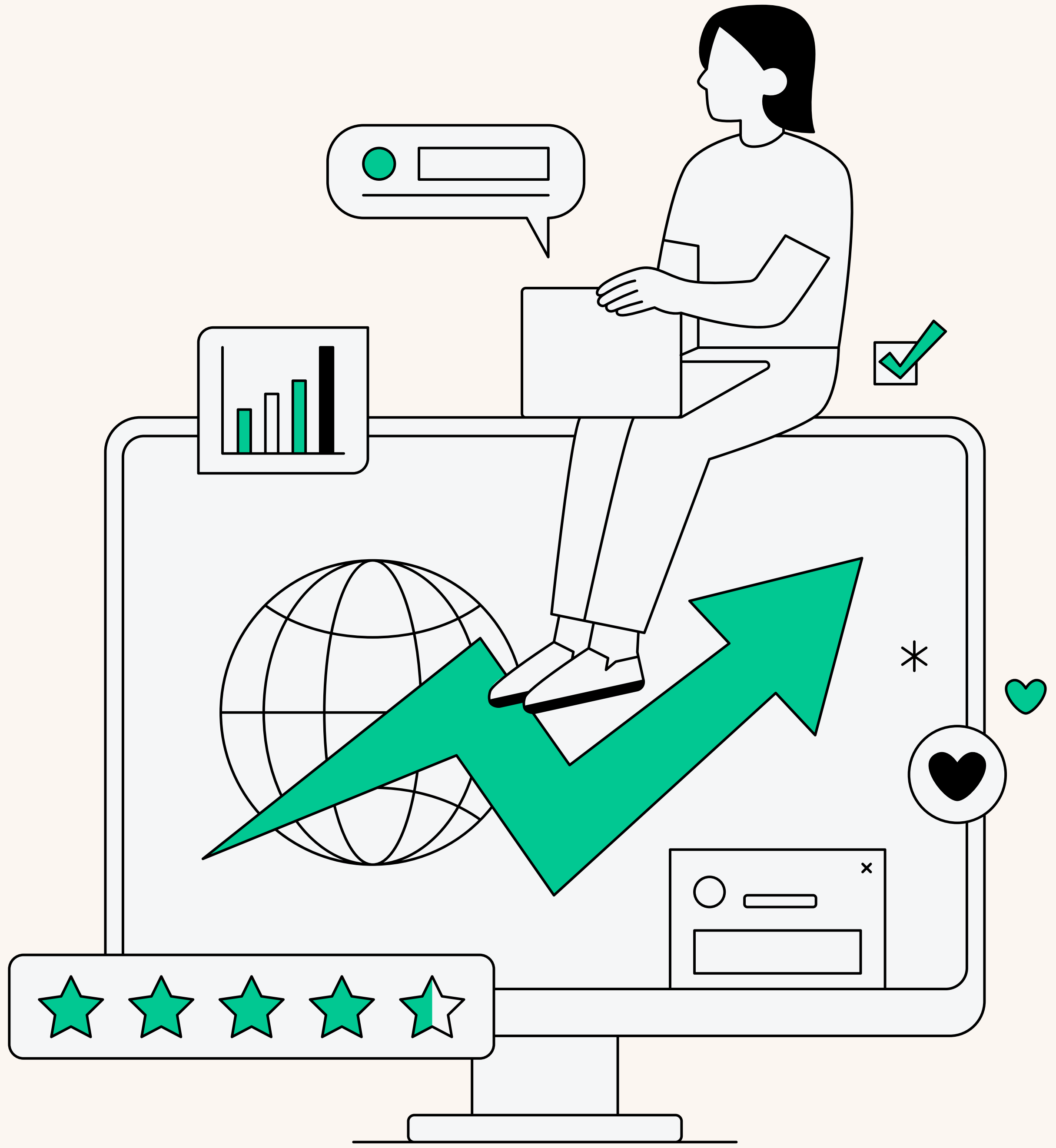
Building an AI-Driven Engagement Predictor on AWS: End-to-End ML Pipeline

