**Columbia University in the City of New York**

AI and OR at Scale on the Cloud

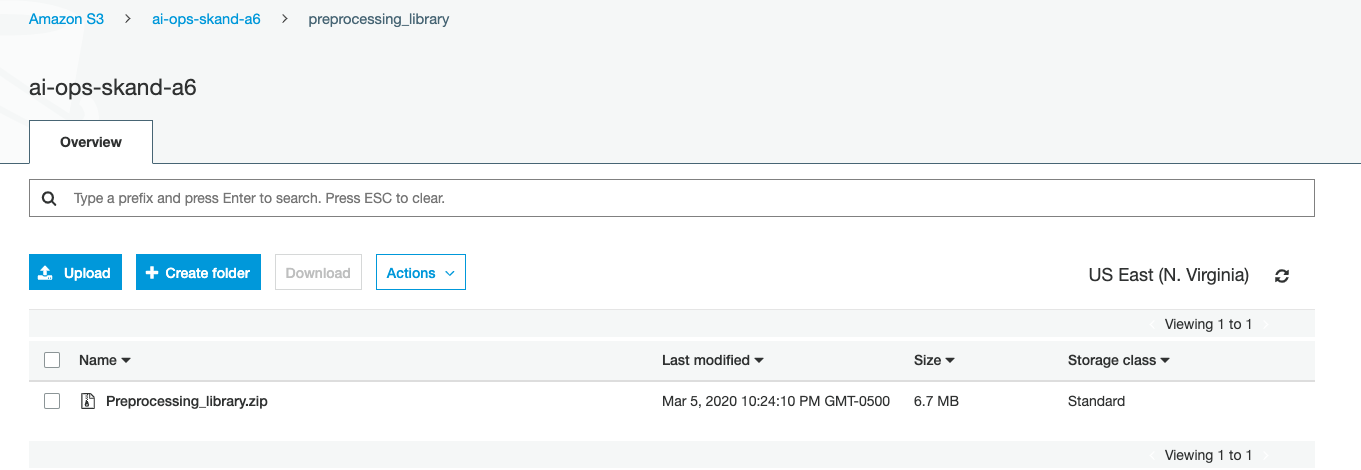
Assignment 6

Group 2: Harsh Mehta (hsm2148) Anunay Sanganal (avs2160) Skand Upmanyu (su2236)

**Step 1:** **Creating a preprocessing library which has the following features:**

* clean text
  + remove stopwords
  + removes twitter handles
  + remove URLs
* tokenize text
* create padding
* converts tokens to indices in the dictionary

This preprocessing library is then uploaded to S3:

****

**Step 2:** **Shuffling the data splitting into 3 parts: Train, Dev and Eval.**

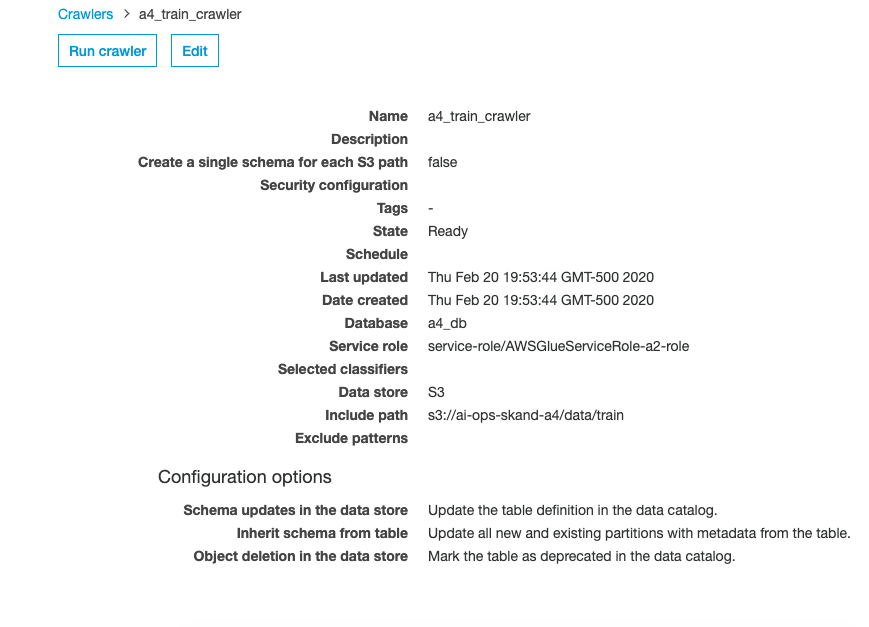
Train test split of data



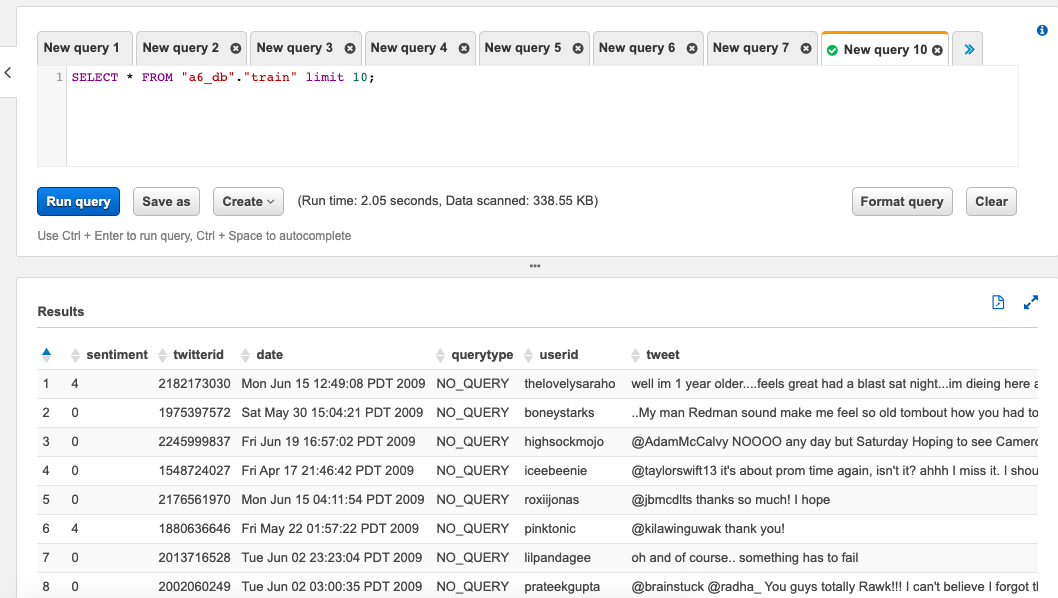
**Step 3: Creating a crawler job to create tables for the three datasets**

TRAIN:

Crawler for Train dataset

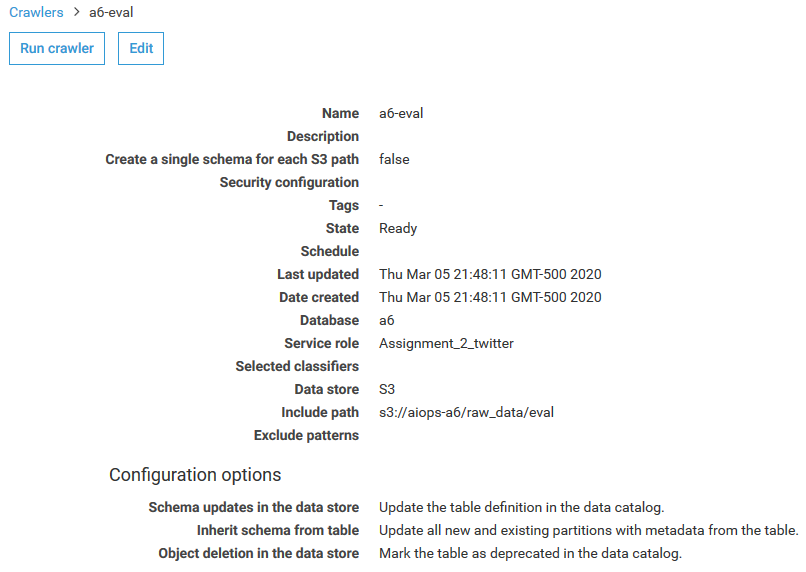


Athena output for train dataset

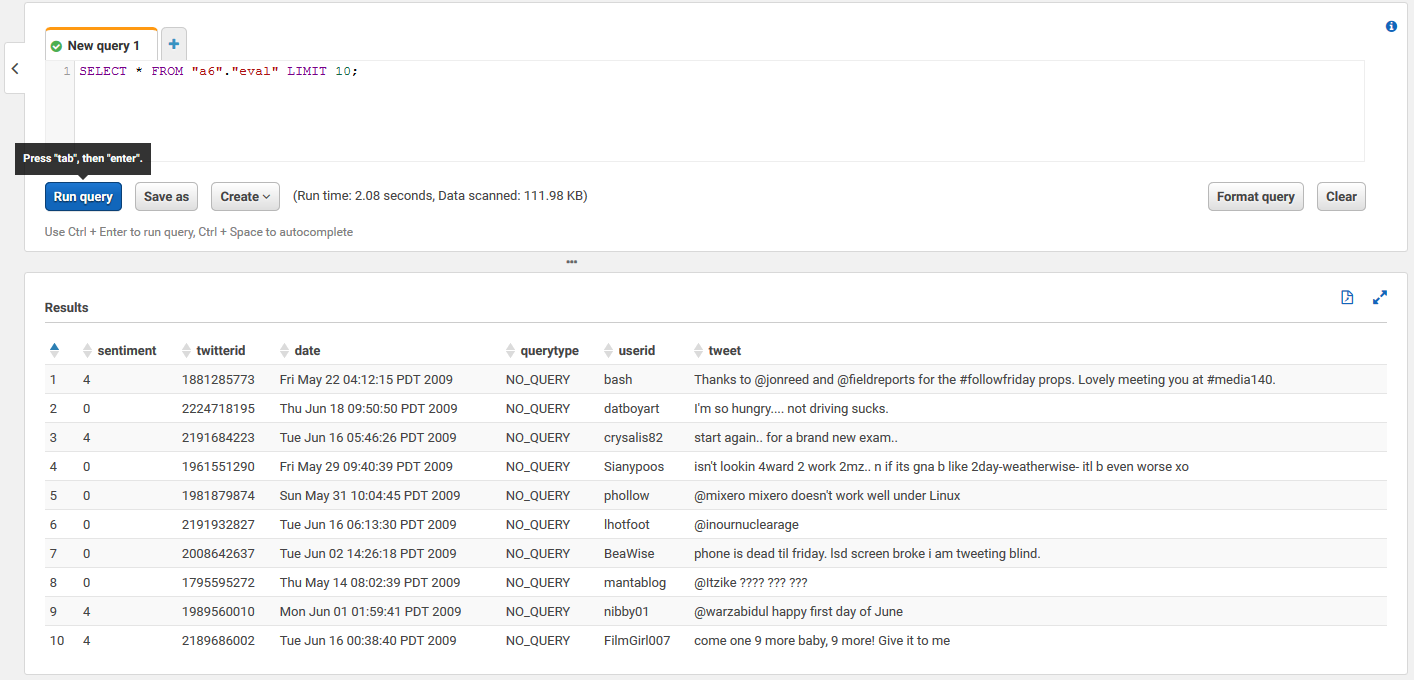


EVAL

Crawler for Eval dataset

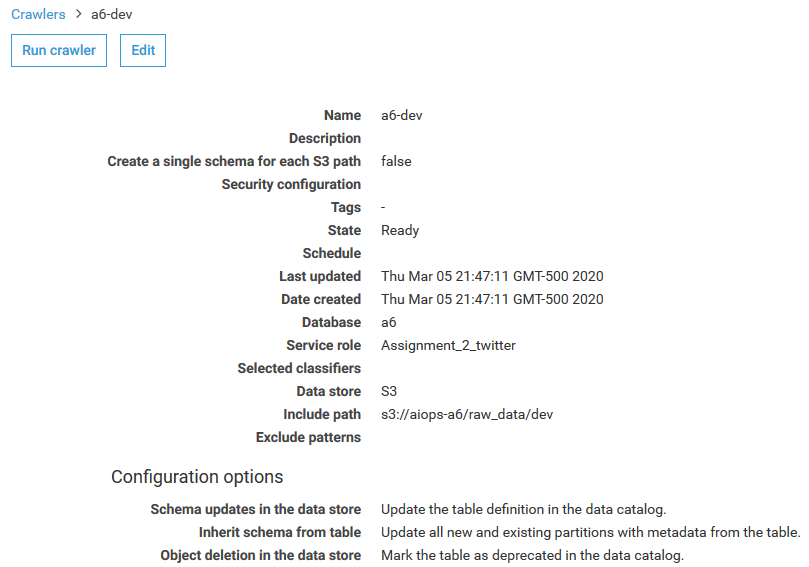


Athena output for Eval dataset

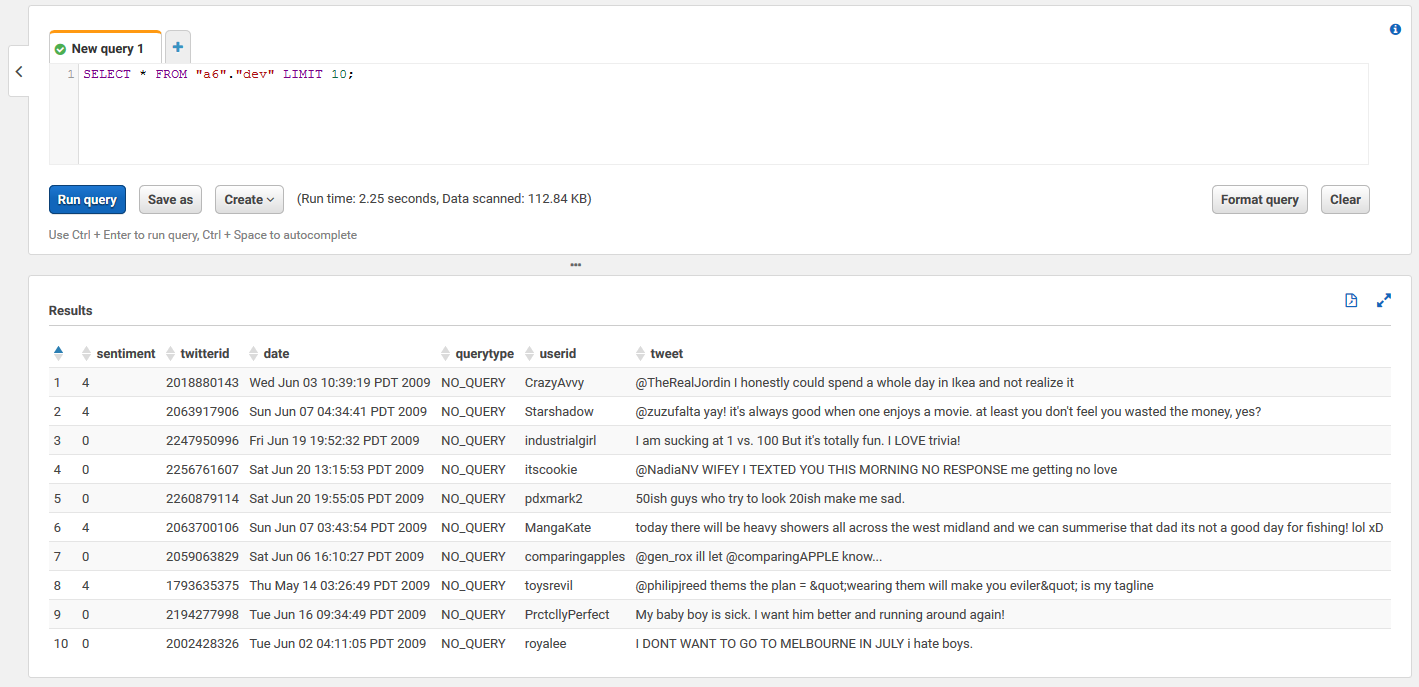


DEV

Crawler for Dev dataset



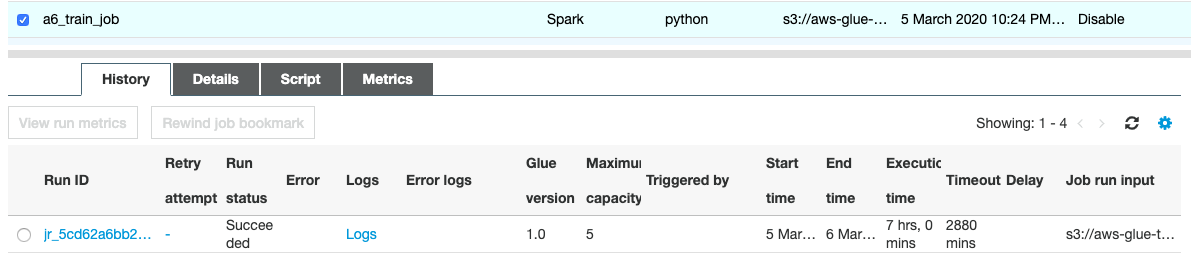
Athena output for Dev dataset



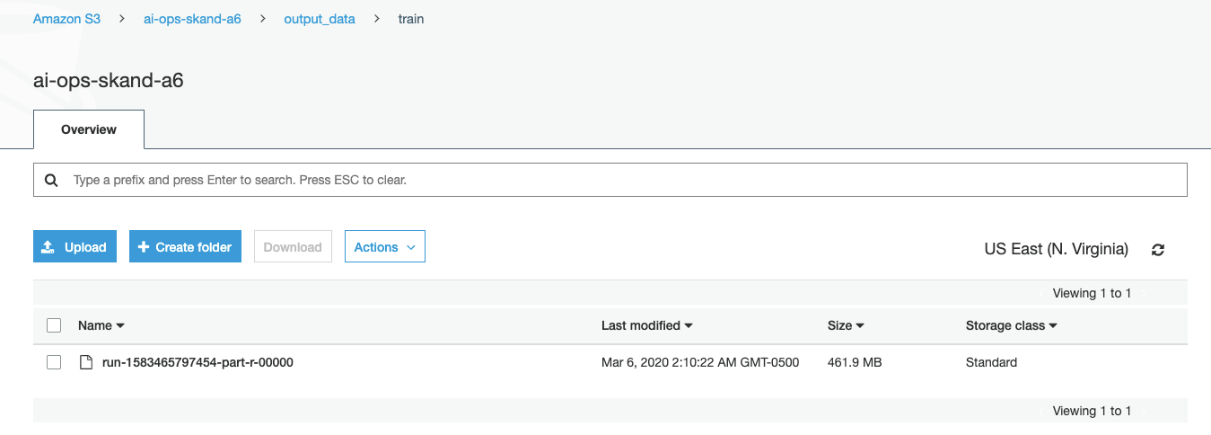
**Step 4: Running a Glue job to create features using the pre-processing library**

