

CS498 IoT Lab 4

Rick Bischoff (rdb4@illinois.edu)

<https://gitlab.engr.illinois.edu/rdb4/sp21-cs498it-lab4>

https://github.com/rdbisch/cs498_iot_lab4

<https://www.youtube.com/watch?v=U6nZxosLLsc>

Part 1 - Machine Learning

Step 1 - Acquiring Dataset

```
(lab4_venv) rdbisch@Rbisc150165:~/CS498_IOT/lab4$ ls -al data/
total 9986028
drwxr-xr-x 1 rdbisch rdbisch      512 Apr 14 16:35 .
drwxr-xr-x 1 rdbisch rdbisch      512 Apr 18 16:34 ..
-rwxr-xr-x 1 rdbisch rdbisch 4988035880 Mar 17 16:34 UC-Emission.xml
-rw-r--r-- 1 rdbisch rdbisch 2618050702 Mar 17 16:57 emission.csv
-rw-r--r-- 1 rdbisch rdbisch 2618050702 Apr 14 16:41 emission_comma.csv
-rw-r--r-- 1 rdbisch rdbisch   1545777 Mar 17 16:58 emission_small.csv
(lab4_venv) rdbisch@Rbisc150165:~/CS498_IOT/lab4$ |
```

Step 2 - Jupyter Notebook

See "Notebook_Practice.ipynb" in git repo.

Step 3 - Data Ingestion

See also Step1.

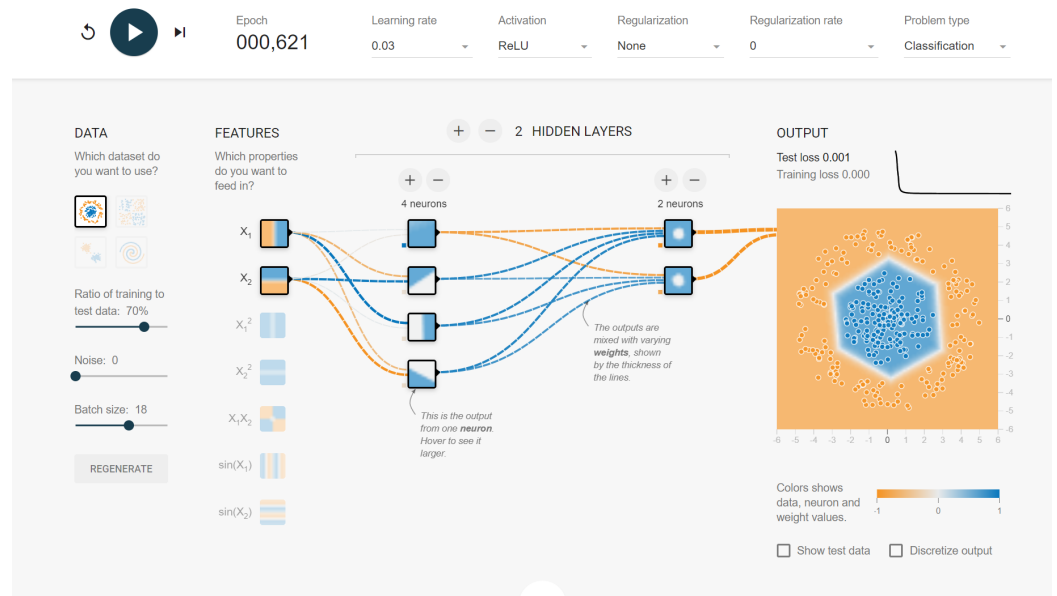
```
ec2-user@ip-172-31-0-191:~ x C:\Windows\System32\Window x rdbisch@Rbisc150165: ~/CS498 x rdbisch@Rbisc150165: ~/CS498 x + -
timestep_time,vehicle_CO,vehicle_CO2,vehicle_HC,vehicle_NOx,vehicle_PMX,vehicle_angle,vehicle_eclclass,vehicle_electricity,vehicle_fue
l,vehicle_id,vehicle_lane,vehicle_noise,vehicle_pos,vehicle_route,vehicle_speed,vehicle_type,vehicle_waiting,vehicle_x,vehicle_y
0.00,15.20,7380.56,0.00,84.89,2.21,50.28,HBEFA3/HDV,0.00,3.13,truck0,5329992#5_0,67.11,7.20,!truck0!var#1,0.00,truck_truck,0.00,1827
5.04,26987.78
0.00,0.00,2416.04,0.01,0.72,0.01,42.25,HBEFA3/PC_G_EU4,0.00,1.04,veh0,5330181#0_0,65.15,5.10,!veh0!var#1,14.72,veh_passenger,0.00,18
279.94,24533.12
1.00,17.92,9898.93,0.00,103.38,2.49,50.28,HBEFA3/HDV,0.00,4.20,truck0,5329992#5_0,73.20,8.21,!truck0!var#1,1.01,truck_truck,0.00,182
75.82,26988.43
1.00,0.00,0.00,0.00,0.00,0.00,42.25,HBEFA3/PC_G_EU4,0.00,0.00,veh0,5330181#0_0,62.72,18.85,!veh0!var#1,13.75,veh_passenger,0.00,1828
9.19,24543.30
1.00,164.78,2624.72,0.81,1.20,0.07,357.00,HBEFA3/PC_G_EU4,0.00,1.13,veh1,-5338968#2_0,55.94,5.10,!veh1!var#1,0.00,veh_passenger,0.00
,29252.01,24424.16
1.00,164.78,2624.72,0.81,1.20,0.07,271.07,HBEFA3/PC_G_EU4,0.00,1.13,veh2,5337487#3_0,55.94,5.10,!veh2!var#1,0.00,veh_passenger,0.00,
23726.22,25284.67
2.00,179.19,1228.61,0.64,0.31,0.17,271.69,HBEFA3/LDV_G_EU6,0.00,0.53,moto1,5335345#1_0,55.94,2.30,!moto1!var#1,0.00,moto_motorcycle,
0.00,22460.55,27366.47
2.00,20.38,12176.78,0.00,120.04,2.74,50.28,HBEFA3/HDV,0.00,5.16,truck0,5329992#5_0,73.41,10.21,!truck0!var#1,1.99,truck_truck,0.00,1
8277.35,26989.70
2.00,0.00,0.00,0.00,0.00,0.00,42.25,HBEFA3/PC_G_EU4,0.00,0.00,veh0,5330181#0_0,61.30,31.36,!veh0!var#1,12.51,veh_passenger,0.00,1829
7.59,24552.56
2.00,149.59,3770.59,0.77,1.68,0.08,356.91,HBEFA3/PC_G_EU4,0.00,1.62,veh1,-5338968#2_0,67.05,7.42,!veh1!var#1,2.32,veh_passenger,0.00
,29251.89,24426.48
2.00,148.12,2942.26,0.74,1.32,0.07,271.09,HBEFA3/PC_G_EU4,0.00,1.26,veh2,5337487#3_0,62.42,6.46,!veh2!var#1,1.36,veh_passenger,0.00,
23724.87,25284.69
2.00,164.78,2624.72,0.81,1.20,0.07,125.41,HBEFA3/PC_G_EU4,0.00,1.13,veh3,724636540#2_0,55.94,5.10,!veh3!var#1,0.00,veh_passenger,0.0
0,26221.37,26484.93
2.00,164.78,2624.72,0.81,1.20,0.07,359.85,HBEFA3/PC_G_EU4,0.00,1.13,veh4,-737358444#2_0,55.94,4.39,!veh4!var#1,0.00,veh_passenger,0.
00,27949.69,24299.88
2.00,164.78,2624.72,0.81,1.20,0.07,89.89,HBEFA3/PC_G_EU4,0.00,1.13,veh5,-5336220#0_0,55.94,5.10,!veh5!var#1,0.00,veh_passenger,0.00,
26908.91,23474.45
3.00,446.17,10029.63,1.95,2.06,0.67,271.69,HBEFA3/LDV_G_EU6,0.00,4.31,moto1,5335345#1_0,83.23,7.99,!moto1!var#1,5.69,moto_motorcycle,
0.00,22454.86,27366.64
3.00,179.19,1228.61,0.64,0.31,0.17,7.10,HBEFA3/LDV_G_EU6,0.00,0.53,moto2,-5341858#10_0,55.94,2.30,!moto2!var#1,0.00,moto_motorcycle,
0.00,26467.87,25514.99
3.00,20.17,5286.11,4.85,60.75,2.01,9.28,HBEFA3/Bus,0.00,2.25,pt_bus_1N:0_0,-5328209#8_0,67.11,12.10,pt_bus_1N:0_0,0.00,pt_bus,0.00,230
43.87,18921.08
3.00,25.86,17200.80,0.00,157.80,3.31,50.28,HBEFA3/HDV,0.00,7.29,truck0,5329992#5_0,75.67,13.48,!truck0!var#1,3.27,truck_truck,0.00,1
8279.87,26991.79
3.00,0.00,0.00,0.00,0.00,0.00,40.62,HBEFA3/PC_G_EU4,0.00,0.00,veh0,5330181#0_0,55.09,39.88,!veh0!var#1,8.52,veh_passenger,0.00,18303
.19,24558.98
3.00,143.93,5255.34,0.78,2.32,0.11,356.91,HBEFA3/PC_G_EU4,0.00,2.26,veh1,-5338968#2_0,68.69,12.21,!veh1!var#1,4.78,veh_passenger,0.0
0,29251.63,24431.26
3.00,134.43,3441.38,0.69,1.51,0.07,271.10,HBEFA3/PC_G_EU4,0.00,1.48,veh2,5337487#3_0,63.69,9.32,!veh2!var#1,2.86,veh_passenger,0.00,
23722.00,25284.75
3.00,148.29,2920.66,0.74,1.31,0.07,125.41,HBEFA3/PC_G_EU4,0.00,1.26,veh3,724636540#2_0,62.25,6.42,!veh3!var#1,1.32,veh_passenger,0.0
0,26222.44,26484.16
3.00,148.04,2955.05,0.74,1.32,0.07,359.71,HBEFA3/PC_G_EU4,0.00,1.27,veh4,-737358444#1_0,62.52,1.28,!veh4!var#1,1.38,veh_passenger,0.
00,27949.68,24301.16
3.00,148.69,3619.69,0.76,1.61,0.08,89.89,HBEFA3/PC_G_EU4,0.00,1.56,veh5,-5336220#0_0,66.38,7.28,!veh5!var#1,2.18,veh_passenger,0.00,
26911.09,23474.45
data/emission_comma.csv
```

