MSADS 508 CyberSentinel Security Solutions

Notebook Dependencies

In [1]:
```
# pip
!pip install --disable-pip-version-check -q pip --upgrade > /dev/null
!pip install --disable-pip-version-check -q wrapt --upgrade > /dev/null
```

In [2]:
```
# AWS CLI and AWS Python SDK (boto3)
!pip install --disable-pip-version-check -q awscli==1.18.216 boto3==1.16.56 botocor
```

In [3]:
```
# SageMaker
!pip install --disable-pip-version-check -q sagemaker==2.29.0
!pip install --disable-pip-version-check -q smdebug==1.0.1
!pip install --disable-pip-version-check -q sagemaker-experiments==0.1.26
```

In [4]:
```
# Hugging Face Transformers (BERT)
!pip install --disable-pip-version-check -q transformers==3.5.1
```

```
  error: subprocess-exited-with-error

  × python setup.py egg_info did not run successfully.
  │ exit code: 1
  ╰─> [2 lines of output]
      /bin/sh: 1: pkg-config: not found
      Failed to find sentencepiece pkgconfig
      [end of output]

  note: This error originates from a subprocess, and is likely not a problem with pi
p.
error: metadata-generation-failed

× Encountered error while generating package metadata.
╰─> See above for output.

note: This is an issue with the package mentioned above, not pip.
hint: See above for details.
```

In [5]:
```
# TorchServe
!pip install --disable-pip-version-check -q torchserve==0.3.0
!pip install --disable-pip-version-check -q torch-model-archiver==0.3.0
```

In [6]:
```
# PyAthena
!pip install --disable-pip-version-check -q PyAthena==2.1.0
```

In [7]:
```
# Redshift
!pip install --disable-pip-version-check -q SQLAlchemy==1.3.22
```

In [8]:
```
# AWS Data Wrangler
!pip install --disable-pip-version-check -q awswrangler==2.15.0
```

```
ERROR: Ignored the following yanked versions: 3.2.0
ERROR: Could not find a version that satisfies the requirement awswrangler==2.15.0
(from versions: 0.0b0, 0.0b2, 0.0b3, 0.0b4, 0.0b5, 0.0b6, 0.0b7, 0.0b8, 0.0b9, 0.0b1
0, 0.0b11, 0.0b12, 0.0b13, 0.0b14, 0.0b15, 0.0b16, 0.0b17, 0.0b18, 0.0b19, 0.0b20,
0.0b21, 0.0b22, 0.0b23, 0.0b24, 0.0b25, 0.0b26, 0.0b27, 0.0b28, 0.0b29, 0.0b30, 0.0b
31, 0.0b32, 0.0.1, 0.0.2, 0.0.3, 0.0.4, 0.0.5, 0.0.6, 0.0.7, 0.0.8, 0.0.9, 0.0.10,
0.0.11, 0.0.12, 0.0.13, 0.0.14, 0.0.15, 0.0.16, 0.0.17, 0.0.18, 0.0.19, 0.0.20, 0.0.
21, 0.0.22, 0.0.23, 0.0.24, 0.0.25, 0.1.0, 0.1.1, 0.1.2, 0.1.3, 0.1.4, 0.2.0, 0.2.1,
0.2.2, 0.2.3, 0.2.4, 0.2.5, 0.2.6, 0.3.0, 0.3.1, 0.3.2, 2.18.0, 2.19.0, 2.20.0, 2.2
0.1, 3.0.0rc1, 3.0.0rc2, 3.0.0rc3, 3.0.0, 3.1.0, 3.1.1, 3.2.1, 3.3.0, 3.4.0, 3.4.1,
3.4.2, 3.5.0, 3.5.1, 3.5.2, 3.6.0, 3.7.0, 3.7.1, 3.7.2, 3.7.3, 3.8.0, 3.9.0, 3.9.1,
3.10.0, 3.10.1, 3.11.0)
ERROR: No matching distribution found for awswrangler==2.15.0
```

In [9]:
```
# StepFunctions
!pip install --disable-pip-version-check -q stepfunctions==2.0.0rc1
```

In [10]:
```
# Matplotlib
!pip install --disable-pip-version-check -q matplotlib==3.1.3
```

```
ERROR: pip's dependency resolver does not currently take into account all the packag
es that are installed. This behaviour is the source of the following dependency conf
licts.
autogluon-multimodal 1.2 requires nvidia-ml-py3==7.352.0, which is not installed.
autogluon-core 1.2 requires matplotlib<3.11,>=3.7.0, but you have matplotlib 3.1.3 w
hich is incompatible.
autogluon-multimodal 1.2 requires jsonschema<4.22,>=4.18, but you have jsonschema 4.
23.0 which is incompatible.
autogluon-multimodal 1.2 requires nltk<3.9,>=3.4.5, but you have nltk 3.9.1 which is
incompatible.
autogluon-multimodal 1.2 requires omegaconf<2.3.0,>=2.1.1, but you have omegaconf 2.
3.0 which is incompatible.
mlflow 2.20.3 requires sqlalchemy<3,>=1.4.0, but you have sqlalchemy 1.3.22 which is
incompatible.
sagemaker-mlflow 0.1.0 requires boto3>=1.34, but you have boto3 1.16.56 which is inc
ompatible.
seaborn 0.13.2 requires matplotlib!=3.6.1,>=3.4, but you have matplotlib 3.1.3 which
is incompatible.
```

In [11]:
```
# Seaborn
!pip install --disable-pip-version-check -q seaborn==0.10.0
!pip install --upgrade seaborn
```

```
Requirement already satisfied: seaborn in /opt/conda/lib/python3.11/site-packages
(0.10.0)
Collecting seaborn
  Using cached seaborn-0.13.2-py3-none-any.whl.metadata (5.4 kB)
Requirement already satisfied: numpy!=1.24.0,>=1.20 in /opt/conda/lib/python3.11/sit
e-packages (from seaborn) (1.26.4)
Requirement already satisfied: pandas>=1.2 in /opt/conda/lib/python3.11/site-package
s (from seaborn) (2.2.3)
Collecting matplotlib!=3.6.1,>=3.4 (from seaborn)
  Using cached matplotlib-3.10.1-cp311-cp311-manylinux_2_17_x86_64.manylinux2014_x86
_64.whl.metadata (11 kB)
Requirement already satisfied: contourpy>=1.0.1 in /opt/conda/lib/python3.11/site-pa
ckages (from matplotlib!=3.6.1,>=3.4->seaborn) (1.3.1)
Requirement already satisfied: cycler>=0.10 in /opt/conda/lib/python3.11/site-packag
es (from matplotlib!=3.6.1,>=3.4->seaborn) (0.12.1)
Requirement already satisfied: fonttools>=4.22.0 in /opt/conda/lib/python3.11/site-p
ackages (from matplotlib!=3.6.1,>=3.4->seaborn) (4.56.0)
Requirement already satisfied: kiwisolver>=1.3.1 in /opt/conda/lib/python3.11/site-p
ackages (from matplotlib!=3.6.1,>=3.4->seaborn) (1.4.7)
Requirement already satisfied: packaging>=20.0 in /opt/conda/lib/python3.11/site-pac
kages (from matplotlib!=3.6.1,>=3.4->seaborn) (24.2)
Requirement already satisfied: pillow>=8 in /opt/conda/lib/python3.11/site-packages
(from matplotlib!=3.6.1,>=3.4->seaborn) (11.1.0)
Requirement already satisfied: pyparsing>=2.3.1 in /opt/conda/lib/python3.11/site-pa
ckages (from matplotlib!=3.6.1,>=3.4->seaborn) (3.2.1)
Requirement already satisfied: python-dateutil>=2.7 in /opt/conda/lib/python3.11/sit
e-packages (from matplotlib!=3.6.1,>=3.4->seaborn) (2.9.0.post0)
Requirement already satisfied: pytz>=2020.1 in /opt/conda/lib/python3.11/site-packag
es (from pandas>=1.2->seaborn) (2024.1)
Requirement already satisfied: tzdata>=2022.7 in /opt/conda/lib/python3.11/site-pack
ages (from pandas>=1.2->seaborn) (2025.1)
Requirement already satisfied: six>=1.5 in /opt/conda/lib/python3.11/site-packages
(from python-dateutil>=2.7->matplotlib!=3.6.1,>=3.4->seaborn) (1.17.0)
Using cached seaborn-0.13.2-py3-none-any.whl (294 kB)
Using cached matplotlib-3.10.1-cp311-cp311-manylinux_2_17_x86_64.manylinux2014_x86_6
4.whl (8.6 MB)
Installing collected packages: matplotlib, seaborn
  Attempting uninstall: matplotlib
    Found existing installation: matplotlib 3.1.3
    Uninstalling matplotlib-3.1.3:
      Successfully uninstalled matplotlib-3.1.3
  Attempting uninstall: seaborn
    Found existing installation: seaborn 0.10.0
    Uninstalling seaborn-0.10.0:
      Successfully uninstalled seaborn-0.10.0
ERROR: pip's dependency resolver does not currently take into account all the packag
es that are installed. This behaviour is the source of the following dependency conf
licts. autogluon-multimodal 1.2 requires nvidia-ml-py3==7.352.0, which is not
installed. autogluon-multimodal 1.2 requires jsonschema<4.22,>=4.18, but you have
jsonschema 4. 23.0 which is incompatible. autogluon-multimodal 1.2 requires
nltk<3.9,>=3.4.5, but you have nltk 3.9.1 which is incompatible. autogluon-multimodal
1.2 requires omegaconf<2.3.0,>=2.1.1, but you have omegaconf 2. 3.0 which is
incompatible. mlflow 2.20.3 requires sqlalchemy<3,>=1.4.0, but you have sqlalchemy
1.3.22 which is
```

```
incompatible.
sagemaker-mlflow 0.1.0 requires boto3>=1.34, but you have boto3 1.16.56 which is inc
ompatible.
Successfully installed matplotlib-3.10.1 seaborn-0.13.2
```

In [12]:
```
setup_dependencies_passed = True
%store setup_dependencies_passed
%store
```

```
Stored 'setup_dependencies_passed' (bool)
Stored variables and their in-db values:
ingest_create_athena_db_passed                  -> True
ingest_create_athena_table_csv_passed           -> True
ingest_create_athena_table_parquet_passed       -> True
s3_cybersentinel_csv                            -> 's3://msads-508-sp25-team6/
MSADS 508 Final Project
s3_private_csv                                  -> 's3://sagemaker-us-east-1-3
67086635748/msads-508-s
s3_public_csv                                   -> 's3://msads-508-sp25-team6'
setup_dependencies_passed                       -> True
setup_iam_roles_passed                          -> True
setup_instance_check_passed                     -> True
setup_s3_bucket_passed                          -> True
```

Check the Environment

In [13]:
```
import boto3

region = boto3.Session().region_name
session = boto3.session.Session()

ec2 = boto3.Session().client(service_name="ec2", region_name=region)
sm = boto3.Session().client(service_name="sagemaker", region_name=region)
```

In [14]:
```
import json

notebook_instance_name = None

try:
    with open("/opt/ml/metadata/resource-metadata.json") as notebook_info:
        data = json.load(notebook_info)
        domain_id = data["DomainId"]
        resource_arn = data["ResourceArn"]
        region = resource_arn.split(":")[3]
        name = data["ResourceName"]
    print("DomainId: {}".format(domain_id))
    print("Name: {}".format(name))
except:
    print("+++++++++++++++++++++++++++++++++++++++++")
    print("[ERROR]: COULD NOT RETRIEVE THE METADATA.")
    print("+++++++++++++++++++++++++++++++++++++++++")
```

```
DomainId: d-euuq0f1eie32
Name: default
```

