

Cloud Guru - 5 - Modeling Lab

<https://acloud.guru/course/aws-certified-machine-learning-specialty/learn/87700af1-74be-5d7b-2143-9e031dd0fb1d/21ddf3e2-05df-c4db-a8df-877d8c5d52e3/watch?backUrl=~2Fcourses&backUrl=~2Fcourses&backUrl=~2Fcourses,~2Fcourses>

The slide has a dark blue background with a large, faint image of a spiral galaxy on the right side. At the top left, it says "MODELING LAB" and at the top right is the "A CLOUD GURU" logo.

Identifying Sensor Locations

Mr. K wants to deploy a network of 10 sensors across the globe. He would like to locate these sensors in the center of the 10 most likely locations for UFO sighting.

What type of generalization are we trying to make?

Do we have enough data? What does our data look like?

What algorithm can help us solve this problem?

Where should we launch the sensors?

A sensor would look like something like this:



Data set (synthetic):

UFO Sightings Dataset

- **reportedTimestamp:** The date when the UFO sighting was reported
- **eventDate:** The date when the UFO sighting happened
- **eventTime:** The time when the UFO sighting happened (24 hour format)
- **latitude:** The location where the UFO sighting happened
- **longitude:** The location where the UFO sighting happened
- **shape:** The shape of the UFO
- **duration:** How many seconds the sighting happened
- **witnesses:** The amount of witnesses that saw the UFO
- **weather:** The weather when the sighting happened
- **firstName:** The first name of the person who saw the UFO
- **lastName:** The last name of the person who saw the UFO
- **sighting:** If a UFO was sighted (is always Y = Yes)
- **physicalEvidence:** Whether there was physical evidence left by UFO (Y = Yes, N = No)
- **contact:** Whether there was contact by UFO beings (Y = Yes, N = No)
- **researchOutcome:** Whether the experts had an explanation for sighting
 - **explained:** There was some explanation to the sighting
 - **unexplained:** There was no explanation to the sighting
 - **probable:** There is enough evidence that the sighting were from extraterrestrials

Find 10 locations and map them

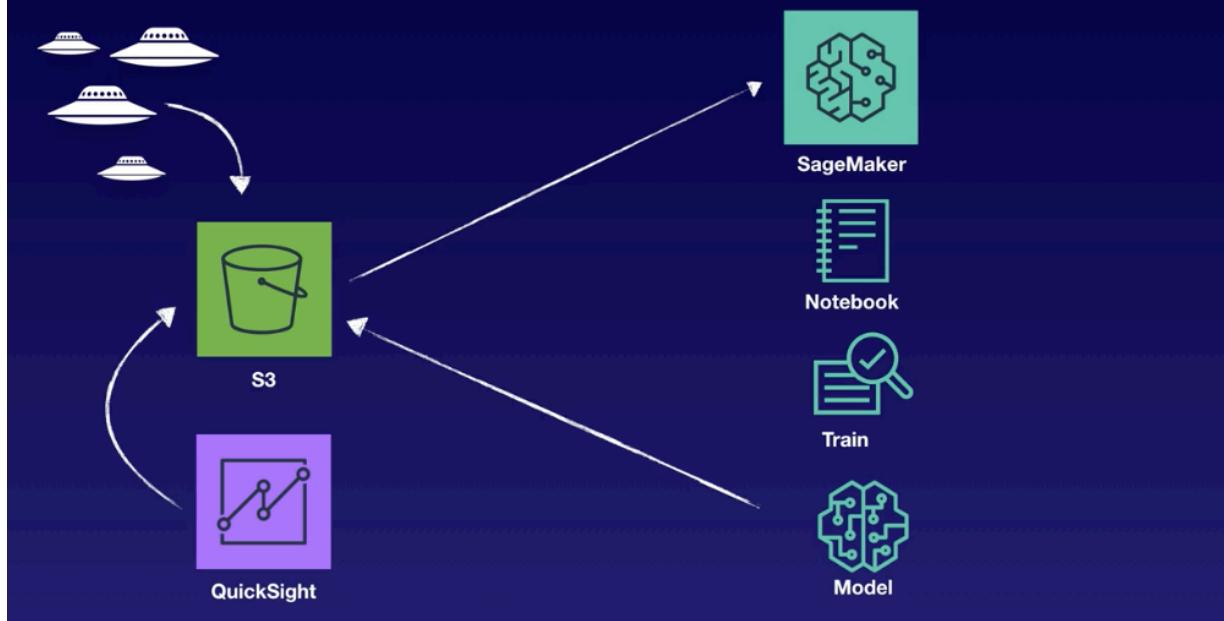
- 10 locations (latitude and longitude)
- Map of locations



Approach:

Use Amazon SageMaker and the K-Means clustering algorithm to locate the 10 best locations.

