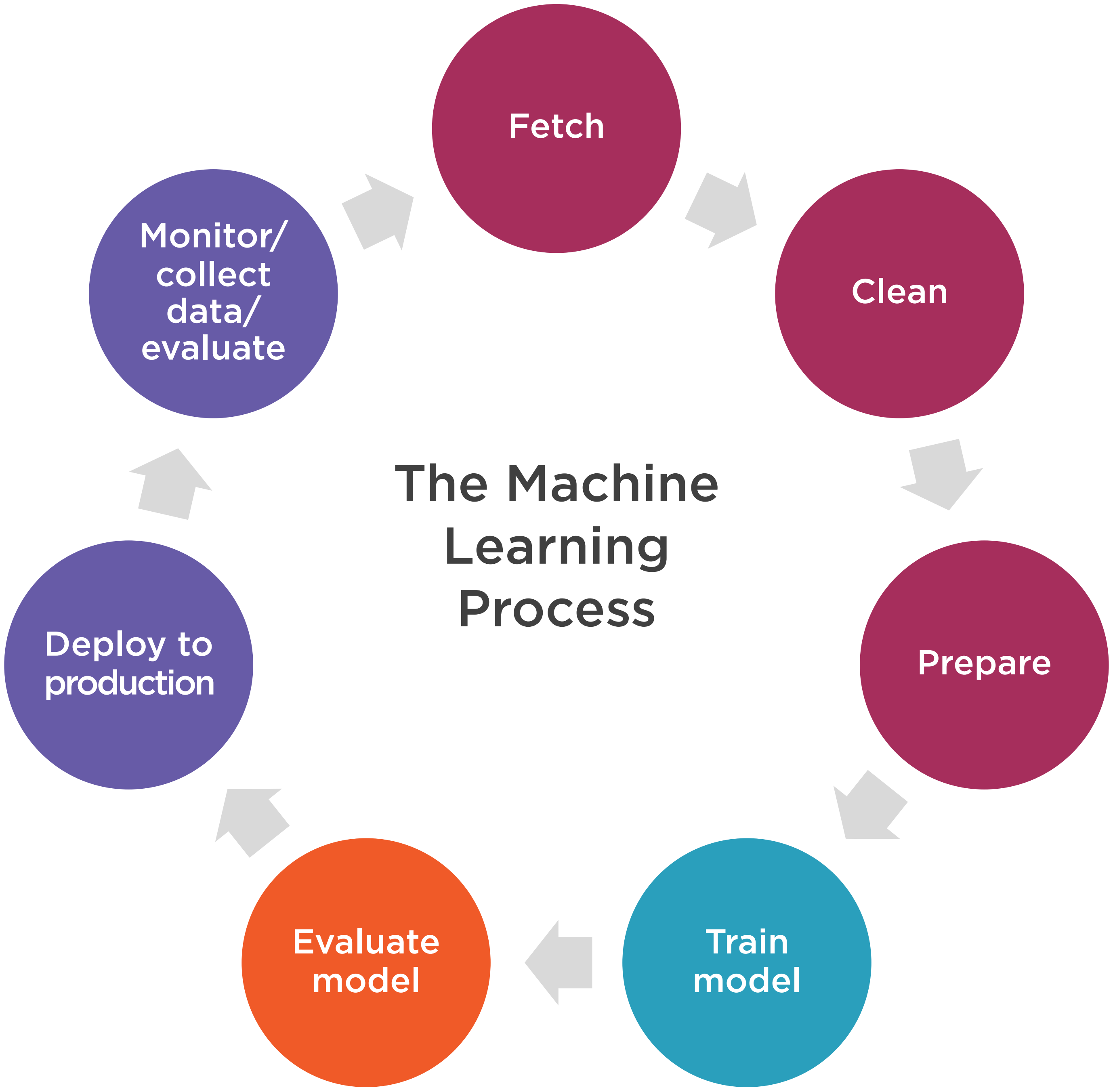
K-Means

Container

The registry path where the training image is stored in Amazon ECR. **Learn more**