In [15]:
```
describe_domain_response = sm.describe_domain(DomainId=domain_id)
print(describe_domain_response["Status"])
```

```
InService
```

In [16]:
```
try:
    get_status_response = sm.get_sagemaker_servicecatalog_portfolio_status()
    print(get_status_response["Status"])
except:
    pass
```

```
Enabled
```

In [17]:
```
if (
    describe_domain_response["Status"] == "InService"
    and get_status_response["Status"] == "Enabled"
    and "default" in name
):
    setup_instance_check_passed = True
    print("[OK] Checks passed! Great Job!! Please Continue.")
else:
    setup_instance_check_passed = False
    print("++++++++++++++++++++++++++++++++++++++++++++++++")
    print("[ERROR]: WE HAVE IDENTIFIED A MISCONFIGURATION.")
    print(describe_domain_response["Status"])
    print(get_status_response["Status"])
    print(name)
    print("++++++++++++++++++++++++++++++++++++++++++++++++")
```

```
[OK] Checks passed! Great Job!! Please Continue.
```

In [18]:
```
print(setup_instance_check_passed)
%store setup_instance_check_passed
%store
```

```
True
Stored 'setup_instance_check_passed' (bool)
Stored variables and their in-db values:
ingest_create_athena_db_passed                  -> True
ingest_create_athena_table_csv_passed           -> True
ingest_create_athena_table_parquet_passed       -> True
s3_cybersentinel_csv                            -> 's3://msads-508-sp25-team6/
MSADS 508 Final Project
s3_private_csv                                  -> 's3://sagemaker-us-east-1-3
67086635748/msads-508-s
s3_public_csv                                   -> 's3://msads-508-sp25-team6'
setup_dependencies_passed                       -> True
setup_iam_roles_passed                          -> True
setup_instance_check_passed                     -> True
setup_s3_bucket_passed                          -> True
```

Create S3 Bucket

In [19]:
```
import boto3
import sagemaker
```

```
session = boto3.session.Session()
region = session.region_name
sagemaker_session = sagemaker.Session()
bucket = sagemaker_session.default_bucket()

s3 = boto3.Session().client(service_name="s3", region_name=region)

setup_s3_bucket_passed = False

print("Default bucket: {}".format(bucket))
```

```
Default bucket: sagemaker-us-east-1-367086635748
```

Verify S3 Bucket Creation

In [20]:
```
%%bash

aws s3 ls s3://${bucket}/
```

```
2025-03-12 02:56:42 msads-508-sp25-team6
2025-03-12 01:49:54 sagemaker-studio-h7mr28bvu1u
2025-03-12 02:26:51 sagemaker-studio-ixhrch8ljza
25-03-12 01:49:56 sagemaker-us-east-1-367086635748
```

In [21]:
```
from botocore.client import ClientError

response = None

try:
    response = s3.head_bucket(Bucket=bucket)
    print(response)
    setup_s3_bucket_passed = True
except ClientError as e:
    print("[ERROR] Cannot find bucket {} in {} due to {}.".format(bucket, response,
```

```
{'ResponseMetadata': {'RequestId': '9AM0P33HA7WKDK5Q', 'HostId': '1kV5enPsFsjf3w/8p/
FKnE1KOJXKwu7azm1dI+cqY+3Bar2cbq0Vpxrdjb+sglczVMaKLNFLboDPb+wYanJka1T4qCXHjZokDODXW7
Zf8NY=', 'HTTPStatusCode': 200, 'HTTPHeaders': {'x-amz-id-2': '1kV5enPsFsjf3w/8p/FKn
E1KOJXKwu7azm1dI+cqY+3Bar2cbq0Vpxrdjb+sglczVMaKLNFLboDPb+wYanJka1T4qCXHjZokDODXW7Zf8
NY=', 'x-amz-request-id': '9AM0P33HA7WKDK5Q', 'date': 'Mon, 14 Apr 2025 05:30:42 GM
T', 'x-amz-bucket-region': 'us-east-1', 'x-amz-access-point-alias': 'false', 'conten
t-type': 'application/xml', 'transfer-encoding': 'chunked', 'server': 'AmazonS3'},
'RetryAttempts': 0}}
```

In [22]:
```
%store setup_s3_bucket_passed

%store
```

```
Stored 'setup_s3_bucket_passed' (bool)
Stored variables and their in-db values:
ingest_create_athena_db_passed                          -> True -> True -> True ->
ingest_create_athena_table_csv_passed                   's3://msads-508-sp25-team6/
ingest_create_athena_table_parquet_passed
s3_cybersentinel_csv
MSADS 508 Final Project
s3_private_csv                                          -> 's3://sagemaker-us-east-1-3
67086635748/msads-508-s
s3_public_csv                                           -> 's3://msads-508-sp25-team6'
setup_dependencies_passed                               -> True -> True -> True ->
setup_iam_roles_passed                                  True
setup_instance_check_passed
setup_s3_bucket_passed
```

Update IAM Roles and Policies

In [23]:
```python
import boto3
import sagemaker
import time
from time import gmtime, strftime

sagemaker_session = sagemaker.Session()
role = sagemaker.get_execution_role()
bucket = sagemaker_session.default_bucket()
region = boto3.Session().region_name

from botocore.config import Config

config = Config(retries={"max_attempts": 10, "mode": "adaptive"})

iam = boto3.client("iam", config=config)
```

In [24]:
```python
role_name = role.split("/")[-1]

print("Role name: {}".format(role_name))
```

```
Role name: LabRole
```

In [25]:
```python
setup_iam_roles_passed = False
```

Pre-Requisite: SageMaker notebook instance ExecutionRole contains IAMFullAccess Policy.

In [26]:
```python
admin = False
post_policies = iam.list_attached_role_policies(RoleName=role_name)["AttachedPolici
for post_policy in post_policies:
    if post_policy["PolicyName"] == "AdministratorAccess":
        admin = True
        break

setup_iam_roles_passed = True
print("[OK] You are all set up to continue with this workshop!")
```

```
[OK] You are all set up to continue with this workshop!
```

In [27]:
```
%store setup_iam_roles_passed

%store
```

```
Stored 'setup_iam_roles_passed' (bool)
Stored variables and their in-db values:
ingest_create_athena_db_passed                  -> True
ingest_create_athena_table_csv_passed           -> True
ingest_create_athena_table_parquet_passed       -> True
s3_cybersentinel_csv                            -> 's3://msads-508-sp25-team6/
MSADS 508 Final Project
s3_private_csv                                  -> 's3://sagemaker-us-east-1-3
67086635748/msads-508-s
s3_public_csv                                   -> 's3://msads-508-sp25-team6'
setup_dependencies_passed                       -> True
setup_iam_roles_passed                          -> True
setup_instance_check_passed                     -> True
setup_s3_bucket_passed                          -> True
```

S3 Bucket Connection & Local File Download

In [28]:
```
import boto3
import sagemaker
import pandas as pd

sess = sagemaker.Session()
bucket = sess.default_bucket()
role = sagemaker.get_execution_role()
region = boto3.Session().region_name
account_id = boto3.client("sts").get_caller_identity().get("Account")

sm = boto3.Session().client(service_name="sagemaker", region_name=region)
```

Set S3 Source Location

In [29]:
```
s3_public_csv = "s3://msads-508-sp25-team6"

%store s3_public_csv
```

```
Stored 's3_public_csv' (str)
```

Set S3 Destination Location

In [30]:
```
s3_private_csv = "s3://{}/msads-508-sp25-team6".format(bucket)
print(s3_private_csv)

%store s3_private_csv
```

```
s3://sagemaker-us-east-1-367086635748/msads-508-sp25-team6
Stored 's3_private_csv' (str)
```

Copy Data From the Public S3 Bucket to the Private S3 Bucket in this Account

In [31]:
```
!aws s3 cp --recursive $s3_public_csv/ $s3_private_csv/ --exclude "*" --include "MS
```

```
copy: s3://msads-508-sp25-team6/MSADS 508 Final Project.csv to s3://sagemaker-us-eas
t-1-367086635748/msads-508-sp25-team6/MSADS 508 Final Project.csv
```

List Files in the Private S3 Bucket in this Account

In [32]:
```
print(s3_private_csv)


!aws s3 ls $s3_private_csv/
```

```
s3://sagemaker-us-east-1-367086635748/msads-508-sp25-team6
                           PRE parquet/
2025-04-14 05:30:44 1138005184 MSADS 508 Final Project.csv
```

Store Variables

In [33]:
```
%store
```

```
Stored variables and their in-db values:
ingest_create_athena_db_passed                  -> True
ingest_create_athena_table_csv_passed           -> True
ingest_create_athena_table_parquet_passed       -> True
s3_cybersentinel_csv                            -> 's3://msads-508-sp25-team6/
MSADS 508 Final Project
s3_private_csv                                  -> 's3://sagemaker-us-east-1-3
67086635748/msads-508-s
s3_public_csv                                   -> 's3://msads-508-sp25-team6'
setup_dependencies_passed                       -> True
setup_iam_roles_passed                          -> True
setup_instance_check_passed                     -> True
setup_s3_bucket_passed                          -> True
```

Athena Database Schema

In [34]:
```
import boto3
import sagemaker

sess = sagemaker.Session()
bucket = sess.default_bucket()
role = sagemaker.get_execution_role()
region = boto3.Session().region_name

ingest_create_athena_db_passed = False
```

Import PyAthena

In [35]:
```
!pip install --disable-pip-version-check -q PyAthena==2.1.0
from pyathena import connect
```

Create Athena Database

In [36]:
```
database_name = "cybersentinel"

# Set S3 staging directory -- this is a temporary directory used for Athena queries
s3_staging_dir = "s3://{0}/athena/staging".format(bucket)

conn = connect(region_name = region, s3_staging_dir = s3_staging_dir)

statement = "CREATE DATABASE IF NOT EXISTS {}".format(database_name)
print(statement)

import pandas as pd
pd.read_sql(statement, conn)
```

```
CREATE DATABASE IF NOT EXISTS cybersentinel
```

```
/tmp/ipykernel_5110/2904263291.py:12: UserWarning: pandas only supports SQLAlchemy c
onnectable (engine/connection) or database string URI or sqlite3 DBAPI2 connection.
Other DBAPI2 objects are not tested. Please consider using SQLAlchemy.
  pd.read_sql(statement, conn)
```

Out[36]: —

Verify the Database has been Created

In [37]:
```
statement = "SHOW DATABASES"

df_show = pd.read_sql(statement, conn)
df_show.head(5)
```

```
/tmp/ipykernel_5110/3999478089.py:3: UserWarning: pandas only supports SQLAlchemy co
nnectable (engine/connection) or database string URI or sqlite3 DBAPI2 connection. O
ther DBAPI2 objects are not tested. Please consider using SQLAlchemy.
  df_show = pd.read_sql(statement, conn)
```

Out[37]:

	database_name
0	cybersentinel
1	default

In [38]:
```
if database_name in df_show.values:
    ingest_create_athena_db_passed = True
```

In [39]:
```
%store ingest_create_athena_db_passed
```

```
Stored 'ingest_create_athena_db_passed' (bool)
```