Visualize the locations in QuickSight.



To note we could create a Training job with SageMaker IDE

The screenshot shows the 'Create training job' page in the Amazon SageMaker console. The top navigation bar includes 'Amazon SageMaker > Training jobs > Create training job'. The main section is titled 'Create training job' and contains the following fields:

- Job settings**:
 - Job name**: A text input field with placeholder text: "Maximum of 63 alphanumeric characters. Can include hyphens (-), but not spaces. Must be unique within your account in an AWS Region." A cursor is shown in the input field.
 - IAM role**: A dropdown menu labeled "Choose an IAM role".
- Algorithm options**: A section describing algorithm selection with a note: "Use an Amazon SageMaker built-in algorithm, your own algorithm, or a third-party algorithm from AWS Marketplace."
- Algorithm source**: A dropdown menu.

But we are going to use a notebook instead.

Amazon SageMaker > Notebook instances > Create notebook instance

Create notebook instance

Amazon SageMaker provides pre-built fully managed notebook instances that run Jupyter notebooks. The notebook instances include example code for common model training and hosting exercises. [Learn more](#)

Notebook instance settings

Notebook instance name

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

Notebook instance type

mLt2.medium

Elastic Inference [Learn more](#)

none

► Additional configuration

Then we can upload the notebook, or Create a new one from scratch.
Conda-python3 env

jupyter ufo-modeling-lab Last Checkpoint: 20 hours ago (unsaved changes)

File Edit View Insert Cell Kernel Widgets Help Trusted | conda_python3 O

UFO Sightings K-Means Clustering

Modeling Lab

The goal of this notebook is to analyze where Mr. K should build his extraterrestrial life facilities using the K-Means algorithm.

What we plan on accomplishing is the following:

1. [Load dataset onto Notebook instance from S3](#)
2. [Cleaning, transforming, and preparing the data](#)
3. [Create and train our model](#)
4. [Viewing the results](#)
5. [Visualize using QuickSight](#)

First let's go ahead and import all the needed libraries.

```
In [ ]: import pandas as pd
import numpy as np
from datetime import datetime

import boto3
from sagemaker import get_execution_role
import sagemaker.amazon.common as smac
```

Create Data Frame with UFO data set

Step 1: Loading the data from Amazon S3

Next, let's get the UFO sightings data that is stored in S3.

```
In [2]: role = get_execution_role()
bucket = 'ml-labs-acg'
prefix = 'ufo_dataset'
data_key = 'ufo_fullset.csv'
data_location = 's3://{}//{}//{}'.format(bucket, prefix, data_key)

df = pd.read_csv(data_location, low_memory=False)
```

```
In [3]: df.head()
```

```
Out[3]:
```

	reportedTimestamp	eventDate	eventTime	shape	duration	witnesses	weather	firstName	lastName	latitude	longitude	sighting	physicalEvidence	category
0	1982-11-29T10:01:48.297Z	1982-11-28	03:17	oval	71	1	snow	Muriel	Bartell	28.039167	-81.950000	Y	N	UFO
1	2006-03-05T18:36:08.186Z	2006-03-05	04:56	light	75	1	partly cloudy	Floy	Heaney	33.660278	-117.998333	Y	Y	UFO
2	2002-07-31T23:33:55.223Z	2002-07-26	13:43	oval	25	1	rain	Evelyn	Champlin	41.325278	-72.193611	Y	Y	UFO
3	1986-08-31T00:50:08.017Z	1986-08-27	16:12	sphere	47	1	mostly cloudy	Holden	Ward	38.254167	-85.759444	Y	N	UFO
4	2004-09-26T08:47:39.860Z	2004-09-25	17:21	disk	59	1	rain	Abigail	Grady	22.308085	69.600603	Y	N	UFO

Rows and attributes: 18k rows and 15 columns (attributes)

```
In [4]: df.shape
```

```
Out[4]: (18000, 15)
```

Step 2: Cleaning, transforming, and preparing the data

Create another DataFrame with just the latitude and longitude attributes

```
In [5]: df_geo = df[['latitude', 'longitude']]
```

```
In [6]: df_geo.head()
```

```
Out[6]:
```

	latitude	longitude
0	28.039167	-81.950000
1	33.660278	-117.998333
2	41.325278	-72.193611
3	38.254167	-85.759444
4	22.308085	69.600603

```
In [7]: df_geo.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 18000 entries, 0 to 17999
Data columns (total 2 columns):
latitude    18000 non-null float64
longitude   18000 non-null float64
dtypes: float64(2)
memory usage: 281.3 KB
```

```
In [8]: missing_values = df_geo.isnull().values.any()
print('Are there any missing values? {}'.format(missing_values))
if(missing_values):
    df_geo[df_geo.isnull().any(axis=1)]
Are there any missing values? False
```

Next, let's go ahead and transform the pandas DataFrame (our dataset) into a numpy.ndarray. When we do this each row is converted to a Record object. According to the documentation, this is what the K-Means algorithm expects as training data. This is what we will use as training data for our model.

