
AWS
re:Invent

AIM308

Choose the right machine learning algorithm in Amazon SageMaker

Denis V. Batalov, PhD
WW Tech Leader, AI/ML
AWS

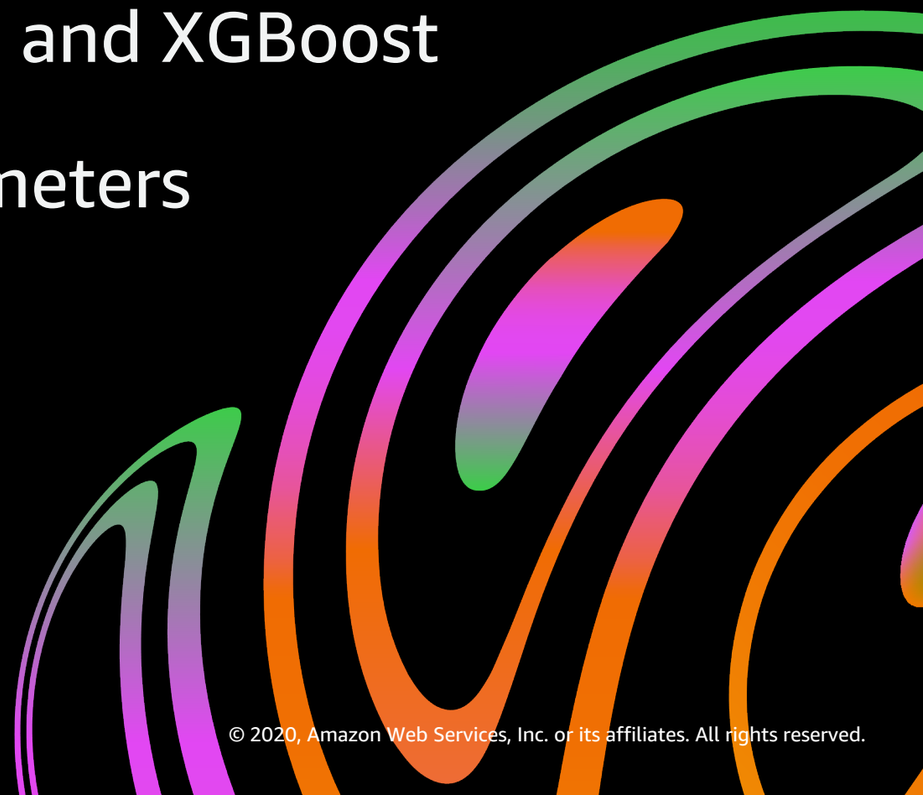
Alberto Danese
Head of Data Science
Nexi

AWS re:Invent

© 2020, Amazon Web Services, Inc. or its affiliates. All rights reserved.

What to expect from this session

- Quick guide on choosing built-in algorithms
- Demo of popular algorithm – Image Classification
- How to choose between Linear Learner, k-NN, and XGBoost
- XGBoost explained – Intuition and hyperparameters
- Nexi – Production use of XGBoost with Amazon SageMaker



© 2020, Amazon Web Services, Inc. or its affiliates. All rights reserved.

