SageMaker Instance

I selected ml.t3.medium for SageMaker notebook instance. I did not need a lot of resources to run the notebook. For training, i used the ml.m5.xlarge because itw as enough for training an image classification model.

EC2 Instance

I used the t2.micro instance type whose pricing is $0.0116 per hour . The goal for me is to minimize the costs for the practice test.

Writing code for EC2 is similar to writing code for python files to be run via a command-line or terminal, unlike SageMaker where the code is run through a managed notebook.

Lambda Function

The lambda function is written to accept data in a JSON object and invoke an endpoint that returns a response with a prediction. this function is intermediate between the endpoint and the end users. This helps isolate the endpoint and better tune security and performance issues. The function then packages information such as status code and body into a dictionary and returns it as the response of the function.

Result Liste from Lambda Function

"[[-9.994961738586426, -5.361684799194336, -4.477149486541748, -1.5664318799972534, -3.6180477142333984, -4.1265177726745605, -3.1639606952667236, -1.6438956260681152,
-7.648074626922607, -1.556246042251587, -0.6145534515380859, -3.9728944301605225, -2.8790643215179443, 0.11794600635766983, -6.999016761779785, -4.531132698059082,
-10.085627555847168, -3.491299629211426, -5.183989524841309, 0.9071376323699951, -7.236753463745117, -4.027315616607666, -8.2979154586792, -5.298381328582764,
-5.985414028167725, -9.395801544189453, -5.135435104370117, -5.614494323730469, -6.592791557312012, -3.022719621658325, -5.401257514953613, -5.063643455505371,
-12.765028953552246, -5.349846839904785, -8.969167709350586, -6.4312238693237305, -6.81397008895874, -5.566531658172607, -1.7709689140319824, -5.814838886260986,
-6.910645008087158, -3.0296523571014404, -0.3117446005344391,
-3.106391668319702, -2.1201705932617188, -7.482645511627197, -2.84690523147583, -1.6788582801818848, -2.6970763206481934, -3.4510350227355957, -7.645983695983887,
-9.026556968688965, -7.3322601318359375, -4.293950080871582, -6.468623638153076, -2.502114772796631, -6.275299549102783, -7.275440216064453, -3.406559467315674,
-4.3225321769714355, -6.352129936218262, -6.506368637084961, -10.113359451293945,
-7.692389488220215, -5.991590976715088, -8.220724105834961, -2.227949619293213,
-5.496256351470947, -3.516862392425537, -2.901611804962158, -0.365703284740448, -5.492257118225098, -5.9145941734313965, -7.558979511260986, -6.080341815948486,
-3.7347638607025146, -7.498807430267334, -2.952998399734497, -5.376616954803467, -6.582009315490723, -1.3019951581954956, -8.43067455291748, -1.9520015716552734,

-1.862418532371521, -7.93348503112793, -7.408131122589111, -2.521777391433716, -10.308574676513672, -3.77668833732605, -0.835669755935669, -9.457703590393066, -7.865936279296875, -6.489157676696777, -9.659119606018066, -5.167778968811035, -3.208366870880127, -6.292455673217773, -5.533816337585449, -6.590264320373535, -7.379441738128662, -8.746428489685059, -3.040257692337036, -5.022286891937256, -5.438302993774414, -7.36074161529541, -9.069205284118652, -4.749107360839844, -1.835982322692871, -3.1804168224334717, -1.2269680500030518, -1.3724141120910645, -2.5797410011291504, -9.594913482666016, -7.267094612121582, -6.939950942993164,
-3.184603691101074, -10.564037322998047, -3.148885726928711, -7.576478004455566, -2.083707809448242, -4.736138820648193, -4.558735370635986, -7.307677268981934, -4.363959789276123, -8.701939582824707, -8.849624633789062, -6.742290496826172, -1.3212451934814453, -7.149965286254883, -8.149775505065918, -7.93416166305542, -1.7762694358825684, -6.070013046264648]]"

Workspace Security

During the project, I encountered a security problem when I wanted to invoke the endpoint from the lambda function. I didn't have to call the endpoint. I had to add privileges to the roles of the lambda function to be able to call the endpoint

Concurrency and Autoscaling

As for concurrency, I opted for provisioned concurrency, as in the long run a project like this one may receive a lot of traffic and having an automatic way of handling concurrency seemed rather efficient. When it came to autoscaling, I set my Maximum Instance Count to 3 as I was using the ml.m5.large instance type which has a substantially good amount of resources.
I set scale-in cool down time period to 30 because if i chose a high number , the AWS will wait a long time before deploying more instances. This helps me avoid incurring costs for mementary spike in traffic. If i chose a low number, then AWS will deploy instances more quickly, but this responsiveness will be more costly.
I set scale-out cool down time period to 30. Because if i chose a high number, the AWS will keep extra instance deployed longer, but this extra capacity will be more costly.