DEPLOYING EMPLOYEE ATTRITION DATASET USING AWS

INTRODUCTION AND OVERVIEW

The Synthetic Employee Attrition Dataset is a simulated dataset designed for the analysis and prediction of employee attrition. It contains detailed information about various aspects of an employee's profile, including demographics, job-related features, and personal circumstances.

The dataset comprises 74,498 samples, split into training and testing sets to facilitate model development and evaluation. Each record includes a unique Employee ID and features that influence employee attrition. The goal is to understand the factors contributing to attrition and develop predictive models to identify at-risk employees.

This dataset is ideal for HR analytics, machine learning model development, and demonstrating advanced data analysis techniques. It provides a comprehensive and realistic view of the factors affecting employee retention, making it a valuable resource for researchers and practitioners in the field of human resources and organizational development

Purpose: *This document aims to guide stakeholders through the process of deploying a machine learning model trained on the Employee Attrition dataset using AWS*

Audience: *Data scientists, developers, and operations teams involved in model deployment and maintenance.*

SYSTEM ARCHITECTURE

Jupyter Notebook(Used for development and experimentation), AWS SageMaker(Provides a managed environment for training and deploying machine learning models. Components : SageMaker Training, SageMaker Endpoints, SageMaker Model Monitor, and SageMaker Pipelines), Amazon S3 ,IAM Roles.

DEPLOYMENT ENVIRONMENT

Hardware specifications:

System Manufacturer – HP

Processor – 12th Gen Intel(R) Core(TM) i7-1255U, 1700 Mhz, 10 Core(s), 12 Logical Processor(s)

Hardware Abstraction Layer – Version= "10.0.22621.2506"

BIOS Version/Date – AMI F.19, 2023/07/03

RAM – 16.0 GB

Total Physical Memory – 15.7 GB

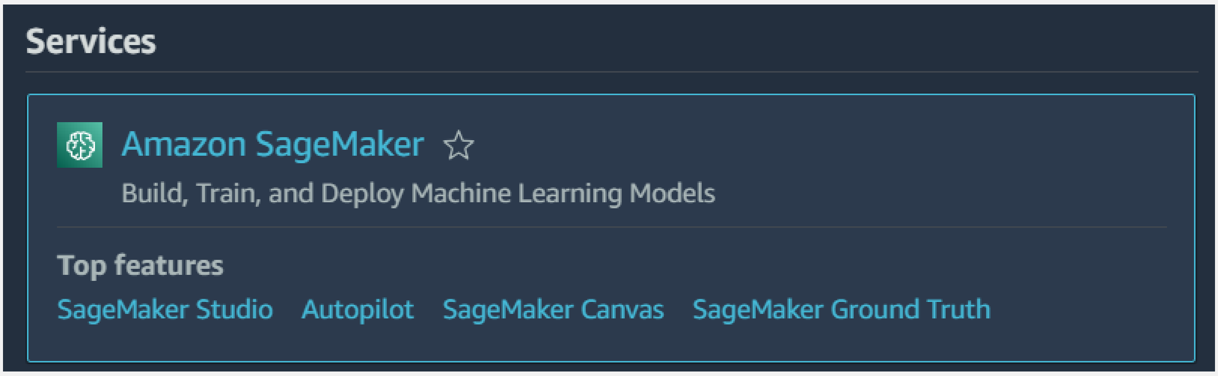
Total Virtual Memory – 32.6 GB

Software dependencies: *When deploying a model on AWS, you need to manage various software dependencies, including machine learning frameworks, model serialization formats, Python packages, and AWS-specific tools and services. Ensuring these dependencies are correctly specified and managed will facilitate a smooth deployment and operation of your model in the AWS environment.*

