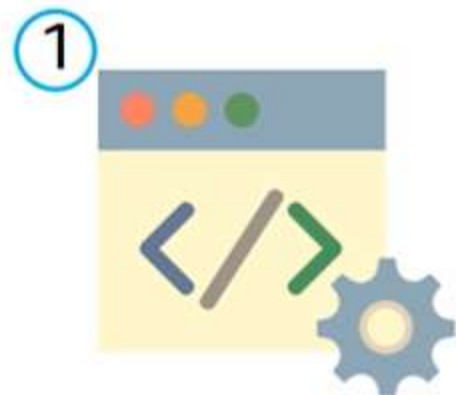
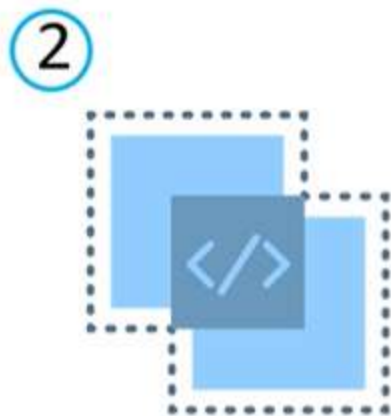


# Amazon SageMaker Components



Amazon  
SageMaker  
Notebooks  
Service



Amazon  
SageMaker  
Training  
Service



Amazon  
SageMaker  
Hosting  
Service

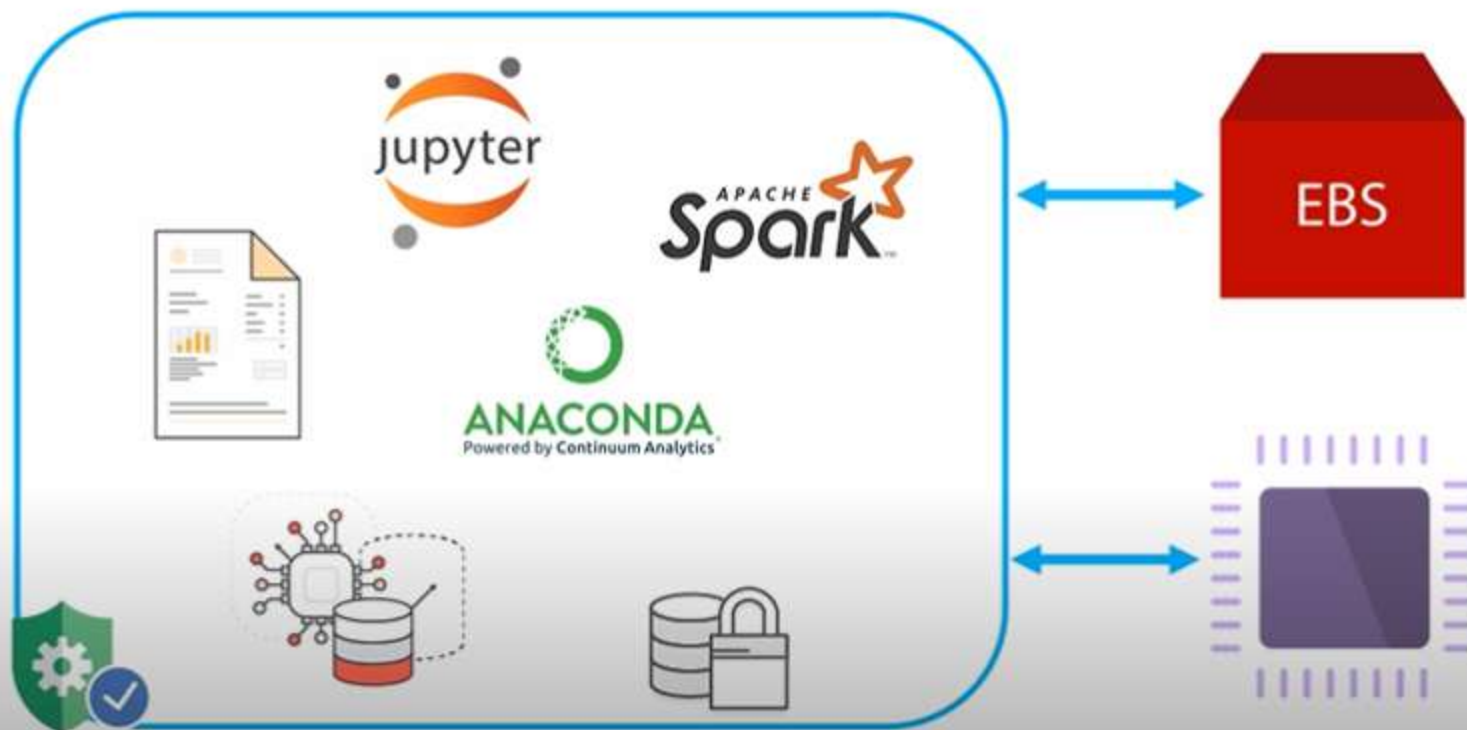
elastic, scalable, secure, and reliable.

# Zero Setup for Exploratory Data Analysis

1



Notebooks

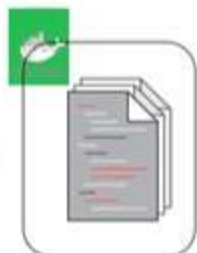


# High Performance, On-Demand



High on-demand training environment performance

2



Training code



Amazon SageMaker  
Algorithms



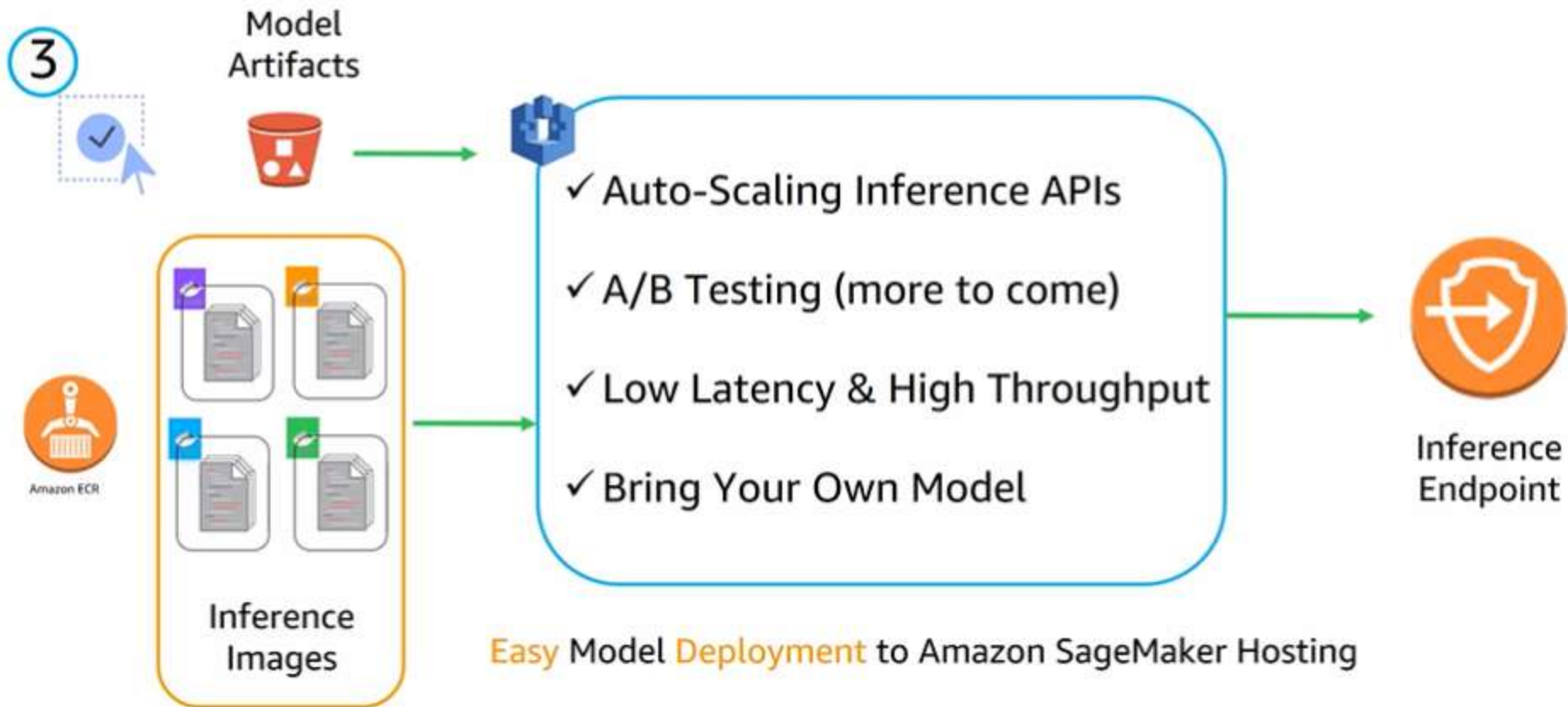
Bring Your Own  
Algorithm



02:07 / 12:47



# Easy Model Deployment



us-west-2.console.aws.amazon.com

us-west-2.console.aws.amazon.com

Services

Resource Groups

Amazon SageMaker

Amazon SageMaker > Notebook instances

Dashboard

Notebook instances

Jobs

Resources

Models

Endpoint configuration

Endpoints

Notebook instances

Open

Start

Update settings

Actions

Create notebook instance

Search notebook instances

< 1 >

Name	Instance	Creation time	Status	Actions
empty				

Feedback

English (US)

© 2006 - 2017, Amazon Web Services, Inc. or its affiliates. All rights reserved. Privacy Policy Terms of Use

us-west-2.console.aws.amazon.com

ServicesResource Groups

Amazon SageMaker

Dashboard

Notebook instances

Jobs

Resources

Models

Endpoint configuration

Endpoints

Amazon SageMaker > Notebook instances > Create notebook instance

Create notebook instance

Notebook instance settings

Notebook instance name

Instance type

ml.t2.medium

IAM role ARN

Notebook instances need permissions to call AWS services, including Amazon SageMaker and Amazon S3.

VPC - optional

Encryption key - optional

An encryption key protects your data. Type the ID or ARN of the AWS KMS key that you want to use.

Cancel

Create notebook instance

Feedback

English (US)

© 2006 - 2017, Amazon Web Services, Inc. or its affiliates. All rights reserved. Privacy Policy Terms of Use

us-west-2.console.aws.amazon.com

ServicesResource Groups

Amazon SageMaker

Dashboard

Notebook instances

Jobs

Resources

Models

Endpoint configuration

Endpoints

Amazon SageMaker > Notebook instances > Create notebook instance

Create notebook instance

Notebook instance settings

Notebook instance name

mynotebookinst

Instance type

ml.t2.medium

IAM role ARN

Notebook instances need permissions to call AWS services, including Amazon SageMaker and Amazon S3.

VPC - optional

Encryption key - optional

An encryption key protects your data. Type the ID or ARN of the AWS KMS key that you want to use.

CancelCreate notebook instance

FeedbackEnglish (US)

© 2006 – 2017, Amazon Web Services, Inc. or its affiliates. All rights reserved. Privacy PolicyTerms of Use

us-west-2.console.aws.amazon.com

ServicesResource Groups

Amazon SageMaker

Dashboard

Notebook instances

Jobs

Resources

Models

Endpoint configuration

Endpoints

Amazon SageMaker > Notebook instances > Create notebook instance

Create notebook instance

Notebook instance settings

Notebook instance name

mynotebookinstance-1

Instance type

ml.t2.medium

ml.m4.xlarge

ml.t2.medium

ml.p2.xlarge

VPC - optional

Encryption key - optional

An encryption key protects your data. Type the ID or ARN of the AWS KMS key that you want to use.

Cancel

Create notebook instance

Feedback

English (US)

© 2006 - 2017, Amazon Web Services, Inc. or its affiliates. All rights reserved. Privacy Policy Terms of Use



us-west-2.console.aws.amazon.com

ServicesResource Groups

Amazon SageMaker

Dashboard

Notebook instances

Jobs

Resources

Models

Endpoint configuration

Endpoints

Amazon SageMaker > Notebook instances > Create notebook instance

Create notebook instance

Notebook instance settings

Notebook instance name

mynotebookinstance-1

Instance type

ml.t2.medium

ml.m4.xlarge

ml.t2.medium

ml.p2.xlarge

VPC - optional

Encryption key - optional

An encryption key protects your data. Type the ID or ARN of the AWS KMS key that you want to use.

Cancel

Create notebook instance

Feedback

English (US)

© 2006 - 2017, Amazon Web Services, Inc. or its affiliates. All rights reserved. Privacy Policy Terms of Use

Amazon SageMaker

Dashboard

Notebook instances

Jobs

Resources

Models

Endpoint configuration

Endpoints

Amazon SageMaker > Notebook instances > Create notebook instance

## Create notebook instance

### Notebook instance settings

Notebook instance name

mynotebookinstance-1

Instance type

ml.t2.medium

IAM role ARN

Notebook instances need permissions to call AWS services, including Amazon SageMaker and Amazon S3.

VPC - optional

Encryption key - optional

An encryption key protects your data. Type the ID or ARN of the AWS KMS key that you want to use.

Cancel

Create notebook instance

us-west-2.console.aws.amazon.com

ServicesResource Groups

Amazon SageMaker

Dashboard

Notebook instances

Jobs

Resources

Models

Endpoint configuration

Endpoints

us-west-2.console.aws.amazon.com

Support

Amazon SageMaker > Notebook instances > Create notebook instance

## Create notebook instance

### Notebook instance settings

Notebook instance name

mynotebookinstance-1

Instance type

ml.t2.medium

IAM role ARN

Notebook instances need permissions to call AWS services, including Amazon SageMaker and Amazon S3.

arn:aws:iam::123456789012:role/sagemakerrole

VPC - optional

Encryption key - optional

An encryption key protects your data. Type the ID or ARN of the AWS KMS key that you want to use.