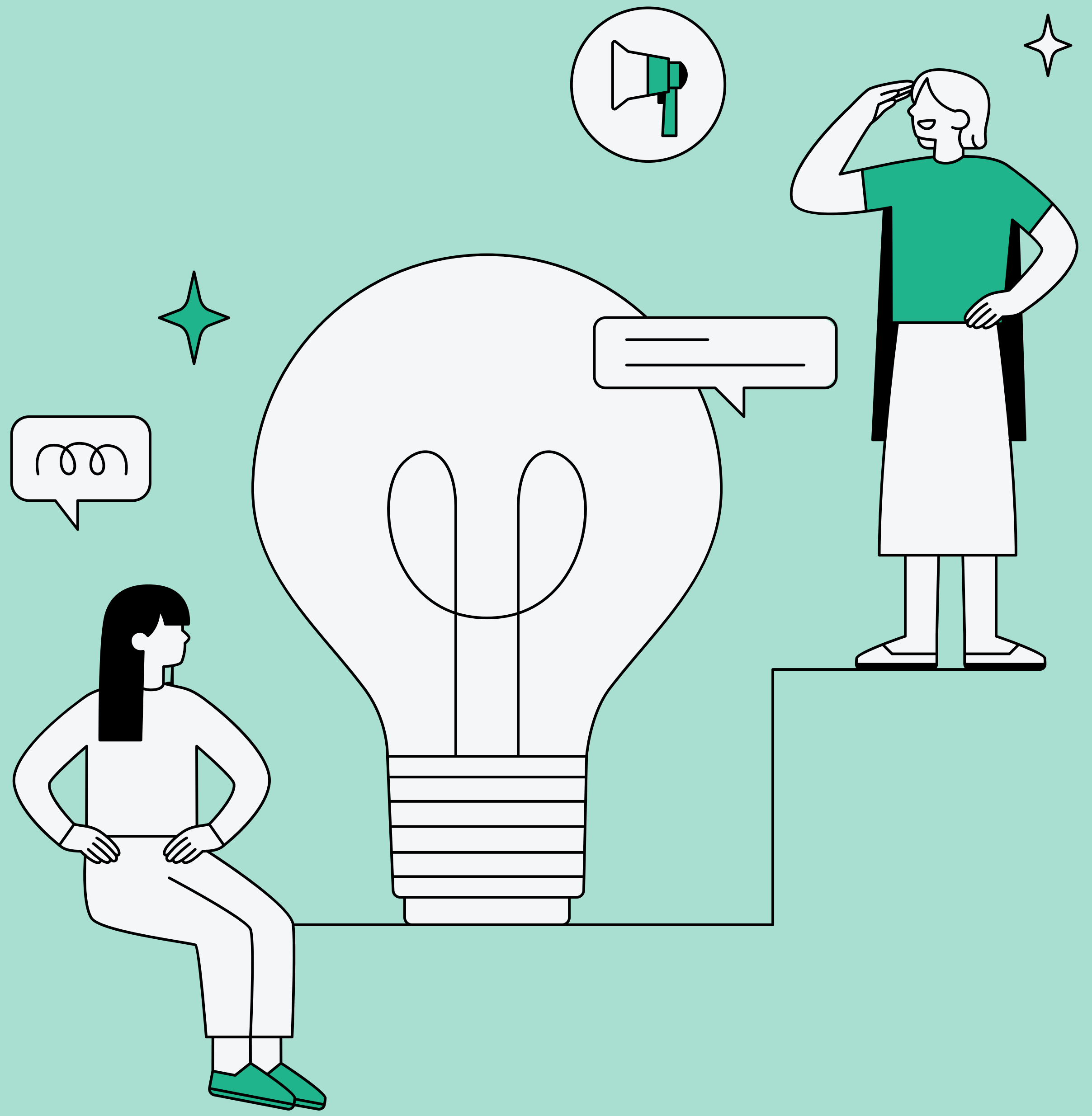
Slicpix / Interactivity.studio

Presented by Yash Singh
Date: 2025/06/25



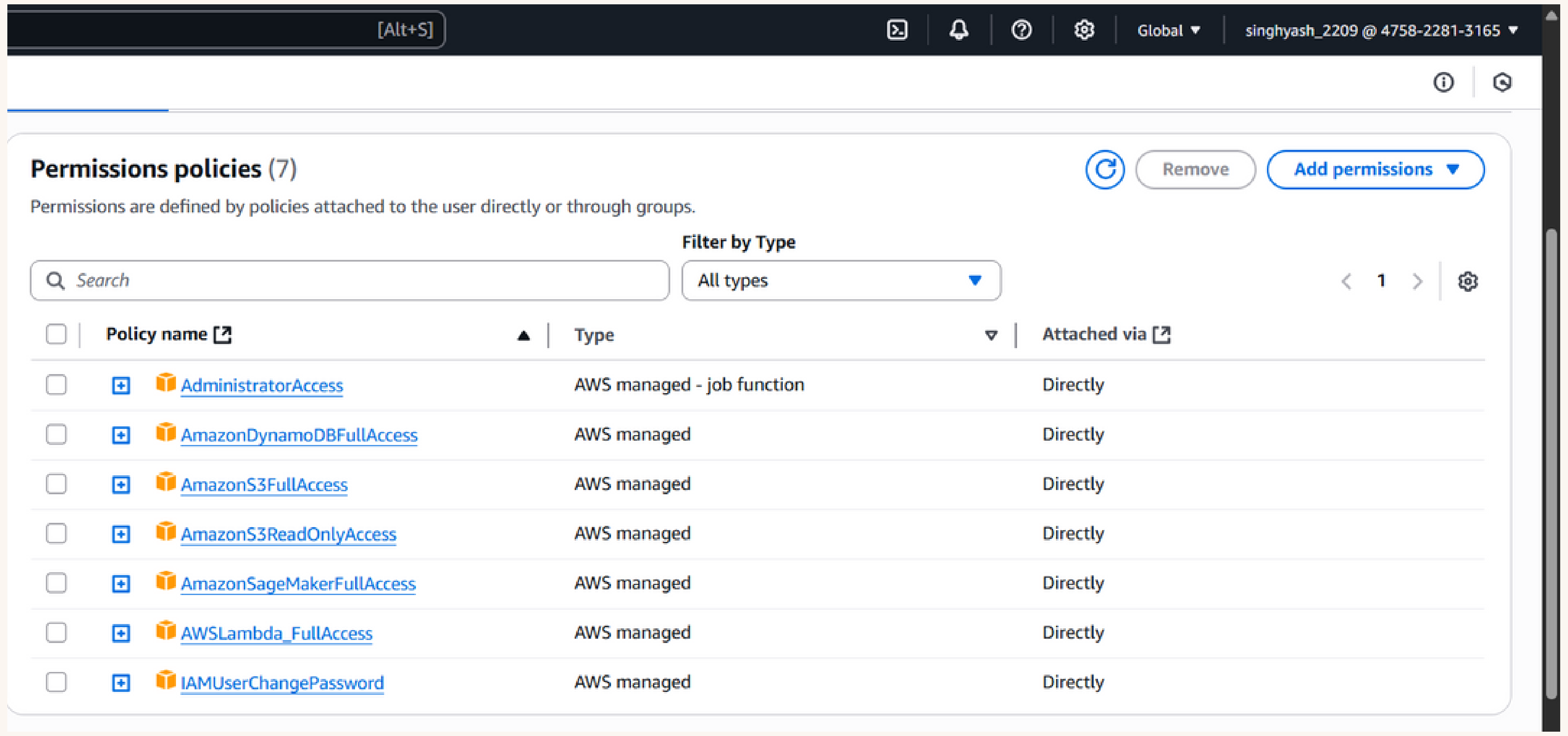
Introduction & Project Overview

This project showcases the end-to-end implementation of an AI-driven engagement predictor using AWS services. Leveraging S3, SageMaker, and Lambda, I built an automated pipeline to analyze plugin interaction data, enabling data-driven decisions for Slicpix's Interactivity.studio and delivering actionable insights to enhance user engagement.



IAM Permissions & Security Setup

To ensure secure and efficient workflow across AWS services, I configured IAM permissions with all necessary managed policies, such as AdministratorAccess, SageMakerFullAccess, S3FullAccess, and LambdaFullAccess. This setup enabled seamless integration of storage, computation, and deployment processes throughout the machine learning pipeline.



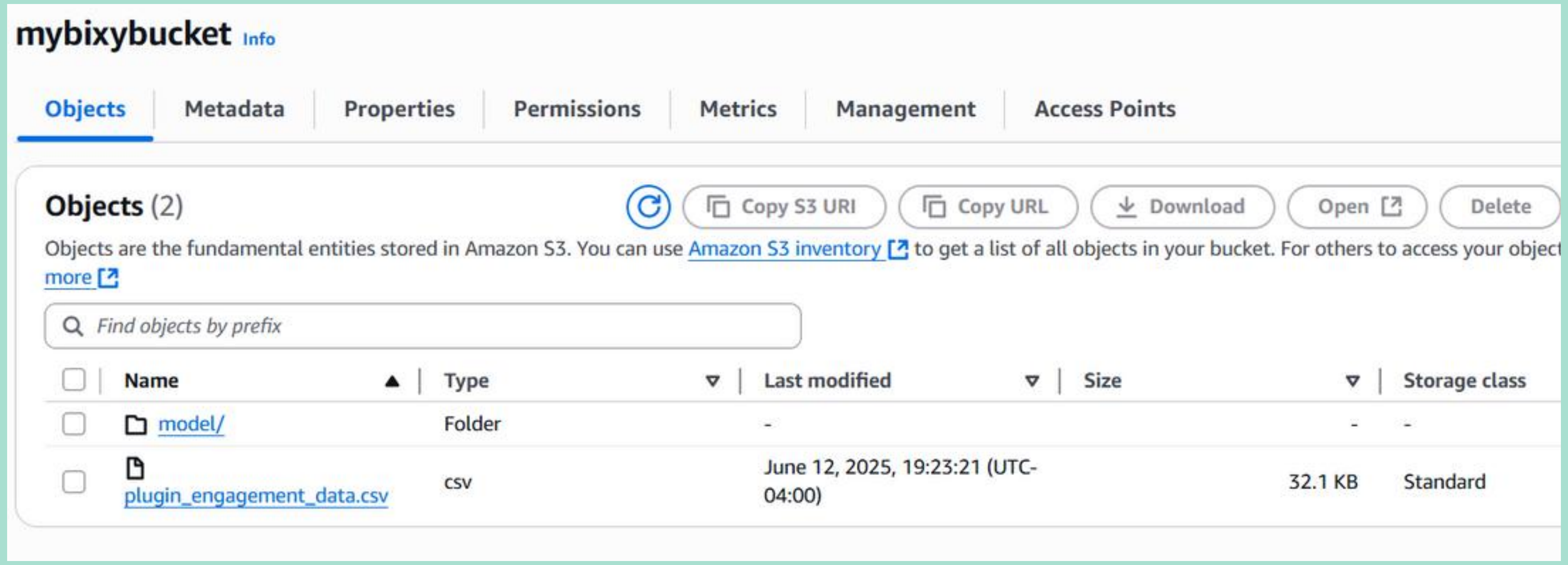
Permissions policies (7)

