Deployment Pipeline in AWS

1. In local laptop, use support\_vector\_machine.ipynb file and train a SVM classification model on the train - train data set.
2. Test the model and save the model file locally using joblib. We have to upload and run this notebook in SageMaker.
3. In AWS console, create a SageMaker notebook instance and open a notebook. Upload the locally trained model, the train - train.csv and the preprocess.ipynb files to the sageMaker notebook.
4. Run the preprocess notebook in SageMaker. The notebook code does the following.

* Load the model file, open it and test and then upload it to a S3 bucket (from where SageMaker will take the model artifacts).
* Create a SageMaker model object from the model stored in S3. SageMaker built-in SVM container will be used for this purpose, as the model was locally trained with SVM algorithm.
* Create an Endpoint Configuration. Endpoint is the interface through which the outer world can use a deployed model for predictions.
* Create an Endpoint for the model.
* Invoke the endpoint from within the deployment notebook to confirm the endpoint and the model are working fine.
* After running the notebook till this point, the endpoint created under: Sagemaker -> Inference -> Endpoints in AWS console.
* Note down the endpoint name displayed. This will be used while creating the Lambda function.

1. Create a IAM role that includes the policy which gives your Lambda function permission to invoke a model endpoint. Select Lambda as the use case in AWS service, while creating the role and attach the policy to the role.
2. Create a Lambda function that calls the SageMaker runtime invoke\_endpoint and returns the prediction.
   * Select “Author from Scratch” and give a function name and select Runtime. Select “Use an existing role” and pick the role you created in the previous step.
   * Under the code section of the lambda, enter the python code. Click “Deploy” after entering the code.
   * Go to the Configuration tab of the Lambda function and add an environment variable “ENDPOINT\_NAME” and set its value as the same endpoint that was created before. Note that this environment variable is used in the Lambda function’s code.
   * This completes setting up of lambda function.
3. Create a REST API and integrate with the Lambda function.
   * Select API Gateway service on AWS console, and select REST API.
   * Click on Build and select “New API. In the next window select “Create Resource” from Actions drop-down menu, and enter a Resource Name.
   * Note down the Resource Name you choose. It will be a part of the URL created by this service and will be used later when we test the deployment from Postman. After creating the resource, select “Create Method” from Actions drop-down menu.
   * Select POST method and “Lambda Function” as Integration type. Enter the name of the Lambda Function created.
   * Then, select “Deploy API” from Actions drop-down menu. Select Deployment stage “New Stage” and give some stage name.
   * Then, finally when you click “Deploy”, you will be given a “Invoke URL”. It will be used in Postman to contact the API gateway.
   * The deployment and setup of the end-to-end communication path is complete.
4. Finally, use Postman App to POST the test data to API gateway and get the prediction result back from AWS cloud.