Operating System: *I am using Microsoft Windows 11 home Single Language*

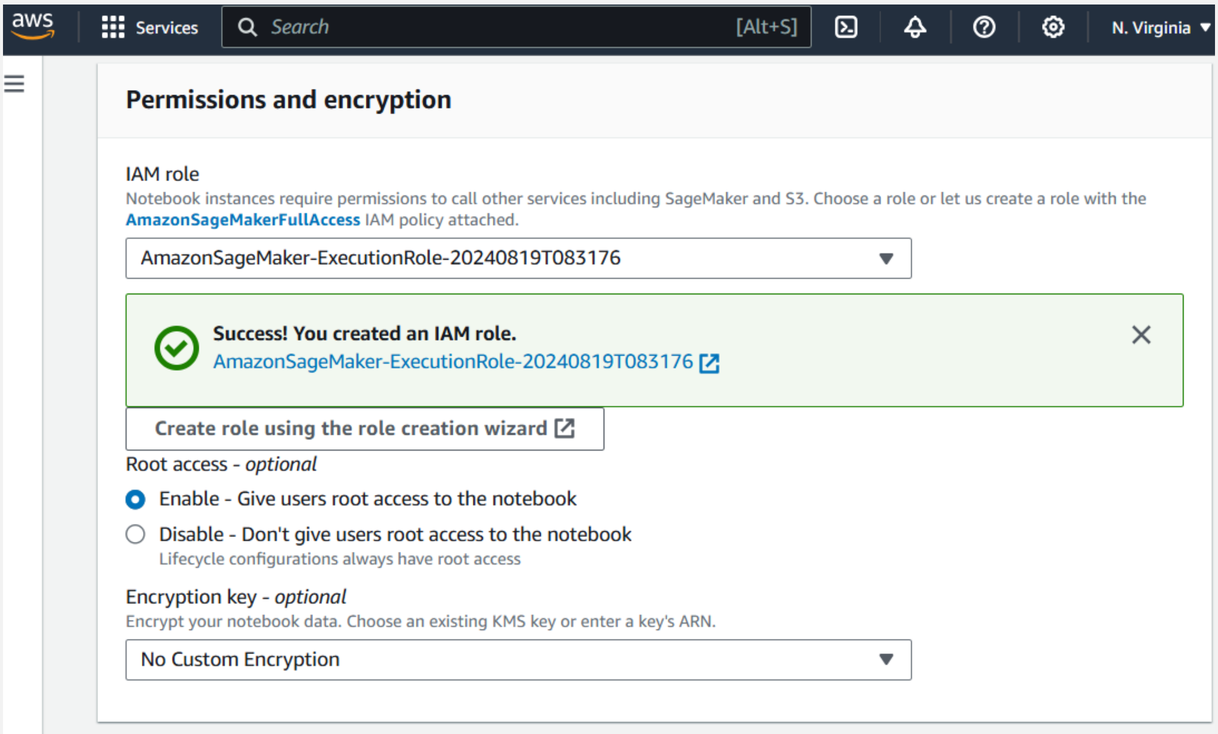
DEPLOYMENT STEPS

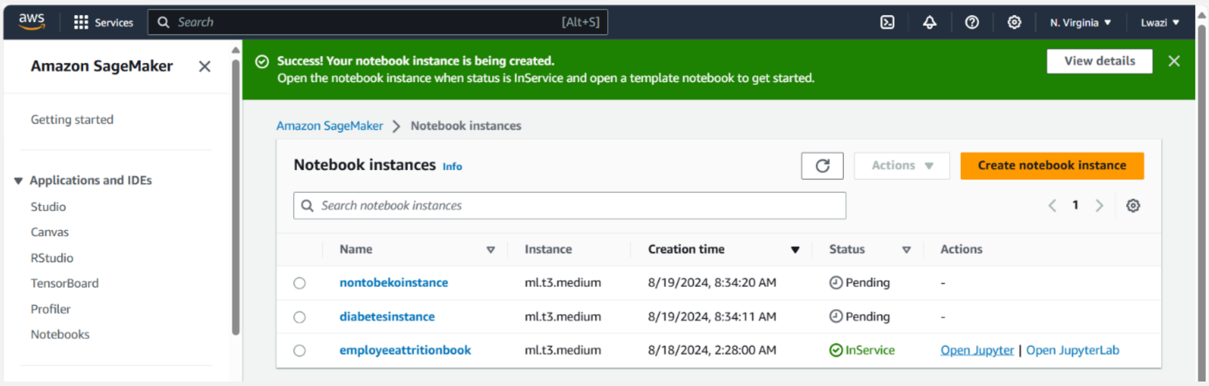
Step 1 : Open AWS SageMaker



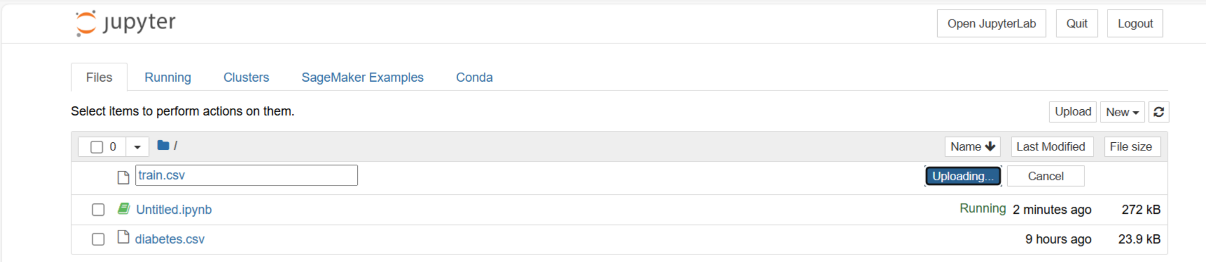
Step 2 : Create Notebook Instance

Create IAM Role

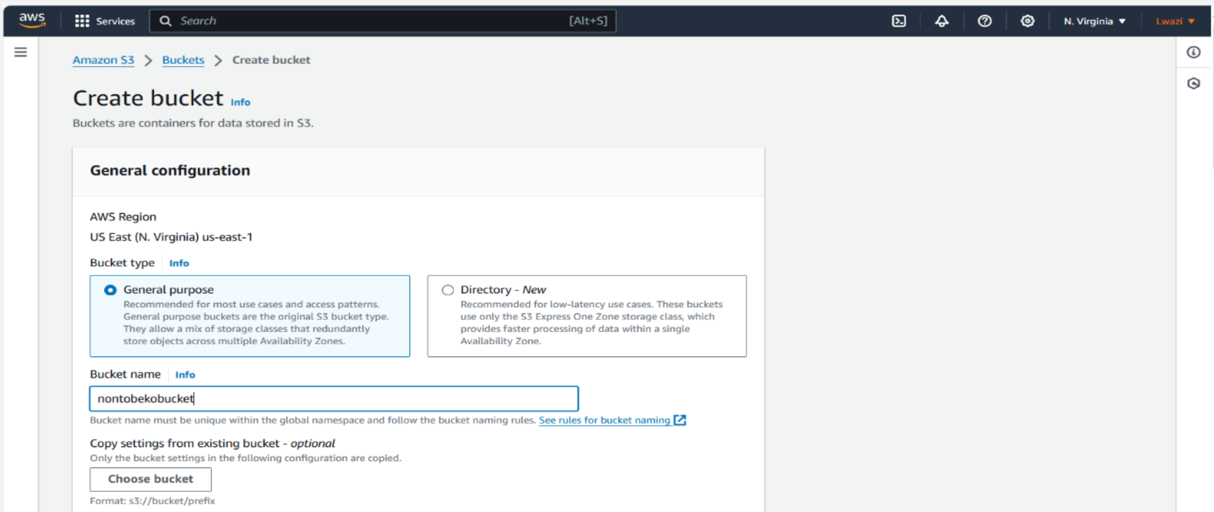


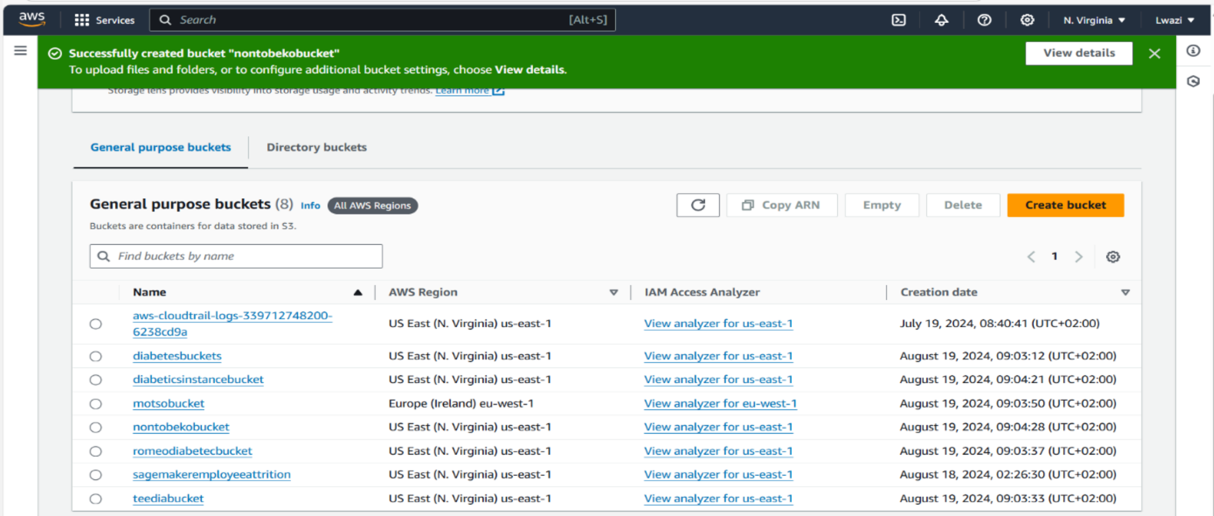


Step 3 : Upload "train.csv" Dataset



Step 4: Create Amazon S3 Bucket





Step 5: Import needed Packages / Libraries

jupyter Untitled Last Checkpoint: 9 hours ago (unsaved changes)

```
In [1]: #importing the packages / libraries that I need

import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
from pandas.plotting import register_matplotlib_converters
register_matplotlib_converters()
import seaborn as sns
pd.set_option('display.max_columns', None)
import warnings
warnings.filterwarnings('ignore')
```

Step 6: Read Dataset

```
In [2]: df = pd.read_csv('train.csv')
        df.head

Out[2]: <bound method NDFrame.head of        Employee ID  Age  Gender  Years at Company    Job Role  Monthly Income  \
        0             8410   31    Male                19   Education            5390
        1            64756   59  Female                 4       Media            5534
        2            30257   24  Female                10  Healthcare            8159
        3            65791   36  Female                 7   Education            3989
        4            65026   56    Male                41   Education            4821
        ...            ...  ...     ...               ...         ...             ...
        59593        37195   50  Female                12   Education            4414
        59594         6266   18    Male                 4  Healthcare            8040
        59595        54887   22  Female                14  Technology            7944
        59596          861   23    Male                 8   Education            2931
        59597        15796   56    Male                19  Technology            6660
```

jupyter Untitled Last Checkpoint: 9 hours ago (autosaved) Logout

File Edit View Insert Cell Kernel Widgets Help Trusted conda_python3

```
       Work-Life Balance Job Satisfaction Performance Rating  \
0              Excellent           Medium            Average
1                   Poor             High                Low
2                   Good             High                Low
3                   Good             High               High
4                   Fair        Very High            Average
...                  ...              ...                ...
59593               Fair             High            Average
59594               Fair             High               High
59595               Fair             High               High
59596               Fair        Very High            Average
59597               Good             High            Average

       Number of Promotions Overtime  Distance from Home    Education Level  \
0                         2       No                  22   Associate Degree
1                         3       No                  21    Master's Degree
2                         0       No                  11  Bachelor's Degree
3                         1       No                  27        High School
4                         0      Yes                  71        High School
...                     ...      ...                 ...                ...
59593                     1      Yes                  66  Bachelor's Degree
59594                     3       No                  42   Associate Degree
59595                     0      Yes                  34    Master's Degree
59596                     0       No                  62  Bachelor's Degree
59597                     0      Yes                  20    Master's Degree

       Marital Status  Number of Dependents Job Level Company Size  \
0             Married                     0       Mid       Medium
1            Divorced                     3       Mid       Medium
2             Married                     3       Mid       Medium
3              Single                     2       Mid        Small
4            Divorced                     0    Senior       Medium
...               ...                   ...       ...          ...
59593          Single                     2    Senior        Small
59594          Single                     0    Senior       Medium
59595         Married                     2     Entry        Small
59596          Single                     0     Entry        Large
59597         Married                     3       Mid       Medium

       Company Tenure Remote Work Leadership Opportunities  \
0                  89          No                       No
1                  21          No                       No
2                  74          No                       No
3                  50         Yes                       No
4                  68          No                       No
...               ...         ...                      ...
59593              35          No                       No
59594              73          No                       No
59595              29          No                      Yes
59596               9          No                       No
59597              81          No                       No

       Innovation Opportunities Company Reputation Employee Recognition  \
0                            No          Excellent               Medium
1                            No               Fair                  Low
2                            No               Poor                  Low
3                            No               Good               Medium
4                            No               Fair               Medium
...                         ...                ...                  ...
59593                       Yes               Poor            Very High
59594                        No               Fair               Medium
59595                        No               Good               Medium
59596                        No               Good                  Low
59597                        No               Good                  Low

       Attrition
0         Stayed
1         Stayed
2         Stayed
3         Stayed
4         Stayed
...          ...
59593       Left
59594       Left
59595     Stayed
59596       Left
59597     Stayed

[59598 rows x 24 columns]>
```