Permissions are defined by policies attached to the user directly or through groups.

Policy name	Type	Attached via
AdministratorAccess	AWS managed - job function	Directly
AmazonDynamoDBFullAccess	AWS managed	Directly
AmazonS3FullAccess	AWS managed	Directly
AmazonS3ReadOnlyAccess	AWS managed	Directly
AmazonSageMakerFullAccess	AWS managed	Directly
AWSLambda_FullAccess	AWS managed	Directly
IAMUserChangePassword	AWS managed	Directly

Data Storage on Amazon S3

I established an Amazon S3 bucket, mybixybucket, as the primary storage hub for all project assets. Uploading the plugin interaction dataset enabled efficient data management and seamless connectivity with SageMaker for model training and Lambda for deployment, streamlining the entire machine learning workflow.



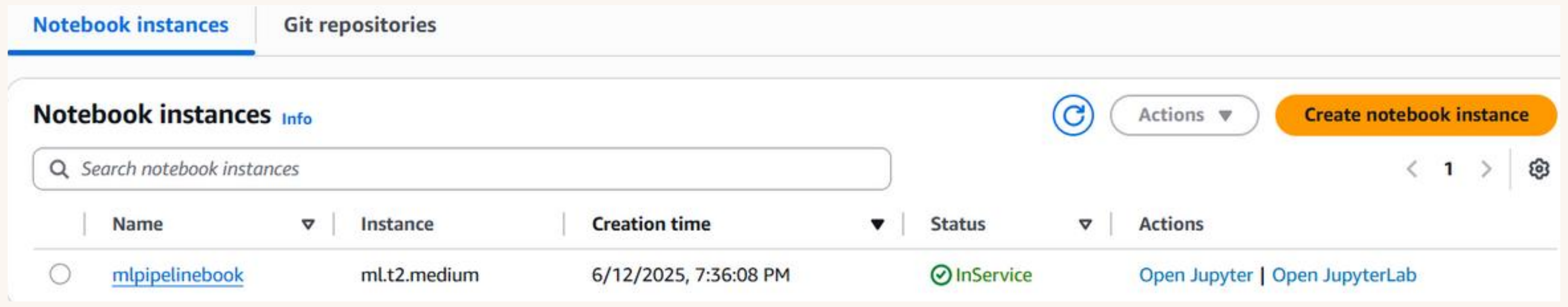
Objects (2)

Objects are the fundamental entities stored in Amazon S3. You can use Amazon S3 inventory to get a list of all objects in your bucket. For others to access your object
more

Name	Type	Last modified	Size	Storage class
model/	Folder	-	-	-
plugin_engagement_data.csv	csv	June 12, 2025, 19:23:21 (UTC-04:00)	32.1 KB	Standard

Launching SageMaker Notebook Instance

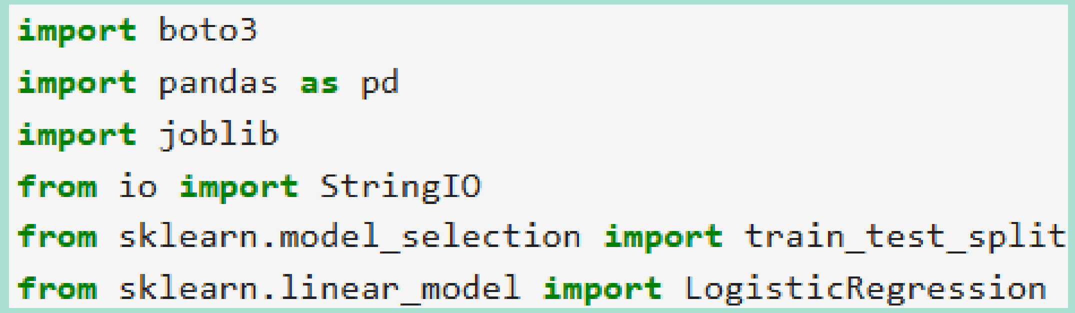
To facilitate model development and data analysis, I launched a SageMaker notebook instance (mlpipelinebook). Utilizing JupyterLab, I efficiently executed Python scripts, seamlessly accessing data from S3 and ensuring consistent storage of both code and machine learning artifacts throughout the project.



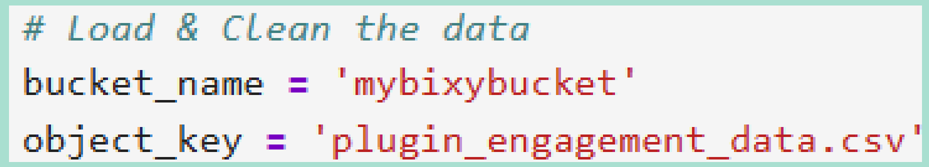
Name	Instance	Creation time	Status	Actions
mlpipelinebook	ml.t2.medium	6/12/2025, 7:36:08 PM	InService	Open Jupyter | Open JupyterLab

Data Loading & Preprocessing (Jupyter Notebook)

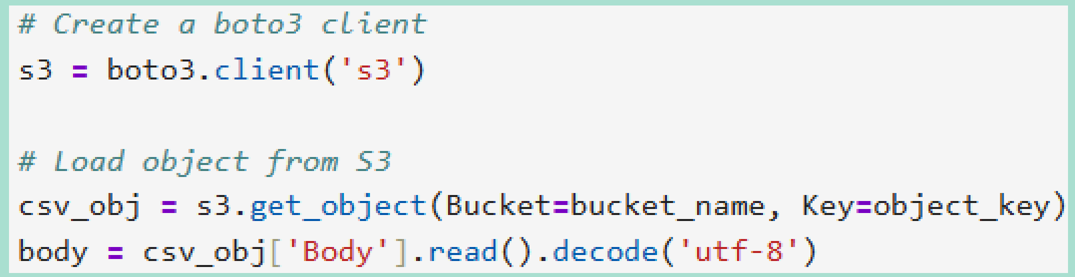
Using Boto3, I programmatically loaded data from S3 into the Jupyter notebook for exploration and verification of key features. The preprocessing steps included addressing missing values, applying one-hot encoding to plugin_type, and performing a 70/30 train-test split. Additional screenshots of this workflow are included in the following slides.



