AMAZON SAGEMAKER PROJECT DOCUMENTATION

A Comprehensive Guide to Training a Logistic Regression Model Using Amazon Sage Maker

Abstract

This document outlines the steps to train a logistic regression model using Amazon Sage Maker. It covers importing libraries, setting up the Sage Maker session, configuring the estimator, and managing input data. The guide provides clear instructions for setting hyperparameters and starting the training job, offering a concise roadmap for leveraging Sage Maker's capabilities in machine learning.

Romeo Sebola

Romeo.sebola@gmail.com

About the Dataset

The Synthetic Employee Attrition Dataset is designed for analysing and predicting employee turnover. It includes 74,498 samples with detailed employee information such as demographics, job roles, and personal circumstances.

Introduction

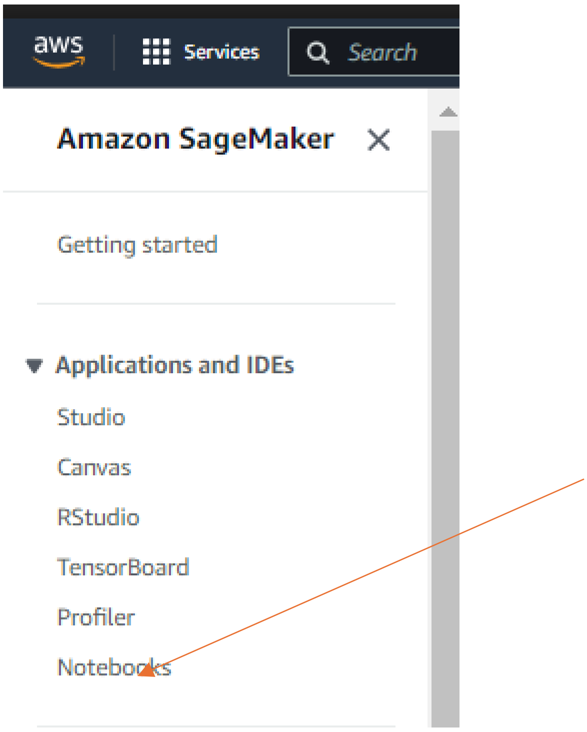
This documentation provides a detailed guide to using Amazon Sage Maker for training a logistic regression model. It includes code snippets and explanations of each step in the process, from data preparation to model training. The goal is to help users understand and execute a machine learning training job using Sage Maker efficiently.

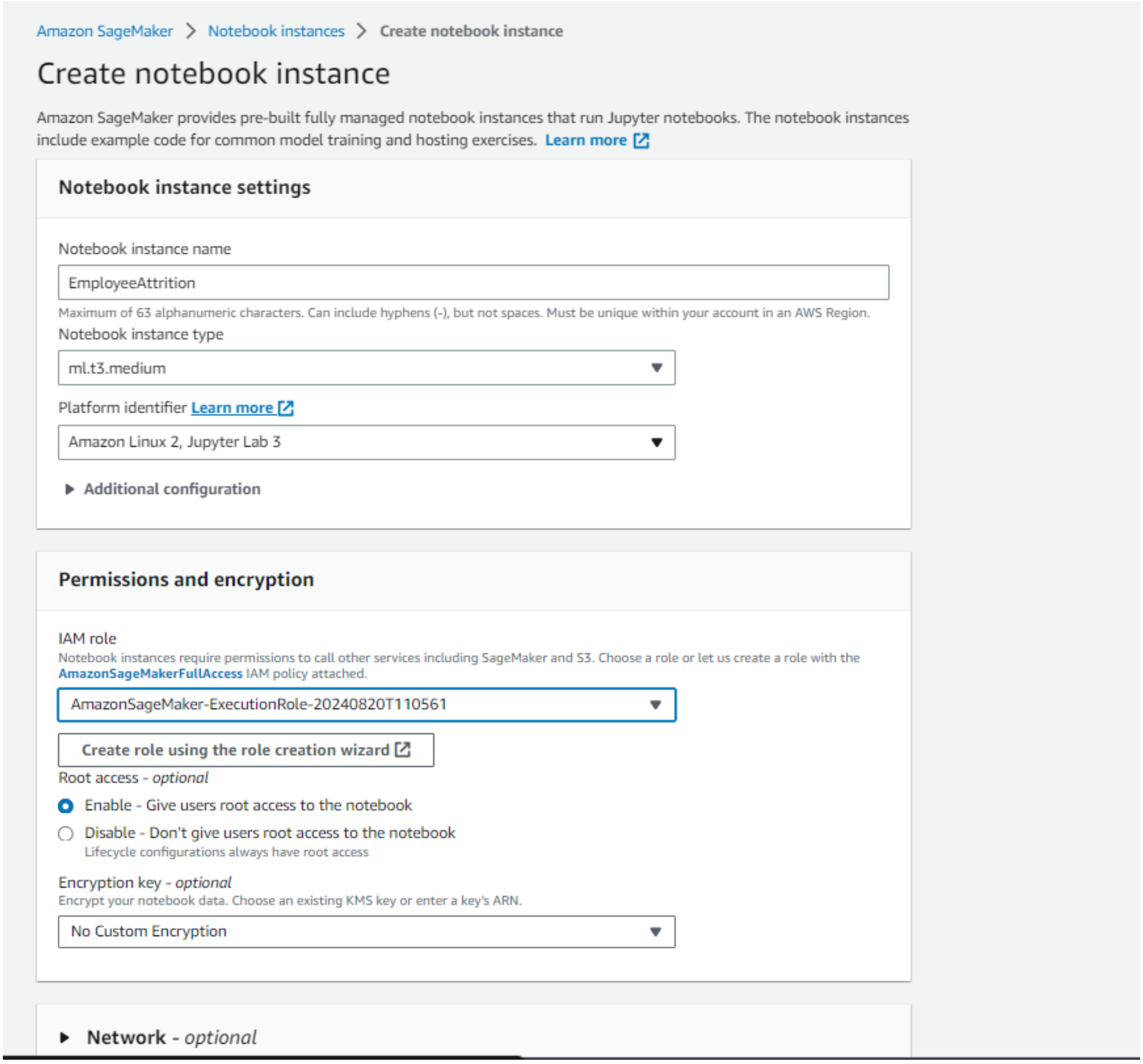
Attrition: Indicates whether the employee has left the company (0 for stayed, 1 for left).