TRAIN

Successful completion of the Glue job

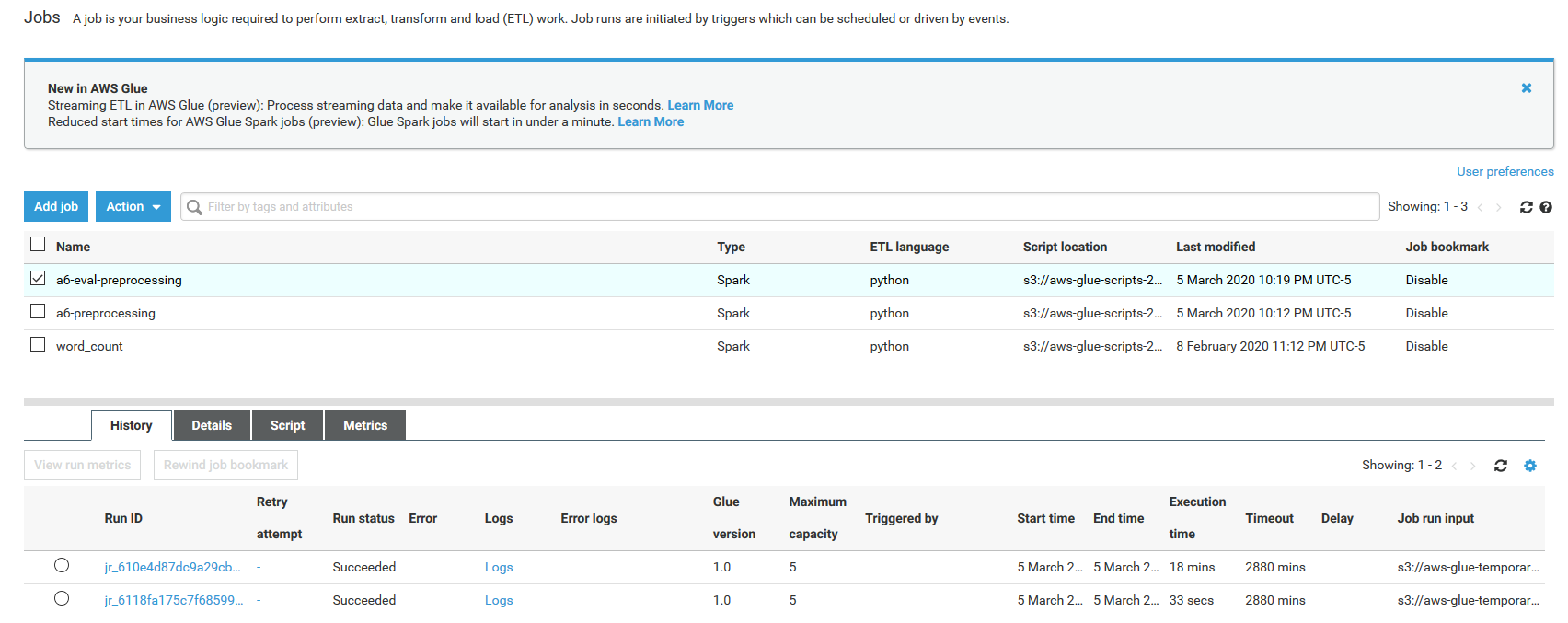


The JSON file generated in S3

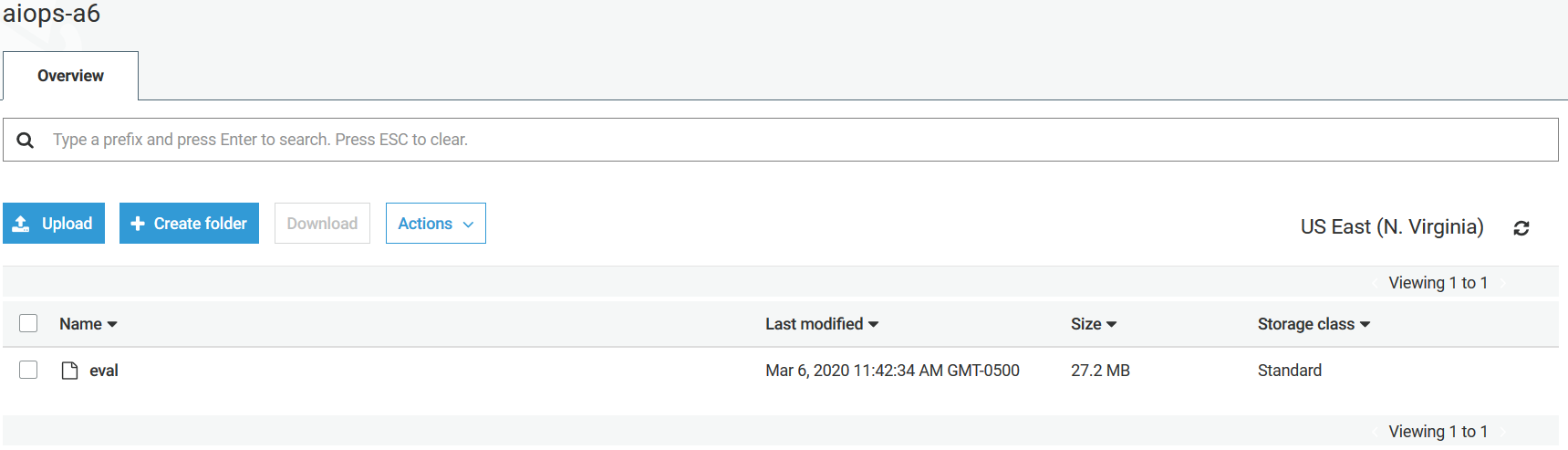


EVAL

Successful completion of the Glue job

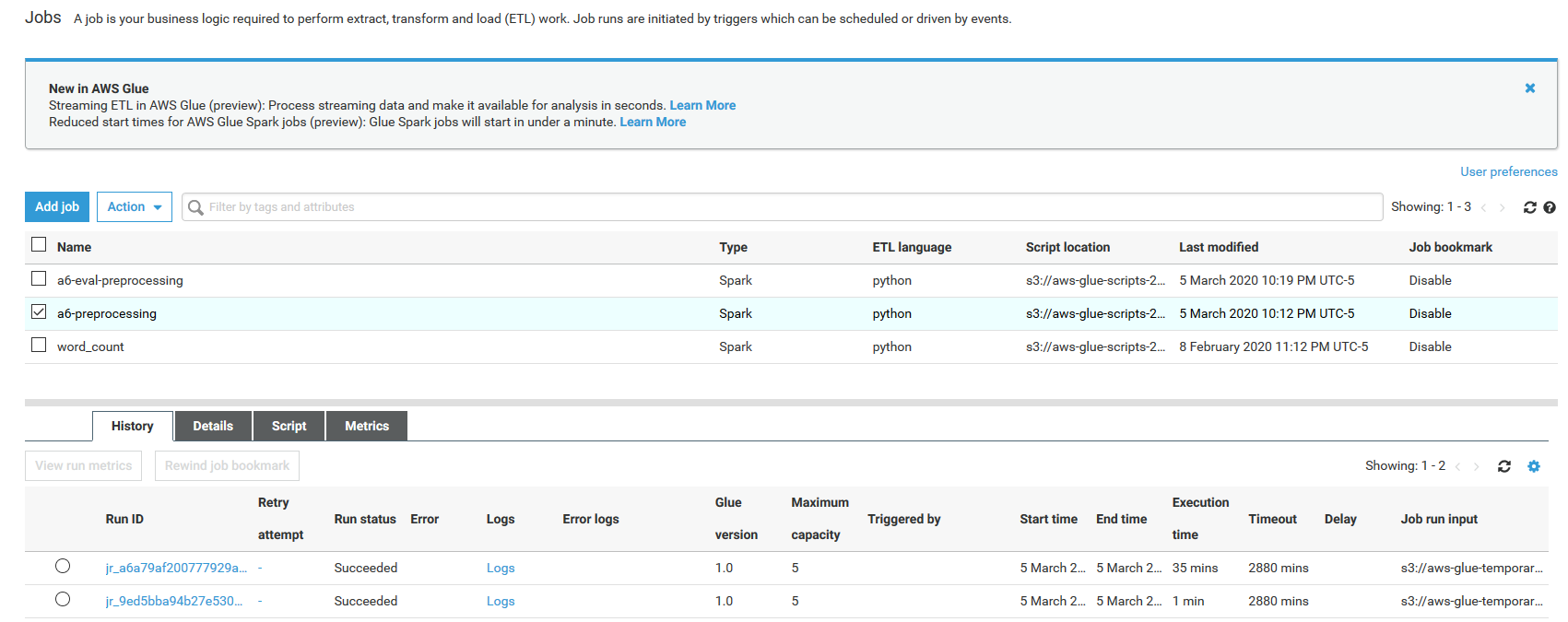


The JSON file generated in S3

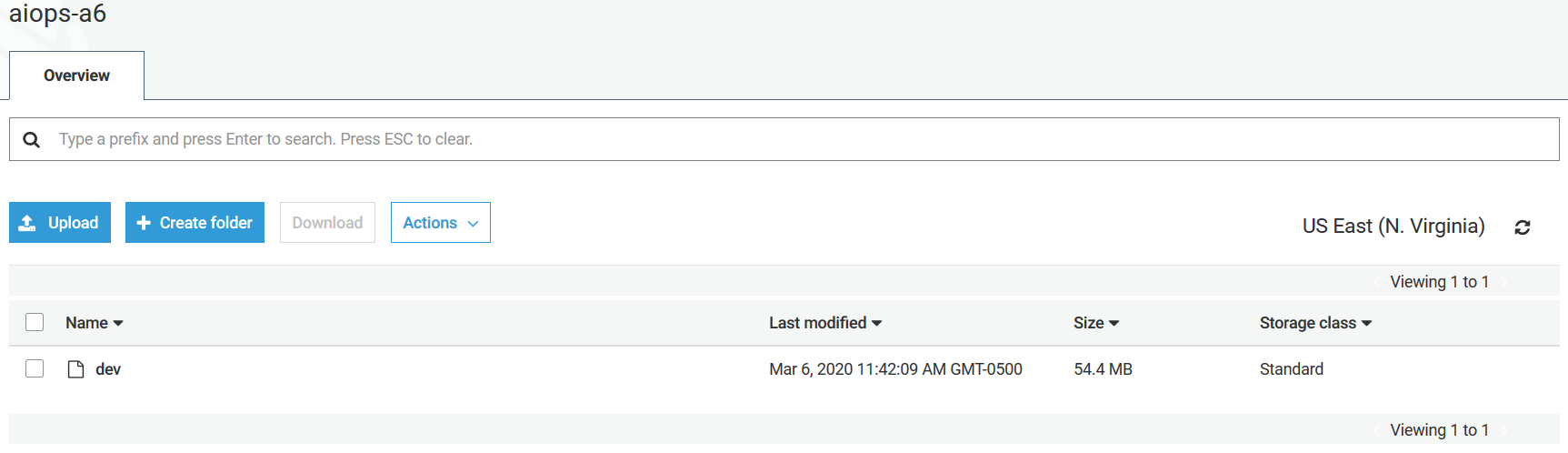


DEV

Successful completion of the Glue job



The JSON file generated in S3



**Step 5: Model training :**

We tried various models which belong to 5 distinct categories:

1. CNN: as given in assignment 4 (we treated this model as the benchmark)
2. Dense Neural Network
3. Deep CNN
4. GRU (Gated recurrent unit)
5. LSTM (Long Short Term Memory)

Let us look into these models in detail:

1. **CNN: as given in assignment 4 (we treated this model as the benchmark)**

Embedding layer

200D

1,193,515 words

Conv1D

100 filters

kernel size = 2

activation = relu

strides = 1

Global Maxpool 1D

Dense layer

1 node

Activation: Sigmoid

Learning rate = 0.0005

Loss = Binary cross-entropy

Optimizer = Adam

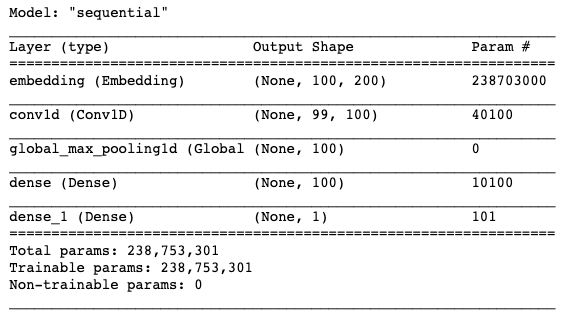
Epochs = 3

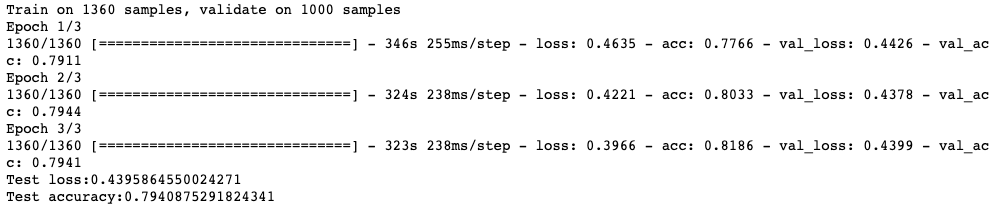
Batch size = 1000

Training accuracy = 0.8186

Validation accuracy = 0.7941

Test accuracy = 0.7941





**Hyperparameters tuned:**

* Learning rate: selected a learning rate which was not too high and too low. Tried different values between 0.00001 to 0.005
* Batch size: a batch size of 1000 gave us better computational performance and fast convergence
* Epochs: We trained for more epochs and observed when the validation accuracy started decreasing. The no. of epochs with the highest validation accuracy was then selected
* We did not change the architecture in this case as this was our benchmark model