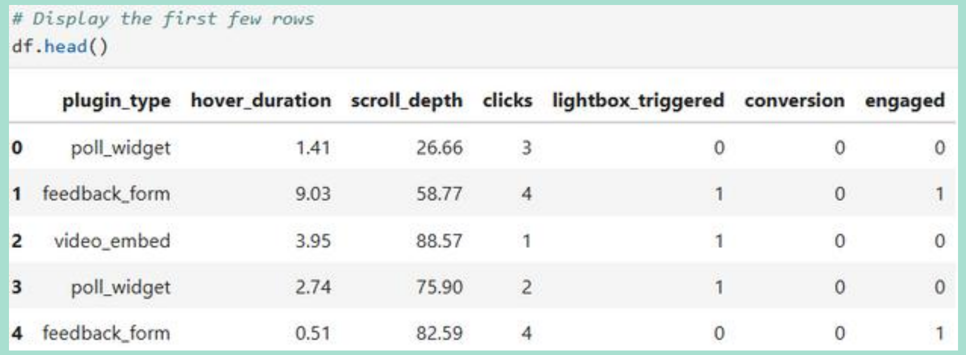
```python
import boto3
import pandas as pd
import joblib
from io import StringIO
from sklearn.model_selection import train_test_split
from sklearn.linear_model import LogisticRegression
```



```python
# Load & Clean the data
bucket_name = 'mybixybucket'
object_key = 'plugin_engagement_data.csv'
```



```python
# Create a boto3 client
s3 = boto3.client('s3')

# Load object from S3
csv_obj = s3.get_object(Bucket=bucket_name, Key=object_key)
body = csv_obj['Body'].read().decode('utf-8')
```

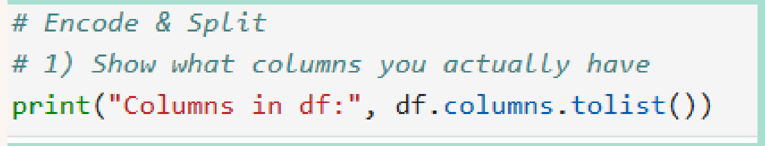


```python
# Display the first few rows
df.head()
```

	plugin_type	hover_duration	scroll_depth	clicks	lightbox_triggered	conversion	engaged
0	poll_widget	1.41	26.66	3	0	0	0
1	feedback_form	9.03	58.77	4	1	0	1
2	video_embed	3.95	88.57	1	1	0	0
3	poll_widget	2.74	75.90	2	1	0	0
4	feedback_form	0.51	82.59	4	0	0	1

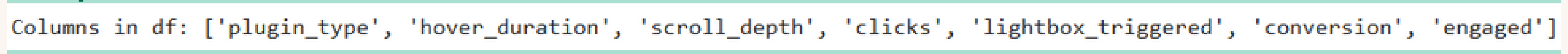
Data Loading & Preprocessing (Jupyter Notebook) (Continued)

Code:



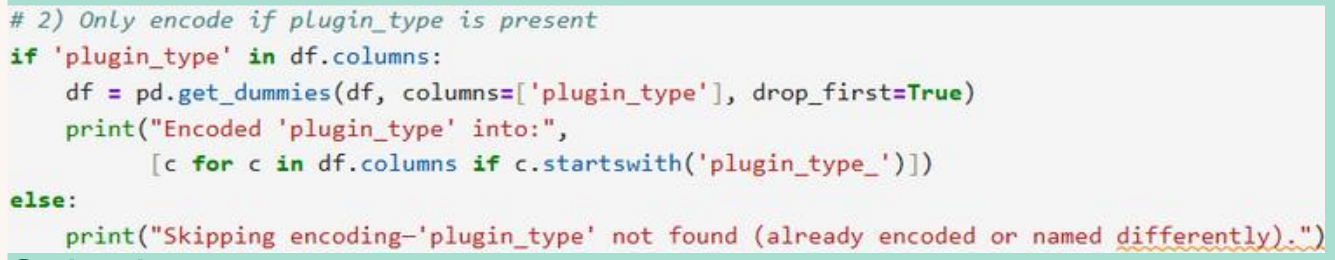
```python
# Encode & Split
# 1) Show what columns you actually have
print("Columns in df:", df.columns.tolist())
```

Output:



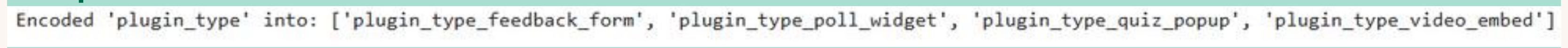
```
Columns in df: ['plugin_type', 'hover_duration', 'scroll_depth', 'clicks', 'lightbox_triggered', 'conversion', 'engaged']
```

Code:



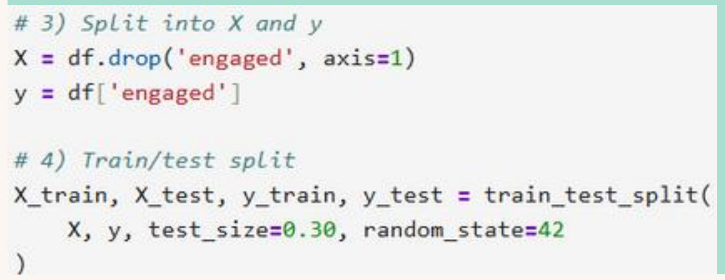
```python
# 2) Only encode if plugin_type is present
if 'plugin_type' in df.columns:
    df = pd.get_dummies(df, columns=['plugin_type'], drop_first=True)
    print("Encoded 'plugin_type' into:",
          [c for c in df.columns if c.startswith('plugin_type_')])
else:
    print("Skipping encoding—'plugin_type' not found (already encoded or named differently).")
```

Output:



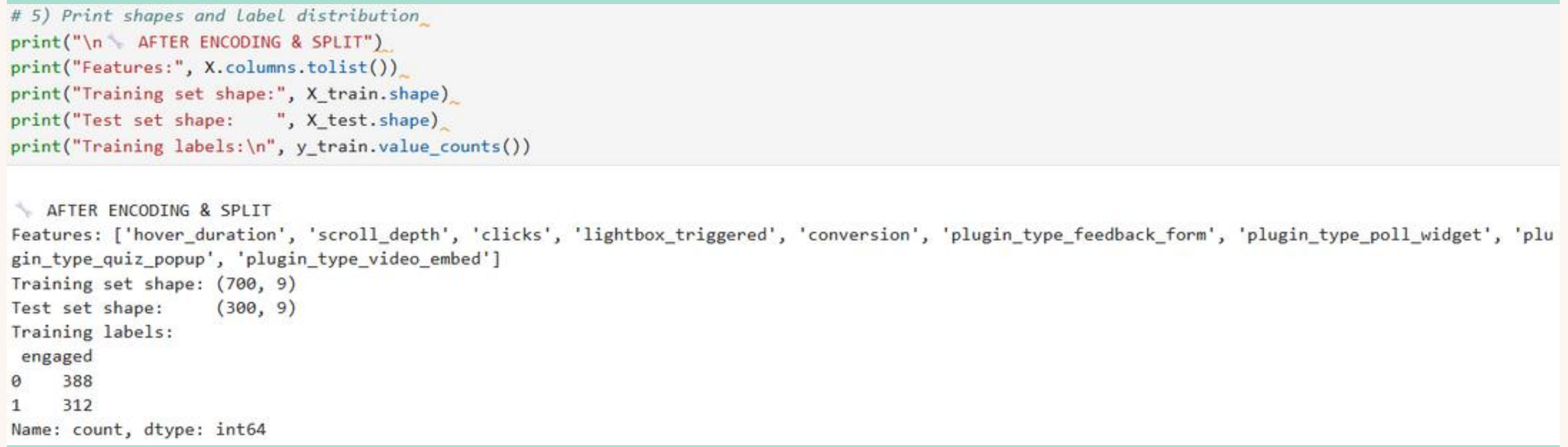
```
Encoded 'plugin_type' into: ['plugin_type_feedback_form', 'plugin_type_poll_widget', 'plugin_type_quiz_popup', 'plugin_type_video_embed']
```

