



*FACULTY OF COMPUTER SCIENCE*

***ASSIGNMENT 4***

*In  
The Class of*

***CSCI5710: SERVERLESS DATA PROCESSING***

*by*

*Janvi Patel [B00863421]*

***Submitted to***

*Prof. Saurabh Dey  
Department of Computer Science  
Dalhousie university.*

*Date: 19<sup>th</sup> July 2021*

Part A. Read AWS official documentation on SageMaker and Comprehend. Learn how these services are used in event-driven applications and explore various use cases.

Once you learn about the services, write ½ page summary on each service highlighting the usage, and a statement on how you will be using these services for “hospital management application” or “barber shop application” or “tourism application” (You can pick any one application).

### **AWS Sage Maker:**

AWS Sage Maker is a service provided by Amazon that enables data scientists and developers to quickly and easily build, train and deploy ML models provided any kind of dataset. There are modules that can be used to build, train, and deploy ML models. [1] AWS Sage Maker has some benefits such as that is highly scalable, provides fast training, maintains uptime that process keeps on running without any stoppage, high data security etc. The SageMaker comes with a lot of built-in optimized ML algorithms which are used directly for training purpose. To build a model when we need data we can directly call it from AWS S3 which is storage service provided by Amazon S3. [2]

Once the model is trained and tuned, Amazon SageMaker provides easy way to deploy the model in the production so that we can start running generating predictions on test data. It deploys model on auto-scaling cluster of Amazon EC2 instances that are spread across multiple availability zones to deliver high performance and high availability. [1, 2]

Developers can create and manage the service using APIs, SDKs which can easily accessed by web API. The jupyter notebook provides various options of languages of coding such as python, JavaScript and JAVA. SageMaker provides three APIs to interact with the service: a high-level API using python-sagemaker-sdk, a mid-level API using boto3, and a low-level API using awscli and docker [3].

If we take example of hospital management application, to showcase how Sage Maker will be used, we firstly take hospital data that contains patient information, no of beds available, discharge date, disease name to predict number of beds needed for further use. ML engineer has used this data to predict the beds for corona patients so that government can know about no of beds needs for the corona patient by the pick in the area. This also cuts down on the amount of time it takes to arrange the beds and can save more life. This data is being stored in S3 bucket and the ML algorithm where the model is being trained and deployed on AWS Sage Maker and will be tested on new dataset available on different S3 Bucket.

## **Amazon Comprehend:**

Amazon Comprehend, the natural-language processing (NLP) service that uses ML to discover the information about unstructured data. This is used to process the unseen information easier to understand. It identifies almost all the elements in the data, which can be people, brands, places, entities, languages etc. [4] Comprehend not only locates any content that contains personally identifiable information, it also redacts and masks that content. It uses tokenization and parts of speech to analyses text and automatically categorizes a collection of text files based on similar keywords [5]. AWS comprehend is fast in processing which can process millions of data in a minute.

In the same breadth, Amazon comprehend can do entity detection, key word extraction, language detection, sentiment analysis, and syntax analysis on a single or group of documents. Amazon comprehend can recognize over 100 languages and do sentimental analysis on over 10 of languages [5].

Input for AWS comprehend is the location of S3 bucket which provides data in text language and the second is the text data. After preforming the operations, the AWS Comprehend provides output in JSON format. [5] Comprehend is completely managed, allowing you to run immediately without having to training model from scratch [4]. Amazon Comprehend also works with other AWS services such as Amazon S3, AWS KMS, and AWS Lambda [5].

When we take up scenario of tourism application to run AWS comprehend on the dataset. Assume that we have data in Amazon S3 bucket and to improve the customer experience by intelligently understanding context and language. The very first stage of any travel plan is booking and reservations. This includes choosing the airlines, arranging local taxi services, hotel room etc. The data can be used to find the cheapest fights, airline duration, hotel room prices and resort to get our tasks easy. Provided data from S3 bucket will be analyzed using AWS comprehend and will provide the output in JSON format.

Part B. Build an event-driven serverless application using AWS SageMaker or GCP ML.

In this part of the assignment, you need to use S3 bucket or equivalent GCP storage, and Lambda Functions or GCP Cloud function.

**Step 1: Create your 1<sup>st</sup> S3 bucket\*\* SourceDataB00xxxxxx and upload the files (from 001 to 299) given in the Train folder. You need to write a script or use the SDK to upload the files on the bucket.**

Below figure (1) shows bucket creation “sourcedatab00863421”.

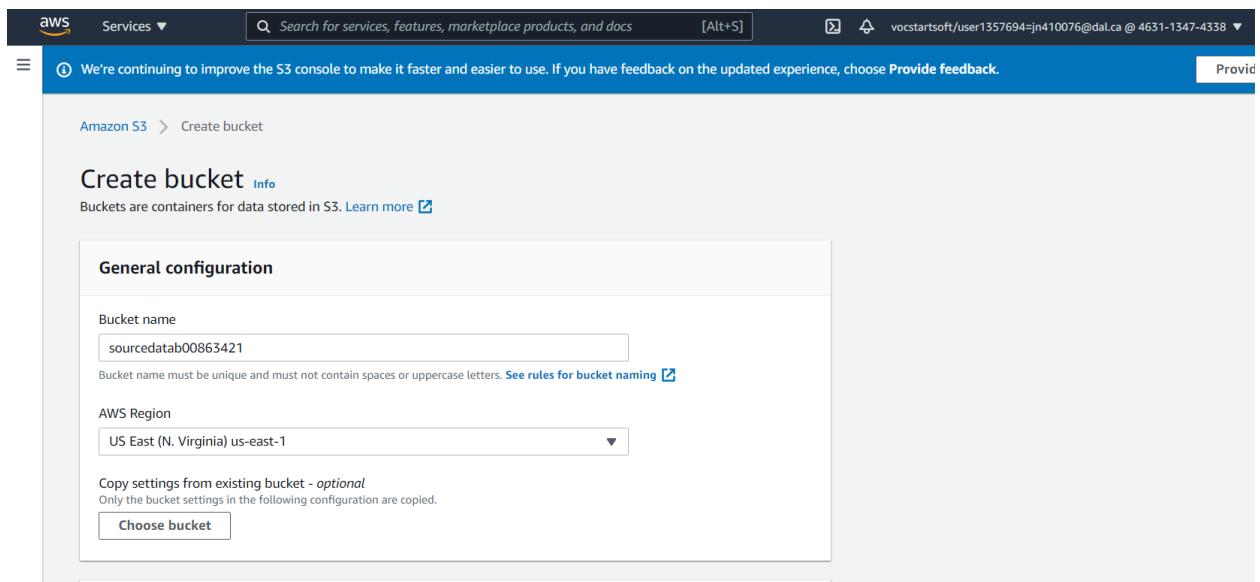


Figure 1: Bucket creation

Below code in figure (2) represents s3 client connection and upload file logic:

```
1 import boto3
2
3
4 def s3_client():
5     """
6     Function: get s3 client
7     Purpose: get s3 client
8     :returns: s3
9     """
10    session = boto3.session.Session()
11    client = session.client('s3')
12    """ :type : pyboto3.s3 """
13    return client
14
15
16 def s3_upload_small_files(inp_file_name, s3_bucket_name, inp_file_key):
17     client = s3_client()
18     s3_bucket_create_response = client.upload_file(inp_file_name, s3_bucket_name, inp_file_key)
19     print(f" ** Successfully file upload done - {s3_bucket_create_response} ")
```

Figure 2: S3 connection and file uploading

Below code in figure (3) represents file uploading from training and testing folder:

```
21
22 for i in range(1, 299):
23     file = i
24     if i < 10_:
25         file = "0" + str(i)
26         inp_file_name = "D:/Study D/Dalhousie/Sem 3/Serverless/Assignments/Assignment 4/Dataset/Train/0" + str(file) + ".txt";
27         inp_file_key = str(file) + ".txt"
28         s3_bucket_name = "sourcedatab00863421"
29         s3_upload_small_files(inp_file_name, s3_bucket_name, inp_file_key)
30
31
32 for i in range(300, 401):
33     inp_file_name = "D:/Study D/Dalhousie/Sem 3/Serverless/Assignments/Assignment 4/Dataset/Test/" + str(file) + ".txt";
34     inp_file_key = str(file) + ".txt"
35     s3_bucket_name = "sourcedatab00863421"
36     s3_upload_small_files(inp_file_name, s3_bucket_name, inp_file_key)
37
```

Figure 3: code for uploading training and testing files

The figure (4) represents the file uploading into source data bucket.

The screenshot displays the AWS Management Console for the Amazon S3 bucket named 'sourcedatab00863421'. The 'Objects' tab is active, showing a list of 14 objects. The objects are named 01.txt through 07.txt, all of type 'txt'. The 'Last modified' column shows dates from July 13, 2021, at 07:29:44 (UTC-03:00) to 07:29:46 (UTC-03:00). The 'Size' column shows file sizes ranging from 1.3 KB to 4.8 KB. The 'Storage class' column shows all objects are stored in the 'Standard' class. The interface includes a left-hand navigation menu with options like Buckets, Access Points, Batch Operations, Storage Lens, Dashboards, and AWS Organizations settings. The top of the console shows the AWS logo, a search bar, and the user's profile information.

<input type="checkbox"/>	Name	Type	Last modified	Size	Storage class
<input type="checkbox"/>	01.txt	txt	July 13, 2021, 07:29:44 (UTC-03:00)	3.9 KB	Standard
<input type="checkbox"/>	02.txt	txt	July 13, 2021, 07:29:44 (UTC-03:00)	2.2 KB	Standard
<input type="checkbox"/>	03.txt	txt	July 13, 2021, 07:29:44 (UTC-03:00)	1.3 KB	Standard
<input type="checkbox"/>	04.txt	txt	July 13, 2021, 07:29:45 (UTC-03:00)	2.5 KB	Standard
<input type="checkbox"/>	05.txt	txt	July 13, 2021, 07:29:45 (UTC-03:00)	4.8 KB	Standard
<input type="checkbox"/>	06.txt	txt	July 13, 2021, 07:29:45 (UTC-03:00)	3.8 KB	Standard
<input type="checkbox"/>	07.txt	txt	July 13, 2021, 07:29:46 (UTC-03:00)	1.7 KB	Standard

Figure 4: successful file uploading

The screen shot below shows that the file uploading is successfully done and the bucket has been kept as public shown in figure (5).