Step 4:

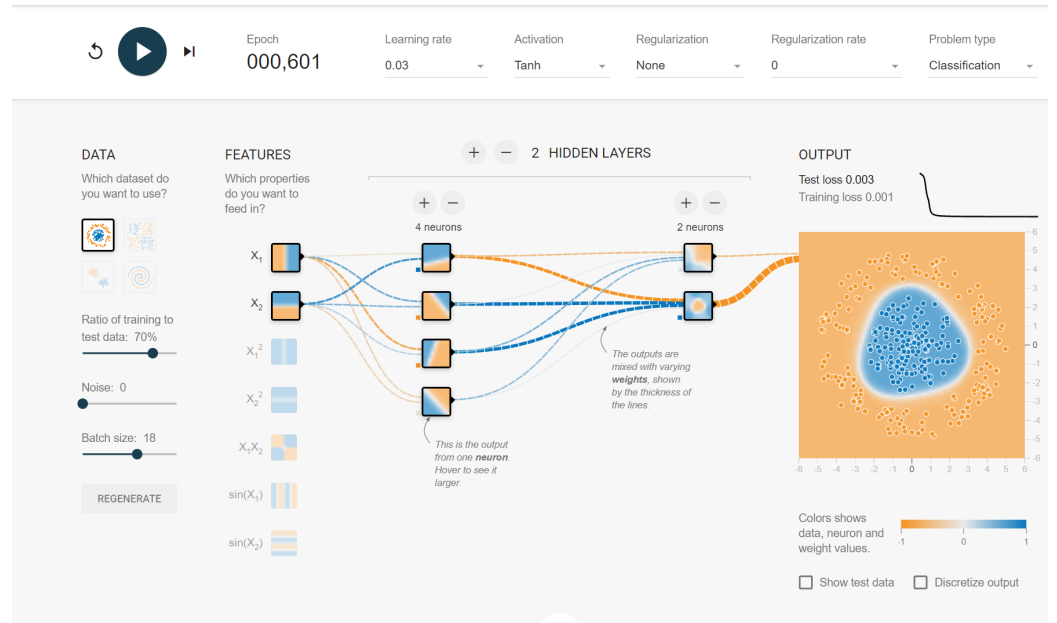
1. Try Relu, Sigmoid, Linear, and Tanh. Run for 600 Epochs. Note the output graphs and losses. **Q1: What do you observe? i.e. Do all activation functions give you a nice training result?**

No. The TanH gave the best classification after 600 epochs, though Sigmoid was catching up fast if I had let it go longer. As it stands @ 600 epochs, ReLU has the 2nd best loss, but is seemingly only capturing a hexagon shape, and linear doesn't seem complicated enough to work.

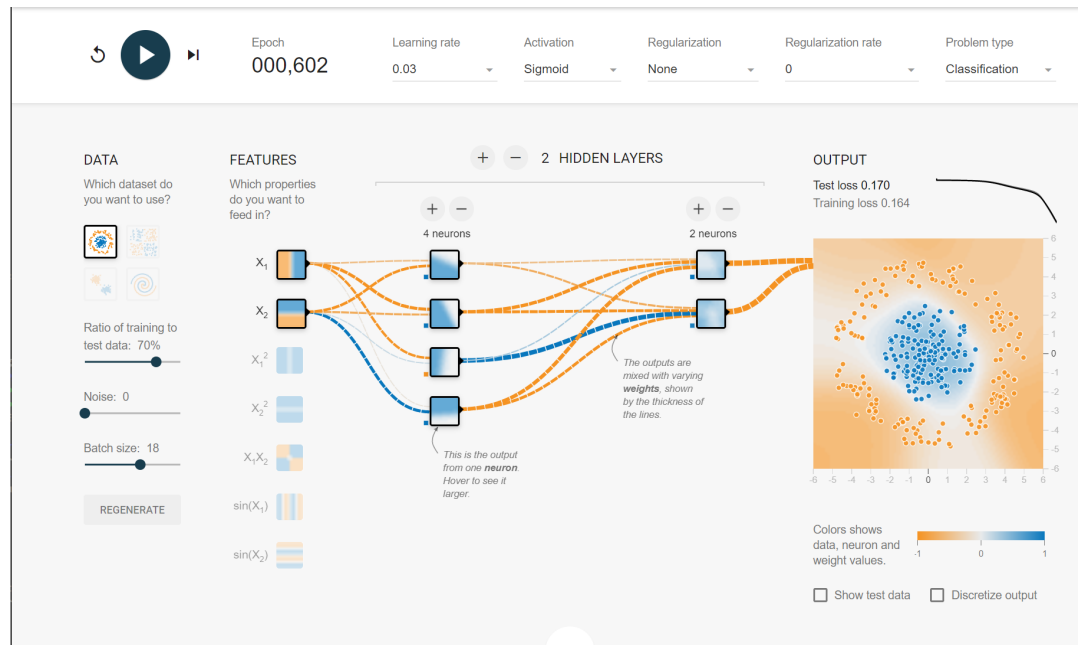
i. ReLU:



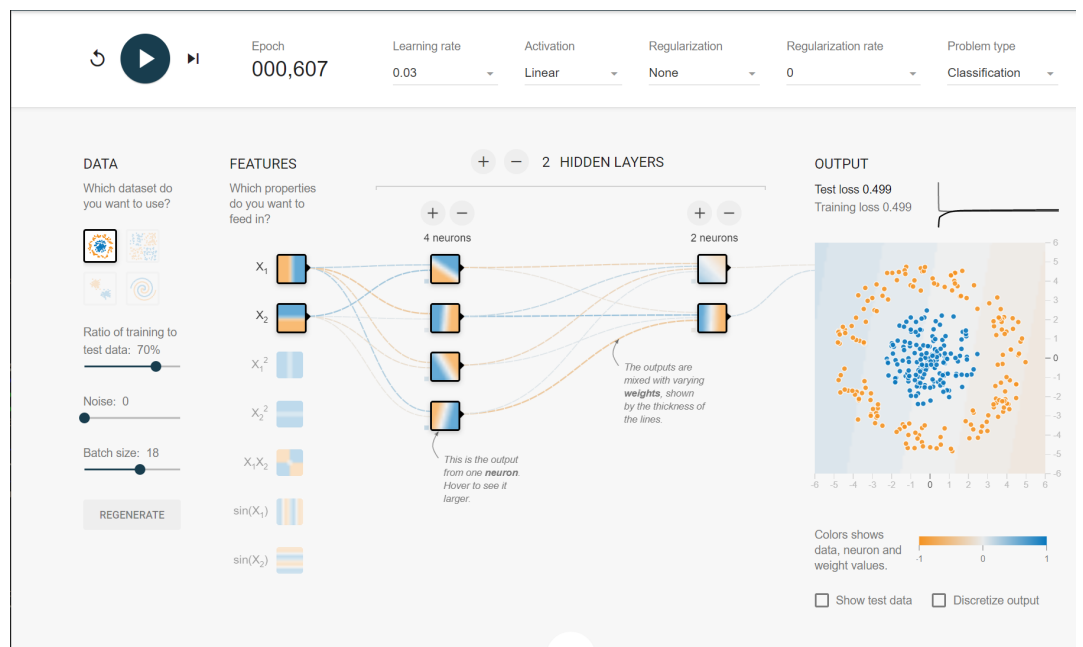
ii. Tanh:



iii. Sigmoid

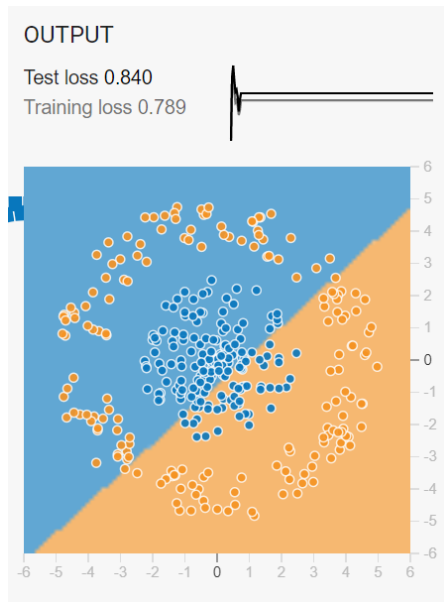


iv. Linear



2. Change the learning rate. Try a large learning rate like 10 and a small learning rate like 0.00001. **Q2: Look at the loss, epochs, etc. What do you observe?**