17 built-in algorithms in Amazon SageMaker

images

LDA

Linear Learner

K-Means

Image Classification

NTM

k-NN

PCA

Object Detection

BlazingText

XGBoost

Random Cut Forest

Semantic Segmentation

Sequence to Sequence

DeepAR

IP Insights

Object2Vec

Factorization Machines

17 built-in algorithms in Amazon SageMaker

text

LDA

NTM

BlazingText

Sequence to Sequence

Linear Learner

k-NN

XGBoost

DeepAR

K-Means

PCA

Random Cut Forest

IP Insights

Image Classification

Object Detection

Semantic Segmentation

Object2Vec

Factorization Machines

17 built-in algorithms in Amazon SageMaker

numbers, tokens

LDA

Linear Learner

K-Means

Image Classification

NTM

k-NN

PCA

Object Detection

BlazingText

XGBoost

Random Cut Forest

Semantic Segmentation

Sequence to Sequence

DeepAR

IP Insights

Object2Vec

Factorization Machines

17 built-in algorithms in Amazon SageMaker

computer vision

LDA

Linear Learner

K-Means

Image Classification

NTM

k-NN

PCA

Object Detection

BlazingText

XGBoost

Random Cut Forest

Semantic Segmentation

Sequence to Sequence

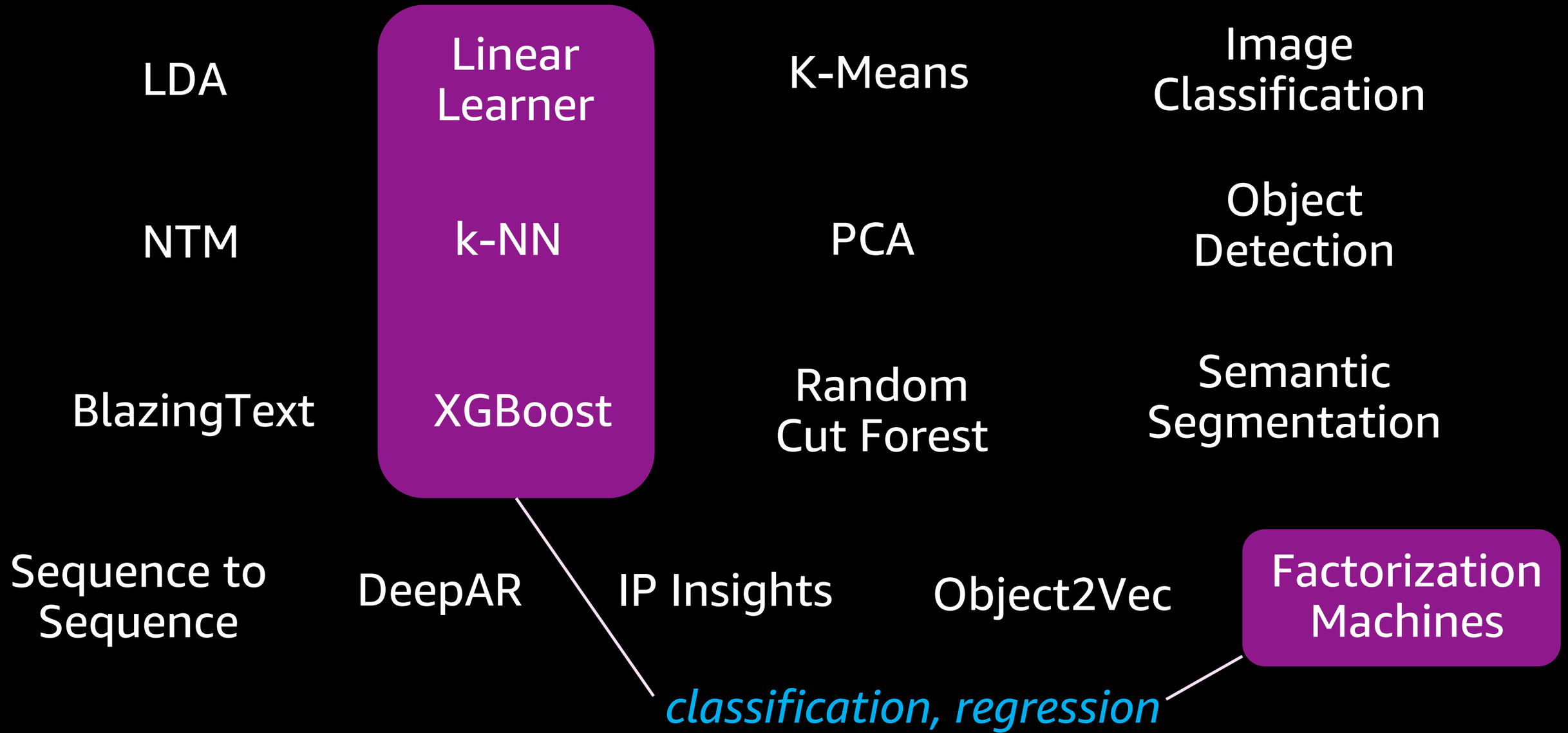
DeepAR

IP Insights

Object2Vec

Factorization Machines

17 built-in algorithms in Amazon SageMaker



17 built-in algorithms in Amazon SageMaker

topic modeling

LDA

NTM

Linear Learner

k-NN

K-Means

PCA

Image Classification

Object Detection

BlazingText

XGBoost

Random Cut Forest

Semantic Segmentation

Sequence to Sequence

DeepAR

IP Insights

Object2Vec

Factorization Machines

17 built-in algorithms in Amazon SageMaker

LDA

Linear Learner

K-Means

Image Classification

NTM

k-NN

PCA

Object Detection

text classification

BlazingText

XGBoost

Random Cut Forest

Semantic Segmentation

embedding

Sequence to Sequence

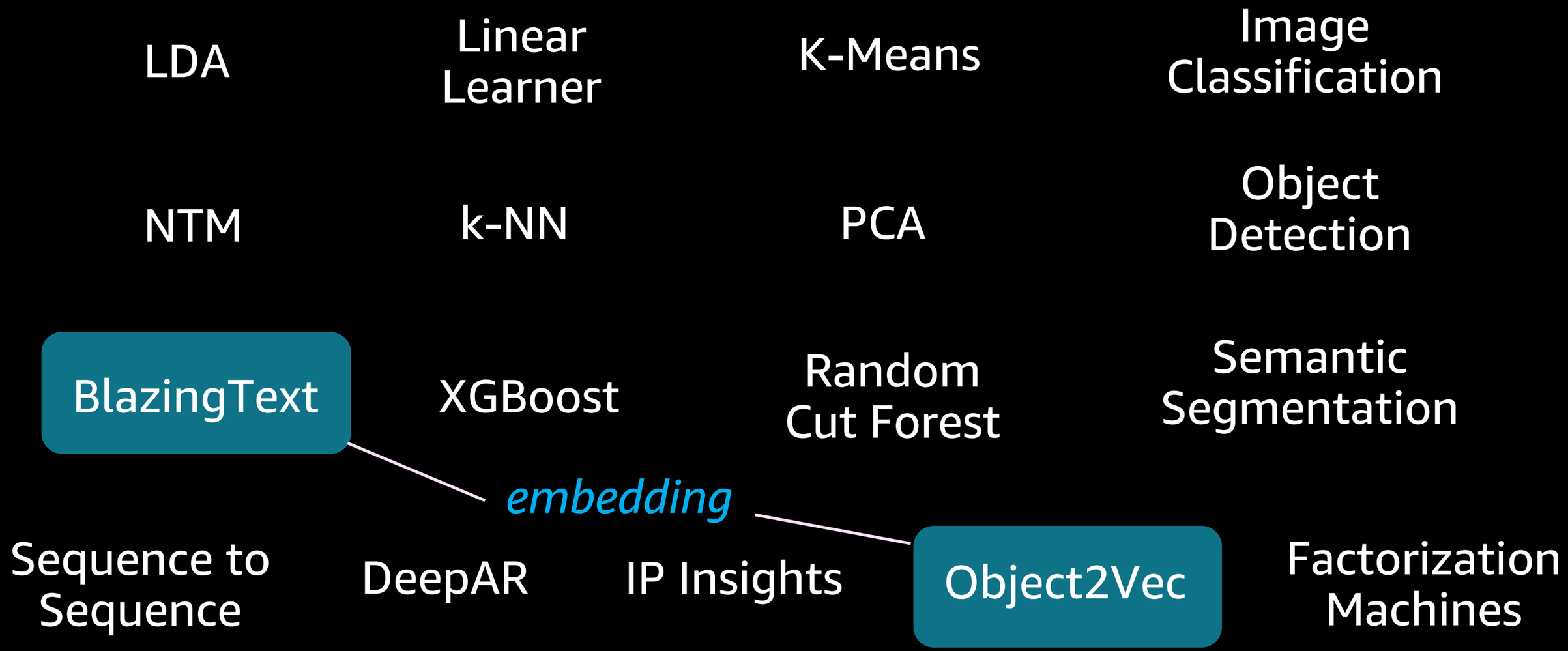
DeepAR

IP Insights

Object2Vec

Factorization Machines

17 built-in algorithms in Amazon SageMaker



17 built-in algorithms in Amazon SageMaker

LDA

Linear Learner

K-Means

Image Classification

dim reduction

NTM

k-NN

PCA

Object Detection

BlazingText

XGBoost

Random Cut Forest

Semantic Segmentation

Sequence to Sequence

DeepAR

IP Insights

Object2Vec

Factorization Machines

17 built-in algorithms in Amazon SageMaker

		clustering	
LDA	Linear Learner	K-Means	Image Classification
NTM	k-NN	PCA	Object Detection
BlazingText	XGBoost	Random Cut Forest	Semantic Segmentation
Sequence to Sequence	DeepAR	IP Insights	Object2Vec	Factorization Machines

17 built-in algorithms in Amazon SageMaker

LDA

Linear Learner

K-Means

Image Classification

NTM

k-NN

PCA

Object Detection

BlazingText

XGBoost

Random Cut Forest

Semantic Segmentation

anomaly detection

Sequence to Sequence

DeepAR

IP Insights

Object2Vec

Factorization Machines

17 built-in algorithms in Amazon SageMaker

LDA

Linear Learner

K-Means

Image Classification

NTM

k-NN

PCA

Object Detection

BlazingText

XGBoost

Random Cut Forest

Semantic Segmentation

translation, etc.