Step 7: Count the empty values for each column

In [3]:
```
#Get the count of the empty values for each columns
df.isna().sum()
```

Out[3]:
```
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
```

Step 8: Get the number of rows and columns

In [4]:
```
#get the number of rows and columns
df.shape
```

Out[4]:
```
(59598, 24)
```

Step 9: Get the number of employees that stayed or left the company

```
In [5]: #Get the number of employees that stayed or left the company
        df['Attrition'].value_counts()

Out[5]: Attrition
        Stayed     31260
        Left       28338
        Name: count, dtype: int64
```

Step 10: Get the data types

```
In [6]: #Get the data types
        df.dtypes

Out[6]: Employee ID                  int64
        Age                          int64
        Gender                      object
        Years at Company             int64
        Job Role                    object
        Monthly Income               int64
        Work-Life Balance           object
        Job Satisfaction            object
        Performance Rating          object
        Number of Promotions         int64
        Overtime                    object
        Distance from Home           int64
        Education Level             object
        Marital Status              object
        Number of Dependents         int64
        Job Level                   object
        Company Size                object
        Company Tenure               int64
        Remote Work                 object
        Leadership Opportunities    object
        Innovation Opportunities    object
        Company Reputation          object
        Employee Recognition        object
        Attrition                   object
        dtype: object
```

Step 11: Check for any missing /null values in the data

```
In [7]: #check for any missing /null values in the data
        df.isnull().values.any()

Out[7]: False
```

Step 12: Get the information about the "train.csv" dataset

```
In [8]: #Get the information about the datasets
        df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 59598 entries, 0 to 59597
Data columns (total 24 columns):
 #   Column                    Non-Null Count  Dtype 
---  ------                    --------------  ----- 
 0   Employee ID               59598 non-null  int64 
 1   Age                       59598 non-null  int64 
 2   Gender                    59598 non-null  object
 3   Years at Company          59598 non-null  int64 
 4   Job Role                  59598 non-null  object
 5   Monthly Income            59598 non-null  int64 
 6   Work-Life Balance         59598 non-null  object
 7   Job Satisfaction          59598 non-null  object
 8   Performance Rating        59598 non-null  object
 9   Number of Promotions      59598 non-null  int64 
 10  Overtime                  59598 non-null  object
 11  Distance from Home        59598 non-null  int64 
 12  Education Level           59598 non-null  object
 13  Marital Status            59598 non-null  object
 14  Number of Dependents      59598 non-null  int64 
 15  Job Level                 59598 non-null  object
 16  Company Size              59598 non-null  object
 17  Company Tenure            59598 non-null  int64 
 18  Remote Work               59598 non-null  object
 19  Leadership Opportunities  59598 non-null  object
 20  Innovation Opportunities  59598 non-null  object
 21  Company Reputation        59598 non-null  object
 22  Employee Recognition      59598 non-null  object
 23  Attrition                 59598 non-null  object
dtypes: int64(8), object(16)
memory usage: 10.9+ MB
```

Step 13: Get all the data types and their unique values

```
In [9]: #Get all the data types and their unique values
        for column in df.columns:
            if df[column].dtype == object:
                print(str(column)+ ' : '+ str(df[column].unique()))
                print(df[column].value_counts())
                print('_________________')
```

```
Gender : ['Male' 'Female']
Gender
Male      32739
Female    26859
Name: count, dtype: int64
_________________
Job Role : ['Education' 'Media' 'Healthcare' 'Technology' 'Finance']
Job Role
Technology    15507
Healthcare    13642
Education     12490
Media          9574
Finance        8385
Name: count, dtype: int64
_________________
Work-Life Balance : ['Excellent' 'Poor' 'Good' 'Fair']
Work-Life Balance
Good         22528
Fair         18046
```

Step 14: Summary of statistics for the numerical columns in the DataFrame

In [10]:
```
df.describe()
```

Out[10]:

	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

Step 15: Check whether there are any missing values in each column

In [11]:
```
df.isna().any(axis=0)
```

Out[11]:
```
Employee ID                 False
Age                         False
Gender                      False
Years at Company            False
Job Role                    False
Monthly Income              False
Work-Life Balance           False
Job Satisfaction            False
Performance Rating          False
Number of Promotions        False
Overtime                    False
Distance from Home          False
Education Level             False
Marital Status              False
Number of Dependents        False
Job Level                   False
Company Size                False
Company Tenure              False
Remote Work                 False
Leadership Opportunities    False
Innovation Opportunities    False
Company Reputation          False
Employee Recognition        False
Attrition                   False
dtype: bool
```

Step 16: Convert attrition to label

In [12]:
```
#lets convert this attrition to label
from sklearn.preprocessing import LabelEncoder
le = LabelEncoder()
df['Attrition']= le.fit_transform(df['Attrition'])
df.head(20)
```

Out[12]:

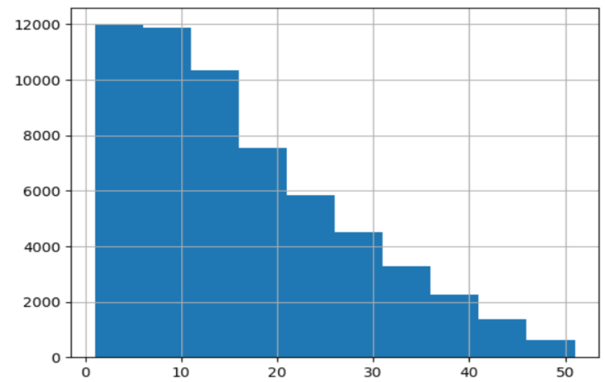
r of ons	Overtime	Distance from Home	Education Level	Marital Status	Number of Dependents	Job Level	Company Size	Company Tenure	Remote Work	Leadership Opportunities	Innovation Opportunities	Company Reputation	Employee Recognition	Attrition
2	No	22	Associate Degree	Married	0	Mid	Medium	89	No	No	No	Excellent	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
3	No	37	Bachelor's Degree	Married	0	Mid	Medium	47	No	No	Yes	Fair	High	0
1	Yes	75	High School	Divorced	3	Entry	Small	93	No	No	No	Good	Medium	0
2	No	5	Master's Degree	Married	4	Entry	Medium	88	No	No	No	Excellent	Low	1
1	Yes	39	High School	Married	4	Entry	Medium	75	No	No	No	Fair	Medium	1
1	Yes	57	PhD	Single	4	Entry	Large	45	No	No	Yes	Good	Low	0
1	No	51	High School	Single	1	Entry	Small	17	No	No	No	Good	Medium	0
2	No	26	Master's Degree	Single	0	Mid	Medium	38	No	No	No	Poor	Medium	0