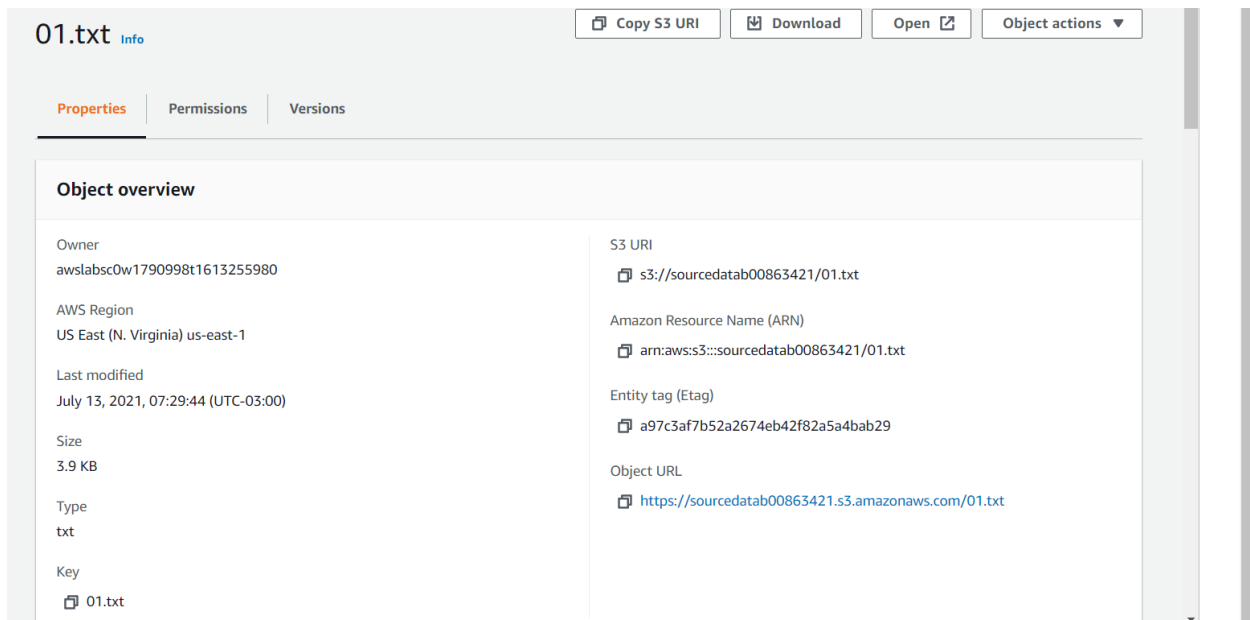


Figure 5: "1.txt" successfully uploading

Next step is to create traindatab00863421 bucket to upload the trainVector.csv file which contains Levenshtein distance and words.

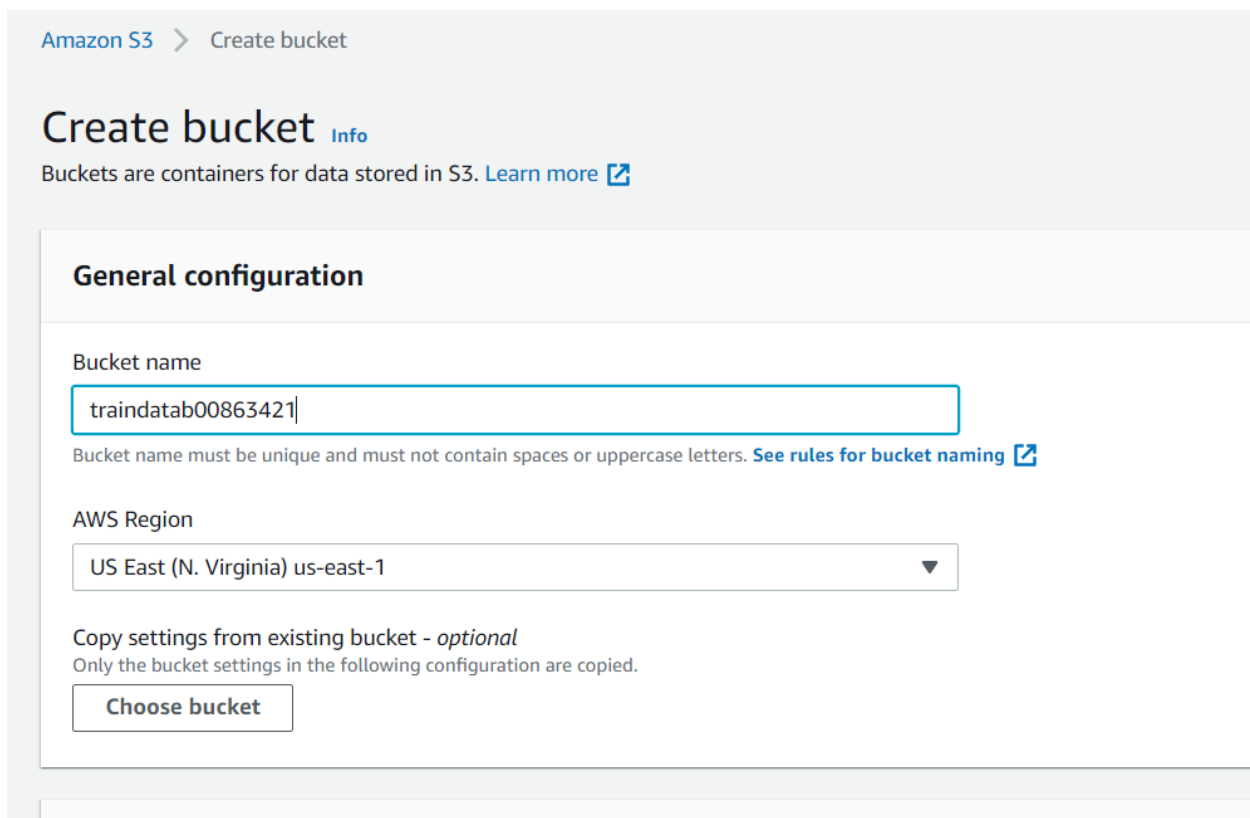
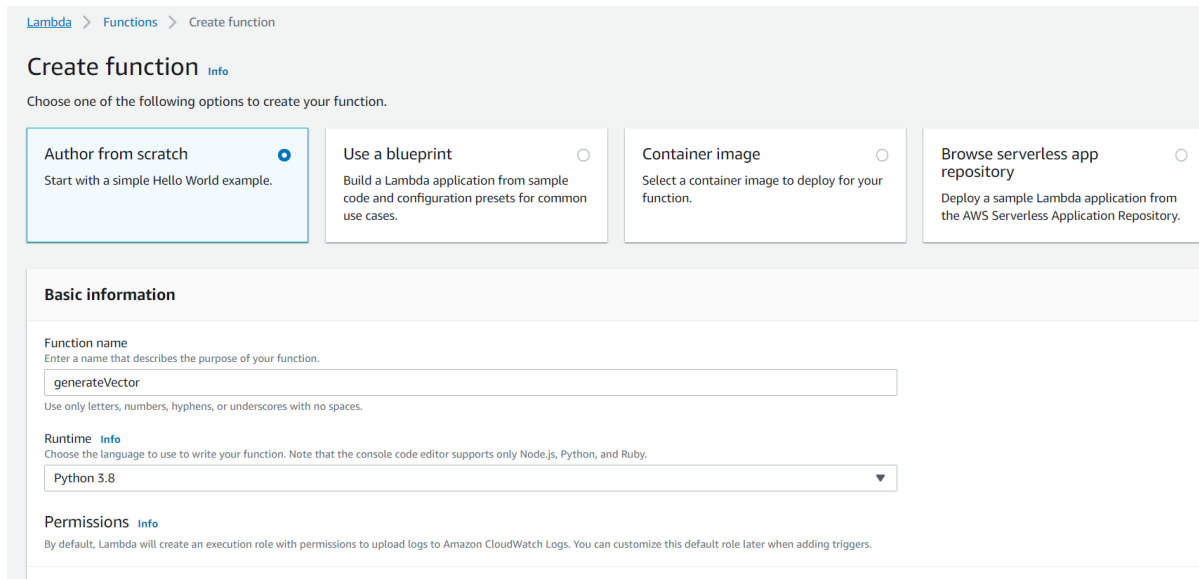


Figure 6: bucket creation - traindatab00863421

To generate the csv file and run the python code, I have created on lambda function named generateVector shown in figure (7) and figure (8) says that train data bucket is connected:



Lambda > Functions > Create function

### Create function [Info](#)

Choose one of the following options to create your function.

**Author from scratch** ☒  
Start with a simple Hello World example.

**Use a blueprint** ☐  
Build a Lambda application from sample code and configuration presets for common use cases.

**Container image** ☐  
Select a container image to deploy for your function.

**Browse serverless app repository** ☐  
Deploy a sample Lambda application from the AWS Serverless Application Repository.

---

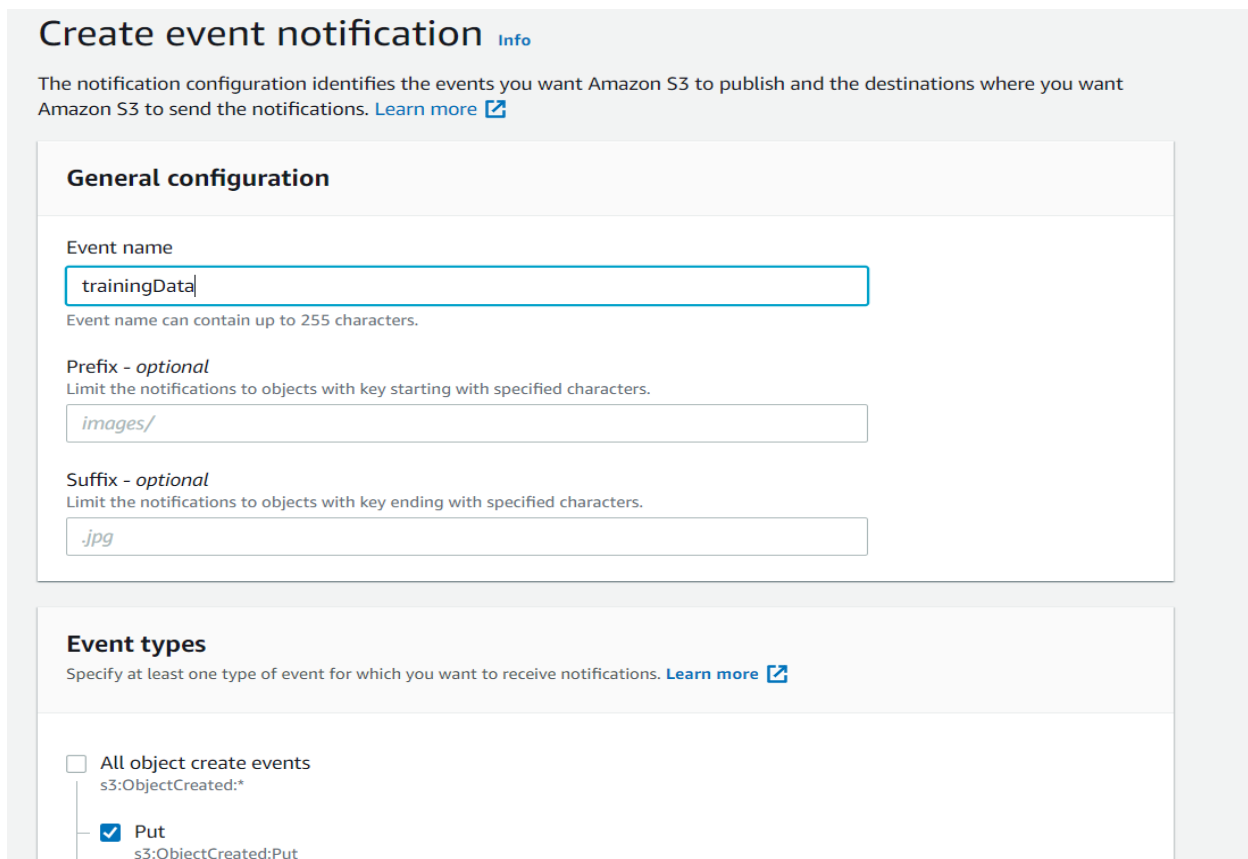
#### Basic information

**Function name**  
Enter a name that describes the purpose of your function.  
  
Use only letters, numbers, hyphens, or underscores with no spaces.