In [40]:
```
%store
```

```
Stored variables and their in-db values:
ingest_create_athena_db_passed                          -> True -> True -> True ->
ingest_create_athena_table_csv_passed                   's3://msads-508-sp25-team6/
ingest_create_athena_table_parquet_passed
s3_cybersentinel_csv
MSADS 508 Final Project
s3_private_csv                                          -> 's3://sagemaker-us-east-1-3
67086635748/msads-508-s
s3_public_csv                                           -> 's3://msads-508-sp25-team6'
setup_dependencies_passed                               -> True -> True -> True ->
setup_iam_roles_passed                                  True
setup_instance_check_passed
setup_s3_bucket_passed
```

Create Athena Table from Local CSV File

In [41]:
```
# Set S3 staging directory -- this is a temporary directory used for Athena queries
s3_staging_dir = "s3://{0}/athena/staging".format(bucket)
```

In [42]:
```
# Set Athena parameters
database_name = "cybersentinel"
table_name_csv = "cyber_sentinel_security_csv"
```

In [43]:
```
conn = connect(region_name = region, s3_staging_dir = s3_staging_dir)
```

In [44]:
```
# SQL statement to execute
statement = """CREATE EXTERNAL TABLE IF NOT EXISTS {}.{}(
            source_ip string, source_port
            string,         destination_ip
            string,       destination_port
            string,    protocol    string,
            flow_duration           float,
            total_fwd_packets         int,
            total_backward_packets    int,
            fwd_packet_length_mean  float,
            bwd_packet_length_mean  float,
            label string

) ROW FORMAT DELIMITED FIELDS TERMINATED BY '\\,' LINES TERMINATED BY '\\n' LOCATIO
TBLPROPERTIES ('skip.header.line.count' = '1')""".format(
    database_name, table_name_csv, s3_private_csv
)

print(statement)
```

```
CREATE EXTERNAL TABLE IF NOT EXISTS cybersentinel.cyber_sentinel_security_csv(
            source_ip  string,  source_port
            string,         destination_ip
            string,       destination_port
            string,    protocol    string,
            flow_duration           float,
            total_fwd_packets         int,
            total_backward_packets    int,
            fwd_packet_length_mean  float,
            bwd_packet_length_mean  float,
            label string

) ROW FORMAT DELIMITED FIELDS TERMINATED BY '\,' LINES TERMINATED BY '\n' LOCATION
's3://sagemaker-us-east-1-367086635748/msads-508-sp25-team6'
TBLPROPERTIES ('skip.header.line.count' = '1')
```

In [45]:
```
import pandas as pd

pd.read_sql(statement, conn)
```

```
/tmp/ipykernel_5110/3803073958.py:3: UserWarning: pandas only supports SQLAlchemy co
nnectable (engine/connection) or database string URI or sqlite3 DBAPI2 connection. O
ther DBAPI2 objects are not tested. Please consider using SQLAlchemy.
  pd.read_sql(statement, conn)
```

Out[45]: —

Verify the table has been created

In [46]:
```
statement = "SHOW TABLES in {}".format(database_name)

df_show = pd.read_sql(statement, conn)
df_show.head(5)
```

```
/tmp/ipykernel_5110/2201015668.py:3: UserWarning: pandas only supports SQLAlchemy co
nnectable (engine/connection) or database string URI or sqlite3 DBAPI2 connection. O
ther DBAPI2 objects are not tested. Please consider using SQLAlchemy.
  df_show = pd.read_sql(statement, conn)
```

Out[46]:

	tab_name
0	cyber_sentinel_security_csv
1	cyber_sentinel_security_parquet

In [47]:
```
if table_name_csv in df_show.values:
    ingest_create_athena_table_csv_passed = True
```

In [48]:
```
%store ingest_create_athena_table_csv_passed
```

```
Stored 'ingest_create_athena_table_csv_passed' (bool)
```

Run a sample query

In [49]:
```
label = "benign"

statement = """SELECT * FROM {}.{}
        WHERE label = '{}' LIMIT 100""".format(
            database_name, table_name_csv, label
        )

print(statement)
```

```
SELECT * FROM cybersentinel.cyber_sentinel_security_csv
        WHERE label = 'benign' LIMIT 100
```

In [50]:
```
df = pd.read_sql(statement, conn)

df.head(5)
```

```
/tmp/ipykernel_5110/2219640993.py:1: UserWarning: pandas only supports SQLAlchemy co
nnectable (engine/connection) or database string URI or sqlite3 DBAPI2 connection. O
ther DBAPI2 objects are not tested. Please consider using SQLAlchemy.
  df = pd.read_sql(statement, conn)
```

Out[50]:

	source_ip	source_por t	destination_ip	destination_por t	protocol	flow_duration	tot
0	209.85.202.95	443	172.31.65.71	50584	tcp	25.0	
1	23.46.61.157	443	172.31.67.24	50170	tcp	36.0	
2	172.31.65.84	49672	13.89.188.5	443	tcp	60091604.0	
3	172.31.65.44	51722	172.31.0.2	53	udp	401.0	
4	45.60.121.36	443	172.31.67.41	51426	tcp	55.0	

In [51]:
```
%store
```

```
Stored variables and their in-db values:
ingest_create_athena_db_passed                  -> True
ingest_create_athena_table_csv_passed           -> True
ingest_create_athena_table_parquet_passed       -> True
s3_cybersentinel_csv                            -> 's3://msads-508-sp25-team6/
MSADS 508 Final Project
s3_private_csv                                  -> 's3://sagemaker-us-east-1-3
67086635748/msads-508-s
s3_public_csv                                   -> 's3://msads-508-sp25-team6'
setup_dependencies_passed                       -> True
setup_iam_roles_passed                          -> True
setup_instance_check_passed                     -> True
setup_s3_bucket_passed                          -> True
```

Data Engineering

In [52]:
```
import boto3
import sagemaker
import pandas as pd
```

```
sess = sagemaker.Session()
bucket = sess.default_bucket()
role = sagemaker.get_execution_role()
region = boto3.Session().region_name
```

In [53]:
```
!aws s3 cp s3://msads-508-sp25-team6/MSADS\ 508\ Final\ Project.csv .
```

```
download: s3://msads-508-sp25-team6/MSADS 508 Final Project.csv to ./MSADS 508 Final
Project.csv
```

In [54]:
```
df = pd.read_csv("MSADS 508 Final Project.csv")
```

```
/tmp/ipykernel_5110/3096427248.py:1: DtypeWarning: Columns (1,3) have mixed types. S
pecify dtype option on import or set low_memory=False.
  df = pd.read_csv("MSADS 508 Final Project.csv")
```

In [55]:
```
df.head()
```

Out[55]:

	Source IP	Source Port	Destination IP	Destination Port	Protocol	Flow Duration	Total Fwd Packets	Total Backward Packets
0	192.168.4.11	450	203.73.24.7	8	tcp	3974862.0	29	44
1	8	4	5	0	tcp	63.0	1	1
2	192.168.4.11	450	203.73.24.7	8	tcp	476078.0	2	6
3	8	4	5	0	tcp	151.0	2	1
4	192.168.4.11	450	203.73.24.7	8	tcp	472507.0	2	5
	8	5	5	0				
	192.168.4.11	450	203.73.24.7	8				
	8	5	5	0				
	192.168.4.11	450	203.73.24.7	8				
	8	6	5	0				

In [56]:
```
# Check for missing values in the dataset
missing_values = df.isnull().sum()

# Display summary statistics
summary_stats = df.describe()
# Display results
missing_values, summary_stats
```

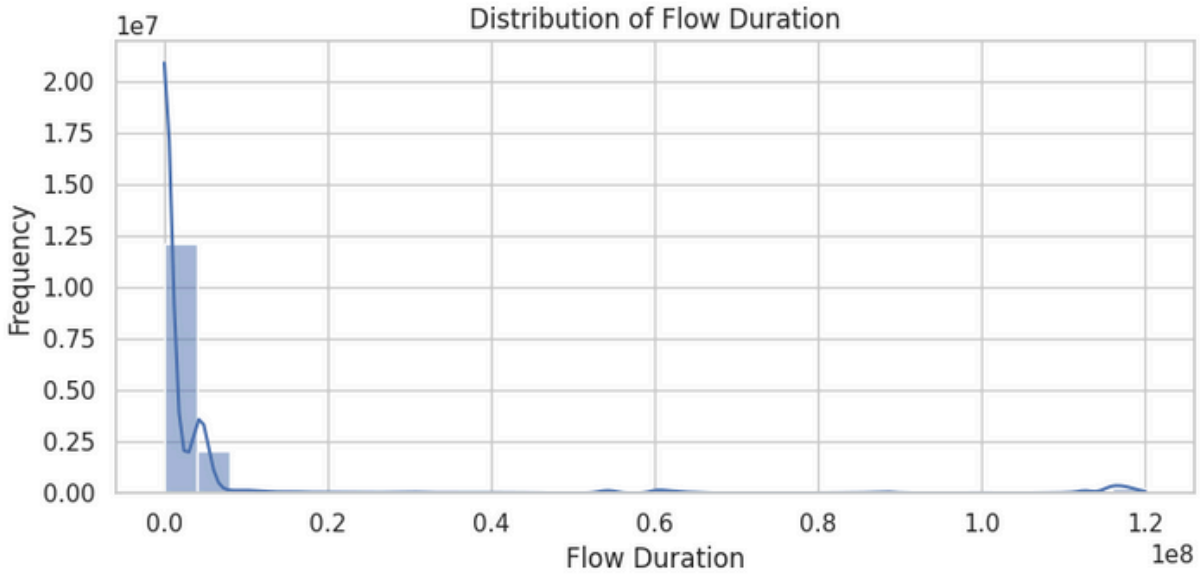
Out[56]:
```
(Source IP   Source Port   0
Destination           IP   0
Destination         Port   0
Protocol   Flow Duration   0
Total Fwd Packets  Total   0
Backward  Packets    Fwd   0
Packet Length Mean   Bwd   0
Packet    Length    Mean   0
Label  dtype: int64,       0
                          0
                          0
```

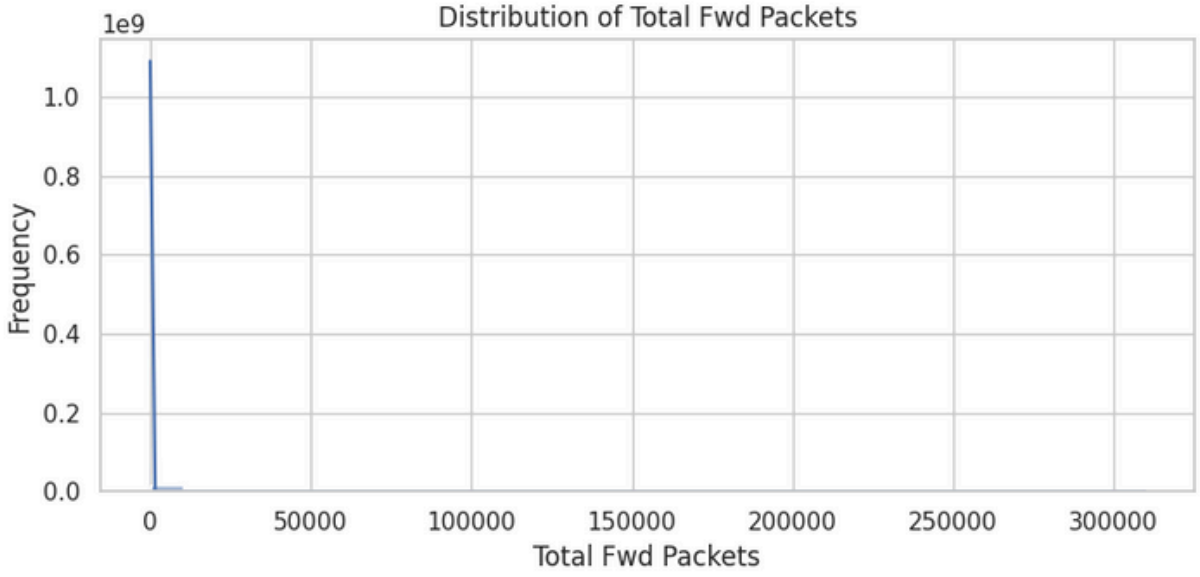
	Flow Duration	Total Fwd Packets	Total Backward Packets
count	1.556042e+07	1.556042e+07	1.556042e+07
mean	6.994334e+06	2.786705e+01	1.113106e+01
std	2.301101e+07	1.560498e+03	2.331984e+02
min	-1.000000e+00	0.000000e+00	0.000000e+00
25%	3.840000e+02	1.000000e+00	1.000000e+00
50%	1.216100e+04	2.000000e+00	2.000000e+00
75%	2.335781e+06	4.000000e+00	5.000000e+00
max	1.200000e+08	3.096280e+05	2.919230e+05