In [13]:
```
stayed = df.Attrition ==0
left = df.Attrition ==1
```

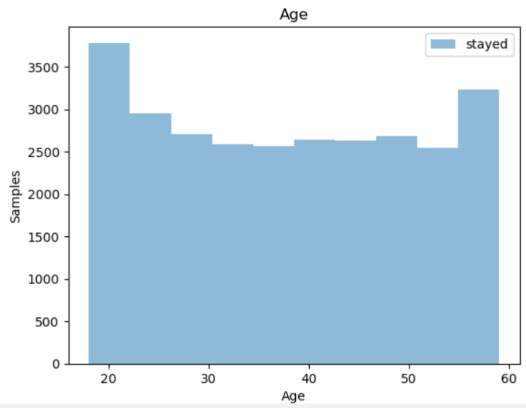
Step 17: Plot Graphs

Histograms based on the number of years in the company and Attrition:

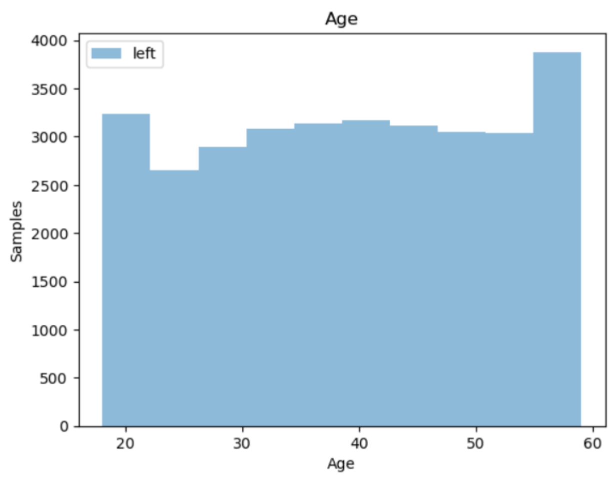
In [14]:
```
df['Years at Company'].hist()
plt.show()
```



In [15]:
```
plt.hist(df[stayed].Age, alpha=0.5, label = 'stayed')
plt.title('Age')
plt.xlabel('Age')
plt.ylabel('Samples')
plt.legend()
plt.show()
```

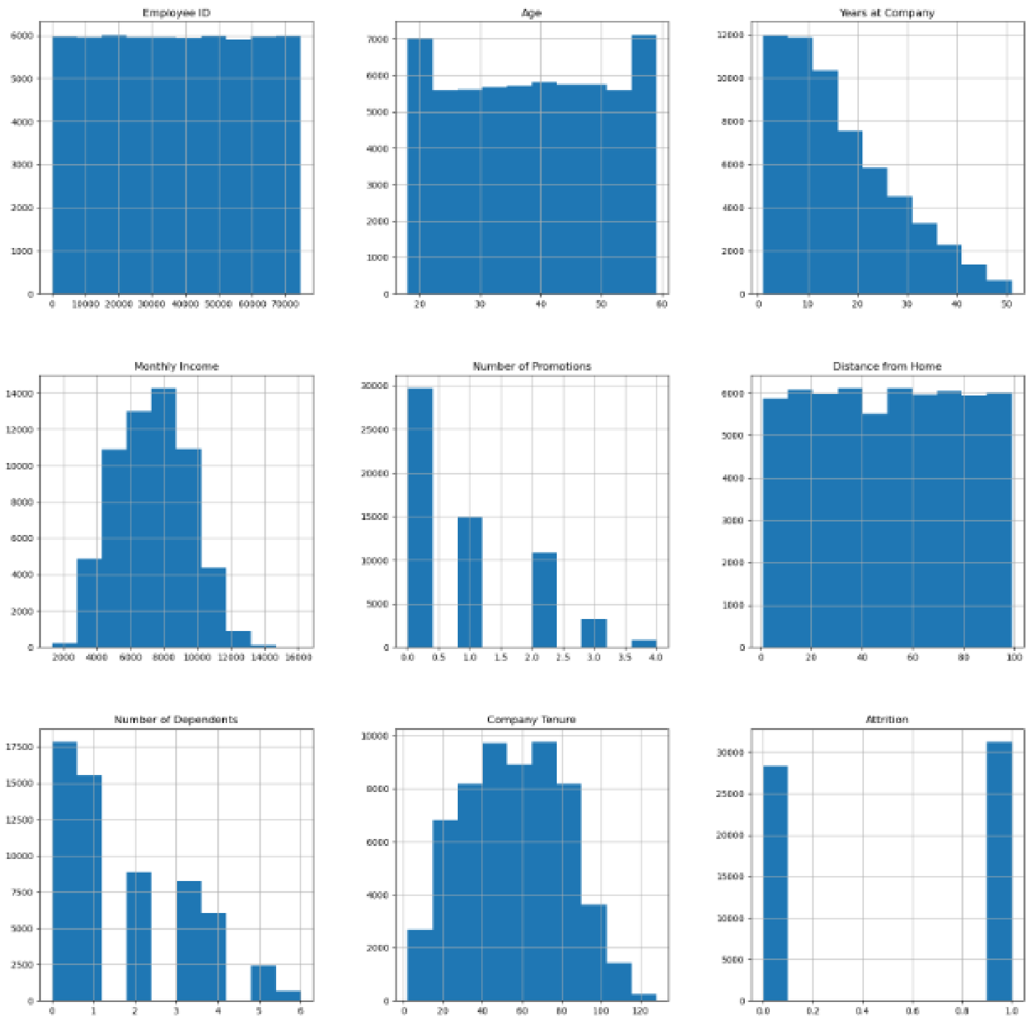


In [16]:
```
plt.hist(df[left].Age, alpha=0.5, label = 'left')
plt.title('Age')
plt.xlabel('Age')
plt.ylabel('Samples')
plt.legend()
plt.show()
```



Plotting Distributions:

In [17]:
```
#plotting the distributions
p = df.hist(figsize=(20,20))
```



Step 18: Convert categorical labels into numerical values using LabelEncoder

In [18]:
```
le.classes_
```

Out[18]:
```
array(['Left', 'Stayed'], dtype=object)
```

In [19]:
```
Attrition_Employee = le.classes_
print(Attrition_Employee)
```

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

Step 19: Split Data into x and y

```
In [22]: #Splitting Data
         x = df.iloc[:,:-1]
         #Remove the last column diabetic
```

```
In [23]: x.head()
```

Out[23]:

	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 Depender
0	8410	31	Male	19	Education	5390	Excellent	Medium	Average	2	No	22	Associate Degree	Married	
1	64756	59	Female	4	Media	5534	Poor	High	Low	3	No	21	Master's Degree	Divorced	
2	30257	24	Female	10	Healthcare	8159	Good	High	Low	0	No	11	Bachelor's Degree	Married	
3	65791	36	Female	7	Education	3989	Good	High	High	1	No	27	High School	Single	
4	65026	56	Male	41	Education	4821	Fair	Very High	Average	0	Yes	71	High School	Divorced	

```
In [24]: y=df.iloc[:,-1]
```

```
In [25]: y.head()
```

```
Out[25]: 0    1
         1    1
         2    1
         3    1
         4    1
         Name: Attrition, dtype: int64
```

Step 20: xTrain and yTrain

```
In [26]: from sklearn.model_selection import train_test_split
```

```
In [27]: xTrain, xTest , yTrain, yTest = train_test_split(x,y,test_size=0.2)
```

```
In [28]: xTrain.head(5)
```

Out[28]:

	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	Nu Depe
656	33881	44	Male	23	Media	6288	Fair	High	Low	0	No	69	Associate Degree	Single	
42724	68010	47	Male	36	Technology	9150	Good	Low	Average	0	Yes	79	Bachelor's Degree	Divorced	
1739	22195	54	Male	13	Finance	8973	Good	Low	High	3	No	24	Bachelor's Degree	Divorced	
26376	14885	59	Male	4	Media	6325	Good	Medium	Below Average	0	Yes	42	Master's Degree	Married	
10397	69384	44	Male	16	Finance	10063	Poor	Low	High	2	No	94	Master's Degree	Married	

```
In [29]: yTrain.head(5)
```

```
Out[29]: 656      0
         42724    1
         1739     0
         26376    1
         10397    0
         Name: Attrition, dtype: int64
```

Step 21: The .join() method combines xTrain and yTrain into a single DataFrame, aligning them based on their indices. trainDF contains all columns from xTrain along with the target column from yTrain. This method is useful for consolidating feature and target data into a single DataFrame for easier manipulation, analysis, or model training.

