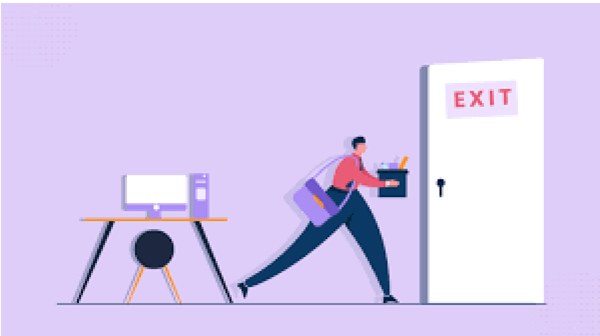


EMPLOYEE ATTRITION CLASSIFICATION DATASET

An In-Depth Synthetic Simulation for Attrition Analysis and Prediction

ABSTRACT

This report details the process of deploying a machine learning model using AWS SageMaker, using the Employee Attrition Classification dataset from Kaggle. The dataset which includes various features related to employee demographics, job roles, serves as a basis for predicting employee turnover within an organization. The report outlines the entire workflow, from data preparation to model training, evaluation, and deployment. Key aspects of AWS are explored, including configuring the SageMaker environment, training the model, and deploying it for inference. Additionally, the project required the management of data and model files, which were stored in an Amazon S3 bucket. The objective is to demonstrate a practical application of AWS SageMaker in implementing machine learning solutions for business problems, showcasing the effectiveness and efficiency of cloud-based machine learning deployments in addressing employee attrition and enhancing organizational retention strategies.

Thapelo Lenzi

Deploying Machine Learning Models in AWS SageMaker

Dataset

This dataset consists of 74 498 entries with each including a unique Employee ID and features that influence employee attrition. The goal is to understand the factors contributing to attrition (Whether the employee has left the company, encoded as 0 for stayed and 1 for left) and develop predictive models to identify at-risk employees. The features include:

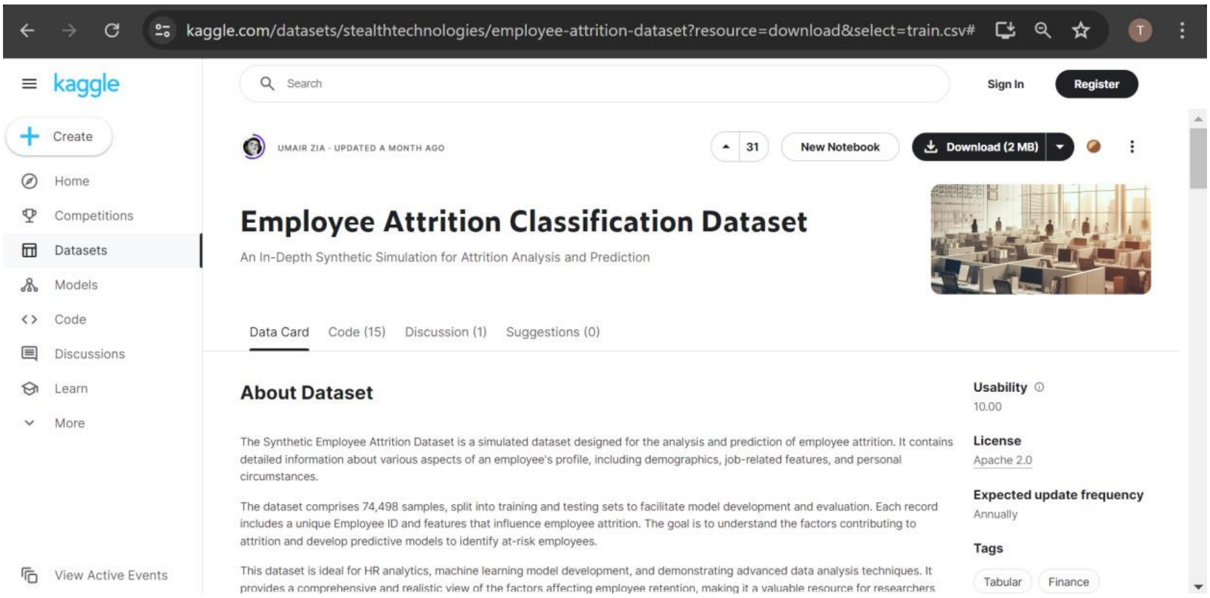
- Age
- Gender
- Years at company
- Monthly income
- Job role
- Work-life balance
- Education level
- And more

https://www.kaggle.com/datasets/stealthtechnologies/employee-attritiondataset?resource=download&select=train.csv

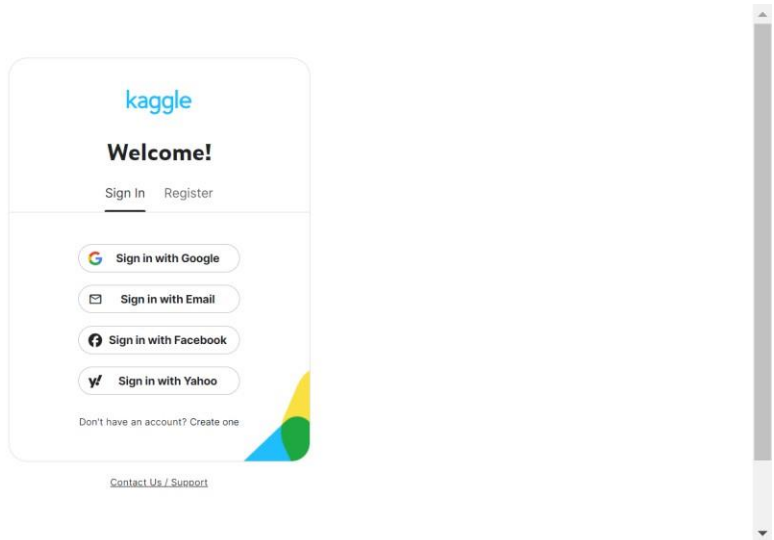
Data Extraction

Obtaining the dataset from Kaggle:

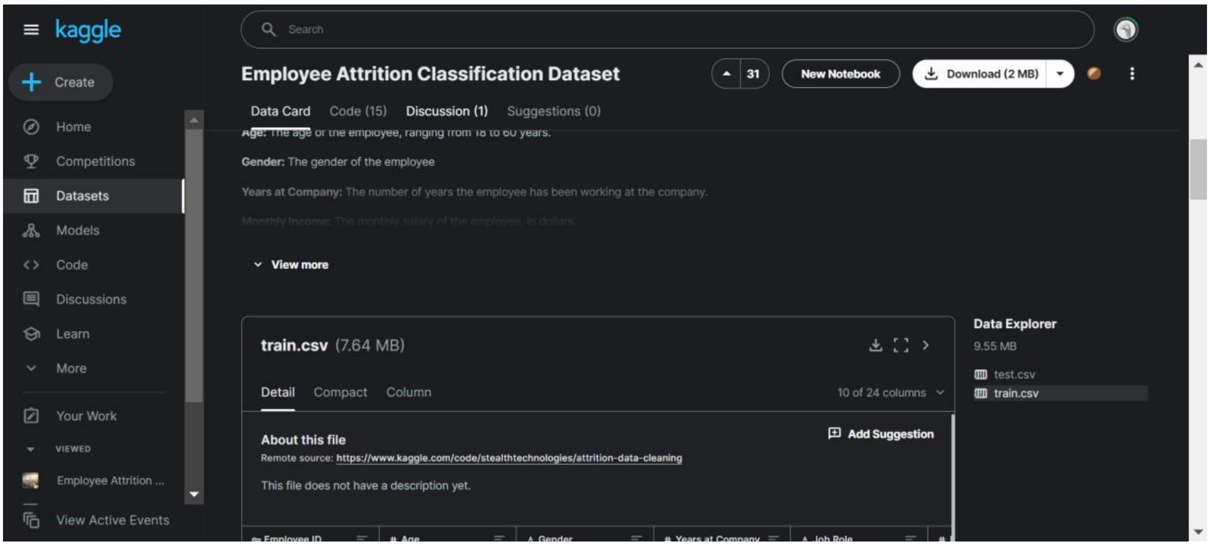
Search for the Employee Attrition Classification dataset on Kaggle.



Sign In:



Go to the dataset page and click on the download button to manually download the dataset:



Data Preparation

The data preparation and exploratory data analysis stages of this project are done using Google Colab.

Loading Libraries, Importing necessary libraries for data manipulation and visualization:

Necessary Packages

```
[20] import pandas as pd
     import numpy as np
     import matplotlib.pyplot as plt
     import seaborn as sn
     pd.set_option('display.max_columns', None)
     import warnings
     warnings.filterwarnings('ignore')

     from sklearn.preprocessing import LabelEncoder
```

Loading the data into a pandas data frame:

```
[4] #Load the data
    data = pd.read_csv("/content/Attrition.csv")
```

Exploratory Data Analysis

The overview of the dataset:

```
[5] #Display the top 5 rows of the dataset
    data.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
0	8410	31	Male	19	Education	5390	Excellent	Medium	Average	2	No	22	Associate Degree
1	64756	59	Female	4	Media	5534	Poor	High	Low	3	No	21	Master's Degree
2	30257	24	Female	10	Healthcare	8159	Good	High	Low	0	No	11	Bachelor's Degree
3	65791	36	Female	7	Education	3989	Good	High	High	1	No	27	High School
4	65026	56	Male	41	Education	4821	Fair	Very High	Average	0	Yes	71	High School

```
#Display the number of rows and columns
data.shape
```

```
(8185, 24)
```

```
#Data type summary
data.info()
```

```
RangeIndex: 8186 entries, 0 to 8185
Data columns (total 24 columns):
 #   Column                       Non-Null Count  Dtype
---  ------                       --------------  -----
 0   Employee ID                  8186 non-null   int64
 1   Age                          8186 non-null   int64
 2   Gender                       8186 non-null   object
 3   Years at Company             8186 non-null   int64
 4   Job Role                     8186 non-null   object
 5   Monthly Income               8186 non-null   int64
 6   Work-Life Balance            8186 non-null   object
 7   Job Satisfaction             8186 non-null   object
 8   Performance Rating           8186 non-null   object
 9   Number of Promotions         8186 non-null   int64
 10  Overtime                     8186 non-null   object
 11  Distance from Home           8186 non-null   int64
 12  Education Level              8186 non-null   object
 13  Marital Status               8186 non-null   object
 14  Number of Dependents         8186 non-null   int64
 15  Job Level                    8186 non-null   object
 16  Company Size                 8186 non-null   object
 17  Company Tenure               8186 non-null   int64
 18  Remote Work                  8186 non-null   object
 19  Leadership Opportunities     8186 non-null   object
 20  Innovation Opportunities     8186 non-null   object
 21  Company Reputation           8185 non-null   object
 22  Employee Recognition         8185 non-null   object
 23  Attrition                    8185 non-null   object
dtypes: int64(8), object(16)
```