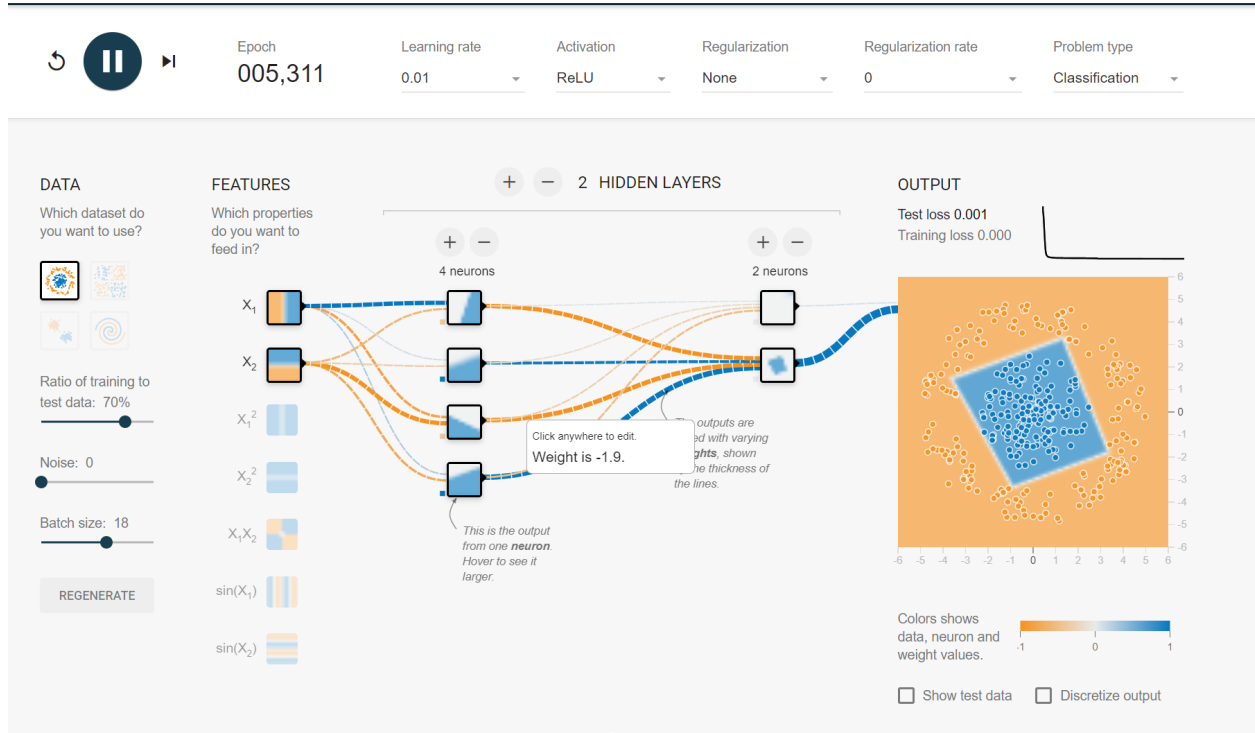
Large learning rates force an answer much too soon and the model never recovers from it. Our previous best, tanH, converges to something like



, whereas the smallest learning rates show signs of improvement but very very slow. This is really inefficient.

3. **Q3: What do you observe? i.e. Do they change over time? Can the weights be negative?** Notice that the thicker the dashed line (-----) is, the heavier the weight is.

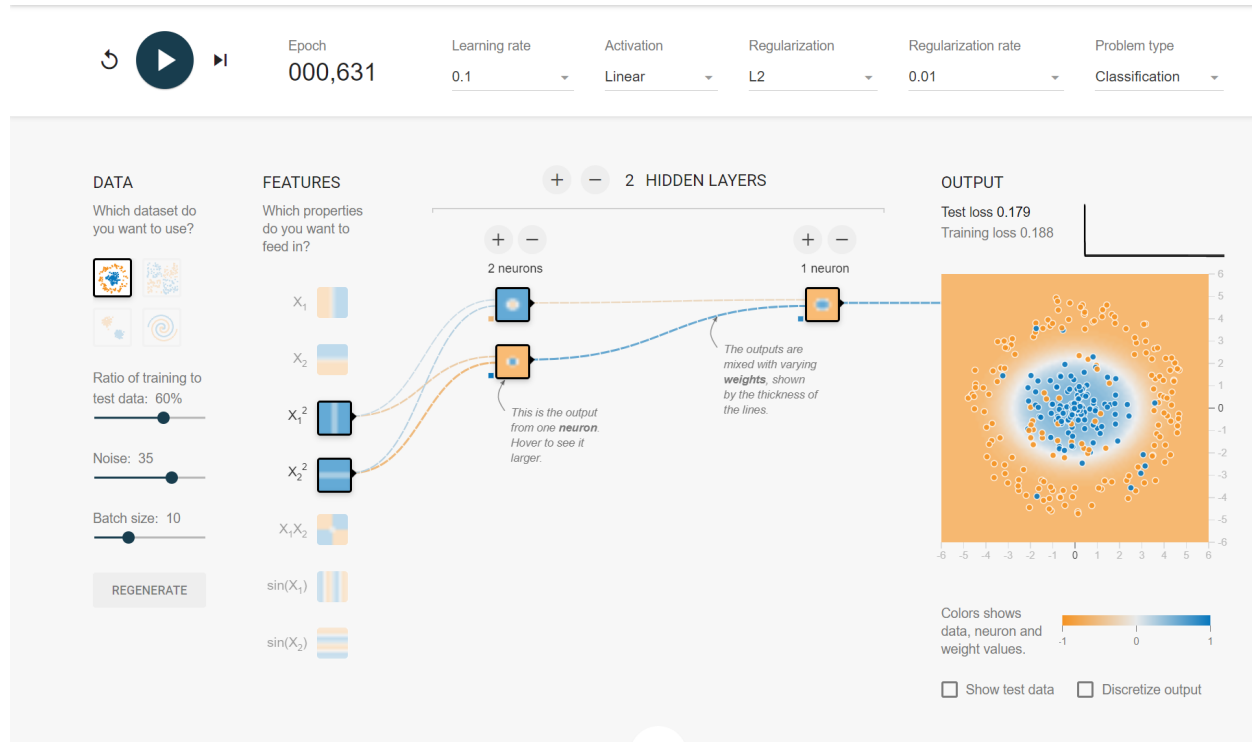
Yes, the weights change over time. This is how the network adjusts its guesses by changing the weights. They can be negative just as well as a positive. Here is a screenshot



4. **Q4: Look at the test and training loss. What do you observe?**

Compared to the earlier ReLU fit without regularization the losses are much higher, but we can see that this one is fitting the data better visually. In this particular case, the test loss is higher than the training loss.

7. Lastly, adjust anything and get the best result you can for this dataset. Take a screenshot.



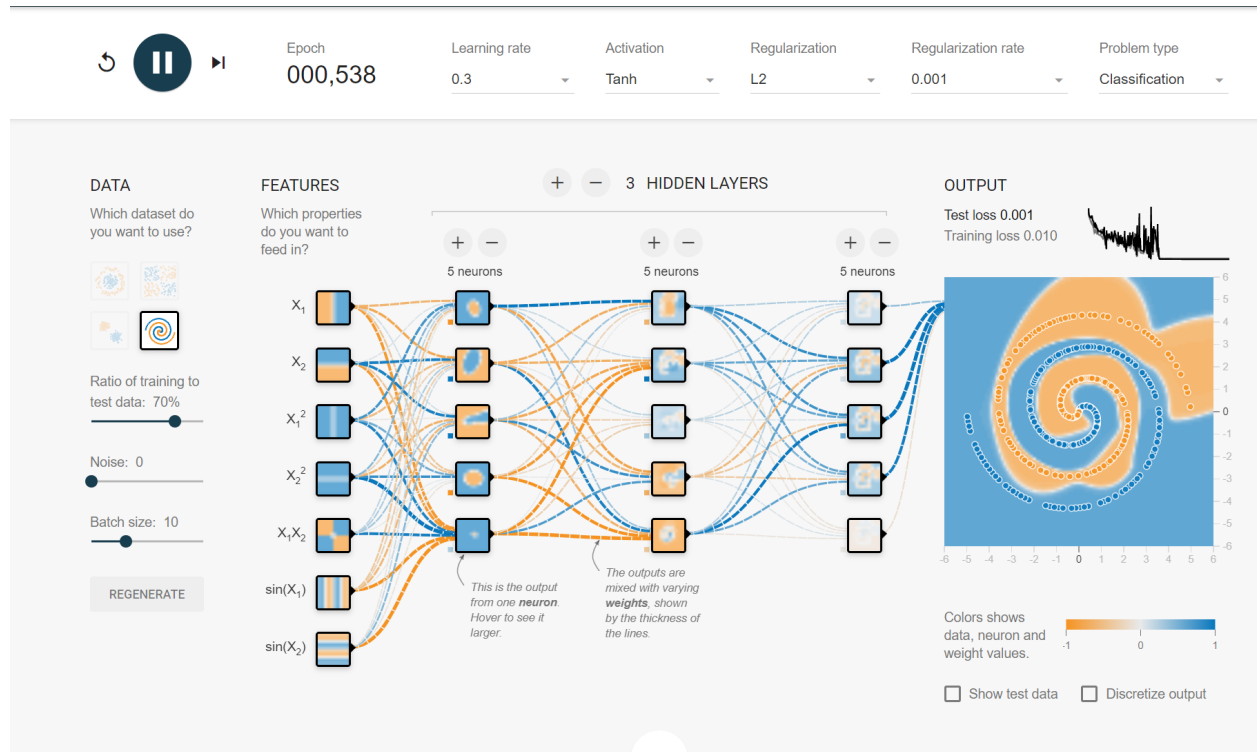
5. **Q5: Why are other features needed to get a better training result in this case?**

Other features are needed because the spiral cannot be defined as a function in just X_1 and X_2 mathematically (vertical line test!) regardless of how well the neural network works. So it needs extra features to be able to capture the complexity.