[See the documentation for input training](#)

K-means reference:

<https://sagemaker.readthedocs.io/en/stable/algorithms/kmeans.html>

K-means algorithm is expecting float 32 (not 64)

```
In [9]: data_train = df_geo.values.astype('float32')
data_train
Out[9]: array([[ 28.039167, -81.95      ],
   [ 33.66028 , -117.99834 ],
   [ 41.32528 , -72.19361 ],
   ...,
   [ 37.49472 , -120.84556 ],
   [ 40.771946, -73.93056 ],
   [ 64.837776, -147.71638 ]], dtype=float32)
```

K-means hyper parameters:

<https://docs.aws.amazon.com/sagemaker/latest/dg/k-means-api-config.html>

Now build our model

Step 3: Create and train our model

In this step we will import and use the built-in SageMaker K-Means algorithm. We will set the number of cluster to 10 (for our 10 sensors), specify the instance type we want to train on, and the location of where we want our model artifact to live.

[See the documentation of hyperparameters here](#)

```
In [ ]: from sagemaker import KMeans

num_clusters = 10
output_location = 's3://' + bucket + '/model-artifacts'

kmeans = KMeans(role=role,
                 train_instance_count=1,
                 train_instance_type='ml.c4.xlarge',
                 output_path=output_location,
                 k=num_clusters)
```

And train our model

```
In [11]: job_name = 'kmeans-geo-job-{}'.format(datetime.now().strftime("%Y%m%d%H%M%S"))
print('Here is the job name {}'.format(job_name))

Here is the job name kmeans-geo-job-20190517133512

In [14]: %%time
kmeans.fit(kmeans.record_set(data_train), job_name=job_name)

2019-05-17 13:37:19 Starting - Starting the training job...
2019-05-17 13:37:21 Starting - Launching requested ML instances.....
2019-05-17 13:38:26 Starting - Preparing the instances for training.....
2019-05-17 13:39:44 Downloading - Downloading input data..

2019-05-17 13:40:12 Training - Training image download completed. Training in progress.
2019-05-17 13:40:12 Uploading - Uploading generated training model
2019-05-17 13:40:12 Completed - Training job completed
Docker entrypoint called with argument(s): train
[05/17/2019 13:40:02 INFO 140619722299200] Reading default configuration from /opt/amazon/lib/python2.7/site-packages/algorithms/resources/default-input.json: {u'_enable_profiler': u'false', u'_tuning_objective_metric': u'', u'_num_gpus': u'auto', u'_local_lloyd_num_trials': u'auto', u'_log_level': u'info', u'_kvstore': u'auto', u'_local_lloyd_init_method': u'kmeans++', u'_force_dense': u'true', u'_epochs': u'1', u'_init_method': u'random', u'_local_lloyd_tol': u'0.0001', u'_local_lloyd_max_iter': u'300', u'_disable_wait_to_read': u>false', u'_extra_center_factor': u'auto', u'_eval_metrics': u'["msd"]', u'_num_kv_servers': u'1', u'_mini_batch_size': u'5000', u'_half_life_time_size': u'0', u'_num_slices': u'1'}
```

This cell will use our 18000 latitudes and longitudes and create 10 clusters and store that data in S3 in a sub-folder called "model-artifacts"

The red is the output of log files from docker image

```
[05/17/2019 13:40:02 INFO 140619722299200] Reading provided configuration from /opt/ml/input/config/hyperparameters.json: {u'feature_dim': u'2', u'k': u'10', u'force_dense': u'True'}
[05/17/2019 13:40:02 INFO 140619722299200] Final configuration: {u'_tuning_objective_metric': u'', u'_extra_center_factor': u'auto', u'_local_lloyd_init_method': u'kmeans++', u'_force_dense': u'True', u'_epochs': u'1', u'_feature_dim': u'2', u'_local_lloyd_tol': u'0.0001', u'_disable_wait_to_read': u>false', u'_eval_metrics': u'["msd"]', u'_num_kv_servers': u'1', u'_mini_batch_size': u'5000', u'_enable_profiler': u'false', u'_num_gpus': u'auto', u'_local_lloyd_num_trials': u'auto', u'_log_level': u'info', u'_init_method': u'random', u'_half_life_time_size': u'0', u'_local_lloyd_max_iter': u'300', u'_kvstore': u'auto', u'_k': u'10', u'_num_slices': u'1'}
```

We can now retrieve the results in S3

Name	Last modified	Size	Storage class
kmeans-geo-job-20190517133512	--	--	--

The mode is located in S3:

Amazon S3 > ml-labs-acg > model-artifacts > kmeans-geo-job-20190517133512 > output

Overview

Type a prefix and press Enter to search. Press ESC to clear.