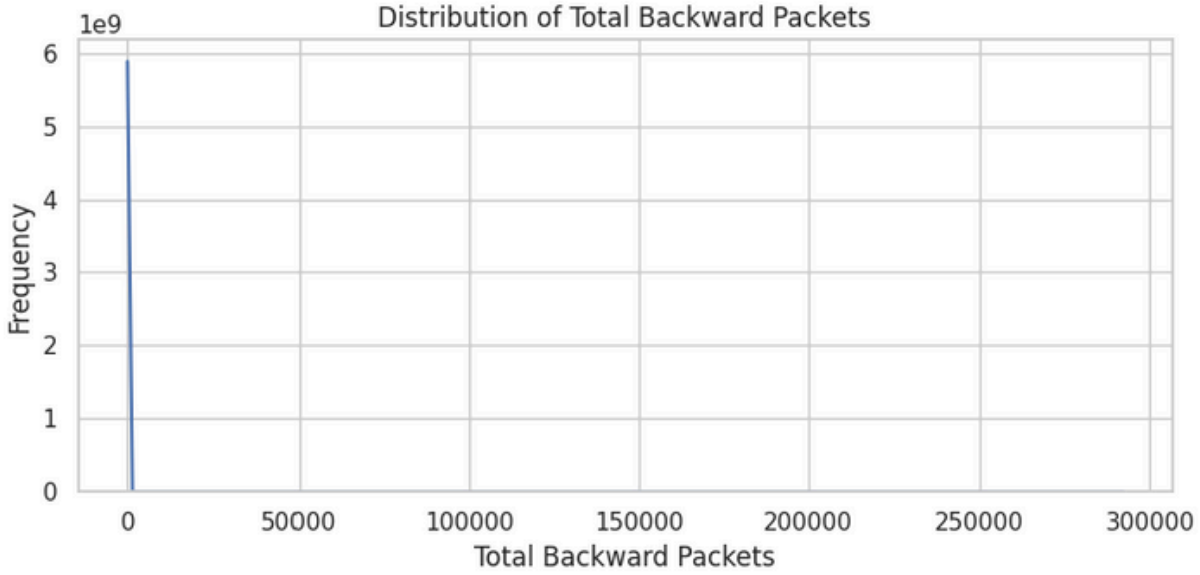
	Fwd Packet Length Mean	Bwd Packet Length Mean
count	1.556042e+07	1.556042e+07
mean	1.336425e+02	8.534740e+01
std	2.571392e+02	2.232792e+02
min	0.000000e+00	0.000000e+00
25%	0.000000e+00	0.000000e+00
50%	7.800000e+01	1.584615e+01
75%	1.270000e+02	2.320000e+02
max	3.037508e+04	4.660441e+03

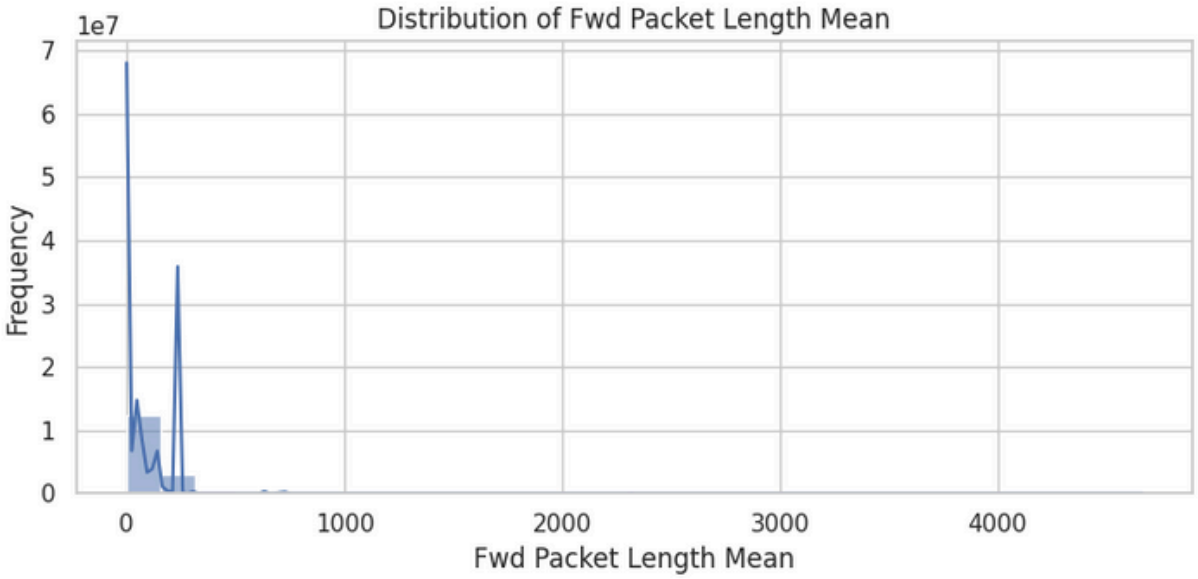
In [57]:
```
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns
```

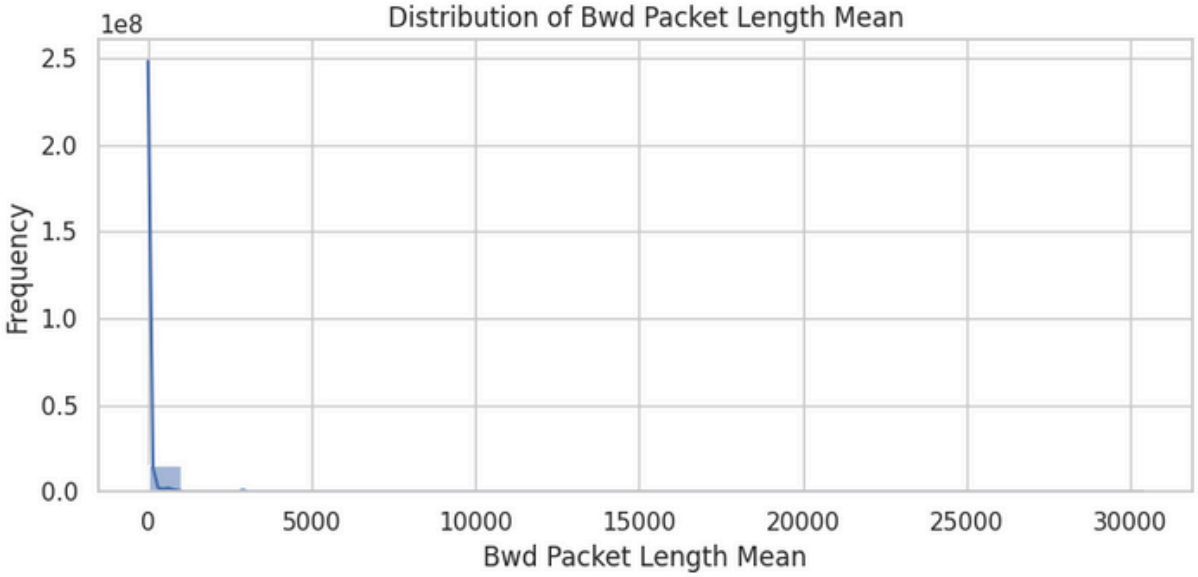
In [58]:
```
# Set plot styles
sns.set(style="whitegrid")

# Plot 1: Distribution of numerical features
num_cols = df.select_dtypes(include=['int64', 'float64']).columns
for col in num_cols:
    plt.figure(figsize=(8, 4))
    sns.histplot(df[col], kde=True, bins=30)
    plt.title(f'Distribution of {col}')
    plt.xlabel(col)
    plt.ylabel('Frequency')
    plt.tight_layout()
    plt.show()
```


Distribution of Flow Duration
1e7
2.00
1.75
1.50
1.25
1.00
0.75
0.50
0.25
0.00
Frequency
0.0
0.2
0.4
0.6
0.8
1.0
1.2
1e8
Flow Duration

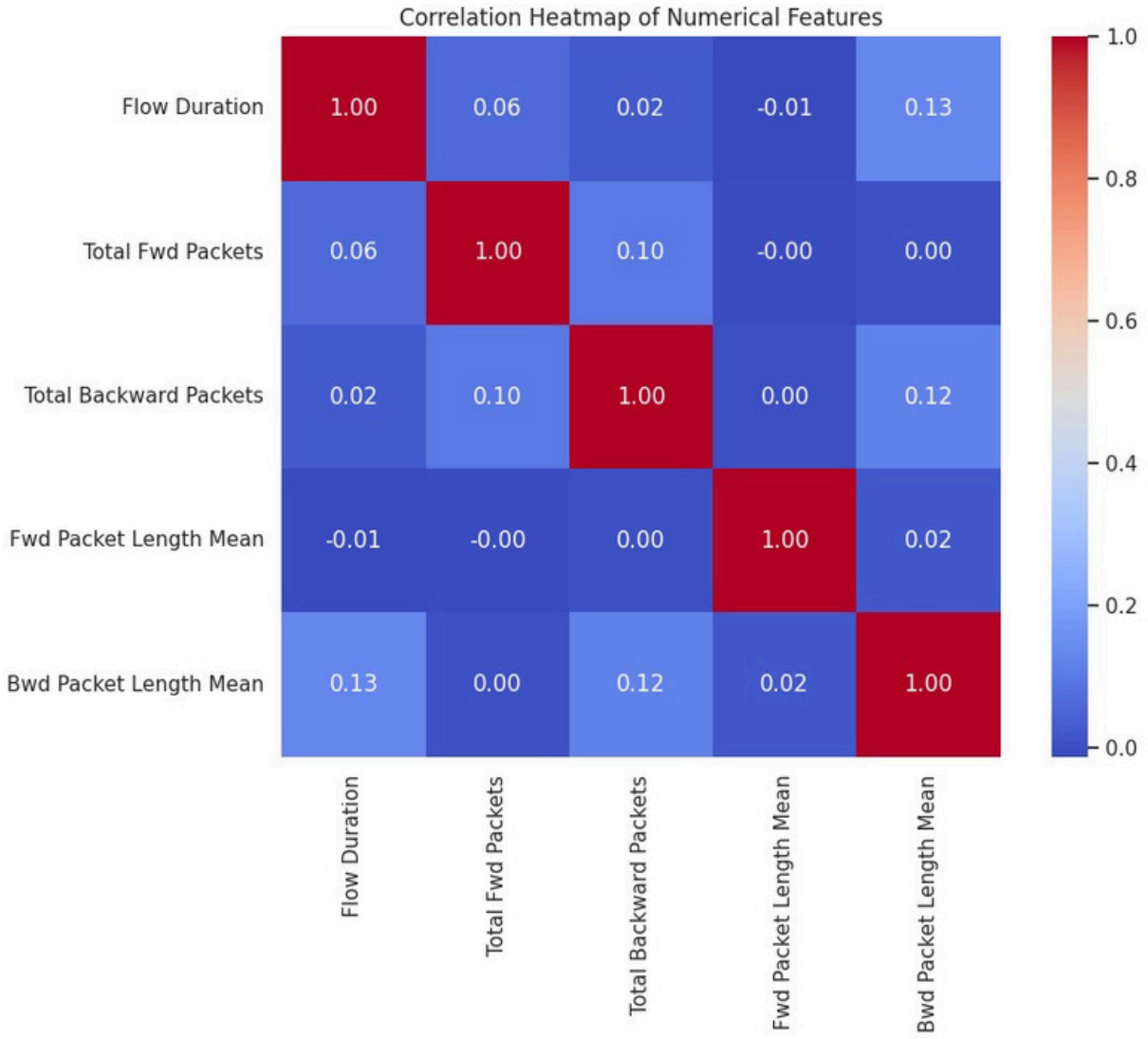

Distribution of Total Fwd Packets
1e9
1.0
0.8
0.6
0.4
0.2
0.0
Frequency
0
50000
100000
150000
200000
250000
300000
Total Fwd Packets