382416733822.dkr.ecr.us-east-1.amazonaws.com/kmeans:1

Evaluating the Model

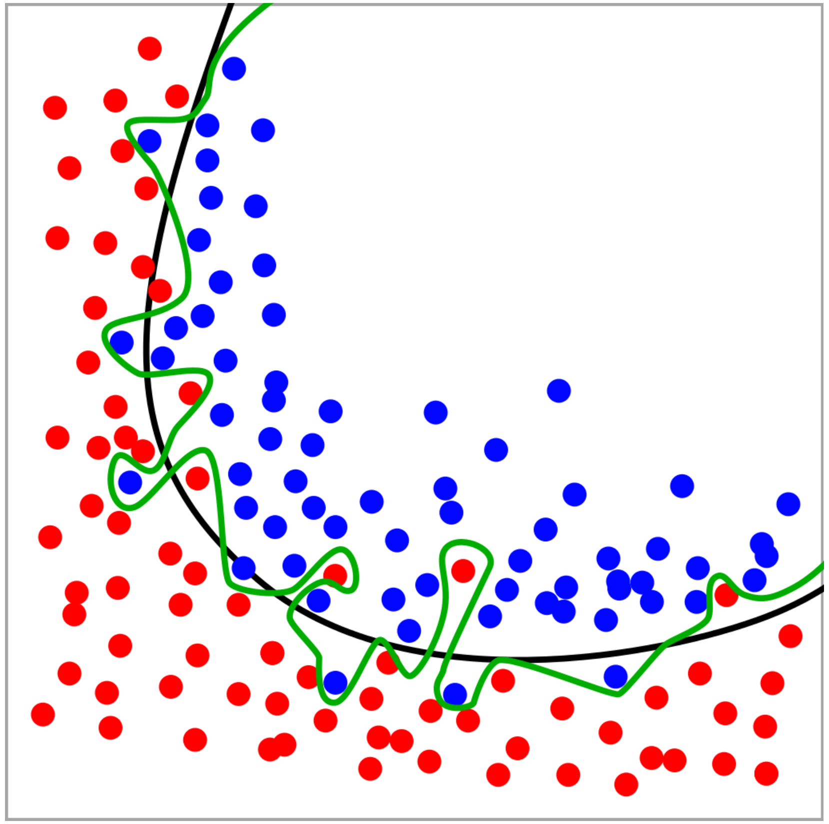


Fetch
Clean
Prepare
Train model
Evaluate model
Deploy to production
Monitor/ collect data/ evaluate
The Machine Learning Process

GOAL

A model that generalizes well

Overfitting



By Chabacano - Own work, CC BY-SA 4.0,
https://commons.wikimedia.org/w/index.php?curid=3610704

The line in green shows **overfitting**

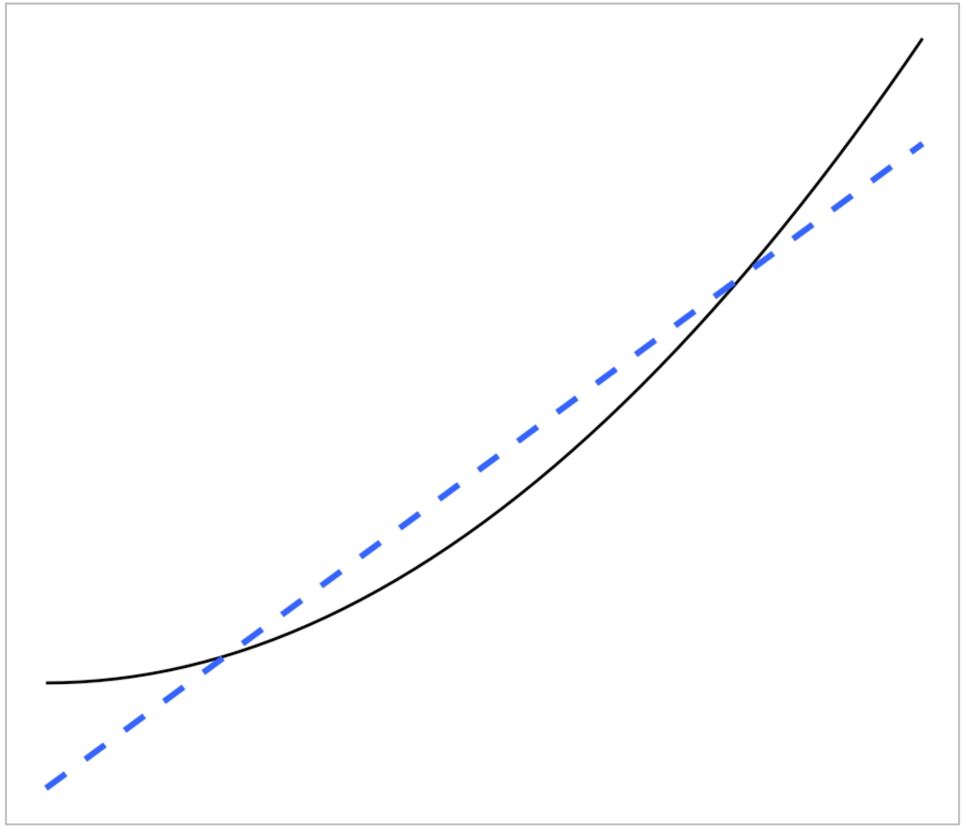
The model is too dependent on the training data; not likely to perform well on new data