Cancel

Create notebook instance

Feedback

English (US)

© 2008 - 2017, Amazon Web Services, Inc. or its affiliates. All rights reserved. Privacy Policy Terms of Use

us-west-2.console.aws.amazon.com

ServicesResource Groups

Amazon SageMaker

Dashboard

Notebook instances

Jobs

Resources

Models

Endpoint configuration

Endpoints

Amazon SageMaker > Notebook instances > Create notebook instance

Create notebook instance

Notebook instance settings

Notebook instance name

mynotebookinstance-1

Instance type

ml.t2.medium

IAM role ARN

Notebook instances need permissions to call AWS services, including Amazon SageMaker and Amazon S3.

arn:aws:iam::123456789012:role/sagemakerrole

VPC - optional

Encryption key - optional

An encryption key protects your data. Type the ID or ARN of the AWS KMS key that you want to use.

CancelCreate notebook instance

FeedbackEnglish (US)© 2006 - 2017, Amazon Web Services, Inc. or its affiliates. All rights reserved. Privacy PolicyTerms of Use

us-west-2.console.aws.amazon.com

ServicesResource Groups

Amazon SageMaker

Dashboard

Notebook instances

Jobs

Resources

Models

Endpoint configuration

Endpoints

Amazon SageMaker > Notebook instances > Create notebook instance

## Create notebook instance

### Notebook instance settings

Notebook instance name

mynotebookinstance-1

Instance type

ml.t2.medium

IAM role ARN

Notebook instances need permissions to call AWS services, including Amazon SageMaker and Amazon S3.

arn:aws:iam::123456789012:role/sagemakerrole

VPC - optional

Default vpc-ac5429c8 (172.31.0.0/16)

Subnet - optional

Security group(s) - optional

Encryption key - optional

An encryption key protects your data. Type the ID or ARN of the AWS KMS key that you want to use.

FeedbackEnglish (US)

© 2006 - 2017, Amazon Web Services, Inc. or its affiliates. All rights reserved. Privacy PolicyTerms of Use

us-west-2.console.aws.amazon.com

Services

Resource Groups

Amazon SageMaker

Dashboard

Notebook instances

Jobs

Resources

Models

Endpoint configuration

Endpoints

us-west-2.console.aws.amazon.com

Support

Amazon SageMaker

Amazon SageMaker > Notebook instances > Create notebook instance

Create notebook instance

Notebook instance settings

Notebook instance name

mynotebookinstance-1

Instance type

ml.t2.medium

IAM role ARN

arn:aws:iam::123456789012:role/sagemakerrole

VPC - optional

sg-0f90c672 (ElasticMapReduce-master)

sg-951271ef (launch-wizard-1)

sg-cd1885ab (default)

sg-d89ccaa5 (ElasticMapReduce-slave)

Encryption key - optional

An encryption key protects your data. Type the ID or ARN of the AWS KMS key that you want to use.

Feedback

English (US)

© 2006 - 2017, Amazon Web Services, Inc. or its affiliates. All rights reserved. Privacy Policy Terms of Use

us-west-2.console.aws.amazon.com

ServicesResource Groups

Amazon SageMaker

Dashboard

Notebook instances

Jobs

Resources

Models

Endpoint configuration

Endpoints

Amazon SageMaker > Notebook instances > Create notebook instance

Create notebook instance

Notebook instance settings

Notebook instance name

mynotebookinstance-1

Instance type

ml.t2.medium

IAM role ARN

Notebook instances need permissions to call AWS services, including Amazon SageMaker and Amazon S3.

arn:aws:iam::123456789012:role/sagemakerrole

VPC - optional

Default vpc-ac5429c8 (172.31.0.0/16)

Subnet - optional

subnet-932d40f7 (172.31.16.0/20) | us-west-2b

Security group(s) - optional

sg-cd1885ab (default)

Encryption key - optional

An encryption key protects your data. Type the ID or ARN of the AWS KMS key that you want to use.

Feedback

English (US)

© 2008 - 2017, Amazon Web Services, Inc. or its affiliates. All rights reserved. Privacy Policy Terms of Use

## Amazon SageMaker

Dashboard

Notebook instances

Jobs

Resources

Models

Endpoint configuration

Endpoints

## Create notebook instance

## Notebook instance settings

Notebook instance name

mynotebookinstance-1

Instance type

ml.t2.medium

IAM role ARN

Notebook instances need permissions to call AWS services, including Amazon SageMaker and Amazon S3.

arn:aws:iam::111111111111:role/sagemakerrole

VPC - optional

Default vpc-ac5429c8 (172.31.0.0/16)

Subnet - optional

subnet-932d40f7 (172.31.16.0/20) | us-west-2b

Security group(s) - optional

sg-cd1885ab (default)

Encryption key - optional

An encryption key protects your data. Type the ID or ARN of the AWS KMS key that you want to use.

Cancel

Create notebook instance



## Amazon SageMaker

Dashboard

Notebook instances

Jobs

Resources

Models

Endpoint configuration

Endpoints

## Create notebook instance

## Notebook instance settings

Notebook instance name

mynotebookinstance-1

Instance type

ml.t2.medium

IAM role ARN

Notebook instances need permissions to call AWS services, including Amazon SageMaker and Amazon S3.

arn:aws:iam::123456789012:role/sagemakerrole

VPC - optional

Default vpc-ac5429c8 (172.31.0.0/16)

Subnet - optional

subnet-932d40f7 (172.31.16.0/20) | us-west-2b

Security group(s) - optional

sg-cd1885ab (default) X

Encryption key - optional

An encryption key protects your data. Type the ID or ARN of the AWS KMS key that you want to use.

Cancel

Create notebook instance

us-west-2.console.aws.amazon.com

ServicesResource Groups

Support

Amazon SageMaker

Dashboard

Notebook instances

Jobs

Resources

Models

Endpoint configuration

Endpoints

Success! You created your notebook instance.

Open the notebook instance when status is InService and open a template notebook to get started.

View details

Amazon SageMaker

Notebook instances

Notebook instances

OpenStartUpdate settingsActions

Create notebook instance

Search notebook instances

<1>

Name	Instance	Creation time	Status	Actions
mynotebookinstance-1	ml.t2.medium	Nov 18, 2017 23:41 UTC	Pending	

Feedback

English (US)

© 2006 – 2017, Amazon Web Services, Inc. or its affiliates. All rights reserved. Privacy Policy Terms of Use

us-west-2.console.aws.amazon.com

us-west-2.console.aws.amazon.com

us-west-2.console.aws.amazon.com

Services

Resource Groups

us-west-2.console.aws.amazon.com

us-west-2.console.aws.amazon.com

us-west-2.console.aws.amazon.com

us-west-2.console.aws.amazon.com

us-west-2.console.aws.amazon.com

us-west-2.console.aws.amazon.com

Amazon SageMaker

Dashboard

Notebook instances

Jobs

Resources

Models

Endpoint configuration

Endpoints

Amazon SageMaker

Notebook instances

Notebook instances

Open

Start

Update settings

Actions

Create notebook instance

Search notebook instances

< 1 >

Name	Instance	Creation time	Status	Actions
mynotebookinstance-1	ml.t2.medium	Nov 18, 2017 23:41 UTC	InService	<a href="#">Stop</a>   <a href="#">Open</a>

Feedback

English (US)

© 2006 - 2017, Amazon Web Services, Inc. or its affiliates. All rights reserved. Privacy Policy Terms of Use

us-west-2.console.aws.amazon.com

ServicesResource Groups

Amazon SageMaker

Dashboard

Notebook instances

Jobs

Resources

Models

Endpoint configuration

Endpoints

us-west-2.console.aws.amazon.com

Support

Amazon SageMaker > Notebook instances



Notebook instances

OpenStartUpdate settingsActions

Create notebook instance

Search notebook instances

< 1 >

Name	Instance	Creation time	Status	Actions
 mynotebookinstance-1	ml.t2.medium	Nov 18, 2017 23:41 UTC	 InService	Stop   Open

Feedback

English (US)

© 2006 - 2017, Amazon Web Services, Inc. or its affiliates. All rights reserved. Privacy PolicyTerms of Use



Files

Running

## Clusters

Conda

Select items to perform actions on them.

Never 

Name 

Last Modified

lost+found

an hour ago

sample-notebooks

an hour ago



Files

### Running

### Clusters

Conda

Select items to perform actions on them.

Upload

Never ☐

□ □

Name 

Last Modified

2

lost+found

an hour ago

sample-notebooks

an hour ago



Files

### Running

### Clusters

Conda

Select items to perform actions on them.

Printer: 

Name 

Last Modified

2

lost+found

an hour ago

sample-notebooks

an hour ago



Files Running Clusters Conda

Select items to perform actions on them.

Upload New ↻

0 / sample-notebooks		Name ↕	Last Modified
<input type="checkbox"/>	..		seconds ago
<input type="checkbox"/>	build_your_own		18 hours ago
<input type="checkbox"/>	in-imageclassification		18 hours ago
<input type="checkbox"/>	in-seq2seq		18 hours ago
<input type="checkbox"/>	in-xgboost		18 hours ago
<input type="checkbox"/>	install_r_kernel		18 hours ago
<input type="checkbox"/>	kmeans		20 minutes ago
<input type="checkbox"/>	kmeans_bring_your_own_model		19 hours ago
<input type="checkbox"/>	linear_time_series_forecast		19 hours ago
<input type="checkbox"/>	pca_kmeans_movie_clustering		18 hours ago
<input type="checkbox"/>	r_backup		12 minutes ago
<input type="checkbox"/>	r_bring_your_own		18 hours ago
<input type="checkbox"/>	sagemaker-python-sdk		19 hours ago
<input type="checkbox"/>	scripts		18 hours ago
<input type="checkbox"/>	tmp		3 minutes ago
<input type="checkbox"/>	xgboost_customer_churn		18 hours ago
<input type="checkbox"/>	xgboost_direct_marketing		3 minutes ago
<input type="checkbox"/>	README.md		16 hours ago





Files Running Clusters Conda

Select items to perform actions on them.

Upload New ↻

0 / sample-notebooks		Name ↓	Last Modified
<input type="checkbox"/>	..		seconds ago
<input type="checkbox"/>	build_your_own		18 hours ago
<input type="checkbox"/>	im-imageclassification		18 hours ago
<input type="checkbox"/>	im-seqseq		18 hours ago
<input type="checkbox"/>	im-xgboost		18 hours ago
<input type="checkbox"/>	install_r_kernel		18 hours ago
<input type="checkbox"/>	kmeans		20 minutes ago
<input type="checkbox"/>	kmeans_bring_your_own_model		19 hours ago
<input type="checkbox"/>	linear_time_series_forecast		19 hours ago
<input type="checkbox"/>	pca_kmeans_movie_clustering		18 hours ago
<input type="checkbox"/>	r_backup		12 minutes ago
<input type="checkbox"/>	r_bring_your_own		18 hours ago
<input type="checkbox"/>	sagemaker-python-sdk		19 hours ago
<input type="checkbox"/>	scripts		18 hours ago
<input type="checkbox"/>	tmp		3 minutes ago
<input type="checkbox"/>	xgboost_customer_churn		18 hours ago
<input type="checkbox"/>	xgboost_direct_marketing		3 minutes ago
<input type="checkbox"/>	README.md		18 hours ago



Files Running Clusters Conda

Select items to perform actions on them.

Upload New ↻

0 / sample-notebooks		Name ↕	Last Modified
<input type="checkbox"/>	..		seconds ago
<input type="checkbox"/>	build_your_own		18 hours ago
<input type="checkbox"/>	im-imageclassification		18 hours ago
<input type="checkbox"/>	im-seq2seq		18 hours ago
<input type="checkbox"/>	im-xgboost		18 hours ago
<input type="checkbox"/>	install_r_kernel		18 hours ago
<input type="checkbox"/>	kmeans		20 minutes ago
<input type="checkbox"/>	kmeans_bring_your_own_model		19 hours ago
<input type="checkbox"/>	linear_time_series_forecast		19 hours ago
<input type="checkbox"/>	pca_kmeans_movie_clustering		18 hours ago
<input type="checkbox"/>	r_backup		12 minutes ago
<input type="checkbox"/>	r_bring_your_own		18 hours ago
<input type="checkbox"/>	sagemaker-python-sdk		19 hours ago
<input type="checkbox"/>	scripts		18 hours ago
<input type="checkbox"/>	tmp		3 minutes ago
<input type="checkbox"/>	xgboost_customer_churn		18 hours ago
<input type="checkbox"/>	xgboost_direct_marketing		3 minutes ago
<input type="checkbox"/>	README.md		18 hours ago



Files Running Clusters Conda

Select items to perform actions on them.

Upload New ↻

0 / sample-notebooks / xgboost_direct_marketing			Name ↓	Last Modified
..				seconds ago
xgboost_direct_marketing.ipynb				19 hours ago
xgboost_direct_marketing_sagemaker.ipynb			Running	4 minutes ago
README.md				19 hours ago



mynotebookinstance-1.notebook.us-west-2.sagemaker.aws

Amazon SageMaker

sample-notebooks/xgboost\_direct\_marketing/

xgboost\_direct\_marketing\_sagemaker

jupyter

xgboost\_direct\_marketing\_sagemaker

Last Checkpoint: 5 minutes ago (autosaved)

Python 3

File Edit View Insert Cell Kernel Widgets Help

Trusted

Run

Markdown

Edit Presentation

Show Presentation

# Targeting Direct Marketing with Amazon SageMaker XGBoost

*Supervised Learning with Gradient Boosted Trees: A Binary Prediction Problem With Unbalanced Classes*

## Contents

1. [Background](#)
2. [Preparation](#)
3. [Data](#)
  - A. [Exploration](#)
  - B. [Transformation](#)
4. [Training](#)
5. [Hosting](#)
6. [Evaluation](#)

## Background

Direct marketing, either through mail, email, phone, etc., is a common tactic to acquire customers. Because resources and a customer's attention is limited, the goal is to only target the subset of prospects who are likely to engage with a specific offer. Predicting those potential customer's based on readily available information like demographics, past interactions, and environmental factors is a common machine learning problem.