Sequence to Sequence

DeepAR

IP Insights

Object2Vec

Factorization Machines

17 built-in algorithms in Amazon SageMaker

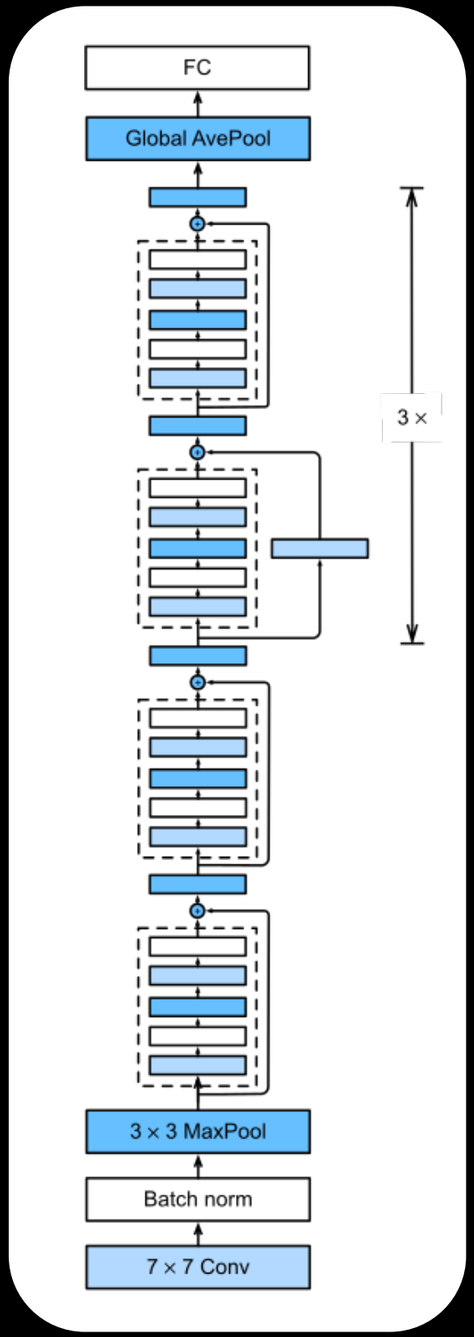
LDA	Linear Learner	K-Means	Image Classification
NTM	k-NN	PCA	Object Detection
BlazingText	XGBoost	Random Cut Forest	Semantic Segmentation
Sequence to Sequence	*forecasting* DeepAR	IP Insights	Object2Vec	Factorization Machines

17 built-in algorithms in Amazon SageMaker

LDA	Linear Learner	K-Means	Image Classification
NTM	k-NN	PCA	Object Detection
BlazingText	XGBoost	Random Cut Forest	Semantic Segmentation
Sequence to Sequence	DeepAR	IP Insights	*recommendations* Object2Vec, Factorization Machines

Image Classification details

- ResNet with 18 through 200 layers
- Pre-trained weights available (ImageNet or CIFAR-10) for transfer learning
- Apache MXNet RecordIO is the recommended input format
- Automated model tuning
- Batch Inference possible



Chest X-ray dataset

National Institutes of Health



- Public dataset of 112K chest X-rays
- From 30805 patients, anonymized
- 14 disease labels text-mined from radiological reports using NLP
- Each image can have multiple labels

Amazon SageMaker

Amazon SageMaker Studio

Dashboard
Search
Images

Ground Truth
Labeling jobs
Labeling datasets
Labeling workforces

Notebook
Notebook instances
Lifecycle configurations
Git repositories

Choose your user name, then choose Open Studio to get started

Add user

Search users

1

User name	Last modified	Created	
denisb	Oct 22, 2020 03:51 UTC	Oct 22, 2020 03:51 UTC	Open Studio