This dataset is valuable for HR analytics and machine learning, helping to understand and predict

Model deployment process

Create Notebook instance





Importing Libraries

```
#import the packages or libraries

import pandas as pd
import numpy as np
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')
```

Loading the Data from the Dataset

```
#Loading the data from the datasets
df=pd.read_csv('train.csv')
df.head(10)
```

	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	24368	38	Female	3	Technology	9977	Fair	High	Below Average	3	No	37	Bachelor's Degree	Married	0
6	64970	47	Male	23	Education	3681	Fair	High	High	1	Yes	75	High School	Divorced	3
7	36999	48	Male	16	Finance	11223	Excellent	Very High	High	2	No	5	Master's Degree	Married	4

Exploratory Data Analysis (EDA)

```
#EDA
#Exploratory Data Analysis
#view the statistics of the datasets
df.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

Viewing the First Few Rows of the Dataset

```
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
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

Checking for Missing Values in the Dataset

```
df.isnull().values
```

```
array([[False, False, False, ..., False, False, False],
       [False, False, False, ..., False, False, False],
       [False, False, False, ..., False, False, False],
       ...,
       [False, False, False, ..., False, False, False],
       [False, False, False, ..., False, False, False],
       [False, False, False, ...,  True,  True,  True]])
```

Viewing the Dataset Information

```python
#Get the information about the datasets
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
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)
memory usage: 1.5+ MB
```

Analyzing the 'Attrition' Column

```python
#Get the count of the number of Employee that stayed or left the company
df['Attrition'].value_counts()
```

```
Attrition
Stayed    4282
Left      3903
Name: count, dtype: int64
```

Analyzing Categorical Columns

```
Gender : ['Male' 'Female']
Gender
Male      4500
Female    3686
Name: count, dtype: int64

_________________
Job Role : ['Education' 'Media' 'Healthcare' 'Technology' 'Finance']
Job Role
Technology    2201
Healthcare    1852
Education     1638
Media         1322
Finance       1173
Name: count, dtype: int64

_________________
Work-Life Balance : ['Excellent' 'Poor' 'Good' 'Fair']
Work-Life Balance
Good         3089
Fair         2525
Excellent    1436
Poor         1136
Name: count, dtype: int64

_________________
Job Satisfaction : ['Medium' 'High' 'Very High' 'Low']
Job Satisfaction
High         4100
Very High    1647
Medium       1612
Low           827
Name: count, dtype: int64

_________________
```

Encoding Categorical Data

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

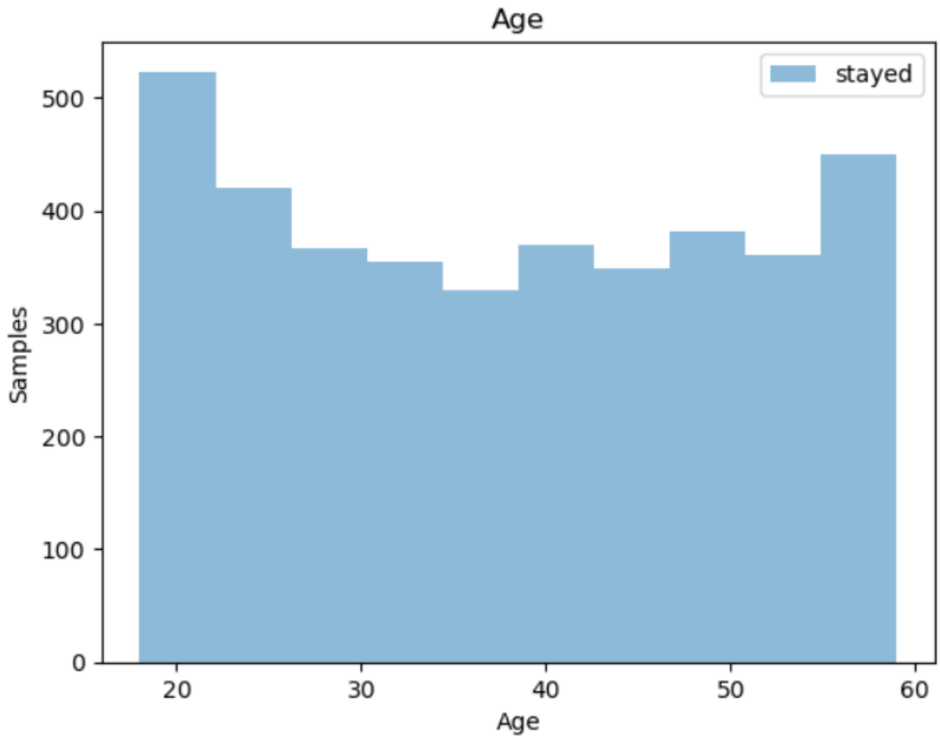
	Employee ID	Age	Gender	Years at Company	Job Role	Monthly Income	Work-Life Balance	Job Satisfaction	Performance Rating	Number of Promotions	Overtime	Distance from Home	Education Level	Marital Status	Number of Dependents
0	8410	31	Male	19	Education	5390	Excellent	Medium	Average	2	No	22	Associate Degree	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	24368	38	Female	3	Technology	9977	Fair	High	Below Average	3	No	37	Bachelor's Degree	Married	0
6	64970	47	Male	23	Education	3681	Fair	High	High	1	Yes	75	High School	Divorced	3

Creating Boolean Masks for 'Attrition' Column

```
stayed = df.Attrition == 0
left = df.Attrition == 1
```