```
[7] #Statistical summary
    data.describe()
```

	Employee ID	Age	Years at Company	Monthly Income	Number of Promotions	Distance from Home	Number of Dependents	Company Tenure
count	8186.000000	8186.000000	8186.000000	8186.000000	8186.000000	8186.000000	8186.000000	8186.000000
mean	37287.220743	38.615075	15.667237	7336.613609	0.827144	49.724408	1.649890	55.863059
std	21446.665299	12.135169	11.291536	2141.167892	0.996414	28.518282	1.556062	25.546607
min	8.000000	18.000000	1.000000	1855.000000	0.000000	1.000000	0.000000	2.000000
25%	18757.250000	28.000000	7.000000	5714.000000	0.000000	25.000000	0.000000	36.000000
50%	37088.500000	39.000000	13.000000	7373.000000	0.000000	50.000000	1.000000	56.000000
75%	55937.750000	49.000000	23.000000	8879.500000	1.000000	74.000000	3.000000	76.000000
max	74488.000000	59.000000	51.000000	15495.000000	4.000000	99.000000	6.000000	127.000000

The above snippets provide a sample of the data and understand the structure of the dataset, including column names and initial values. They provide the dimensions of the dataset, which can be observed as 8185 rows and 24 columns, which help in understanding the size of the dataset.

The data type and statistical summaries, include the number of non-null entries in each column and datatypes together with metrics such as the mean, minimum and maximum. These help with understanding the distribution and range of the numeric data and identifying any missing values. From the above, it can be noted that the dataset has two data types (int64 and object) and the last three columns have some null values.

Handling Missing Data

The missing values can be explicitly displayed, as shown below. Since there is only one value missing in each of the three columns and there are a lot of entries, the missing values can be dropped and will not make any significant change in the analysis.

```
[11] #Get number of duplicated data
     data.duplicated().sum()

     0
```

```
[8]  #Get number of empty rows in each column
     data.isnull().sum()

     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           1
     Employee Recognition         1
     Attrition                    1
     dtype: int64
```

```
[12] #Drop the empty values
     data.dropna(inplace = True)
```

```
[14] #verify the empty values are removed
     data.isnull().values.any()

     False
```

```
[19] #Display the count summary for each column
     for column in data.columns:
       if data[column].dtype == object:
         print(f"{column}: {data[column].unique()}")
         print(data[column].value_counts())
         print('..........................')
```

```
..........................
Innovation Opportunities: ['No' 'Yes']
Innovation Opportunities
No     6873
Yes    1312
Name: count, dtype: int64
..........................
Company Reputation: ['Excellent' 'Fair' 'Poor' 'Good']
Company Reputation
Good         4036
Poor         1667
Fair         1655
Excellent     827
Name: count, dtype: int64
..........................
Employee Recognition: ['Medium' 'Low' 'High' 'Very High']
Employee Recognition
Low          3267
Medium       2439
High         2062
Very High     417
Name: count, dtype: int64
..........................
Attrition: ['Stayed' 'Left']
Attrition
Stayed    4282
Left      3903
Name: count, dtype: int64
..........................
```

The above is a summary of each column. In the dependent column (Attrition), there are two unique values with 'Stayed' having the greater count.

Feature Engineering

The dependent column is originally in text form. In order for it to be used in the model, it is converted to numerical form using the LabelEncoder from the scikit-learn library, as seen in the snippet below.

```
le = LabelEncoder()
data['Attrition'] = le.fit_transform(data['Attrition'])
data.head(5)
```

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	Attrition
2	No	22	Associate Degree	Married	0	Mid	Medium	89	No	No	No	Excellent	Medium	1
3	No	21	Master's Degree	Divorced	3	Mid	Medium	21	No	No	No	Fair	Low	1
0	No	11	Bachelor's Degree	Married	3	Mid	Medium	74	No	No	No	Poor	Low	1
1	No	27	High School	Single	2	Mid	Small	50	Yes	No	No	Good	Medium	1
0	Yes	71	High School	Divorced	0	Senior	Medium	68	No	No	No	Fair	Medium	1

```
le.classes_
```

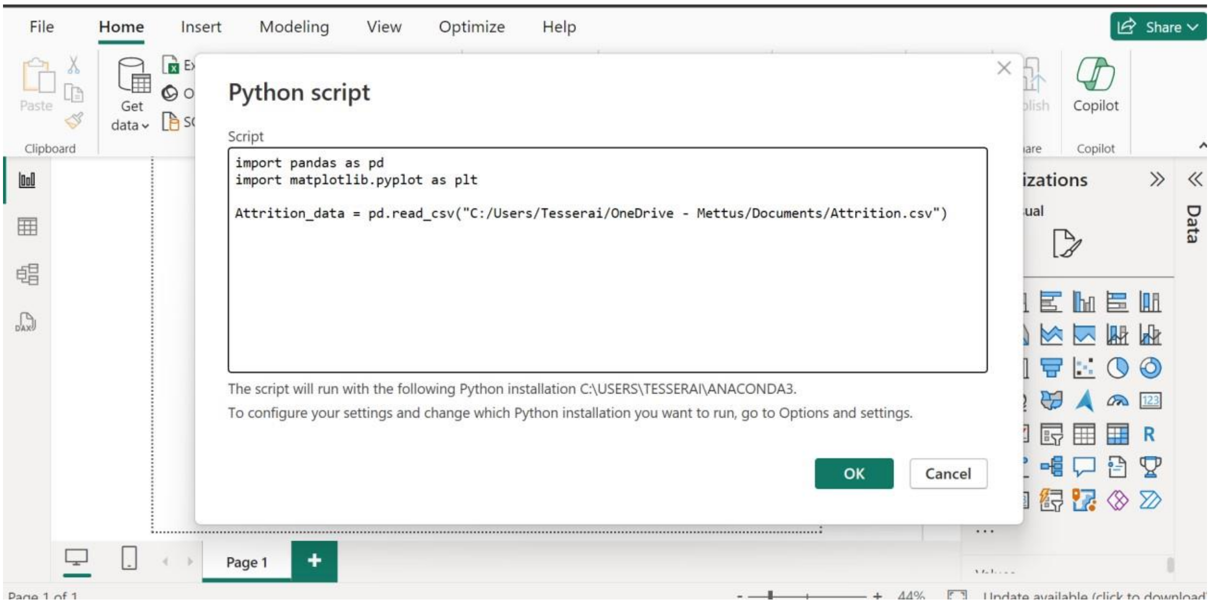
```
array(['Left', 'Stayed'], dtype=object)
```

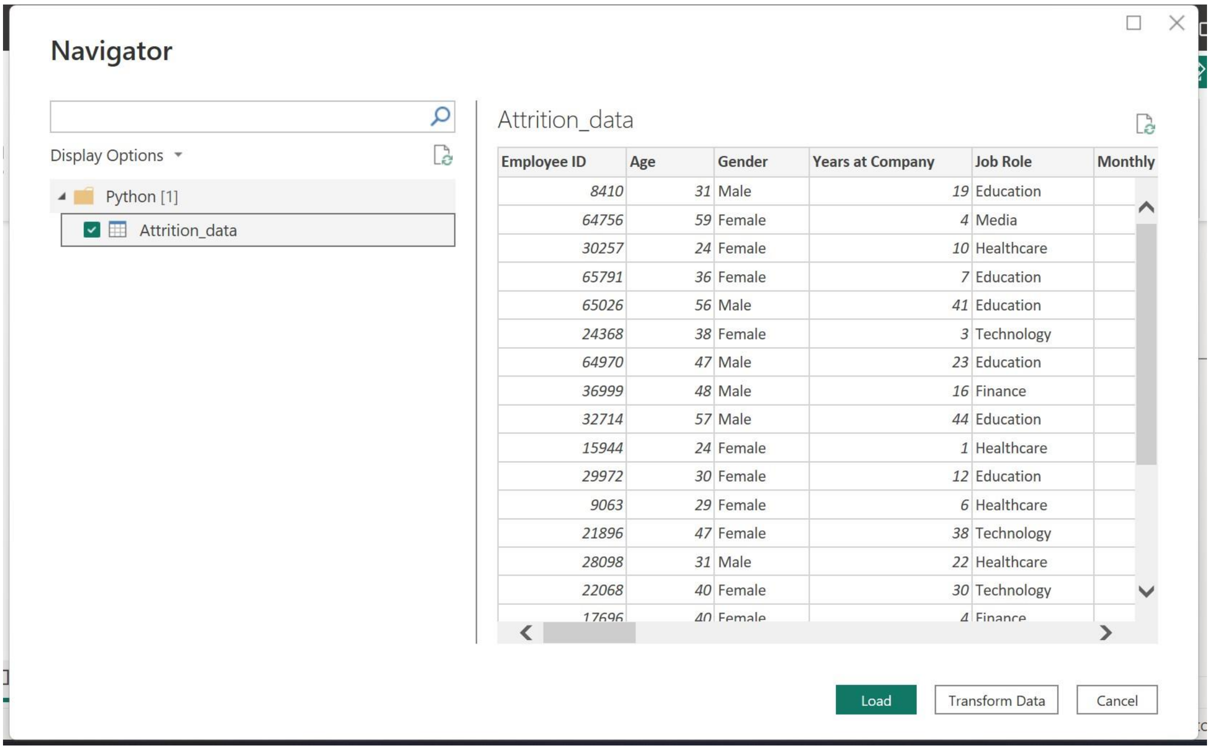
```
[23] Employee_Attrition = le.classes_
     print(Employee_Attrition)
```