Studio Summary

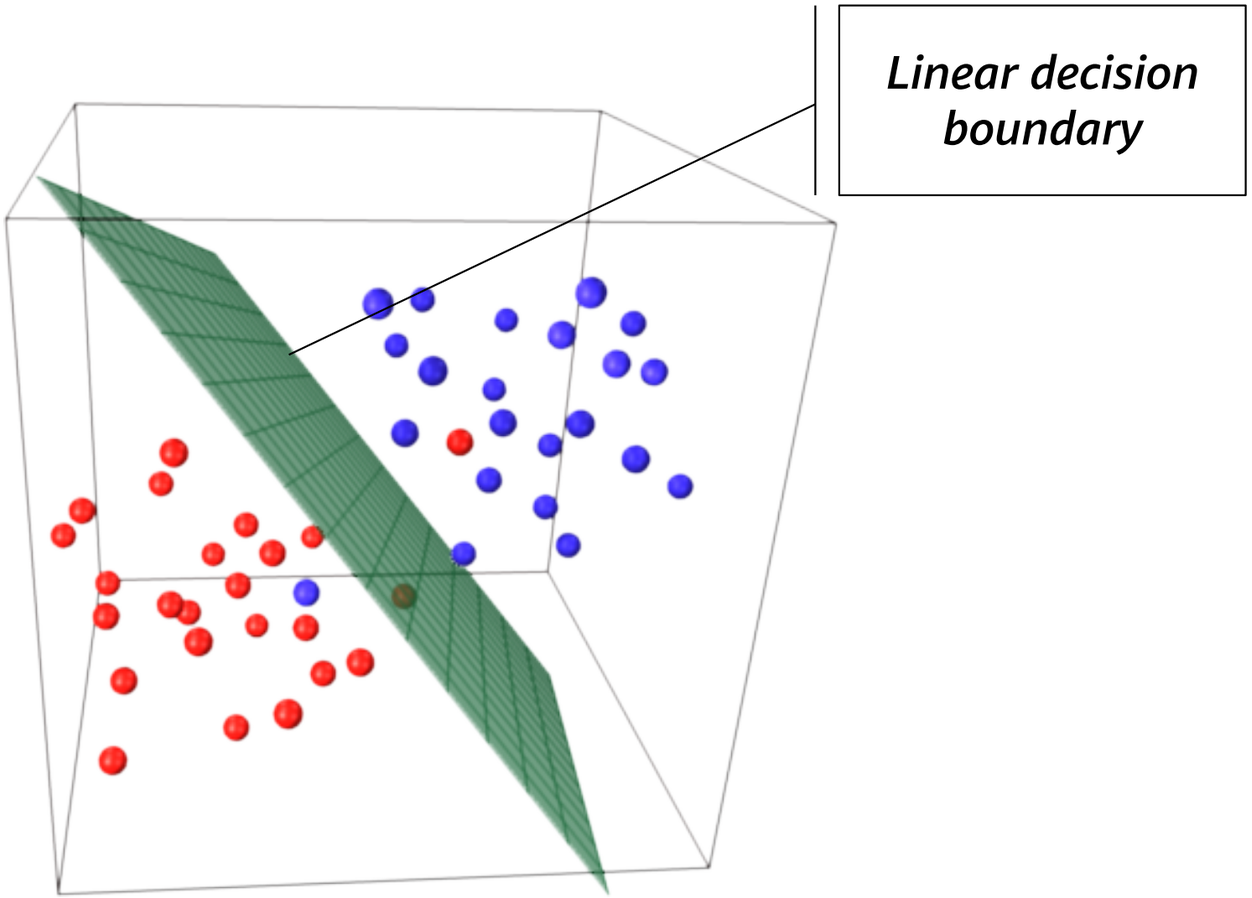
How to delete Studio | Delete Studio

Status	Studio ID	Execution role	Authentication method
Ready	d-pdovqdwnoub1	arn:aws:iam::184208354117:role/service-role/AmazonSageMaker-ExecutionRole-	AWS Identity and Access Management (IAM)

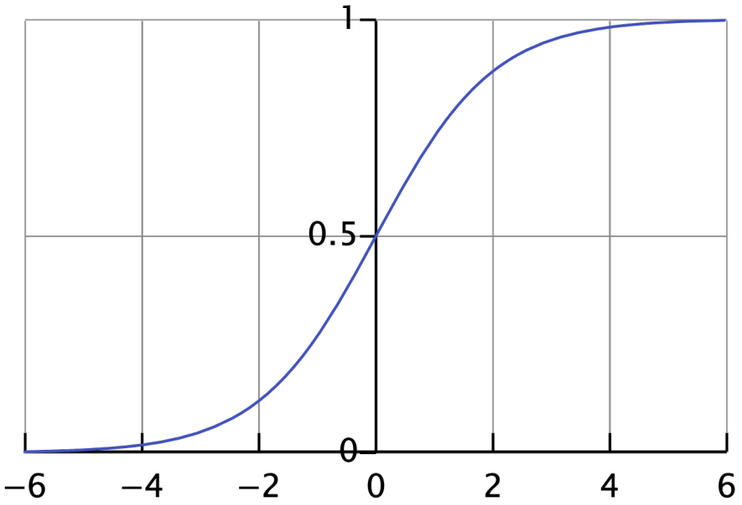
© 2008 - 2020, Amazon Web Services, Inc. or its affiliates. All rights reserved.

Guide for classification / regression algorithms

Linear learner



Logistic curve outputs values in 0.1

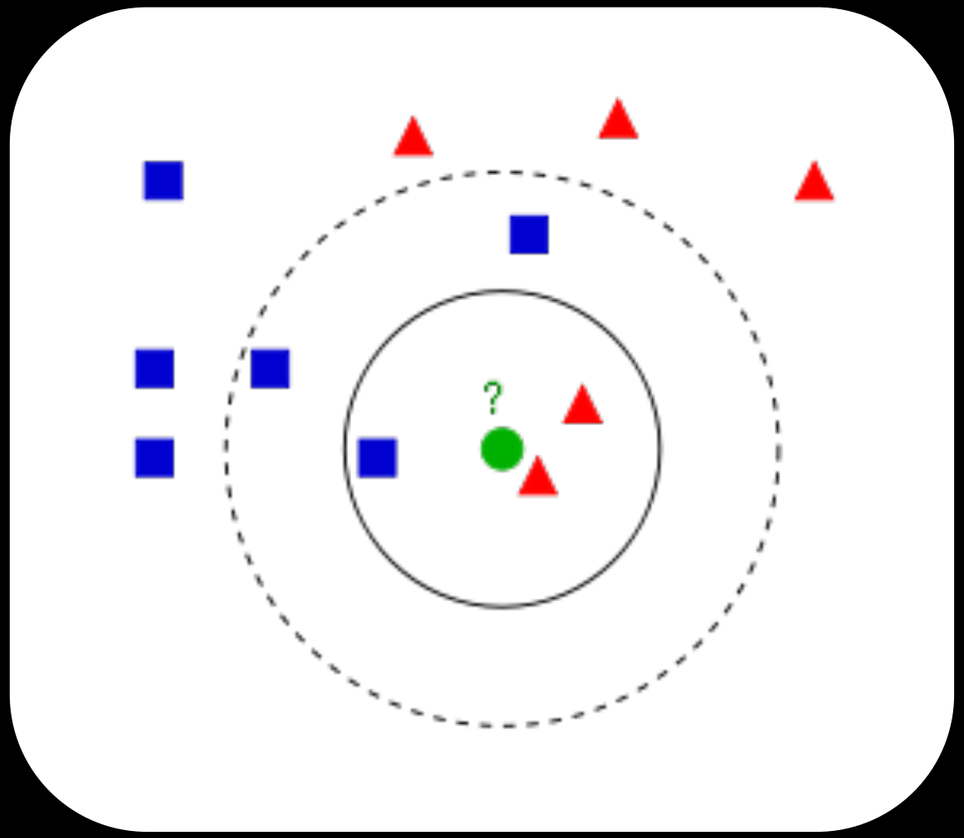


Amazon blog post on Linear Learner



Guide for classification / regression algorithms

k-NN



1. Sample size
2. Dimension reduction
    - Random projection
    - Johnson-Lindenstrauss transform
3. FAISS index – Flat, IVFFlat, IVFPQ

Amazon blog post on k-NN



Built-in XGBoost: How it works

Alberto Danese
Head of Data Science
Nexi



AWS re:Invent

© 2020, Amazon Web Services, Inc. or its affiliates. All rights reserved.