6. **Q6: What do you observe (with regards to training speed, loss, etc.)?**

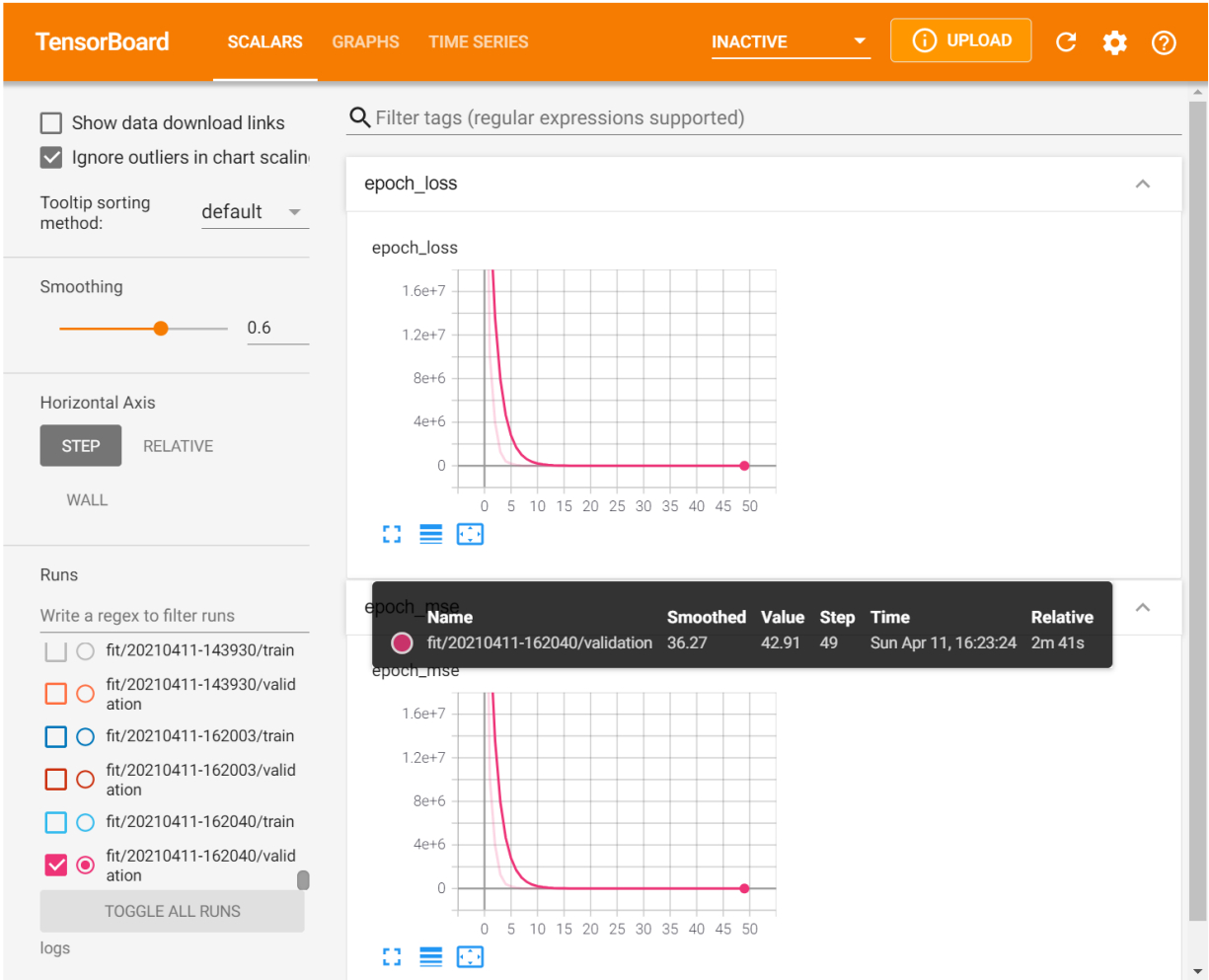
Training speed slows down because there are more parameters (weights) to adjust. Adding more layers or more nodes always makes the training loss lower because of a tendency to overfit, but the test error doesn't necessarily show the same behavior.

Lastly, adjust anything and get the best result you can for this dataset. Take a screenshot.



Step 5 - Machine Learning on Traffic Emission Data

Screen Shot of Small Dataset




```

In [235]: # Hyperparameters
learning_rate = 0.001 ### FILL IN A NUMBER
epochs = 50 ### FILL IN A NUMBER
batch_size = 125 ### FILL IN A NUMBER

# Label
label_name = "vehicle_CO2"
shuffle = True

#---Create a sequential model---#
model = tf.keras.models.Sequential([
    # Add the feature layer
    feature_layer,

    # First hidden layer with 20 nodes
    tf.keras.layers.Dense(units=20,
                           activation='relu',
                           kernel_regularizer=tf.keras.regularizers.l1(l=0.1),
                           name='Hidden1'),

    # First hidden layer with 20 nodes
    tf.keras.layers.Dense(units=20,
                           activation='relu',
                           kernel_regularizer=tf.keras.regularizers.l1(l=0.1),
                           name='Hidden3'),

    # Output layer
    tf.keras.layers.Dense(units=1,
                           activation='linear',
                           name='Output')

])

model.compile(optimizer=tf.keras.optimizers.Adam(lr=learning_rate),
              loss=tf.keras.losses.MeanSquaredError(),
              metrics=['mse'])

#---Train the Model---#
# Keras TensorBoard callback.
logdir = "logs/fit/" + datetime.now().strftime("%Y%m%d-%H%M%S")
tensorboard_callback = tf.keras.callbacks.TensorBoard(log_dir=logdir)

train_lbl = np.array(train_df_norm["vehicle_CO2"])
#train_df = train_df.drop(columns=["vehicle_CO2"])
# Split the datasets into features and label.
train_ft = {name:np.array(value) for name, value in train_df_norm.items()}
# train_lbl = np.array(train_ft.pop(label_name))

val_lbl = np.array(valid_df_norm["vehicle_CO2"])
#val_df = val_df.drop(columns=["vehicle_CO2"])
val_ft = {name:np.array(value) for name, value in valid_df_norm.items()}

# Keras TensorBoard callback.

```

Screen Shot of Big Dataset

See Notebook_Learning2.ipynb for results for the entire data set.

Part 2

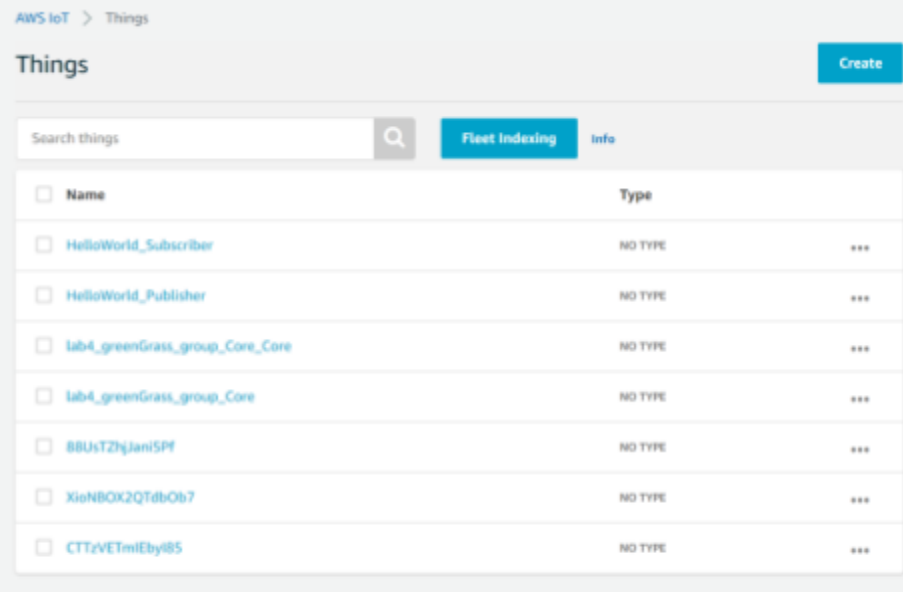
1. Compared with TCP-based client-server communication, what are the pros and cons for MQTT-based publish/subscribe.

MQTT offers a pretty handy abstraction for network communications. By abstracting communications away into “Topics”, and then further abstracting the communication to “Publish” and “Subscribe” patterns.

There are downsides though. Because it is so lightweight, it is not a very good solution for heavy payloads, like photos or videos. It also requires developers to expend more effort than they would otherwise under a fully managed protocol. Further, while the use of “topics” simplifies many things, the complexity of the system will dramatically increase over time as more and more topics are created. To the devices, this will be transparent, but to any central systems or human beings working on it, the complexity will become a bottleneck.

