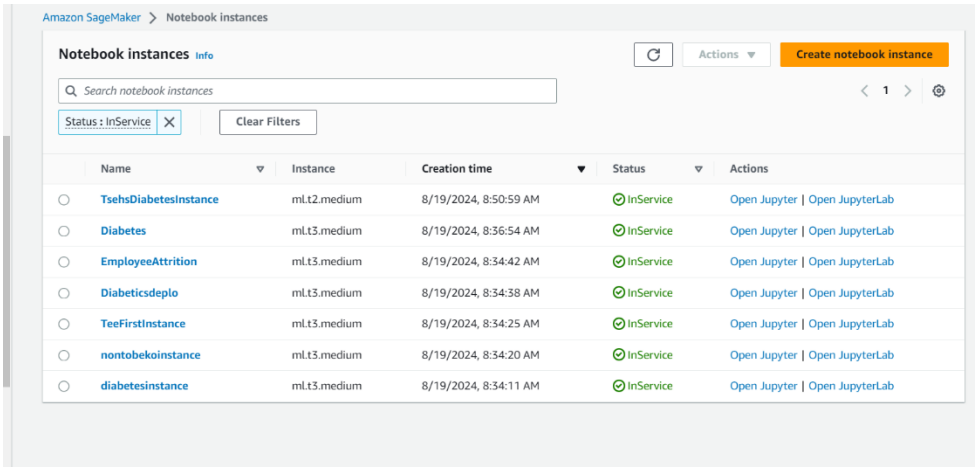
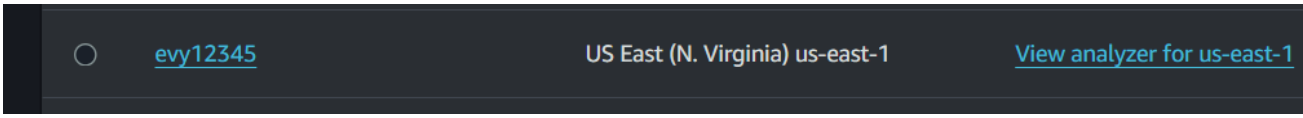
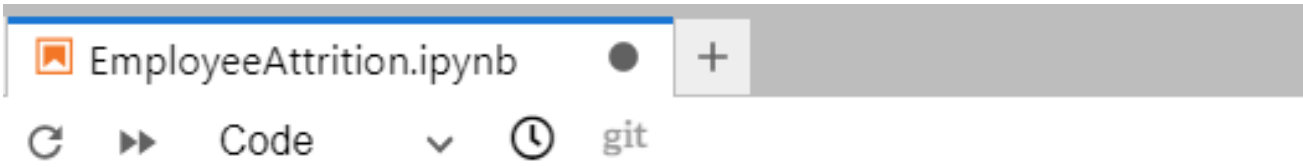
1. Create a notebook instance(“EmployeeAttrition”)



2. Create a s3 bucket(“evy12345”)



3. Create a notebook



4. Import necessary libraries

```
[84]: # Import Libraries
import pandas as pd
import numpy as np
import seaborn as sns
import warnings
import matplotlib.pyplot as plt
from sklearn.model_selection import train_test_split
from pandas.plotting import register_matplotlib_converters
register_matplotlib_converters()
pd.set_option('display.max_columns', None)
warnings.filterwarnings('ignore')
```

5. Import the train data

```
[85]: #import the train data
df=pd.read_csv('train.csv')
df.head()
```

[85]:

	Employee ID	Age	Gender	Years at Company	Job Role	Monthly Income	Work-Life Balance	Job Satisfaction	Performance Rating	Number of Promotions	Overtime	Distance from Home	Education Level	Marital Status	Number of Dependents
0	8410	31	Male	19	Education	5390	Excellent	Medium	Average	2	No	22	Associate Degree	Married	0
1	64756	59	Female	4	Media	5534	Poor	High	Low	3	No	21	Master's Degree	Divorced	3
2	30257	24	Female	10	Healthcare	8159	Good	High	Low	0	No	11	Bachelor's Degree	Married	3
3	65791	36	Female	7	Education	3989	Good	High	High	1	No	27	High School	Single	2
4	65026	56	Male	41	Education	4821	Fair	Very High	Average	0	Yes	71	High School	Divorced	0