Code:



```python
# 3) Split into X and y
X = df.drop('engaged', axis=1)
y = df['engaged']

# 4) Train/test split
X_train, X_test, y_train, y_test = train_test_split(
    X, y, test_size=0.30, random_state=42
)
```

Code & Output:

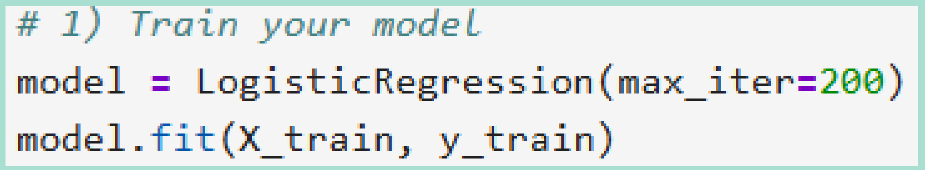


```python
# 5) Print shapes and label distribution
print("\n AFTER ENCODING & SPLIT")
print("Features:", X.columns.tolist())
print("Training set shape:", X_train.shape)
print("Test set shape:    ", X_test.shape)
print("Training labels:\n", y_train.value_counts())
```

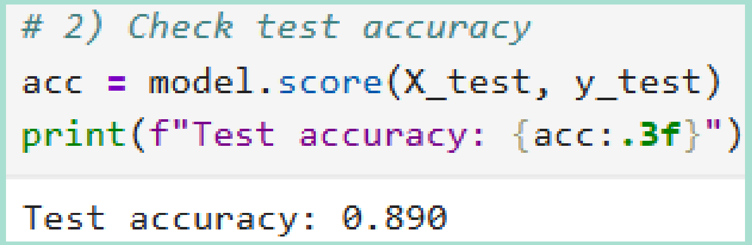
```
 AFTER ENCODING & SPLIT
Features: ['hover_duration', 'scroll_depth', 'clicks', 'lightbox_triggered', 'conversion', 'plugin_type_feedback_form', 'plugin_type_poll_widget', 'plu
gin_type_quiz_popup', 'plugin_type_video_embed']
Training set shape: (700, 9)
Test set shape:     (300, 9)
Training labels:
 engaged
0    388
1    312
Name: count, dtype: int64
```

Model Training, Evaluation, and Saving

Leveraging SageMaker and scikit-learn, I trained a logistic regression model on the processed data, achieving a test accuracy of 0.890. The model was serialized as a joblib file and uploaded to S3, facilitating Lambda integration. Relevant code outputs and performance metrics are shown in the next slides.

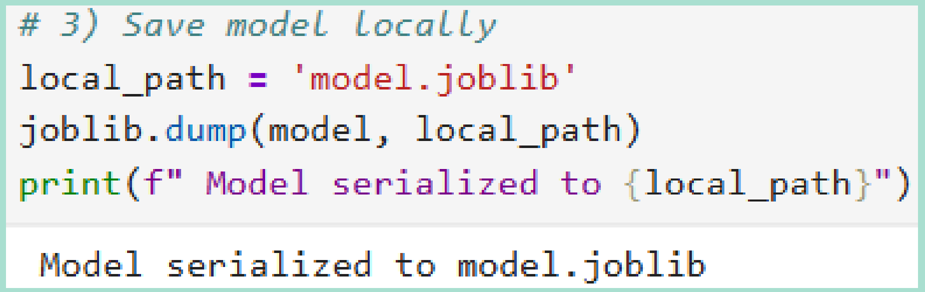


```
# 1) Train your model
model = LogisticRegression(max_iter=200)
model.fit(X_train, y_train)
```



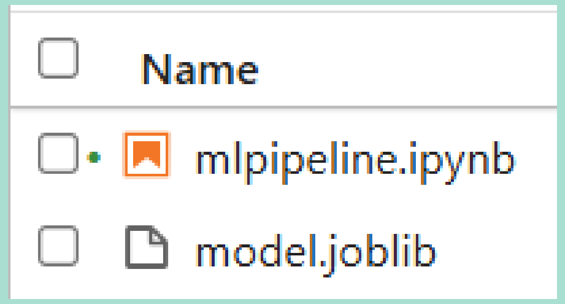
```
# 2) Check test accuracy
acc = model.score(X_test, y_test)
print(f"Test accuracy: {acc:.3f}")
```

```
Test accuracy: 0.890
```

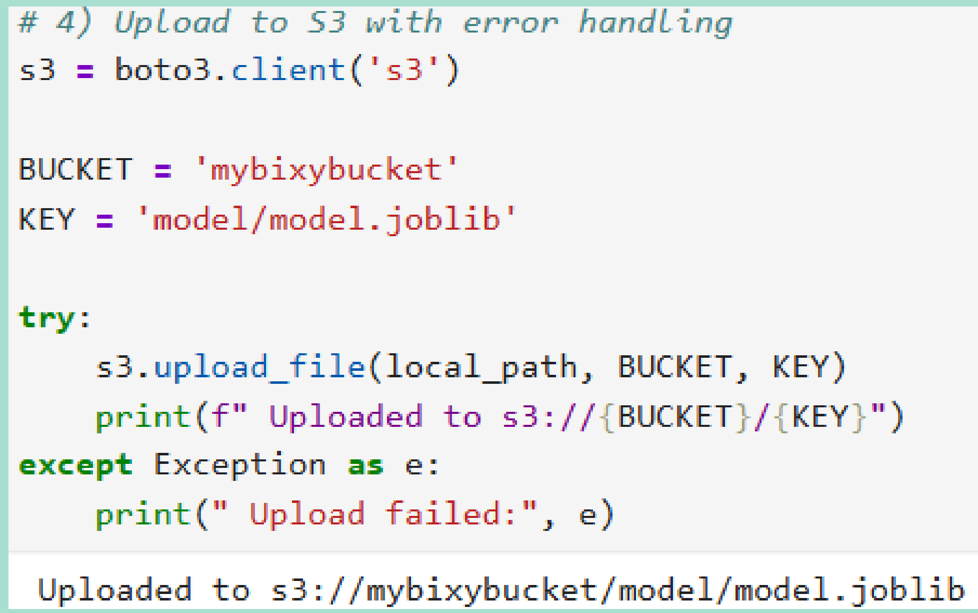


```
# 3) Save model locally
local_path = 'model.joblib'
joblib.dump(model, local_path)
print(f" Model serialized to {local_path}")
```

```
 Model serialized to model.joblib
```



```
Name
mlpipeline.ipynb
model.joblib
```