```
['Left' 'Stayed']
```

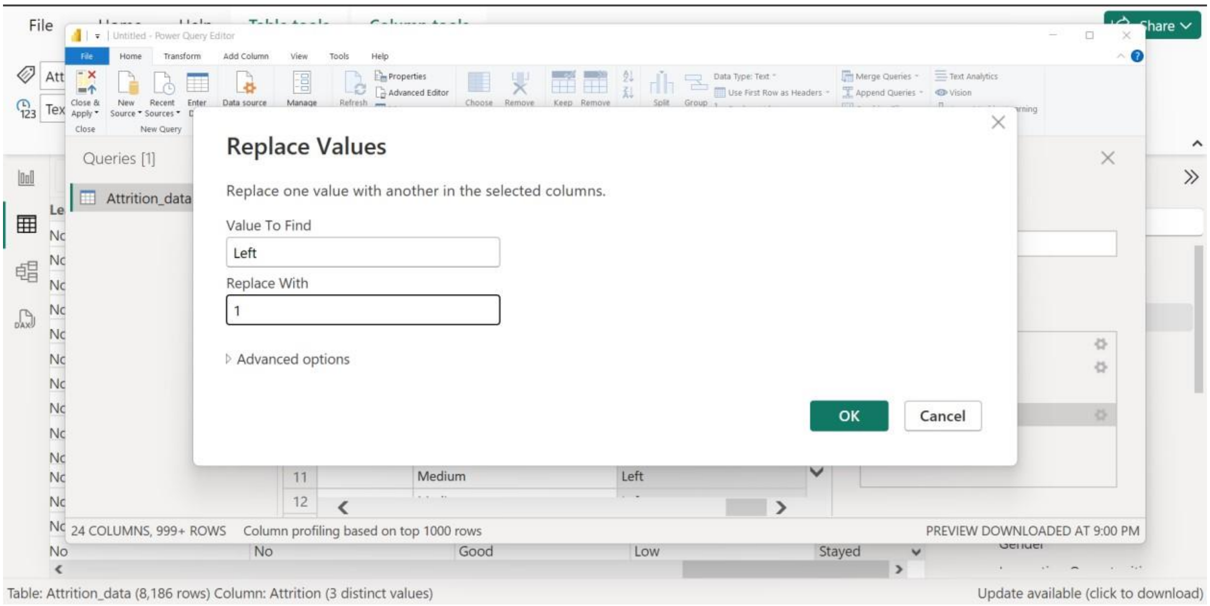
Data Visualization using PowerBI Desktop

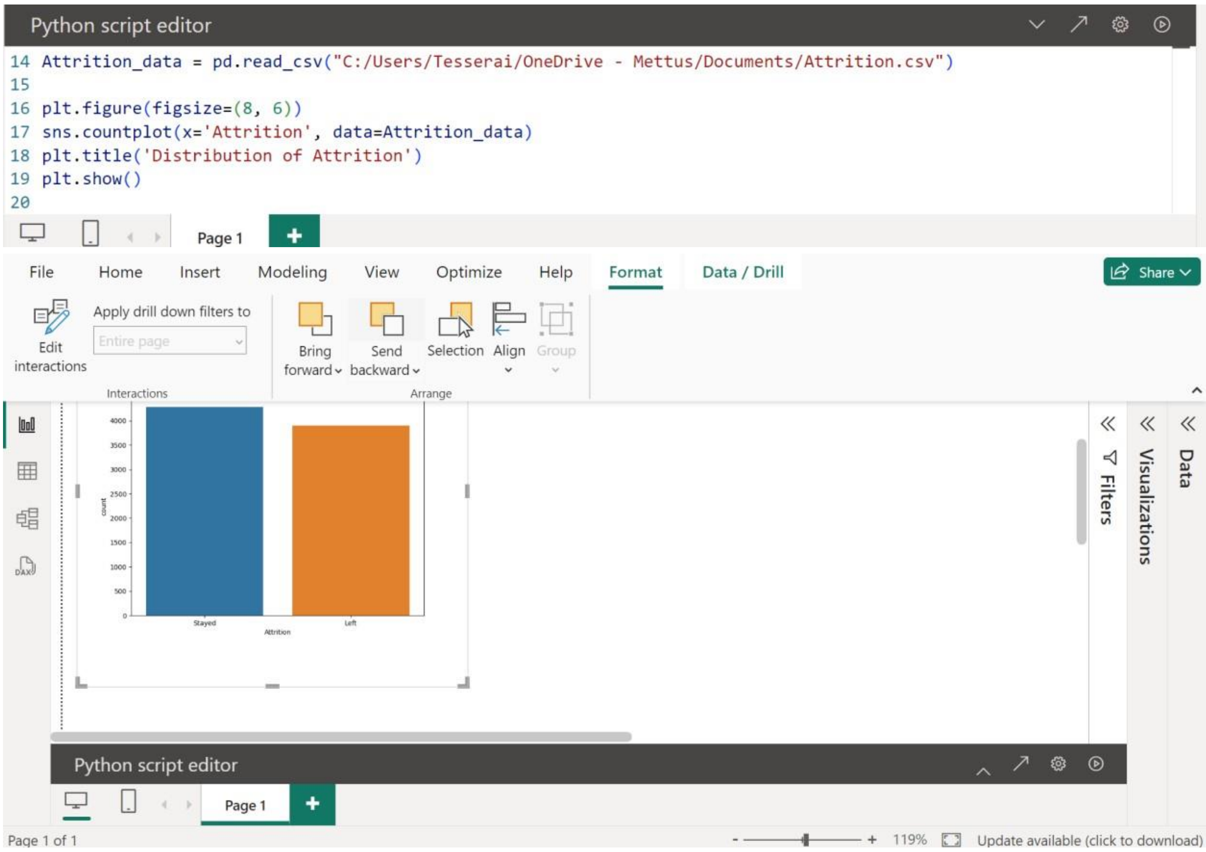
Connecting to Python Script in PowerBI Desktop:



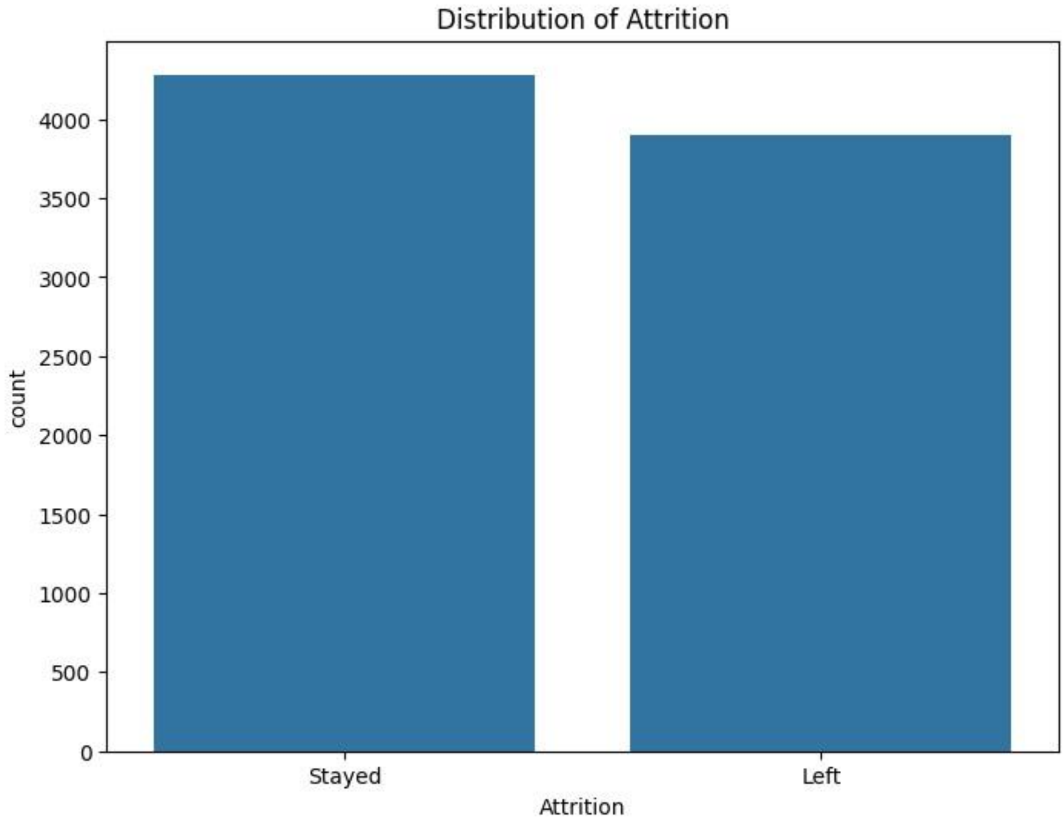


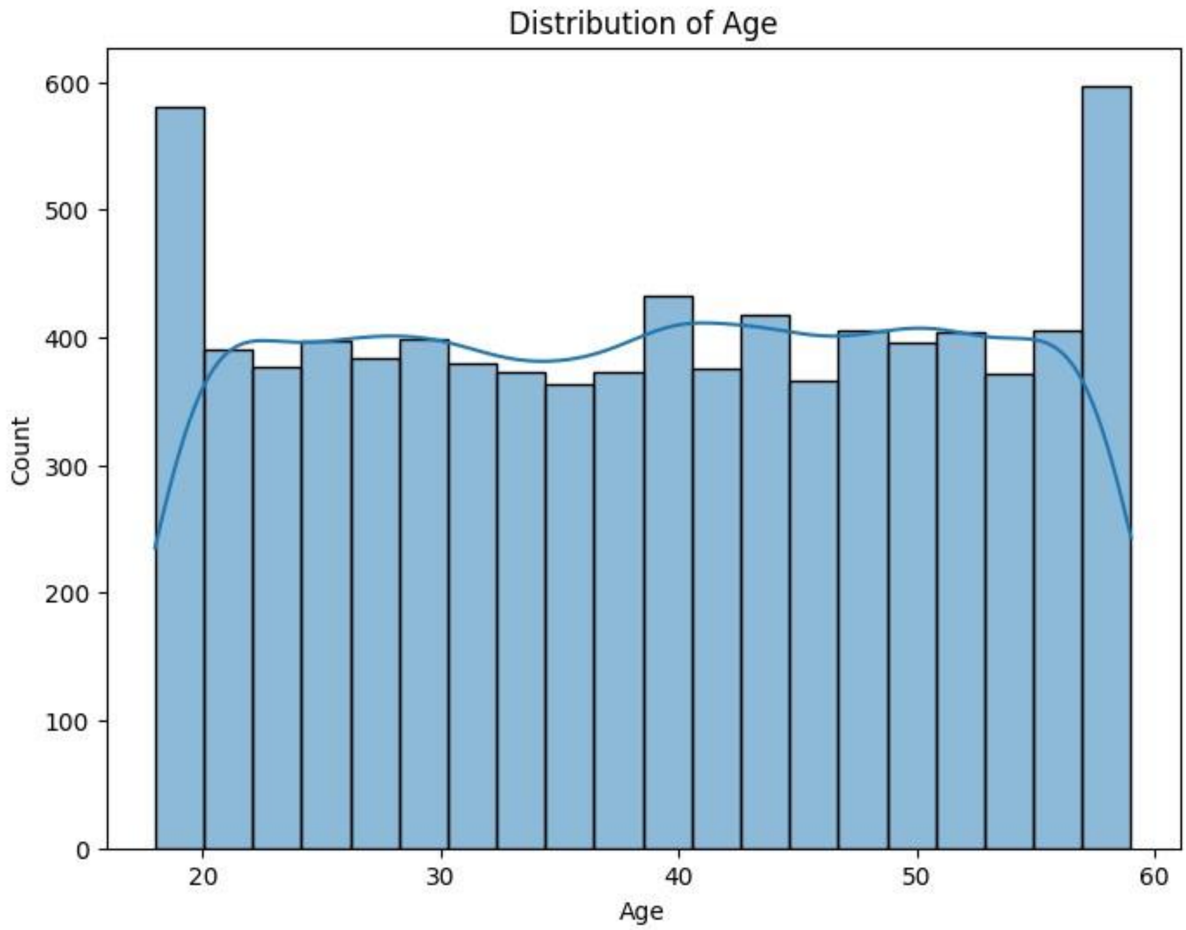
Transforming the Attrition data column from categorical to numerical:

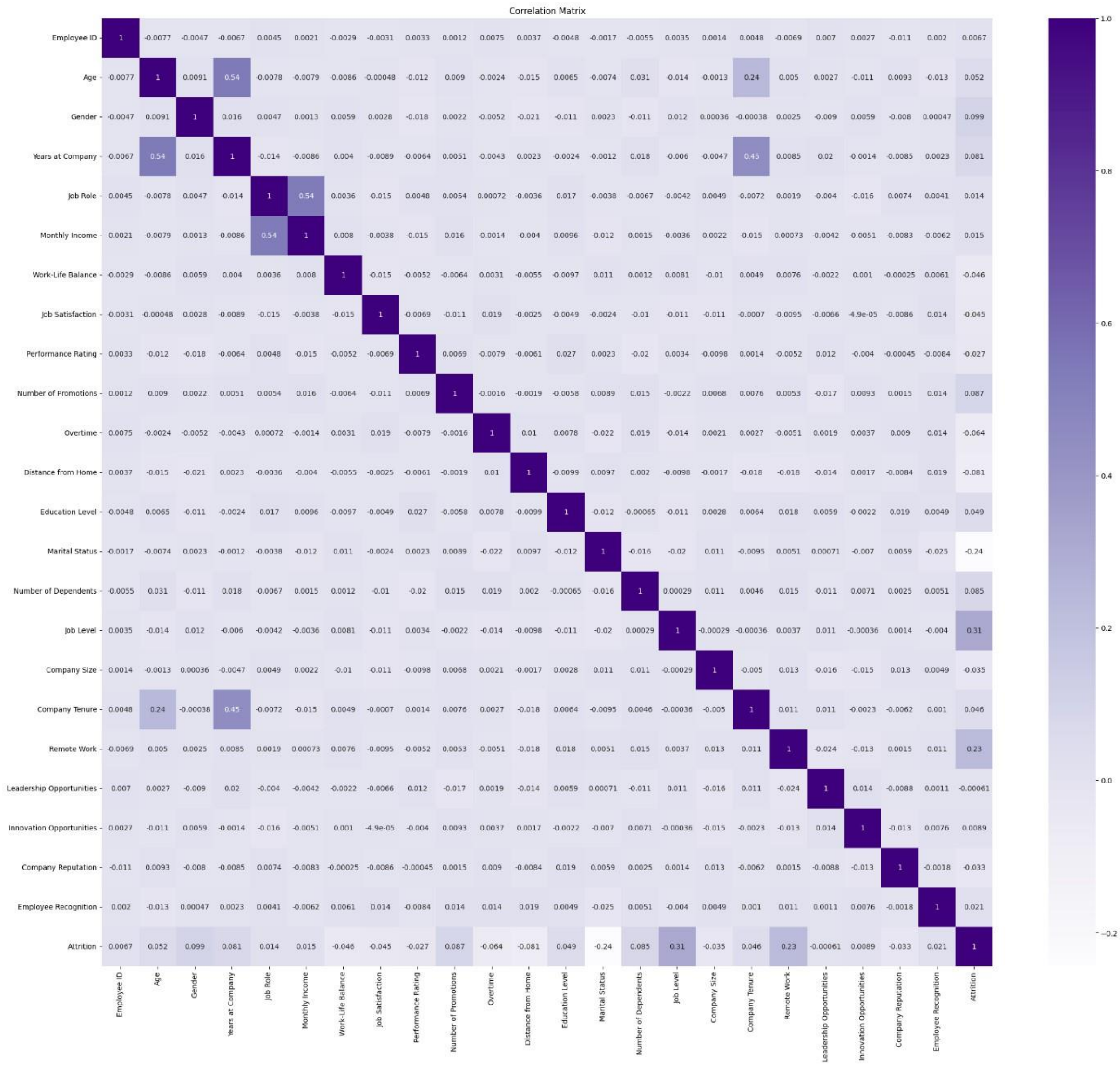




More EDA Visuals




Distribution of Age
Count
Age
0
100
200
300
400
500
600
20
30
40
50
60



Data Splitting