Upload Create folder Download Actions US East (N. Virginia) Viewing 1 to 1

Name	Last modified	Size	Storage class
model.tar.gz	May 17, 2019 9:40:08 AM GMT-0400	1.4 KB	Standard

Viewing 1 to 1

We can now deploy it and let people make inferences on it.
We could also deserialize it and see it

Step 4: Viewing the results

In this step we will take a look at the model artifact SageMaker created for us and stored onto S3. We have to do a few special things to see the latitude and longitude for our 10 clusters (and the center points of those clusters)

[See the documentation of deserialization here](#)

At this point we need to "deserialize" the model artifact. Here we are going to open and review them in our notebook instance. We can unzip the model artifact which will contain `model_algo-1`. This is just a serialized Apache MXNet object. From here we can load that serialized object into a `numpy.ndarray` and then extract the clustered centroids from the `numpy.ndarray`.

After we extract the results into a DataFrame of latitudes and longitudes, we can create a CSV with that data, load it onto S3 and then visualize it with QuickSight.

passed as part of `InputDataConfig` in `create_training_job`.

Note

For all Amazon SageMaker algorithms, the `ChannelName` in `InputDataConfig` must be set to `train`. Some algorithms also support a validation or test input channels. These are typically used to evaluate the model's performance by using a hold-out dataset. Hold-out datasets are not used in the initial training but can be used to further tune the model.

Trained Model Deserialization

Amazon SageMaker models are stored as `model.tar.gz` in the S3 bucket specified in `OutputDataConfig.S3OutputPath` parameter of the `create_training_job` call. You can specify most of these model artifacts when creating a hosting model. You can also open and review them in your notebook instance. When `model.tar.gz` is untarred, it contains `model_algo-1`, which is a serialized Apache MXNet object. For example, you use the following to load the k-means model into memory and view it:

```
import mxnet as mx
print(mx.ndarray.load('model_algo-1'))
```

Document Conventions

Download model from S3 onto notebook, unzip it

```
In [17]: import os
model_key = 'model-artifacts/' + job_name + '/output/model.tar.gz'

boto3.resource('s3').Bucket(bucket).download_file(model_key, 'model.tar.gz')
os.system('tar -zxf model.tar.gz')
os.system('unzip model_algo-1')

Out[17]: 2304
```

Select items to perform actions on them.		
	Name	Last Modified
<input type="checkbox"/>	0	
<input type="checkbox"/>	ufo-modeling-lab.ipynb	Running a minute ago 29.3 kB
<input type="checkbox"/>	model.tar.gz	seconds ago 1.41 kB
<input type="checkbox"/>	model_algo-1	8 minutes ago 152 B
<input type="checkbox"/>	state_b5c6eca3-3736-409d-9776-c3c271435d88	8 minutes ago 1.33 kB

Now use MXnet library to deserialize it.

Since it's note installed on the notebook, install it fist

```
In [18]: pip install mxnet
      (from mxnet) (2.20.1)
      Collecting graphviz<0.9.0,>=0.8.1 (from mxnet)
        Downloading https://files.pythonhosted.org/packages/53/39/4ab213673844e0c004bed8a0781a0721a3f6bb23eb8854ee75c236428
        892/graphviz-0.8.4-py3-none-any.whl
      Requirement already satisfied: urllib3<1.25,>=1.21.1 in /home/ec2-user/anaconda3/envs/python3/lib/python3.6/site-packages (from requests>=2.20.0->mxnet) (1.23)
      Requirement already satisfied: idna<2.8,>=2.5 in /home/ec2-user/anaconda3/envs/python3/lib/python3.6/site-packages (from requests>=2.20.0->mxnet) (2.6)
      Requirement already satisfied: certifi>=2017.4.17 in /home/ec2-user/anaconda3/envs/python3/lib/python3.6/site-packages (from requests>=2.20.0->mxnet) (2019.3.9)
      Requirement already satisfied: chardet<3.1.0,>=3.0.2 in /home/ec2-user/anaconda3/envs/python3/lib/python3.6/site-packages (from requests>=2.20.0->mxnet) (3.0.4)
      Installing collected packages: numpy, graphviz, mxnet
        Found existing installation: numpy 1.15.4
        Uninstalling numpy-1.15.4:
          Successfully uninstalled numpy-1.15.4
      Successfully installed graphviz-0.8.4 mxnet-1.4.1 numpy-1.14.6
      You are using pip version 10.0.1, however version 19.1.1 is available.
      You should consider upgrading via the 'pip install --upgrade pip' command.
```

```
In [19]: import mxnet as mx
Kmeans_model_params = mx.ndarray.load('model_algo-1')
```