**Runtime** [Info](#)  
Choose the language to use to write your function. Note that the console code editor supports only Node.js, Python, and Ruby.

**Permissions** [Info](#)  
By default, Lambda will create an execution role with permissions to upload logs to Amazon CloudWatch Logs. You can customize this default role later when adding triggers.

Figure 7: generate Vector AWS lambda



### Create event notification [Info](#)

The notification configuration identifies the events you want Amazon S3 to publish and the destinations where you want Amazon S3 to send the notifications. [Learn more](#)

#### General configuration

**Event name**  
  
Event name can contain up to 255 characters.

**Prefix - optional**  
Limit the notifications to objects with key starting with specified characters.

**Suffix - optional**  
Limit the notifications to objects with key ending with specified characters.

#### Event types

Specify at least one type of event for which you want to receive notifications. [Learn more](#)

- ☐ All object create events  
s3:ObjectCreated:\*
- ☒ Put  
s3:ObjectCreated:Put

Figure 8: Connection S3 bucket to lambda function

## Destination

Before Amazon S3 can publish messages to a destination, you must grant the Amazon S3 principal the necessary permissions to call the relevant API to publish messages to an SNS topic, an SQS queue, or a Lambda function.
 [Learn more](#)

### Destination

Choose a destination to publish the event. [Learn more](#)

☒ **Lambda function**  
 Run a Lambda function script based on S3 events.

☐ **SNS topic**  
 Send notifications to email, SMS, or an HTTP endpoint.

☐ **SQS queue**  
 Send notifications to an SQS queue to be read by a server.

### Specify Lambda function

☒ Choose from your Lambda functions
 ☐ Enter Lambda function ARN

### Lambda function

generateVector

Cancel

Save changes

Figure 9: the trigger provided to s3 by stating the destination

Event notifications (1)					
Send a notification when specific events occur in your bucket. <a href="#">Learn more</a>					
	Name	Event types	Filters	Destination type	Destination
<input type="checkbox"/>	trainingData	Put	-	Lambda function	<a href="#">generateVector</a>

Figure 10: event notification trigger

I have generated policy as a next step for both the buckets: Permissions – generate policy, Type of policy: s3 bucket, Principal: \*, AWS service: all as shown in below figure 11:



## AWS Policy Generator

The AWS Policy Generator is a tool that enables you to create policies that control access to Amazon Web Services (AWS) products and resources. For more information about creating policies, see [key concepts in Using AWS Identity and Access Management](#). Here are [sample policies](#).

### Step 1: Select Policy Type

A Policy is a container for permissions. The different types of policies you can create are an [IAM Policy](#), an [S3 Bucket Policy](#), an [SNS Topic Policy](#), a [VPC Endpoint Policy](#), and an [SQS Queue Policy](#).

Select Type of Policy S3 Bucket Policy

### Step 2: Add Statement(s)

A statement is the formal description of a single permission. See [a description of elements](#) that you can use in statements.

Effect ☒ Allow ☐ Deny

Principal

Use a comma to separate multiple values.

AWS Service Amazon S3 ☐ All Services (\*\*)

Use multiple statements to add permissions for more than one service.

Actions -- Select Actions -- ☒ All Actions (\*\*)

Amazon Resource Name (ARN) arn:aws:s3:::traindataab0086

ARN should follow the following format: arn:aws:s3:::<bucket\_name>/<key\_name>.  
Use a comma to separate multiple values.

[Add Conditions \(Optional\)](#)

[Add Statement](#)

Figure 11: AWS policy generator


## Bucket policy

The bucket policy, written in JSON, provides access to the objects stored in the bucket. Bucket policies don't apply to objects owned by other accounts. [Learn more](#)

[Policy examples](#)

[Policy generator](#)

Bucket ARN

 arn:aws:s3:::traindataab00863421

### Policy

```
1 {  
2   "Id": "Policy1626174147240",  
3   "Version": "2012-10-17",  
4   "Statement": [  
5     {  
6       "Sid": "Stmt1626174144903",  
7       "Action": "s3:*",  
8       "Effect": "Allow",  
9       "Resource": "arn:aws:s3:::traindataab00863421",  
10      "Principal": "*"   
11    }  
12  ]  
13 }
```

Figure 12: provided json format into bucket policy

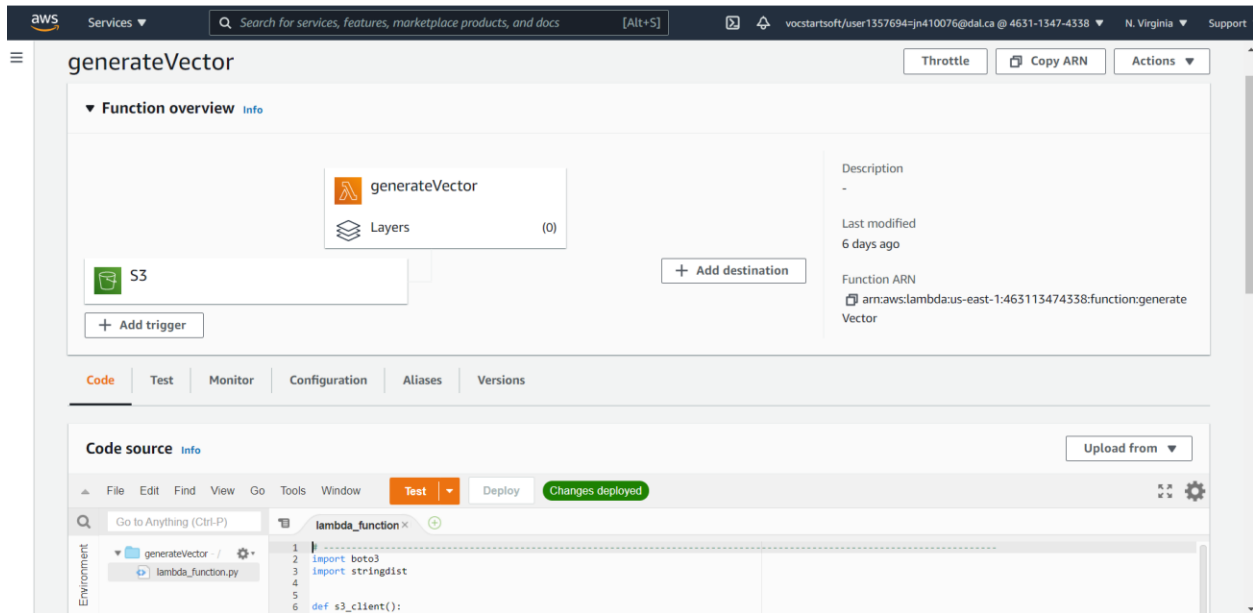


Figure 13: Connected to S3 and uploaded code to lambda function

Below are the next steps, I have followed shown in figure 14 to 18:

- 1) S3 client connection.
- 2) Read the uploaded files from sourcedatab00863421 bucket.
- 3) Append the file content.
- 4) Define stop words list.
- 5) Remove the stop words.
- 6) Convert the content into lower case.
- 7) Calculate the levenshtein distance generation.
- 8) Once the distance is calculated, the current word, next word and distance will be saved into csv file.
- 9) The files 1-299 will be stored into trainVector.csv and files from 300-401 will be stored into testVector.csv

```

1  # -----
2  import boto3
3  import stringdist
4
5
6  def s3_client():
7      """
8          Function: get s3 client
9          Purpose: get s3 client
10         :returns: s3
11         """
12     session = boto3.session.Session()
13     client = session.client('s3')
14     """ :type : pyboto3.s3 """
15     return client
16

```

Figure 14: S3 client connection

```

70 def lambda_handler(event, context):
71     s3_bucket_name = "sourcedatab00863421"
72     s3 = boto3.resource('s3')
73     obj = s3.Bucket(s3_bucket_name)
74     # Iterates through all the objects, doing the pagination for you. Each obj
75     # is an ObjectSummary, so it doesn't contain the body. You'll need to call
76     # get to get the whole body.
77     filesContent = ""
78     for obj in obj.objects.all():
79         body = obj.get()['Body'].read()
80         content = body.decode('utf-8')
81         filesContent = filesContent + content
82     removeStopWords(filesContent)
83     return {
84         'statusCode': 200,
85         'body': json.dumps('Hello from Lambda!')}
86     }
87

```

Figure 15: lambda handler to generate the distance

```

40 def removeStopWords(content):
41     # https://www.geeksforgeeks.org/removing-stop-words-nltk-python/
42     # https://www.tutorialspoint.com/python_text_processing/python_remove_stopwords.htm
43     stopWords = ['ourselves', 'hers', 'between', 'yourself', 'but', 'again', 'there', 'about', 'once',
44                 'during', 'out', 'very', 'having', 'with', 'they', 'own', 'an', 'be', 'some',
45                 'for', 'do', 'its', 'yours', 'such', 'into', 'of', 'most', 'itself', 'other',
46                 'off', 'is', 's', 'am', 'or', 'who', 'as', 'from', 'him', 'each', 'the', 'themselves',
47                 'until', 'below', 'are', 'we', 'these', 'your', 'his', 'through', 'don', 'nor', 'me',
48                 'were', 'her', 'more', 'himself', 'this', 'down', 'should', 'our', 'their', 'while',
49                 'above', 'both', 'up', 'to', 'ours', 'had', 'she', 'all', 'no', 'when', 'at', 'any',
50                 'before', 'them', 'same', 'and', 'been', 'have', 'in', 'will', 'on', 'does', 'yourselves',
51                 'then', 'that', 'because', 'what', 'over', 'why', 'so', 'can', 'did', 'not', 'now', 'under',
52                 'he', 'you', 'herself', 'has', 'just', 'where', 'too', 'only', 'myself', 'which', 'those', 'i',
53                 'after', 'few', 'whom', 't', 'being', 'if', 'theirs', 'my', 'against', 'a', 'by', 'doing', 'it',
54                 'how', 'further', 'was', 'here', 'than', 'your', 'yours', 'yourself', 'yourselves', 'he',
55                 'him', 'his', 'himself', 'she',
56                 'she's', 'her', 'hers', 'herself', 'it', 'it's', 'its', 'itself', 'they', 'them',
57                 'their', 'theirs', 'themselves', 'what', 'which', 'who', 'whom', 'this',
58                 'that', 'that'll', 'these', 'those', 'am', 'is', 'are', 'was', 'were', 'be',
59                 'been', 'being', 'have', 'has', 'had', 'having', 'do', 'does', 'did', 'doing',
60                 'a', 'an', 'the', 'and', 'but', 'if', 'or', 'because', 'as', 'until',
61                 'while', 'of', 'at']
62     content = content.lower()
63     words = content.split()
64     filteredWords = [word for word in words if not word in stopWords]
65     levenshtein_distance_gen(filteredWords)

