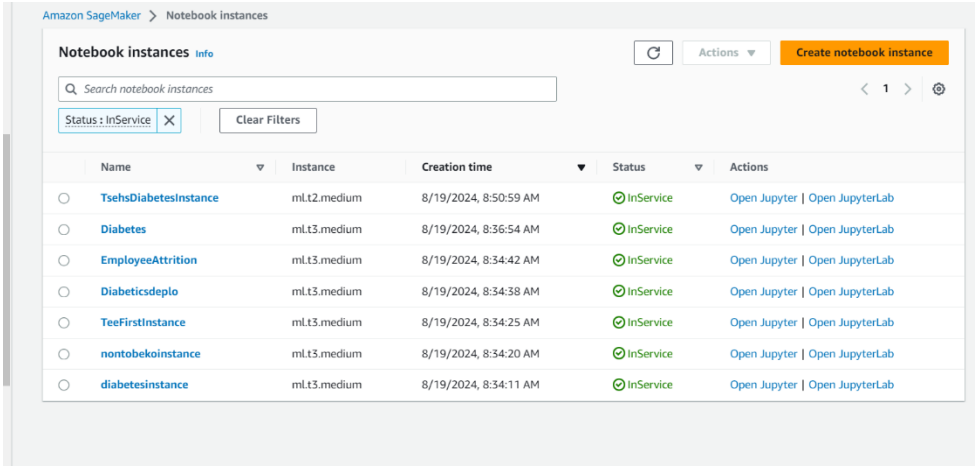
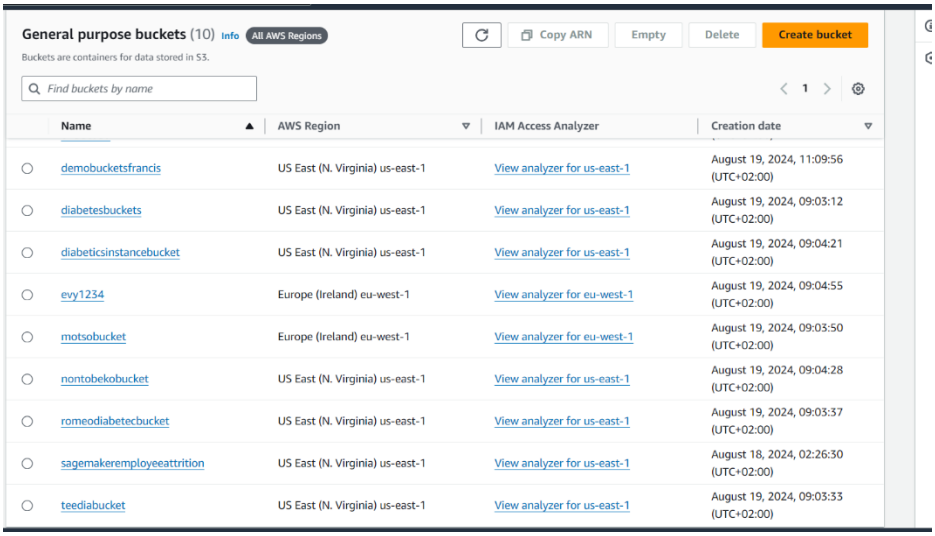
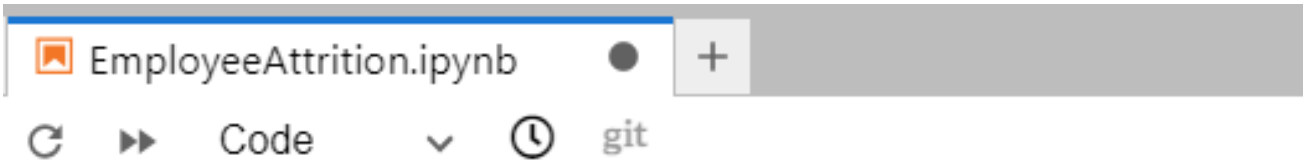
Create a notebook instance(“EmployeeAttrition”)