Distribution of Total Backward Packets
1e9
6
5
4
3
2
1
0
Frequency
0
50000
100000
150000
200000
250000
300000
Total Backward Packets





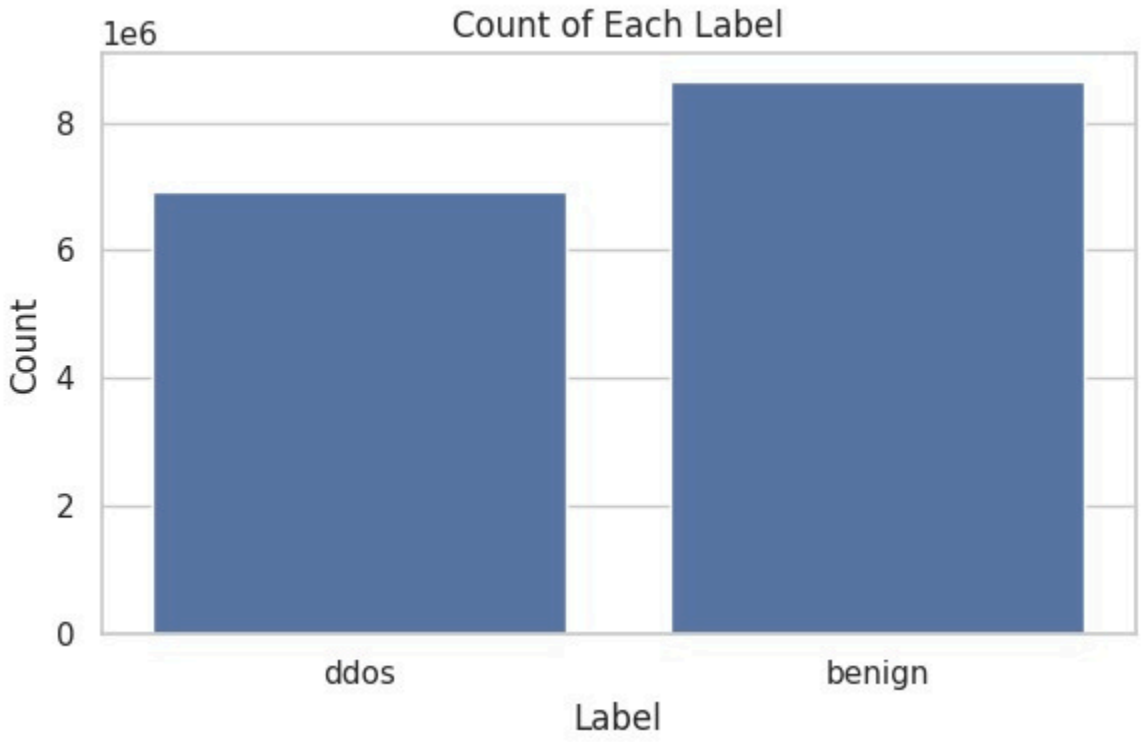
In [59]:
```
plt.figure(figsize=(10, 8))
corr_matrix = df[num_cols].corr()
sns.heatmap(corr_matrix, annot=True, cmap='coolwarm', fmt=".2f", square=True)
plt.title('Correlation Heatmap of Numerical Features')
plt.tight_layout()
plt.show()
```



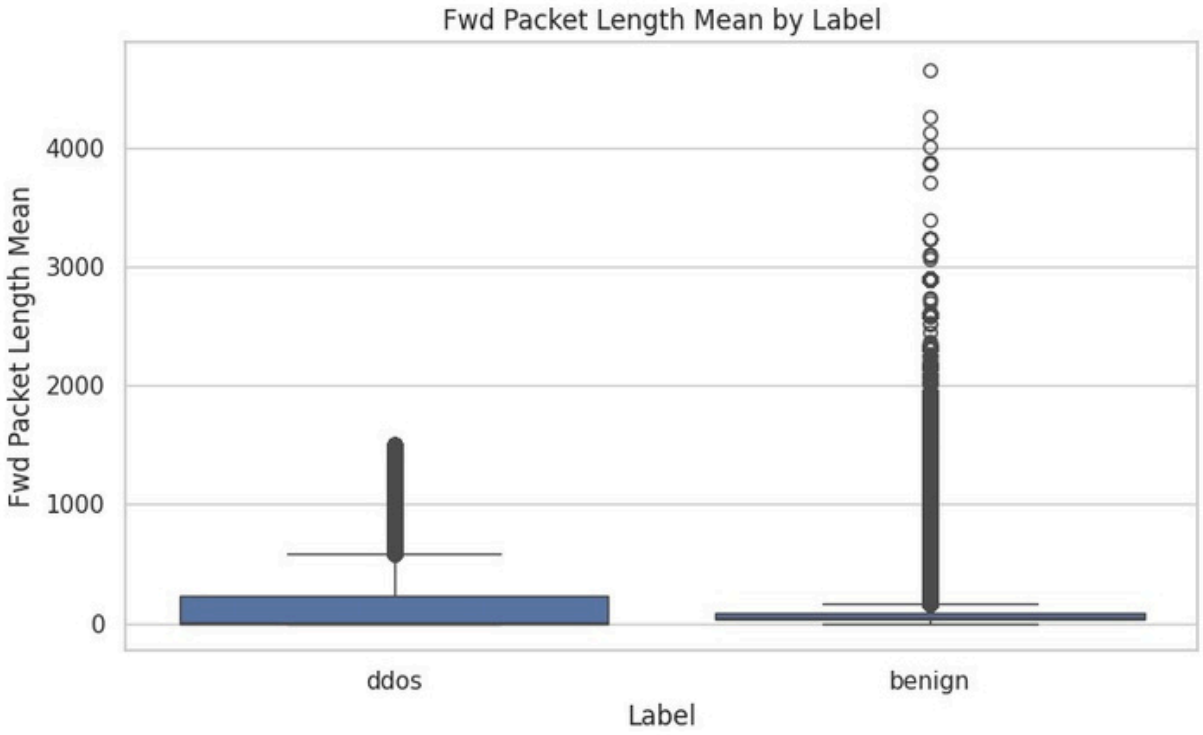
In [60]:
```
print(sns.__version__)
```

```
0.13.2
```

In [61]:
```
# Plot 3: Count plot for 'Label'
plt.figure(figsize=(6, 4))
sns.countplot(x='Label', data=df)
plt.title('Count of Each Label')
plt.xlabel('Label')
plt.ylabel('Count')
plt.tight_layout()
plt.show()
```



In [62]:
```
# Plot 4: Boxplot of packet lengths by label
plt.figure(figsize=(8, 5))
sns.boxplot(x='Label', y='Fwd Packet Length Mean', data=df)
plt.title('Fwd Packet Length Mean by Label')
plt.tight_layout()
plt.show()
```



Data Engineering

In [63]:
```
# Check for missing values in the dataset
missing_values = df.isnull().sum()


# Display summary statistics
summary_stats = df.describe()

# Display results
missing_values, summary_stats
```

Out[63]:
```
(Source IP                0
 Source Port              0
 Destination IP           0
 Destination Port         0
 Protocol                 0
 Flow Duration            0
 Total Fwd Packets        0
 Total Backward Packets   0
 Fwd Packet Length Mean   0
 Bwd Packet Length Mean   0
 Label                    0
 dtype: int64,
        Flow Duration  Total Fwd Packets  Total Backward Packets \
 count   1.556042e+07       1.556042e+07            1.556042e+07
mean     6.994334e+06       2.786705e+01            1.113106e+01
std      2.301101e+07       1.560498e+03            2.331984e+02
min     -1.000000e+00       0.000000e+00            0.000000e+00
25%      3.840000e+02       1.000000e+00            1.000000e+00
50%      1.216100e+04       2.000000e+00            2.000000e+00
75%      2.335781e+06       4.000000e+00            5.000000e+00
max      1.200000e+08       3.096280e+05            2.919230e+05

        Fwd Packet Length Mean  Bwd Packet Length Mean
 count            1.556042e+07            1.556042e+07
 mean             1.336425e+02            8.534740e+01
 std              2.571392e+02            1.232792e+02
 min              0.000000e+00            0.000000e+00
 25%              0.000000e+00            0.000000e+00
 50%              7.800000e+01            1.584615e+01
 75%              1.270000e+02            1.320000e+02
 max              3.037508e+04            4.660441e+03)
```

In [64]:
```
# Display column names to identify discrepancies
df.columns
```

Out[64]:
```
Index(['Source IP', 'Source Port', 'Destination IP', 'Destination Port',
       'Protocol', 'Flow Duration', 'Total Fwd Packets',
       'Total Backward Packets', 'Fwd Packet Length Mean',
       'Bwd Packet Length Mean', 'Label'],
      dtype='object')
```

In [65]:
```
# Adjust the list of essential features based on the available columns
available_features = [
```

```
    'Flow Duration', 'Source Port', 'Destination Port', 'Protocol',
    'Total Fwd Packets', 'Fwd Packet Length Mean', 'Bwd Packet Length Mean',
    'Label'  # Keeping the target column
]

# Ensure only the required columns are retained
df_cleaned = df[available_features].copy()

# Display the cleaned dataset using print instead of a tool function
print("Cleaned Dataset:")
print(df_cleaned.head())
```

```
Cleaned Dataset:
   Flow Duration Source Port Destination Port Protocol Total Fwd Packets \
0      3974862.0        4504                80      tcp                 29
1           63.0        4504                80      tcp                  1
2       476078.0        4505                80      tcp                  2
3          151.0        4505                80      tcp                  2
4       472507.0        4506                80      tcp                  2

     Fwd Packet Length Mean Bwd Packet Length Mean Label
0                  2.965517              1359.340909 ddos
1                  0.000000                 0.000000 ddos
2                 43.000000               506.166667 ddos
3                  0.000000                 0.000000 ddos
4                 36.500000               210.000000 ddos
```

In [66]:
```
# Define num_cols (Numeric Columns Only)
num_cols = df_cleaned.select_dtypes(include=['float64', 'int64']).columns

# Step 1: Remove Duplicates (Only for Numeric Columns)
df_cleaned = df_cleaned.drop_duplicates(subset=num_cols)
```