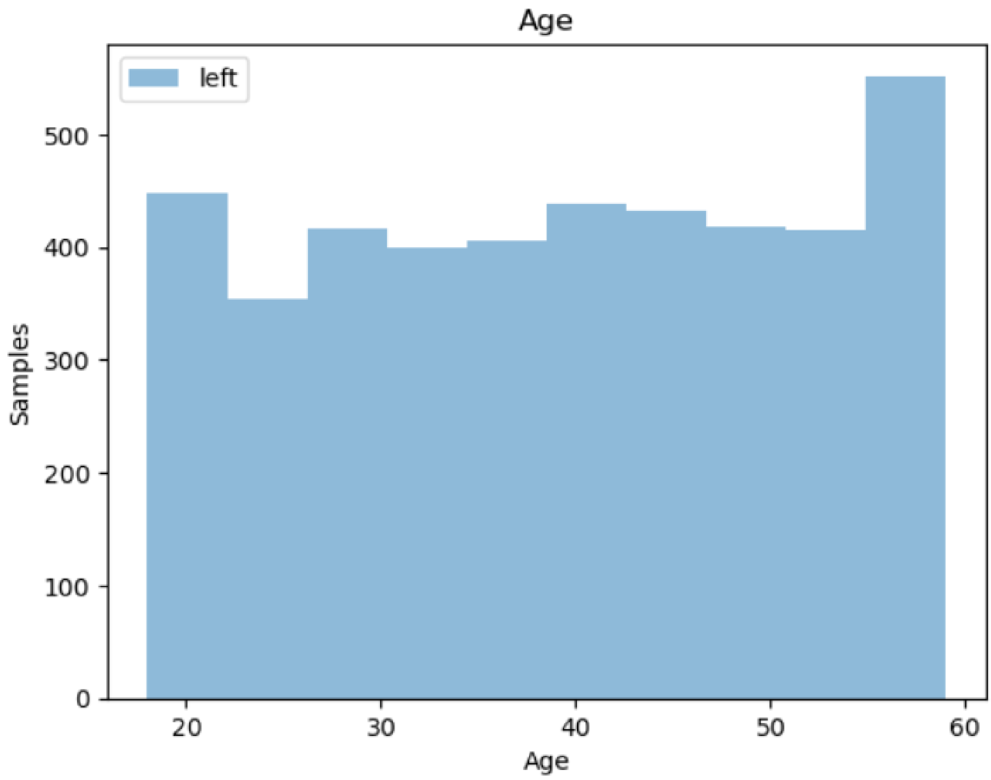
Plotting Histogram of Ages for Employees Who Stayed

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



Plotting Histogram of Ages for Employees Who Left

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



Viewing Encoded Class Labels

```
: le.classes_
```

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

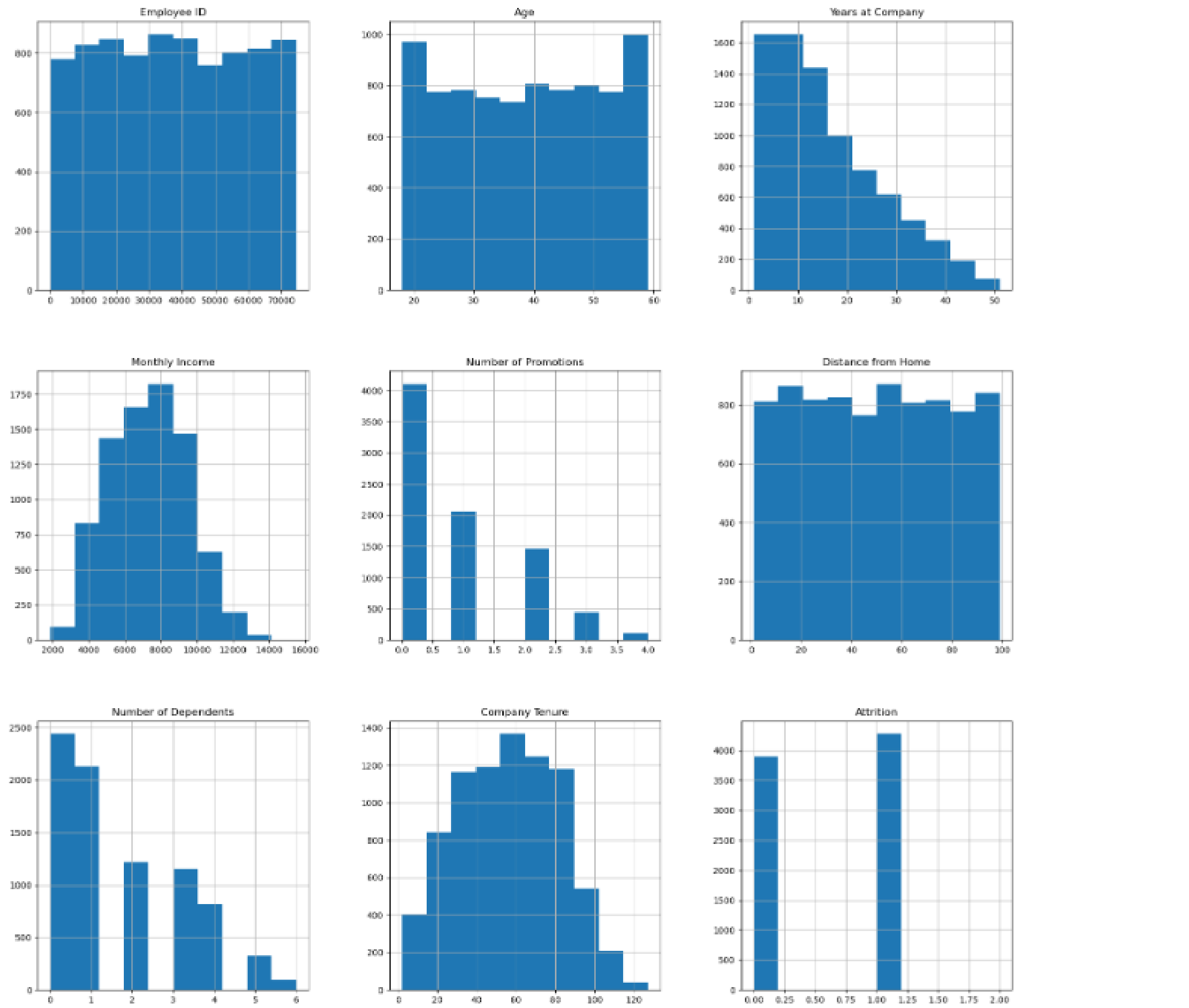
Displaying Encoded Class Labels

```
5]: Attrition_Employee = le.classes_
    print(Attrition_Employee)
```

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

Plotting the Distribution of Numerical Columns

```
#ploting the distribution
p = df.hist(figsize = (20,20))
```