Now create a data frame that has the centroid of the clusters

```
In [20]: cluster_centroids_kmeans = pd.DataFrame(Kmeans_model_params[0].asnumpy())
cluster_centroids_kmeans.columns=df_geo.columns
cluster_centroids_kmeans
```

	latitude	longitude
0	41.286369	-74.856453
1	-3.558636	115.825752
2	35.375927	-117.235794
3	48.477852	3.664200
4	36.134438	-97.897385
5	26.707329	-81.378113
6	46.405426	-120.561981
7	25.992533	-146.748108
8	38.832069	-85.299072
9	61.760826	-148.924332

Now convert to CSV and upload to S3

```
In [23]: from io import StringIO
csv_buffer = StringIO()
cluster_centroids_kmeans.to_csv(csv_buffer, index=False)
s3_resource = boto3.resource('s3')
s3_resource.Object(bucket, 'results/ten_locations_kmeans.csv').put(Body=csv_buffer.getvalue())
Out[23]: {'ResponseMetadata': {'RequestId': '5B2C5F338C4D94A2',
'HostId': 'EBaTdqW46uapRaIWfzr0UMENSLV4vuhXsUML5S9b4QC4MP0heG2FECRYKJqYeSum2J8ikhHdrY=',
'HTTPStatusCode': 200,
'HTTPHeaders': {'x-amz-id-2': 'EBaTdqW46uapRaIWfzr0UMENSLV4vuhXsUML5S9b4QC4MP0heG2FECRYKJqYeSum2J8ikhHdrY=',
'x-amz-request-id': '5B2C5F338C4D94A2',
'date': 'Fri, 17 May 2019 13:53:38 GMT',
'etag': '"51e129efa7a05a163e90bd3fd0433c70"',
'content-length': '0',
'server': 'AmazonS3',
'RetryAttempts': 0,
'ETag': '"51e129efa7a05a163e90bd3fd0433c70"'}}
```

We now have a CSV file with 10 latitudes and longitudes

The screenshot shows the AWS S3 console interface. At the top, there's a navigation bar with the AWS logo, 'Services' dropdown, 'Resource Groups' dropdown, and user information ('Brock Tubre - ACG', 'Global', 'Support'). Below the navigation is a breadcrumb trail: 'Amazon S3 > ml-labs-acg > results'. The main area is titled 'Overview' and contains a search bar with placeholder text 'Type a prefix and press Enter to search. Press ESC to clear.' Below the search bar are buttons for 'Upload', '+ Create folder', 'Download', and 'Actions'. To the right of these buttons, it says 'US East (N. Virginia)' and has a refresh icon. A table below lists the contents of the bucket, with one item: 'ten_locations_kmeans.csv'. The table has columns: 'Name', 'Last modified', 'Size', and 'Storage class'. The file 'ten_locations_kmeans.csv' is listed with a size of '223.0 B' and 'Standard' storage class. At the bottom of the table, it says 'Viewing 1 to 1'.

Now we can use it with Quicksight



We select "points on map"

A screenshot of the QuickSight interface. The URL in the browser bar is https://us-east-1.quicksight.aws.amazon.com/sn/analyses/a0fd7123-f960-4899-aece-5949901a938f. The main workspace shows a data set named "SPICE modeling-lab-d..." containing fields "latitude" and "longitude". On the left, there are navigation buttons for "Add", "Undo", and "Redo", and sections for "Visualize", "Filter", "Story", and "Parameters". The "Visual types" section is open, displaying various chart icons. A tooltip for the "Points on map" icon is visible, stating: "Points on map: At least 1 geospatial field in Geospatial or 1 latitude and 1 longitude field in Geospatial". In the top right, there's a "Capture" button, a "Share" button, and a user profile for "N. Virginia brockt...". A modal window titled "Importing:" shows the status: "0 rows were imported to SPICE" and "0 rows were skipped".

And we can see the map where to launch the sensors