This notebook presents an example problem to predict if a customer will enroll for a term deposit at a bank, after one or more phone calls. The steps include:

- Preparing your Amazon SageMaker notebook
- Downloading data from the internet into Amazon SageMaker
- Investigating and transforming the data so that it can be fed to Amazon SageMaker algorithms
- Estimating a model using the Gradient Boosting algorithm
- Evaluating the effectiveness of the model
- Setting the model up to make on-going predictions

### Preparation

To start, let's define:

- IAM role for training/hosting to access resources
- The name of the S3 bucket we want to use
- An S3 prefix

```
In [1]: import os
import boto3

role = 'arn:aws:iam::111111111111:role/sagemakerrole'
bucket = 'sagemaker-pdx1'
prefix = 'xgboost'
```

Now let's bring in the Python libraries that we'll use throughout the analysis

```
In [2]: import numpy as np          # For matrix operations and numerical processing
import pandas as pd              # For handling tabular data
import matplotlib.pyplot as plt  # For charts and visualizations
from IPython.display import Image # For displaying images in the notebook
from IPython.display import display # For displaying outputs in the notebook
from sklearn.datasets import dump_svmlight_file # For outputting data to libsvm format for xgboost
from time import gtime, strftime # For labeling IN models, endpoints, etc.
import sys                      # For writing outputs to notebook
import math                     # For ceiling function
import json                     # For parsing hosting outputs
```

## Data

Let's start by downloading a dataset from UCI's ML Repository.

```
In [ ]: wget https://archive.ics.uci.edu/ml/machine-learning-databases/00222/bank-additional.zip
        unzip bank-additional.zip
```

### Preparation

To start, let's define:

- IAM role for training/hosting to access resources
- The name of the S3 bucket we want to use
- An S3 prefix

```
In [1]: import os
import boto3

role = 'arn:aws:iam::111111111111:role/sagemakerrole'
bucket = 'sagemaker-pdxi'
prefix = 'xgboost'
```

Now let's bring in the Python libraries that we'll use throughout the analysis

```
In [2]: import numpy as np          # For matrix operations and numerical processing
import pandas as pd              # For munging tabular data
import matplotlib.pyplot as plt  # For charts and visualizations
from IPython.display import Image # For displaying images in the notebook
from IPython.display import display # For displaying outputs in the notebook
from sklearn.datasets import dump_svmlight_file # For outputting data to libsvm format for xgboost
from time import gmtime, strftime # For labeling IM models, endpoints, etc.
import sys                      # For writing outputs to notebook
import math                     # For ceiling function
import json                     # For parsing hosting outputs
```

## Data

Let's start by downloading a dataset from UCI's ML Repository.

```
In [ ]: wget https://archive.ics.uci.edu/ml/machine-learning-databases/00222/bank-additional.zip
unzip bank-additional.zip
```

### Preparation

To start, let's define:

- IAM role for training/hosting to access resources
- The name of the S3 bucket we want to use
- An S3 prefix

```
In [12]: import os
import boto3

role = 'arn:aws:iam::9-666666666666:role/sagemakerrole'
bucket = 'sagemaker-pdx1'
prefix = 'xgboost'
```

Now let's bring in the Python libraries that we'll use throughout the analysis

```
In [2]: import numpy as np          # For matrix operations and numerical processing
import pandas as pd              # For munging tabular data
import matplotlib.pyplot as plt  # For charts and visualizations
from IPython.display import Image # For displaying images in the notebook
from IPython.display import display # For displaying outputs in the notebook
from sklearn.datasets import dump_svmlight_file # For outputting data to libsvm format for xgboost
from time import gmtime, strftime # For labeling IN models, endpoints, etc.
import sys                      # For writing outputs to notebook
import math                     # For ceiling function
import json                     # For parsing hosting outputs
```

## Data

Let's start by downloading a dataset from UCI's ML Repository.

```
In [ ]: wget https://archive.ics.uci.edu/ml/machine-learning-databases/00222/bank-additional.zip
unzip bank-additional.zip
```



### Preparation

To start, let's define:

- IAM role for training/hosting to access resources
- The name of the S3 bucket we want to use
- An S3 prefix

```
In [12]: import os
import boto3

role = 'arn:aws:iam::9-666666666666:role/sagemakerrole'
bucket = 'sagemaker-pdx1'
prefix = 'xgboost'
```

Now let's bring in the Python libraries that we'll use throughout the analysis

```
In [2]: import numpy as np          # For matrix operations and numerical processing
import pandas as pd              # For munging tabular data
import matplotlib.pyplot as plt  # For charts and visualizations
from IPython.display import Image # For displaying images in the notebook
from IPython.display import display # For displaying outputs in the notebook
from sklearn.datasets import dump_svmlight_file # For outputting data to libsvm format for xgboost
from time import gmtime, strftime # For labeling IN models, endpoints, etc.
import sys                       # For writing outputs to notebook
import math                      # For ceiling function
import json                      # For parsing hosting outputs
```

## Data

Let's start by downloading a dataset from UCI's ML Repository.

```
In [ ]: wget https://archive.ics.uci.edu/ml/machine-learning-databases/00222/bank-additional.zip
        unzip bank-additional.zip
```

mynotebookinstance-1.notebook.us-west-2.sagemaker.aws

Amazon SageMaker

sample-notebooks/xgboost\_direct\_marketing/

xgboost\_direct\_marketing\_sagemaker

jupyter

xgboost\_direct\_marketing\_sagemaker

Last Checkpoint: 5 minutes ago (unsaved changes)

File

Edit

View

Insert

Cell

Kernel

Widgets

Help

Trusted

Python 3

import math

import json

# For ceiling function

# For parsing hosting outputs

Data

Let's start by downloading a dataset from UCI's ML Repository.

In [ ]:

```
!wget https://archive.ics.uci.edu/ml/machine-learning-databases/00222/bank-additional.zip
!unzip bank-additional.zip

--2017-11-19 00:42:39-- https://archive.ics.uci.edu/ml/machine-learning-databases/00222/bank-additional.zip
Resolving archive.ics.uci.edu (archive.ics.uci.edu)... 128.195.10.249
Connecting to archive.ics.uci.edu (archive.ics.uci.edu)[128.195.10.249]:443... connected.
HTTP request sent, awaiting response... 200 OK
Length: 444572 (434K) [application/zip]
Saving to: 'bank-additional.zip'

bank-additional.zip 0%[          ] 0 --.-KB/s
```

Now lets read this into a Pandas data frame and take a look.

In [ ]:

```
data = pd.read_csv('./bank-additional/bank-additional-full.csv', sep=';')
pd.set_option('display.max_columns', 500) # Make sure we can see all of the columns
pd.set_option('display.max_rows', 20) # Keep the output on one page
data
```

Let's talk about the data. At a high level, we can see:

- We have a little over 40K customer records, and 20 features for each customer
- The features are mixed; some numeric, some categorical
- The data appears to be sorted, at least by time and contact, maybe more

Specifics on each of the features:

Demographics:

- age: Customer's age (numeric)

mynotebookinstance-1.notebook.us-west-2.sagemaker.aws

Amazon SageMaker

sample-notebooks/xgboost\_direct\_marketing/

xgboost\_direct\_marketing\_sagemaker

+

+

3<

↺

↻

↕

↕

Run

⏏

C

Markdown

⌨

Edit Presentation

Show Presentation

Out[15]:

	age	job	marital	education	default	housing	loan	contact	month	day_of_week	duration	campaign	pdays	previous	outcome
0	56	housemaid	married	basic.4y	no	no	no	telephone	may	mon	261	1	999	0	nonexistent
1	57	services	married	high.school	unknown	no	no	telephone	may	mon	149	1	999	0	nonexistent
2	37	services	married	high.school	no	yes	no	telephone	may	mon	226	1	999	0	nonexistent
3	40	admin.	married	basic.6y	no	no	no	telephone	may	mon	151	1	999	0	nonexistent
4	56	services	married	high.school	no	no	yes	telephone	may	mon	307	1	999	0	nonexistent
5	45	services	married	basic.9y	unknown	no	no	telephone	may	mon	198	1	999	0	nonexistent
6	59	admin.	married	professional.course	no	no	no	telephone	may	mon	139	1	999	0	nonexistent
7	41	blue-collar	married	unknown	unknown	no	no	telephone	may	mon	217	1	999	0	nonexistent
8	24	technician	single	professional.course	no	yes	no	telephone	may	mon	380	1	999	0	nonexistent
9	25	services	single	high.school	no	yes	no	telephone	may	mon	50	1	999	0	nonexistent
...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...
41178	62	retired	married	university.degree	no	no	no	cellular	nov	thu	483	2	6	3	success
41179	64	retired	divorced	professional.course	no	yes	no	cellular	nov	fri	151	3	999	0	nonexistent
41180	36	admin.	married	university.degree	no	no	no	cellular	nov	fri	254	2	999	0	nonexistent
41181	37	admin.	married	university.degree	no	yes	no	cellular	nov	fri	281	1	999	0	nonexistent
41182	29	unemployed	single	basic.4y	no	yes	no	cellular	nov	fri	112	1	9	1	success
41183	73	retired	married	professional.course	no	yes	no	cellular	nov	fri	334	1	999	0	nonexistent
41184	46	blue-collar	married	professional.course	no	no	no	cellular	nov	fri	383	1	999	0	nonexistent
41185	56	retired	married	university.degree	no	yes	no	cellular	nov	fri	189	2	999	0	nonexistent
41186	44	technician	married	professional.course	no	no	no	cellular	nov	fri	442	1	999	0	nonexistent
41187	74	retired	married	professional.course	no	yes	no	cellular	nov	fri	239	3	999	1	failure

41188 rows x 21 columns

Let's talk about the data. At a high level, we can see:

- We have a little over 40K customer records, and 20 features for each customer
- The features are mixed: some numeric, some categorical

Present

Slides

Themes

Help

Out[15]:

loan	contact	month	day_of_week	duration	campaign	pdays	previous	poutcome	emp.var.rate	cons.price.idx	cons.conf.idx	euribor3m	nr.employed	
no	telephone	may	mon	261	1	999	0	nonexistent	1.1	93.994	-36.4	4.857	5191.0	no
no	telephone	may	mon	149	1	999	0	nonexistent	1.1	93.994	-36.4	4.857	5191.0	no
no	telephone	may	mon	226	1	999	0	nonexistent	1.1	93.994	-36.4	4.857	5191.0	no
no	telephone	may	mon	151	1	999	0	nonexistent	1.1	93.994	-36.4	4.857	5191.0	no
yes	telephone	may	mon	307	1	999	0	nonexistent	1.1	93.994	-36.4	4.857	5191.0	no
no	telephone	may	mon	198	1	999	0	nonexistent	1.1	93.994	-36.4	4.857	5191.0	no
no	telephone	may	mon	139	1	999	0	nonexistent	1.1	93.994	-36.4	4.857	5191.0	no
no	telephone	may	mon	217	1	999	0	nonexistent	1.1	93.994	-36.4	4.857	5191.0	no
no	telephone	may	mon	380	1	999	0	nonexistent	1.1	93.994	-36.4	4.857	5191.0	no
no	telephone	may	mon	50	1	999	0	nonexistent	1.1	93.994	-36.4	4.857	5191.0	no
...	...	...	...	...	...	...	...	...	...	...	...	...	...	...
no	cellular	nov	thu	483	2	6	3	success	-1.1	94.767	-50.8	1.031	4963.6	yes
no	cellular	nov	fri	151	3	999	0	nonexistent	-1.1	94.767	-50.8	1.028	4963.6	no
no	cellular	nov	fri	254	2	999	0	nonexistent	-1.1	94.767	-50.8	1.028	4963.6	no
no	cellular	nov	fri	281	1	999	0	nonexistent	-1.1	94.767	-50.8	1.028	4963.6	yes
no	cellular	nov	fri	112	1	9	1	success	-1.1	94.767	-50.8	1.028	4963.6	no
no	cellular	nov	fri	334	1	999	0	nonexistent	-1.1	94.767	-50.8	1.028	4963.6	yes
no	cellular	nov	fri	383	1	999	0	nonexistent	-1.1	94.767	-50.8	1.028	4963.6	no
no	cellular	nov	fri	189	2	999	0	nonexistent	-1.1	94.767	-50.8	1.028	4963.6	no
no	cellular	nov	fri	442	1	999	0	nonexistent	-1.1	94.767	-50.8	1.028	4963.6	yes
no	cellular	nov	fri	239	3	999	1	failure	-1.1	94.767	-50.8	1.028	4963.6	no

Let's talk about the data. At a high level, we can see:

- We have a little over 40K customer records, and 20 features for each customer

mynotebookinstance-1.notebook.us-west-2.sagemaker.aws

Amazon SageMaker

sample-notebooks/xgboost\_direct\_marketing/

xgboost\_direct\_marketing\_sagemaker

Run Edit Presentation Show Presentation

Out[15]:

loan	contact	month	day_of_week	duration	campaign	pdays	previous	poutcome	emp.var.rate	cons.price.idx	cons.conf.idx	euribor3m	nr.employed	y
no	telephone	may	mon	261	1	999	0	nonexistent	1.1	93.994	-36.4	4.857	5191.0	no
no	telephone	may	mon	149	1	999	0	nonexistent	1.1	93.994	-36.4	4.857	5191.0	no
no	telephone	may	mon	226	1	999	0	nonexistent	1.1	93.994	-36.4	4.857	5191.0	no
no	telephone	may	mon	151	1	999	0	nonexistent	1.1	93.994	-36.4	4.857	5191.0	no
yes	telephone	may	mon	307	1	999	0	nonexistent	1.1	93.994	-36.4	4.857	5191.0	no
no	telephone	may	mon	198	1	999	0	nonexistent	1.1	93.994	-36.4	4.857	5191.0	no
no	telephone	may	mon	139	1	999	0	nonexistent	1.1	93.994	-36.4	4.857	5191.0	no
no	telephone	may	mon	217	1	999	0	nonexistent	1.1	93.994	-36.4	4.857	5191.0	no
no	telephone	may	mon	380	1	999	0	nonexistent	1.1	93.994	-36.4	4.857	5191.0	no
no	telephone	may	mon	50	1	999	0	nonexistent	1.1	93.994	-36.4	4.857	5191.0	no
...	...	...	...	...	...	...	...	...	...	...	...	...	...	...
no	cellular	nov	thu	483	2	6	3	success	-1.1	94.767	-50.8	1.031	4963.6	yes
no	cellular	nov	fri	151	3	999	0	nonexistent	-1.1	94.767	-50.8	1.028	4963.6	no
no	cellular	nov	fri	254	2	999	0	nonexistent	-1.1	94.767	-50.8	1.028	4963.6	no
no	cellular	nov	fri	281	1	999	0	nonexistent	-1.1	94.767	-50.8	1.028	4963.6	yes
no	cellular	nov	fri	112	1	9	1	success	-1.1	94.767	-50.8	1.028	4963.6	no
no	cellular	nov	fri	334	1	999	0	nonexistent	-1.1	94.767	-50.8	1.028	4963.6	yes
no	cellular	nov	fri	383	1	999	0	nonexistent	-1.1	94.767	-50.8	1.028	4963.6	no
no	cellular	nov	fri	189	2	999	0	nonexistent	-1.1	94.767	-50.8	1.028	4963.6	no
no	cellular	nov	fri	442	1	999	0	nonexistent	-1.1	94.767	-50.8	1.028	4963.6	yes
no	cellular	nov	fri	239	3	999	1	failure	-1.1	94.767	-50.8	1.028	4963.6	no

Let's talk about the data. At a high level, we can see:

- We have a little over 40K customer records, and 20 features for each customer

Present

Slides

Themes

Help

mynotebookinstance-1.notebook.us-west-2.sagemaker.aws

Amazon SageMaker

sample-notebooks/xgboost\_direct\_marketing/

xgboost\_direct\_marketing\_sagemaker

Run Edit Presentation Show Presentation

Out[15]:

loan	contact	month	day_of_week	duration	campaign	pdays	previous	poutcome	emp.var.rate	cons.price.idx	cons.conf.idx	euribor3m	nr.employed	y
no	telephone	may	mon	261	1	999	0	nonexistent	1.1	93.994	-36.4	4.857	5191.0	no
no	telephone	may	mon	149	1	999	0	nonexistent	1.1	93.994	-36.4	4.857	5191.0	no
no	telephone	may	mon	226	1	999	0	nonexistent	1.1	93.994	-36.4	4.857	5191.0	no
no	telephone	may	mon	151	1	999	0	nonexistent	1.1	93.994	-36.4	4.857	5191.0	no
yes	telephone	may	mon	307	1	999	0	nonexistent	1.1	93.994	-36.4	4.857	5191.0	no
no	telephone	may	mon	198	1	999	0	nonexistent	1.1	93.994	-36.4	4.857	5191.0	no
no	telephone	may	mon	139	1	999	0	nonexistent	1.1	93.994	-36.4	4.857	5191.0	no
no	telephone	may	mon	217	1	999	0	nonexistent	1.1	93.994	-36.4	4.857	5191.0	no
no	telephone	may	mon	380	1	999	0	nonexistent	1.1	93.994	-36.4	4.857	5191.0	no
no	telephone	may	mon	50	1	999	0	nonexistent	1.1	93.994	-36.4	4.857	5191.0	no
...	...	...	...	...	...	...	...	...	...	...	...	...	...	...
no	cellular	nov	thu	483	2	6	3	success	-1.1	94.767	-50.8	1.031	4963.6	yes
no	cellular	nov	fri	151	3	999	0	nonexistent	-1.1	94.767	-50.8	1.028	4963.6	no
no	cellular	nov	fri	254	2	999	0	nonexistent	-1.1	94.767	-50.8	1.028	4963.6	no
no	cellular	nov	fri	281	1	999	0	nonexistent	-1.1	94.767	-50.8	1.028	4963.6	yes
no	cellular	nov	fri	112	1	9	1	success	-1.1	94.767	-50.8	1.028	4963.6	no
no	cellular	nov	fri	334	1	999	0	nonexistent	-1.1	94.767	-50.8	1.028	4963.6	yes
no	cellular	nov	fri	383	1	999	0	nonexistent	-1.1	94.767	-50.8	1.028	4963.6	no
no	cellular	nov	fri	189	2	999	0	nonexistent	-1.1	94.767	-50.8	1.028	4963.6	no
no	cellular	nov	fri	442	1	999	0	nonexistent	-1.1	94.767	-50.8	1.028	4963.6	yes
no	cellular	nov	fri	239	3	999	1	failure	-1.1	94.767	-50.8	1.028	4963.6	no

Let's talk about the data. At a high level, we can see:

- We have a little over 40K customer records, and 20 features for each customer

Present

Slides

Themes

Help



mynotebookinstance-1.notebook.us-west-2.sagemaker.aws

Amazon SageMaker

sample-notebooks/xgboost\_direct\_marketing/

xgboost\_direct\_marketing\_sagemaker

Run

Markdown

Edit Presentation

Show Presentation

Let's talk about the data. At a high level, we can see:

- We have a little over 40K customer records, and 20 features for each customer
- The features are mixed; some numeric, some categorical
- The data appears to be sorted, at least by `time` and `contact`, maybe more

**Specifics on each of the features:**

**Demographics:**

- `age` : Customer's age (numeric)
- `job` : Type of job (categorical: 'admin', 'services', ...)
- `marital` : Marital status (categorical: 'married', 'single', ...)
- `education` : Level of education (categorical: 'basic.4y', 'high.school', ...)

**Past customer events:**

- `default` : Has credit in default? (categorical: 'no', 'unknown', ...)
- `housing` : Has housing loan? (categorical: 'no', 'yes', ...)
- `loan` : Has personal loan? (categorical: 'no', 'yes', ...)

**Past direct marketing contacts:**

- `contact` : Contact communication type (categorical: 'cellular', 'telephone', ...)
- `month` : Last contact month of year (categorical: 'may', 'nov', ...)
- `day_of_week` : Last contact day of the week (categorical: 'mon', 'fri', ...)
- `duration` : Last contact duration, in seconds (numeric). Important note: If duration = 0 then `y` = 'no'.

**Campaign information:**

- `campaign` : Number of contacts performed during this campaign and for this client (numeric, includes last contact)
- `pdays` : Number of days that passed by after the client was last contacted from a previous campaign (numeric)
- `previous` : Number of contacts performed before this campaign and for this client (numeric)
- `postcome` : Outcome of the previous marketing campaign (categorical: 'nonexistent', 'success', ...)

**External environment factors:**

Present

Slides

Themes

Help

mynotebookinstance-1.notebook.us-west-2.sagemaker.aws

Amazon SageMaker

sample-notebooks/xgboost\_direct\_marketing/

xgboost\_direct\_marketing\_sagemaker

Code

Edit Presentation

Show Presentation

Target variable:

- y: Has the client subscribed a term deposit? (binary: 'yes','no')

### Exploration

Let's start exploring the data. First, let's understand how the features are distributed.

```
In [ ]: # Frequency tables for each categorical feature
for column in data.select_dtypes(include=['object']).columns:
    display(pd.crosstab(index=data[column], columns='% observations', normalize='columns'))

# Histograms for each numeric features
display(data.describe())
%matplotlib inline
hist = data.hist(bins=30, sharey=True, figsize=(10, 10))
```

Notice that:

- Almost 90% of the values for our target variable y are "no", so most customers did not subscribe to a term deposit.
- Many of the predictive features take on values of "unknown". Some are more common than others. We should think carefully as to what causes a value of "unknown" (are these customers non-representative in some way?) and how we that should be handled.
  - Even if "unknown" is included as it's own distinct category, what does it mean given that, in reality, those observations likely fall within one of the other categories of that feature?
- Many of the predictive features have categories with very few observations in them. If we find a small category to be highly predictive of our target outcome, do we have enough evidence to make a generalization about that?
- Contact timing is particularly skewed. Almost a third in May and less than 1% in December. What does this mean for predicting our target variable next December?
- There are no missing values in our numeric features. Or missing values have already been imputed.
  - pdays takes a value near 1000 for almost all customers. Likely a placeholder value signifying no previous contact.
- Several numeric features have a very long tail. Do we need to handle these few observations with extremely large values differently?
- Several numeric features (particularly the macroeconomic ones) occur in distinct buckets. Should these be treated as categorical?

Next, let's look at how our features relate to the target that we are attempting to predict.

```
In [ ]: display(data.corr())
```

Present

Slides

Themes

Help



Run Edit Presentation Show Presentation

admin	0.253035
blue-collar	0.224677
entrepreneur	0.035350
housemaid	0.025736
management	0.070992
retired	0.041760
self-employed	0.034500
services	0.096363
student	0.021244
technician	0.162713

Notice that:

- Almost 90% of the values for our target variable `y` are "no", so most customers did not subscribe to a term deposit.
- Many of the predictive features take on values of "unknown". Some are more common than others. We should think carefully as to what causes a value of "unknown" (are these customers non-representative in some way?) and how we that should be handled.
  - Even if "unknown" is included as it's own distinct category, what does it mean given that, in reality, those observations likely fall within one of the other categories of that feature?
- Many of the predictive features have categories with very few observations in them. If we find a small category to be highly predictive of our target outcome, do we have enough evidence to make a generalization about that?
- Contact timing is particularly skewed. Almost a third in May and less than 1% in December. What does this mean for predicting our target variable next December?
- There are no missing values in our numeric features. Or missing values have already been imputed.
  - `pdays` takes a value near 1000 for almost all customers. Likely a placeholder value signifying no previous contact.
- Several numeric features have a very long tail. Do we need to handle these few observations with extremely large values differently?
- Several numeric features (particularly the macroeconomic ones) occur in distinct buckets. Should these be treated as categorical?

Next, let's look at how our features relate to the target that we are attempting to predict.

```
In [ ]: display(data.corr())
```

Notice that:

- Features vary widely in their relationship with one another. Some with highly negative correlation, others with highly positive correlation.

Run Edit Presentation Show Presentation

- Many of the predictive features take on values of "unknown". Some are more common than others. We should think carefully as to what causes a value of "unknown" (are these customers non-representative in some way?) and how we that should be handled.
  - Even if "unknown" is included as it's own distinct category, what does it mean given that, in reality, those observations likely fall within one of the other categories of that feature?
- Many of the predictive features have categories with very few observations in them. If we find a small category to be highly predictive of our target outcome, do we have enough evidence to make a generalization about that?
- Contact timing is particularly skewed. Almost a third in May and less than 1% in December. What does this mean for predicting our target variable next December?
- There are no missing values in our numeric features. Or missing values have already been imputed.
  - pdays takes a value near 1000 for almost all customers. Likely a placeholder value signifying no previous contact.
- Several numeric features have a very long tail. Do we need to handle these few observations with extremely large values differently?
- Several numeric features (particularly the macroeconomic ones) occur in distinct buckets. Should these be treated as categorical?

Next, let's look at how our features relate to the target that we are attempting to predict.

```
In [17]: display(data.corr())
```

	age	duration	campaign	pdays	previous	emp.var.rate	cons.price.idx	cons.conf.idx	euribor3m	nr.employed
age	1.000000	-0.000866	0.004594	-0.034369	0.024365	-0.000371	0.000857	0.129372	0.010767	-0.017725
duration	-0.000866	1.000000	-0.071699	-0.047577	0.020640	-0.027968	0.005312	-0.008173	-0.032897	-0.044703
campaign	0.004594	-0.071699	1.000000	0.052584	-0.079141	0.150754	0.127836	-0.013733	0.135133	0.144095
pdays	-0.034369	-0.047577	0.052584	1.000000	-0.587514	0.271004	0.078889	-0.091342	0.296889	0.372605
previous	0.024365	0.020640	-0.079141	-0.587514	1.000000	-0.420489	-0.203130	-0.050936	-0.454494	-0.501333
emp.var.rate	-0.000371	-0.027968	0.150754	0.271004	-0.420489	1.000000	0.775334	0.196041	0.972245	0.906970
cons.price.idx	0.000857	0.005312	0.127836	0.078889	-0.203130	0.775334	1.000000	0.058966	0.688230	0.522034
cons.conf.idx	0.129372	-0.008173	-0.013733	-0.091342	-0.050936	0.196041	0.058966	1.000000	0.277686	0.100513
euribor3m	0.010767	-0.032897	0.135133	0.296889	-0.454494	0.972245	0.688230	0.277686	1.000000	0.945154
nr.employed	-0.017725	-0.044703	0.144095	0.372605	-0.501333	0.906970	0.522034	0.100513	0.945154	1.000000

Notice that:

- Features vary widely in their relationship with one another. Some with highly negative correlation, others with highly positive correlation.



mynotebookinstance-1.notebook.us-west-2.sagemaker.aws

Amazon SageMaker

sample-notebooks/xgboost\_direct\_marketing/

xgboost\_direct\_marketing\_sagemaker

Code

Edit Presentation

Show Presentation

Following this logic, let's remove the economic features and `duration` from our data as they would need to be forecasted with high precision to use as inputs in future predictions.

Even if we were to use values of the economic indicators from the previous quarter, this value is likely not as relevant for prospects contacted early in the next quarter as those contacted later on.

```
In [19]: model_data = model_data.drop(['duration', 'emp.var.rate', 'cons.price.idx', 'cons.conf.idx', 'euribor3m', 'ar.employed'])
```

When building a model whose primary goal is to predict a target value on new data, it is important to understand overfitting. Supervised learning models are designed to minimize error between their predictions of the target value and actuals, in the data they are given. This last part is key, as frequently in their quest for greater accuracy, machine learning models bias themselves toward picking up on minor idiosyncrasies within the data they are shown. These idiosyncrasies then don't repeat themselves in subsequent data, meaning those predictions can actually be made less accurate, at the expense of more accurate predictions in the training phase.