Mapping Colors and Plotting Bar Chart for 'Attrition'

```
#check the balance of the data by plotting the count of Attrition
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    4282
0    3903
2       1
Name: count, dtype: int64
```

Label Encoding Categorical Columns

```python
#Define the columns to be label encoded
label_cols = ['Gender','Job Role','Overtime','Education Level','Marital Status','Company Size','Remote Work', 'Leadership Opportunities','Innovation Opportunities','Work-Life

#Initialize labe encoders
label_encoders = {col: LabelEncoder() for col in label_cols}

#Apply the Label Encoding
for col in label_cols:
    df[col] = label_encoders[col].fit_transform(df[col])
```

```python
df.head(10)
```

	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	Company Size	Company Tenure	Rem W
0	8410	31	1	19	0	5390	0	2	0	2	0	22	0	1	0	1	1	89	
1	64756	59	0	4	3	5534	3	0	3	3	0	21	3	0	3	1	1	21	
2	30257	24	0	10	2	8159	2	0	3	0	0	11	1	1	3	1	1	74	
3	65791	36	0	7	0	3989	2	0	2	1	0	27	2	2	2	1	2	50	
4	65026	56	1	41	0	4821	1	3	0	0	1	71	2	0	0	2	1	68	
5	24368	38	0	3	4	9977	1	0	1	3	0	37	1	1	0	1	1	47	
6	64970	47	1	23	0	3681	1	0	2	1	1	75	2	0	3	0	2	93	
7	36999	48	1	16	1	11223	0	3	2	2	0	5	3	1	4	0	1	88	
8	32714	57	1	44	0	3773	2	2	2	1	1	39	2	1	4	0	1	75	
9	15944	24	0	1	2	7319	3	0	0	1	1	57	4	2	4	0	0	45	

Removing the Last Column and Selecting Remaining Columns

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

Selecting the Target Column

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

```python
y.head()
```

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

Splitting Data into Training and Testing Sets

```python
from sklearn.model_selection import train_test_split
```

```python
X_Train, X_Test, Y_Train, Y_Test = train_test_split(x,y, test_size = 0.3)
```

```python
trainDF = X_Train.join(Y_Train)
```

```python
X_Train.head(5)
```

	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	Company Size	Company Tenure	R
5794	15029	42	1	28	0	4399	2	0	0	0	0	70	0	1	0	1	2	35	
3116	34680	39	1	25	2	7852	1	0	0	1	0	56	4	0	1	1	1	39	
1774	52289	31	1	4	4	11030	1	0	0	0	0	25	0	1	2	2	2	25	
1718	22781	26	0	1	4	10252	0	0	1	1	0	30	0	0	1	2	0	78	
8135	16113	47	1	5	2	8291	1	0	0	2	0	2	0	2	5	2	2	11	

Creating and Inspecting the Training DataFrame

```
trainDF=X_Train.join(Y_Train)
trainDF.head(5)
```

	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	Company Size	Company Tenure	R
5794	15029	42	1	28	0	4399	2	0	0	0	0	70	0	1	0	1	2	35	
3116	34680	39	1	25	2	7852	1	0	0	1	0	56	4	0	1	1	1	39	
1774	52289	31	1	4	4	11030	1	0	0	0	0	25	0	1	2	2	2	25	
1718	22781	26	0	1	4	10252	0	0	1	1	0	30	0	0	1	2	0	78	
8135	16113	47	1	5	2	8291	1	0	0	2	0	2	0	2	5	2	2	11	

```
testDF = X_Test.join(Y_Test)
```

Creating and Inspecting the Testing DataFrame

```
testDF = X_Test.join(Y_Test)
testDF.head(5)
```

	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	Company Size	Company Tenure	R
3524	46223	19	1	5	2	7423	2	0	0	2	0	94	2	1	2	0	1	48	
23	24208	36	1	13	3	5874	1	0	0	0	0	16	2	2	1	1	1	40	
486	15082	39	1	7	1	6991	0	1	0	1	1	65	1	1	1	0	2	84	
1452	72295	50	0	27	4	8465	2	0	0	1	1	68	1	2	1	0	1	102	
952	4812	45	0	15	0	4604	1	0	2	0	0	45	1	2	0	1	2	17	

Defining a List of Specific 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',
         ]
```

Filtering the Training DataFrame to Specific Columns

```
trainDF= trainDF[column]
trainDF.head(10)
```

	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	In Oppc
5794	1	42	1	28	0	1	0	1	0	4399	2	0	0	70	2	35	0	0	
3116	1	39	1	25	2	0	4	1	1	7852	1	0	0	56	1	39	1	1	
1774	1	31	1	4	4	1	0	2	2	11030	1	0	0	25	2	25	0	0	
1718	0	26	0	1	4	0	0	2	1	10252	0	0	0	30	0	78	0	0	
8135	0	47	1	5	2	2	0	2	5	8291	1	0	0	2	2	11	0	0	
3343	1	53	1	32	2	0	1	1	6	6045	2	0	0	85	2	46	1	0	
564	0	36	0	12	4	2	1	1	1	8656	1	1	1	73	2	71	1	0	
773	1	31	1	15	2	0	3	0	0	8403	2	2	1	71	2	24	0	0	
4213	0	43	1	25	0	2	1	1	0	3435	0	2	1	7	1	88	0	0	
3354	1	19	1	10	2	1	2	0	2	6249	2	1	1	85	1	22	1	0	

25. Filtering the Testing DataFrame to Specific Columns (Excluding the First Column)

```
#Save the trained data#
#write training set#

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

```
testDF.head()
```

	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
3524	19	1	5	2	1	2	0	2	7423	2	0	0	94	1	48	0	0	0
23	36	1	13	3	2	2	1	1	5874	1	0	0	16	1	40	0	0	0
486	39	1	7	1	1	1	0	1	6991	0	1	1	65	2	84	0	0	0
1452	50	0	27	4	2	1	0	1	8465	2	0	1	68	1	102	0	0	0
952	45	0	15	0	2	1	1	0	4604	1	0	0	45	2	17	0	0	0

26. Saving the Filtered Training DataFrame to a CSV File

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

Importing Libraries for Cloud Integration and Pattern Matching

```
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 expresession//
```