About me

Head of **data science** at Nexi (market leader in the Italian payment ecosystem)

Previously worked in credit risk (Cerved) and consulting (Between, EY)

M.Sc. in computer engineering (Politecnico di Milano, 2007)

Kaggle competitions Grandmaster since 2018 (https://www.kaggle.com/albedan)



Alberto Danese

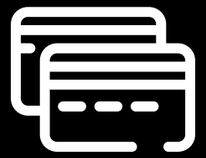
About Nexi

The leading Italian PayTech with full coverage of the payment ecosystem



Merchant services and solutions

1-stop solution provider for merchants of all categories and sizes



Cards and digital payments

Comprehensive portfolio leading toward complete digitalization of payments



Digital banking Solutions

Driving adoption of advanced banking solutions and developing Open Banking

About Nexi

Let’s talk **data**! In 2019, Nexi managed

- 900,000 merchants with 3.5 billion transactions
- 42.2 million cards with 2.6 billion transactions

Mostly tabular data

Amazon SageMaker is at the heart of all **advanced analytics** and **machine learning** projects

Built-in XGBoost is our **first choice** for many of our challenges, whether dealing with classification, regression, or ranking tasks

Getting a grasp on how XGBoost works

FIRST, THERE WERE (DECISION) TREES

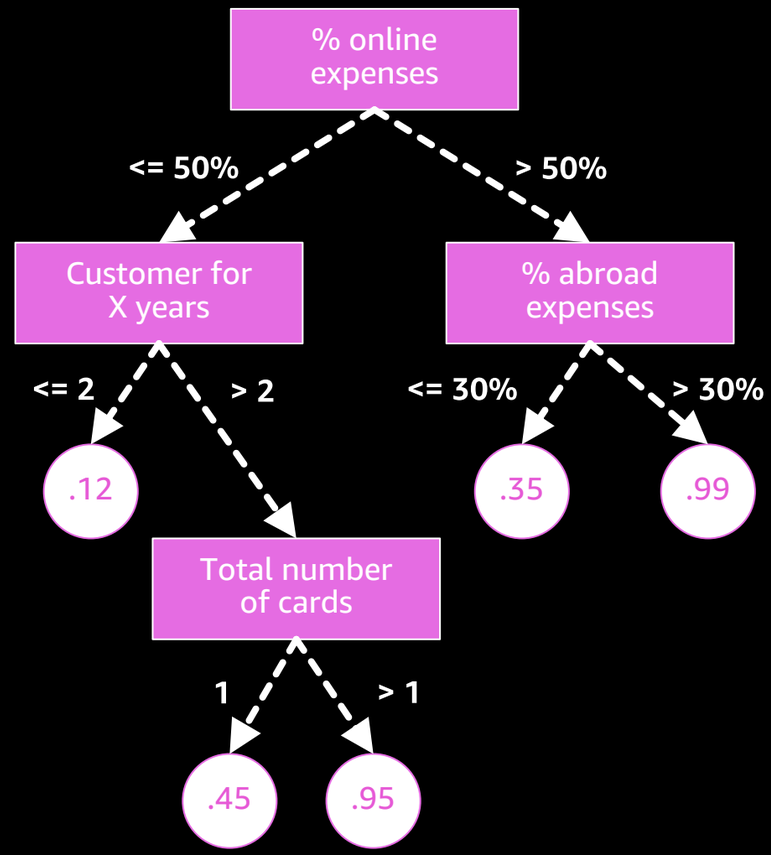
A **single decision** tree can be seen as a long list of nested *if – then – else* clauses

Training finds the best splits (according to different possible metrics) and handles missing values / outliers

Easy to understand, low predictive power by itself . . . but it's the building block of more complex architectures

PROPENSITY TO BUY PRODUCT X

WITH A DECISION TREE



Getting a grasp on how XGBoost works

FIRST, THERE WERE (DECISION) TREES

ID	% online expenses	Customer for X years	% abroad expenses	Total number of cards	Actual target	Predicted target	Residual
1	70	2	0	1	1	.35	.65
2	40	4	47	2	1	.95	.05
3	0	5	12	1	0	.45	-.45
4	95	15	70	3	1	.99	.01
5	32	1	12	4	0	.12	-.12
6	54	1	60	1	1	.99	.01



The *residual* is the difference between the actual target and the fit of the algorithm, i.e., the predicted target