The most common way of preventing this is to build models with the concept that a model shouldn't only be judged on its fit to the data it was trained on, but also on "new" data. There are several different ways of operationalizing this, holdout validation, cross-validation, leave-one-out validation, etc. For our purposes, we'll simply randomly split the data into 3 uneven groups. The model will be trained on 70% of data, it will then be evaluated on 20% of data to give us an estimate of the accuracy we hope to have on "new" data, and 10% will be held back as a final testing dataset which will be used later on.

```
In [9]: train_data, validation_data, test_data = np.split(model_data.sample(frac=1, random_state=1729), [int(0.7 * len(model_data)), int(0.8 * len(model_data))])
```

Amazon SageMaker's XGBoost container expects data in the libSVM data format. This expects features and the target variable to be provided as separate arguments. Let's split these apart. Notice that although repetitive it's easiest to do this after the train/validation/test split rather than before. This avoids any misalignment issues due to random reordering.

```
In [10]: dump_svmlight_file(X=train_data.drop(['y_no', 'y_yes'], axis=1), y=train_data['y_yes'], f='train.libsvm')
dump_svmlight_file(X=validation_data.drop(['y_no', 'y_yes'], axis=1), y=validation_data['y_yes'], f='validation.libsvm')
dump_svmlight_file(X=test_data.drop(['y_no', 'y_yes'], axis=1), y=test_data['y_yes'], f='test.libsvm')
```

Now we'll copy the file to S3 for Amazon SageMaker's managed training to pickup.

```
In [ ]: boto3.Session().resource('s3').Bucket(bucket).Object(os.path.join(prefix, 'train/train.libsvm')).upload_file('train.libsvm')
boto3.Session().resource('s3').Bucket(bucket).Object(os.path.join(prefix, 'train/validation.libsvm')).upload_file('validation.libsvm')
boto3.Session().resource('s3').Bucket(bucket).Object(os.path.join(prefix, 'train/test.libsvm')).upload_file('test.libsvm')
```

Present

Slides

Themes

Help



mynotebookinstance-1.notebook.us-west-2.sagemaker.aws

Amazon SageMaker

sample-notebooks/xgboost\_direct\_marketing/

xgboost\_direct\_marketing\_sagemaker

Run

C

Markdown

Edit Presentation

Show Presentation

Following this logic, let's remove the economic features and `duration` from our data as they would need to be forecasted with high precision to use as inputs in future predictions.

Even if we were to use values of the economic indicators from the previous quarter, this value is likely not as relevant for prospects contacted early in the next quarter as those contacted later on.

```
In [19]: model_data = model_data.drop(['duration', 'emp.var.rate', 'cons.price.idx', 'cons.conf.idx', 'euribor3m', 'ar.employed'])
```

When building a model whose primary goal is to predict a target value on new data, it is important to understand overfitting. Supervised learning models are designed to minimize error between their predictions of the target value and actuals, in the data they are given. This last part is key, as frequently in their quest for greater accuracy, machine learning models bias themselves toward picking up on minor idiosyncrasies within the data they are shown. These idiosyncrasies then don't repeat themselves in subsequent data, meaning those predictions can actually be made less accurate, at the expense of more accurate predictions in the training phase.

The most common way of preventing this is to build models with the concept that a model shouldn't only be judged on its fit to the data it was trained on, but also on "new" data. There are several different ways of operationalizing this, holdout validation, cross-validation, leave-one-out validation, etc. For our purposes, we'll simply randomly split the data into 3 uneven groups. The model will be trained on 70% of data, it will then be evaluated on 20% of data to give us an estimate of the accuracy we hope to have on "new" data, and 10% will be held back as a final testing dataset which will be used later on.

```
In [20]: train_data, validation_data, test_data = np.split(model_data.sample(frac=1, random_state=1729), [int(0.7 * len(model_data))])
```

Amazon SageMaker's XGBoost container expects data in the libSVM data format. This expects features and the target variable to be provided as separate arguments. Let's split these apart. Notice that although repetitive it's easiest to do this after the train/validation/test split rather than before. This avoids any misalignment issues due to random reordering.

```
In [10]: dump_svmlight_file(X=train_data.drop(['y_no', 'y_yes'], axis=1), y=train_data['y_yes'], f='train.libsvm')
dump_svmlight_file(X=validation_data.drop(['y_no', 'y_yes'], axis=1), y=validation_data['y_yes'], f='validation.libsvm')
dump_svmlight_file(X=test_data.drop(['y_no', 'y_yes'], axis=1), y=test_data['y_yes'], f='test.libsvm')
```

Now we'll copy the file to S3 for Amazon SageMaker's managed training to pickup.

```
In [ ]: boto3.Session().resource('s3').Bucket(bucket).Object(os.path.join(prefix, 'train/train.libsvm')).upload_file('train.libsvm')
boto3.Session().resource('s3').Bucket(bucket).Object(os.path.join(prefix, 'train/validation.libsvm')).upload_file('validation.libsvm')
boto3.Session().resource('s3').Bucket(bucket).Object(os.path.join(prefix, 'train/test.libsvm')).upload_file('test.libsvm')
```

Present

Slides

Themes

Help

mynotebookinstance-1.notebook.us-west-2.sagemaker.aws

Amazon SageMaker

sample-notebooks/xgboost\_direct\_marketing/

xgboost\_direct\_marketing\_sagemaker

Code

Edit Presentation

Show Presentation

Following this logic, let's remove the economic features and `duration` from our data as they would need to be forecasted with high precision to use as inputs in future predictions.

Even if we were to use values of the economic indicators from the previous quarter, this value is likely not as relevant for prospects contacted early in the next quarter as those contacted later on.

```
In [19]: model_data = model_data.drop(['duration', 'emp.var.rate', 'cons.price.idx', 'cons.conf.idx', 'euribor3m', 'ar.employed'])
```

When building a model whose primary goal is to predict a target value on new data, it is important to understand overfitting. Supervised learning models are designed to minimize error between their predictions of the target value and actuals, in the data they are given. This last part is key, as frequently in their quest for greater accuracy, machine learning models bias themselves toward picking up on minor idiosyncrasies within the data they are shown. These idiosyncrasies then don't repeat themselves in subsequent data, meaning those predictions can actually be made less accurate, at the expense of more accurate predictions in the training phase.

The most common way of preventing this is to build models with the concept that a model shouldn't only be judged on its fit to the data it was trained on, but also on "new" data. There are several different ways of operationalizing this, holdout validation, cross-validation, leave-one-out validation, etc. For our purposes, we'll simply randomly split the data into 3 uneven groups. The model will be trained on 70% of data, it will then be evaluated on 20% of data to give us an estimate of the accuracy we hope to have on "new" data, and 10% will be held back as a final testing dataset which will be used later on.

```
In [20]: train_data, validation_data, test_data = np.split(model_data.sample(frac=1, random_state=1729), [int(0.7 * len(model_data))])
```

Amazon SageMaker's XGBoost container expects data in the libSVM data format. This expects features and the target variable to be provided as separate arguments. Let's split these apart. Notice that although repetitive it's easiest to do this after the train/validation/test split rather than before. This avoids any misalignment issues due to random reordering.

```
In [10]: dump_svmlight_file(X=train_data.drop(['y_no', 'y_yes'], axis=1), y=train_data['y_yes'], f='train.libsvm')
dump_svmlight_file(X=validation_data.drop(['y_no', 'y_yes'], axis=1), y=validation_data['y_yes'], f='validation.libsvm')
dump_svmlight_file(X=test_data.drop(['y_no', 'y_yes'], axis=1), y=test_data['y_yes'], f='test.libsvm')
```

Now we'll copy the file to S3 for Amazon SageMaker's managed training to pickup.

```
In [ ]: boto3.Session().resource('s3').Bucket(bucket).Object(os.path.join(prefix, 'train/train.libsvm')).upload_file('train.libsvm')
boto3.Session().resource('s3').Bucket(bucket).Object(os.path.join(prefix, 'train/validation.libsvm')).upload_file('validation.libsvm')
boto3.Session().resource('s3').Bucket(bucket).Object(os.path.join(prefix, 'train/test.libsvm')).upload_file('test.libsvm')
```

Present

Slides

Themes

Help

mynotebookinstance-1.notebook.us-west-2.sagemaker.aws

Amazon SageMaker

sample-notebooks/xgboost\_direct\_marketing/

xgboost\_direct\_marketing\_sagemaker

Run

Markdown

Edit Presentation

Show Presentation

Following this logic, let's remove the economic features and `duration` from our data as they would need to be forecasted with high precision to use as inputs in future predictions.

Even if we were to use values of the economic indicators from the previous quarter, this value is likely not as relevant for prospects contacted early in the next quarter as those contacted later on.

```
In [19]: model_data = model_data.drop(['duration', 'emp.var.rate', 'cons.price.idx', 'cons.conf.idx', 'euribor3m', 'ar.employed'])
```

When building a model whose primary goal is to predict a target value on new data, it is important to understand overfitting. Supervised learning models are designed to minimize error between their predictions of the target value and actuals, in the data they are given. This last part is key, as frequently in their quest for greater accuracy, machine learning models bias themselves toward picking up on minor idiosyncrasies within the data they are shown. These idiosyncrasies then don't repeat themselves in subsequent data, meaning those predictions can actually be made less accurate, at the expense of more accurate predictions in the training phase.

The most common way of preventing this is to build models with the concept that a model shouldn't only be judged on its fit to the data it was trained on, but also on "new" data. There are several different ways of operationalizing this, holdout validation, cross-validation, leave-one-out validation, etc. For our purposes, we'll simply randomly split the data into 3 uneven groups. The model will be trained on 70% of data, it will then be evaluated on 20% of data to give us an estimate of the accuracy we hope to have on "new" data, and 10% will be held back as a final testing dataset which will be used later on.

```
In [20]: train_data, validation_data, test_data = np.split(model_data.sample(frac=1, random_state=1729), [int(0.7 * len(model_data))])
```

Amazon SageMaker's XGBoost container expects data in the libSVM data format. This expects features and the target variable to be provided as separate arguments. Let's split these apart. Notice that although repetitive it's easiest to do this after the train/validation/test split rather than before. This avoids any misalignment issues due to random reordering.

```
In [*]: dump_svmlight_file(X=train_data.drop(['y_no', 'y_yes'], axis=1), y=train_data['y_yes'], f='train.libsvm')
dump_svmlight_file(X=validation_data.drop(['y_no', 'y_yes'], axis=1), y=validation_data['y_yes'], f='validation.libsvm')
dump_svmlight_file(X=test_data.drop(['y_no', 'y_yes'], axis=1), y=test_data['y_yes'], f='test.libsvm')
```

Now we'll copy the file to S3 for Amazon SageMaker's managed training to pickup.

```
In [ ]: boto3.Session().resource('s3').Bucket(bucket).Object(os.path.join(prefix, 'train/train.libsvm')).upload_file('train.libsvm')
boto3.Session().resource('s3').Bucket(bucket).Object(os.path.join(prefix, 'train/validation.libsvm')).upload_file('validation.libsvm')
boto3.Session().resource('s3').Bucket(bucket).Object(os.path.join(prefix, 'train/test.libsvm')).upload_file('test.libsvm')
```

Present

Slides

Themes

Help

mynotebookinstance-1.notebook.us-west-2.sagemaker.aws

Amazon SageMaker

sample-notebooks/xgboost\_direct\_marketing/

xgboost\_direct\_marketing\_sagemaker

Run Edit Presentation Show Presentation

The most common way of preventing this is to build models with the concept that a model shouldn't only be judged on its fit to the data it was trained on, but also on "new" data. There are several different ways of operationalizing this, holdout validation, cross-validation, leave-one-out validation, etc. For our purposes, we'll simply randomly split the data into 3 uneven groups. The model will be trained on 70% of data, it will then be evaluated on 20% of data to give us an estimate of the accuracy we hope to have on "new" data, and 10% will be held back as a final testing dataset which will be used later on.

```
In [20]: train_data, validation_data, test_data = np.split(model_data.sample(frac=1, random_state=1729), [int(0.7 * len(model_data))])
```

Amazon SageMaker's XGBoost container expects data in the libSVM data format. This expects features and the target variable to be provided as separate arguments. Let's split these apart. Notice that although repetitive it's easiest to do this after the train/validation/test split rather than before. This avoids any misalignment issues due to random reordering.

```
In [21]: dump_svmlight_file(X=train_data.drop(['y_no', 'y_yes'], axis=1), y=train_data['y_yes'], f='train.libsvm')
dump_svmlight_file(X=validation_data.drop(['y_no', 'y_yes'], axis=1), y=validation_data['y_yes'], f='validation.libsvm')
dump_svmlight_file(X=test_data.drop(['y_no', 'y_yes'], axis=1), y=test_data['y_yes'], f='test.libsvm')
```

Now we'll copy the file to S3 for Amazon SageMaker's managed training to pickup.

```
In [ ]: boto3.Session().resource('s3').Bucket(bucket).Object(os.path.join(prefix, 'train/train.libsvm')).upload_file('train.libsvm')
boto3.Session().resource('s3').Bucket(bucket).Object(os.path.join(prefix, 'train/validation.libsvm')).upload_file('validation.libsvm')
boto3.Session().resource('s3').Bucket(bucket).Object(os.path.join(prefix, 'train/test.libsvm')).upload_file('test.libsvm')
```

## Training

Now we know most of our features have skewed distributions, some are highly correlated with one another, and some appear to have non-linear relationships with our target variable. Also, for targeting future prospects, good predictive accuracy is preferred to being able to explain why that prospect was targeted. Taken together, these aspects make gradient boosted trees a good candidate algorithm.