Section 1.3-1.4

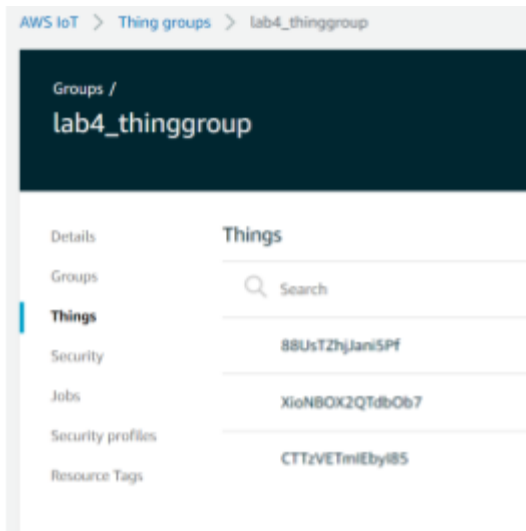
IoT Core Setup, Rules, Policies, etc.



The screenshot shows the AWS IoT Things console. At the top, there's a breadcrumb 'AWS IoT > Things' and a 'Create' button. Below that is a search bar labeled 'Search things' with a magnifying glass icon. To the right of the search bar are two buttons: 'Fleet Indexing' and 'Info'. The main content is a table with two columns: 'Name' and 'Type'. The table lists eight Things, each with a checkbox in the 'Name' column and 'NO TYPE' in the 'Type' column. Each row also has a three-dot menu icon on the right.

Name	Type
<input type="checkbox"/> HelloWorld_Subscriber	NO TYPE
<input type="checkbox"/> HelloWorld_Publisher	NO TYPE
<input type="checkbox"/> lab4_greenGrass_group_Core_Core	NO TYPE
<input type="checkbox"/> lab4_greenGrass_group_Core	NO TYPE
<input type="checkbox"/> 8BUstZhjani5Pf	NO TYPE
<input type="checkbox"/> XieNBOK2QTdbOb7	NO TYPE
<input type="checkbox"/> CTTzVETmiEbyl8S	NO TYPE

Screenshot of available Things in AWS IoT



Screenshot of IoT Thing Group I created to manage policies easier



Example policy created to allow IoT things to use MQTT

Section 1.5 Create Many Devices

I modified “createThing-Cert.py” in obvious ways to create “Things” en-masse, and make the certificates easier to manage. I also added code to automatically add the “Things” created here to the “Thing Group” mentioned above, to make management of policies easier.

```

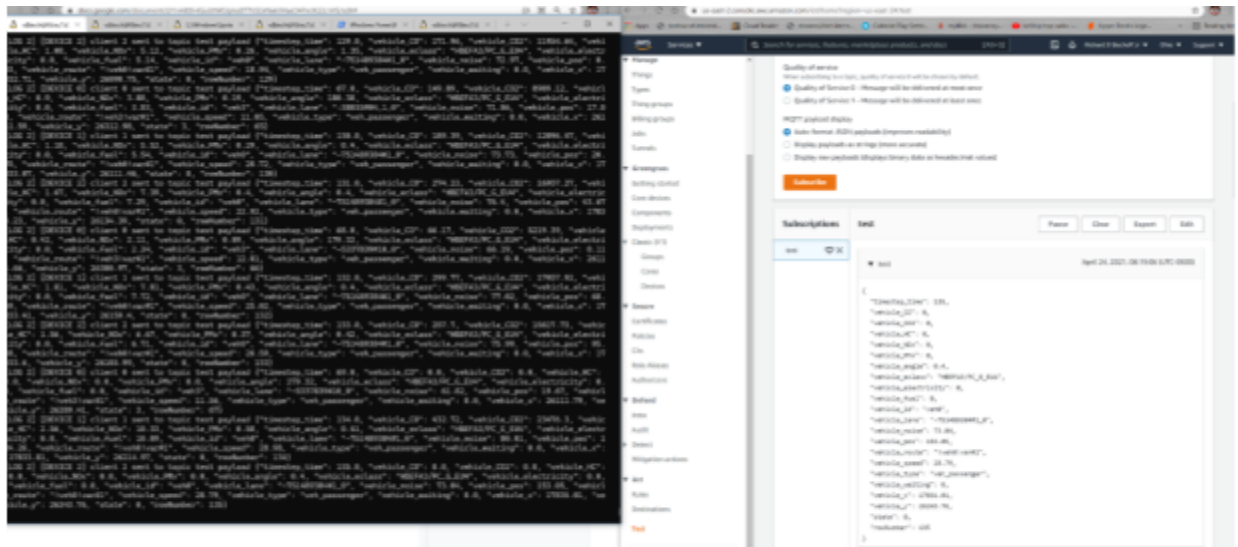
[LOG 2] [DEVICE 283] creating device cert data/certificates/device_283/device_283.certificate.pem key data/c
evice_283.private.pem
[LOG 2] [DEVICE 284] creating device cert data/certificates/device_284/device_284.certificate.pem key data/c
evice_284.private.pem
[LOG 2] [DEVICE 285] creating device cert data/certificates/device_285/device_285.certificate.pem key data/c
evice_285.private.pem
[LOG 2] [DEVICE 286] creating device cert data/certificates/device_286/device_286.certificate.pem key data/c
evice_286.private.pem
[LOG 2] [DEVICE 287] creating device cert data/certificates/device_287/device_287.certificate.pem key data/c
evice_287.private.pem
[LOG 2] [DEVICE 288] creating device cert data/certificates/device_288/device_288.certificate.pem key data/c
evice_288.private.pem
[LOG 2] [DEVICE 289] creating device cert data/certificates/device_289/device_289.certificate.pem key data/c
evice_289.private.pem
[LOG 2] [DEVICE 290] creating device cert data/certificates/device_290/device_290.certificate.pem key data/c
evice_290.private.pem
[LOG 2] [DEVICE 291] creating device cert data/certificates/device_291/device_291.certificate.pem key data/c
evice_291.private.pem
[LOG 2] [DEVICE 292] creating device cert data/certificates/device_292/device_292.certificate.pem key data/c
evice_292.private.pem
[LOG 2] [DEVICE 293] creating device cert data/certificates/device_293/device_293.certificate.pem key data/c
evice_293.private.pem
[LOG 2] [DEVICE 294] creating device cert data/certificates/device_294/device_294.certificate.pem key data/c
evice_294.private.pem
[LOG 2] [DEVICE 295] creating device cert data/certificates/device_295/device_295.certificate.pem key data/c
evice_295.private.pem
[LOG 2] [DEVICE 296] creating device cert data/certificates/device_296/device_296.certificate.pem key data/c
evice_296.private.pem
[LOG 2] [DEVICE 297] creating device cert data/certificates/device_297/device_297.certificate.pem key data/c
evice_297.private.pem
[LOG 2] [DEVICE 298] creating device cert data/certificates/device_298/device_298.certificate.pem key data/c
evice_298.private.pem
[LOG 2] [DEVICE 299] creating device cert data/certificates/device_299/device_299.certificate.pem key data/c
evice_299.private.pem
Users at state 1: [7, 56, 83, 97, 103, 105, 122, 124, 143, 144, 203, 226, 251, 282]
Users at state 2: [16, 19, 23, 51, 81, 141, 165, 169, 188, 195, 221, 265, 269, 280, 289]
Users at state 3: [0, 15, 17, 29, 46, 49, 55, 67, 116, 126, 150, 187, 199, 210]
Users at state 4: [4, 11, 12, 43, 45, 73, 82, 87, 93, 102, 104, 159, 163, 197, 202, 233, 260, 272, 273]
send now?
Waiting on input. S = Send. D = Disconnect...

```

Section 1.6 Use the Device Simulator

I modified “emulator_client.py” to reference the certificates created above. I added logging to make it easier to track topics and devices on the screen. I also added code so that the “Things” would emit a single row from the vehicle datasets per event. Then I modified the loop to run until there is no more rows to read, and then disconnect it from AWS..

This is the same data I will be analyzing in later parts of the lab.



Snapshot of the device simulator publishing data

Section 1.6 Greengrass

Note: In the repository, this code lives in the greengrass_module4/ sub-directory.

I am using my Raspberry Pi for my final project, so rather than risk that project, I opted to use an EC2 instance instead. The configuration of this was relatively straightforward, and had the GreenGrass publisher/subscriber tutorial running OK.