```
[29] #Libraries
     from sklearn.model_selection import train_test_split
```

```
x = data.drop('Attrition', axis=1)
y = data['Attrition']

x_train, x_test, y_train, y_test = train_test_split(x,y, test_size=0.2, random_state=42)
```

```
#Shape of the train and test data
x_train.shape
```

```
(6548, 23)
```

```
[32] y_train.shape
```

```
(6548,)
```

```
[33] x_test.shape
```

```
(1637, 23)
```

```
[34] y_test.shape
```

```
(1637,)
```

Model Training

The primary goal of the dataset is to classify employees into two categories: those who stayed and those who left the company. Logistic regression is specifically designed for binary classification tasks, making it ideal for distinguishing between these two outcomes. It estimates the probability that an employee falls into one of the two categories

```
[25] #Libraries
     from sklearn.linear_model import LogisticRegression
     from sklearn import metrics

[26] model = LogisticRegression()

[27] model.fit(x_train, y_train)

     LogisticRegression
     LogisticRegression()

[28] expected = y_train
     predicted = model.predict(x_train)
```

Model Evaluation

```
print(metrics.classification_report(expected, predicted))

              precision    recall  f1-score   support

           0       0.66      0.65      0.65      3126
           1       0.68      0.70      0.69      3422

    accuracy                           0.67      6548
   macro avg       0.67      0.67      0.67      6548
weighted avg       0.67      0.67      0.67      6548

[30] print(metrics.confusion_matrix(expected, predicted))