Specifying Bucket Name and File Paths

```
#Specify bucket name
bucketNM='romeodiabetecbucket'
TrainFile = r'attritiondata/traineddataattritions/traineddataattritions.csv'
TestFile = r'attritiondata/testddataattritions/testddataattritions.csv'
ValFile = r'attritiondata/Val/Val.csv'
ModelFolder = r'attritiondata/model/'
```

Constructing S3 Paths for Data and Model Storage

```
#Loading
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)
```

Constructing the S3 Path for Model Output

```
s3ModelOutput
```

```
's3://romeodiabetecbucket/attritiondata/model/'
```

To document the code for uploading a file to an S3 bucket using boto3, you can structure it like this in your Word document:

Uploading a File to an S3 Bucket

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

Uploading a Test Data File to an S3 Bucket

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

Importing SageMaker and Getting Execution Role

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

Creating a SageMaker Session and Retrieving the Execution Role

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

Retrieving Docker Image URI for SageMaker Estimator

```
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 version: latest.
INFO:sagemaker.image_uris:Ignoring unnecessary instance type: None.
```

Configuring a SageMaker Estimator for Logistic Regression Model

```
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.
```

Setting and Retrieving Hyperparameters for SageMaker Estimator

```
LogisticModel.set_hyperparameters(predictor_type='binary_classifier', mini_batch_size=100)
```

```
LogisticModel.hyperparameters()
```

```
{'predictor_type': 'binary_classifier', 'mini_batch_size': 100}
```

Configuring S3 Input Data for SageMaker Training

```
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.
```

Starting the Training Job for SageMaker Estimator

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

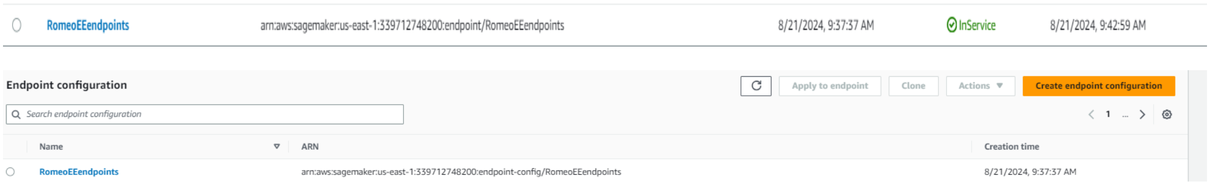
```
INFO:sagemaker:Creating training-job with name: Logistic-Demo-v1-2024-08-20-12-34-10-548
2024-08-20 12:34:10 Starting - Starting the training job...
2024-08-20 12:34:25 Starting - Preparing the instances for training...
2024-08-20 12:34:56 Downloading - Downloading input data...
2024-08-20 12:35:26 Downloading - Downloading the training image......
2024-08-20 12:36:37 Training - Training image download completed. Training in progress....Docker entrypoint called with argument(s): train
Running default environment configuration script
[08/20/2024 12:37:01 INFO 139985293625152] Reading default configuration from /opt/amazon/lib/python3.8/site-packages/algorithm/resources/default-input.json: {'mini_batch_size': '1000', 'epochs': '15', 'feature_dim': 'auto', 'use_bias': 'true', 'binary_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': '0.01', 'init_bias': '0.0', 'optimizer': 'auto', 'loss': 'auto', 'margin': '1.0', 'quantile': '0.5', 'loss_insensitivity': '0.01', 'huber_delta': '1.0', 'num_classes': '1', 'accuracy_top_k': '3', 'wd': 'auto', 'l1': 'auto', 'momentum': 'auto', 'learning_rate': 'auto', 'beta_1': 'auto', 'beta_2': 'auto', 'bias_lr_mult': 'auto', 'bias_wd_mult': 'auto', 'use_lr_scheduler': 'true', 'lr_scheduler_step': 'auto', 'lr_scheduler_factor': 'auto', 'lr_scheduler_minimum_lr': 'auto', 'positive_example_weight_mult': '1.0', 'balance_multiclass_weights': 'false', 'normalize_data': 'true', 'normalize_label': 'auto', 'unbias_data': 'auto', 'unbias_label': 'auto', 'num_point_for_scaler': '10000', '_kvstore': 'auto', '_num_gpus': 'auto', '_num_kv_servers': 'auto', '_log_level': 'info', '_tuning_objective_metric': '', 'early_stopping_patience': '3', 'early_stopping_tolerance': '0.001', '_enable_profiler': 'false'}
[08/20/2024 12:37:01 INFO 139985293625152] Merging with provided configuration from /opt/ml/input/config/hyperparameters.json: {'mini_batch_size': '100', 'predictor_type': 'binary_classifier'}
[08/20/2024 12:37:01 INFO 139985293625152] Final configuration: {'mini_batch_size': '100', 'epochs': '15', 'feature_dim': 'auto', 'use_bias': 'true', 'binary_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': '0.01', 'init_bias': '0.0', 'optimizer': 'auto', 'loss': 'auto', 'margin': '1.0', 'quantile': '0.5', 'loss_insensitivity': '0.01', 'huber_delta': '1.0', 'num_classes': '1', 'accuracy_top_k': '3', 'wd': 'auto', 'l1': 'auto', 'momentum': 'auto', 'learning_rate': 'auto', 'beta_1': 'auto', 'beta_2': 'auto', 'bias_lr_mult': 'auto', 'bias_wd_mult': 'auto', 'use_lr_scheduler': 'true', 'lr_scheduler_step': 'auto', 'lr_scheduler_factor': 'auto', 'lr_scheduler_minimum_lr': 'auto', 'positive_example_weight_mult': '1.0', 'balance_multiclass_weights': 'false', 'normalize_data': 'true', 'normalize_label': 'auto', 'unbias_data': 'auto', 'unbias_label': 'auto', 'num_point_for_scaler': '10000', '_kvstore': 'auto', '_num_gpus': 'auto', '_num_kv_servers': 'auto', '_log_level': 'info', '_tuning_objective_metric': '', 'early_stopping_patience': '3', 'early_stopping_tolerance': '0.001', '_enable_profiler': 'false', 'predictor_type': 'binary_classifier'}
```