1. Create a s3 bucket(“evy1234”)



2. Create a notebook



3. 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')
```

4. 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

5. 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
```

6. Describe the data

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

[49]:

	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

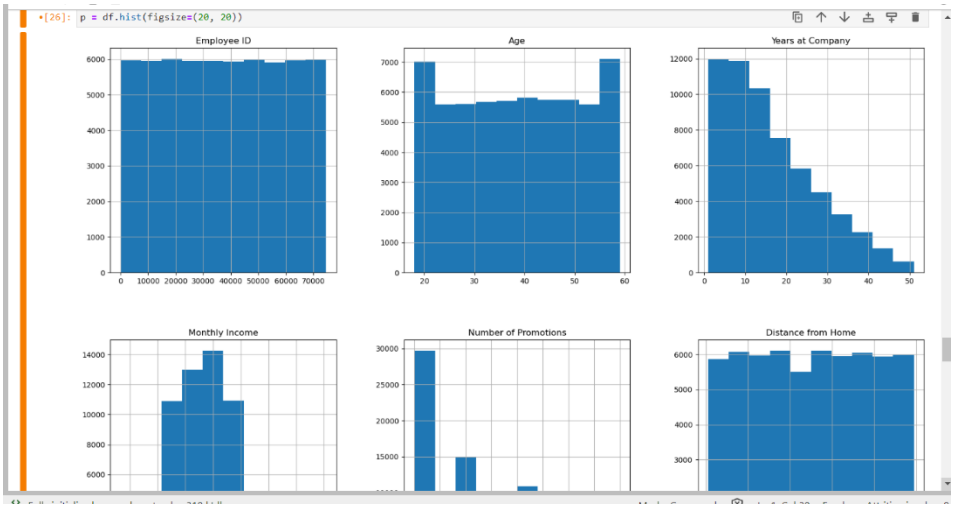
7. Checking for any repeating column names or same

```
[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
```

8. Visualization of the data

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



9. Label encoding

```
[74]: # Label encoding
columns=['Gender', 'Monthly Income', 'Job Role', 'Work-Life Balance', 'Job Satisfaction', 'Performance Rating', 'Overtime', 'Education Level', 'Marital
       'Remote Work', 'Leadership Opportunities', 'Innovation Opportunities', 'Company Reputation', 'Employee Recognition', 'Attrition']
le =LabelEncoder()
df[columns]=df[columns].apply(le.fit_transform)
```

```
[75]: df.head()
```

[75]:

	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	Job Level
0	8410	31	1	19	0	2611	0	2	0	2	0	22	0	1	0	1
1	64756	59	0	4	3	2755	3	0	3	3	0	21	3	0	3	1
2	30257	24	0	10	2	5380	2	0	3	0	0	11	1	1	3	1
3	65791	36	0	7	0	1212	2	0	2	1	0	27	2	2	2	1
4	65026	56	1	41	0	2042	1	3	0	0	1	71	2	0	0	2

10. Predict target

```
[53]: # Presict target
y = df.Attrition
y.tail()
```

```
[53]: 59593    0
      59594    0
      59595    1
      59596    0
      59597    1
      Name: Attrition, dtype: int64
```

```
[54]: y.head()
```

```
[54]: 0    1
      1    1
      2    1
      3    1
      4    1
      Name: Attrition, dtype: int64
```

11. Features selection

```
[76]: # Feature selection
columns=[ 'Age', 'Gender', 'Years at Company', 'Job Role', 'Monthly Income', 'Work-Life Balance', 'Job Satisfaction', 'Performance Rating', 'Number of
       'Company Tenure', 'Remote Work', 'Leadership Opportunities', 'Innovation Opportunities', 'Company Reputation', 'Employee Recognition']
x=df[columns]
x.head()
```

[76]:

	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	Job Level	Company Size	Company Tenure	Re
0	31	1	19	0	2611	0	2	0	2	0	22	0	1	1	1	89	
1	59	0	4	3	2755	3	0	3	3	0	21	3	0	1	1	21	
2	24	0	10	2	5380	2	0	3	0	0	11	1	1	1	1	74	
3	36	0	7	0	1212	2	0	2	1	0	27	2	2	1	2	50	
4	56	1	41	0	2042	1	3	0	0	1	71	2	0	2	1	68	