1. **Dense Neural Network**

Embedding layer

200D

1,193,515 words

Dense layer

1 node

Activation: Sigmoid

Dropout

0.5

Dense

256 nodes

activation = relu

ddsvs

Dropout

0.5

Dropout

0.5

Dense

128 nodes

activation = tanh

c

Dense

64 nodes

activation = relu

f

Dropout

0.5

Dense

32 nodes

activation = tanh

Dropout

0.2

Learning rate = 0.0005

Loss = Binary cross-entropy

Optimizer = Adam

Epochs = 3

Batch size = 1000

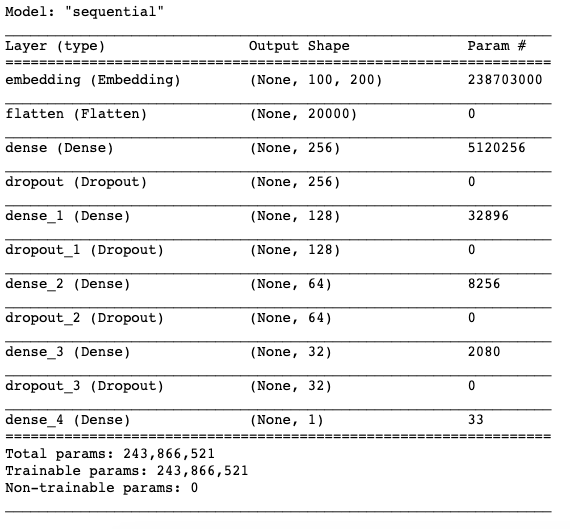
Training accuracy = 0.7982

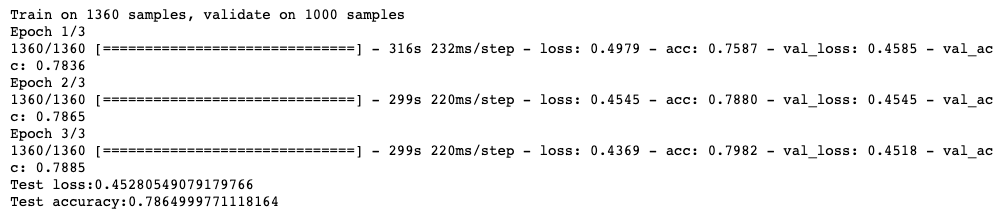
Validation accuracy = 0.7885

Test accuracy = 0.7865

**Hyperparameters tuned:**

* #Layers, nodes, activation, dropout %: tried different architectures and selected the one with the highest validation accuracy
* Learning rate: selected a learning rate which was not too high and too low. Tried different values between 0.00001 to 0.005
* Batch size: a batch size of 1000 gave us better computational performance and fast convergence
* Optimizer: Adam optimizer helped us achieve faster convergence
* Epochs: We trained for more epochs and observed when the validation accuracy started decreasing. The no. of epochs with the highest validation accuracy was then selected





1. **Deep CNN**

Embedding layer

200D

1,193,515 words

Conv1D

128 filters

kernel size = 5

activation = relu

strides = 1

Maxpool 1D

Dense layer

1 node

Activation: Sigmoid

Conv1D

256 filters

kernel size = 5

activation = relu

strides = 1

Maxpool 1D

Conv1D

512 filters

kernel size = 5

activation = relu

strides = 1

Maxpool 1D

Conv1D

1024 filters

kernel size = 5

activation = relu

strides = 1

Maxpool 1D

Learning rate = 0.0005

Loss = Binary cross-entropy

Optimizer = Adam

Epochs = 2

Batch size = 1000

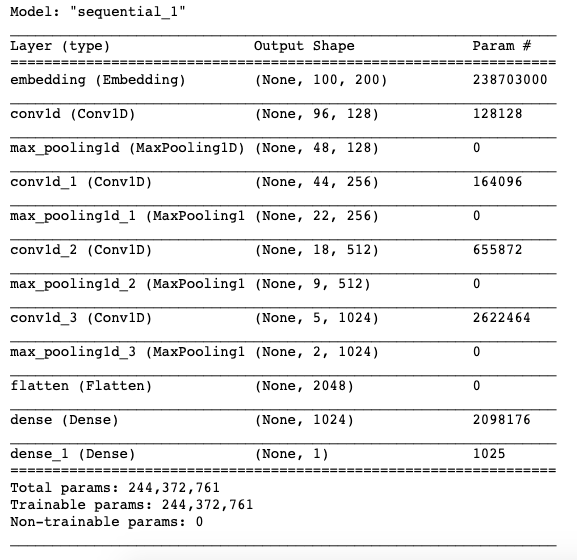
Training accuracy = 0.8066

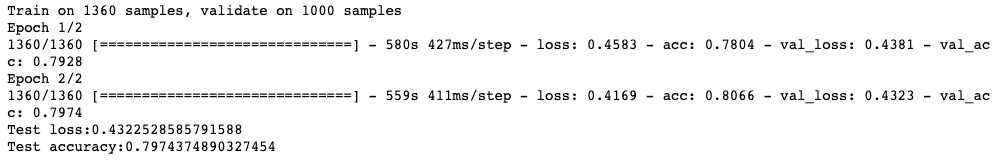
Validation accuracy = 0.7974

Test accuracy = 0.7974

**Hyperparameters tuned:**

* #Layers, nodes, activation: tried different architectures and selected the one with the highest validation accuracy
* Learning rate: selected a learning rate which was not too high and too low. Tried different values between 0.00001 to 0.005
* Batch size: a batch size of 1000 gave us better computational performance and fast convergence
* Optimizer: Adam optimizer helped us achieve faster convergence
* Epochs: We trained for more epochs and observed when the validation accuracy started decreasing. The no. of epochs with the highest validation accuracy was then selected





1. **GRU (Gated recurrent unit)**

Embedding layer

200D

1,193,515 words

Bidirectional

GRU

64 hidden units

Dense layer

1 node

Activation: Sigmoid

Bidirectional

GRU

64 hidden units

Learning rate = 0.0005

Loss = Binary cross-entropy

Optimizer = Adam

Epochs = 4

Batch size = 1000

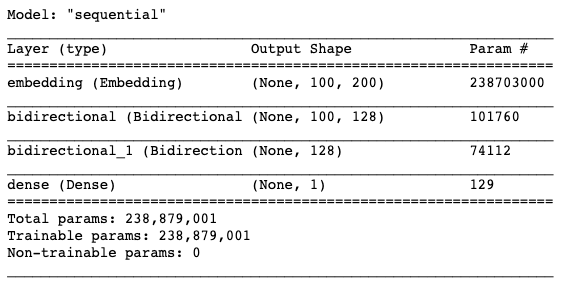
Training accuracy = 0.8009

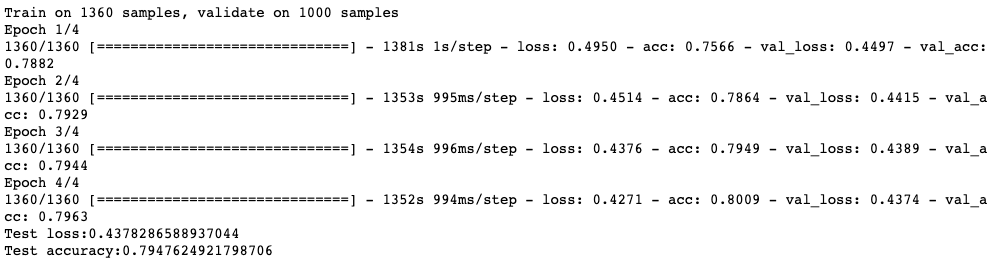
Validation accuracy = 0.7963

Test accuracy = 0.7947

**Hyperparameters tuned:**

* Bidirectional, single direction, hidden units, dropout %: tried different architectures and selected the one with the highest validation accuracy
* Learning rate: selected a learning rate which was not too high and too low. Tried different values between 0.00001 to 0.005
* Batch size: a batch size of 1000 gave us better computational performance and fast convergence
* Optimizer: Adam optimizer helped us achieve faster convergence
* Epochs: We trained for more epochs and observed when the validation accuracy started decreasing. The no. of epochs with the highest validation accuracy was then selected

****

****

1. **LSTM (Long Short Term Memory)**

Embedding layer

200D

1,193,515 words

Bidirectional LSTM

64 hidden units

Dense layer

1 node

Activation: Sigmoid

Bidirectional LSTM

64 hidden units

Learning rate = 0.0005

Loss = Binary cross-entropy

Optimizer = Adam

Epochs = 5

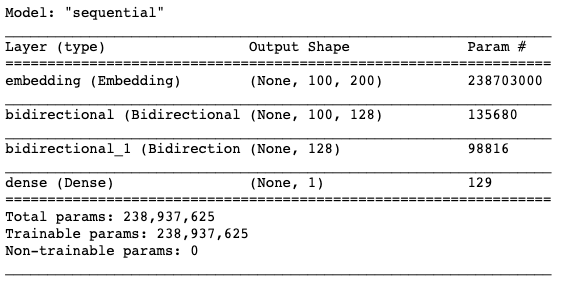
Batch size = 1000

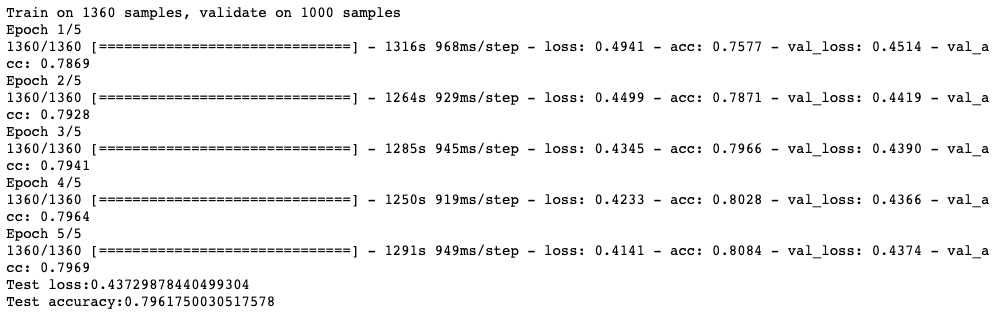
Training accuracy = 0.8084

Validation accuracy = 0.7969

Test accuracy = 0.7961

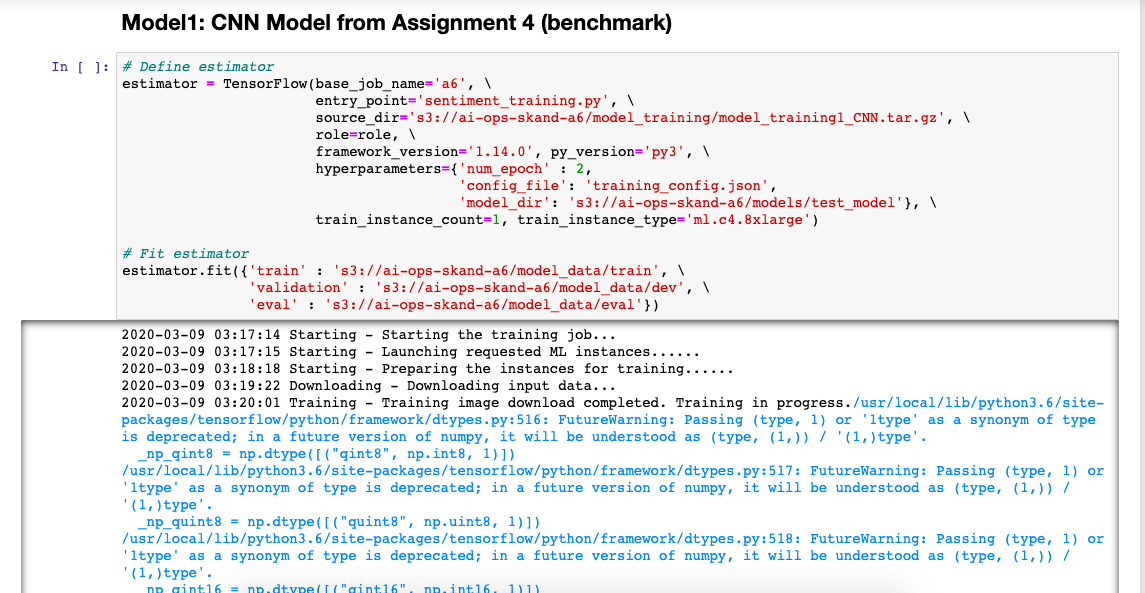
* Bidirectional, single direction, hidden units, dropout %: tried different architectures and selected the one with the highest validation accuracy
* Learning rate: selected a learning rate which was not too high and too low. Tried different values between 0.00001 to 0.005
* Batch size: a batch size of 1000 gave us better computational performance and fast convergence
* Optimizer: Adam optimizer helped us achieve faster convergence
* Epochs: We trained for more epochs and observed when the validation accuracy started decreasing. The no. of epochs with the highest validation accuracy was then selected