```

Figure 16: filtering the stop words

```
def levenshtein_distance_gen(filteredWords):
    listContent = ""
    # find out levenshtein distance between current and next word
    for i in range(len(filteredWords) - 1):
        currentWord, nextWord = filteredWords[i], filteredWords[i + 1]
        count = stringdist.levenshtein(currentWord, nextWord)
        body = currentWord + "," + nextWord + "," + str(count)
        listContent = listContent + '\n' + body
    headers = "Current_Word" + "," + "Next_Word" + "," + "Levenshtein_distance" + '\n'
    distance = headers + listContent
    print(distance[0:120])
    writeContent(distance)
```

Figure 17: levenshtein distance generation on content

```
18 def writeContent(distance):
19     # push the content in the csv file
20     s3 = boto3.resource('s3')
21     bucketName = "traindataab00863421"
22     fileName = "trainVector.csv"
23     s3.Object(bucketName, fileName).put(Body=distance)
24     print('Successfully generation of trainVector.csv on bucket named - traindataab00863421')
25
```

Figure 18: Write the content to S3 bucket

Below is the screen shot for train data b00863421 bucket csv file uploadation:

Amazon S3 > traindataab00863421

## traindataab00863421 [Info](#)

**Objects** | Properties | Permissions | Metrics | Management | Access Points

**Objects (1)**  
Objects are the fundamental entities stored in Amazon S3. You can use [Amazon S3 inventory](#) to get a list of all objects in your bucket. For others to access your objects, you'll need to explicitly grant them permissions. [Learn more](#)

Find objects by prefix

<input type="checkbox"/>	Name	Type	Last modified	Size	Storage class
<input type="checkbox"/>	trainVector.csv	csv	July 13, 2021, 07:42:40 (UTC-03:00)	443.2 KB	Standard

Figure 19: trainVector.csv successful uploading

These figures showcase output of train Vector successfully generation code. CMD prints the messages as shown in figure 20 and the train Vector csv is generated as shown in figure 21.

```

Run: generateVector
C:\Users\Janvi.DESKTOP-P36UV26\PycharmProjects\pythonProject\venv\Scripts\python.exe C:/Users/Janvi.DESKTOP-P36UV26/PycharmProjects/pythonProject/generateVector.py
Current_Word,Next_Word,Levenshtein_distance
ink,helps,5
helps,drive,5
drive,democracy,7
democracy,asia,8
asia,kyrgyz,6

Successfully generation of trainVector.csv on bucket named - traindata08863421

Process finished with exit code 0

```

Figure 20: CMD message of successful uploading

	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O
1	Current_W	Next_Wor	Levenshtein_distance												
2															
3	ink	helps	5												
4	helps	drive	5												
5	drive	democracy	7												
6	democracy	asia	8												
7	asia	kyrgyz	6												
8	kyrgyz	republic	9												
9	republic	small	7												
10	small	mountainc	10												
11	mountainc	state	9												
12	state	former	5												
13	former	soviet	4												
14	soviet	republic	8												
15	republic	using	7												
16	using	invisible	7												
17	invisible	ink	7												
18	ink	ultraviolet	10												
19	ultraviolet	readers	10												
20	readers	country's	7												
21	country's	elections	7												
22	elections	part	8												
23	part	drive	5												
24	drive	prevent	4												
25	prevent	multiple	8												
26	multiple	voting.	6												
27	voting.	new	6												
28	new	technology	9												
29	technology	causing	8												

Figure 21: training csv file content

Same way, I have repeated the process for test dataset which contains distance from file 300 to 401. Below is the screen shot for test data bucket:

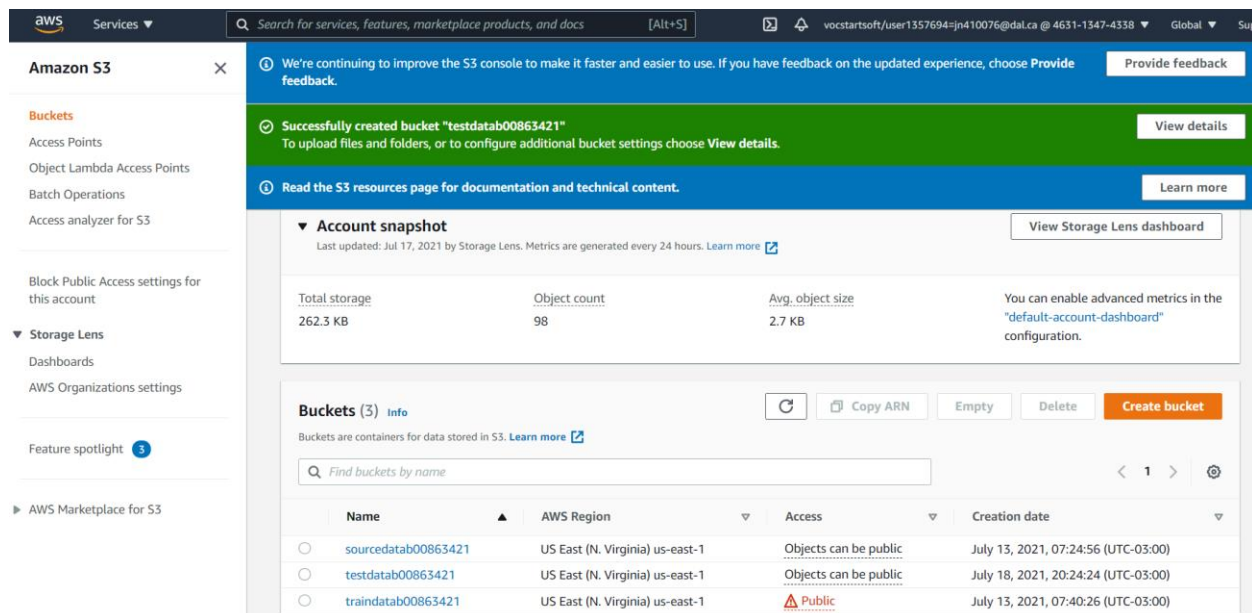


Figure 22: test data s3 bucket successful creation

When, I have computed the previous same code for test data I have uploaded test Vector csv file.

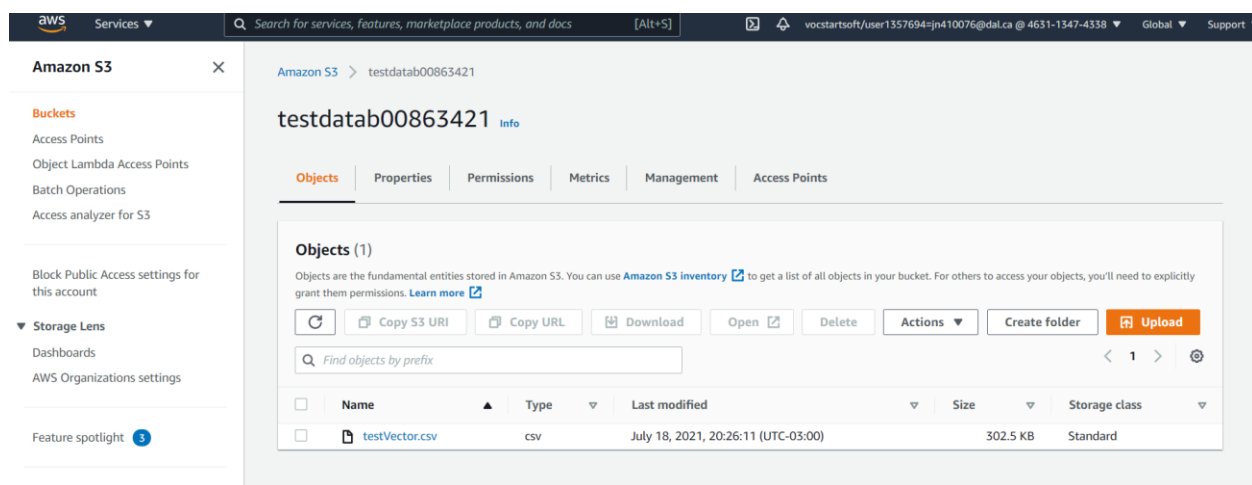


Figure 23: test Vector.csv uploading successful

To generate and upload the csv file to buckets I have created on lambda function named 'kmeanClustering' and connected the lambda function to 2 S3 bucket. As shown in figure 24,25.