There are several intricacies to understanding the algorithm, but at a high level, gradient boosted trees works by combining predictions from many simple models, each of which tries to address the weaknesses of the previous models. By doing this the collection of simple models can actually outperform large, complex models. Other Amazon SageMaker notebooks elaborate on gradient boosting trees further and how they differ from similar algorithms.

xgboost is an extremely popular, open-source package for gradient boosted trees. It is computationally powerful, fully featured, and has been successfully used in many machine learning competitions. Let's start with a simple xgboost model, trained using Amazon SageMaker's managed, distributed training framework.

Present

Slides

Themes

Help



mynotebookinstance-1.notebook.us-west-2.sagemaker.aws

Amazon SageMaker

sample-notebooks/xgboost\_direct\_marketing/

xgboost\_direct\_marketing\_sagemaker

Run

Markdown

Edit Presentation

Show Presentation

In [21]:  
dump\_svmlight\_file(X=train\_data.drop(['y\_no', 'y\_yes'], axis=1), y=train\_data['y\_yes'], f='train.libsvm')  
dump\_svmlight\_file(X=validation\_data.drop(['y\_no', 'y\_yes'], axis=1), y=validation\_data['y\_yes'], f='validation.libsvm')  
dump\_svmlight\_file(X=test\_data.drop(['y\_no', 'y\_yes'], axis=1), y=test\_data['y\_yes'], f='test.libsvm')

Now we'll copy the file to S3 for Amazon SageMaker's managed training to pickup.

In [22]:  
boto3.Session().resource('s3').Bucket(bucket).Object(os.path.join(prefix, 'train/train.libsvm')).upload\_file('train.libsvm')  
boto3.Session().resource('s3').Bucket(bucket).Object(os.path.join(prefix, 'train/validation.libsvm')).upload\_file('validation.libsvm')  
boto3.Session().resource('s3').Bucket(bucket).Object(os.path.join(prefix, 'train/test.libsvm')).upload\_file('test.libsvm')

## Training

Now we know most of our features have skewed distributions, some are highly correlated with one another, and some appear to have non-linear relationships with our target variable. Also, for targeting future prospects, good predictive accuracy is preferred to being able to explain why that prospect was targeted. Taken together, these aspects make gradient boosted trees a good candidate algorithm.

There are several intricacies to understanding the algorithm, but at a high level, gradient boosted trees works by combining predictions from many simple models, each of which tries to address the weaknesses of the previous models. By doing this the collection of simple models can actually outperform large, complex models. Other Amazon SageMaker notebooks elaborate on gradient boosting trees further and how they differ from similar algorithms.

`xgboost` is an extremely popular, open-source package for gradient boosted trees. It is computationally powerful, fully featured, and has been successfully used in many machine learning competitions. Let's start with a simple `xgboost` model, trained using Amazon SageMaker's managed, distributed training framework.

First we'll need to specify training parameters. This includes:

1. The role to use
2. Our training job name
3. The `xgboost` algorithm container
4. Training instance type and count
5. S3 location for training data
6. S3 location for output data
7. Algorithm hyperparameters
8. Stopping conditions

Present

Slides

Themes

Help

mynotebookinstance-1.notebook.us-west-2.sagemaker.aws

Amazon SageMaker

sample-notebooks/xgboost\_direct\_marketing/

xgboost\_direct\_marketing\_sagemaker

Run

Markdown

Edit Presentation

Show Presentation

## Training

Now we know most of our features have skewed distributions, some are highly correlated with one another, and some appear to have non-linear relationships with our target variable. Also, for targeting future prospects, good predictive accuracy is preferred to being able to explain why that prospect was targeted. Taken together, these aspects make gradient boosted trees a good candidate algorithm.

There are several intricacies to understanding the algorithm, but at a high level, gradient boosted trees works by combining predictions from many simple models, each of which tries to address the weaknesses of the previous models. By doing this the collection of simple models can actually outperform large, complex models. Other Amazon SageMaker notebooks elaborate on gradient boosting trees further and how they differ from similar algorithms.

`xgboost` is an extremely popular, open-source package for gradient boosted trees. It is computationally powerful, fully featured, and has been successfully used in many machine learning competitions. Let's start with a simple `xgboost` model, trained using Amazon SageMaker's managed, distributed training framework.

First we'll need to specify training parameters. This includes:

1. The role to use
2. Our training job name
3. The `xgboost` algorithm container
4. Training instance type and count
5. S3 location for training data
6. S3 location for output data
7. Algorithm hyperparameters
8. Stopping conditions

```
In [ ]: job_name = 'xgboost-dm-' + strftime("%Y-%m-%d-%H-%M-%S", gmtime())
        print("Training job", job_name)

        create_training_params = \
        {
            "RoleArn": role,
            "TrainingJobName": job_name,
            "AlgorithmSpecification": {
                "TrainingImage": "032969728358.dkr.ecr.us-west-2.amazonaws.com/xgboost-learner:latest",
                "TrainingInputMode": "File"
```

Present

Slides

Themes

Help

mynotebookinstance-1.notebook.us-west-2.sagemaker.aws

Amazon SageMaker

sample-notebooks/xgboost\_direct\_marketing/

xgboost\_direct\_marketing\_sagemaker

Run

Markdown

Edit Presentation

Show Presentation

## Training

Now we know most of our features have skewed distributions, some are highly correlated with one another, and some appear to have non-linear relationships with our target variable. Also, for targeting future prospects, good predictive accuracy is preferred to being able to explain why that prospect was targeted. Taken together, these aspects make gradient boosted trees a good candidate algorithm.

There are several intricacies to understanding the algorithm, but at a high level, gradient boosted trees works by combining predictions from many simple models, each of which tries to address the weaknesses of the previous models. By doing this the collection of simple models can actually outperform large, complex models. Other Amazon SageMaker notebooks elaborate on gradient boosting trees further and how they differ from similar algorithms.

`xgboost` is an extremely popular, open-source package for gradient boosted trees. It is computationally powerful, fully featured, and has been successfully used in many machine learning competitions. Let's start with a simple `xgboost` model, trained using Amazon SageMaker's managed, distributed training framework.

First we'll need to specify training parameters. This includes:

1. The role to use
2. Our training job name
3. The `xgboost` algorithm container
4. Training instance type and count
5. S3 location for training data
6. S3 location for output data
7. Algorithm hyperparameters
8. Stopping conditions

```
In [ ]: job_name = 'xgboost-dm-' + strftime('%Y-%m-%d-%H-%M-%S', gmtime())
        print("Training job", job_name)

        create_training_params = \
        {
            "RoleArn": role,
            "TrainingJobName": job_name,
            "AlgorithmSpecification": {
                "TrainingImage": "032969728358.dkr.ecr.us-west-2.amazonaws.com/xgboost-learner:latest",
                "TrainingInputMode": "File"
```

Present

Slides

Themes

Help

mynotebookinstance-1.notebook.us-west-2.sagemaker.aws

Amazon SageMaker

sample-notebooks/xgboost\_direct\_marketing/

xgboost\_direct\_marketing\_sagemaker

Run

Markdown

Edit Presentation

Show Presentation

3. The xgboost algorithm container

4. Training instance type and count

5. S3 location for training data

6. S3 location for output data

7. Algorithm hyperparameters

8. Stopping conditions

```
In [ ]: job_name = 'xgboost-dm-' + strftime('%Y-%m-%d-%H-%M-%S', gmtime())
print("Training job", job_name)

create_training_params = \
{
    "RoleArn": role,
    "TrainingJobName": job_name,
    "AlgorithmSpecification": {
        "TrainingImage": "032969728358.dkr.ecr.us-west-2.amazonaws.com/xgboost-learner:latest",
        "TrainingInputMode": "File"
    },
    "ResourceConfig": {
        "InstanceCount": 2,
        "InstanceType": "ml.c4.2xlarge",
        "VolumeSizeInGB": 50
    },
    "InputDataConfig": {
        {
            "ChannelName": "train",
            "DataSource": {
                "S3DataSource": {
                    "S3DataType": "S3Prefix",
                    "S3Uri": "s3://{}/{}/train".format(bucket, prefix),
                    "S3DataDistributionType": "FullyReplicated"
                }
            },
            "ContentType": "libsvm",
            "CompressionType": "None"
        }
    },
    "OutputDataConfig": {
        "S3OutputPath": "s3://{}/{}/output".format(bucket, prefix)
    },
    "HyperParameters": {
```

Present

Slides

Themes

Help

mynotebookinstance-1.notebook.us-west-2.sagemaker.aws

Amazon SageMaker

sample-notebooks/xgboost\_direct\_marketing/

xgboost\_direct\_marketing\_sagemaker

Run

Markdown

Edit Presentation

Show Presentation

```
In [ ]: job_name = 'xgboost-dm-' + strftime("%Y-%m-%d-%H-%M-%S", gmtime())
print("Training job", job_name)

create_training_params = \
{
    "RoleArn": role,
    "TrainingJobName": job_name,
    "AlgorithmSpecification": {
        "TrainingImage": "032969728358.dkr.ecr.us-west-2.amazonaws.com/xgboost-learner:latest",
        "TrainingInputMode": "File"
    },
    "ResourceConfig": {
        "InstanceCount": 2,
        "InstanceType": "ml.c4.2xlarge",
        "VolumeSizeInGB": 50
    },
    "InputDataConfig": [
        {
            "ChannelName": "train",
            "DataSource": {
                "S3DataSource": {
                    "S3DataType": "S3Prefix",
                    "S3Uri": "s3://{}{}train".format(bucket, prefix),
                    "S3DataDistributionType": "FullyReplicated"
                }
            },
            "ContentType": "libsvm",
            "CompressionType": "None"
        }
    ],
    "OutputDataConfig": {
        "S3OutputPath": "s3://{}{}output".format(bucket, prefix)
    },
    "HyperParameters": {
        "max_depth": "5",
        "eta": "0.2",
        "gamma": "4",
        "min_child_weight": "6",
        "subsample": "0.8",
        "silent": "0",
        "objective": "binary:logistic",
        "num_class": "1",

```

Present

Slides

Themes

Help

mynotebookinstance-1.notebook.us-west-2.sagemaker.aws

Amazon SageMaker

sample-notebooks/xgboost\_direct\_marketing/

xgboost\_direct\_marketing\_sagemaker

Run

Markdown

Edit Presentation

Show Presentation

```
In [ ]: job_name = 'xgboost-dm-' + strftime("%Y-%m-%d-%H-%M-%S", gmtime())
print("Training job", job_name)

create_training_params = \
{
    "RoleArn": role,
    "TrainingJobName": job_name,
    "AlgorithmSpecification": {
        "TrainingImage": "032969728358.dkr.ecr.us-west-2.amazonaws.com/xgboost-learner:latest",
        "TrainingInputMode": "File"
    },
    "ResourceConfig": {
        "InstanceCount": 2,
        "InstanceType": "ml.c4.2xlarge",
        "VolumeSizeInGB": 50
    },
    "InputDataConfig": [
        {
            "ChannelName": "train",
            "DataSource": {
                "S3DataSource": {
                    "S3DataType": "S3Prefix",
                    "S3Uri": "s3://{}{}train".format(bucket, prefix),
                    "S3DataDistributionType": "FullyReplicated"
                }
            },
            "ContentType": "libsvm",
            "CompressionType": "None"
        }
    ],
    "OutputDataConfig": {
        "S3OutputPath": "s3://{}{}output".format(bucket, prefix)
    },
    "HyperParameters": {
        "max_depth": "5",
        "eta": "0.2",
        "gamma": "4",
        "min_child_weight": "6",
        "subsample": "0.8",
        "silent": "0",
        "objective": "binary:logistic",
        "num_class": "1",
    }
}
```

Present

Slides

Themes

Help



mynotebookinstance-1.notebook.us-west-2.sagemaker.aws

Amazon SageMaker

sample-notebooks/xgboost\_direct\_marketing/

xgboost\_direct\_marketing\_sagemaker

Run

Markdown

Edit Presentation

Show Presentation

```
contenttype : libsvm ,
"CompressionType": "None"
}
},
"OutputDataConfig": {
  "S3OutputPath": "s3://{}/{}/output".format(bucket, prefix)
},
"HyperParameters": {
  "max_depth": "5",
  "eta": "0.2",
  "gamma": "4",
  "min_child_weight": "6",
  "subsample": "0.8",
  "silent": "0",
  "objective": "binary:logistic",
  "num_class": "1",
  "num_round": "100",
  "train_file_name": "train.libsvm",
  "val_file_name": "validation.libsvm"
},
"StoppingCondition": {
  "MaxRuntimeInSeconds": 60 * 60
}
}

In [ ]: client = boto3.client('sagemaker')
client.create_training_job(**create_training_params)

status = client.describe_training_job(TrainingJobName=job_name)['TrainingJobStatus']
print(status)
client.get_waiter('TrainingJob_Created').wait(TrainingJobName=job_name)
status = client.describe_training_job(TrainingJobName=job_name)['TrainingJobStatus']
print("Training job ended with status: " + status)
if status == 'Failed':
    message = client.describe_training_job(TrainingJobName=job_name)['FailureReason']
    print('Training failed with the following error: {}'.format(message))
    raise Exception('Training job failed')
```

Hosting

Present

Slides

Themes

Help

mynotebookinstance-1.notebook.us-west-2.sagemaker.aws

Amazon SageMaker

sample-notebooks/xgboost\_direct\_marketing/

xgboost\_direct\_marketing\_sagemaker

Code

Edit Presentation

Show Presentation

```
        "max_depth": "5",
        "eta": "0.2",
        "gamma": "4",
        "min_child_weight": "6",
        "subsample": "0.8",
        "silent": "0",
        "objective": "binary:logistic",
        "num_class": "1",
        "num_round": "100",
        "train_file_name": "train.libsvm",
        "val_file_name": "validation.libsvm"
    },
    "StoppingCondition": {
        "MaxRuntimeInSeconds": 60 * 60
    }
}
```

```
In [ ]: client = boto3.client('sagemaker')
client.create_training_job(**create_training_params)

status = client.describe_training_job(TrainingJobName=job_name)['TrainingJobStatus']
print(status)
client.get_waiter('TrainingJob_Created').wait(TrainingJobName=job_name)
status = client.describe_training_job(TrainingJobName=job_name)['TrainingJobStatus']
print("Training job ended with status: " + status)
if status == 'Failed':
    message = client.describe_training_job(TrainingJobName=job_name)['FailureReason']
    print("Training failed with the following error: {}".format(message))
    raise Exception('Training job failed')
```