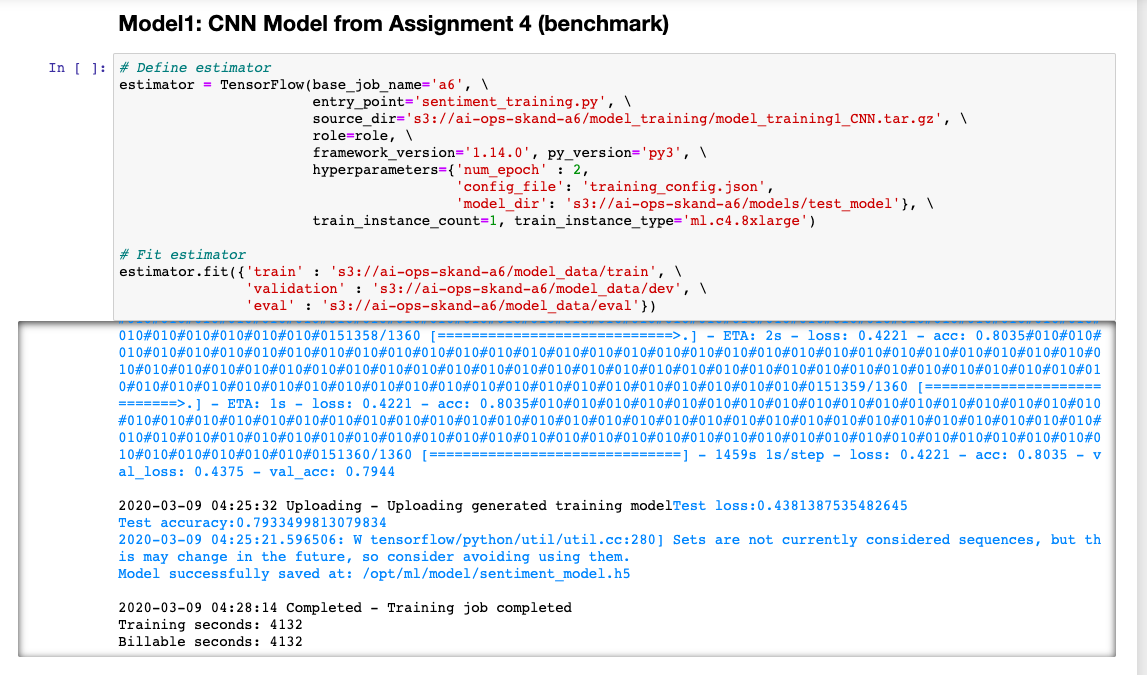
****

****

**The best validation accuracy was achieved using Deep CNN**

Sagemaker training screenshots:

****

****

**Misclassification:**

Next, we manually checked which tweets are getting misclassified to improve the model further (if there is scope)

|  |  |  |  |
| --- | --- | --- | --- |
| **Tweet** | **Prediction** | **Actual Sentiment** | **Comments** |
| @Itzike ???? ??? ??? | 0.76422673 | 0 | *No sentiment* |
| Grill in the oven is almost as good as on a grill. But I have no yard to grill in | 0.95532244 | 0 | *Mixed emotions* |
| Reading reviews.. One big flaw, only 8GB of storage on the palm. LAF!! Roll on iPhone release - 32GB on board. Defo not iPhone killer. X | 0.2646958 | 1 | *Confusing* |
| thinking of you.. | 0.51759243 | 0 | *Subject to interpretation* |
| I am so bored, someone entertain me? | 0.2892712 | 1 | *Neutral* |
| No more ice | 0.72411424 | 0 | *Subject to interpretation* |
| wow the things that could happen to LeBRON James. SMH. lol. | 0.26916614 | 1 | *Confusing SMH and lol together* |
| m off... studying again | 0.046333984 | 1 | *May be mislabeled* |
| First day of finals and I don't have any | 0.47228208 | 1 | *Confusing* |
| @kirawr OMG..and you didnt tell me this before..Hhmm?!?! I cant see not zombie siching on you tho bwahahaha i love you | 0.52002203 | 0 | *Mixed emotions* |
| @luckyluckster lol so we are quoting tupac now? i am not mad at it | 0.21758725 | 1 | *Confusing* |
| @hye\_jin i'm actually back at tcnj taking classes omgsh you're in belize?! so lucky!! how long are you there for?? oo what are the NCLEX? | 0.5652111 | 0 | *Looks mislabeled* |
| @v18rocks MAGIC. only hope left. | 0.21589482 | 1 | *Looks mislabeled* |

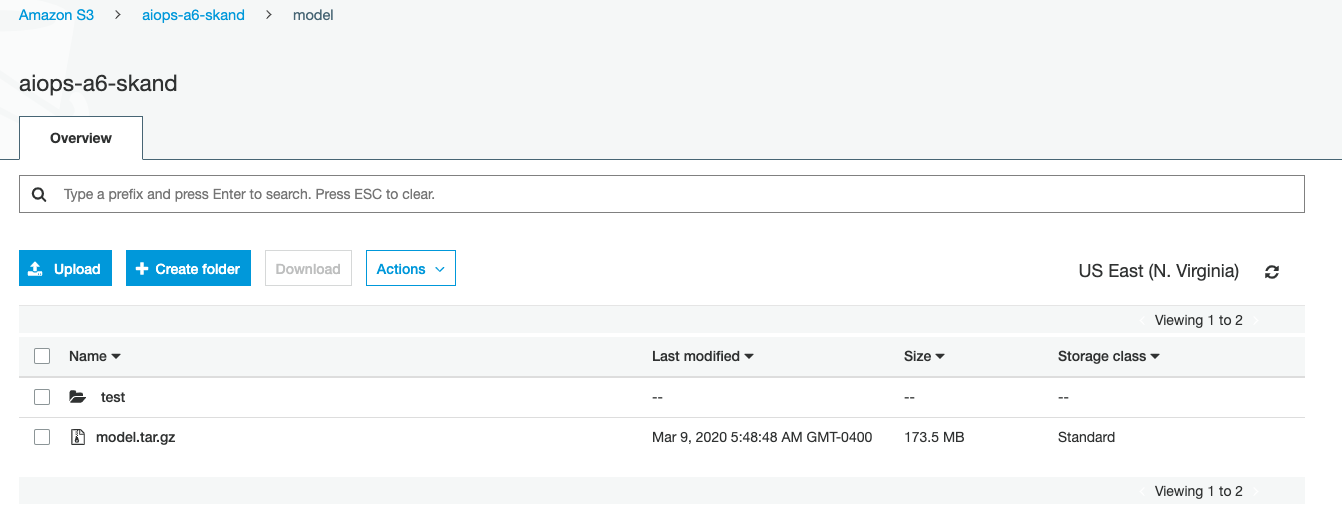
Most of the misclassification that we found while skimming at the misclassified tweets were mostly correct or just could not be improved any further. 20% of error looks like a reasonable estimate that even a human would be prone to while labelling the tweets. Therefore, we believe that the model performs as per our expectations.

**Step 6: Sagemaker Implementation for hosting model:**

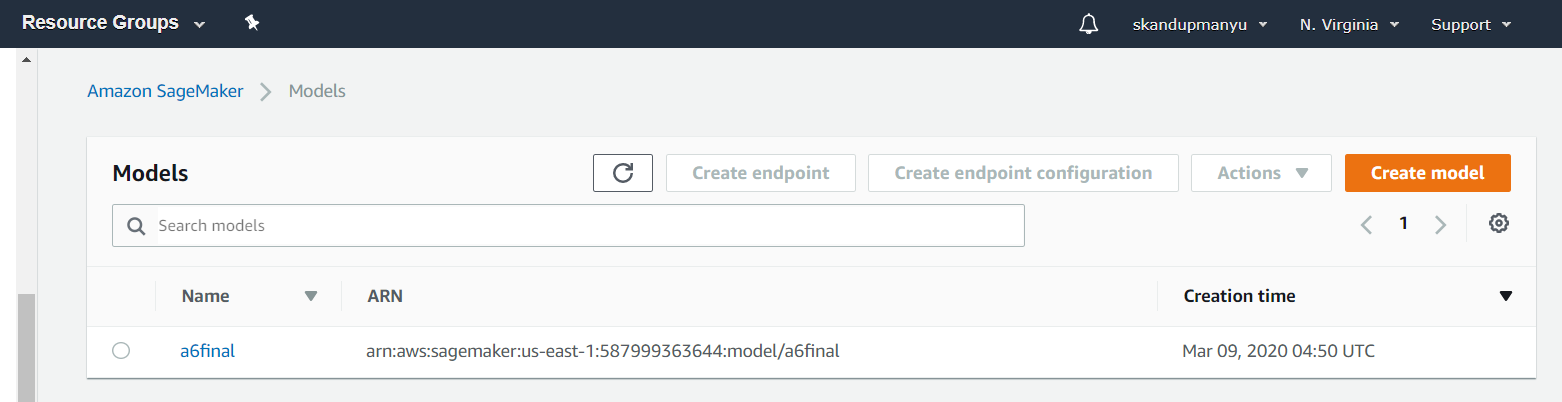
Model implementation required serving the model as an endpoint. Our models with 200D embeddings were huge in size (>3GB) and AWS Educate as well as free tier account did not allow us to serve these models.