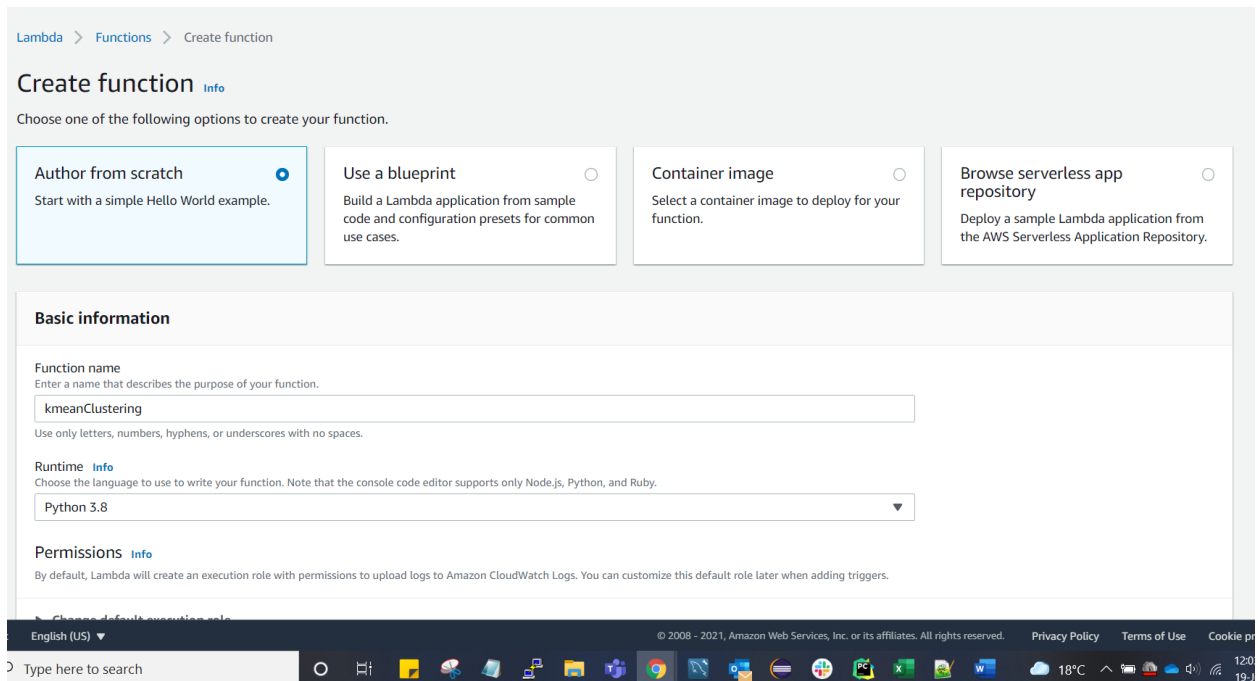


Figure 24: kmean clustering lambda function

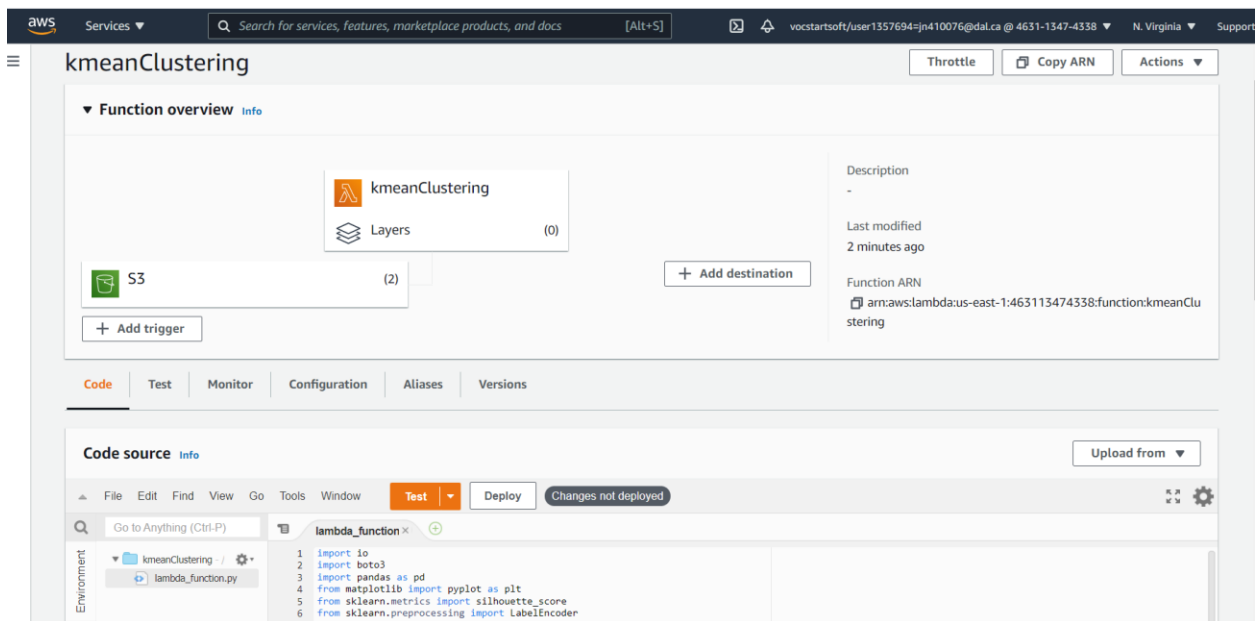


Figure 25: S3 connection successful

Below are the steps I have followed to code kmean clustering:

1. Cleaning the data

2. Label encoder on categorical data.
3. Model training for kmean clustering which contains a feature that represents cluster no from 2 to 10 which will automatically be selected based on the silhouette score using wcss.
4. Once the model is generated, it will be stored and is used to predict the labels.
5. Cluster is evaluated using silhouette score and cluster centroid.
6. After that, cluster will be displayed as PC1 vs PC2 format.
7. Once the model has been generated, the test data will be fitted in model generated.
8. The process from step 4,5,6 will be repeated.

Code for kmean cluster []:

```
uploadFiles.py x kmeanCluster.py x twitterSentimentAnalysis.py x generateVector.py x
1 import io
2 import boto3
3 import pandas as pd
4 from matplotlib import pyplot as plt
5 from sklearn.metrics import silhouette_score
6 from sklearn.preprocessing import LabelEncoder
7 from sklearn.cluster import KMeans
8 import numpy as np
9 from sklearn.decomposition import PCA
10
11
12 # References: #https://towardsdatascience.com/finding-and-removing-duplicate-rows-in-pandas
13 # https://towardsdatascience.com/categorical-encoding-using-label-encoding-and-one-hot-encoder
14 # https://jakevdp.github.io/PythonDataScienceHandbook/05.11-k-means.html
15 # https://stackoverflow.com/questions/55291667/getting-typeerror-slicenone-none-none-0-is-an-i
16 # https://dzone.com/articles/kmeans-silhouette-score-explained-with-python-exam
17 # https://matplotlib.org/stable/tutorials/colors/colors.html
18
19
20 def clean_data(X):
21     # record columns to delete
22     X.dropna()
23
24     # duplicate rows removal
25     X.drop_duplicates(inplace=True)
26     return X
27
```

Figure 26: data cleaning



```

28
29 def label_encoder(X):
30     # creating instance of label encoder
31     labelencoder = LabelEncoder()
32     # Assigning numerical values and storing in another column
33     X['Current_Word'] = labelencoder.fit_transform(X['Current_Word'])
34     X['Next_Word'] = labelencoder.fit_transform(X['Next_Word'])
35     return X
36
37
38 def model_train(train_data):
39     # https://towardsdatascience.com/machine-learning-algorithms-part-9-k-means-example-in-python-f2ad05ed5203#:~:text=K%2DMeans%20
40     wcss = []
41     for i in range(2, 10):
42         kmeans = KMeans(n_clusters=i, init='k-means++', max_iter=300, n_init=10, random_state=0)
43         kmeans.fit(train_data)
44         wcss.append(kmeans.inertia_)
45     return kmeans
46
47
48 def cluster_prediction(data, kmeans):
49     # Y label
50     clusterNo = kmeans.predict(data)
51     data = pd.DataFrame(data)
52     data['clusterNo'] = pd.DataFrame(clusterNo)
53     data = data.replace(np.nan, 0)
54     return data

```

Figure 27: model training and cluster prediction

```

57 def cluster_evaluation(data, kmeans):
58     # Silhouette Score calculation
59     score = silhouette_score(data, kmeans.labels_, metric='euclidean')
60     print('Silhouette Score: %.3f' % score)
61
62     # Centroid cluster
63     cent = kmeans.cluster_centers_
64     print('Centroid of Cluster', cent)
65
66
67 def display_clusters(data, train):
68     # Plotting the cluster
69     clusterCount = data['clusterNo'].nunique()
70     colors = ['#1f77b4', '#ff7f0e', '#2ca02c', '#d62728', '#9467bd', '#8c564b', '#e377c2', '#7f7f7f', '#bcbd22',
71             '#ffff99', '#4b0082']
72     for i in range(0, clusterCount):
73         plt.scatter(data[data['clusterNo'] == i].iloc[:, 0], data[data['clusterNo'] == i].iloc[:, 1], color=colors[i])
74     plt.xlabel('PC 1')
75     plt.ylabel('PC 2')
76     if (train == True):
77         plt.title('Cluster plot on training data')
78     else:
79         plt.title('Cluster plot on testing data')
80
81     plt.show()
82
83

```

Figure 28: cluster evaluation and display cluster logic

Lambda handler for training data and testing data are shown in figure 29 and figure 30.

```

83 def lambda_handler(event, context):
84
85     s3_bucket_name = "traindata00863421"
86     filename = "trainVector.csv"
87     s3 = boto3.client('s3')
88     train_data = s3.get_object(Bucket=s3_bucket_name, Key=filename)
89     train_data = train_data['Body'].read()
90     train_data = pd.read_csv(io.BytesIO(train_data), header=0, delimiter=",", low_memory=False)
91     train_data = clean_data(train_data)
92     train_data = label_encoder(train_data)
93     pca = PCA(2)
94     train_data = pca.fit_transform(train_data)
95     kmeans = model_train(train_data)
96     train_data = cluster_prediction(train_data, kmeans)
97     print(train_data)
98     cluster_evaluation(train_data, kmeans)
99     display_clusters(train_data, True)

```

Figure 29: lambda handler for training data

```

100
101     s3_bucket_name = "testdata00863421"
102     filename = "testVector.csv"
103     test_data = s3.get_object(Bucket=s3_bucket_name, Key=filename)
104     test_data = test_data['Body'].read()
105     test_data = pd.read_csv(io.BytesIO(test_data), header=0, delimiter=",", low_memory=False)
106     test_data = clean_data(test_data)
107     test_data = label_encoder(test_data)
108     pca = PCA(2)
109     test_data = pca.fit_transform(test_data)
110     test_data = cluster_prediction(test_data, kmeans)
111     print(test_data)
112     display_clusters(test_data, False)
113     # TODO implement
114     return {
115         'statusCode': 200,
116         'body': json.dumps('Hello from Lambda!')
117     }

```

Figure 30: lambda handler for testing data

Here, I have applied PCA on the dataset.

The Principal Component Analysis (PCA) is a **linear dimensionality reduction** technique that can be utilized for extracting information from a high-dimensional space by projecting it into a lower-dimensional sub-space. It tries to preserve the essential parts that have more variation of the data and remove the non-essential parts with fewer variation. [16]