Deploying the Trained Model

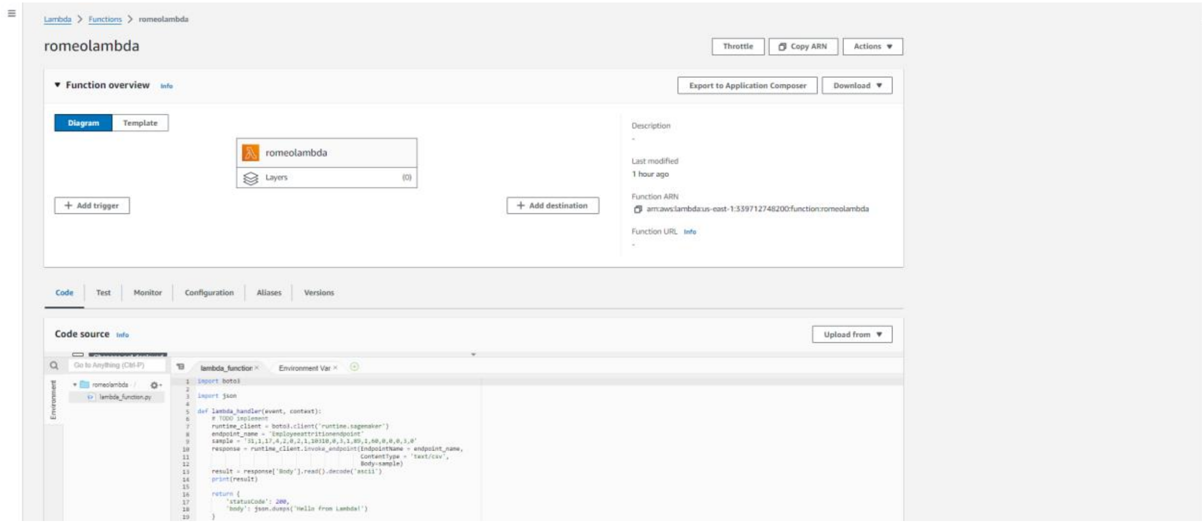
```
#Deploying the Trained Model

predictmodel=LogisticModel.deploy(initial_instance_count=1, instance_type='ml.m4.xlarge',
                                   endpoint_name = 'RomeoEEendpoints')
```

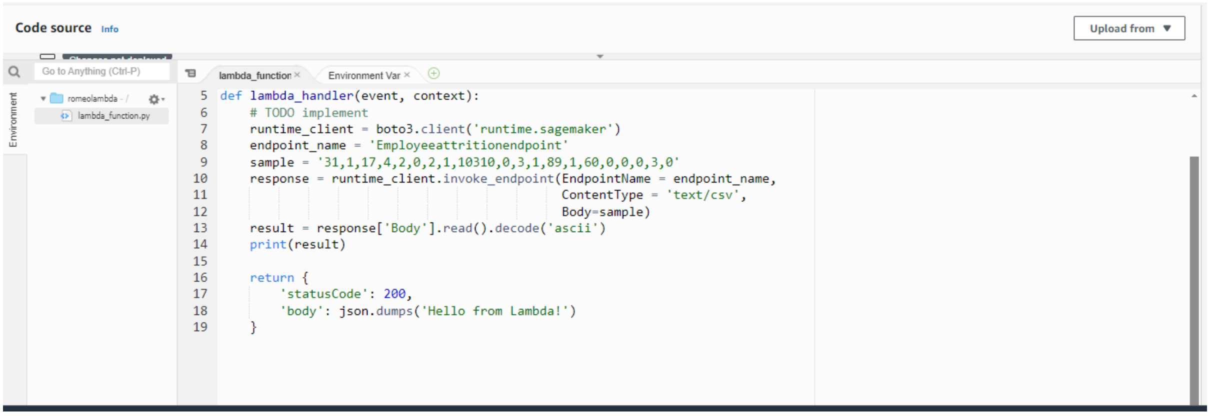
```
INFO:sagemaker:Creating model with name: Logistic-Demo-v1-2024-08-21-07-37-36-202
INFO:sagemaker:Creating endpoint-config with name RomeoEEendpoints
INFO:sagemaker:Creating endpoint with name RomeoEEendpoints
----------!
```