I then modified "basicDiscovery.py". I removed the payload option from the command line and instead changed it to a data argument. Then, I would send over the data row by row, similar to how emulator_client works.

```

2021-04-24 09:01:41,321 - AWSIoTPythonSDK.core.protocol.internal.workers - DEBUG - Dispatching [puback] event
2021-04-24 09:01:42,322 - AWSIoTPythonSDK.core.protocol.mqtt_core - INFO - Performing sync publish...
Published topic hello/world/pubsub: {"timestep_time": 45.0, "vehicle_CO": 0.0, "vehicle_CO2": 0.0, "vehicle_HC": 0.0, "vehicle_NOx": 0.0, "vehicle_PMx": 0.0, "vehicle_angle": 313.08, "vehicle_eclass": "HBEFA3/PC_G_EU4", "vehicle_electricity": 0.0, "vehicle_fuel": 0.0, "vehicle_id": "veh0", "vehicle_lane": "5328198#2_0", "vehicle_noise": 47.14, "vehicle_pos": 136.81, "vehicle_route": "!veh0!var#1", "vehicle_speed": 4.39, "vehicle_type": "veh_passenger", "vehicle_waiting": 0.0, "vehicle_x": 18142.51, "vehicle_y": 24945.88, "sequence": 45}
2021-04-24 09:01:42,323 - AWSIoTPythonSDK.core.protocol.internal.workers - DEBUG - Produced [puback] event

2021-04-24 09:01:42,323 - AWSIoTPythonSDK.core.protocol.internal.workers - DEBUG - Dispatching [puback] event
2021-04-24 09:01:43,325 - AWSIoTPythonSDK.core.protocol.mqtt_core - INFO - Performing sync publish...
Published topic hello/world/pubsub: {"timestep_time": 46.0, "vehicle_CO": 113.73, "vehicle_CO2": 4525.78, "vehicle_HC": 0.62, "vehicle_NOx": 1.94, "vehicle_PMx": 0.09, "vehicle_angle": 300.31, "vehicle_eclass": "HBEFA3/PC_G_EU4", "vehicle_electricity": 0.0, "vehicle_fuel": 1.95, "vehicle_id": "veh0", "vehicle_lane": ":37980183_4_0", "vehicle_noise": 65.24, "vehicle_pos": 5.18, "vehicle_route": "!veh0!var#1", "vehicle_speed": 5.94, "vehicle_type": "veh_passenger", "vehicle_waiting": 0.0, "vehicle_x": 18137.55, "vehicle_y": 24948.99, "sequence": 46}
2021-04-24 09:01:43,327 - AWSIoTPythonSDK.core.protocol.internal.workers - DEBUG - Produced [puback] event

2021-04-24 09:01:43,327 - AWSIoTPythonSDK.core.protocol.internal.workers - DEBUG - Dispatching [puback] event
2021-04-24 09:01:44,338 - AWSIoTPythonSDK.core.protocol.mqtt_core - INFO - Performing sync publish...
Published topic hello/world/pubsub: {"timestep_time": 47.0, "vehicle_CO": 113.56, "vehicle_CO2": 5519.14, "vehicle_HC": 0.65, "vehicle_NOx": 2.36, "vehicle_PMx": 0.11, "vehicle_angle": 247.02, "vehicle_eclass": "HBEFA3/PC_G_EU4", "vehicle_electricity": 0.0, "vehicle_fuel": 2.37, "vehicle_id": "veh0", "vehicle_lane": ":37980183_4_0", "vehicle_noise": 66.79, "vehicle_pos": 12.83, "vehicle_route": "!veh0!var#1", "vehicle_speed": 7.65, "vehicle_type": "veh_passenger", "vehicle_waiting": 0.0, "vehicle_x": 18130.47, "vehicle_y": 24947.44, "sequence": 47}
2021-04-24 09:01:44,332 - AWSIoTPythonSDK.core.protocol.internal.workers - DEBUG - Produced [puback] event

```

Greengrass, publish.sh

```

2021-04-24 09:01:27,291 - AWSIoTPythonSDK.core.protocol.internal.workers - DEBUG - Produced [message] event
2021-04-24 09:01:27,291 - AWSIoTPythonSDK.core.protocol.internal.workers - DEBUG - Dispatching [message] event
2021-04-24 09:01:27,292 - AWSIoTPythonSDK.core.protocol.internal.clients - DEBUG - Invoking custom event callback...
Received message on topic hello/world/pubsub: b'{"timestep_time": 30.0, "vehicle_CO": 0.0, "vehicle_CO2": 0.0, "vehicle_HC": 0.0, "vehicle_NOx": 0.0, "vehicle_PMx": 0.0, "vehicle_angle": 359.31, "vehicle_eclass": "HBEFA3/PC_G_EU4", "vehicle_electricity": 0.0, "vehicle_fuel": 0.0, "vehicle_id": "veh0", "vehicle_lane": "744926153#6_0", "vehicle_noise": 62.66, "vehicle_pos": 56.88, "vehicle_route": "!veh0!var#1", "vehicle_speed": 13.68, "vehicle_type": "veh_passenger", "vehicle_waiting": 0.0, "vehicle_x": 18277.04, "vehicle_y": 24882.34, "sequence": 30}'
2021-04-24 09:01:28,299 - AWSIoTPythonSDK.core.protocol.internal.workers - DEBUG - Produced [message] event
2021-04-24 09:01:28,299 - AWSIoTPythonSDK.core.protocol.internal.workers - DEBUG - Dispatching [message] event
2021-04-24 09:01:28,300 - AWSIoTPythonSDK.core.protocol.internal.clients - DEBUG - Invoking custom event callback...
Received message on topic hello/world/pubsub: b'{"timestep_time": 31.0, "vehicle_CO": 0.0, "vehicle_CO2": 0.0, "vehicle_HC": 0.0, "vehicle_NOx": 0.0, "vehicle_PMx": 0.0, "vehicle_angle": 358.58, "vehicle_eclass": "HBEFA3/PC_G_EU4", "vehicle_electricity": 0.0, "vehicle_fuel": 0.0, "vehicle_id": "veh0", "vehicle_lane": "744926153#6_0", "vehicle_noise": 59.15, "vehicle_pos": 68.18, "vehicle_route": "!veh0!var#1", "vehicle_speed": 11.3, "vehicle_type": "veh_passenger", "vehicle_waiting": 0.0, "vehicle_x": 18276.76, "vehicle_y": 24893.63, "sequence": 31}'
2021-04-24 09:01:29,301 - AWSIoTPythonSDK.core.protocol.internal.workers - DEBUG - Produced [message] event
2021-04-24 09:01:29,301 - AWSIoTPythonSDK.core.protocol.internal.workers - DEBUG - Dispatching [message] event
2021-04-24 09:01:29,302 - AWSIoTPythonSDK.core.protocol.internal.clients - DEBUG - Invoking custom event callback...
Received message on topic hello/world/pubsub: b'{"timestep_time": 32.0, "vehicle_CO": 0.0, "vehicle_CO2": 0.0, "vehicle_HC": 0.0, "vehicle_NOx": 0.0, "vehicle_PMx": 0.0, "vehicle_angle": 353.61, "vehicle_eclass": "HBEFA3/PC_G_EU4", "vehicle_electricity": 0.0, "vehicle_fuel": 0.0, "vehicle_id": "veh0", "vehicle_lane": ":37980090_6_0", "vehicle_noise": 52.2, "vehicle_pos": 2.94, "vehicle_route": "!veh0!var#1", "vehicle_speed": 6.8, "vehicle_type": "veh_passenger", "vehicle_waiting": 0.0, "vehicle_x": 18276.16, "vehicle_y": 24980.39, "sequence": 32}'

```

Greengrass, sub.sh

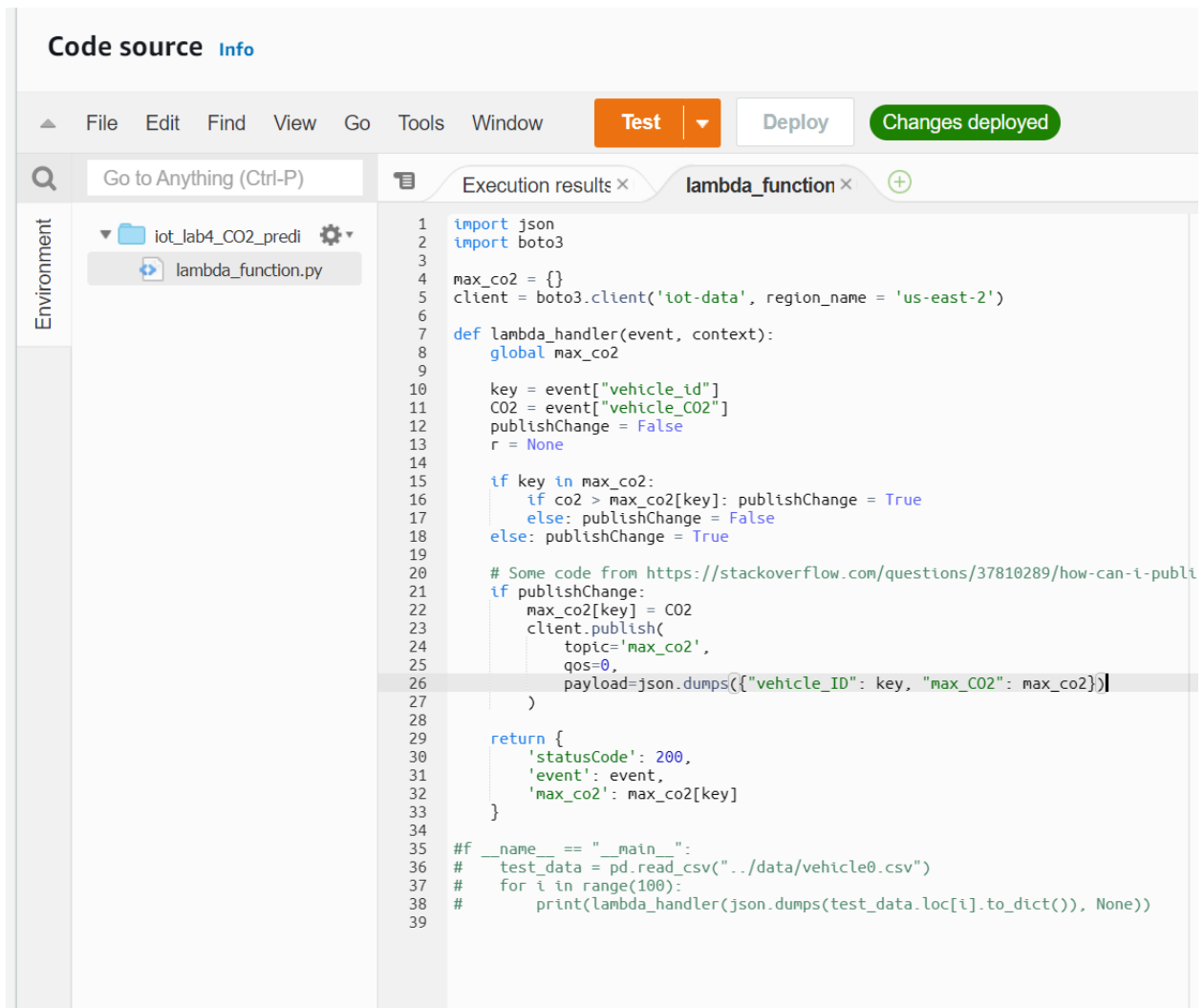
Section 2 Data Inference

Section 2.1 Lambda

Due to problems with python2 compatibility. I threw away the provided `deploy_package` lambda and started my own from scratch.