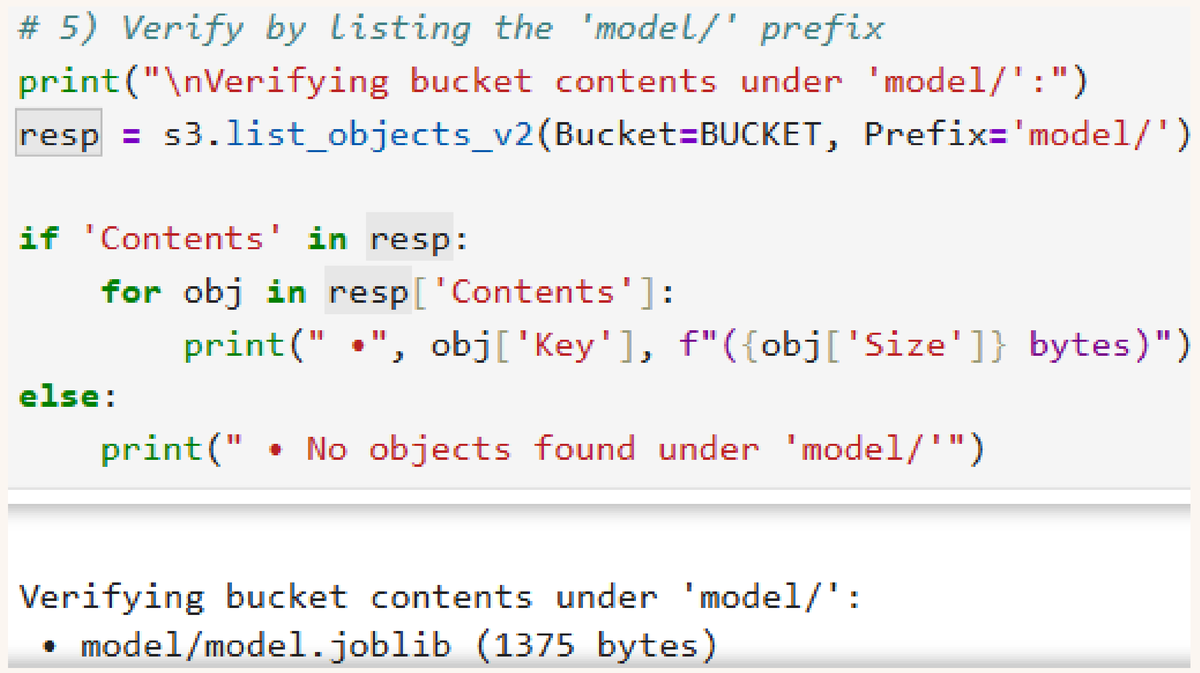


```
# 4) Upload to S3 with error handling
s3 = boto3.client('s3')

BUCKET = 'mybixybucket'
KEY = 'model/model.joblib'

try:
    s3.upload_file(local_path, BUCKET, KEY)
    print(f" Uploaded to s3://{BUCKET}/{KEY}")
except Exception as e:
    print(" Upload failed:", e)
```

```
 Uploaded to s3://mybixybucket/model/model.joblib
```

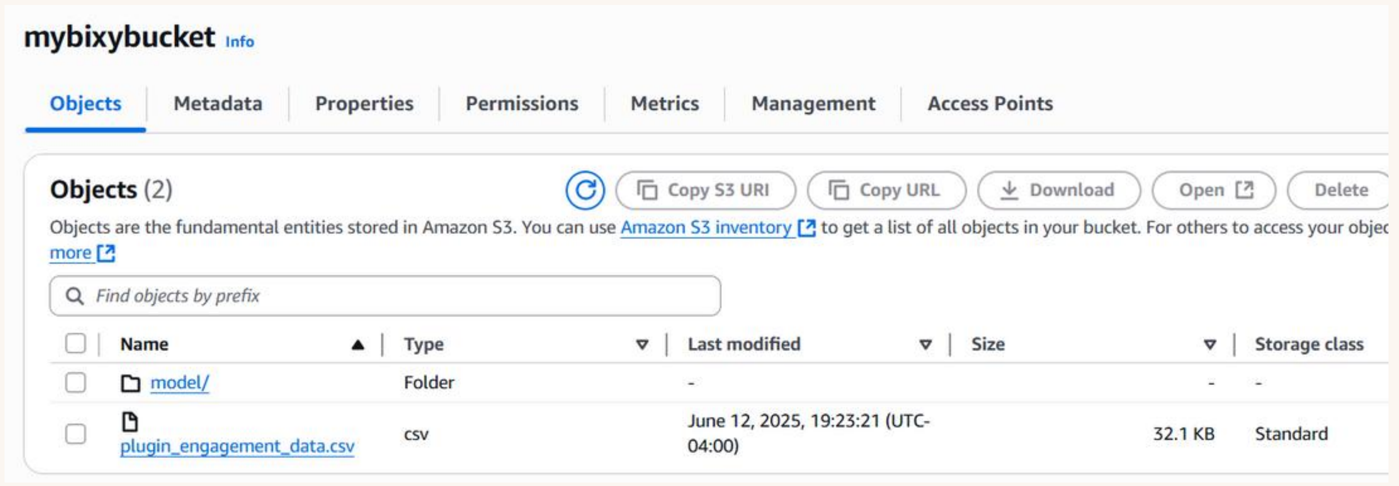
Verifying Model in S3

To verify successful artifact storage, I listed the contents of the model directory within the S3 bucket directly from the notebook. This step confirmed that the serialized model (model/model.joblib) was correctly uploaded and accessible for downstream Lambda inference and integration.



```python
# 5) Verify by listing the 'model/' prefix
print("\nVerifying bucket contents under 'model/':")
resp = s3.list_objects_v2(Bucket=BUCKET, Prefix='model/')

if 'Contents' in resp:
    for obj in resp['Contents']:
        print(" •", obj['Key'], f"({obj['Size']} bytes)")
else:
    print(" • No objects found under 'model/'")
```

```
Verifying bucket contents under 'model/':
 • model/model.joblib (1375 bytes)
```