## AWS Sage Maker Creation:

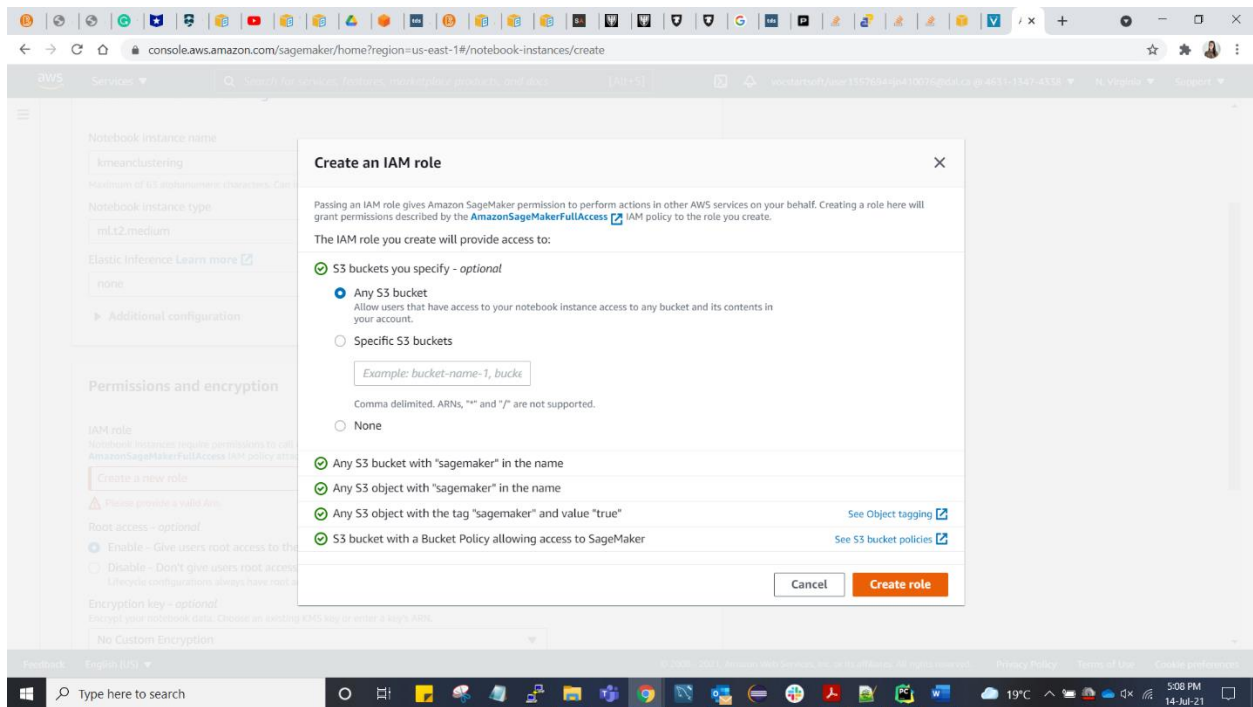


Figure 31: Creating IAM role for Sage Maker

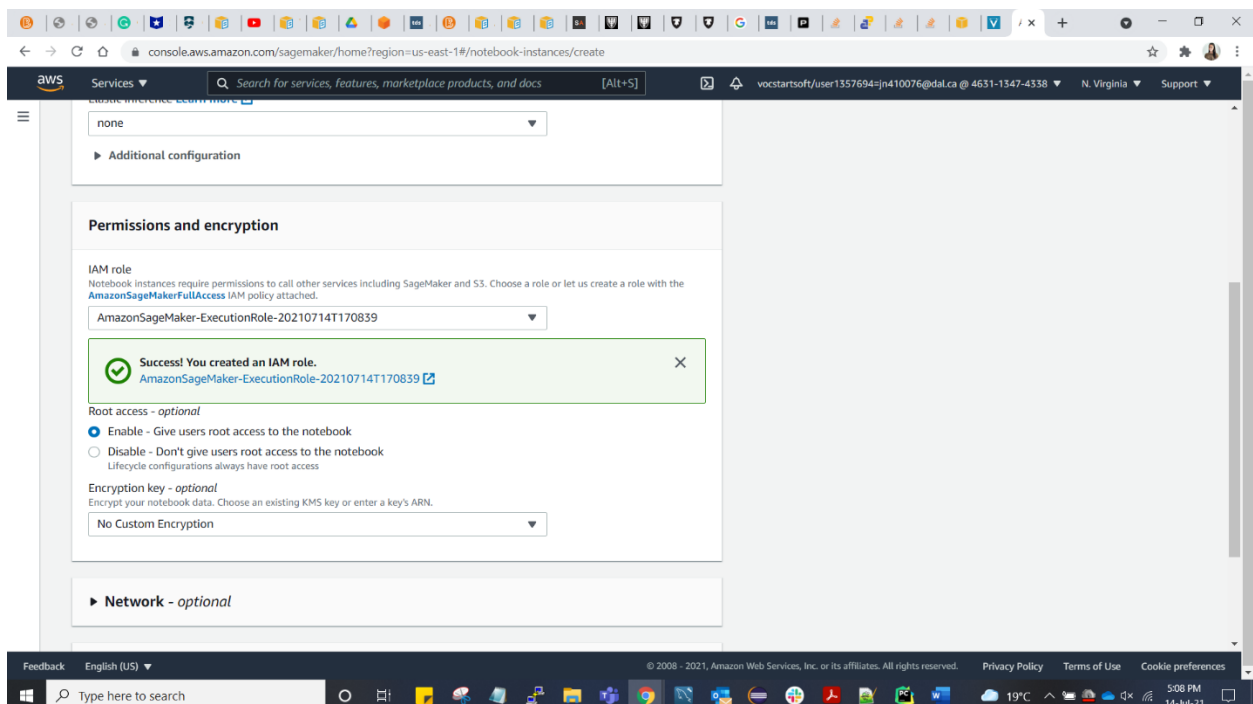


Figure 32: Permissions of Sage Maker

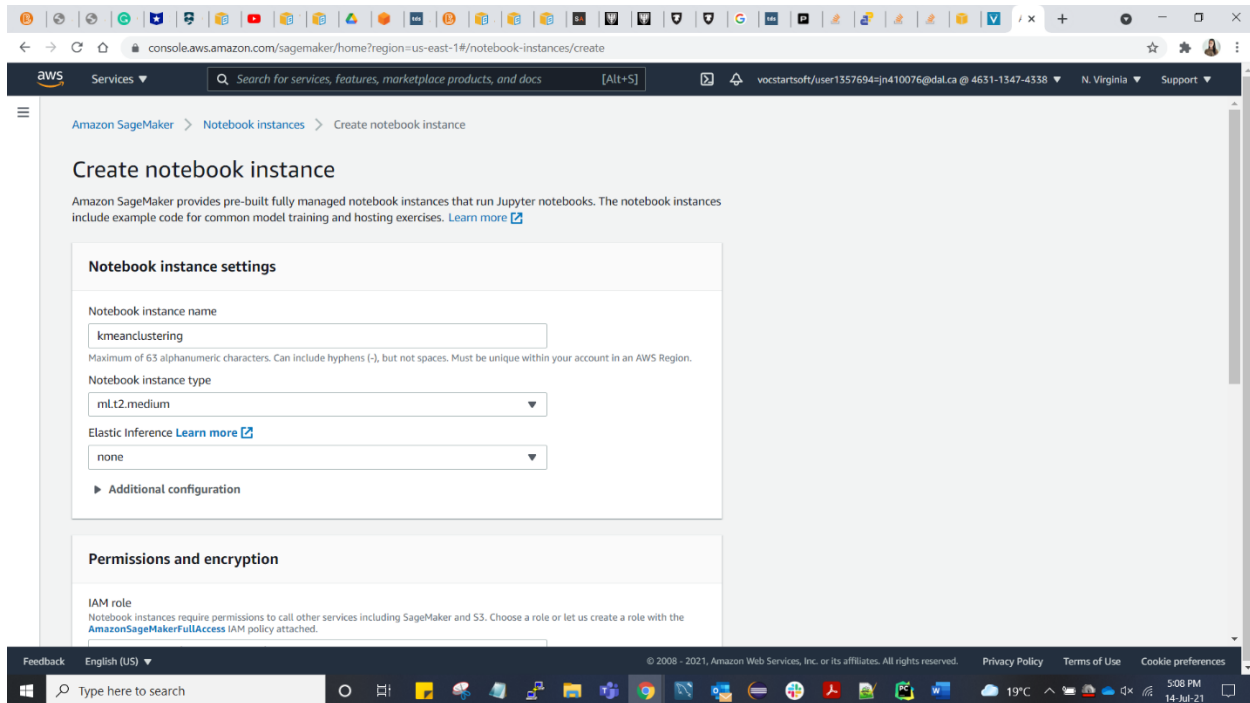


Figure 33: Notebook instance name for Sage Maker

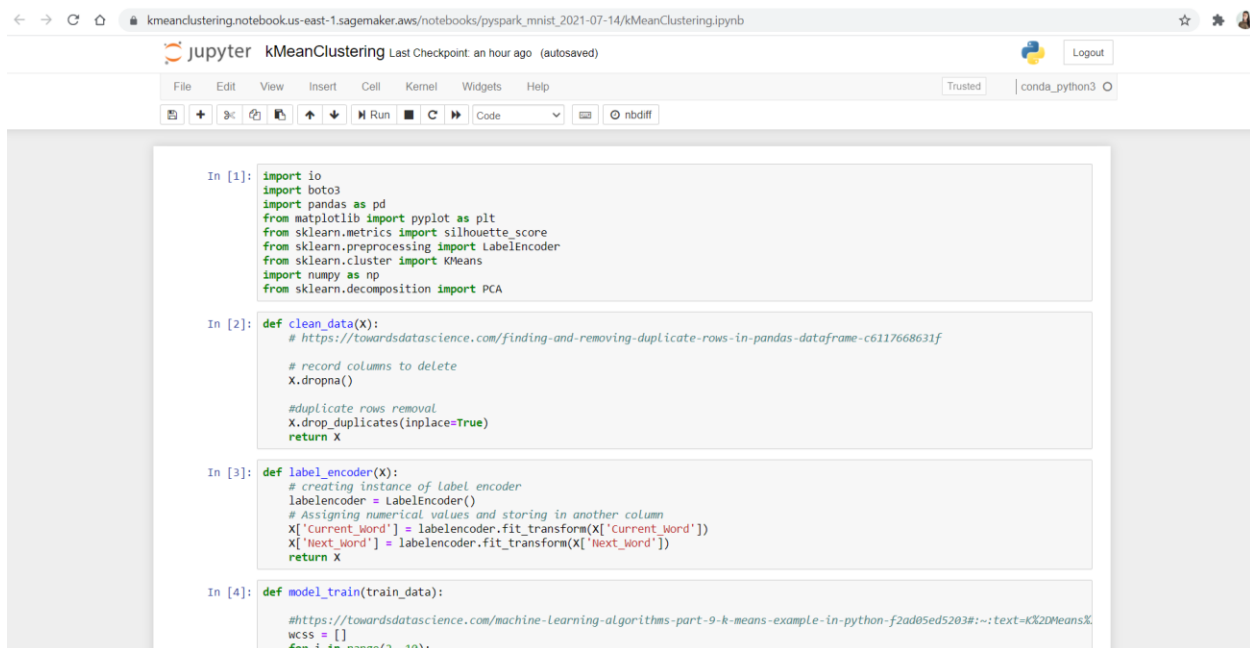


Figure 34: jupyter notebook for kmean clustering code

## Output:

Below screen shots [35 to 37] represents for training and testing data:

- 1) PC0 and PC1 and cluster No
- 2) Silhouette score
- 3) Centroid of cluster
- 4) Clusters

	0	1	clusterNo
0	780.228738	-282.987346	6
1	1703.273224	-853.339759	1
2	2657.332461	-378.480656	7
3	3688.943999	-1190.090614	7
4	2595.559828	2074.282097	2
...	...	...	...
31927	1782.771035	-1237.705598	1
31928	3357.790859	-669.461471	7
31929	2109.979909	1702.049383	2
31930	1582.988896	-1709.184147	1
31931	-3831.378874	1298.093039	3

[31932 rows x 3 columns]