We want the black line

To prevent:

- Use more data
- Add some “noise”
- Remove features
- Early stopping

Underfitting



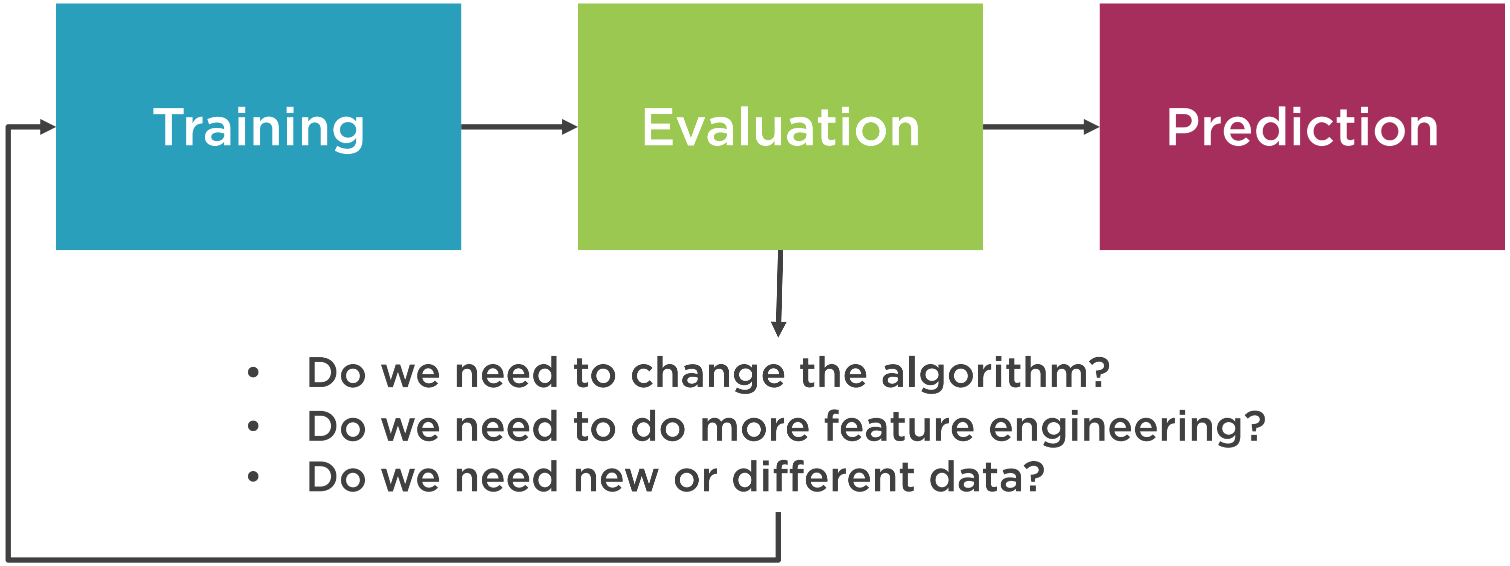
The line in blue shows **underfitting**

The model is too simple and doesn't accurately reflect the data

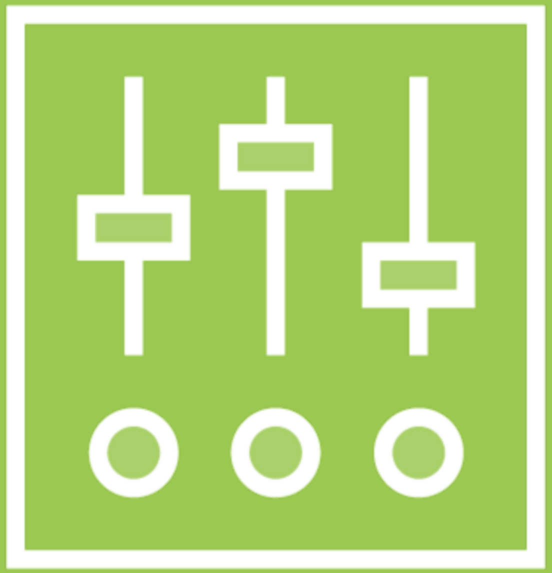
To prevent:

- Use more data
- Add more features
- Train longer

Tuning the Model



Hyperparameter Tuning



Hyperparameters

The “knobs” you can use to control the behavior of your algorithm

Best training job | Training jobs | Training job definitions | Tuning Job configuration | Tags

Best training job summary

This training job is the best training job for only this hyperparameter tuning job.