In [30]:
```
trainDF= xTrain.join(yTrain)
trainDF.head()
```
Out[30]:

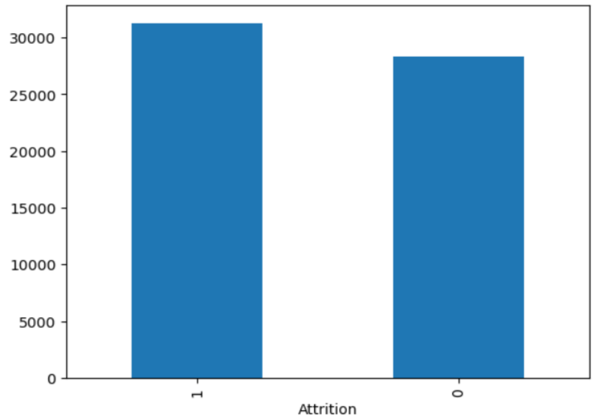
	Employee ID	Age	Gender	Years at Company	Job Role	Monthly Income	Work-Life Balance	Job Satisfaction	Performance Rating	Number of Promotions	Overtime	Distance from Home	Education Level	Marital Status	Nu Depe
656	33881	44	Male	23	Media	6288	Fair	High	Low	0	No	69	Associate Degree	Single	
42724	68010	47	Male	36	Technology	9150	Good	Low	Average	0	Yes	79	Bachelor's Degree	Divorced	
1739	22195	54	Male	13	Finance	8973	Good	Low	High	3	No	24	Bachelor's Degree	Divorced	
26376	14885	59	Male	4	Media	6325	Good	Medium	Below Average	0	Yes	42	Master's Degree	Married	
10397	69384	44	Male	16	Finance	10063	Poor	Low	High	2	No	94	Master's Degree	Married	

In [31]:
```
testDF = xTest.join(yTest)
testDF.head()
```
Out[31]:

	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	Num Depe
12608	37481	56	Male	22	Technology	10554	Poor	High	Below Average	4	No	13	Associate Degree	Married	
35077	24839	46	Male	17	Finance	7244	Poor	Very High	High	1	No	37	Bachelor's Degree	Married	
38445	35575	47	Male	37	Media	5933	Excellent	Medium	High	0	No	97	Associate Degree	Married	
14326	55371	47	Male	13	Finance	9290	Good	High	Average	0	Yes	52	High School	Married	

Step 22: Creates and displays a bar chart showing the counts of each unique value in the Attrition column.

In [33]:
```
color_wheel = {1: "#0392cf", 2: "7bc043"}
colors = df["Attrition"].map(lambda x: color_wheel.get(x+1))
print(df.Attrition.value_counts())
p = df.Attrition.value_counts().plot(kind="bar")
```
```
Attrition
1    31260
0    28338
Name: count, dtype: int64
```



Step 23: Label Encoding

In [35]:
```
#Label encoding
columns = ['Gender', 'Monthly Income', 'Job Role', 'Work-Life Balance', 'Job Satisfaction', 'Performance Rating', 'Overtime', 'Ed
       'Remote Work', 'Leadership Opportunities', 'Innovation Opportunities', 'Company Reputation', 'Employee Recognition', 'Attr
le = LabelEncoder()
df[columns]=df[columns].apply(le.fit_transform)
df.head()
```
Out[35]:

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

Step 24: Predict

In [36]:
```
# Predict target
y = df.Attrition
y.tail()
```

Out[36]:
```
59593    0
59594    0
59595    1
59596    0
59597    1
Name: Attrition, dtype: int64
```

Step 25: Feature Selection

In [37]:
```
# Feature selection
columns = ['Age', 'Gender', 'Years at Company', 'Job Role', 'Monthly Income', 'Work-Life Balance', 'Job Satisfaction', 'Performar
      'Company Tenure', 'Remote Work', 'Leadership Opportunities', 'Innovation Opportunities', 'Company Reputation', 'Employee R
x = df[columns]
x.head()
```

Out[37]:

	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
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

Step 26: Training Data

In [53]:
```
#Displaying the training data
TrainData = xTrain.join(yTrain)
TrainData.head()
```

Out[53]:

	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	Nun Depe
27869	30490	55	Male	28	Healthcare	9185	Fair	Medium	Below Average	0	Yes	23	High School	Single	
30503	47426	26	Female	10	Technology	9719	Fair	High	Average	1	No	28	High School	Married	
43242	33737	56	Male	45	Education	6139	Excellent	High	Average	3	No	61	High School	Single	
6403	25911	41	Female	15	Education	5214	Fair	High	Low	1	No	16	Master's Degree	Married	
16551	45851	30	Male	16	Technology	9935	Fair	High	Average	2	Yes	11	Bachelor's Degree	Married	

Step 27: Testing Data

In [55]:
```
#Display all the testing data
TestData=xTest.join(yTest)
TestData.head()
```

Out[55]:

	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	Nun Depe
34689	49286	28	Male	5	Media	5293	Fair	High	Average	0	Yes	70	High School	Married	
36243	37667	21	Male	5	Media	6818	Good	Very High	High	0	No	19	Associate Degree	Married	
56151	55562	27	Female	4	Healthcare	8749	Fair	High	Average	0	No	65	Master's Degree	Single	
19403	31150	48	Female	35	Technology	9857	Fair	Low	Below Average	3	No	2	Bachelor's Degree	Single	
14843	37912	29	Male	19	Technology	10892	Excellent	High	High	0	No	71	High School	Single	

Step 28: Saving Test and Training Data to CSV Files

```
In [56]: # Saving the trained data
TrainData.to_csv('TrainData.csv', index=False, index_label='Row', header=False)
```

```
In [57]: #Saving the test data
TestData.to_csv('TestData.csv', index=False, index_label='Row', header=False)
```