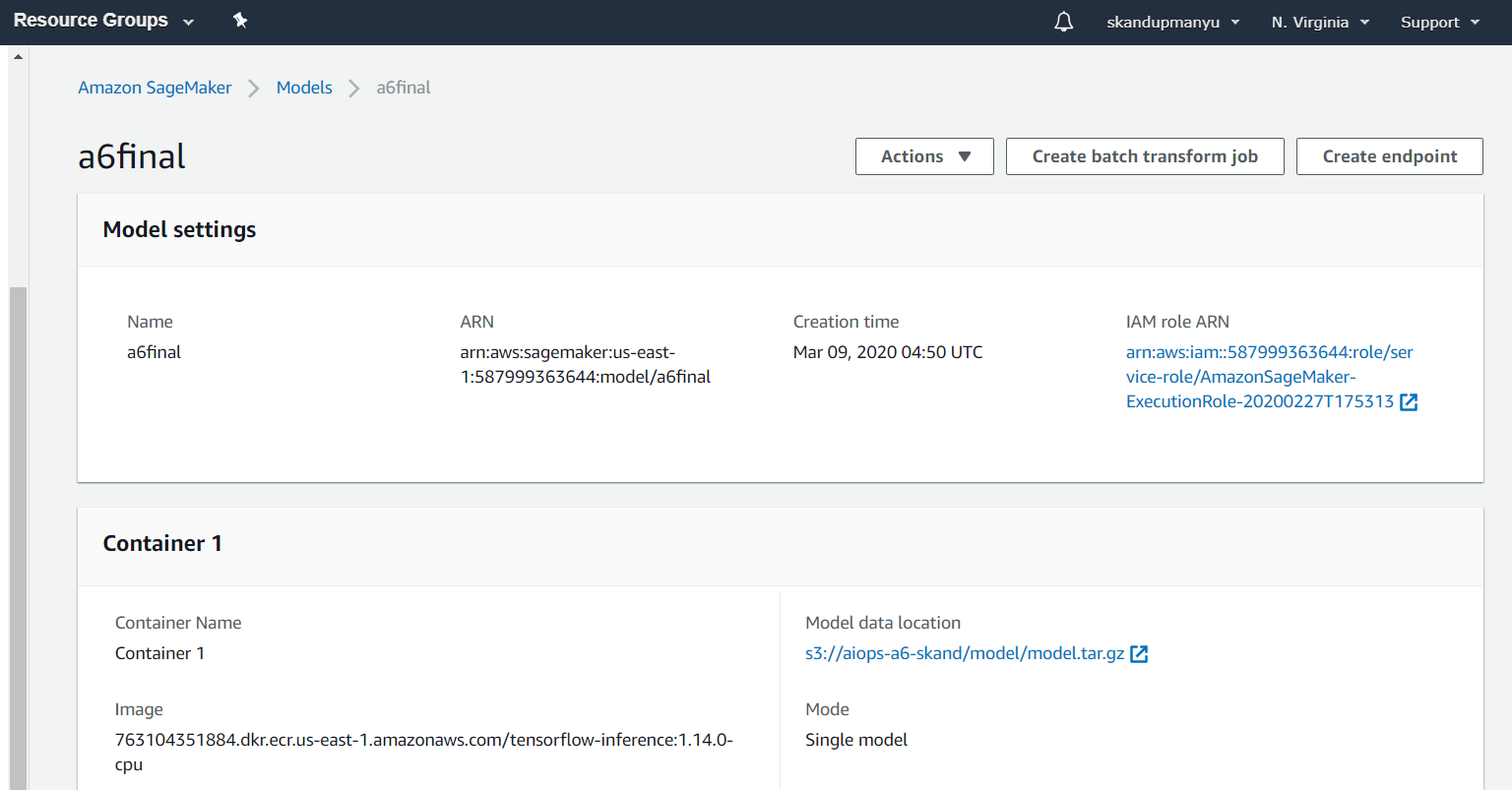
To reduce the model size we kept out model architecture the same but reduced our embedding dimensions to 25D. However, this led to a very small decrease in test accuracy (~0.25%)

Stored model in S3:

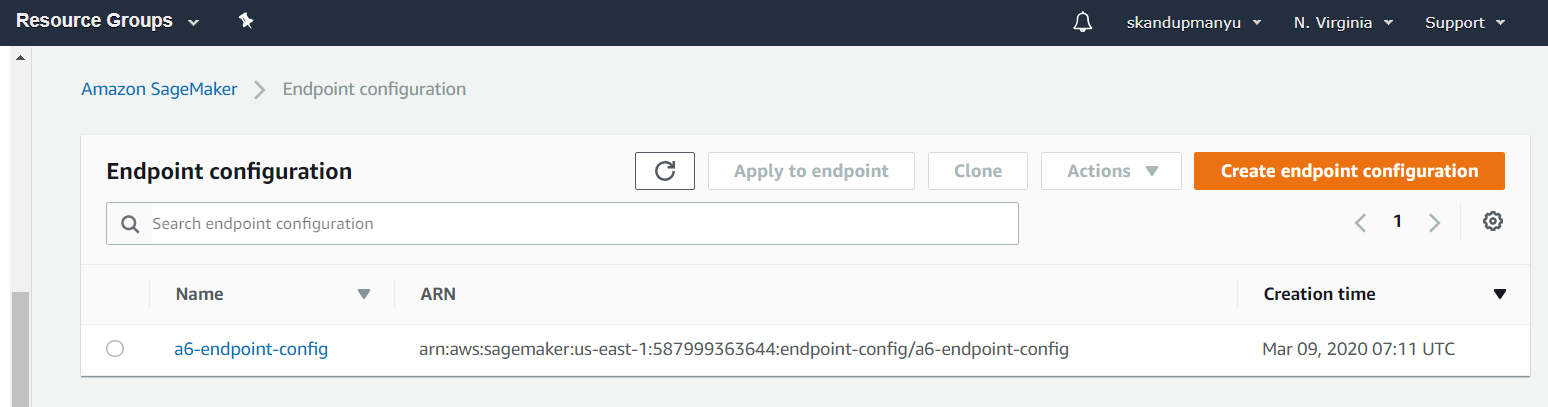


Sagemaker Model:

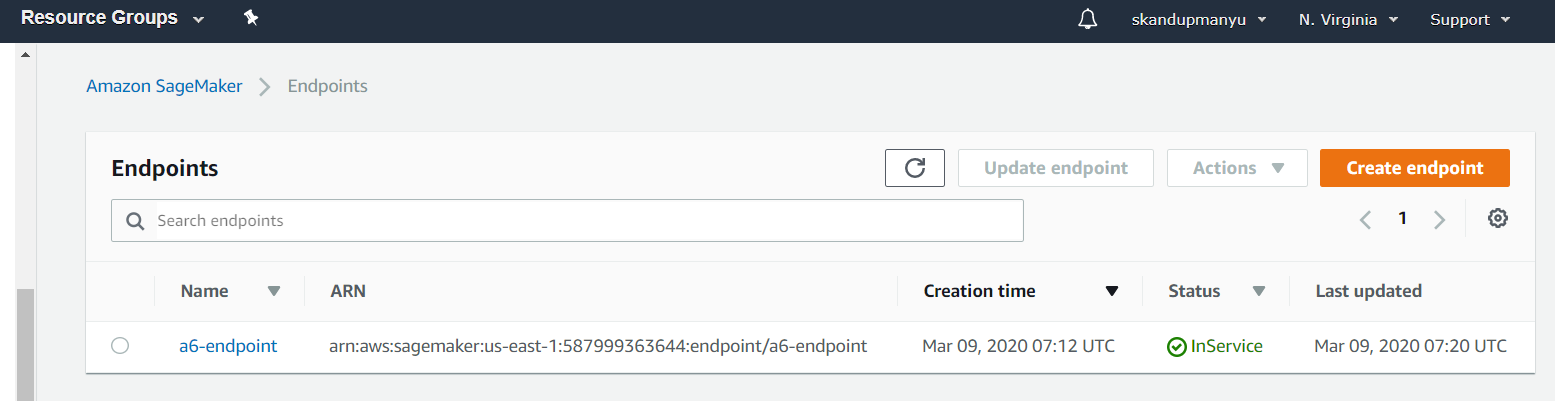


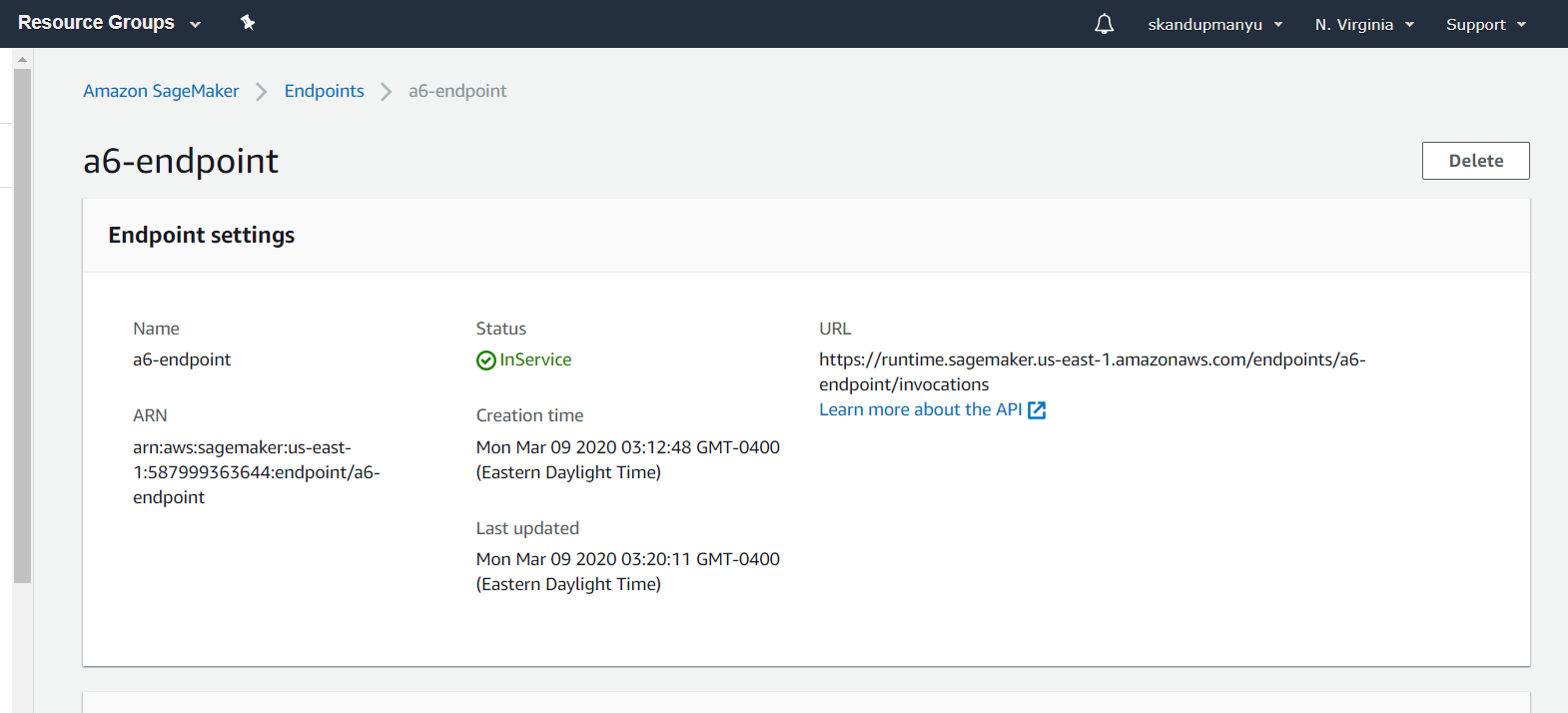


Config:



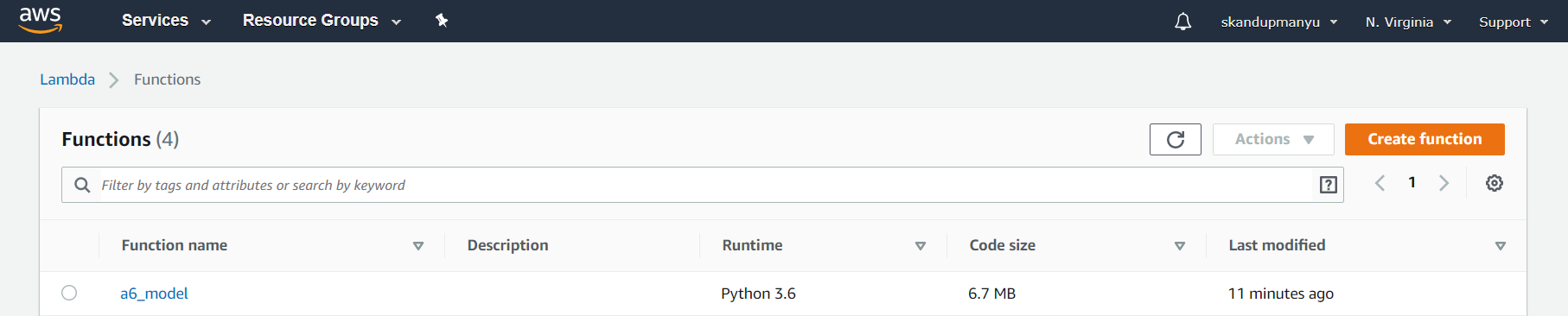
Endpoint:

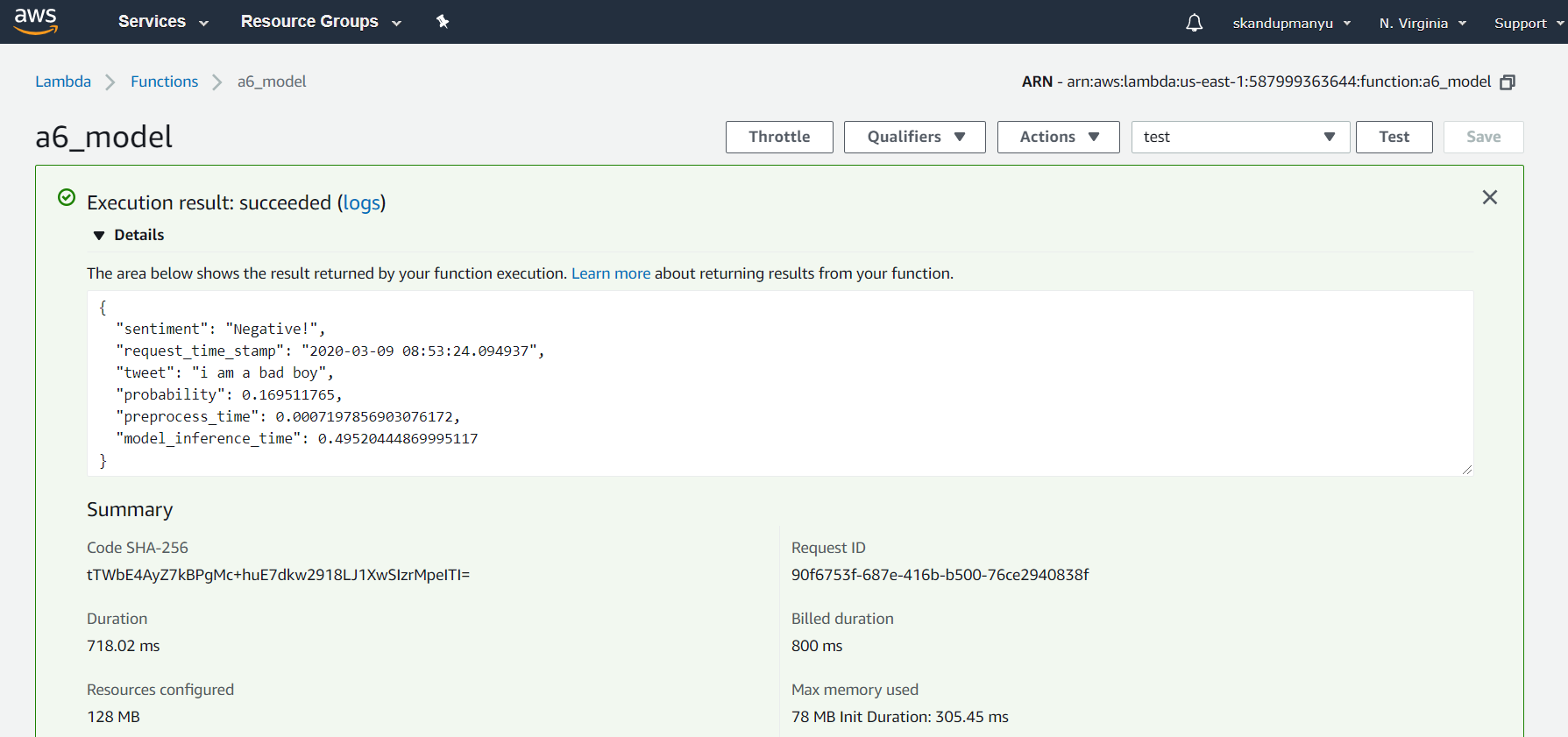




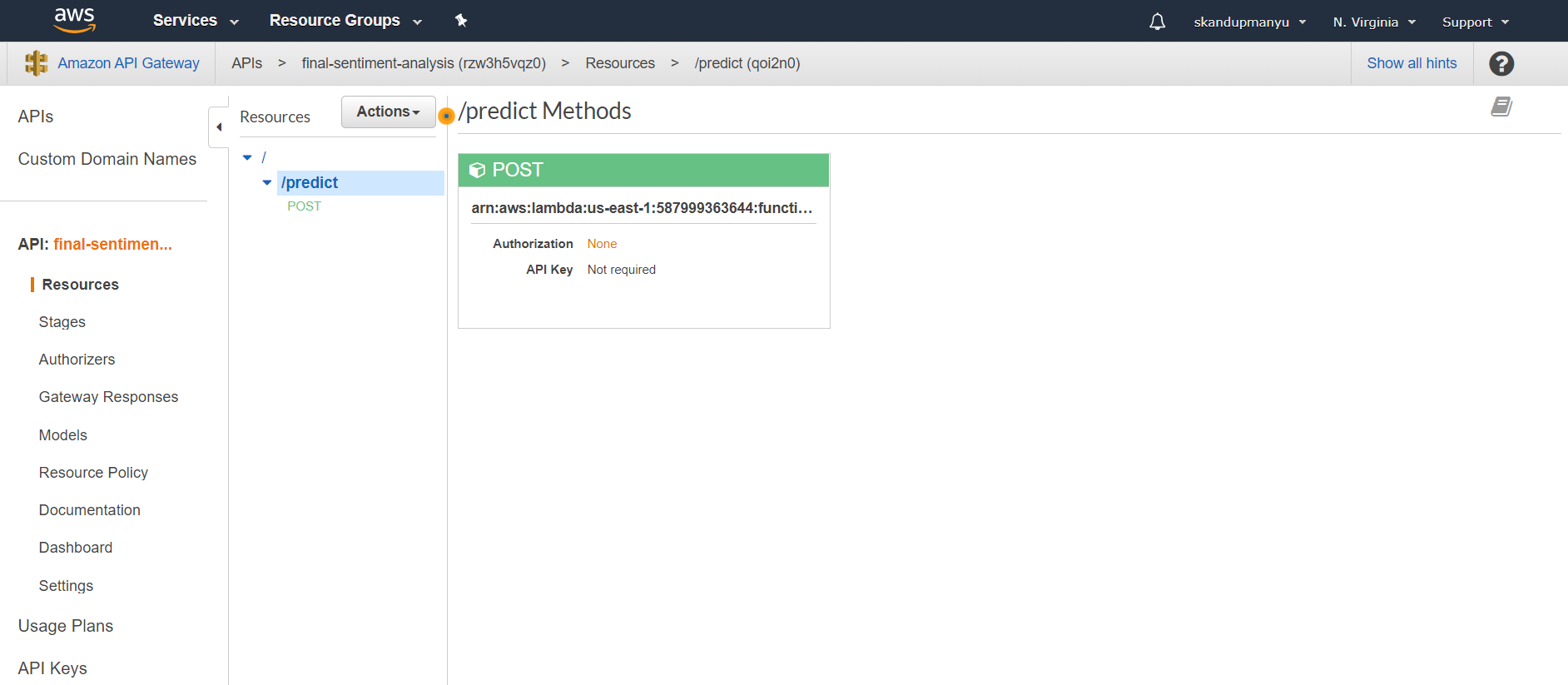
**Step 7: API Gateway and Lambda implementation:**

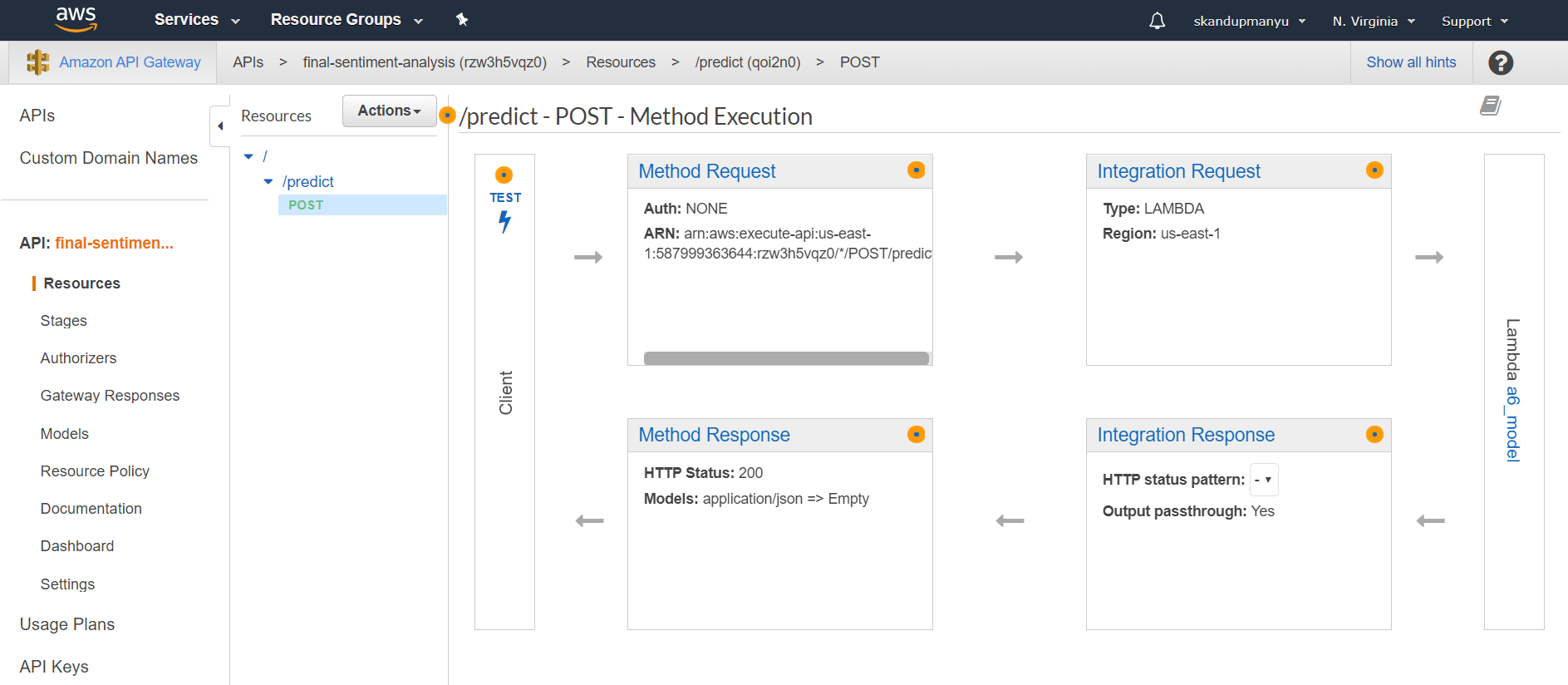
Lambda:

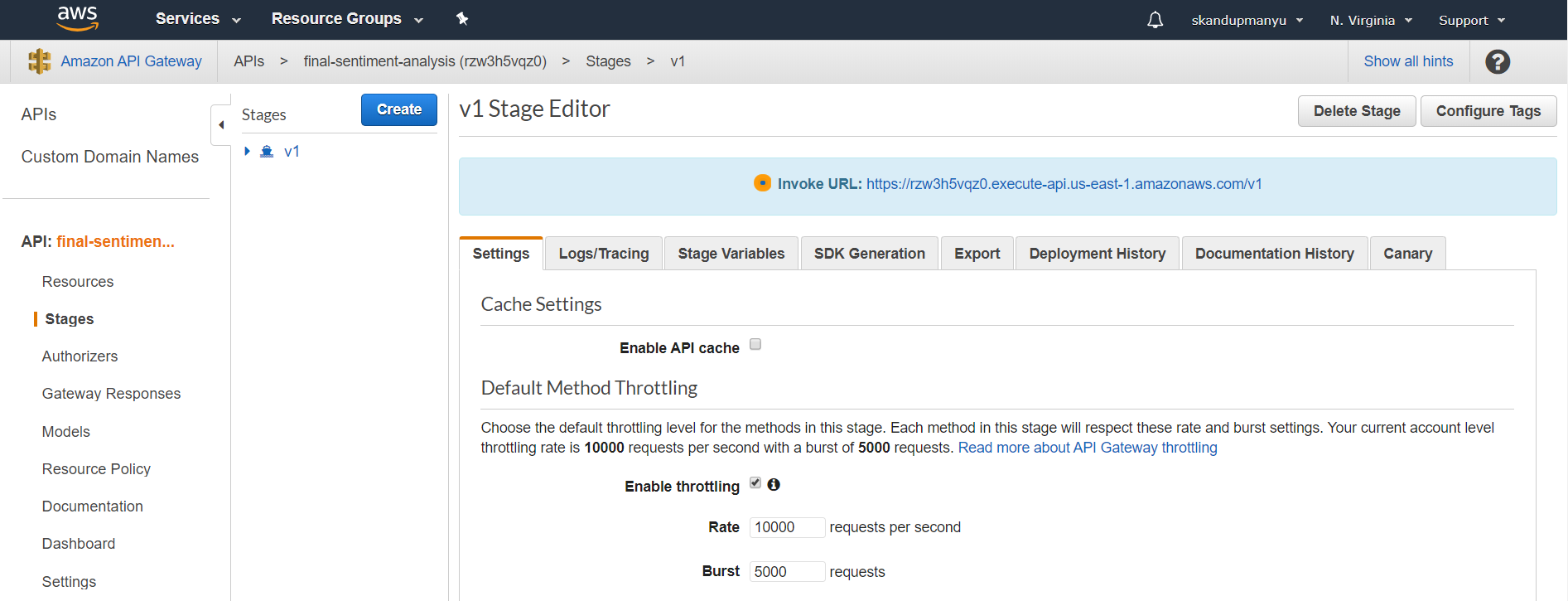
****

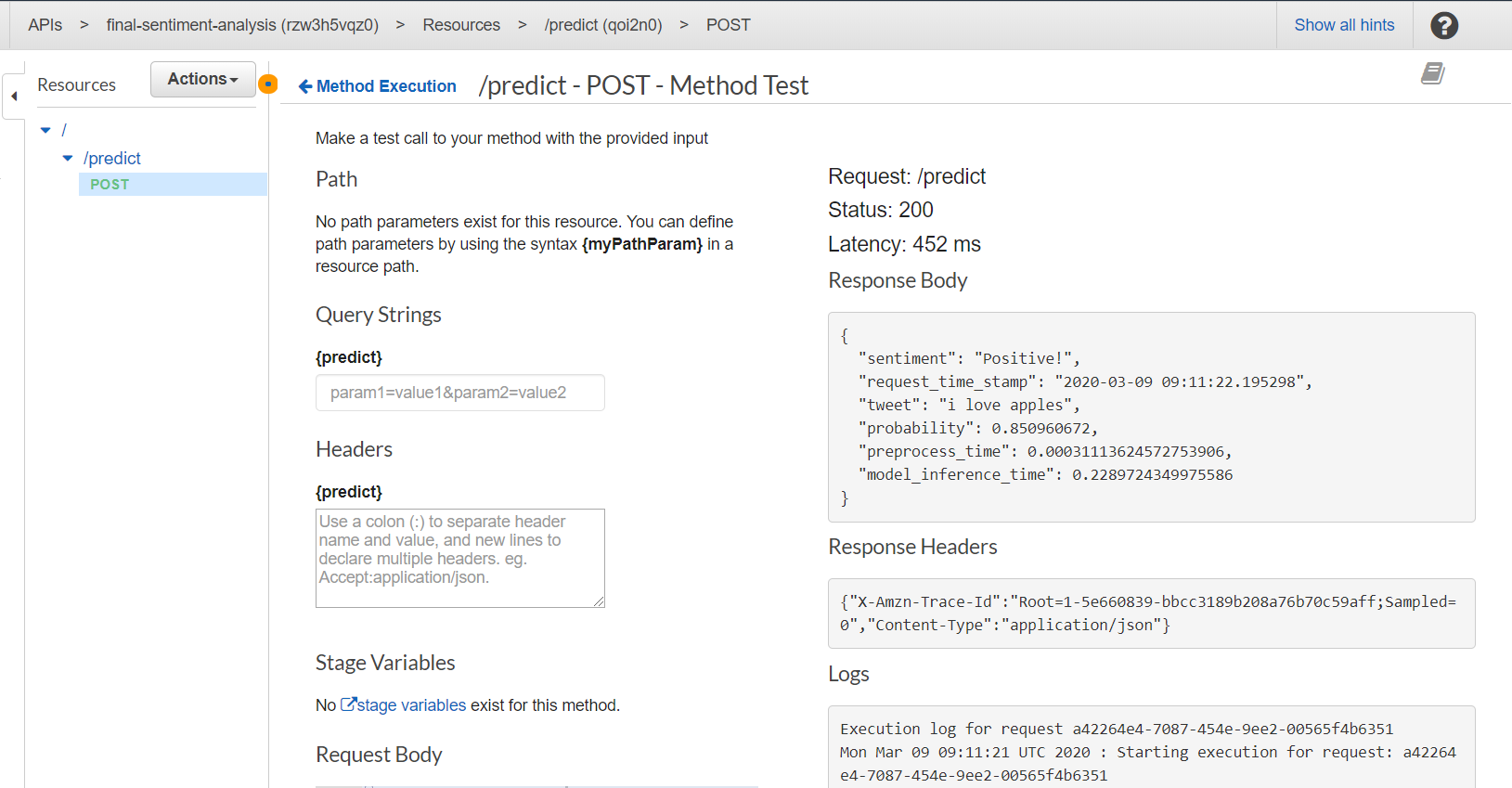


API Gateway:

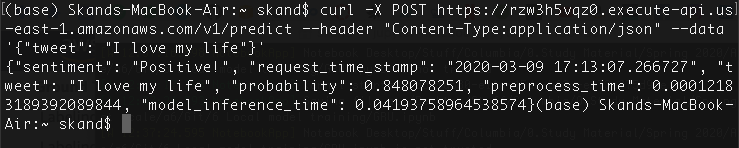




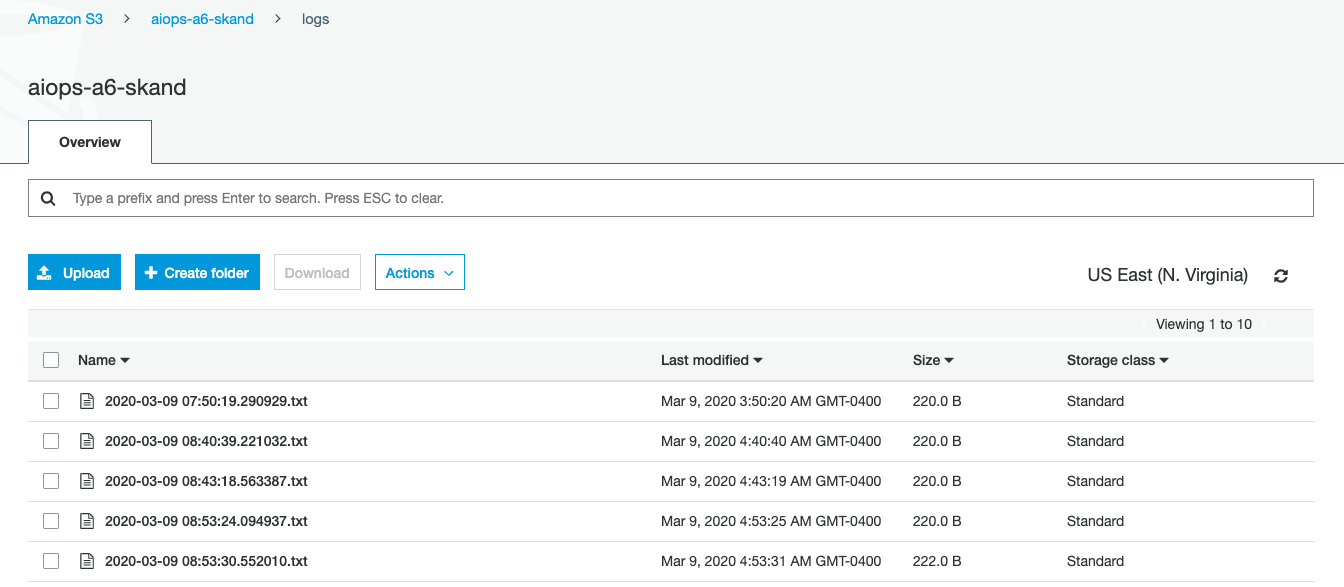




API call for the final model:



Logs:



**API Gateway link**

<https://rzw3h5vqz0.execute-api.us-east-1.amazonaws.com/v1/predict>

**API call:**

curl -X POST https://rzw3h5vqz0.execute-api.us-east-1.amazonaws.com/v1/predict --header "Content-Type:application/json" --data '{"tweet": "I love my life"}'

**Github:**

<https://github.com/skandupmanyu/AI-Ops-A6>

**Notes:**

* If you find the endpoint is not active, kindly let us know. We’ll start it immediately.
* Also, kindly let us know when the evaluation is complete, so that we can delete our model endpoint (our model is being served in our private account due to limitations in AWS Educate)