     [[2020 1106]
      [1039 2383]]
```

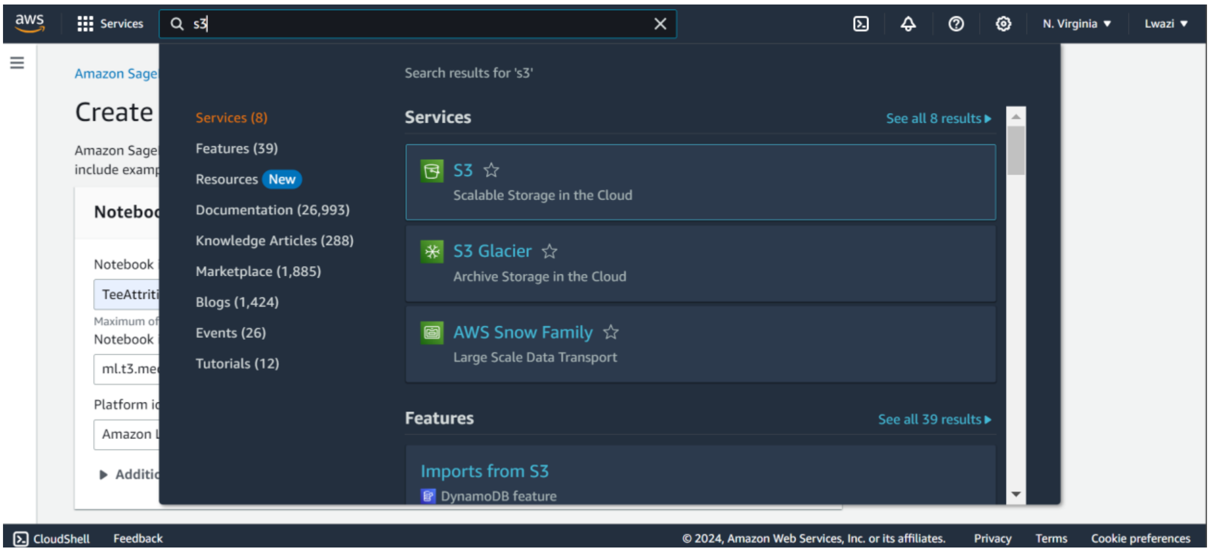
From the above metrics snippets, the model shows a balanced performance in classifying between the two classes. For the ‘0’, the precision is 66% and the recall is 65%, indicating a moderate performance in identifying the ‘0’ class. The ‘1’ class, however, there is a better performance in detecting this class, with a precision and recall of 68% and 70%, respectively. From this, it can be observed that the model has more difficulty predicting the stayed class compared to the left class.

Amazon Web Service

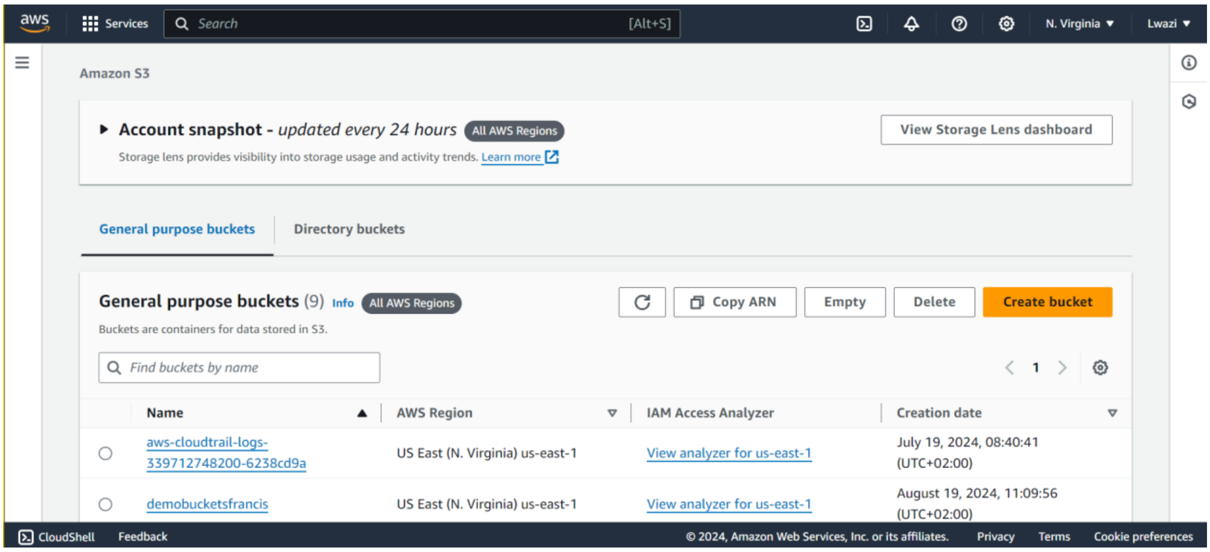
Creating an S3 Bucket:

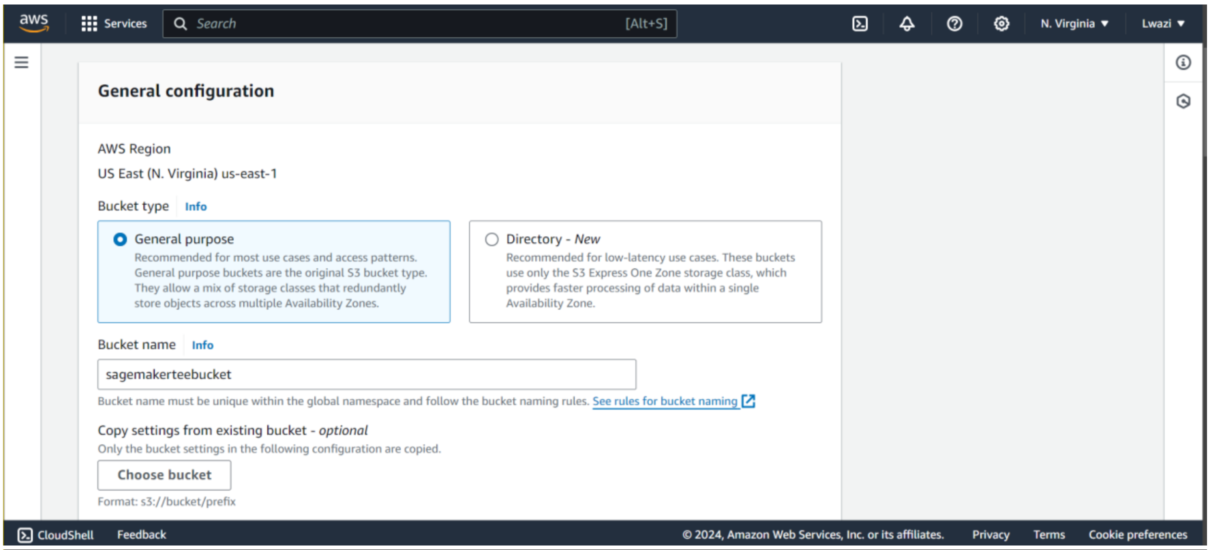
An Amazon S3 bucket named ‘sagemakerteebucket’ is created to store the datasets and model files used in the project. The configuration is set to general purpose, which is recommended for most use cases and access patterns. This type of bucket allows for the distribution of storage across multiple availability zones, which ensures redundancy and reliability. The bucket is created in the US East N. Virginia) region. The bucket will be used for managing the input and output data for the AWS SageMaker model training and deployment process.

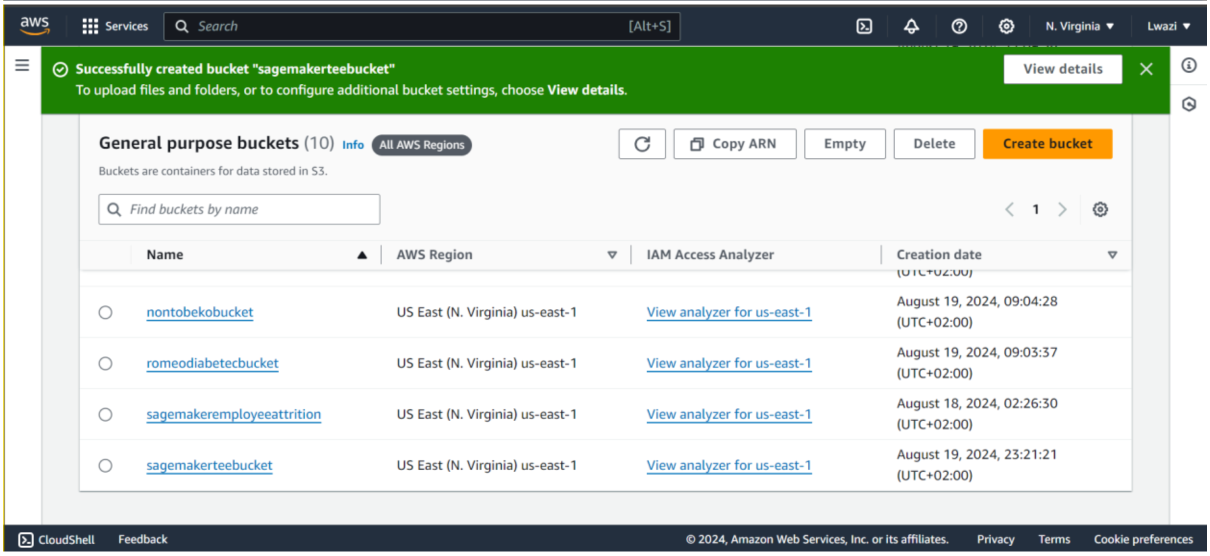
On the search bar in the aws console dashboard, search for S3:



Create a new bucket:



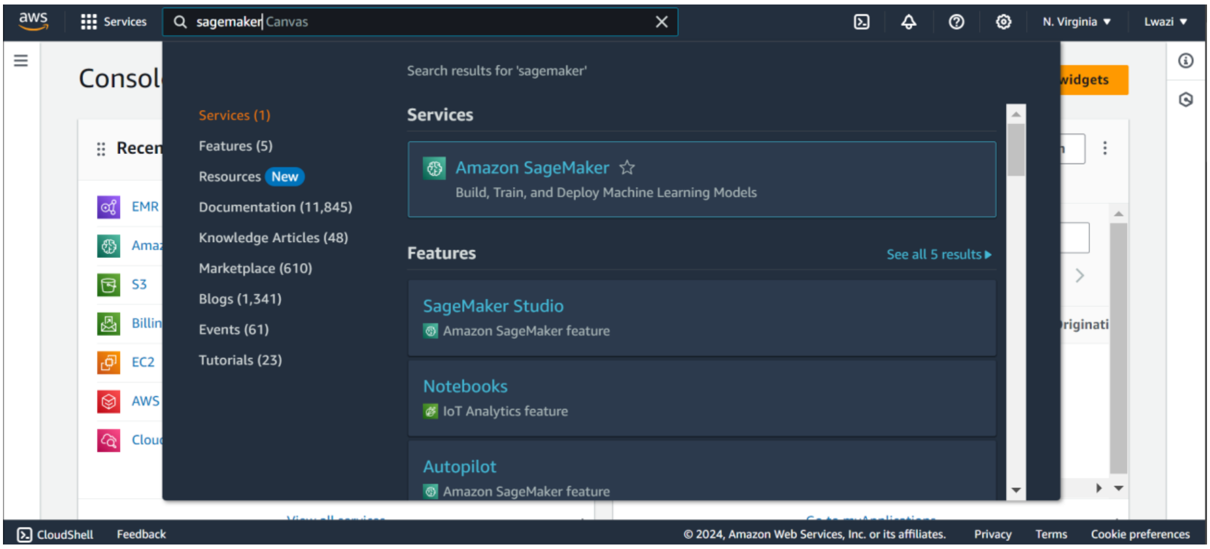

aws
Services
Search
[Alt+S]
N. Virginia
Lwazi
General configuration
AWS Region
US East (N. Virginia) us-east-1
Bucket type Info
General purpose
Recommended for most use cases and access patterns. General purpose buckets are the original S3 bucket type. They allow a mix of storage classes that redundantly store objects across multiple Availability Zones.
Directory - New
Recommended for low-latency use cases. These buckets use only the S3 Express One Zone storage class, which provides faster processing of data within a single Availability Zone.
Bucket name Info
sagemakerteebucket
Bucket name must be unique within the global namespace and follow the bucket naming rules. See rules for bucket naming
Copy settings from existing bucket - optional
Only the bucket settings in the following configuration are copied.
Choose bucket
Format: s3://bucket/prefix
CloudShell
Feedback
© 2024, Amazon Web Services, Inc. or its affiliates.
Privacy
Terms
Cookie preferences


aws
Services
Search
[Alt+S]
N. Virginia
Lwazi
Successfully created bucket "sagemakerteebucket"
To upload files and folders, or to configure additional bucket settings, choose View details.
View details
General purpose buckets (10) Info
All AWS Regions
Copy ARN
Empty
Delete
Create bucket
Buckets are containers for data stored in S3.
Find buckets by name
1
Name
AWS Region
IAM Access Analyzer
Creation date
nontobekobucket
US East (N. Virginia) us-east-1
View analyzer for us-east-1
August 19, 2024, 09:04:28 (UTC+02:00)
romeodiabetecbucket
US East (N. Virginia) us-east-1
View analyzer for us-east-1
August 19, 2024, 09:03:37 (UTC+02:00)
sagemakeremployeeattrition
US East (N. Virginia) us-east-1
View analyzer for us-east-1
August 18, 2024, 02:26:30 (UTC+02:00)
sagemakerteebucket
US East (N. Virginia) us-east-1
View analyzer for us-east-1
August 19, 2024, 23:21:21 (UTC+02:00)
CloudShell
Feedback
© 2024, Amazon Web Services, Inc. or its affiliates.
Privacy
Terms
Cookie preferences

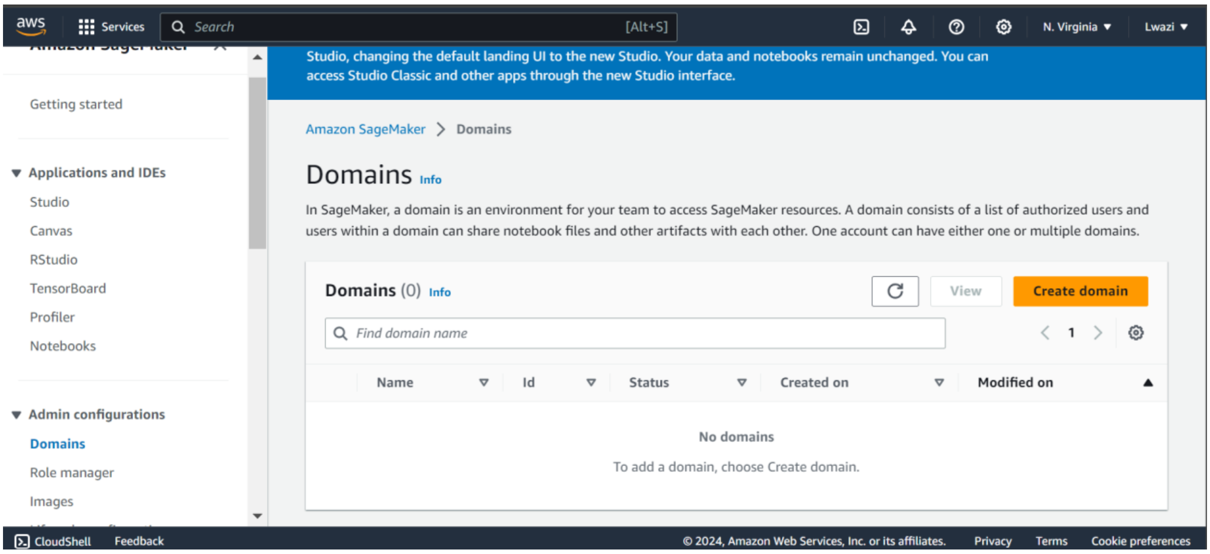
Amazon SageMaker

The following captures the process of setting up a new notebook instance in AWS SageMaker. The instance is named TeeAttrition and is configured to use the 'ml.t3.medium' instance type, which provides a balance of compute, memory and networking resources. This notebook instance is the environment where the model training and evaluation will be conducted.

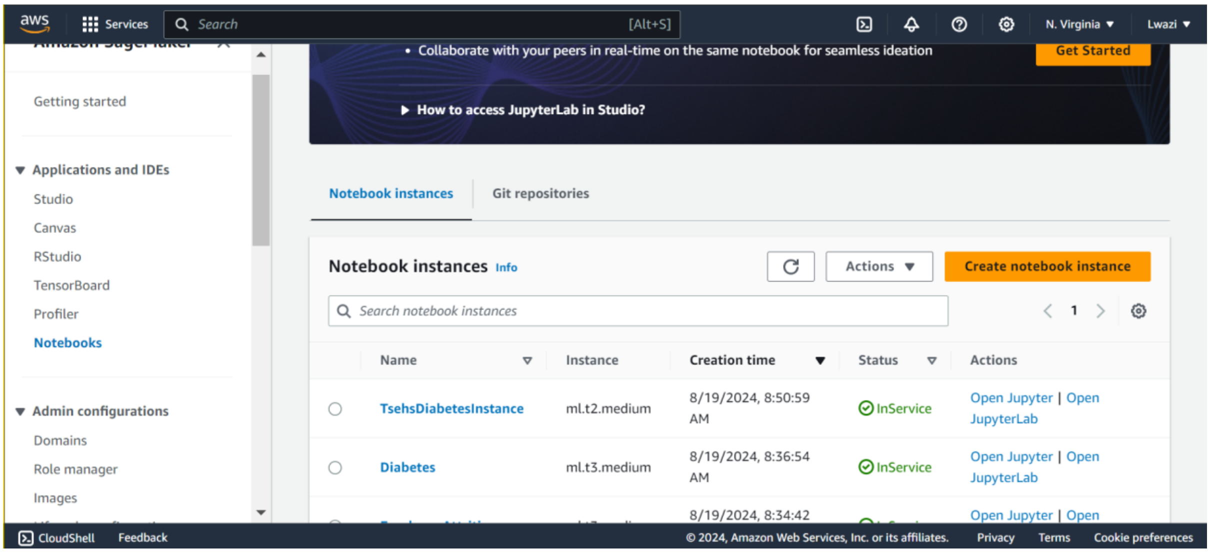
On the search bar in the aws console dashboard, search for sagemaker:

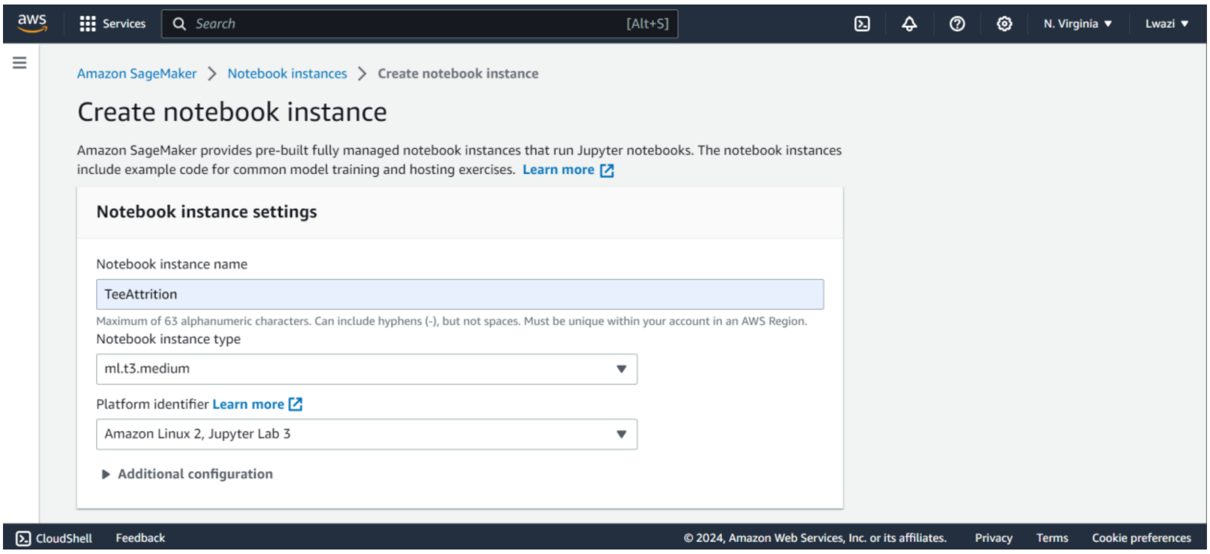


On the side bar, in the Amazon SageMaker dashboard, navigate to notebooks under Applications and IDEs:



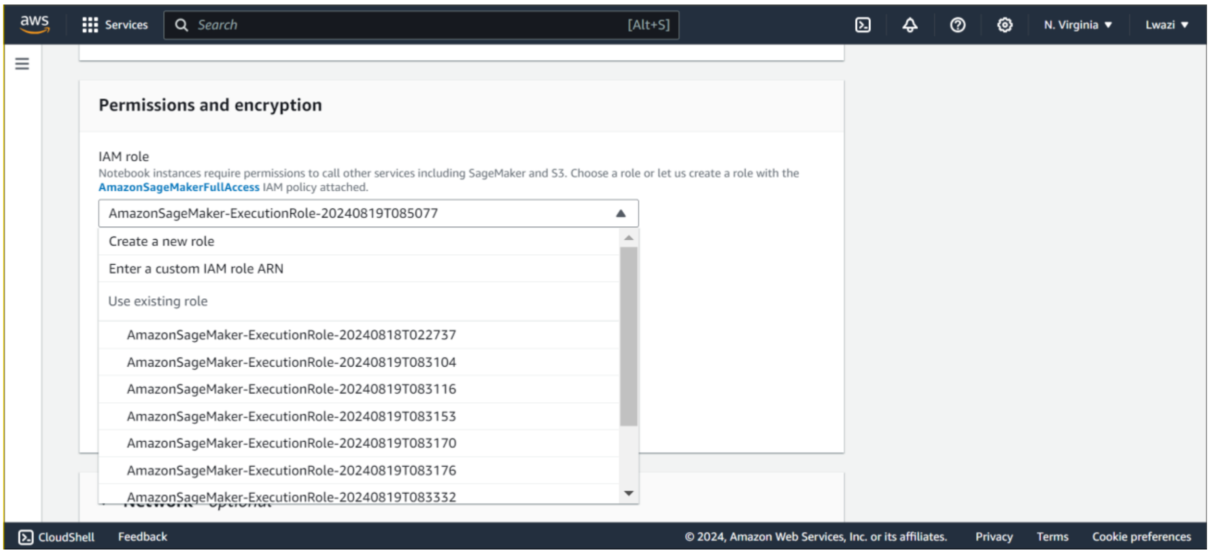
Create a notebook instance:

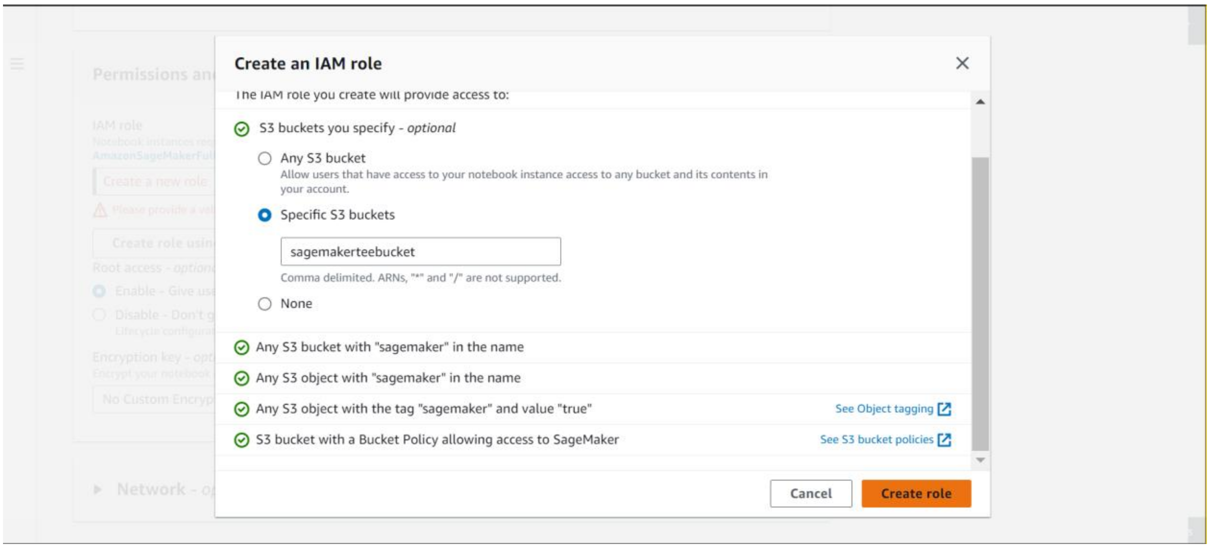