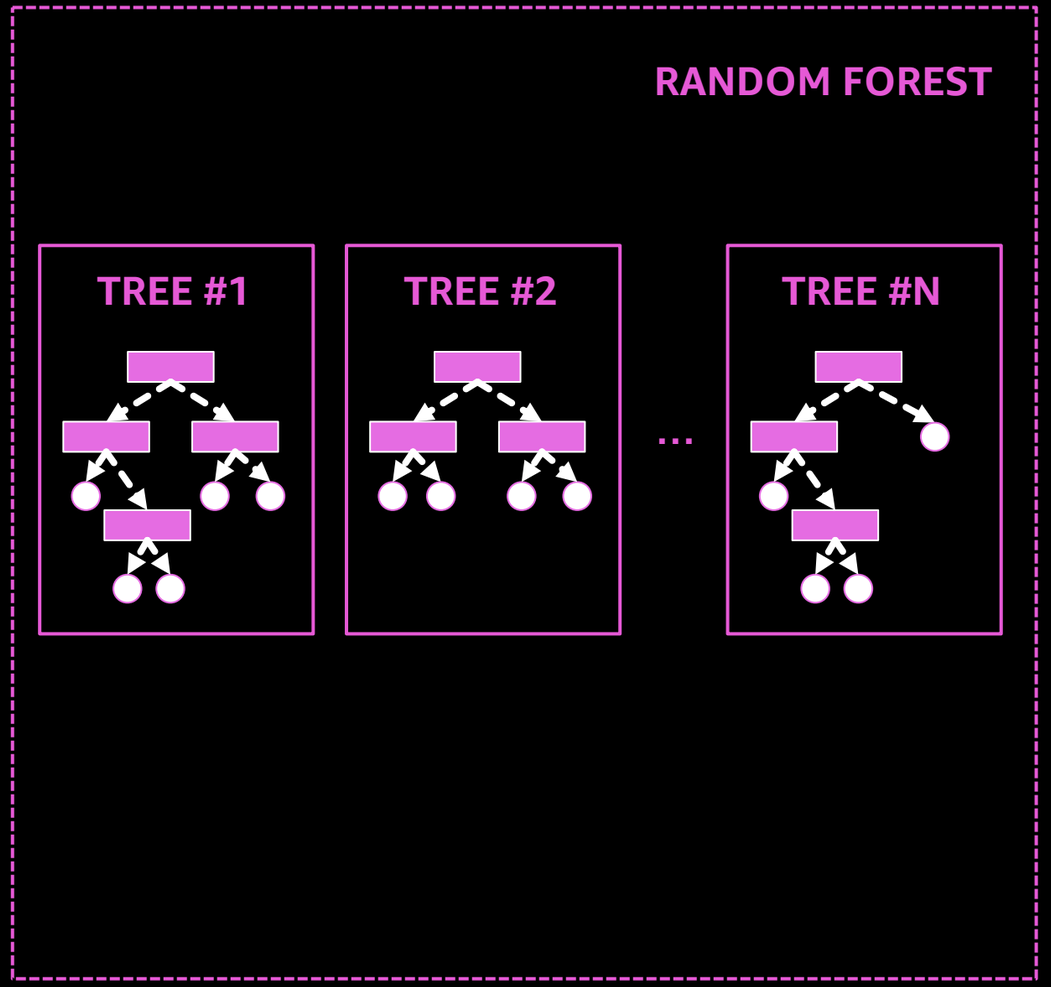
Getting a grasp on how XGBoost works

SECOND, THERE WERE (RANDOM) FORESTS

Random forests are the easiest way to create an *ensemble*

- Many trees are created in parallel
- Each is trained on a small **portion** of the training data
- Each ***weak*** *learner* provides a prediction
- Final fit is the result of a **voting** / average mechanism

Performance increases over single trees



Getting a grasp on how XGBoost works

SECOND, THERE WERE (RANDOM) FORESTS

TREE #1

ID	% online expenses	Customer for X years	% abroad expenses	Total number of cards	Actual target	Predicted target	Residual
1	70	2	0	1	1	…	…
2	40	4	47	2	1	…	…
3	0	5	12	1	0	…	…
4	95	15	70	3	1	…	…
5	32	1	12	4	0	…	…
6	54	1	60	1	1	…	…

TREE #2

Different trees *see* different portions of data (subsampling)

Getting a grasp on how XGBoost works

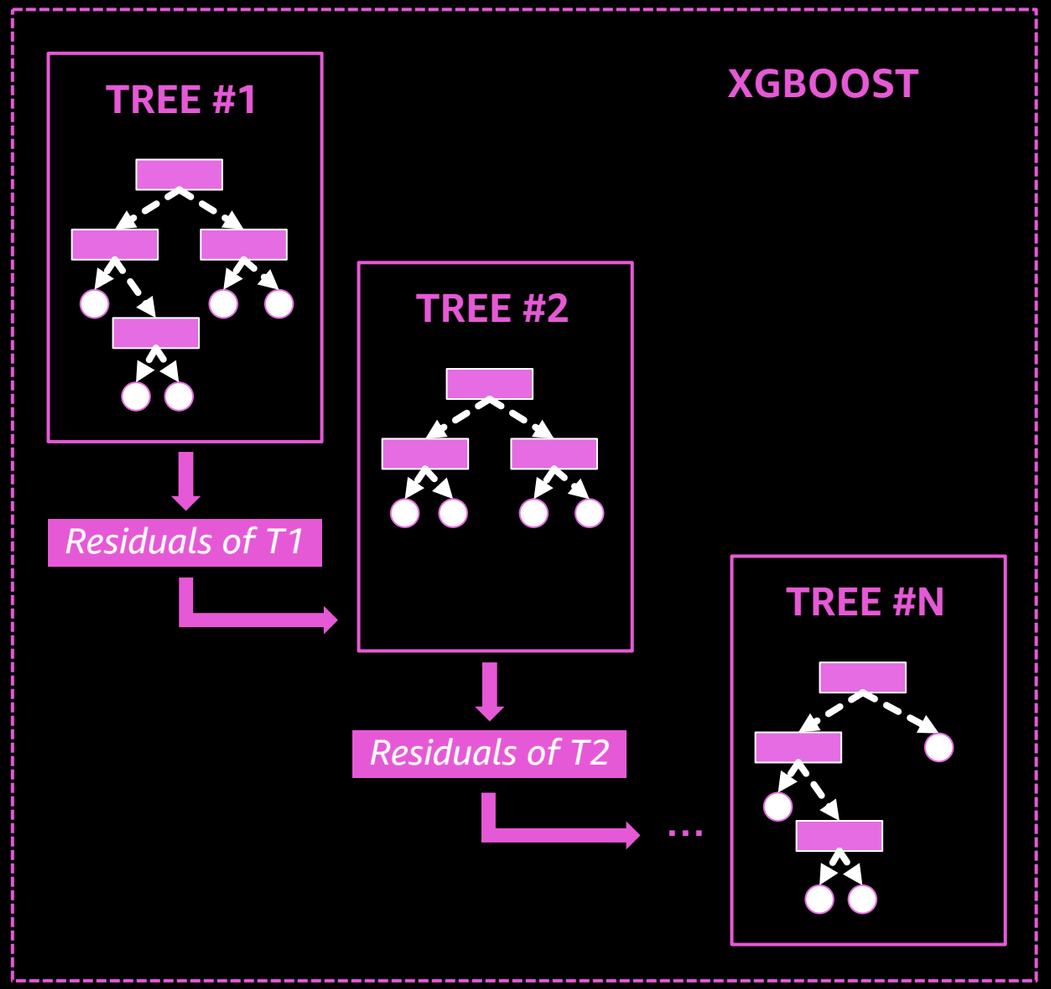
AND FINALLY, GRADIENT BOOSTING AND XGBOOST

XGBoost (an efficient implementation of gradient boosting) is a much more refined way to create an *ensemble*

- Many trees are created **sequentially**
- For each tree, the *residual* with respect to the actual target is computed and becomes the new *target* for the following tree
- This way, each tree learns how to improve on the previous trees' (classification / regression) mistakes

Predictive performance increases over forests

XGBoost scales well over dozens of cores



Getting a grasp on how XGBoost works

AND FINALLY, GRADIENT BOOSTING AND XGBOOST

The first tree is trained on the actual, original target

TREE #1

ID	% online expenses	Customer for X years	% abroad expenses	Total number of cards	Actual target	Predicted target	Residual
1	70	2	0	1	1	.35	.65
2	40	4	47	2	1	.95	.05
3	0	5	12	1	0	.45	-.45
4	95	15	70	3	1	.99	.01
5	32	1	12	4	0	.12	-.12
6	54	1	60	1	1	.99	.01

Subsampling is possible (and recommended) but not mandatory

Getting a grasp on how XGBoost works

AND FINALLY, GRADIENT BOOSTING AND XGBOOST

The second one deals with a different machine learning problem –
The features are the same, but the target becomes the *residuals* of the first fit

TREE #2

ID	% online expenses	Customer for X years	% abroad expenses	Total number of cards	Target from Tree #1	Predicted target	Residual
1	70	2	0	1	.65	…	…
2	40	4	47	2	.05	…	…
3	0	5	12	1	-.45	…	…
4	95	15	70	3	.01	…	…
5	32	1	12	4	-.12	…	…
6	54	1	60	1	.01	…	…