12. Split the data and save the x_train and x_test

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

[79]: print(x.shape, x_train.shape, x_test.shape)
      (59598, 21) (47678, 21) (11920, 21)

[80]: # saving the data
      x_train.to_csv('xtrain.csv', index=False, index_label='Row', header=False)

[81]: x_test.to_csv('x_test.csv', index=False, index_label='Row', header=False)
```

13. Import necessary libraries

```
In [210]: # import libraries
          import boto3
          import re
```

14. Create files and upload them to s3

```
In [211]: # Create the files to load the data
          bucketName = 'evy1234'
          TrainFile = r'AttritionData/traindata/x_train.csv'
          TestFile = r'AttritionData/testdata/x_test.csv'
          valFile = r'AttritionModel/val/val.csv'
          model = r'AttritionModel/model'

In [212]: # 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)
```

15. Output the path where my model will be stored

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

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

16. Store the files to s3

```
In [214]: with open('x_train.csv','rb') as f:
              boto3.Session().resource('s3').Bucket(bucketName).Object(TrainFile).upload_fileobj(f)
          INFO:botocore.credentials:Found credentials from IAM Role: BaseNotebookInstanceEc2InstanceRole

In [215]: with open('x_test.csv','rb') as f:
              boto3.Session().resource('s3').Bucket(bucketName).Object(TestFile).upload_fileobj(f)
          INFO:botocore.credentials:Found credentials from IAM Role: BaseNotebookInstanceEc2InstanceRole
```

17. Train the model

```
In [216]: # Train the model

In [217]: import sagemaker
          from sagemaker import get_execution_role

In [218]: sagemakerSess=sagemaker.Session()
          role=get_execution_role()

In [219]: sagemakerSess.boto_region_name

Out[219]: 'us-east-1'

In [220]: ECRdockercontainer=sagemaker.amazon.amazon_estimator.get_image_uri(sagemakerSess.boto_region_name,'linear-learner','latest')

          WARNING:sagemaker.deprecations:The method get_image_uri has been renamed in sagemaker>=2.
          See: https://sagemaker.readthedocs.io/en/stable/v2.html for details.
          INFO:sagemaker.image_uris:Same images used for training and inference. Defaulting to image scope: inference.
          WARNING:sagemaker.image_uris:Defaulting to the only supported framework/algorithm version: 1. Ignoring framework/algorithm vers
          ion: latest.
          INFO:sagemaker.image_uris:Ignoring unnecessary instance type: None.

In [221]: LogisticModel=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'
                                                      )
```

```
In [221]: LogisticModel=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.

In [222]: LogisticModel.set_hyperparameters(predictor_type='binary_classifier', mini_batch_size = 100)

In [223]: LogisticModel.hyperparameters()

Out[223]: {'predictor_type': 'binary_classifier', 'mini_batch_size': 100}

In [187]: trainConfig=sagemaker.session.s3_input(s3_data=s3Train,content_type='text/csv')

          WARNING:sagemaker.deprecations:The class sagemaker.session.s3_input has been renamed in sagemaker>=2.
          See: https://sagemaker.readthedocs.io/en/stable/v2.html for details.

In [188]: LogisticModel.fit({'train':trainConfig})

          INFO:sagemaker:Creating training-job with name: Logistic-Demo-v1-2024-08-20-07-18-20-879

          2024-08-20 07:18:21 Starting - Starting the training job...
          2024-08-20 07:18:36 Starting - Preparing the instances for training...
          2024-08-20 07:19:04 Downloading - Downloading input data...
```

18. Deploy the model

```
#Deploying the Trained Model

predictmodel=LogisticModel.deploy(initial_instance_count=1, instance_type='ml.m4.xlarge',
                                  endpoint_name = 'LogisticRegression-demo-v1')
```