Step 29: Upload to S3 Bucket, Do the Training and Deployment

```
In [43]: #First lest us upload out trained and test data to s3 bucket or other cloud service
#We do the training and deployment in your sageMaker
#Import needed Libraries
```

```
In [58]: import boto3 #this package is to integrate with s3 bucket or other cloude service//
import re #this package is to folow a strict pattern to save your work/regular expression//
```

```
In [67]: bucketNM = 'nontobekobucket'
TrainFile = r'attritiondata/traineddataattritions/TrainData.csv'
TestFile = r'attritiondata/testddataattritions/TestData.csv'
ValFile = r'attritiondata/Val/Val.csv'
ModelFolder = r'attritiondata/model/'
```

```
In [68]: s3ModelOutput = r's3://{0}/{1}'.format(bucketNM,ModelFolder)
s3Train = r's3://{0}/{1}'.format(bucketNM,TrainFile)
s3Test = r's3://{0}/{1}'.format(bucketNM,TestFile)
s3Val = r's3://{0}/{1}'.format(bucketNM,ValFile)
```

```
In [69]: s3ModelOutput
Out[69]: 's3://nontobekobucket/attritiondata/model/'
```

```
In [71]: with open('TrainData.csv','rb') as f:
    boto3.Session().resource('s3').Bucket(bucketNM).Object(TrainFile).upload_fileobj(f)
```

```
In [72]: with open('TestData.csv','rb') as f:
    boto3.Session().resource('s3').Bucket(bucketNM).Object(TestFile).upload_fileobj(f)
```

```
In [92]: import sagemaker
from sagemaker import get_execution_role
```

```
In [93]: sagemakerSess=sagemaker.Session()
role=get_execution_role()
```

```
In [94]: sagemakerSess.boto_region_name
Out[94]: 'us-east-1'
```

```
In [95]: 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 [96]: 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 [55]: LogisticModel.hyperparameters()
Out[55]: {'predictor_type': 'binary_classifier', 'mini_batch_size': 100}

In [56]: trainConfig=sagemaker.session.s3_input(s3_data=s3Train,content_type='text/csv')

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

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

INFO:sagemaker:Creating training-job with name: Logistic-Demo-v1-2024-08-20-07-47-03-233

2024-08-20 07:47:03 Starting - Starting the training job...
2024-08-20 07:47:17 Starting - Preparing the instances for training...
2024-08-20 07:47:48 Downloading - Downloading input data...
2024-08-20 07:48:14 Downloading - Downloading the training image......
2024-08-20 07:49:30 Training - Training image download completed. Training in progress....Docker entrypoint called with argum
ent(s): train
Running default environment configuration script
[08/20/2024 07:49:53 INFO 140067904063296] Reading default configuration from /opt/amazon/lib/python3.8/site-packages/algorit
hm/resources/default-input.json: {'mini_batch_size': '1000', 'epochs': '15', 'feature_dim': 'auto', 'use_bias': 'true', 'bina
ry_classifier_model_selection_criteria': 'accuracy', 'f_beta': '1.0', 'target_recall': '0.8', 'target_precision': '0.8', 'num