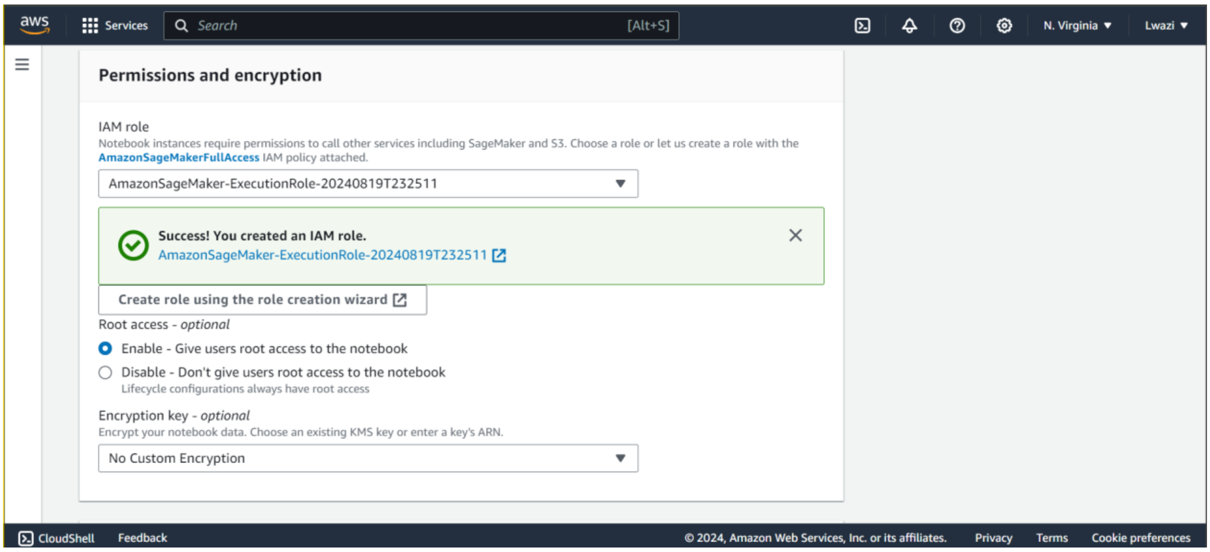


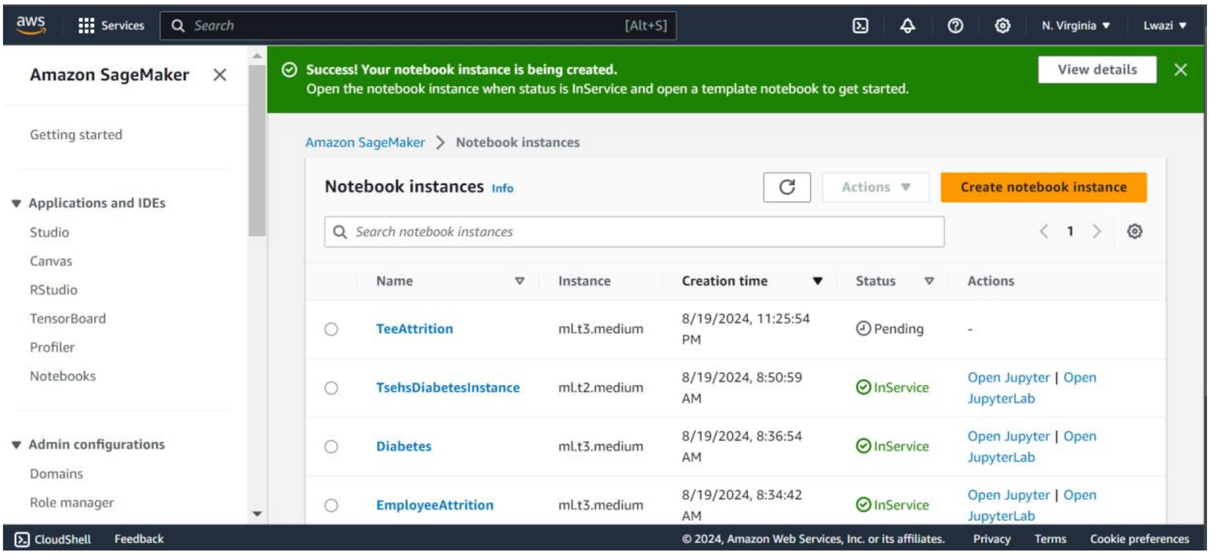
Creating an IAM role:

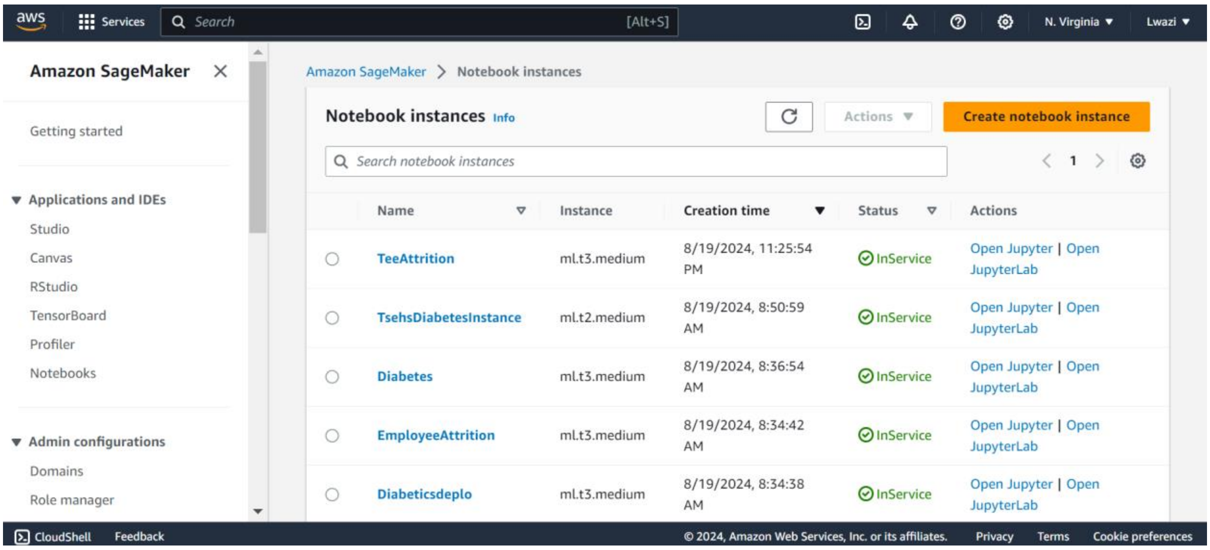
The IAM role allows the SageMaker notebook instance to access the selected S3 bucket. The role is restricted to a specific bucket, which ensures that the SageMaker instance can read and write only to the designated storage loaction. This is important for handling input datasets and storing the model outputs.


aws
Services
Search
[Alt+S]
N. Virginia
Lwazi
Permissions and encryption
IAM role
Notebook instances require permissions to call other services including SageMaker and S3. Choose a role or let us create a role with the AmazonSageMakerFullAccess IAM policy attached.
AmazonSageMaker-ExecutionRole-20240819T085077
Create a new role
Enter a custom IAM role ARN
Use existing role
AmazonSageMaker-ExecutionRole-20240818T022737
AmazonSageMaker-ExecutionRole-20240819T083104
AmazonSageMaker-ExecutionRole-20240819T083116
AmazonSageMaker-ExecutionRole-20240819T083153
AmazonSageMaker-ExecutionRole-20240819T083170
AmazonSageMaker-ExecutionRole-20240819T083176
AmazonSageMaker-ExecutionRole-20240819T083332
CloudShell
Feedback
© 2024, Amazon Web Services, Inc. or its affiliates.
Privacy
Terms
Cookie preferences

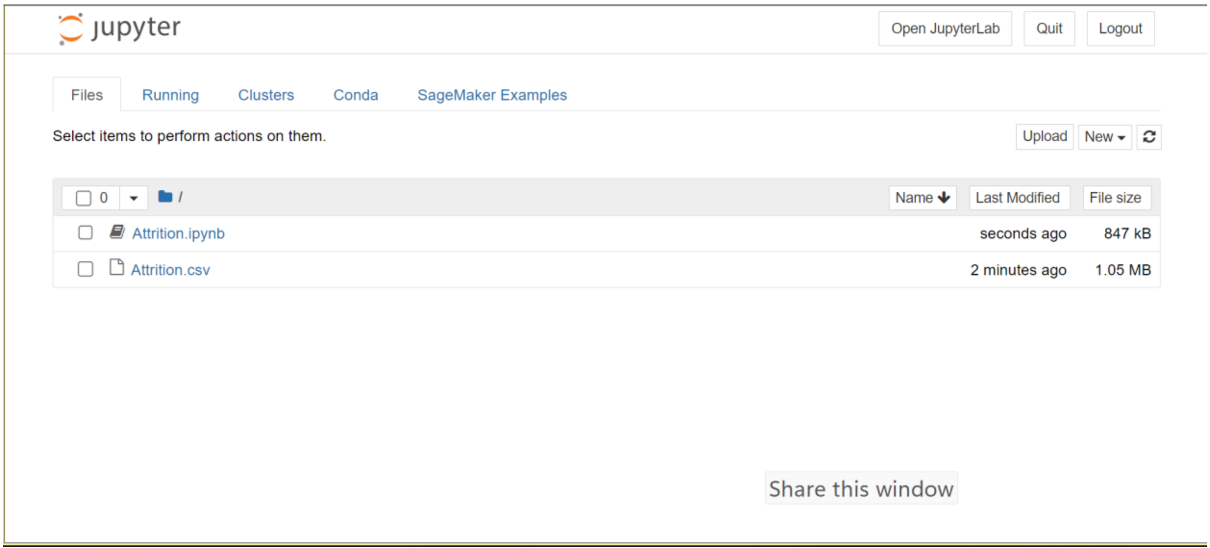

Create an IAM role
The IAM role you create will provide access to:
S3 buckets you specify - optional
Any S3 bucket
Allow users that have access to your notebook instance access to any bucket and its contents in your account.
Specific S3 buckets
sagemakerteebucket
Comma delimited. ARNs, "*" and "/" are not supported.
None
Any S3 bucket with "sagemaker" in the name
Any S3 object with "sagemaker" in the name
Any S3 object with the tag "sagemaker" and value "true"
See Object tagging
S3 bucket with a Bucket Policy allowing access to SageMaker
See S3 bucket policies
Cancel
Create role


aws
Services
Search
[Alt+S]
N. Virginia
Lwazi
Permissions and encryption
IAM role
Notebook instances require permissions to call other services including SageMaker and S3. Choose a role or let us create a role with the AmazonSageMakerFullAccess IAM policy attached.
AmazonSageMaker-ExecutionRole-20240819T232511
Success! You created an IAM role.
AmazonSageMaker-ExecutionRole-20240819T232511
Create role using the role creation wizard
Root access - optional
Enable - Give users root access to the notebook
Disable - Don't give users root access to the notebook
Lifecycle configurations always have root access
Encryption key - optional
Encrypt your notebook data. Choose an existing KMS key or enter a key's ARN.
No Custom Encryption
CloudShell
Feedback
© 2024, Amazon Web Services, Inc. or its affiliates.
Privacy
Terms
Cookie preferences


aws
Services
Search
[Alt+S]
N. Virginia
Lwazi
Amazon SageMaker
Getting started
Applications and IDEs
Studio
Canvas
RStudio
TensorBoard
Profiler
Notebooks
Admin configurations
Domains
Role manager
Success! Your notebook instance is being created.
Open the notebook instance when status is InService and open a template notebook to get started.
View details
Amazon SageMaker > Notebook instances
Notebook instances Info
Actions
Create notebook instance
Search notebook instances
Name
Instance
Creation time
Status
Actions
TeeAttrition
ml.t3.medium
8/19/2024, 11:25:54 PM
Pending
TsehsDiabetesInstance
ml.t2.medium
8/19/2024, 8:50:59 AM
InService
Open Jupyter | Open JupyterLab
Diabetes
ml.t3.medium
8/19/2024, 8:36:54 AM
InService
Open Jupyter | Open JupyterLab
EmployeeAttrition
ml.t3.medium
8/19/2024, 8:34:42 AM
InService
Open Jupyter | Open JupyterLab
CloudShell
Feedback
© 2024, Amazon Web Services, Inc. or its affiliates.
Privacy
Terms
Cookie preferences



Uploading the attrition dataset together with the google colab notebook to be edited:



Continuing from the data splitting portion of the project. The x and y, test and train data is combined, respectively, for better display.

Data Splitting

In [23]:
```python
#Libraries
from sklearn.model_selection import train_test_split
```

In [24]:
```python
x = data.drop(columns = ['Employee ID', 'Attrition'], axis=1)
y = data['Attrition']