Final result is the sum of all the trees' predictions

Getting a grasp on how XGBoost works

A DEEPER UNDERSTANDING

1

- Every tree improves on the residuals of the previous ones
  - The **in-sample** accuracy will inevitably improve
  - Dealing with **overfitting** is crucial (different from random forests)
  - There's no point in using a **default** number of trees

- *Gradient boosting* is the way subsequent trees are fitted
  - One could do GB of **something different** from trees, like GB of linear regressions
  - Trees do have **inherent strengths** (e.g., handling missing values, outliers) that make GB tree implementations like XGBoost the *de facto* standard

- *Fitting the residuals* is the core of GB
  - But fitting **functions** of the residuals makes it possible to optimize for different metrics
  - XGBoost **natively** supports a large number of common predictive tasks and metrics (e.g., RMSE, AUC, log loss)

Getting a grasp on how XGBoost works

A DEEPER UNDERSTANDING

- Every tree improves on the residuals of the previous ones
  - The **in-sample** accuracy will inevitably improve
  - Dealing with **overfitting** is crucial (different from random forests)
  - There's no point in using a **default** number of trees

2

- *Gradient boosting* is the way subsequent trees are fitted
  - One could do GB of **something different** from trees, like GB of linear regressions
  - Trees do have **inherent strengths** (e.g., handling missing values, outliers) that make GB tree implementations like XGBoost the *de facto* standard

- *Fitting the residuals* is the core of GB
  - But fitting **functions** of the residuals makes it possible to optimize for different metrics
  - XGBoost **natively** supports a large number of common predictive tasks and metrics (e.g., RMSE, AUC, log loss)

Getting a grasp on how XGBoost works

A DEEPER UNDERSTANDING

- Every tree improves on the residuals of the previous ones
  - The **in-sample** accuracy will inevitably improve
  - Dealing with **overfitting** is crucial (different from random forests)
  - There's no point in using a **default** number of trees

- *Gradient boosting* is the way subsequent trees are fitted
  - One could do GB of **something different** from trees, like GB of linear regressions
  - Trees do have **inherent strengths** (e.g., handling missing values, outliers) that make GB tree implementations like XGBoost the *de facto* standard

3

- *Fitting the residuals* is the core of GB
  - But fitting **functions** of the residuals makes it possible to optimize for different metrics
  - XGBoost **natively** supports a large number of common predictive tasks and metrics (e.g., RMSE, AUC, log loss)

Getting a grasp on how XGBoost works

BOOSTING VS. TREES

The training of each weak learner (a tree) and the boosting process are two different phases – each with **its own parameters**

SINGLE-TREE TRAINING

- Max depth / number of leaves
- Rows and columns subsampling
- Positive scaling
- Different approaches to regularization

BOOSTING PROCESS

- Learning rate
- Number of iterations (i.e., trees)
- Early stopping

Getting a grasp on how XGBoost works

BOOSTING VS. TREES

The training of each weak learner (a tree) and the boosting process are two different phases – each with **its own parameters**

SINGLE-TREE TRAINING

- Max depth / number of leaves
- Rows and columns subsampling
- Positive scaling
- Different approaches to regularization

BOOSTING PROCESS

- Learning rate
- Number of iterations (i.e., trees)
- Early stopping

Getting a grasp on how XGBoost works

BOOSTING VS. TREES

The training of each weak learner (a tree) and the boosting process are two different phases – each with **its own parameters**

SINGLE-TREE TRAINING

- Max depth / number of leaves
- Rows and columns subsampling
- Positive scaling
- Different approaches to regularization

BOOSTING PROCESS

- Learning rate
- Number of iterations (i.e., trees)
- Early stopping

XGBoost as a built-in algorithm

SEAMLESS DISTRIBUTED HPO AND TRAINING

Distributing a training job on a cluster of multiple instances is trivial

Similarly, hyper-parameter optimization can leverage multiple instances with Bayesian Optimization and Random Search

```
# construct a SageMaker estimator that calls the xgboost-container
sm_estimator = sagemaker.estimator.Estimator(image_name=container,
                                             hyperparameters=hyperparameters,
                                             role=sagemaker.get_execution_role(),
                                             train_instance_count=10,
                                             train_instance_type='ml.m5.2xlarge',
                                             train_volume_size=5, # 5 GB
                                             output_path=output_path)
```

XGBoost as a built-in algorithm

API OR BATCH SCORING INTEGRATED INTO THE AWS ECOSYSTEM

Deploying trained XGBoost models

- Configuring and creating an HTTPS endpoint for on-demand inference
- Using batch transform (for large datasets, when no persistent endpoint is needed)

```
# Setting hyper-parameters, defining validation strategy, fitting
…
# Define and start batch transform job
sm_transformer = sm_estimator.transformer(1, 'ml.m4.xlarge', assemble_with = 'Line', accept = 'text/csv')
input_location = 's3://{}/{}/batch/{}'.format(bucket, prefix, batch_file) # use input data with ID column
sm_transformer.transform(input_location, split_type='Line', content_type='text/csv', input_filter='$[1:]',
join_source='Input', output_filter='$[0,-1]')
sm_transformer.wait()
```

https://aws.amazon.com/it/blogs/machine-learning/associating-prediction-results-with-input-data-using-amazon-sagemaker-batch-transform/

Using built-in XGBoost in Nexi

ONE OF OUR USE CASES

Task

Finding reliable and interested customers for a new premium product

Data

Customer and transactional data (behavioral)

Information about blocked cards for different reasons (e.g., insolvency)