In [67]:
```
print("Dropped duplicate rows based on numeric columns.")
```

```
Dropped duplicate rows based on numeric columns.
```

In [68]:
```
# Handle missing values (column-wise to prevent shape errors)
for col in num_cols:
    df_cleaned[col] = df_cleaned[col].fillna(df_cleaned[col].median())
```

In [69]:
```
# Remove highly correlated features (> 0.95) - Only for numeric columns
df_numeric = df_cleaned[num_cols]  # Ensure it's numeric only
cor_matrix = df_numeric.corr().abs()
upper_tri = np.triu(np.ones(cor_matrix.shape), k=1)

# Identify columns to drop based on correlation threshold
to_drop = [col for col in cor_matrix.columns if (cor_matrix[col].values * upper_tri
```

In [70]:
```
# **Fix:** Ensure that columns exist before dropping
essential_features = ['Flow Duration', 'Source Port', 'Destination Port', 'Protocol
                      'Total Fwd Packets', 'Fwd Packet Length Mean', 'Bwd Packet Le
                     ]

to_drop = [col for col in to_drop if col in df_cleaned.columns and col not in essen
```

```
df_cleaned.drop(columns=to_drop, inplace=True, errors='ignore')  # Ignore missing c
print("Dropped correlated features:", to_drop)
```

```
Dropped correlated features: []
```

In [71]:
```
# Display Final Data
print("Cleaned Network Traffic Data:")
print(df_cleaned.head())
```

```
Cleaned Network Traffic Data:
   Flow Duration Source Port Destination Port Protocol Total Fwd Packets \
0      3974862.0        4504                80      tcp                  29
1           63.0        4504                80      tcp                   1
2       476078.0        4505                80      tcp                   2
3          151.0        4505                80      tcp                   2
4       472507.0        4506                80      tcp                   2

     Fwd Packet Length Mean Bwd Packet Length Mean Label
0                  2.965517               1359.340909 ddos
1                  0.000000                  0.000000 ddos
2                 43.000000                506.166667 ddos
3                  0.000000                  0.000000 ddos
4                 36.500000                210.000000 ddos
```

In [72]:
```
print("Available columns in df_cleaned:", df_cleaned.columns)
```

```
Available columns in df_cleaned: Index(['Flow Duration', 'Source Port', 'Destination
Port', 'Protocol',
       'Total Fwd Packets', 'Fwd Packet Length Mean', 'Bwd Packet Length Mean',
       'Label'],
      dtype='object')
```

In [73]:
```
from sklearn.preprocessing import LabelEncoder, StandardScaler

# Ensure df_cleaned is the correct dataset
df_encoded = df_cleaned.copy()

# Step 1: Encode categorical variables (Label and Protocol)
label_encoder = LabelEncoder()
df_encoded["Protocol"] = label_encoder.fit_transform(df_encoded["Protocol"])
df_encoded["Label"] = label_encoder.fit_transform(df_encoded["Label"])

# Step 2: Convert all columns to numeric safely
df_encoded = df_encoded.apply(pd.to_numeric, errors='coerce')

# Step 3: Drop rows with NaN values (from encoding or invalid data)
df_encoded = df_encoded.dropna()

# Step 4: Standardize features (excluding Label)
scaler = StandardScaler()
X_scaled = scaler.fit_transform(df_encoded.drop(columns=["Label"]))
y = df_encoded["Label"].astype(int)  # Ensure Label is integer

# Step 5: Convert back to DataFrame
df_scaled = pd.DataFrame(X_scaled, columns=df_encoded.drop(columns=["Label"]).colum
df_scaled["Label"] = y
```

```
print("Successfully cleaned, encoded, and scaled data!")
print(df_scaled.head())
```

```
Successfully cleaned, encoded, and scaled data!
   Flow Duration Source Port Destination Port Protocol Total Fwd Packets \
0      -0.282271    -1.141231          -0.74716 0.009922           -0.009472
1      -0.418600    -1.141231          -0.74716 0.009922           -0.022637
2      -0.402273    -1.141192          -0.74716 0.009922           -0.022167
3      -0.418597    -1.141192          -0.74716 0.009922           -0.022167
4      -0.402396    -1.141153          -0.74716 0.009922           -0.022167

      Fwd Packet Length Mean Bwd Packet Length Mean Label
0                -0.835627                 3.623101     1.0
1                -0.856321                -0.656087     1.0
2                -0.556262                 0.937319     1.0
3                -0.856321                -0.656087     1.0
4                -0.601619                 0.004991     1.0
```

In [74]:
```
# Check for missing values in the Label column
df_scaled["Label"].isnull().sum()
```

Out[74]: 3897081

In [75]:
```
# Check for NaN and NA values in df_scaled

nan_summary = df_scaled.isna().sum()   # Count NaN values per column
na_summary = df_scaled.isnull().sum()   # Count NA values per column

# Display only columns that have missing values
missing_data = pd.DataFrame({
    "NaN Count": nan_summary[nan_summary > 0],
    "NA Count": na_summary[na_summary > 0]
})
```

In [76]:
```
!pip install imbalanced-learn
```

```
Requirement already satisfied: imbalanced-learn in /opt/conda/lib/python3.11/site-pa
ckages (0.13.0)
Requirement already satisfied: numpy<3,>=1.24.3 in /opt/conda/lib/python3.11/site-pa
ckages (from imbalanced-learn) (1.26.4)
Requirement already satisfied: scipy<2,>=1.10.1 in /opt/conda/lib/python3.11/site-pa
ckages (from imbalanced-learn) (1.15.2)
Requirement already satisfied: scikit-learn<2,>=1.3.2 in /opt/conda/lib/python3.11/s
ite-packages (from imbalanced-learn) (1.5.2)
Requirement already satisfied: sklearn-compat<1,>=0.1 in /opt/conda/lib/python3.11/s
ite-packages (from imbalanced-learn) (0.1.3)
Requirement already satisfied: joblib<2,>=1.1.1 in /opt/conda/lib/python3.11/site-pa
ckages (from imbalanced-learn) (1.4.2)
Requirement already satisfied: threadpoolctl<4,>=2.0.0 in /opt/conda/lib/python3.11/
site-packages (from imbalanced-learn) (3.5.0)
```

In [77]:
```
# Import IsolationForest
from sklearn.ensemble import IsolationForest
from sklearn.metrics import classification_report
import numpy as np
```

```
# Drop rows where Label is NaN
df_scaled = df_scaled.dropna(subset=["Label"])

# Define features and labels
X = df_scaled.drop(columns=["Label"])
y = df_scaled["Label"].astype(int)  # Ensure integer labels for classification_repo

# Train the model
iso_forest = IsolationForest(n_estimators=100, contamination=0.1, random_state=42)
iso_forest.fit(X)

# Predict anomalies
preds = iso_forest.predict(X)
preds = np.where(preds == -1, 1, 0)  # Convert to 1 (anomaly), 0 (normal)

# Print classification report
print("Isolation Forest Classification Report (after cleaning NaNs):\n")
print(classification_report(y, preds))
```

```
Isolation Forest Classification Report (after cleaning NaNs):

              precision    recall  f1-score   support

           0       0.23      0.75      0.35   1213124
           1       0.32      0.04      0.08   3264199

    accuracy                           0.23   4477323
   macro avg       0.27      0.40      0.21   4477323
weighted avg       0.29      0.23      0.15   4477323
```

In [78]:
```
# Add new engineered features to df_cleaned  df_cleaned["Flow Bytes/s"]  =
(df_cleaned["Fwd Packet Length Mean"] + df_cleaned["Bw df_cleaned["Flow Packets/s"]
= df_cleaned["Total Fwd Packets"] / (df_cleaned["Flow df_cleaned["Asymmetry Ratio"]
= (
    df_cleaned["Fwd Packet Length Mean"] - df_cleaned["Bwd Packet Length Mean"]
) / (df_cleaned["Fwd Packet Length Mean"] + df_cleaned["Bwd Packet Length Mean"] +

# Show the updated DataFrame with new features
df_cleaned[["Flow Bytes/s", "Flow Packets/s", "Asymmetry Ratio"]].head()
```

Out[78]:

	Flow Bytes/s	Flow Packets/s	Asymmetry Ratio
0	0.000343	0.000007	-0.995646
1	0.000000	0.015873	0.000000
2	0.001154	0.000004	-0.843399
3	0.000000	0.013245	0.000000
4	0.000522	0.000004	-0.703854

Data Modeling

```
df_cleaned["Flow Bytes/s"] = (df_cleaned["Fwd Packet Length Mean"] +␣
  ↪df_cleaned["BwdPacketLengthMean"])/(df_cleaned["FlowDuration"]+1e-6)
df_cleaned["FlowPackets/s"]=df_cleaned["TotalFwdPackets"]/␣
  ↪(df_cleaned["FlowDuration"]+1e-6)
df_cleaned["AsymmetryRatio"]=(
    df_cleaned["Fwd Packet Length Mean"] - df_cleaned["Bwd Packet Length Mean"]
) / (df_cleaned["Fwd Packet Length Mean"] + df_cleaned["Bwd Packet Length␣
  ↪Mean"] + 1e-6)

# Show the updated DataFrame with new features
df_cleaned[["Flow Bytes/s", "Flow Packets/s", "Asymmetry Ratio"]].head()
```

[78]:

	FlowBytes/s	FlowPackets/s	AsymmetryRatio
0	0.000343	0.000007	-0.995646
1	0.000000	0.015873	0.000000
2	0.001154	0.000004	-0.843399
3	0.000000	0.013245	0.000000
4	0.000522	0.000004	-0.703854

1.8 DataModeling

[79]:
```
!aws s3 cp s3://msads-508-sp25-team6/Test\ Data.csv .
```

```
download: s3://msads-508-sp25-team6/Test Data.csv to ./Test Data.csv
```

[80]:
```
!aws s3 cp s3://msads-508-sp25-team6/Train\ Data.csv .
```

```
download: s3://msads-508-sp25-team6/Train Data.csv to ./Train Data.csv
```

[81]:
```
!aws s3 cp s3://msads-508-sp25-team6/Validation\ Data.csv .
```

```
download: s3://msads-508-sp25-team6/Validation Data.csv to ./Validation Data.csv
```

[3]:
```
train_df = pd.read_csv("Train Data.csv")
```

[4]:
```
test_df = pd.read_csv("Test Data.csv")
```

[6]:
```
validation_df = pd.read_csv("Validation Data.csv")
```

[7]:
```
# Display the first few rows of each dataset
print("Train Data:")
display(train_df.head())

print("\nTest Data:")
display(test_df.head())
print("\nValidation Data:")
display(validation_df.head())
```

Train Data:

	SourcePort	DestinationPort	Protocol	FlowDuration	TotalFwdPackets	\
0	443	52721	0	-0.303953	-0.016576	
1	80	41498	0	-0.123066	-0.015295	
2	50750	443	0	-0.288460	-0.013372	
3	53902	80	0	-0.303118	-0.017217	
4	60419	80	0	-0.303878	-0.017217	

	TotalBackwardPackets	FwdPacketLengthMean	BwdPacketLengthMean	\
0	-0.047732	-0.692310	-0.519728	
1	-0.030579	1.203792	-0.201807	
2	-0.026291	-0.283250	2.050864	
3	-0.043444	-0.692310	-0.519728	
4	-0.043444	-0.692310	-0.519728	

	Label	Source Octet 1	Source Octet 2	Source Octet 3	Source Octet 4	\
0	1	108	161	188	192	
1	0	172	31	69	25	
2	1	172	31	66	53	
3	0	18	219	193	20	
4	0	18	218	115	60	