### Hosting

Now that we've trained the `xgboost` algorithm on our data, let's setup a model which can later be hosted.

```
In [ ]: primary_container = {
    'Image': "032969728358.dkr.ecr.us-west-2.amazonaws.com/xgboost-learner:latest",
    'ModelDataUrl': client.describe_training_job(TrainingJobName=job_name)['ModelArtifacts']['S3ModelArtifacts']
}
```

Present

Slides

Themes

Help



mynotebookinstance-1.notebook.us-west-2.sagemaker.aws

Amazon SageMaker

sample-notebooks/xgboost\_direct\_marketing/

xgboost\_direct\_marketing\_sagemaker

Run Edit Presentation Show Presentation

```
"max_depth":5,"eta":0.2,"gamma":4,"min_child_weight":6,"subsample":0.8,"silent":0,"objective":"binary:logistic","num_class":1,"num_round":100,"train_file_name":"train.libsvm","val_file_name":"validation.libsvm"},{"StoppingCondition":{"MaxRuntimeInSeconds":60*60}}
```

Training job xgboost-dm-2017-11-19-00-45-02

```
In [*]: client = boto3.client('sagemaker')
client.create_training_job(**create_training_params)

status = client.describe_training_job(TrainingJobName=job_name)['TrainingJobStatus']
print(status)
client.get_waiter('TrainingJob_Created').wait(TrainingJobName=job_name)
status = client.describe_training_job(TrainingJobName=job_name)['TrainingJobStatus']
print("Training job ended with status: " + status)
if status == 'Failed':
    message = client.describe_training_job(TrainingJobName=job_name)['FailureReason']
    print('Training failed with the following error: {}'.format(message))
    raise Exception('Training job failed')
```

InProgress

## Hosting

Now that we've trained the xgboost algorithm on our data, let's setup a model which can later be hosted.

```
In [ ]: primary_container = {
```

Present

Slides

Themes

Help

mynotebookinstance-1.notebook.us-west-2.sagemaker.aws

Amazon SageMaker

sample-notebooks/xgboost\_direct\_marketing/

xgboost\_direct\_marketing\_sagemaker

Run

Markdown

Edit Presentation

Show Presentation

```
        "eta":0.2",
        "gamma":4",
        "min_child_weight":6",
        "subsample":0.8",
        "silent":0",
        "objective":"binary:logistic",
        "num_class":1",
        "num_round":100",
        "train_file_name":"train.libsvm",
        "val_file_name":"validation.libsvm"
    },
    "StoppingCondition": {
        "MaxRuntimeInSeconds": 60 * 60
    }
}
```

Training job xgboost-dm-2017-11-19-00-45-02

```
In [24]: client = boto3.client('sagemaker')
client.create_training_job(**create_training_params)

status = client.describe_training_job(TrainingJobName=job_name)['TrainingJobStatus']
print(status)
client.get_waiter('TrainingJob_Created').wait(TrainingJobName=job_name)
status = client.describe_training_job(TrainingJobName=job_name)['TrainingJobStatus']
print("Training job ended with status: " + status)
if status == 'Failed':
    message = client.describe_training_job(TrainingJobName=job_name)['FailureReason']
    print('Training failed with the following error: {}'.format(message))
    raise Exception('Training job failed')
```

InProgress  
Training job ended with status: Completed

### Hosting

Now that we've trained the `xgboost` algorithm on our data, let's setup a model which can later be hosted.

```
In [ ]: primary_container = {
```

Present

Slides

Themes

Help

mynotebookinstance-1.notebook.us-west-2.sagemaker.aws

Amazon SageMaker

sample-notebooks/xgboost\_direct\_marketing/

xgboost\_direct\_marketing\_sagemaker

Run Edit Presentation Show Presentation

```
        "eta":0.2,
        "gamma":4,
        "min_child_weight":6,
        "subsample":0.8,
        "silent":0,
        "objective":"binary:logistic",
        "num_class":1,
        "num_round":100,
        "train_file_name":"train.libsvm",
        "val_file_name":"validation.libsvm"
    },
    "StoppingCondition": {
        "MaxRuntimeInSeconds": 60 * 60
    }
}
```

Training job xgboost-dm-2017-11-19-00-45-02

```
In [24]: client = boto3.client('sagemaker')
client.create_training_job(**create_training_params)

status = client.describe_training_job(TrainingJobName=job_name)['TrainingJobStatus']
print(status)
client.get_waiter('TrainingJob_Created').wait(TrainingJobName=job_name)
status = client.describe_training_job(TrainingJobName=job_name)['TrainingJobStatus']
print("Training job ended with status: " + status)
if status == 'Failed':
    message = client.describe_training_job(TrainingJobName=job_name)['FailureReason']
    print('Training failed with the following error: {}'.format(message))
    raise Exception('Training job failed')
```

InProgress  
Training job ended with status: Completed

### Hosting

Now that we've trained the `xgboost` algorithm on our data, let's setup a model which can later be hosted.

```
In [ ]: primary_container = {
```

Present

Slides

Themes

Help

mynotebookinstance-1.notebook.us-west-2.sagemaker.aws

Amazon SageMaker

sample-notebooks/xgboost\_direct\_marketing/

xgboost\_direct\_marketing\_sagemaker

Code

Edit Presentation

Show Presentation

## Hosting

Now that we've trained the `xgboost` algorithm on our data, let's setup a model which can later be hosted.

```
In [ ]: primary_container = {
    'Image': '032969728358.dkr.ecr.us-west-2.amazonaws.com/xgboost-learner:latest',
    'ModelDataUrl': client.describe_training_job(TrainingJobName=job_name)['ModelArtifacts']['S3ModelArtifacts']
}

create_model_response = client.create_model(
    ModelName = job_name,
    ExecutionRoleArn = role,
    PrimaryContainer = primary_container)

print(create_model_response['ModelArn'])
```

```
In [ ]: endpoint_config_name = 'xgboost-endpoint-config-' + strftime("%Y-%m-%d-%H-%M-%S", gmtime())
print(endpoint_config_name)
create_endpoint_config_response = client.create_endpoint_config(
    EndpointConfigName = endpoint_config_name,
    ProductionVariants=[{
        'InstanceType': 'ml.c4.xlarge',
        'InitialInstanceCount': 1,
        'ModelName': job_name,
        'VariantName': 'AllTraffic'}])

print("Endpoint Config Arn: " + create_endpoint_config_response['EndpointConfigArn'])
```

Now that we've specified how our endpoint should be configured, we can create them. This can be done in the background, but for now let's run a loop that updates us on the status of the endpoints so that we know when they are ready for use.

```
In [ ]: endpoint_name = 'xgboost-endpoint-' + strftime("%Y-%m-%d-%H-%M-%S", gmtime())
print(endpoint_name)
create_endpoint_response = client.create_endpoint(
    EndpointName=endpoint_name,
    EndpointConfigName=endpoint_config_name)
print(create_endpoint_response['EndpointArn'])
```

Present

Slides

Themes

Help

mynotebookinstance-1.notebook.us-west-2.sagemaker.aws

Amazon SageMaker

sample-notebooks/xgboost\_direct\_marketing/

xgboost\_direct\_marketing\_sagemaker

Code

Edit Presentation

Show Presentation

## Hosting

Now that we've trained the `xgboost` algorithm on our data, let's setup a model which can later be hosted.

```
In [25]: primary_container = {
    'Image': '032969728358.dkr.ecr.us-west-2.amazonaws.com/xgboost-learner:latest',
    'ModelDataUrl': client.describe_training_job(TrainingJobName=job_name)['ModelArtifacts']['S3ModelArtifacts']
}

create_model_response = client.create_model(
    ModelName = job_name,
    ExecutionRoleArn = role,
    PrimaryContainer = primary_container)

print(create_model_response['ModelArn'])

arn:aws:sagemaker:us-west-2:811689727410:model/xgboost-dm-2017-11-19-00-45-02
```

```
In [ ]: endpoint_config_name = 'xgboost-endpoint-config-' + strftime("%Y-%m-%d-%H-%M-%S", gmtime())
print(endpoint_config_name)
create_endpoint_config_response = client.create_endpoint_config(
    EndpointConfigName = endpoint_config_name,
    ProductionVariants=[{
        'InstanceType': 'ml.c4.xlarge',
        'InitialInstanceCount': 1,
        'ModelName': job_name,
        'VariantName': 'AllTraffic'})]

print("Endpoint Config Arn: " + create_endpoint_config_response['EndpointConfigArn'])
```

Now that we've specified how our endpoint should be configured, we can create them. This can be done in the background, but for now let's run a loop that updates us on the status of the endpoints so that we know when they are ready for use.

```
In [ ]: endpoint_name = 'xgboost-endpoint-' + strftime("%Y-%m-%d-%H-%M-%S", gmtime())
print(endpoint_name)
create_endpoint_response = client.create_endpoint(
    EndpointName=endpoint_name,
```

Present

Slides

Themes

Help

mynotebookinstance-1.notebook.us-west-2.sagemaker.aws

Amazon SageMaker

sample-notebooks/xgboost\_direct\_marketing/

xgboost\_direct\_marketing\_sagemaker

Code

Edit Presentation

Show Presentation

## Hosting

Now that we've trained the `xgboost` algorithm on our data, let's setup a model which can later be hosted.

```
In [25]: primary_container = {
    'Image': '032969728358.dkr.ecr.us-west-2.amazonaws.com/xgboost-learner:latest',
    'ModelDataUrl': client.describe_training_job(TrainingJobName=job_name)['ModelArtifacts']['S3ModelArtifacts']
}

create_model_response = client.create_model(
    ModelName = job_name,
    ExecutionRoleArn = role,
    PrimaryContainer = primary_container)

print(create_model_response['ModelArn'])

arn:aws:sagemaker:us-west-2:811689727410:model/xgboost-dm-2017-11-19-00-45-02
```

```
In [ ]: endpoint_config_name = 'xgboost-endpoint-config-' + strftime("%Y-%m-%d-%H-%M-%S", gmtime())
print(endpoint_config_name)
create_endpoint_config_response = client.create_endpoint_config(
    EndpointConfigName = endpoint_config_name,
    ProductionVariants=[{
        'InstanceType': 'ml.c4.xlarge',
        'InitialInstanceCount': 1,
        'ModelName': job_name,
        'VariantName': 'AllTraffic'}])

print("Endpoint Config Arn: " + create_endpoint_config_response['EndpointConfigArn'])
```

Now that we've specified how our endpoint should be configured, we can create them. This can be done in the background, but for now let's run a loop that updates us on the status of the endpoints so that we know when they are ready for use.

```
In [ ]: endpoint_name = 'xgboost-endpoint-' + strftime("%Y-%m-%d-%H-%M-%S", gmtime())
print(endpoint_name)
create_endpoint_response = client.create_endpoint(
    EndpointName=endpoint_name,
```

Present

Slides

Themes

Help



## Hosting

Now that we've trained the `xgboost` algorithm on our data, let's setup a model which can later be hosted.

```
In [25]: primary_container = {
    'Image': '032969728358.dkr.ecr.us-west-2.amazonaws.com/xgboost-learner:latest',
    'ModelDataUrl': client.describe_training_job(TrainingJobName=job_name)['ModelArtifacts']['S3ModelArtifacts']
}

create_model_response = client.create_model(
    ModelName = job_name,
    ExecutionRoleArn = role,
    PrimaryContainer = primary_container)

print(create_model_response['ModelArn'])

arn:aws:sagemaker:us-west-2:811689727410:model/xgboost-dm-2017-11-19-00-45-02
```

```
In [ ]: endpoint_config_name = 'xgboost-endpoint-config-' + strftime("%Y-%m-%d-%H-%M-%S", gmtime())
print(endpoint_config_name)
create_endpoint_config_response = client.create_endpoint_config(
    EndpointConfigName = endpoint_config_name,
    ProductionVariants=[{
        'InstanceType': 'ml.c4.xlarge',
        'InitialInstanceCount': 1,
        'ModelName': job_name,
        'VariantName': 'AllTraffic'}])

print("Endpoint Config Arn: " + create_endpoint_config_response['EndpointConfigArn'])
```

Now that we've specified how our endpoint should be configured, we can create them. This can be done in the background, but for now let's run a loop that updates us on the status of the endpoints so that we know when they are ready for use.

```
In [ ]: endpoint_name = 'xgboost-endpoint-' + strftime("%Y-%m-%d-%H-%M-%S", gmtime())
print(endpoint_name)
create_endpoint_response = client.create_endpoint(
    EndpointName=endpoint_name,
```

## Hosting

Now that we've trained the `xgboost` algorithm on our data, let's setup a model which can later be hosted.

```
In [25]: primary_container = {
        'Image': '032969728358.dkr.ecr.us-west-2.amazonaws.com/xgboost-learner:latest',
        'ModelDataUrl': client.describe_training_job(TrainingJobName=job_name)['ModelArtifacts']['S3ModelArtifacts']}

create_model_response = client.create_model(
    ModelName = job_name,
    ExecutionRoleArn = role,
    PrimaryContainer = primary_container)

print(create_model_response['ModelArn'])

arn:aws:sagemaker:us-west-2:811689727410:model/xgboost-dm-2017-11-19-00-45-02
```

```
In [ ]: endpoint_config_name = 'xgboost-endpoint-config-' + strftime("%Y-%m-%d-%H-%M-%S", gmtime())
print(endpoint_config_name)
create_endpoint_config_response = client.create_endpoint_config(
    EndpointConfigName = endpoint_config_name,
    ProductionVariants=[{
        'InstanceType': 'ml.c4.xlarge',
        'InitialInstanceCount': 1,
        'ModelName': job_name,
        'VariantName': 'AllTraffic'}])

print("Endpoint Config Arn: " + create_endpoint_config_response['EndpointConfigArn'])
```