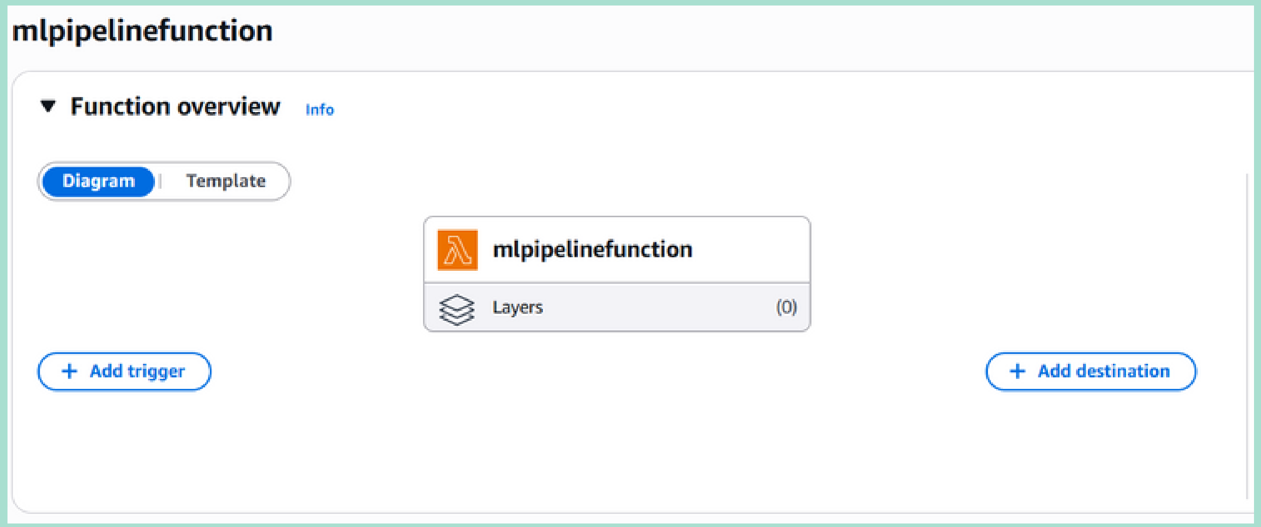
Objects (2)

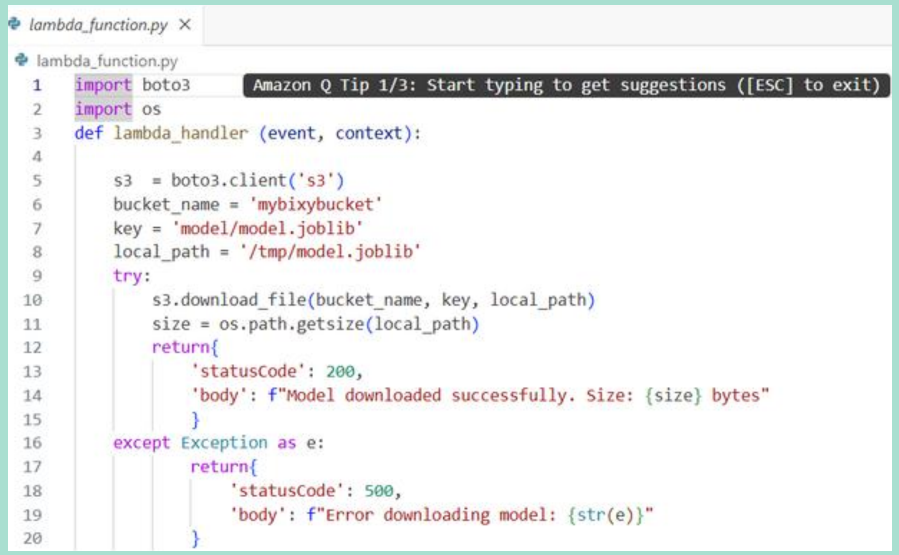
Objects are the fundamental entities stored in Amazon S3. You can use Amazon S3 inventory to get a list of all objects in your bucket. For others to access your objec
more

Name	Type	Last modified	Size	Storage class
model/	Folder	-	-	-
plugin_engagement_data.csv	csv	June 12, 2025, 19:23:21 (UTC-04:00)	32.1 KB	Standard

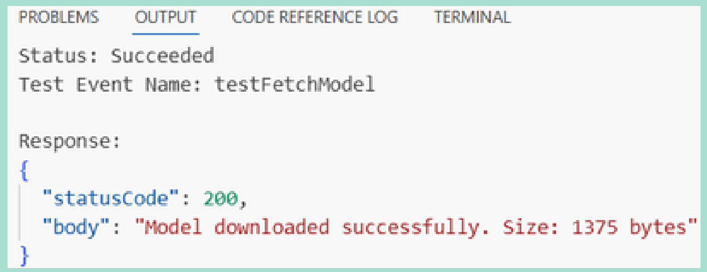
Data Storage on Amazon S3

I developed a Lambda function (mlpipelinefunction) to securely retrieve the trained model from S3 using Boto3. The function validated model accessibility by downloading it to the Lambda environment and returning its file size. Successful test execution confirmed readiness for scalable, serverless inference in a production workflow.




lambda_function.py
lambda_function.py
Amazon Q Tip 1/3: Start typing to get suggestions ([ESC] to exit)

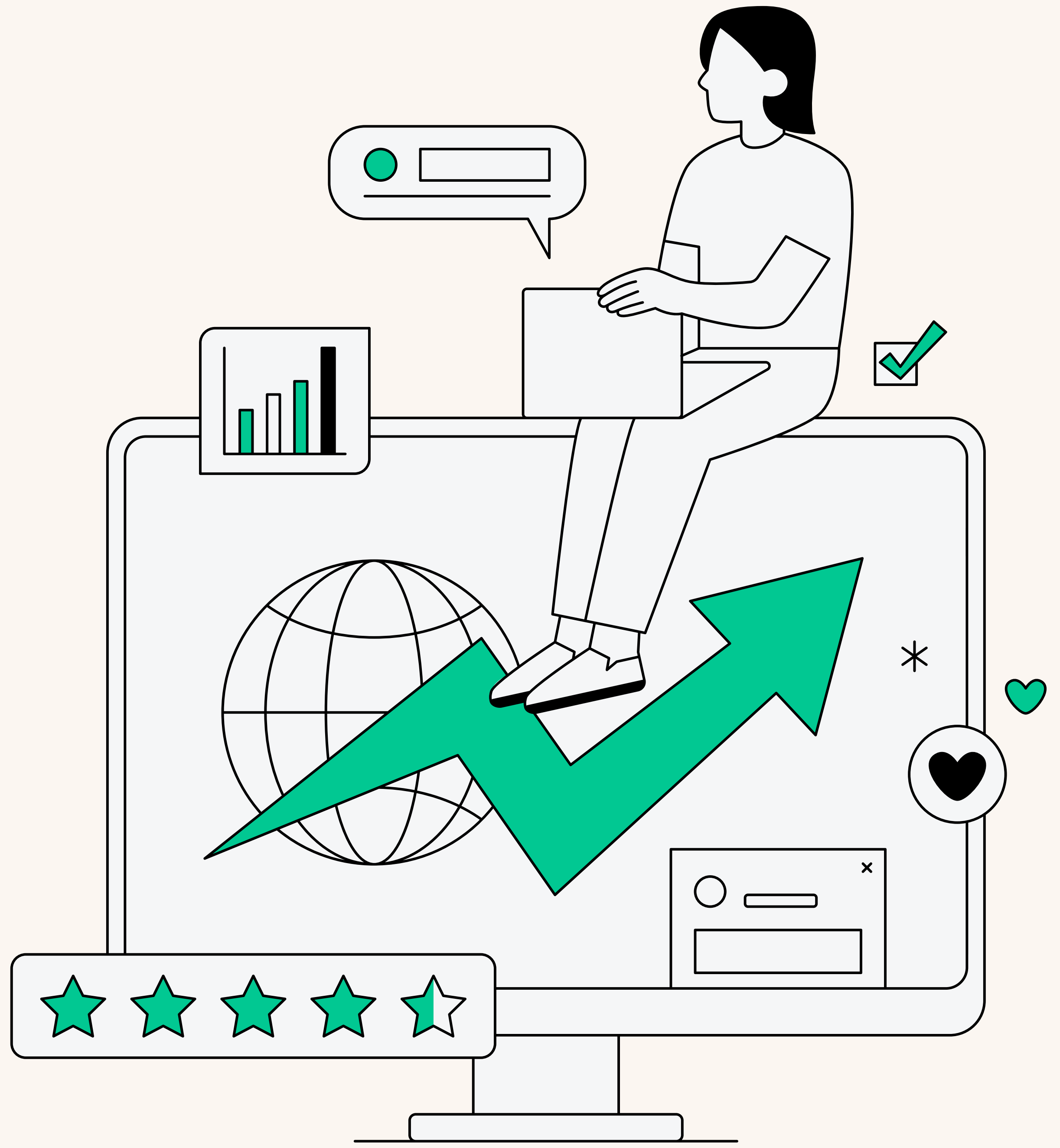
```python
import boto3
import os
def lambda_handler (event, context):

    s3  = boto3.client('s3')
    bucket_name = 'mybixybucket'
    key = 'model/model.joblib'
    local_path = '/tmp/model.joblib'
    try:
        s3.download_file(bucket_name, key, local_path)
        size = os.path.getsize(local_path)
        return{
            'statusCode': 200,
            'body': f"Model downloaded successfully. Size: {size} bytes"
        }
    except Exception as e:
            return{
                'statusCode': 500,
                'body': f"Error downloading model: {str(e)}"
            }
```



```
Status: Succeeded
Test Event Name: testFetchModel

Response:
{
  "statusCode": 200,
  "body": "Model downloaded successfully. Size: 1375 bytes"
}
```