_models': 'auto', 'num_calibration_samples': '10000000', 'init_method': 'uniform', 'init_scale': '0.07', 'init_sigma': '0.0
1', 'init_bias': '0.0', 'optimizer': 'auto', 'loss': 'auto', 'margin': '1.0', 'quantile': '0.5', 'loss_insensitivity': '0.0
1', 'huber_delta': '1.0', 'num_classes': '1', 'accuracy_top_k': '3', 'wd': 'auto', 'l1': 'auto', 'momentum': 'auto', 'learnin
g_rate': 'auto', 'beta_1': 'auto', 'beta_2': 'auto', 'bias_lr_mult': 'auto', 'bias_wd_mult': 'auto', 'use_lr_scheduler': 'tru
e', 'lr_scheduler_step': 'auto', 'lr_scheduler_factor': 'auto', 'lr_scheduler_minimum_lr': 'auto', 'positive_example_weight_m
ult': '1.0', 'balance_multiclass_weights': 'false', 'normalize_data': 'true', 'normalize_label': 'auto', 'unbias_data': 'aut
o', 'unbias_label': 'auto', 'num_point_for_scaler': '10000', '_kvstore': 'auto', '_num_gpus': 'auto', '_num_kv_servers': 'aut
```

```
In [57]: LogisticModel.fit({'train':trainConfig})

[08/20/2024 07:51:08 INFO 140067904063296] #quality_metric: host=algo-1, train binary_log_loss <score>=0.6592329571064
[08/20/2024 07:51:08 INFO 140067904063296] Best model found for hyperparameters: {"optimizer": "adam", "learning_rate": 0.00
5, "l1": 0.0, "wd": 0.0001, "lr_scheduler_step": 10, "lr_scheduler_factor": 0.99, "lr_scheduler_minimum_lr": 1e-05}
[08/20/2024 07:51:08 INFO 140067904063296] Saved checkpoint to "/tmp/tmp7yukcw2n/mx-mod-0000.params"
[08/20/2024 07:51:08 INFO 140067904063296] Test data is not provided.
#metrics {"StartTime": 1724140195.952883, "EndTime": 1724140268.6347115, "Dimensions": {"Algorithm": "Linear Learner", "Hos
t": "algo-1", "Operation": "training"}, "Metrics": {"initialize.time": {"sum": 639.3227577209473, "count": 1, "min": 639.3227
577209473, "max": 639.3227577209473}, "epochs": {"sum": 15.0, "count": 1, "min": 15, "max": 15}, "check_early_stopping.time":
{"sum": 2.131938934326172, "count": 6, "min": 0.16427040100097656, "max": 0.7276535034179688}, "update.time": {"sum": 67776.0
34116745, "count": 6, "min": 11073.944807052612, "max": 11430.325269699097}, "finalize.time": {"sum": 4238.261938095093, "cou
nt": 1, "min": 4238.261938095093, "max": 4238.261938095093}, "setuptime": {"sum": 2.0117759704589844, "count": 1, "min": 2.01
17759704589844, "max": 2.0117759704589844}, "totaltime": {"sum": 72787.25695610046, "count": 1, "min": 72787.25695610046, "ma
x": 72787.25695610046}}}


2024-08-20 07:51:28 Uploading - Uploading generated training model
2024-08-20 07:51:28 Completed - Training job completed
Training seconds: 220
Billable seconds: 220
```

```
In [62]: #Deploying the Trained Model

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

INFO:sagemaker:Creating model with name: Logistic-Demo-v1-2024-08-20-07-58-55-501
INFO:sagemaker:Creating endpoint-config with name LogisticRegression-nontobeko
INFO:sagemaker:Creating endpoint with name LogisticRegression-nontobeko