6. Check for missing values

```
[83]: # Checking for missing
      df.isnull().sum()
```

```
[83]: Employee ID                0
      Age                        0
      Gender                     0
      Years at Company           0
      Job Role                   0
      Monthly Income             0
      Work-Life Balance          0
      Job Satisfaction           0
      Performance Rating         0
      Number of Promotions       0
      Overtime                   0
      Distance from Home         0
      Education Level            0
      Marital Status             0
      Number of Dependents       0
      Job Level                  0
      Company Size               0
      Company Tenure             0
      Remote Work                0
      Leadership Opportunities   0
      Innovation Opportunities   0
      Company Reputation         0
      Employee Recognition       0
      Attrition                  0
      dtype: int64
```

7. Describe the data

```
[49]: # Describe the data
      df.describe()
```

	Employee ID	Age	Years at Company	Monthly Income	Number of Promotions	Distance from Home	Number of Dependents	Company Tenure
count	59598.000000	59598.000000	59598.000000	59598.000000	59598.000000	59598.000000	59598.000000	59598.000000
mean	37227.118729	38.565875	15.753901	7302.397983	0.832578	50.007651	1.648075	55.758415
std	21519.150028	12.079673	11.245981	2151.457423	0.994991	28.466459	1.555689	25.411090
min	1.000000	18.000000	1.000000	1316.000000	0.000000	1.000000	0.000000	2.000000
25%	18580.250000	28.000000	7.000000	5658.000000	0.000000	25.000000	0.000000	36.000000
50%	37209.500000	39.000000	13.000000	7354.000000	1.000000	50.000000	1.000000	56.000000
75%	55876.750000	49.000000	23.000000	8880.000000	2.000000	75.000000	3.000000	76.000000
max	74498.000000	59.000000	51.000000	16149.000000	4.000000	99.000000	6.000000	128.000000

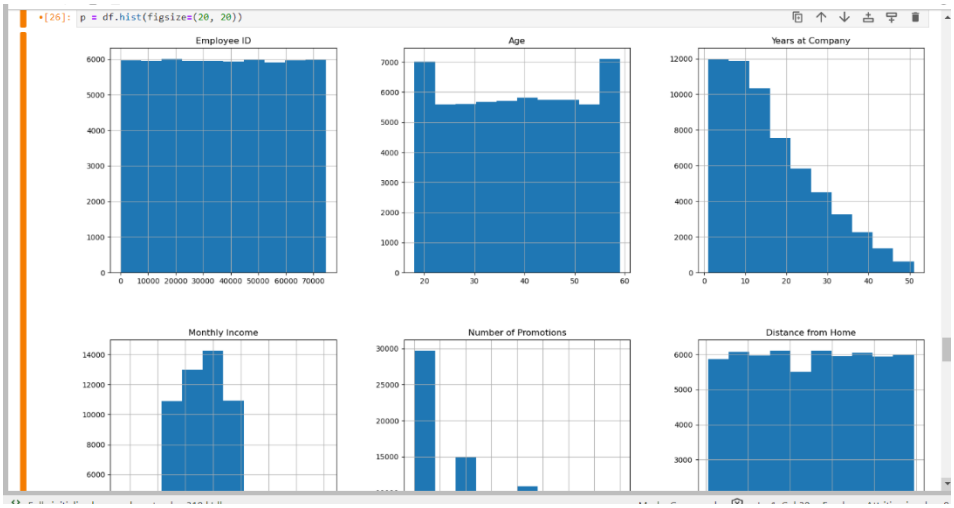
8. Checking for any repeating column names or same

```
[50]: for column in df.columns:
          if df[column].dtype == object:
              print(str(column) + ' : '+str(df[column].unique()))
              print(df[column].value_counts())
```

```
Gender : ['Male' 'Female']
Male      32739
Female    26859
Name: Gender, dtype: int64
Job Role : ['Education' 'Media' 'Healthcare' 'Technology' 'Finance']
Technology    15507
Healthcare    13642
Education     12490
Media          9574
Finance        8385
Name: Job Role, dtype: int64
Work-Life Balance : ['Excellent' 'Poor' 'Good' 'Fair']
Good         22528
Fair         18046
Excellent    10719
Poor          8305
Name: Work-Life Balance, dtype: int64
Job Satisfaction : ['Medium' 'High' 'Very High' 'Low']
High         29779
Very High    12111
Medium       11817
Low           5891
Name: Job Satisfaction, dtype: int64
Performance Rating : ['Average' 'Low' 'High' 'Below Average']
Average          35810
High             11888
Below Average     8950
Low               2950
```