	Destination Octet 1	Destination Octet 2	Destination Octet 3	\
0	172	31	64	
1	18	219	193	
2	92	223	231	
3	172	31	69	
4	172	31	69	

	Destination Octet 4
0	45
1	20
2	190
3	25
4	28

Test Data:

	Source Port	Destination Port	Protocol	Flow Duration	Total Fwd Packets	\
0	80	54802	0	-0.303900	-0.015295	
1	52194	3389	0	-0.205084	-0.012731	
2	62310	80	0	0.747266	-0.016576	
3	57775	3389	0	-0.213707	-0.012731	
4	21742	3389	0	0.504680	-0.012731	

	TotalBackwardPackets	FwdPacketLengthMean	BwdPacketLengthMean	\
0	-0.034868	1.203792	-0.142501	

1	-0.017715	0.374374	0.274172	
2	-0.047732	-0.692310	-0.519728	
3	-0.017715	0.471714	0.358617	
4	-0.013427	0.471714	0.248824	

	Label	Source Octet 1	Source Octet 2	Source Octet 3	Source Octet 4 \
0	0	172	31	69	28
1	1	211	170	54	2
2	0	18	219	5	43
3	1	96	32	246	226
4	1	200	32	82	227

	Destination Octet 1	Destination Octet 2	Destination Octet 3 \
0	18	219	5
1	172	31	66
2	172	31	69
3	172	31	64
4	172	31	69

	Destination Octet 4
0	43
1	92
2	25
3	95
4	9

Validation Data:

	Source Port	Destination Port	Flow Duration	Total Fwd Packets\
0	80	0	59310	-0.109258	-0.015295
1	59602	0	80	-0.303896	-0.017217
2	54344	1	53	-0.303901	-0.017217
3	18899	0	34122	-0.303954	0.021873
4	80	0	50098	2.515912	-0.015295

	Total Backward Packets	Fwd Packet Length Mean	Bwd Packet Length Mean \
0	-0.030579	1.203792	-0.131806
1	-0.043444	-0.692310	-0.519728
2	-0.043444	-0.351620	-0.123056
3	0.218136	-0.205610	2.307534
4	-0.026291	1.203792	-0.231169

	Label	Source Octet 1	Source Octet 2	Source Octet 3	Source Octet 4 \
0	0	172	31	69	25
1	0	18	219	193	20
2	1	172	31	69	11
3	1	59	166	0	5
4	0	172	31	69	25

	Destination Octet 1	Destination Octet 2	Destination Octet 3	Destination Octet 4
0	18	219	193	20
1	172	31	69	25
2	172	31	0	2
3	149	171	126	6
4	18	219	193	20

[8]:
```
# Identify and drop IP address octet columns
ip_octets = [col for col in train_df.columns if "Octet" in col]
# Separate features and target labels
X_train = train_df.drop(columns=ip_octets + ["Label"])
y_train = train_df["Label"]

X_test = test_df.drop(columns=ip_octets + ["Label"])
y_test = test_df["Label"]
X_val = validation_df.drop(columns=ip_octets + ["Label"])
y_val = validation_df["Label"]
```

[9]:
```
import numpy as np
from sklearn.impute import SimpleImputer
from sklearn.ensemble import IsolationForest
from sklearn.metrics import classification_report, confusion_matrix
from sklearn.preprocessing import StandardScaler
from keras.models import Sequential
```

```
c:\Users\barro\anaconda3\Lib\site-packages\h5py\__init__.py:36: UserWarning: h5py is running against HDF5 1.14.6 when it was built against 1.14.5, this may cause problems
  _warn(("h5py is running against HDF5 {0} when it was built against {1}, "
```

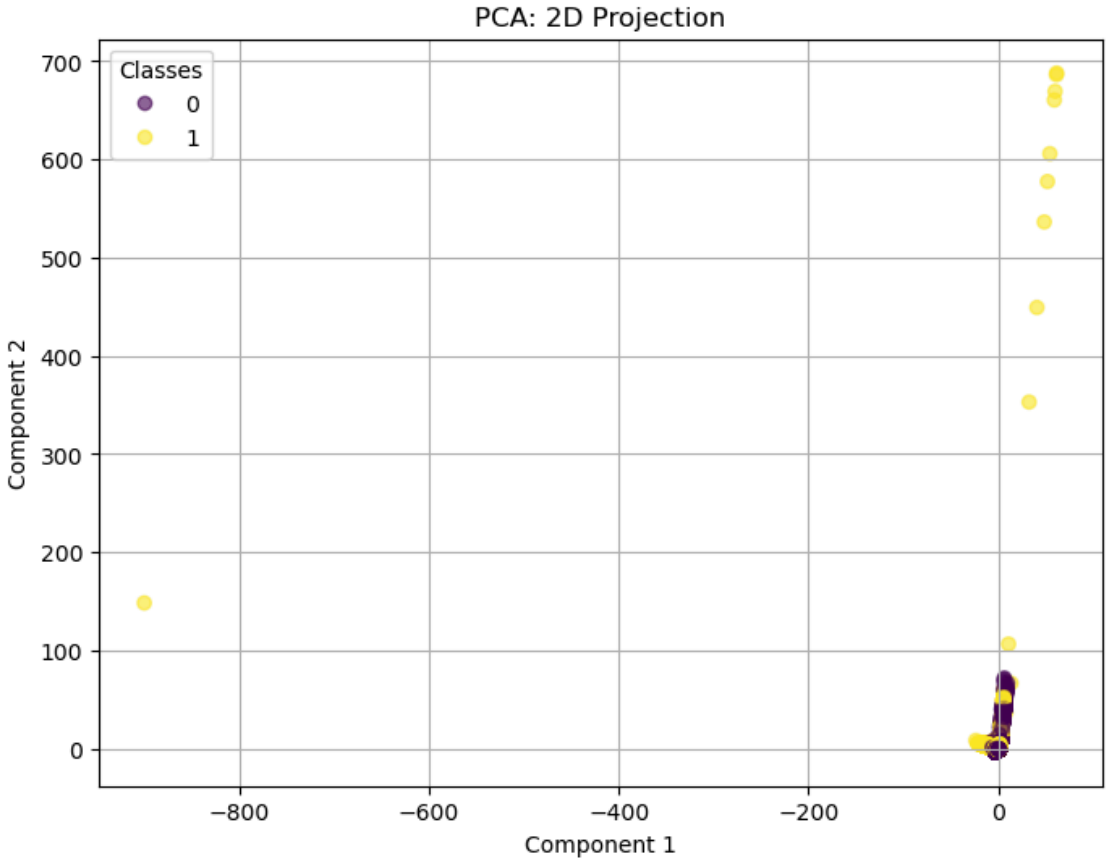
[10]:
```
# Handle missing values using mean imputation
imputer = SimpleImputer(strategy="mean")
X_train_imputed = imputer.fit_transform(X_train)
X_test_imputed = imputer.transform(X_test)
X_val_imputed = imputer.transform(X_val)

# Remove entries in test set with missing labels
valid_indices = ~y_test.isna()
```

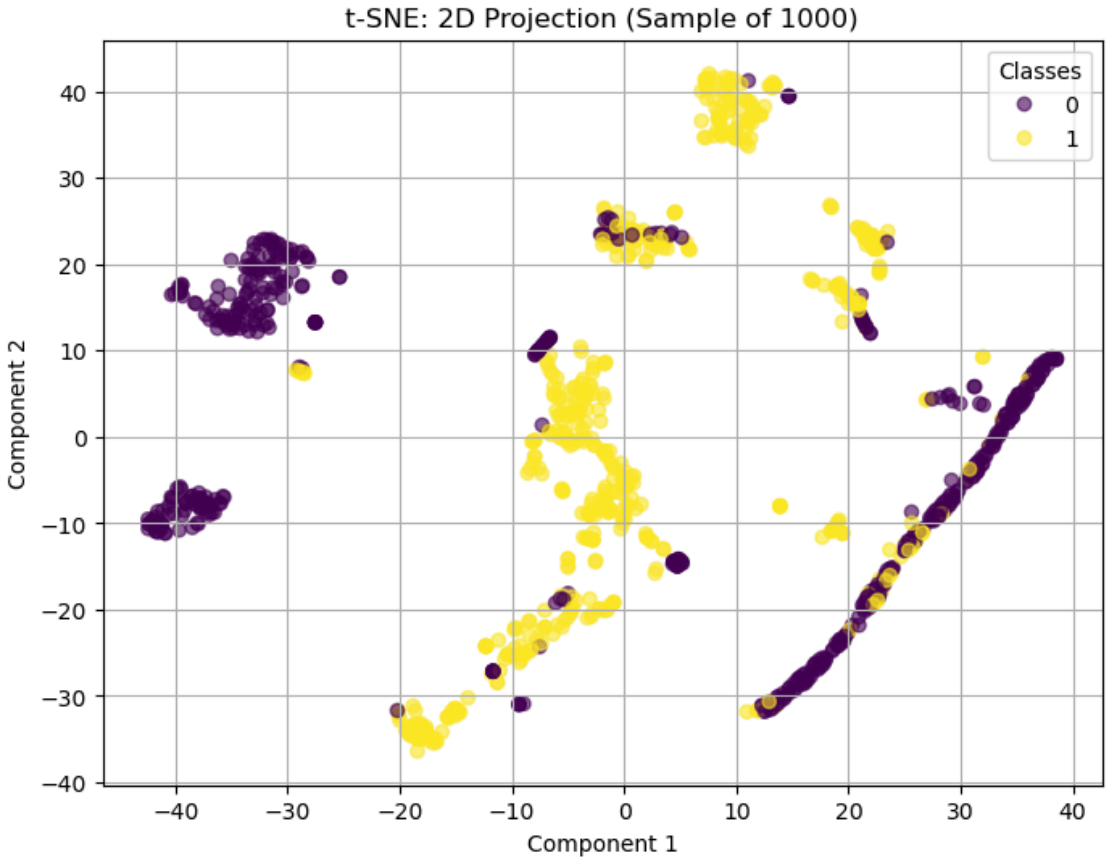
```
X_test_clean = X_test_imputed[valid_indices]
y_test_clean = y_test[valid_indices].astype(int)    # Ensure integer labels
```