Create model

Name	Status	Objective metric	Value
MyTuningJob-021-45c74607	Completed	validation:auc	0.728410005569458

Best training job hyperparameters

Name	Type	Value
_tuning_objective_metric	FreeText	validation:auc
alpha	Continuous	0.33456731978167253
eta	Continuous	0.11901669387139481
eval_metric	FreeText	auc
max_depth	Integer	2
min_child_weight	Continuous	1.1917764639235051
num_round	FreeText	100
objective	FreeText	binary:logistic
rate_drop	FreeText	0.3
tweedie_variance_power	FreeText	1.4

Use the Validation Dataset for Hyperparameter Tuning

Hyperparameter Tuning in SageMaker

#1

Choose your objective metric (the thing you're trying to optimize)

#2

Define ranges for hyperparameters

#3

SageMaker runs training jobs to optimize for the objective metric

Hyperparameter tuning is time consuming!

How Did the Model Do with Predictions?

True Values	Predictions
Fraud	Fraud
Not Fraud	Not Fraud
Not Fraud	Fraud
Fraud	Not Fraud
Fraud	Fraud
Not Fraud	Not Fraud
Fraud	Fraud
Not Fraud	Fraud
Fraud	Fraud

The Confusion Matrix

	Predicted Class **POSITIVE** ("fraud")	**Predicted Class** **NEGATIVE** ("not fraud")
Actual Class **POSITIVE** ("fraud")	**True Positive** **(TP)** "fraud" was *correctly* predicted as "fraud"	**False Negative** **(FN)** "fraud" was *incorrectly* predicted as "not fraud"
Actual Class **NEGATIVE** ("not fraud")	**False Positive** **(FP)** "not fraud" was *incorrectly* predicted as "fraud"	**True Negative** **(TN)** "not fraud" was *correctly* predicted as "not fraud"

The Confusion Matrix

	Predicted Class **POSITIVE** ("fraud")	Predicted Class **NEGATIVE** ("not fraud")
Actual Class **POSITIVE** ("fraud")	**True Positive (TP)** "fraud" was *correctly* predicted as "fraud"	**False Negative (FN)** "fraud" was *incorrectly* predicted as "not fraud"
Actual Class **NEGATIVE** ("not fraud")	**False Positive (FP)** "not fraud" was *incorrectly* predicted as "fraud"	**True Negative (TN)** "not fraud" was *correctly* predicted as "not fraud"

The Confusion Matrix

	Predicted Class **POSITIVE** (“fraud”)	Predicted Class **NEGATIVE** (“not fraud”)
Actual Class **POSITIVE** (“fraud”)	**True Positive (TP)** “fraud” was *correctly* predicted as “fraud”	**False Negative (FN)** “fraud” was *incorrectly* predicted as “not fraud”
Actual Class **NEGATIVE** (“not fraud”)	**False Positive (FP)** “not fraud” was *incorrectly* predicted as “fraud”	**True Negative (TN)** “not fraud” was *correctly* predicted as “not fraud”

The Confusion Matrix

	Predicted Class **POSITIVE** (“fraud”)	Predicted Class **NEGATIVE** (“not fraud”)
Actual Class **POSITIVE** (“fraud”)	**True Positive** **(TP)** “fraud” was *correctly* predicted as “fraud”	**False Negative** **(FN)** “fraud” was *incorrectly* predicted as “not fraud”
Actual Class **NEGATIVE** (“not fraud”)	**False Positive** **(FP)** “not fraud” was *incorrectly* predicted as “fraud”	**True Negative** **(TN)** “not fraud” was *correctly* predicted as “not fraud”

Metrics for Classification Problems

ACCURACY

$$\frac{TP+TN}{TP+FP+TN+FN}$$

Percentage of predictions that were correct

Less effective with lots of true negatives

Example: Predicting fraud with little to no fraud data

PRECISION

$$\frac{TP}{TP+FP}$$

Percentage of positive predictions that were correct

Use when the cost of false positives is high

Example: An email is flagged and deleted as spam when it really isn't

RECALL

$$\frac{TP}{TP+FN}$$

Percentage of actual positives that were correctly identified

Use when the cost of false negatives is high

Example: You have cancer, but screening does not find it

Demo

Training and evaluating the model in SageMaker Studio

Key Points to Remember

Algorithms

- Linear learner
- XGBoost
- K-nearest neighbors
- K-means
- Principal component analysis (PCA)

Split the data into training, validation, and test (70/20/10)

Evaluation

- Goal is to have the model generalize, not overfit or underfit
- Hyperparameter tuning
- Confusion matrix
- Accuracy, precision, recall

Up Next:
Deploying and Monitoring the Model