9. Visualization of the data

```
[26]: p = df.hist(figsize=(20, 20))
```



10. Label encoding

```
# Label encoding
columns = ['Gender','Job Role','Overtime','Education Level','Marital Status','Company Size','Remote Work',
           'Leadership Opportunities', 'Innovation Opportunities','Work-Life Balance', 'Job Satisfaction','Performance Rating'
           'Company Reputation','Job Level', 'Employee Recognition', 'Attrition']


label_encoders = {col: LabelEncoder() for col in columns}


for col in columns:
    df[col] = label_encoders[col].fit_transform(df[col])
```

```
df.head()
```

	Employee ID	Age	Gender	Years at Company	Job Role	Monthly Income	Work-Life Balance	Job Satisfaction	Performance Rating	Number of Promotions	Overtime	Distance from Home	Education Level	Marital Status	Number of Dependents	J Le
0	8410	31	1	19	0	5390	0	2	0	2	0	22	0	1	0	
1	64756	59	0	4	3	5534	3	0	3	3	0	21	3	0	3	
2	30257	24	0	10	2	8159	2	0	3	0	0	11	1	1	3	
3	65791	36	0	7	0	3989	2	0	2	1	0	27	2	2	2	
4	65026	56	1	41	0	4821	1	3	0	0	1	71	2	0	0	

11. Predict target

```
# drop the Attrition column
x=df.iloc[:, :-1]
```

```
x.head()
```

ice ng	Number of Promotions	Overtime	Distance from Home	Education Level	Marital Status	Number of Dependents	Job Level	Company Size	Company Tenure	Remote Work	Leadership Opportunities	Innovation Opportunities	Company Reputation	Employee Recognition
0	2	0	22	0	1	0	1	1	89	0	0	0	0	2
3	3	0	21	3	0	3	1	1	21	0	0	0	1	1
3	0	0	11	1	1	3	1	1	74	0	0	0	3	1
2	1	0	27	2	2	2	1	2	50	1	0	0	2	2
0	0	1	71	2	0	0	2	1	68	0	0	0	1	2

12. Predict target

```
In [171]: y.head()

Out[171]: 0    1
          1    1
          2    1
          3    1
          4    1
          Name: Attrition, dtype: int64
```

13. Split the data

```
In [172]: x_train, x_test, y_train, y_test = train_test_split(x, y, test_size=0.2)

In [173]: print(x.shape, x_train.shape, x_test.shape)

          (59598, 23) (47678, 23) (11920, 23)
```

14. Join the test and the train data

```
In [174]: TestData=x_test.join(y_test)

In [175]: TrainData=x_train.join(y_train)
```

15. Re-arrange the columns

```
: column = ['Attrition',
            'Age','Gender',
            'Years at Company',
            'Job Role','Marital Status',
            'Education Level',
            'Job Level',
            'Number of Dependents',
            'Monthly Income',
            'Work-Life Balance',
            'Job Satisfaction','Overtime',
            'Distance from Home','Company Size',
            'Company Tenure','Remote Work',
            'Leadership Opportunities',
            'Innovation Opportunities',
            'Company Reputation',
            'Employee Recognition',
           ]
```

16. Create the new columns data frame

```
In [206]: TestData=TestData[column]
          TestData.head()
```

Out[206]:

	Attrition	Age	Gender	Years at Company	Job Role	Marital Status	Education Level	Job Level	Number of Dependents	Monthly Income	Work-Life Balance	Job Satisfaction	Overtime	Distance from Home	Company Size	Company Tenure
21518	1	32	1	5	1	2	3	1	2	9410	1	3	1	5	1	84
54763	1	37	0	20	3	1	1	1	6	5803	0	2	0	23	2	72
49749	0	27	1	11	0	2	2	0	2	4005	0	0	0	94	2	70
6631	0	31	0	1	4	0	2	0	2	9684	1	0	1	80	0	24
32516	1	47	1	20	2	1	1	1	0	9830	3	1	0	47	2	47

```
In [181]: TrainData = TrainData[column]
          TrainData.head()
```

Out[181]:

	Attrition	Age	Gender	Years at Company	Job Role	Marital Status	Education Level	Job Level	Number of Dependents	Monthly Income	Work-Life Balance	Job Satisfaction	Overtime	Distance from Home	Company Size	Company Tenure
5927	0	55	0	7	4	1	1	2	1	7624	2	0	0	80	2	10
1231	0	57	1	2	4	2	0	1	0	8196	1	1	0	85	1	63
30022	1	40	1	25	4	0	1	2	0	9944	2	3	0	89	0	40
15778	1	27	1	16	3	0	4	0	1	5443	1	0	1	50	2	86
9296	0	22	0	8	2	1	2	2	0	7647	1	0	1	22	1	40

17. Drop the attrition table

```
TestData = TestData [column[1:]]
TestData.head()
```

	Attrition	Age	Gender	Years at Company	Job Role	Marital Status	Education Level	Job Level	Number of Dependents	Monthly Income	Work-Life Balance	Job Satisfaction	Overtime	Distance from Home	Company Size	Company Tenure
21518	1	32	1	5	1	2	3	1	2	9410	1	3	1	5	1	84
54763	1	37	0	20	3	1	1	1	6	5803	0	2	0	23	2	72
49749	0	27	1	11	0	2	2	0	2	4005	0	0	0	94	2	70
6631	0	31	0	1	4	0	2	0	2	9684	1	0	1	80	0	24
32516	1	47	1	20	2	1	1	1	0	9830	3	1	0	47	2	47

18. Save the train and the test data

```
TrainData.to_csv('TrainData.csv',index=False, index_label='Row',header=False, columns=column)
```

```
TestData.to_csv('TestData.csv',index=False, index_label='Row',header=False, columns=column)
```

19. Import the necessary libraries

```
# import libraries
import boto3
import re
```

20. Create files

```
# Create the files to load the data
bucketName = 'evy12345'
TrainFile = r'AttritionData/TrainData/TrainData.csv'
TestFile = r'AttritionData/TestData/TestData.csv'
valFile = r'AttritionModel/val/val.csv'
model = r'AttritionModel/model'
```

21. Upload files to s3

```
# load the data to s3
s3ModelOutput = r's3://{0}/{1}'.format(bucketName, model)
s3Train = r's3://{0}/{1}'.format(bucketName, TrainFile)
s3Test = r's3://{0}/{1}'.format(bucketName, TestFile)
s3Val = r's3://{0}/{1}'.format(bucketName, valFile)
```

22. Output the path where my model will be stored

```
In [213]: # The path of my s3 bucket
          s3ModelOutput

Out[213]: 's3://evy1234/AttritionModel/model'
```

23. Store the files to s3

```
with open('TrainData.csv','rb') as f:
    boto3.Session().resource('s3').Bucket(bucketName).Object(TrainFile).upload_fileobj(f)
```

```
with open('TestData.csv','rb') as f:
    boto3.Session().resource('s3').Bucket(bucketName).Object(TestFile).upload_fileobj(f)
```