Verifying the Deployed Model

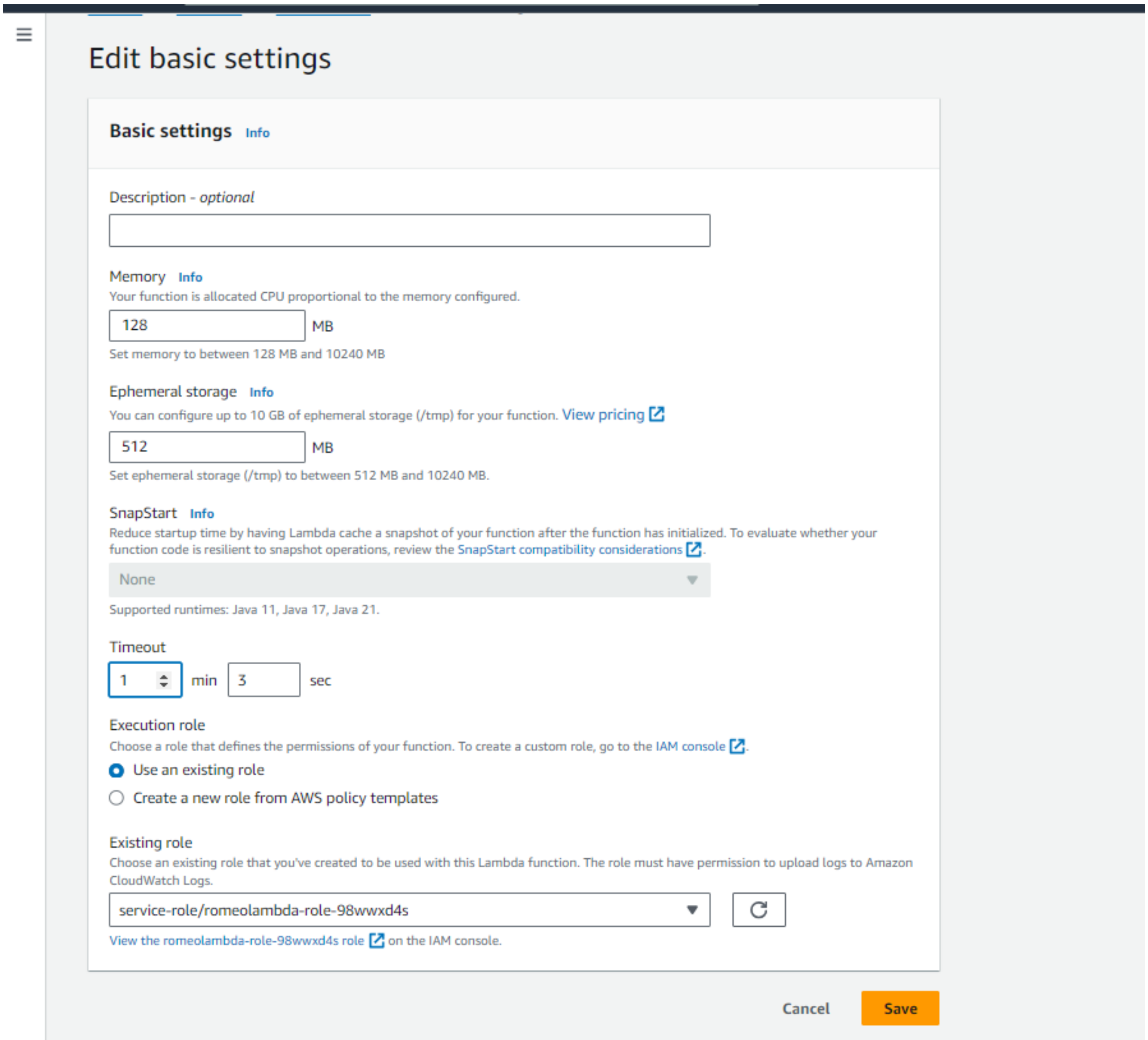


Create a lambda function to consume the model using test dataset

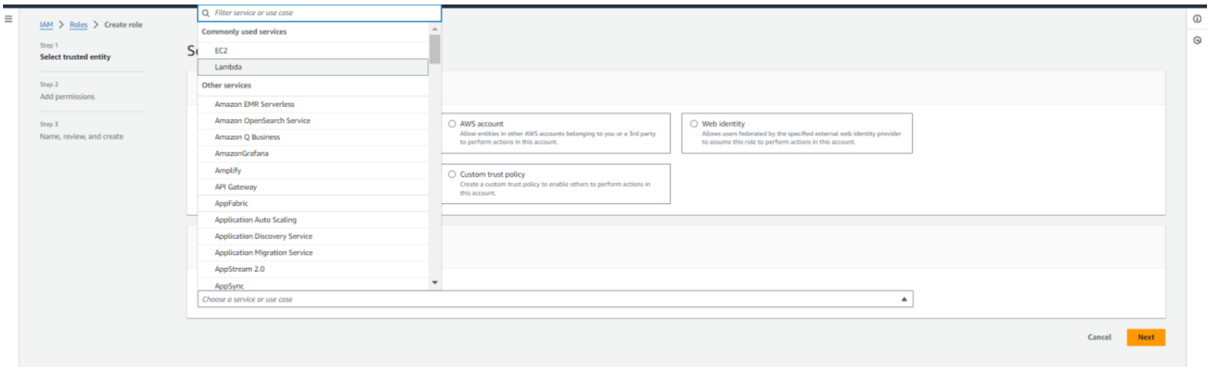


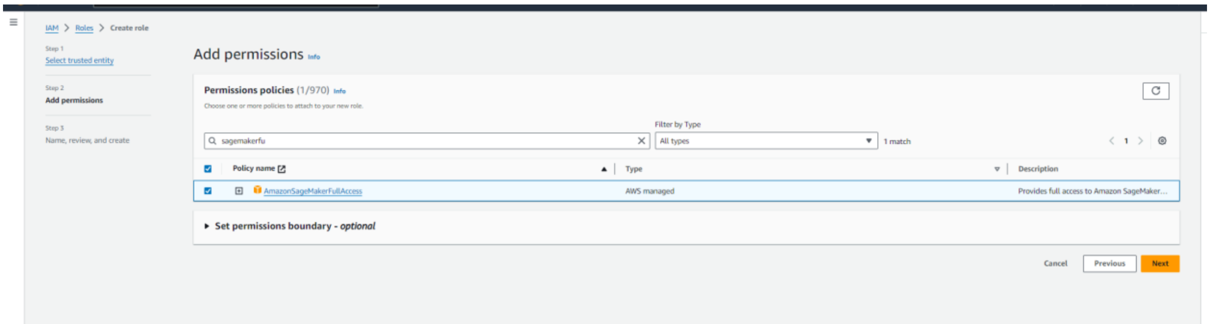


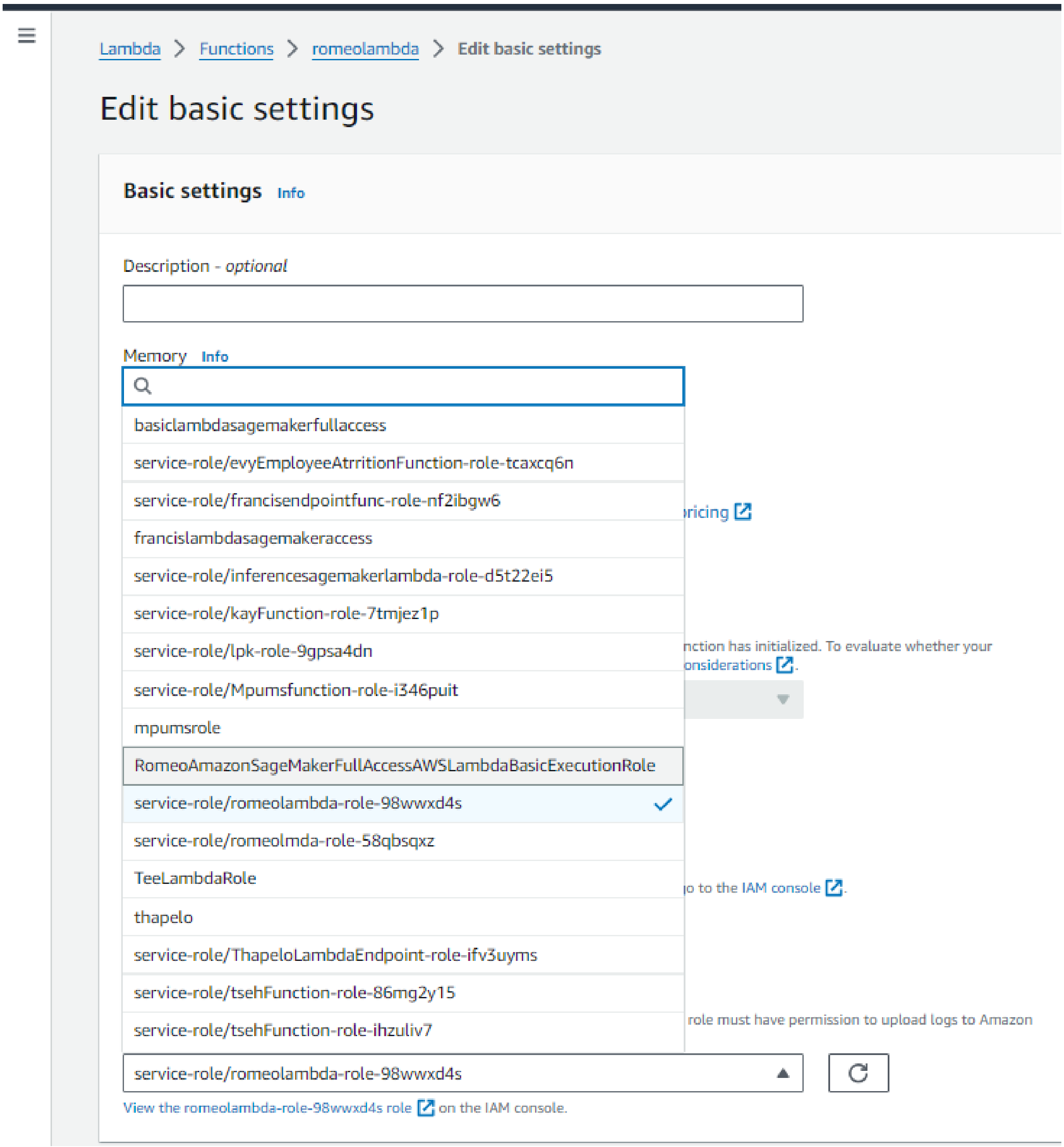
Configure the lambda function



Go to IAM to create role

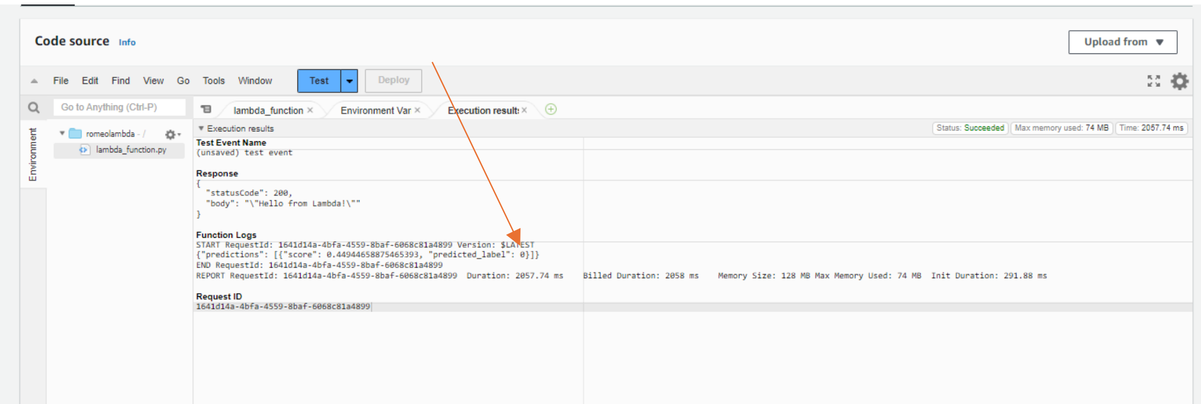




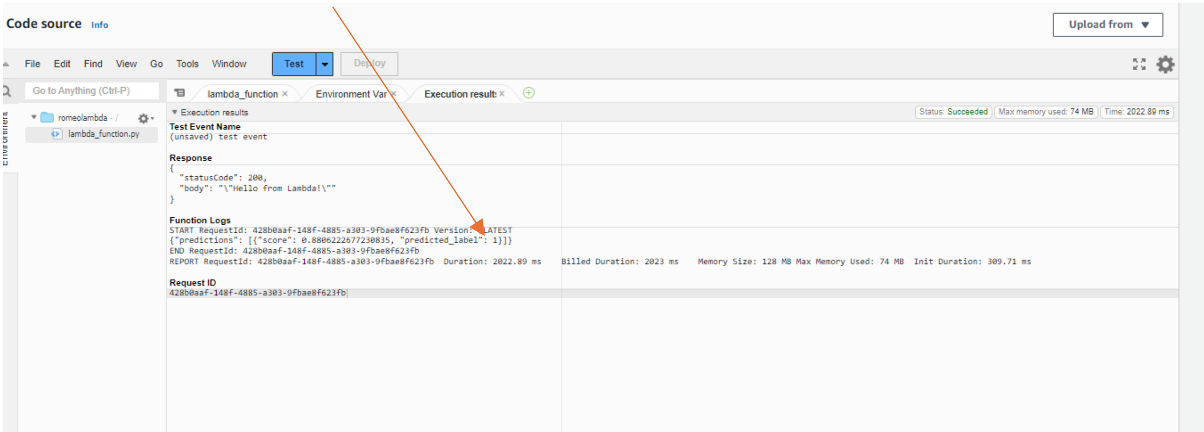


Deploy and Test

-The predicted label indicates “0” That means the Employee will stay



-The predicted label indicates “1” That means the Employee will Leave



Conclusion

In this project, we successfully developed a logistic regression model to predict employee attrition using the Synthetic Employee Attrition Dataset. The process involved multiple steps, from data preparation and exploratory data analysis to model training and deployment using Amazon SageMaker.

This demonstrates the end-to-end process of building, training, and deploying a machine learning model in a cloud environment. The deployed model can now be used for HR analytics to predict employee turnover, providing valuable insights for decision-making. By leveraging Amazon SageMaker, the process was efficient and scalable, showcasing the power of cloud-based machine learning.