Silhouette Score: 0.376

Centroid of Cluster:

```
[[-1614.75362371 1972.13759455]
 [ 1617.83779753 -1887.49129485]
 [ 2017.77539224 1643.5671693 ]
 [-3449.03531013 282.72039174]
 [-1846.46224294 -1684.70517695]
 [ 398.47519148 3561.47671167]
 [-131.03025207 86.70192261]
 [ 3762.35285827 -326.90068724]
 [-153.89411565 -3604.55610328]]
```

Figure 35: Cluster no on training data, cluster evaluation

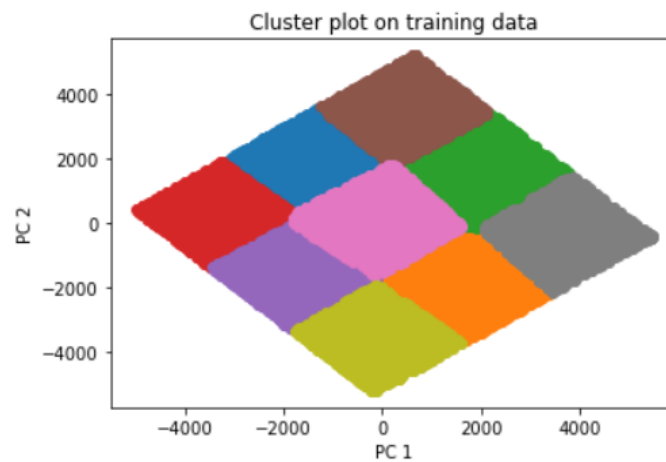


Figure 36: Cluster plot for training data

	0	1	clusterNo
0	1008.097342	-2538.316639	1
1	3269.229581	340.217124	7
2	1404.866292	1672.865057	2
3	-248.628428	102.240442	6
4	1105.148407	-1352.727269	1
...	...	...	...
16550	34.591623	2988.377449	5
16551	-2899.157668	-45.811320	3
16552	-2293.815072	-755.924884	4
16553	-2351.520767	552.364860	3
16554	-3137.604844	16.574127	3

[16555 rows x 3 columns]



Figure 37: Output for testing data

Part C. Build an event-driven serverless application using AWS Comprehend.

In this part of the assignment, you need to use S3 bucket, Lambda Functions, and AWS Comprehend.

[B00xxxxxx = your B00 number] used in bucket naming

In this section, I have performed tweet cleaning, sentiment analysis using AWS comprehend and then upload the csv file with the sentiment shown in the next screenshots.

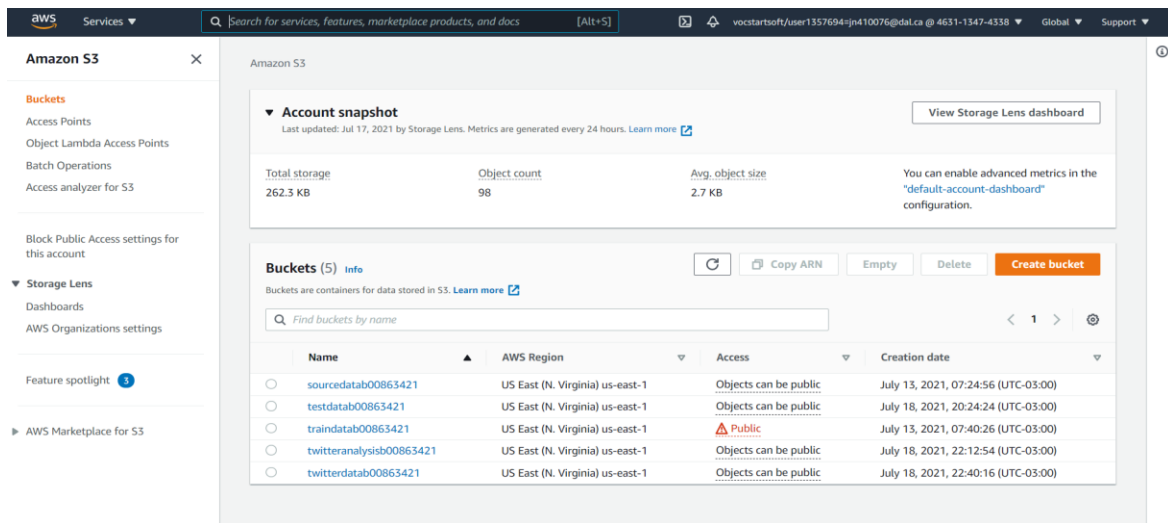


Figure 38: twitter data bucket creation

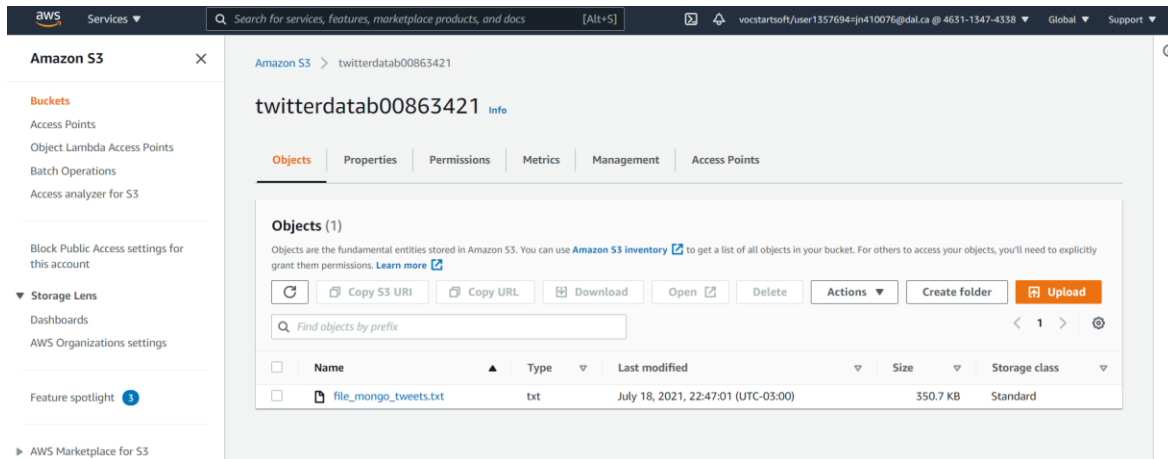


Figure 39: file successfully uploading in the bucket

Code:

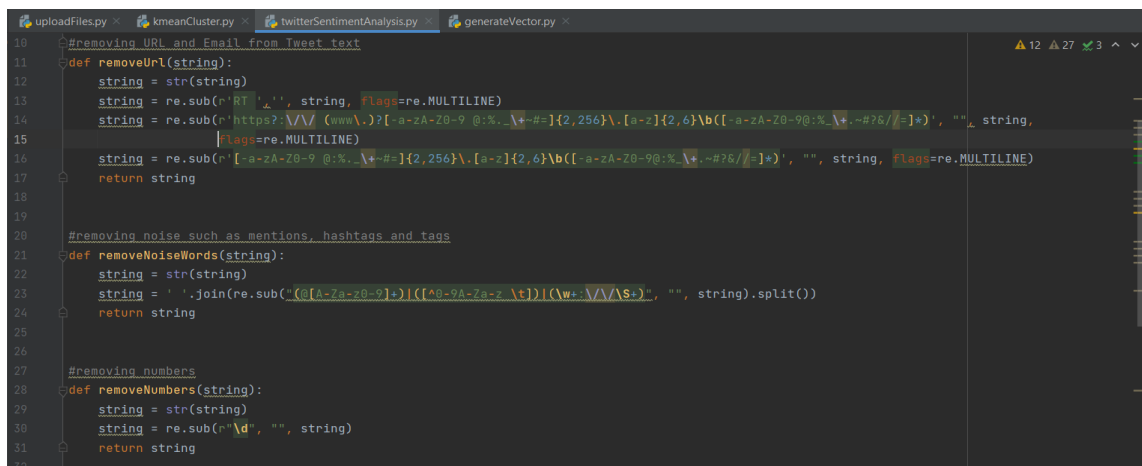


Figure 40: tweet cleaning

```

34 def lambda_handler(event, context):
35     s3 = boto3.client('s3')
36
37     sourceBucketName = "twitterdata00863421"
38     fileName = "file_mongo_tweets.txt"
39     targetBucketName = "twitteranalysis00863421"
40     targetFileName = "twitteranalysis00863421.csv"
41
42     bucketObject = s3.get_object(Bucket=sourceBucketName, Key=fileName)
43     fileContent = bucketObject['Body'].read().decode('utf-8')
44     fileLines = fileContent.split('\n')
45     comprehendObject = boto3.client('comprehend', region_name='us-east-1')
46
47     cleanTweets = []
48     output = 'Tweet, Sentiment, Positive Sentiment Score, Negative Sentiment Score, Neutral Sentiment Score, Mixed Sentiment Score'
49
50     for line in fileLines:
51         line = line.translate(str.maketrans('', '', string.punctuation))
52         cleanLine = removeUrl(line)
53         cleanLine = removeNumbers(cleanLine)
54         cleanLine = removeNoiseWords(cleanLine)
55         cleanTweets.append(cleanLine)
56
57     for (index, item) in enumerate(cleanTweets):
58         # So as not to exhaust credits of AWS.
59         if index <= 9 and item != '':
60             result = comprehendObject.detect_sentiment(Text=item, LanguageCode='en')
61             finalOutput = finalOutput + '\n' + item + ',' + result['Sentiment'] + ',' + str(
62                 result['SentimentScore']['Positive']) + ',' + str(result['SentimentScore']['Negative']) + ',' + str(

```

Figure 41: Reading tweet text file from S3 bucket

The image above represent the code for comprehend where I am performing the sentiment analysis.

```

63         result['SentimentScore']['Neutral']) + ',' + str(result['SentimentScore']['Mixed'])
64
65     print(finalOutput)
66     s3.put_object(Bucket=targetBucketName, Key=str(targetFileName), Body=finalOutput)
67
68     return {
69         'statusCode': 200,
70         'body': json.dumps('Hello from Lambda!')}
71