24. Train the model

```
import sagemaker
from sagemaker import get_execution_role
```

```
sagemaker.config INFO - Not applying SDK defaults from location: /etc/xdg/sagemaker/config.yaml
sagemaker.config INFO - Not applying SDK defaults from location: /home/ec2-user/.config/sagemaker/config.yaml
```

```
sagemakerSess=sagemaker.Session()
role=get_execution_role()
```

```
sagemakerSess.boto_region_name
```

```
'us-east-1'
```

```
ECRdockercontainer=sagemaker.amazon.amazon_estimator.get_image_uri(sagemakerSess.boto_region_name,'linear-learner','latest')
```

```
The method get_image_uri has been renamed in sagemaker>=2.
See: https://sagemaker.readthedocs.io/en/stable/v2.html for details.
Defaulting to the only supported framework/algorithm version: 1. Ignoring framework/algorithm version: latest.
```

```
LogisticEmployeeAttrition=sagemaker.estimator.Estimator(image_uri=ECRdockercontainer,
                                        role=role,
                                         train_instance_count=1,
                                         train_instance_type='ml.m4.xlarge',
                                         output_path=s3ModelOutput,
                                         sagemaker_session=sagemakerSess,
                                         base_job_name = 'Logistic-Demo-v1'
                                        )
```

```
WARNING:sagemaker.deprecations:train_instance_count has been renamed in sagemaker>=2.
See: https://sagemaker.readthedocs.io/en/stable/v2.html for details.
WARNING:sagemaker.deprecations:train_instance_type has been renamed in sagemaker>=2.
See: https://sagemaker.readthedocs.io/en/stable/v2.html for details.
```

```
LogisticEmployeeAttrition.fit({'train':trainConfig})
```

```
[08/21/2024 08:44:49 INFO 140382492686144] #quality_metric: host=algo-1, train binary_log_loss <score>=0.6586828703905728
[08/21/2024 08:44:49 INFO 140382492686144] Best model found for hyperparameters: {"optimizer": "adam", "learning_rate": 0.005, "wd": 0.0001, "l1": 0.0, "lr_scheduler_step": 10, "lr_scheduler_factor": 0.99, "lr_scheduler_minimum_lr": 0.0001}
[08/21/2024 08:44:49 INFO 140382492686144] Saved checkpoint to "/tmp/tmpouep4sw4/mx-mod-0000.params"
[08/21/2024 08:44:49 INFO 140382492686144] Test data is not provided.
#metrics {"StartTime": 1724229816.5294223, "EndTime": 1724229889.464379, "Dimensions": {"Algorithm": "Linear Learner", "Host": "algo-1", "Operation": "training"}, "Metrics": {"initialize.time": {"sum": 695.7247257232666, "count": 1, "min": 695.7247257232666, "max": 695.7247257232666}, "epochs": {"sum": 15.0, "count": 1, "min": 15, "max": 15}, "check_early_stopping.time": {"sum": 2.1028518676757812, "count": 6, "min": 0.1671314239501953, "max": 0.7407665252685547}, "update.time": {"sum": 67996.1142539978, "count": 6, "min": 11133.790731430054, "max": 11720.49617767334}, "finalize.time": {"sum": 4216.479778289795, "count": 1, "min": 4216.479778289795, "max": 4216.479778289795}, "setuptime": {"sum": 2.167224884033203, "count": 1, "min": 2.167224884033203, "max": 2.167224884033203}, "totaltime": {"sum": 73041.28837585449, "count": 1, "min": 73041.28837585449, "max": 73041.28837585449}}}

2024-08-21 08:45:09 Uploading - Uploading generated training model
2024-08-21 08:45:09 Completed - Training job completed
Training seconds: 210
Billable seconds: 210
```

25. Deploy the model

```
: #Deploying the Trained Model

Model=LogisticEmployeeAttrition.deploy(initial_instance_count=1, instance_type='ml.m4.xlarge',
                                       endpoint_name = 'EmployeeAttrition')
```

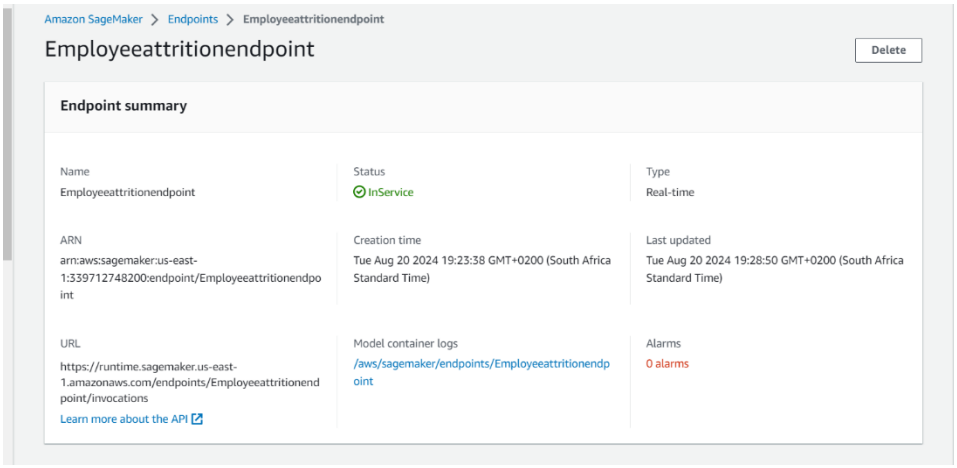
```
INFO:sagemaker:Creating model with name: Logistic-Demo-v1-2024-08-21-08-53-34-497
INFO:sagemaker:Creating endpoint-config with name EmployeeAttrition
INFO:sagemaker:Creating endpoint with name EmployeeAttrition

--------!
```