I first misunderstood the lab and actually created a python script that took the model from part 1 and scored it. I was, however, unfortunately unable to get that to work in AWS Lambdas because of the environment restrictions.

So instead I simply implemented a max-CO2 filter as instructed.



The screenshot shows a code editor interface with a menu bar (File, Edit, Find, View, Go, Tools, Window) and buttons for 'Test', 'Deploy', and 'Changes deployed'. The left sidebar shows the 'Environment' with a folder 'iot_lab4_CO2_predi' and a file 'lambda_function.py'. The main editor area displays the following Python code:

```
1 import json
2 import boto3
3
4 max_co2 = {}
5 client = boto3.client('iot-data', region_name = 'us-east-2')
6
7 def lambda_handler(event, context):
8     global max_co2
9
10    key = event["vehicle_id"]
11    CO2 = event["vehicle_CO2"]
12    publishChange = False
13    r = None
14
15    if key in max_co2:
16        if CO2 > max_co2[key]: publishChange = True
17        else: publishChange = False
18    else: publishChange = True
19
20    # Some code from https://stackoverflow.com/questions/37810289/how-can-i-publish-to-iot-core
21    if publishChange:
22        max_co2[key] = CO2
23        client.publish(
24            topic='max_co2',
25            qos=0,
26            payload=json.dumps({"vehicle_ID": key, "max_CO2": max_co2[key]})
27        )
28
29    return {
30        'statusCode': 200,
31        'event': event,
32        'max_co2': max_co2[key]
33    }
34
35 # if __name__ == "__main__":
36 #     test_data = pd.read_csv("../data/vehicle0.csv")
37 #     for i in range(100):
38 #         print(lambda_handler(json.dumps(test_data.loc[i].to_dict()), None))
39
```

Here is a screenshot of it working using the messages published from emulator_client.py.

Subscriptions

test

max_co2

max_co2

Pause

Clear

Export

Edit

▼ max_co2

April 24, 2021, 09:10:49 (UTC-0500)

```
{
  "vehicle_ID": "veh3",
  "max_CO2": {
    "veh0": 9975.05,
    "veh3": 4851.03
  }
}
```

▼ max_co2

April 24, 2021, 09:10:47 (UTC-0500)

```
{
  "vehicle_ID": "veh0",
  "max_CO2": {
    "veh3": 8761.14,
    "veh0": 8073.01
  }
}
```

Section 3 - IoT Analytics

I setup the channels, pipelines, data stores, and datasets as instructed.

CHANNEL

iot_lab4_analytics_channel

● Active

Actions ▾

Channel ARN

A channel Amazon Resource Name (ARN) uniquely identifies this channel.

```
arn:aws:iotanalytics:us-east-2:971920375555:channel/iot_lab4_analytics_channel
```

Creation date

Apr 18, 2021 12:38:15 PM -0500

Last updated date

Apr 18, 2021 12:38:15 PM -0500

Last message arrival time

Apr 24, 2021 9:11:07 AM -0500

Channel store

Service-managed store

Channel data retention period

[Edit](#)

Your channel data will be retained indefinitely.

Channel size

0.000013602 GB as of Apr 22, 2021 6:13:00 PM -0500.

Tags

[Edit](#)

No tags

Monitoring

Add to dashboard

1h

3h

12h

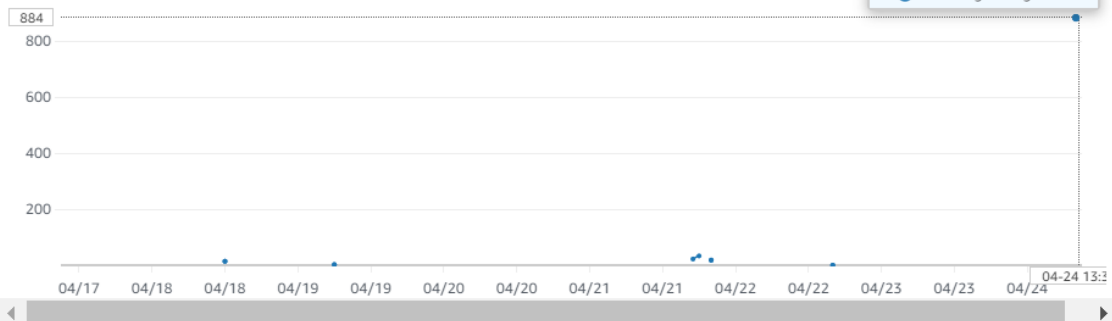
1d

3d

1w



IncomingMessages



iot_lab4_analytics_pipeline

Actions ▾

Overview

Details

Channel inputs

Edit

Name	Type
iot_lab4_analytics_channel	Channel

Activities

Edit

Name	Type
------	------

Data store outputs

Edit

Name	Type
iot_lab4_analytics_datastore	Data store

Tags

Edit

No tags

Monitoring

Add to dashboard

1h

3h

12h

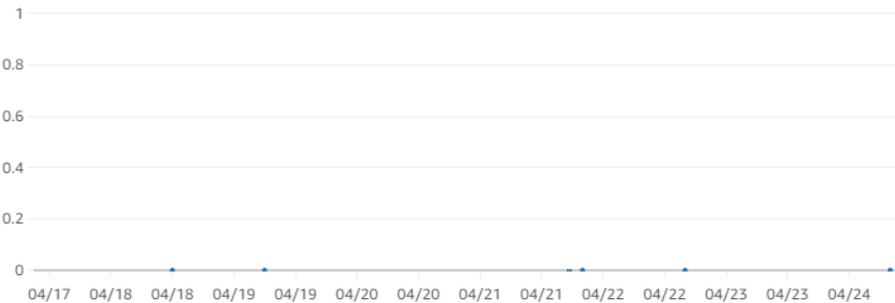
1d

3d

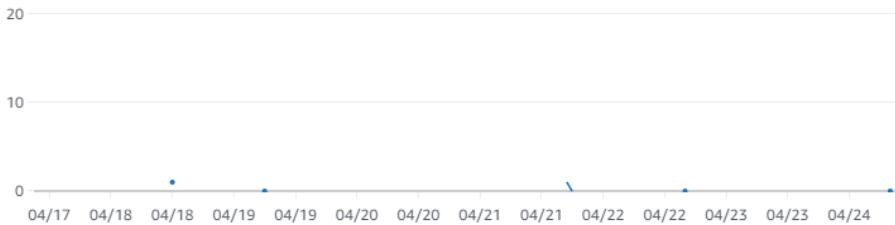
1w



ActivityExecutionError-DatastoreActivity-16



PipelineConcurrentExecutionCount



DATA SET

iot_lab4_analytics_dataset

SUCCEEDED

Actions ▾

Details

Content

Data set ARN

A data set Amazon Resource Name (ARN) uniquely identifies this data set.

arn:aws:iotanalytics:us-east-2:971920375555:dataset/iot_lab4_analytics_dataset

Details

SQL query

Edit

SELECT * FROM iot_lab4_analytics_datastore

Delta window

Edit

Delta window has not been set yet.

Result preview

thingid	temperature	humidity	datetime	timestep_time
				0.0

The default schema is pretty terrible because it is using all of the test data I sent through during development to create it.

I did not spend a ton of time creating nifty visualizations in Jupyter, but I did play around with the dataset and used some visualizations to help debug data flows. The print out is available in the zip-archive and github as "iot_lab4_analytics_notebook2.pdf".



When loading data from IoT Analytics datasets, the client should be initialized first:

Now we can get the data location (URL) for the given dataset and start working with the data (In order to need to perform `get_dataset_content`, you need to grant iot analytics corresponding IAM permission):

[illegible]

```
In [15]: import pandas as pd
```

```
In [16]: dataset_url = client.get_dataset_content(datasetName = dataset)['entries'][0]['dataURI']  
data = pd.read_csv(dataset_url)
```

```
In [17]: data
```

Out[17]:

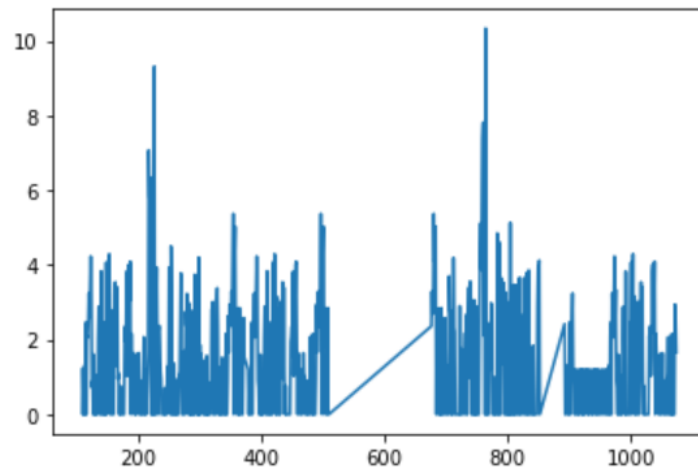
_time	vehicle_co	vehicle_co2	vehicle_hc	vehicle_nox	vehicle_pmx	...	vehicle_x	vehicle_y	notify_topic_arn	message	device_id	state	data	rownumber
NaN	NaN	NaN	NaN	NaN	NaN	...	NaN	NaN	NaN	Hello from AWS IoT console	NaN	NaN	NaN	NaN
NaN	NaN	2416.04	NaN	NaN	NaN	...	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN
NaN	NaN	999.00	NaN	NaN	NaN	...	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN
NaN	NaN	999.00	NaN	NaN	NaN	...	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN
0.0	0.00	NaN	NaN	NaN	NaN	...	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN
...
225.0	0.00	0.00	0.0	0.00	0.00	...	18379.41	27793.25	NaN	NaN	NaN	0.0	NaN	225.0
224.0	0.00	0.00	0.0	0.00	0.00	...	18382.60	27788.48	NaN	NaN	NaN	0.0	NaN	224.0
115.0	0.00	0.00	0.0	0.00	0.00	...	26409.19	26013.59	NaN	NaN	NaN	3.0	NaN	113.0
227.0	0.00	0.00	0.0	0.00	0.00	...	18380.31	27803.61	NaN	NaN	NaN	0.0	NaN	227.0
226.0	74.02	2700.56	0.4	1.07	0.04	...	18377.08	27798.75	NaN	NaN	NaN	0.0	NaN	226.0

```
In [18]: data2 = data[data["row"] >= 0]
```

```
In [27]: data3 = data2[["vehicle_nox", "vehicle_co2", "vehicle_x"]]
```

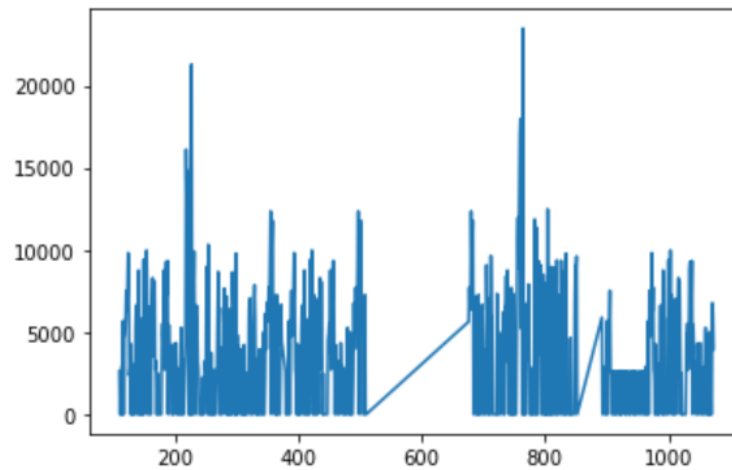
```
In [28]: data3["vehicle_nox"].plot()
```

```
Out[28]: <AxesSubplot:>
```



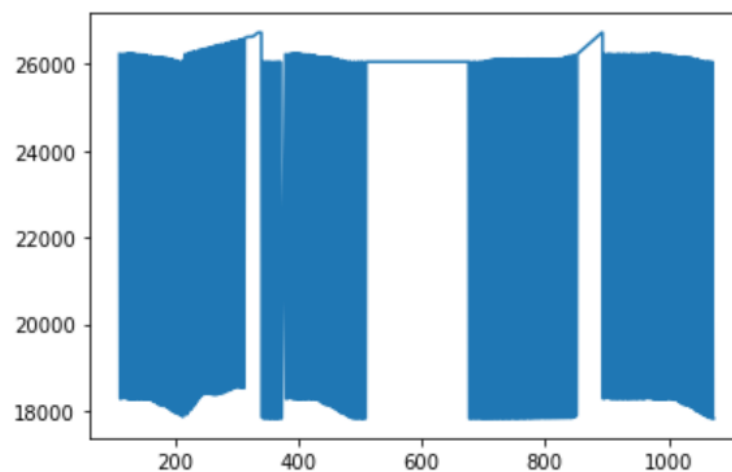
```
In [29]: data3["vehicle_co2"].plot()
```

```
Out[29]: <AxesSubplot:>
```



```
In [31]: data3["vehicle_x"].plot()
```

```
Out[31]: <AxesSubplot:>
```



Section 4 - AWS IoT Device Defender

Successfully created an audit

AWS IoT > Device Defender > Audit > Audit Results

Audit results (1)

Name	Date	Status	Summary
On-demand	April 24, 2021, 09:27:01 (UTC-0500)	<div>In progress</div> <div>Cancel</div>	14 of 14 in progress

Uh-oh, I failed some audit items!

AWS IoT > Device Defender > Audit > Audit Results > Audit Report

Audit Report

On-demand - April 24, 2021, 09:27:01 (UTC-0500)

Audit findings

Audit task ID

a016717f8445f9752c735bd55e0cd844

Started at

April 24, 2021, 09:27:01 (UTC-0500)

Non-compliant checks (2 of 14)

Check name	Severity	Non-compliant resources	% Resources	Mitigation
IoT policies overly permissive	Critical	4	100%	IoT policies overly permissive ⓘ
Logging disabled	Low	1	100%	Logging disabled ⓘ

The failed audit on IoT policies looks interesting. Taking a deeper look:

4 of 4 policies non-compliant

Mitigation

Create and attach policies with the minimum permissions required to do required tasks. You can use policy variables to help achieve this.

Non-compliant policy (4)

Suppress Finding

Start mitigation actions

< 1 >

	Finding	Reason	Version	Policy
<input type="checkbox"/>	2593de7af129414fa67c6aeb34362e48	Policy allows broad access to IoT data plane actions: [iot:Connect].	8	lab4_policy
<input type="checkbox"/>	f056501fad426daa5e92f4cf5fbb5bc6	Policy allows broad access to IoT data plane actions: [iot:Subscribe, iot:Connect, iot:GetThingShadow, iot:DeleteThingShadow, iot:UpdateThingShadow, iot:Publish].	1	lab4_greenGrass_group_Core_Core-policy
<input type="checkbox"/>	2196ea43687678d95288ba97fca99967	Policy allows broad access to IoT data plane actions: [iot:Subscribe, iot:Connect, iot:Publish].	1	HelloWorld_Subscriber-policy
<input type="checkbox"/>	5bca7600e408c5cf8f559c128eaabc7b	Policy allows broad access to IoT data plane actions: [iot:Subscribe, iot:Connect, iot:Publish].	1	HelloWorld_Publisher-policy

The last 3 were Amazon suggested policies, so... but the first line item is the one I made for the earlier part of the lab. Better fix that. I modified the policy to allow ONLY my Things, and then restricted it to the “test” topic I used throughout. Wa-la, that passes the audit!

Audit Findings

IoT policies overly permissive

3 of 4 policies non-compliant

Mitigation

Create and attach policies with the minimum permissions required to do required tasks. You can use

Non-compliant policy (3)

<input type="checkbox"/>	Finding	Reason
<input type="checkbox"/>	050d2db87192c85b3d7d1b1721a7b658	Policy allows broad access to IoT data plane [iot:Publish].
<input type="checkbox"/>	66bb92c6dc4f9184a297e39306db6a7b	Policy allows broad access to IoT data plane
<input type="checkbox"/>	7f231885ef1cd8eeb1c39bc31267cda1	Policy allows broad access to IoT data plane

Appendix

This is a list of the things that went wrong configuring Part 2!

Part 2

Difficulties So far:

1. Github code was python 2 not python3
2. Configuration difficulties....what was my region!?
3. Can't figure out how to run the emulator.... What is my endpoint
4. Ok I found the endpoint but it's timing out what the heck
5. Turns out my policies are wrong....

6. OK Got policy fixed.... But I noticed there doesn't seem to be a policy attached to my things, which is weird... I added everything to a new thing group and attached a policy to that. Let's see if that worked?
7. Noooooooooooooope. My policy wasn't correct, still being blocked.. I know this because I found a "non-policy" policy that made it work! Ugh.
8. Now I've discovered that the emulator code is not python3 compat either..fixing....ack that wasn't it. I actually have to fill in the publish call with something useful.
9. Ok that's all set. Things are connecting and the code works, but not seeing any messages coming through on AWS console
10. Alright! Success. This worked by changing the policy to all topics, not just "test". But now I have no clarity why "test" didn't work...
11. Spent a few hours playing around with pub/sub
12. Creating an EC2 instance to test Green Grass.....how do I connect!? An hour later....
13. Greengrass installed and running on EC2!
14. OK many hours passed....Finally got greengrass to work I think. The daemon was publishing its local network address, as well as the amazon private IP address, and so on. I had to manually go change it in the IoT console to the public IP and it worked.
15. Trying to understand the "Deploy_service" lambda instructions. All we're provided with is a scikit learn model with no model specifications....
16. Skipping scikit learn lambda problem until TA gets back to me...so working on AWS IoT analytics instead. Had to go through instructions twice to get it to work. Now I am in a Sagemaker notebook with a giant presigned URL
17. OK Got sagemaker to work now back to data. I spent a few hours writing tensorflow scoring code to code my model from part #1...only to find out that you can't do that with amazon lambdas, but instead you have to use ECR or another service!
18. Someone on slack posted that we dont need to score a model at all (#15).... Ugh! Rewrote lambda just to aggregate max co2.
19. Have a new lambda written, but getting forbidden errors. Have to figure out how to allow lambda to publish to MQTT