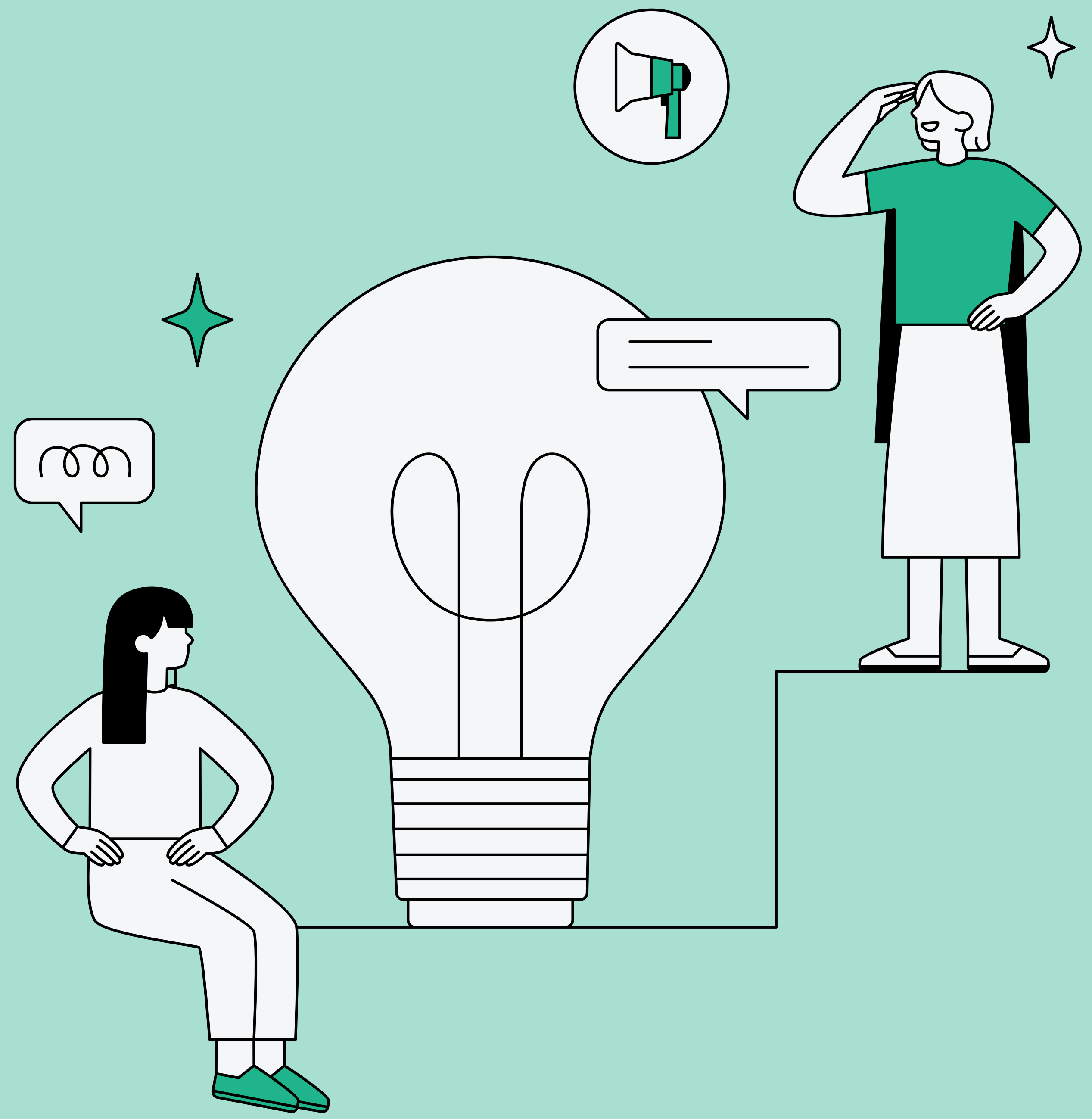
Summary Pipeline Architecture

This end-to-end pipeline leverages core AWS services in a modular design. Data and model artifacts are stored in S3, while SageMaker handles preprocessing and model development. Deployment is achieved through Lambda, enabling a scalable, cost-effective, and agile workflow for machine learning experimentation and inference.



Interpretation & Key Learnings

This project illustrates the real-world deployment of an AWS-based machine learning pipeline, achieving reliable engagement prediction and seamless service integration. Notable achievements include streamlined storage, computation, and inference workflows. Key challenges involved IAM configuration and S3-Lambda integration. Looking ahead, optimizing security policies, automating retraining, and adding API Gateway deployment are recommended next steps.



References

- Amazon Web Services. (2024). Amazon SageMaker documentation. https://docs.aws.amazon.com/sagemaker/
- Amazon Web Services. (2024). AWS Lambda documentation. https://docs.aws.amazon.com/lambda/
- Pedregosa, F., et al. (2011). Scikit-learn: Machine Learning in Python. Journal of Machine Learning Research, 12, 2825-2830.

Presented by Sandra Haro

Thank you very much!

www.reallygreatsite.com