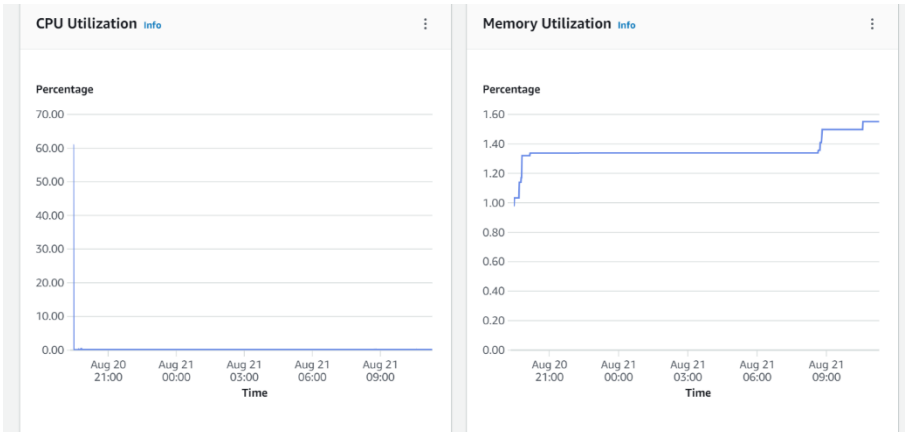
26. Model deployed successfully

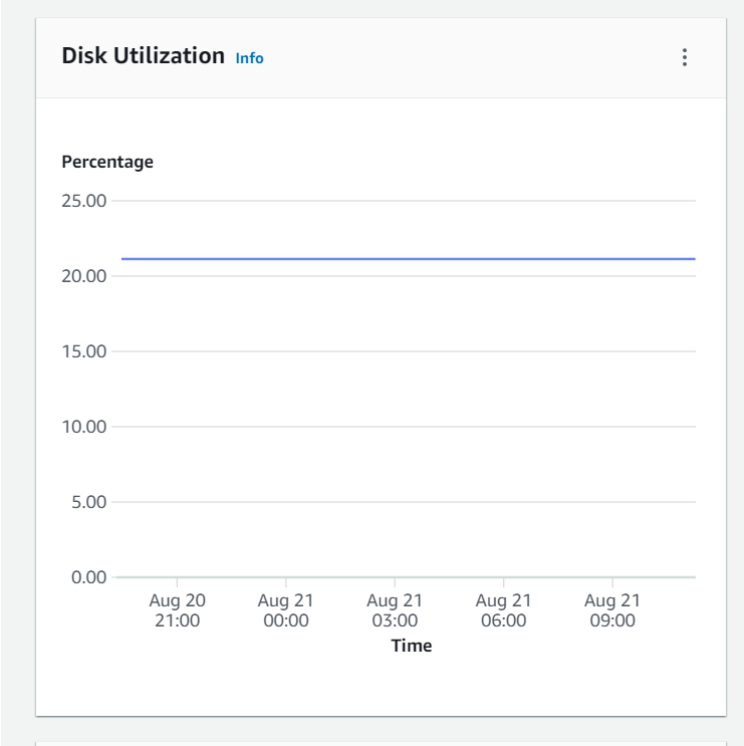
	Name	ARN	Creation time	Status	Last updated
○	EmployeeAttrition	arn:aws:sagemaker:us-east-1:339712748200:endpoint/EmployeeAttrition	8/21/2024, 10:53:35 AM	InService	8/21/2024, 10:57:39 AM

The Endpoint

https://runtime.sagemaker.us-east-1.amazonaws.com/endpoints/Employeeattritionendpoint/invocations