```

Figure 42: final output into S3 bucket

```

un: twitterSentimentAnalysis
C:\Users\Janvi.DESKTOP-P36UV26\PycharmProjects\pythonProject\venv\Scripts\python.exe C:/Users/Janvi.DESKTOP-P36UV26/PycharmProjects/pythonProject/twitterSentimentAnalysis.py
Tweet, Sentiment, Positive Sentiment Score, Negative Sentiment Score, Neutral Sentiment Score, Mixed Sentiment Score
RT LunionSuite PortauPrince fans takes to the streets to celebrate Haiti's Quarterfinal Gold Cup win against Canada ,NEUTRAL,0.3054550290107727,0.0014724412467330694,0.69298660755
Canada thank you for a successful StrokeMonth,POSITIVE,0.973151624027283,0.011226603761315346,0.015451391227543354,0.000170334373251535
Thanks to FAST more people in Canada are recognizing the signs of ,POSITIVE,0.6671866774559021,0.025488805025815964,0.3002881705760956,0.007036389317363501
Authorities say they plan to return the artifacts in the near future and are working to determine who is criminally ,NEUTRAL,0.016546757891774178,0.2438882738351022,0.62051200866
Many families travel to a cottage or lake house to enjoy the Canada Day long weekend If you have a First Aid Kit a ,NEUTRAL,0.3782193958759308,0.003102522809058428,0.617648839950
Arnprior is setting fireworks off as they do each year from the island below the bridge at Hydro Park,NEUTRAL,0.09714554995298386,0.2243763655424118,0.6749534606933594,0.00352459
Canada Day Open Closed Fireworks,NEUTRAL,0.039773814380168915,0.03642244264483452,0.9237711429595947,3.255877163610421e-05
Barrie Muskoka SimcoeCounty,NEUTRAL,0.0005460968241095543,0.0002114945964422077,0.999239444732666,3.031375172213302e-06
,NEUTRAL,0.012584115378558636,0.004845325369387865,0.9741135239601135,0.00845707952762268
JasonChatfield Canada sucks,NEGATIVE,9.372492058901116e-05,0.9981383085250854,0.0017626808257773519,5.190799129195511e-06
Looking for something to do this Canada Day long weekend Learn more about our planned celebrations events and act ,NEUTRAL,0.40803372859954834,0.00208928226493299,0.5897696614265
We hope everyone is enjoying the long weekend Happy Canada Day Weekend,POSITIVE,0.9423585534095764,0.001981507521122694,0.05480172857642174,0.0008582472801208496
longweekend celebration ,NEUTRAL,0.31997751925468445,0.0065695918165147305,0.5646914839744568,0.10874141752719879

```

Figure 43: Output screen shot into CMD



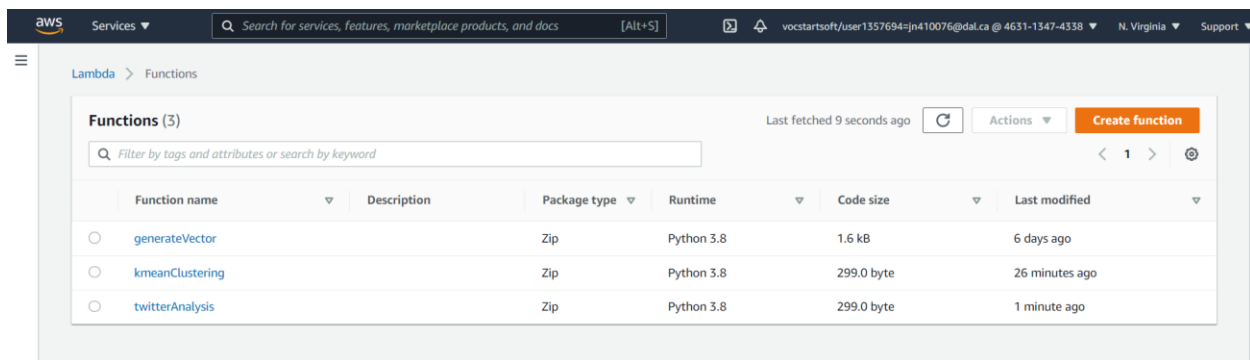


Figure 44: twitter analysis lambda function

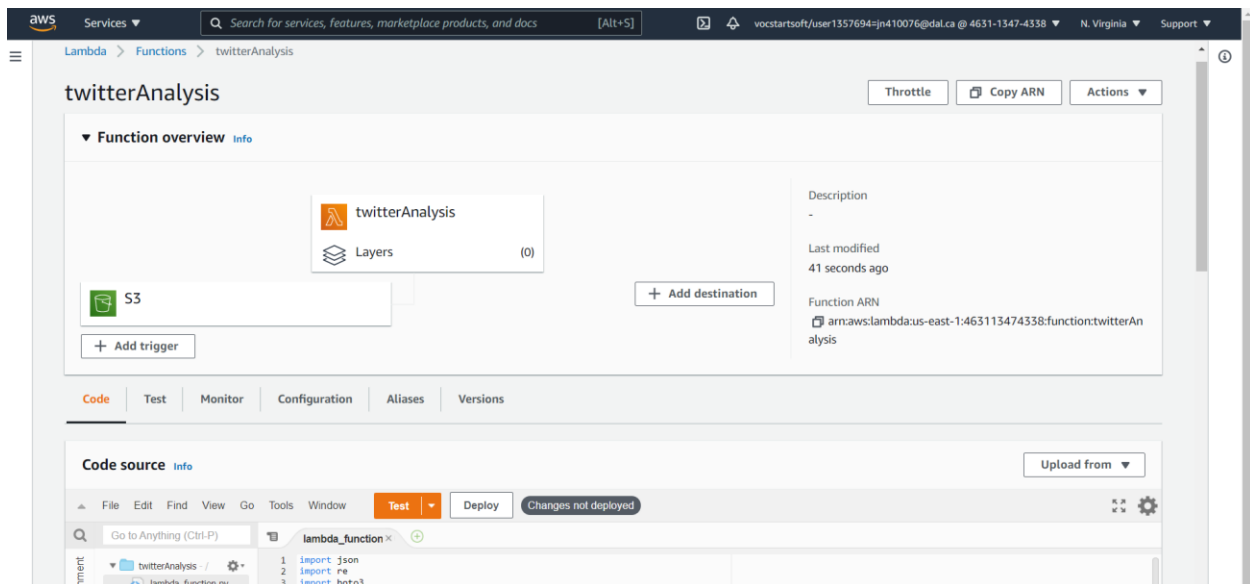


Figure 45: connection with s3 bucket

When we upload the tweet file into S3 bucket, the lambda will be triggered and the output will be store into .csv file which is represented as below with tweet, sentiment, positive sentiment score, negative sentiment score, neutral sentiment score and mixed sentiment score etc.

The screenshot shows an Excel spreadsheet with the following data:

Tweet	Sentiment	Positive Sentiment Score	Negative Sentiment Score	Neutral Sentiment Score	Mixed Sentiment Score
RT UnionSuite PortauPrince fans takes to the streets to celebrate Haitis Quarterfinal Gold Cup win against Canada	NEUTRAL	0.305455029	0.001472441	0.692986608	8.59E-05
Canada thank you for a successful StrokeMonth	POSITIVE	0.973151624	0.011226604	0.015451391	0.000170334
Thanks to FAST more people in Canada are recognizing the signs of	POSITIVE	0.667186677	0.025488805	0.300288171	0.007036389
Authorities say they plan to return the artifacts in the near future and are working to determine who is criminally	NEUTRAL	0.016546758	0.243888274	0.620512009	0.119052969
Many families travel to a cottage or lake house to enjoy the Canada Day long weekend If you have a First Aid Kit a	NEUTRAL	0.378219396	0.003102523	0.617648884	0.001029157
Arnprior is setting fireworks off as they do each year from the island below the bridge at Hydro Park	NEUTRAL	0.097714555	0.224376366	0.674953461	0.003524599
Canada Day Open Closed Fireworks	NEUTRAL	0.039773814	0.036422443	0.923771143	3.26E-05
Barrie Muskoka SimcoeCountv	NEUTRAL	0.000546097	0.000211495	0.999239445	3.03E-06

Figure 46: twitter analysis output

## References:

- [1] "What Is Amazon SageMaker? - Amazon SageMaker", *Docs.aws.amazon.com*, 2021. [Online]. Available: <https://docs.aws.amazon.com/sagemaker/latest/dg/whatis.html>. [Accessed on 18 July, 2021].
- [2] Medium. 2021. *AWS SageMaker*. [online] Available at: <https://towardsdatascience.com/aws-sagemaker-db5451e02a79> [Accessed 18 July 2021].
- [3] "Start Your Machine Learning on AWS SageMaker", *Medium*, 2021. [Online]. Available: <https://medium.com/weareservian/machine-learning-on-aws-sagemaker-53e1a5e218d9>. [Accessed on 18 July, 2021].
- [4] Amazon Web Services, Inc. 2021. Amazon Comprehend - Natural Language Processing (NLP) and Machine Learning (ML). [online] Available at: <https://aws.amazon.com/comprehend/> [Accessed 18 July 2021].
- [5] "What Is Amazon Comprehend? - Amazon Comprehend", *Docs.aws.amazon.com*, 2021. [Online]. Available: <https://docs.aws.amazon.com/comprehend/latest/dg/what-is.html>. [Accessed on 10 July, 2021].
- [6] "Comprehend — Boto3 Docs 1.17.109 documentation", *Boto3.amazonaws.com*, 2021. [Online]. Available: [https://boto3.amazonaws.com/v1/documentation/api/latest/reference/services/comprehend.html#Comprehend.Client.detect\\_sentiment](https://boto3.amazonaws.com/v1/documentation/api/latest/reference/services/comprehend.html#Comprehend.Client.detect_sentiment). [Accessed on 12 July, 2021].
- [7] Medium. 2021. *Finding and removing duplicate rows in Pandas DataFrame*. [online] Available at: <https://towardsdatascience.com/finding-and-removing-duplicate-rows-in-pandas-dataframe-c6117668631f> [Accessed 20 July 2021].
- [8] Medium. 2021. *Categorical encoding using Label-Encoding and One-Hot-Encoder*. [online] Available at: <https://towardsdatascience.com/categorical-encoding-using-label-encoding-and-one-hot-encoder-911ef77fb5bd> [Accessed 20 July 2021]
- [9] VanderPlas, J., 2021. *In Depth: k-Means Clustering / Python Data Science Handbook*. [online] Jakevdp.github.io. Available at: <https://jakevdp.github.io/PythonDataScienceHandbook/05.11-k-means.html> [Accessed 20 July 2021].
- [10] Getting TypeError: '&#39;(slice(None, 0., Avadhanula, S. and Fatly, A., 2021. *Getting TypeError: '(slice(None, None, None), 0)' is an invalid key*. [online] Stack Overflow. Available at: <https://stackoverflow.com/questions/55291667/getting-typeerror-slicenone-none-none-0-is-an-invalid-key> [Accessed 20 July 2021].

[11] dzone.com. 2021. *KMeans Silhouette Score With Python Examples - DZone AI*. [online] Available at: <https://dzone.com/articles/kmeans-silhouette-score-explained-with-python-exam> [Accessed 20 July 2021].

[12] Matplotlib.org. 2021. *Specifying Colors — Matplotlib 3.4.2 documentation*. [online] Available at: <https://matplotlib.org/stable/tutorials/colors/colors.html> [Accessed 20 July 2021].

[13] <https://stackabuse.com/levenshtein-distance-and-text-similarity-in-python>

[14] <https://www.geeksforgeeks.org/removing-stop-words-nltk-python/>

[15] [https://www.tutorialspoint.com/python\\_text\\_processing/python\\_remove\\_stopwords.html](https://www.tutorialspoint.com/python_text_processing/python_remove_stopwords.html)

[16] <https://www.datacamp.com/community/tutorials/principal-component-analysis-in-python>

[17] <https://medium.com/analytics-vidhya/working-with-twitter-data-b0aa5419532>