x_train, x_test, y_train, y_test = train_test_split(x,y, test_size=0.2, random_state=42)
```

In [29]:
```python
#Displaying the training data
TrainData = x_train.join(y_train)
TrainData.head()
```

Out[29]:

	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
2926	50	1	38	4	7026	2	1	0	1	0	3	3
3781	23	1	6	0	5002	0	0	0	2	1	14	1
4498	47	1	26	2	8037	2	3	0	0	0	35	1
4141	50	1	23	2	7252	2	2	3	0	0	96	2
5245	49	1	28	4	11314	1	0	0	0	0	14	1

In [30]:
```python
#Display all the test data
TestData=x_test.join(y_test)
TestData.head()
```

Out[30]:

	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
5065	28	0	12	0	4638	1	2	0	0	0	61	1
5871	18	0	2	1	7769	1	3	0	0	0	39	1
3011	21	1	3	2	8030	3	2	1	2	0	71	4
3065	21	0	1	0	4268	2	2	0	0	1	54	0
1320	55	1	16	2	7527	2	1	0	1	0	10	3

In [37]:
```python
# Saving the trained data
TrainData.to_csv('TrainData.csv', index=False, index_label='Row', header=False)
```

In [38]:
```python
#Saving the test data
TestData.to_csv('TestData.csv', index=False, index_label='Row', header=False)
```

jupyter — Open JupyterLab | Quit | Logout

Files | Running | Clusters | Conda | SageMaker Examples

Select items to perform actions on them. — Upload | New

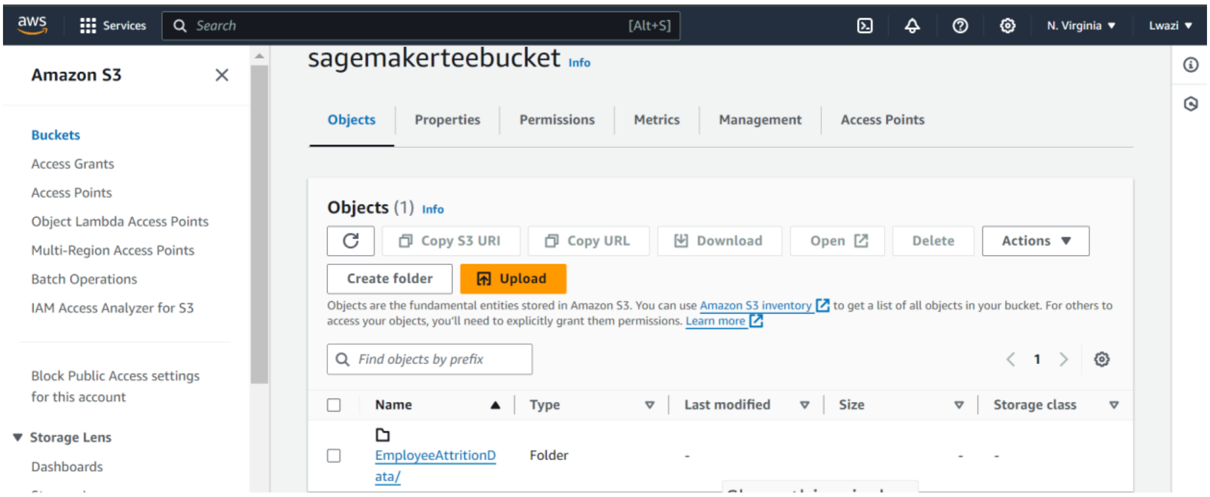
0 /		Name	Last Modified	File size
Attrition.ipynb	Running		a minute ago	910 kB
Attrition.csv			23 minutes ago	1.05 MB
TestData.csv			seconds ago	86.2 kB
TrainData.csv			a minute ago	345 kB

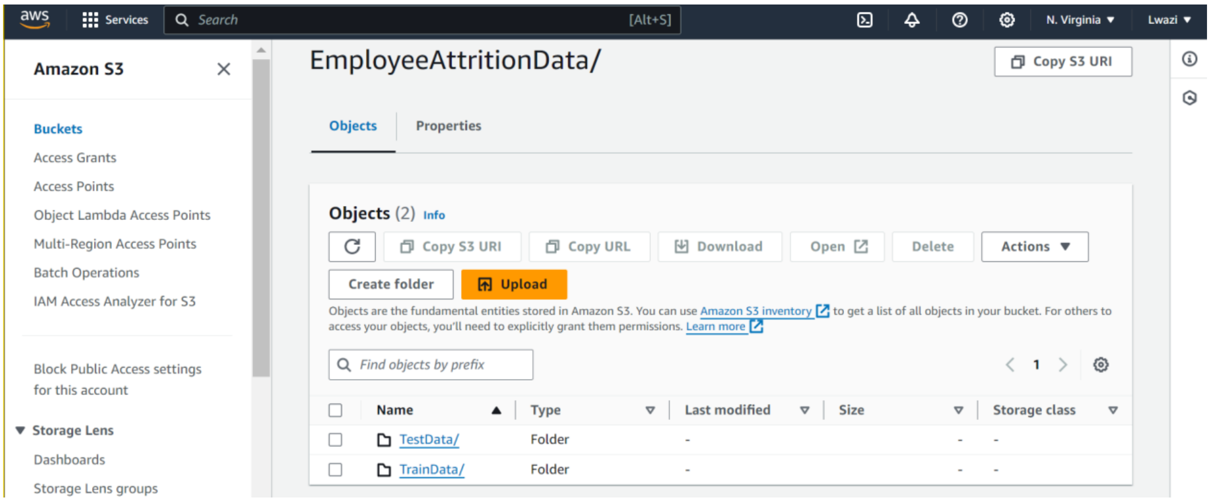
The saved data can be see on the notebook home list.

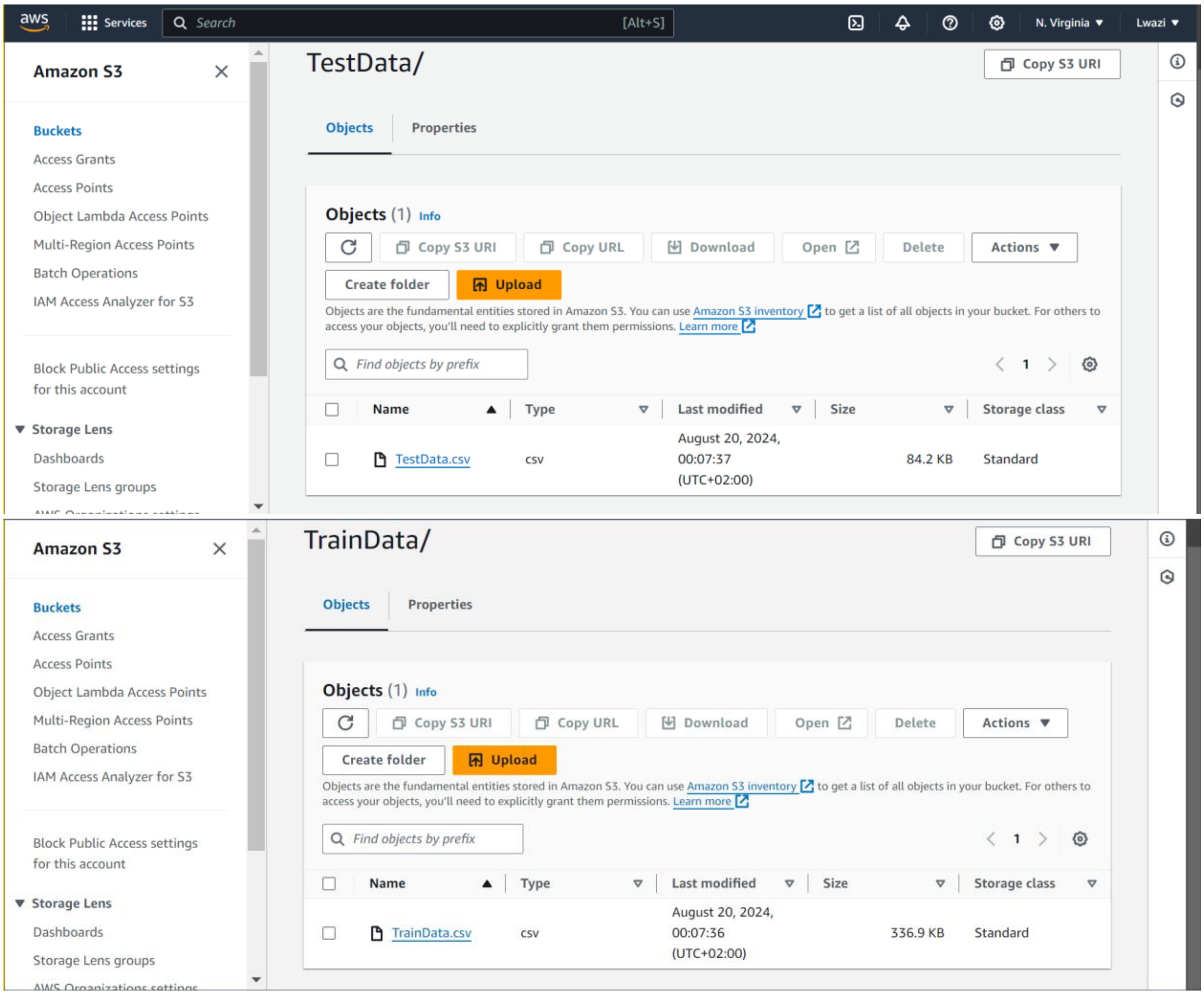
From there, the boto3 library is used to upload the files to the S3 bucket. The paths to the training and testing datasets are defined, their S3 paths are constructed, then they are uploaded to the to the designated S3 bucket loactions.