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

Amazon SageMaker > Endpoints

Endpoints

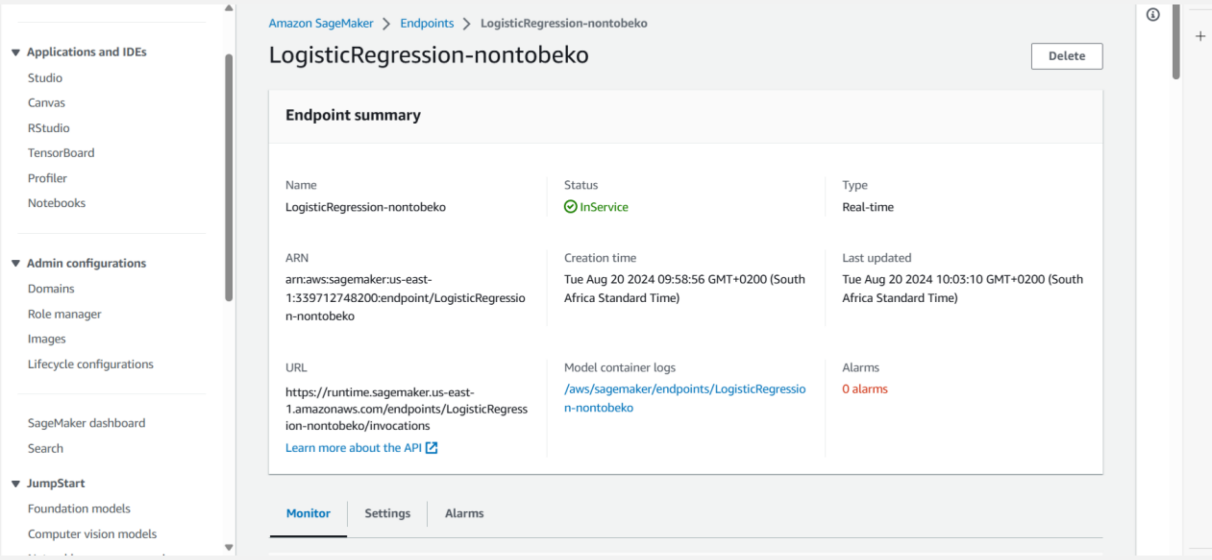
Update endpoint | Actions | Create endpoint

Search endpoints

1

Status = InService | Clear Filters

	Name	ARN	Creation time	Status	Last updated
○	LogisticRegression-nontobeko	arn:aws:sagemaker:us-east-1:339712748200:endpoint/LogisticRegression-nontobeko	8/20/2024, 9:58:56 AM	InService	8/20/2024, 10:03:10 AM



TESTING AND VALIDATION

Procedure for testing the deployed model to ensure it performs as expected:

(a) *Environment Configuration*
(b) *Data Preparation*
(c) *Input Data Validation*
(d) *Testing*
(e) *Prediction Output & Accuracy Assessment*
(f) *Performance Testing*
(g) *Integration Testing*
(h) *Validation Against Baselines*
(i) *Bias and Fairness Testing*
(j) *Documentation of Testing Results*
(k) *Iterative Refinement*

MONITORING AND LOGGING

Monitoring the performance and health of a deployed model is crucial for ensuring it continues to operate effectively and meets service level expectations. AWS provides tools and services that can be leveraged for performance monitoring : AWS SageMaker Model Monitor.

AWS SageMaker Model Monitor - *Provides automatic monitoring of the performance of your machine learning models deployed in SageMaker. It helps you detect data drift, anomalies, and changes in model performance.*

SCALABILITY AND PERFORMANCE

Scalability Considerations

When scaling a machine learning model to handle increased traffic or larger datasets on AWS, several considerations come into play to ensure optimal performance and efficiency:

Compute Resources*: Use services like* ***Amazon SageMaker*** *or* ***AWS Lambda*** *for scalable compute resources. SageMaker offers managed infrastructure that can scale based on your model's requirements. [Auto-scaling, Data Storage, Load Balancing, Caching]*

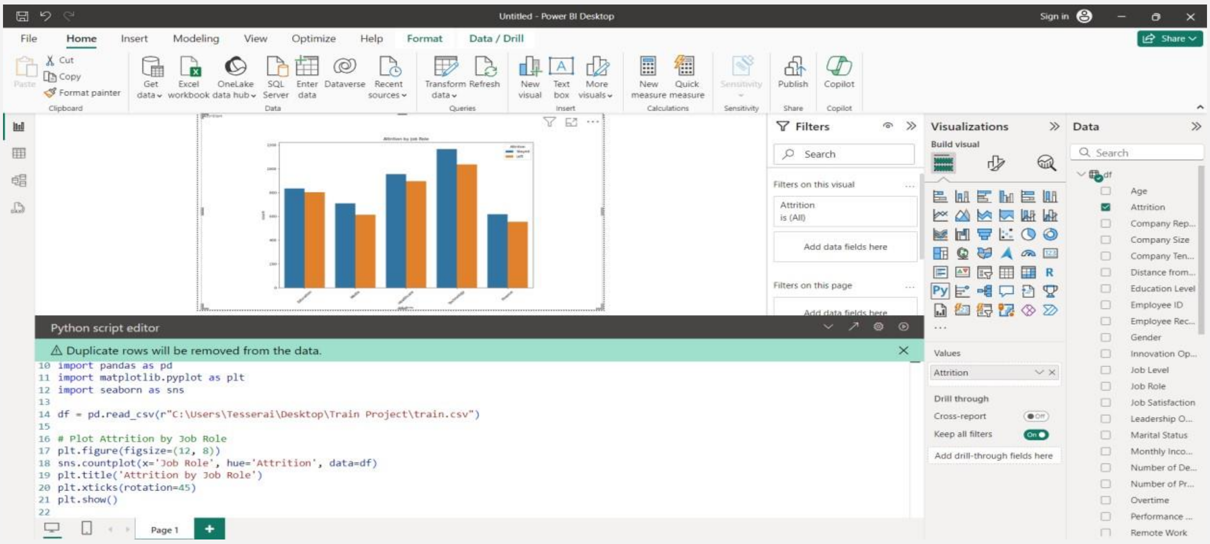
Performance Optimization

Optimizing the performance of a machine learning model on AWS involves enhancing its speed, efficiency, and resource utilization. Here are techniques and benchmarks for achieving optimal performance:

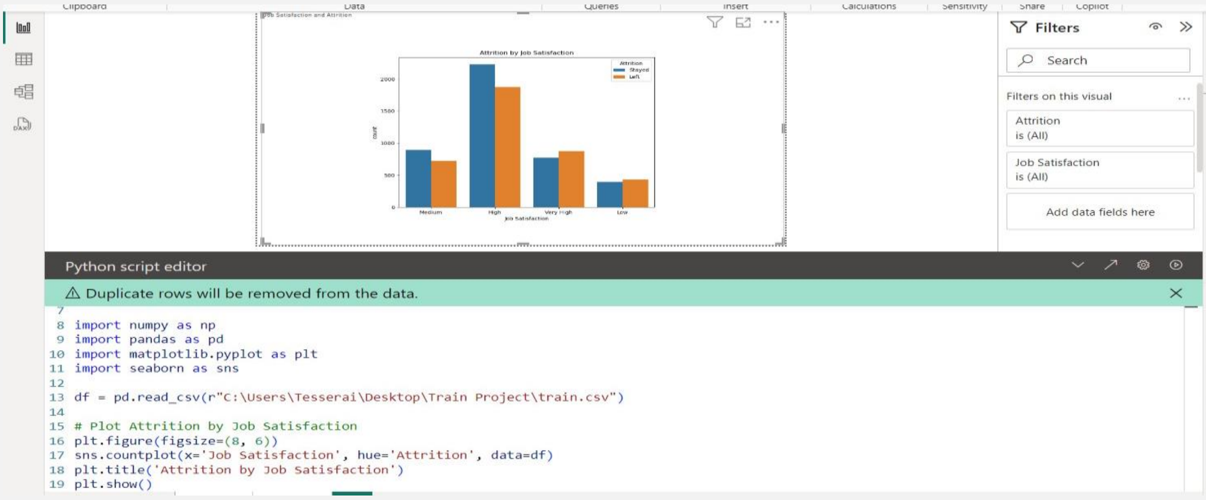
Model Optimization, Hardware Acceleration, Batch Processing, Model Compression, Pipeline Optimization, Benchmarking and Monitoring.

DATA VISUALIZATION USING POWER BI

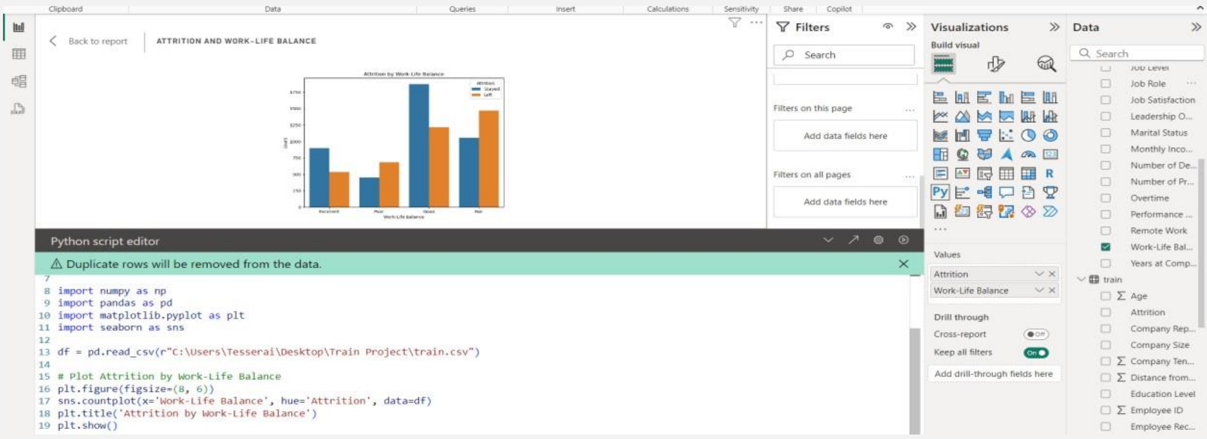
Plot Attrition by Job Role



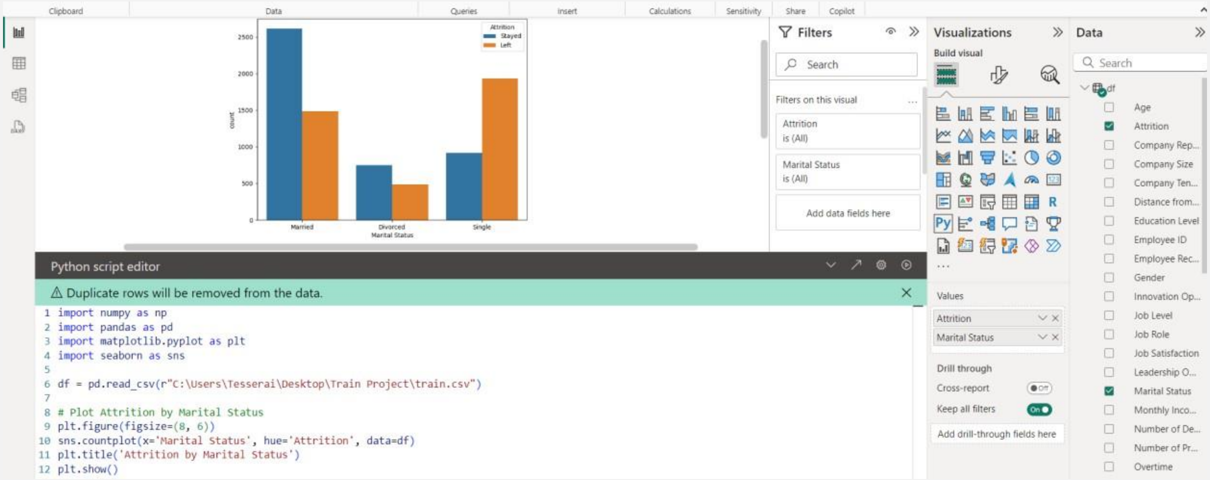
Plot Attrition by Job Satisfaction



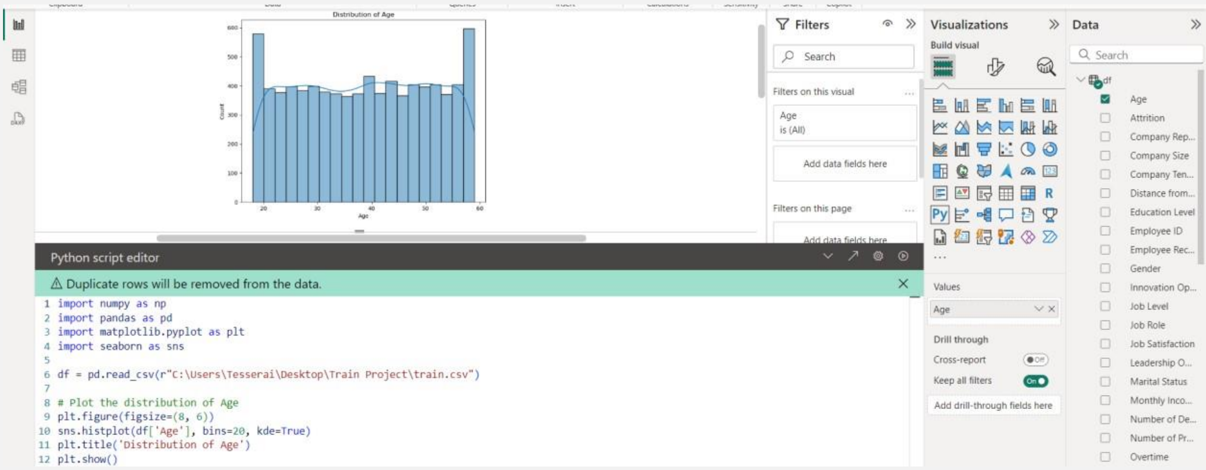
Plot Attrition by Work-Life Balance



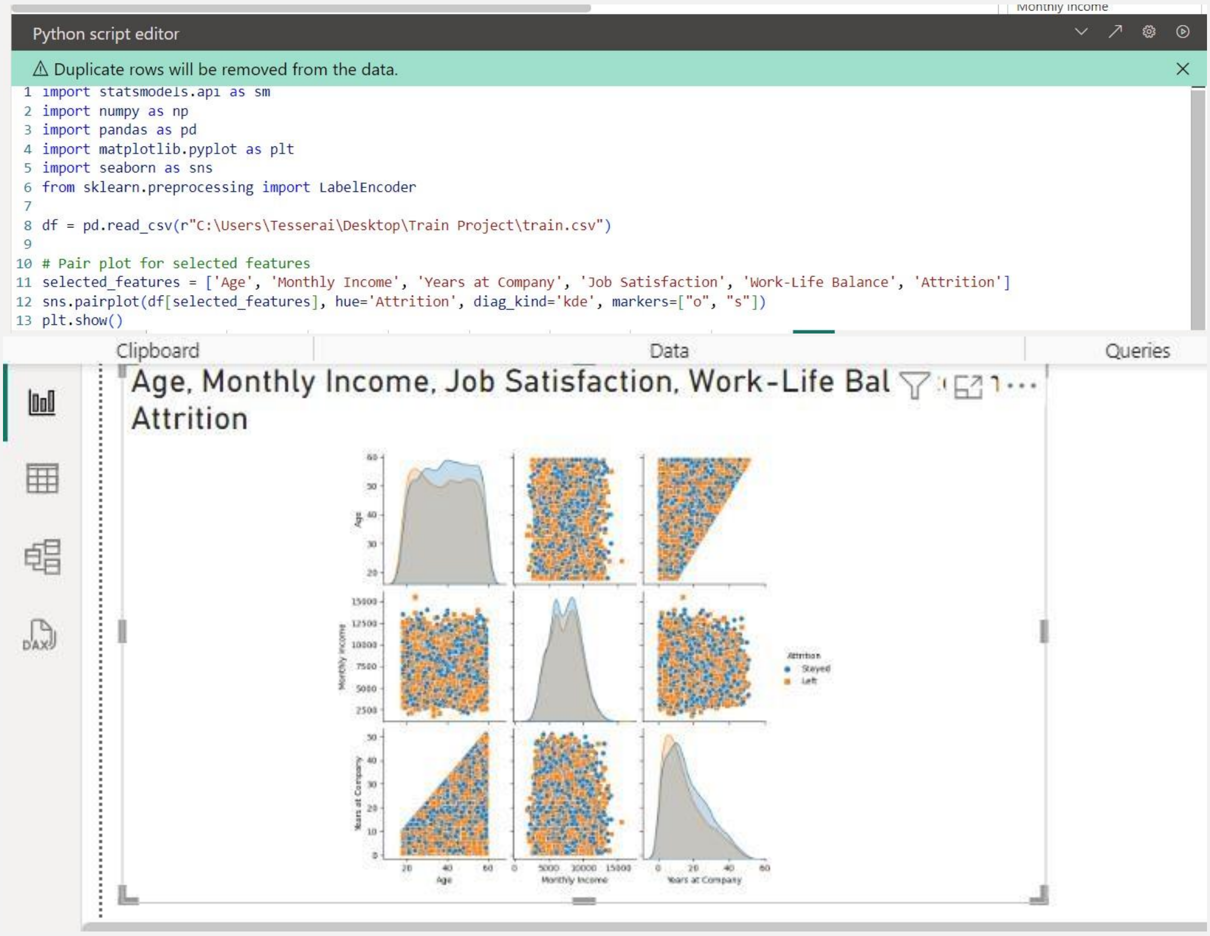
Plot Attrition by Marital Status



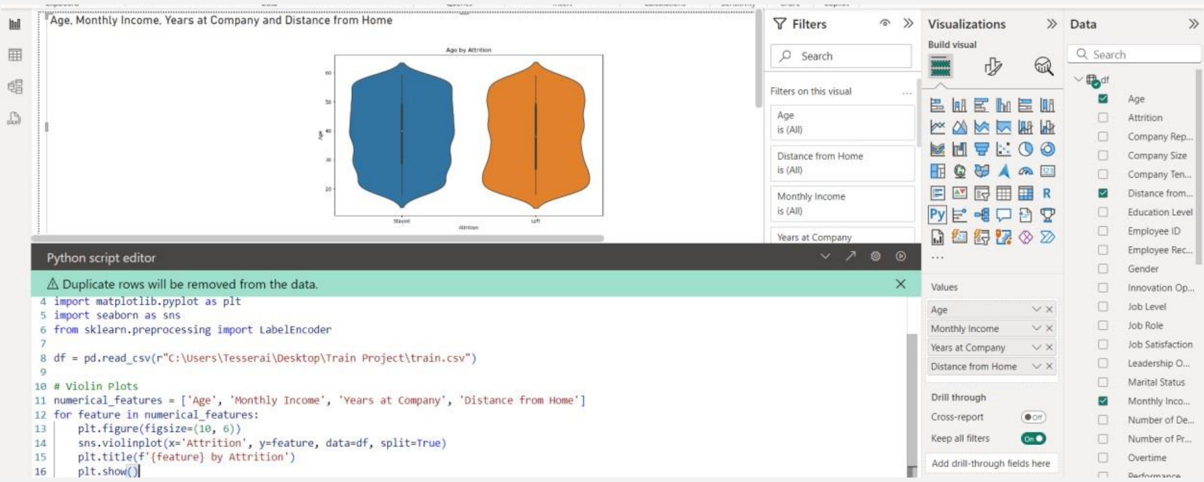
Plot the distribution of Age



Plot the confusion matrix



Violin Plots



DATA VISUALIZATION USING PYTHON

Plotting Hitmap