Now that we've specified how our endpoint should be configured, we can create them. This can be done in the background, but for now let's run a loop that updates us on the status of the endpoints so that we know when they are ready for use.

```
In [ ]: endpoint_name = 'xgboost-endpoint-' + strftime("%Y-%m-%d-%H-%M-%S", gmtime())
print(endpoint_name)
create_endpoint_response = client.create_endpoint(
    EndpointName=endpoint_name,
```



mynotebookinstance-1.notebook.us-west-2.sagemaker.aws

Amazon SageMaker

sample-notebooks/xgboost\_direct\_marketing/

xgboost\_direct\_marketing\_sagemaker

Run

Markdown

Edit Presentation

Show Presentation

## Hosting

Now that we've trained the `xgboost` algorithm on our data, let's setup a model which can later be hosted.

```
In [25]: primary_container = {
        'Image': '032969728358.dkr.ecr.us-west-2.amazonaws.com/xgboost-learner:latest',
        'ModelDataUrl': client.describe_training_job(TrainingJobName=job_name)['ModelArtifacts']['S3ModelArtifacts']
    }

    create_model_response = client.create_model(
        ModelName = job_name,
        ExecutionRoleArn = role,
        PrimaryContainer = primary_container)

    print(create_model_response['ModelArn'])

arn:aws:sagemaker:us-west-2:811689727410:model/xgboost-dm-2017-11-19-00-45-02
```

```
In [26]: endpoint_config_name = 'xgboost-endpoint-config-' + strftime("%Y-%m-%d-%H-%M-%S", gmtime())
        print(endpoint_config_name)
        create_endpoint_config_response = client.create_endpoint_config(
            EndpointConfigName = endpoint_config_name,
            ProductionVariants=[{
                'InstanceType': 'ml.c4.xlarge',
                'InitialInstanceCount': 1,
                'ModelName': job_name,
                'VariantName': 'AllTraffic'})]

        print("Endpoint Config Arn: " + create_endpoint_config_response['EndpointConfigArn'])

xgboost-endpoint-config-2017-11-19-00-53-48
Endpoint Config Arn: arn:aws:sagemaker:us-west-2:811689727410:endpoint-config/xgboost-endpoint-config-2017-11-19-00-53-48
```

Now that we've specified how our endpoint should be configured, we can create them. This can be done in the background, but for now let's run a loop that updates us on the status of the endpoints so that we know when they are ready for use.

Present

Slides

Themes

Help

mynotebookinstance-1.notebook.us-west-2.sagemaker.aws

Amazon SageMaker

sample-notebooks/xgboost\_direct\_marketing/

xgboost\_direct\_marketing\_sagemaker

Run

Markdown

Edit Presentation

Show Presentation

## Hosting

Now that we've trained the `xgboost` algorithm on our data, let's setup a model which can later be hosted.

```
In [25]: primary_container = {
        'Image': '032969728358.dkr.ecr.us-west-2.amazonaws.com/xgboost-learner:latest',
        'ModelDataUrl': client.describe_training_job(TrainingJobName=job_name)['ModelArtifacts']['S3ModelArtifacts']
    }

    create_model_response = client.create_model(
        ModelName = job_name,
        ExecutionRoleArn = role,
        PrimaryContainer = primary_container)

    print(create_model_response['ModelArn'])

arn:aws:sagemaker:us-west-2:811689727410:model/xgboost-dm-2017-11-19-00-45-02
```

In [26]: endpoint\_config\_name = 'xgboost-endpoint-config-' + strftime("%Y-%m-%d-%H-%M-%S", gmtime())
 print(endpoint\_config\_name)
 create\_endpoint\_config\_response = client.create\_endpoint\_config(
 EndpointConfigName = endpoint\_config\_name,
 ProductionVariants=[{
 'InstanceType': 'ml.c4.xlarge',
 'InitialInstanceCount': 1,
 'ModelName': job\_name,
 'VariantName': 'AllTraffic'})]

 print("Endpoint Config Arn: " + create\_endpoint\_config\_response['EndpointConfigArn'])

xgboost-endpoint-config-2017-11-19-00-53-48
Endpoint Config Arn: arn:aws:sagemaker:us-west-2:811689727410:endpoint-config/xgboost-endpoint-config-2017-11-19-00-53-48

Now that we've specified how our endpoint should be configured, we can create them. This can be done in the background, but for now let's run a loop that updates us on the status of the endpoints so that we know when they are ready for use.

```
In [27]: endpoint_name = 'xgboost-endpoint-' + strftime("%Y-%m-%d-%H-%M-%S", gmtime())
```

Present

Slides

Themes

Help

mynotebookinstance-1.notebook.us-west-2.sagemaker.aws

Amazon SageMaker

sample-notebooks/xgboost\_direct\_marketing/

xgboost\_direct\_marketing\_sagemaker

Run

Markdown

Edit Presentation

Show Presentation

## Hosting

Now that we've trained the `xgboost` algorithm on our data, let's setup a model which can later be hosted.

```
In [25]: primary_container = {
        'Image': '032969728358.dkr.ecr.us-west-2.amazonaws.com/xgboost-learner:latest',
        'ModelDataUrl': client.describe_training_job(TrainingJobName=job_name)['ModelArtifacts']['S3ModelArtifacts']
    }

    create_model_response = client.create_model(
        ModelName = job_name,
        ExecutionRoleArn = role,
        PrimaryContainer = primary_container)

    print(create_model_response['ModelArn'])

arn:aws:sagemaker:us-west-2:811689727410:model/xgboost-dm-2017-11-19-00-45-02
```

```
In [26]: endpoint_config_name = 'xgboost-endpoint-config-' + strftime("%Y-%m-%d-%H-%M-%S", gmtime())
        print(endpoint_config_name)
        create_endpoint_config_response = client.create_endpoint_config(
            EndpointConfigName = endpoint_config_name,
            ProductionVariants=[{
                'InstanceType': 'ml.c4.xlarge',
                'InitialInstanceCount': 1,
                'ModelName': job_name,
                'VariantName': 'AllTraffic'})]

        print("Endpoint Config Arn: " + create_endpoint_config_response['EndpointConfigArn'])

xgboost-endpoint-config-2017-11-19-00-53-48
Endpoint Config Arn: arn:aws:sagemaker:us-west-2:811689727410:endpoint-config/xgboost-endpoint-config-2017-11-19-00-53-48
```

Now that we've specified how our endpoint should be configured, we can create them. This can be done in the background, but for now let's run a loop that updates us on the status of the endpoints so that we know when they are ready for use.

```
To do this, we can use the following code:
```

Present

Slides

Themes

Help

mynotebookinstance-1.notebook.us-west-2.sagemaker.aws

Amazon SageMaker

sample-notebooks/xgboost\_direct\_marketing/

xgboost\_direct\_marketing\_sagemaker

Run

Markdown

Edit Presentation

Show Presentation

## Hosting

Now that we've trained the `xgboost` algorithm on our data, let's setup a model which can later be hosted.

```
In [25]: primary_container = {
        'Image': '032969728358.dkr.ecr.us-west-2.amazonaws.com/xgboost-learner:latest',
        'ModelDataUrl': client.describe_training_job(TrainingJobName=job_name)['ModelArtifacts']['S3ModelArtifacts']
    }

    create_model_response = client.create_model(
        ModelName = job_name,
        ExecutionRoleArn = role,
        PrimaryContainer = primary_container)

    print(create_model_response['ModelArn'])

arn:aws:sagemaker:us-west-2:811689727410:model/xgboost-dm-2017-11-19-00-45-02
```

```
In [26]: endpoint_config_name = 'xgboost-endpoint-config-' + strftime("%Y-%m-%d-%H-%M-%S", gmtime())
        print(endpoint_config_name)
        create_endpoint_config_response = client.create_endpoint_config(
            EndpointConfigName = endpoint_config_name,
            ProductionVariants=[{
                'InstanceType': 'ml.c4.xlarge',
                'InitialInstanceCount': 1,
                'ModelName': job_name,
                'VariantName': 'AllTraffic'})]

        print("Endpoint Config Arn: " + create_endpoint_config_response['EndpointConfigArn'])

xgboost-endpoint-config-2017-11-19-00-53-48
Endpoint Config Arn: arn:aws:sagemaker:us-west-2:811689727410:endpoint-config/xgboost-endpoint-config-2017-11-19-00-53-48
```

Now that we've specified how our endpoint should be configured, we can create them. This can be done in the background, but for now let's run a loop that updates us on the status of the endpoints so that we know when they are ready for use.

```
To do this, we can use the following code:
```

Present

Slides

Themes

Help

mynotebookinstance-1.notebook.us-west-2.sagemaker.aws

Amazon SageMaker

sample-notebooks/xgboost\_direct\_marketing/

xgboost\_direct\_marketing\_sagemaker

Code

Edit Presentation

Show Presentation

## Evaluation

There are many ways to compare the performance of a machine learning model, but let's start by simply by comparing actual to predicted values. In this case, we're simply predicting whether the customer subscribed to a term deposit ( 1 ) or not ( 0 ), which produces a simple confusion matrix.

```
In [28]: runtime = boto3.client('sagemaker-runtime')

In [29]: def do_predict(data, endpoint_name, content_type):
    payload = '\n'.join(data)
    response = runtime.invoke_endpoint(EndpointName=endpoint_name,
                                      ContentType=content_type,
                                      Body=payload)

    result = response['Body'].read()
    result = result.decode('utf-8')
    result = result.split(',')
    preds = [float(num) for num in result]
    preds = [round(num) for num in preds]
    return preds

def batch_predict(data, batch_size, endpoint_name, content_type):
    items = len(data)
    arrs = []

    for offset in range(0, items, batch_size):
        if offset+batch_size < items:
            results = do_predict(data[offset:(offset+batch_size)], endpoint_name, content_type)
            arrs.extend(results)
        else:
            arrs.extend(do_predict(data[offset:items], endpoint_name, content_type))
        sys.stdout.write('.')
    return(arrs)

In [ ]: %time
import json

with open('test.libsvm', 'r') as f:
    payload = f.read().strip()

labels = [int(line.split(' ')[0]) for line in payload.split('\n')]
test_data = [line for line in payload.split('\n')]
preds = batch_predict(test_data, 1000, endpoint_name, 'text/x-libsvm')
```

Present

Slides

Themes

Help

mynotebookinstance-1.notebook.us-west-2.sagemaker.aws

Amazon SageMaker

sample-notebooks(xgboost\_direct\_marketing)

xgboost\_direct\_marketing.sagemaker

Code

Edit Presentation

Show Presentation

```
        Body=payload)

    result = response['Body'].read()
    result = result.decode("utf-8")
    result = result.split(',')
    preds = [float(num) for num in result]
    preds = [round(num) for num in preds]
    return preds

def batch_predict(data, batch_size, endpoint_name, content_type):
    items = len(data)
    arrs = []

    for offset in range(0, items, batch_size):
        if offset+batch_size < items:
            results = do_predict(data[offset:(offset+batch_size)], endpoint_name, content_type)
            arrs.extend(results)
        else:
            arrs.extend(do_predict(data[offset:items], endpoint_name, content_type))
        sys.stdout.write('.')
    return(arrs)

In [34]: %time
import json

with open('test.libsvm', 'r') as f:
    payload = f.read().strip()

labels = [int(line.split(' ')[0]) for line in payload.split('\n')]
test_data = [line for line in payload.split('\n')]
preds = batch_predict(test_data, 1000, endpoint_name, 'text/x-libsvm')

print '\nerror rate=%f' % (sum(1 for i in range(len(preds)) if preds[i]!=labels[i]) / float(len(preds)))

.....
error rate=0.095897
CPU times: user 32 ms, sys: 0 ns, total: 32 ms
Wall time: 650 ms

In [ ]:
```

Present

Slides

Themes

Help





00:41 / 10:17



# Challenge: Operationalization



01

## Framework

Choose a framework that is best-suited for the task at hand.

© 2018 Amazon Web Services, Inc. or its Affiliates. All rights reserved.



00:56 / 10:17





# Challenge: Operationalization



01

## Framework

Choose a framework that is best-suited for the task at hand.

## Models

02

Build the models using the chosen framework.

03

## Train Models to Make Predictions

Train a model using sample data to make accurate predictions on bigger data sets.



# Challenge: Operationalization



01

## Framework

Choose a framework that is best-suited for the task at hand.

## Models

02

Build the models using the chosen framework.

03

## Train Models to Make Predictions

Train a model using sample data to make accurate predictions on bigger data sets.

## Integrate

04

Integrate the model with the application.

05

## Deploy

Deploy the application, the model, and the framework on a platform.

# Amazon SageMaker



**Build**

**Train**

**Deploy**



Machine  
Learning  
Model

© 2018 Amazon Web Services, Inc. or its affiliates. All rights reserved.



01:32 / 10:17



# Amazon SageMaker



Amazon  
SageMaker



# How to Deploy a Model With...



Amazon  
SageMaker

© 2018 Amazon Web Services, Inc. or its Affiliates. All rights reserved.



# How to Deploy a Model With...



**Optimal  
Performance**

on

**Multiple  
Platforms**



© 2018 Amazon Web Services, Inc. or its affiliates. All rights reserved.





# How to Deploy a Model With...



**Optimal  
Performance**

on

TensorFlow XGBoost  
mxnet PyTorch

**Multiple  
Platforms**





# Problem: Many to Many



Platform

© 2018 Amazon Web Services, Inc. or its Affiliates. All rights reserved.