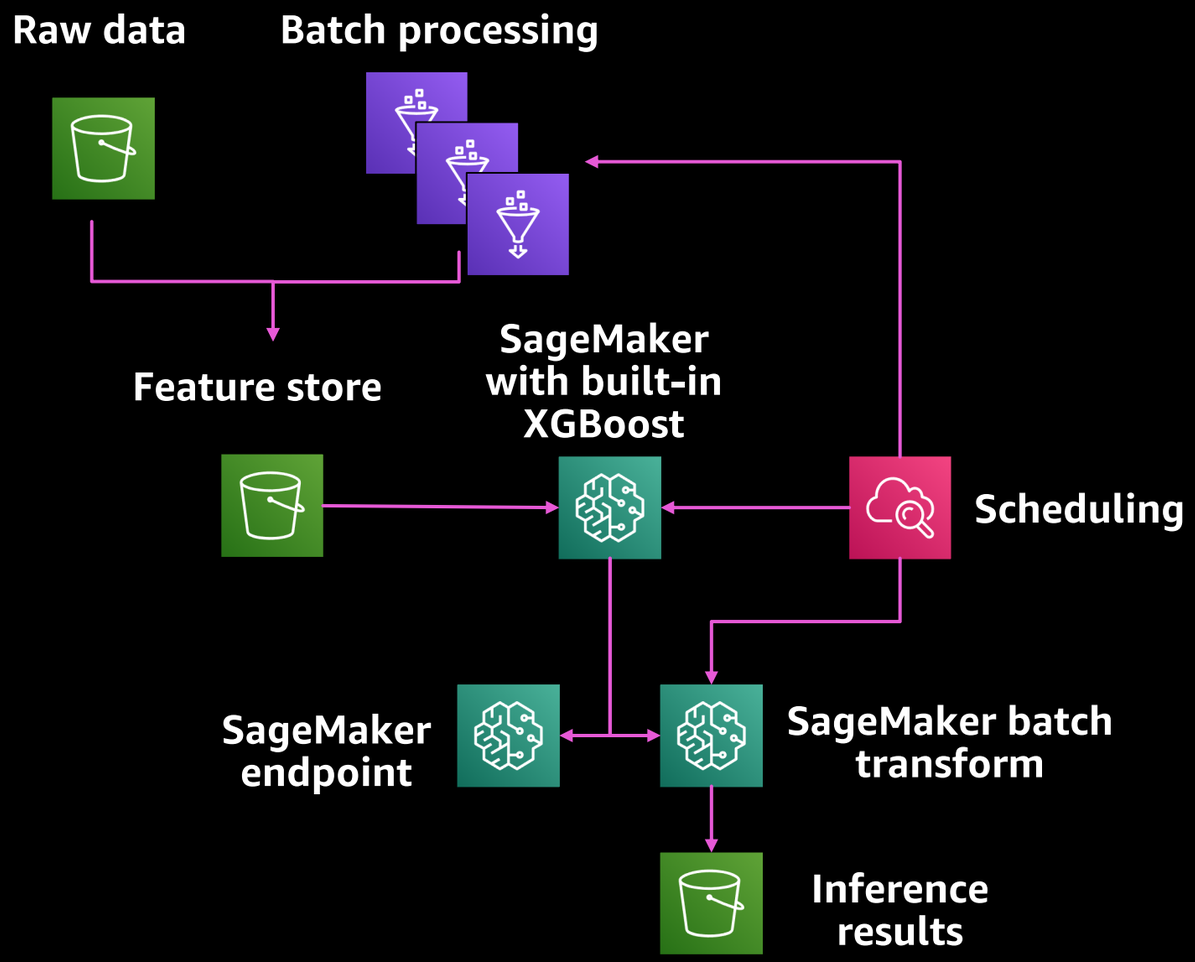
Approach

Binary classification problem with tabular data

Time variant problem with not-trivial validation scheme

Using built-in XGBoost in Nexi

HOW WE LEVERAGED BUILT-IN XGBOOST – ARCHITECTURE AND KEY ELEMENTS



Integration with Nexi's AWS architecture

Transactions are batch processed with AWS Glue in order to feed a feature store

Orchestration using Cloudwatch and Lambda functions

Training and distributed HPO

Heavily unbalanced task – proper use of hyperparameters is crucial (es. *scale positive weight*)

Distributed HPO with Bayesian optimization

Batch Transform with I/O join feature

Trained model used via API for demo purposes only

Actual use via scheduled Batch Transform with I/O join in order to have proper handling of customer IDs

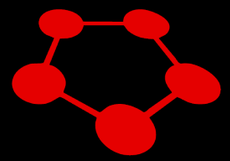
Popular frameworks and BYOC option



script mode







Chainer





Bring Your
Own Container

AWS Marketplace: Pretrained models

Prediction problem	Top AWS Marketplace models (use them as a simple API without ML knowledge)			
Text	Mphasis	Text summarization / sentiment analysis	7Park Data	Named Entity Recognition
	ElectrifAi	Contract clause extraction	quantiphi	Sentence classification
Computer vision	GTRIIP	GTRIIP Passport Page Detection	VITECH Lab	PPE detection
	DeepVision	Face / object recognition	CVEDIA	CVEDIA vehicle and human detection
	Provectus	Worker safety models	TensorIoT	PPE detection
Customer segmentation	experian	Customer segmentation (accessible on request only)	Prosper Insights & Analytics	Various customer segmentation models
	Perception Health	Customer segmentation based on vulnerability to a disease	ClosedLoop	Customer segmentation based on vulnerability to a disease

AWS Marketplace: Algorithms

Prediction problem	Amazon SageMaker built-in	Top AWS Marketplace algorithms			
Classification	• Linear Learner • XGBoost • k-NN • Factorization Machines	H2O.ai	H2O.ai H2O-3 AutoML algorithm	PEAK	Customer churn classification
		Intel	Intel DAAL k-Nearest Neighbors	Mphasis The Next Applied	Ticket classification
		AutoGluon	AutoGluon Tabular	sensifai	Computer vision / audio classification
Regression	• Linear Learner • XGBoost • k-NN	RocketML	Sparse / Dense Random Forests and GB regression	H2O.ai	H2O.ai H2O-3 GBM algorithm
Anomaly detection	• IP Insights • Random Cut Forest	bigfinite	Bigfinite OutlierDetection	TIBCO	Autoencoder for anomaly detection
Forecasting	• DeepAR	PEAK	Demand forecasting	VITech Lab	Retail store product demand forecasting

17 built-in algorithms in Amazon SageMaker

LDA	Linear Learner	K-Means	Image Classification
NTM	k-NN	PCA	Object Detection
BlazingText	XGBoost	Random Cut Forest	Semantic Segmentation
Sequence to Sequence	DeepAR	IP Insights	Object2Vec	Factorization Machines

17 built-in algorithms in Amazon SageMaker

LDA	Linear Learner	K-Means	Image Classification
NTM	k-NN	PCA	Object Detection
BlazingText	XGBoost	Random Cut Forest	Semantic Segmentation
Sequence to Sequence	DeepAR	IP Insights	Object2Vec	Factorization Machines

- Documentation
- Hyperparameter guidance
- Instance type guidance
- Sample notebooks



Thank you!



AWS re:Invent

© 2020, Amazon Web Services, Inc. or its affiliates. All rights reserved.



Please complete the session survey

AWS re:Invent

© 2020, Amazon Web Services, Inc. or its affiliates. All rights reserved.