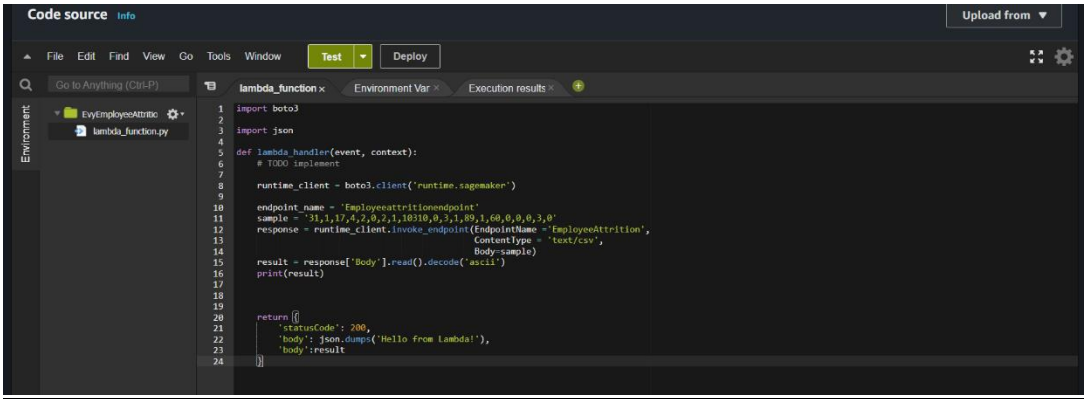


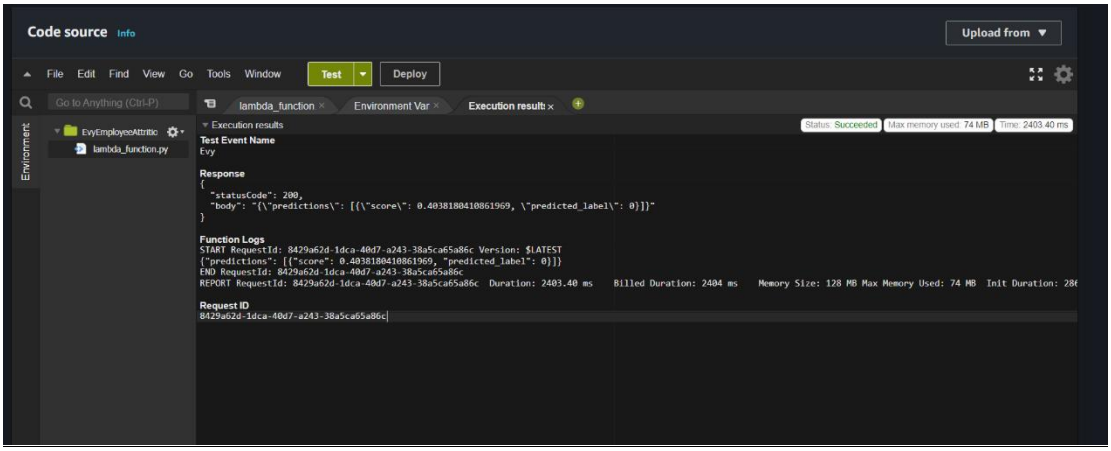




Lambda functions

☐	EvyEmployeeAttrition	-	Zip	Python 3.12	11 minutes ago


Code source Info
Upload from
File Edit Find View Go Tools Window
Test
Deploy
Go to Anything (Ctrl-P)
lambda_function
Environment Var
Execution results
Environment
EvyEmployeeAttritio
lambda_function.py
import boto3
import json
def lambda_handler(event, context):
# TODO implement
runtime_client = boto3.client('runtime.sagemaker')
endpoint_name = 'Employeeattritionendpoint'
sample = '31,1,17,4,2,0,2,1,10310,0,3,1,80,1,60,0,0,0,3,0'
response = runtime_client.invoke_endpoint(EndpointName ='EmployeeAttrition',
ContentType = 'text/csv',
Body=sample)
result = response['Body'].read().decode('ascii')
print(result)
return {
'statusCode': 200,
'body': json.dumps('Hello from Lambda!'),
'body':result
}


Code source Info
Upload from
File Edit Find View Go Tools Window
Test
Deploy
Go to Anything (Ctrl-P)
lambda_function
Environment Var
Execution result
Environment
EvyEmployeeAttritio
lambda_function.py
Execution results
Status: Succeeded
Max memory used: 74 MB
Time: 2403.40 ms
Test Event Name
Evy
Response
{
"statusCode": 200,
"body": "{\"predictions\": [{\"score\": 0.403818041861969, \"predicted_label\": 0}]}"
}
Function Logs
START RequestId: 8429a62d-1dca-40d7-a243-38a5ca65a86c Version: $LATEST
{"predictions": [{"score": 0.403818041861969, "predicted_label": 0}]}
END RequestId: 8429a62d-1dca-40d7-a243-38a5ca65a86c
REPORT RequestId: 8429a62d-1dca-40d7-a243-38a5ca65a86c Duration: 2403.40 ms Billed Duration: 2404 ms Memory Size: 128 MB Max Memory Used: 74 MB Init Duration: 28
Request ID
8429a62d-1dca-40d7-a243-38a5ca65a86c