```
In [38]: #Saving the test data
         TestData.to_csv('TestData.csv', index=False, index_label='Row', header=False)

In [ ]:  #Uploading the trained and test data to Amazon S3 bucket

In [39]: #Necessary packages
         import boto3
         import re

In [40]: bucketName = 'sagemakerteebucket'
         TrainFile = r'EmployeeAttritionData/TrainData/TrainData.csv'
         TestFile = r'EmployeeAttritionData/TestData/TestData.csv'
         ValFile = r'EmployeeAttritionData/ValData/Val.csv'
         ModelFolder = r'EmployeeAttrition/model/'

In [41]: s3ModelOutput = r's3://{0}/{1}'.format(bucketName,ModelFolder)
         s3Train = r's3://{0}/{1}'.format(bucketName,TrainFile)
         s3Test = r's3://{0}/{1}'.format(bucketName,TestFile)
         s3Val = r's3://{0}/{1}'.format(bucketName,ValFile)

In [42]: s3ModelOutput
Out[42]: 's3://sagemakerteebucket/EmployeeAttrition/model/'

In [43]: with open('TrainData.csv', 'rb') as f:
             boto3.Session().resource('s3').Bucket(bucketName).Object(TrainFile).upload_fileobj(f)

In [44]: with open('TestData.csv', 'rb') as f:
             boto3.Session().resource('s3').Bucket(bucketName).Object(TestFile).upload_fileobj(f)
```

From the following snippets, it can be observed that the folders and datasets, have indeed been uploaded on the specified S3 bucket.







To Be Continued...