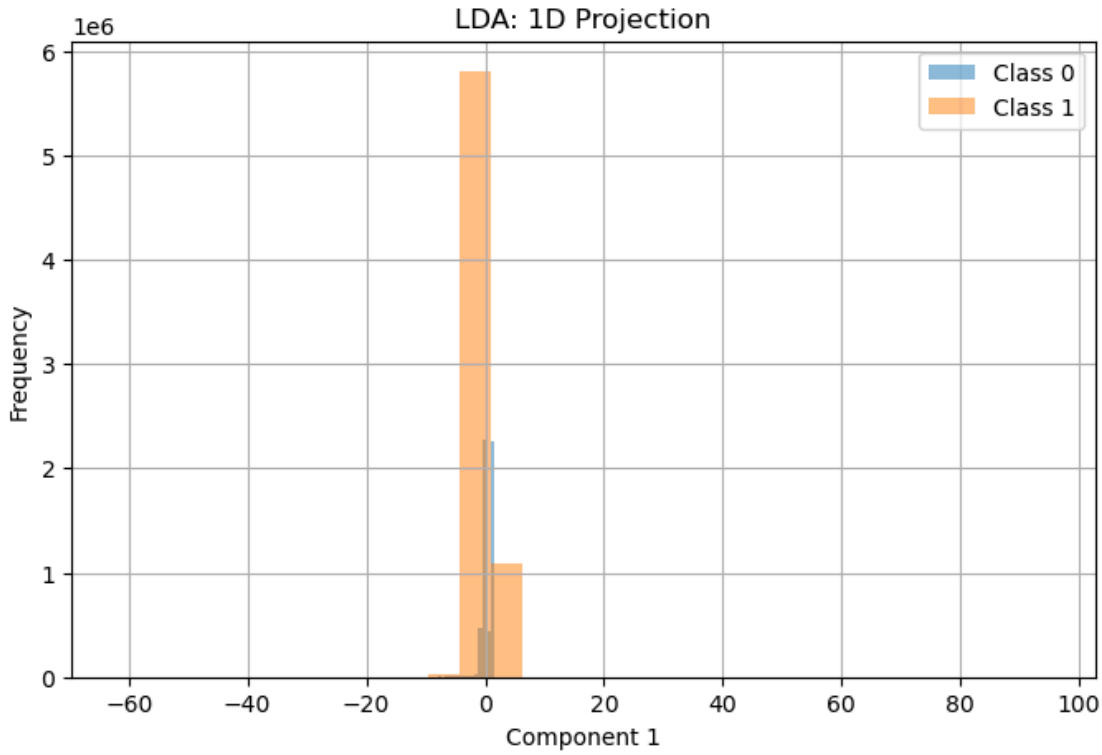
[11]:
```
from sklearn.ensemble import IsolationForest
from sklearn.metrics import classification_report, confusion_matrix
from sklearn.preprocessing import StandardScaler
from sklearn.decomposition import PCA
from sklearn.discriminant_analysis import LinearDiscriminantAnalysis as LDA
from sklearn.manifold import TSNE
import matplotlib.pyplot as plt


# Scale the data
scaler = StandardScaler()
X_train_scaled = scaler.fit_transform(X_train_imputed)
X_test_scaled = scaler.transform(X_test_imputed)
#PCA
pca = PCA(n_components=2)
X_pca = pca.fit_transform(X_train_scaled)
# LDA (requires class labels)
lda = LDA(n_components=1)
X_lda = lda.fit_transform(X_train_scaled, y_train)
# t-SNE (use subset for faster processing if dataset is large)
tsne = TSNE(n_components=2, random_state=42, perplexity=30, n_iter=1000)
X_tsne = tsne.fit_transform(X_train_scaled[:1000])
y_tsne = y_train[:1000]
# Plot function for 2D
def plot_2D(X, y, title):


    plt.figure(figsize=(8, 6)) scatter = plt.scatter(X[:, 0], X[:, 1], c=y,
    cmap='viridis', alpha=0.6) plt.title(title) plt.xlabel("Component 1")
    plt.ylabel("Component 2") plt.legend(*scatter.legend_elements(),
    title="Classes") plt.grid(True) plt.show()




# Visualizations
plot_2D(X_pca, y_train, "PCA: 2D Projection")
plot_2D(X_tsne, y_tsne, "t-SNE: 2D Projection (Sample of 1000)")
# LDA is 1D → plot as a histogram instead
plt.figure(figsize=(8, 5))
for label in sorted(y_train.unique()):
```

```
    plt.hist(X_lda[y_train == label], bins=30, alpha=0.5, label=f"Class␣
↪{label}")
plt.title("LDA:1DProjection")
plt.xlabel("Component 1")
plt.ylabel("Frequency")
plt.legend()
plt.grid(True)
plt.show()
```

c:\Users\barro\anaconda3\Lib\site-packages\sklearn\manifold_t_sne.py:1164: FutureWarning: 'n_iter' was renamed to 'max_iter' in version 1.5 and will be removed in 1.7.
  warnings.warn(
c:\Users\barro\anaconda3\Lib\site-packages\IPython\core\pylabtools.py:170: UserWarning: Creating legend with loc="best" can be slow with large amounts of data.
  fig.canvas.print_figure(bytes_io, **kw)




t-SNE: 2D Projection (Sample of 1000)
Classes
0
1
Component 2
Component 1
40
30
20
10
0
−10
−20
−30
−40
−40
−30
−20
−10
0
10
20
30
40



[21]:
```
# Scale the data
scaler = StandardScaler()
X_train_scaled = scaler.fit_transform(X_train_imputed)
X_test_scaled = scaler.transform(X_test_imputed)
```

[22]:
```
# PCA
pca = PCA(n_components=2)
X_pca = pca.fit_transform(X_train_scaled)
```

[23]:
```
# LDA (requires class labels)
lda = LDA(n_components=1)
X_lda = lda.fit_transform(X_train_scaled, y_train)
```

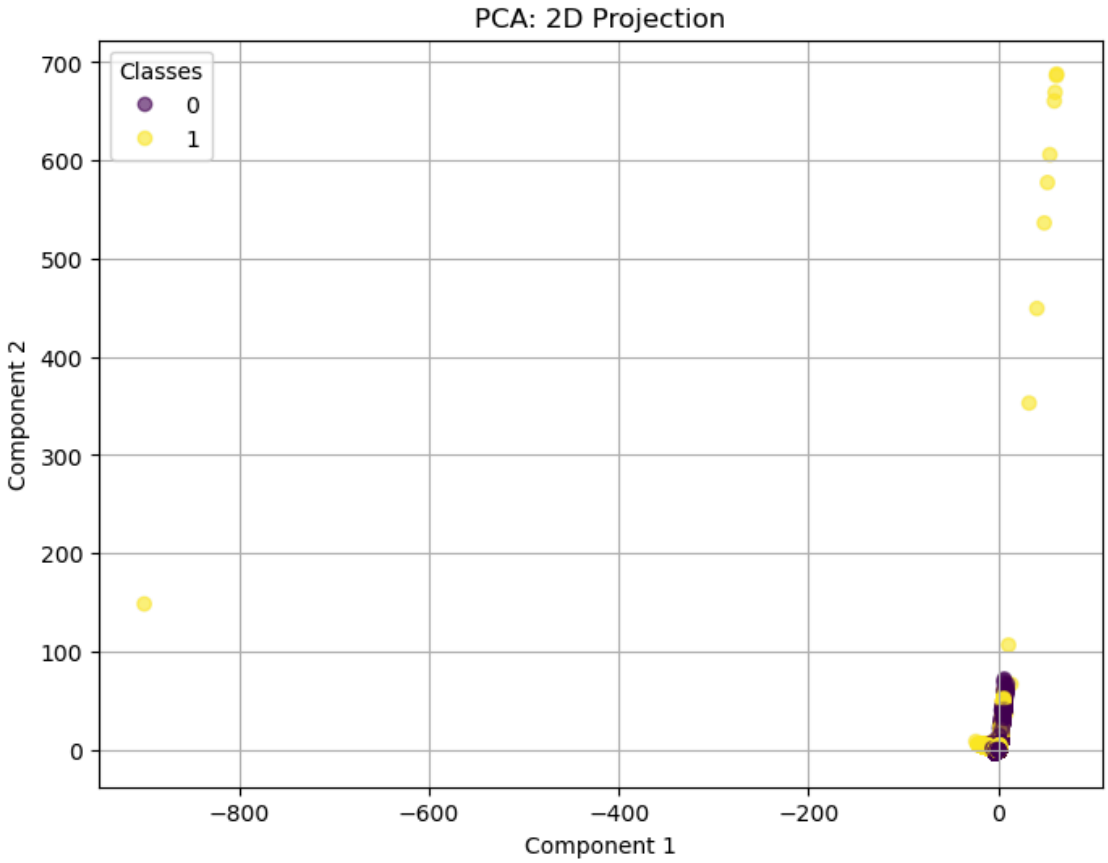
[24]:
```
# t-SNE (use subset for faster processing if dataset is large)
tsne = TSNE(n_components=2, random_state=42, perplexity=30, n_iter=1000)
X_tsne = tsne.fit_transform(X_train_scaled[:1000])
y_tsne = y_train[:1000]
```

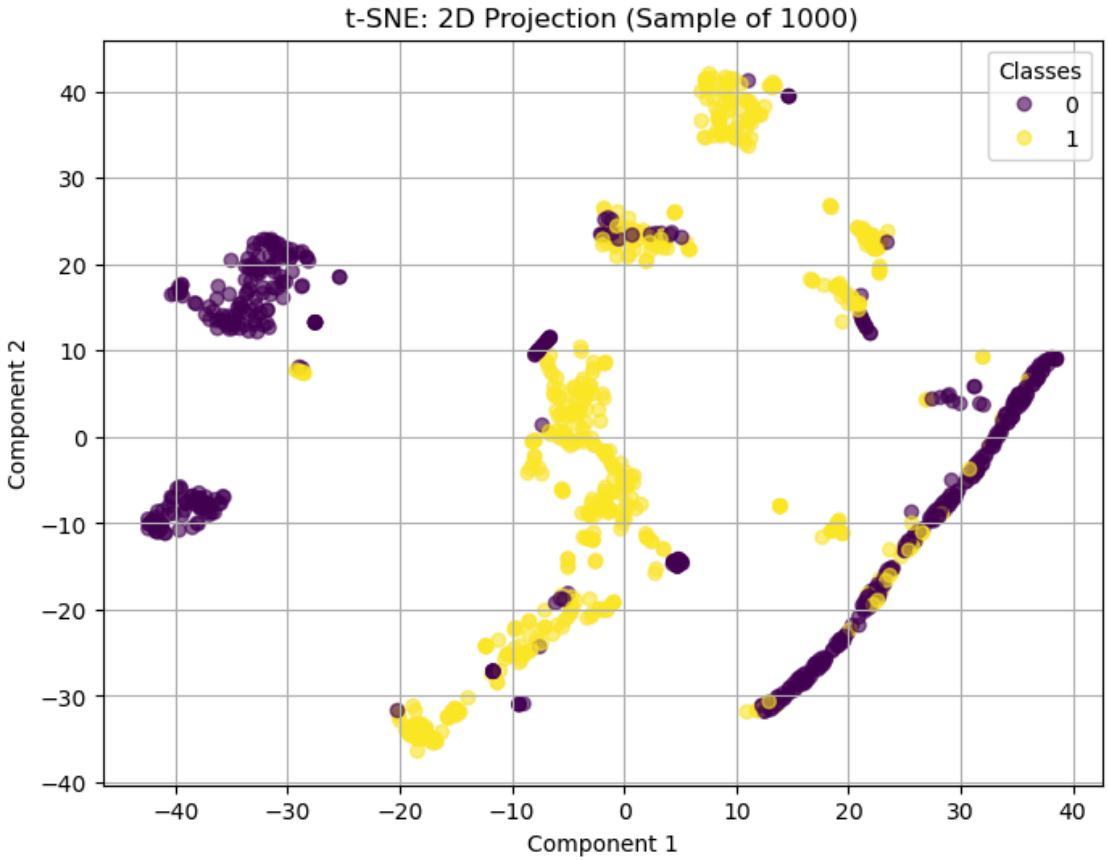
```
c:\Users\barro\anaconda3\Lib\site-packages\sklearn\manifold\_t_sne.py:1164:
FutureWarning: 'n_iter' was renamed to 'max_iter' in version 1.5 and will be
removed in 1.7.
  warnings.warn(
```

[25]:
```
# Plot function for 2D
def plot_2D(X, y, title):
    plt.figure(figsize=(8, 6)) scatter = plt.scatter(X[:, 0], X[:, 1], c=y, cmap='viridis',
    alpha=0.6) plt.title(title) plt.xlabel("Component 1") plt.ylabel("Component 2")
    plt.legend(*scatter.legend_elements(), title="Classes") plt.grid(True) plt.show()
```

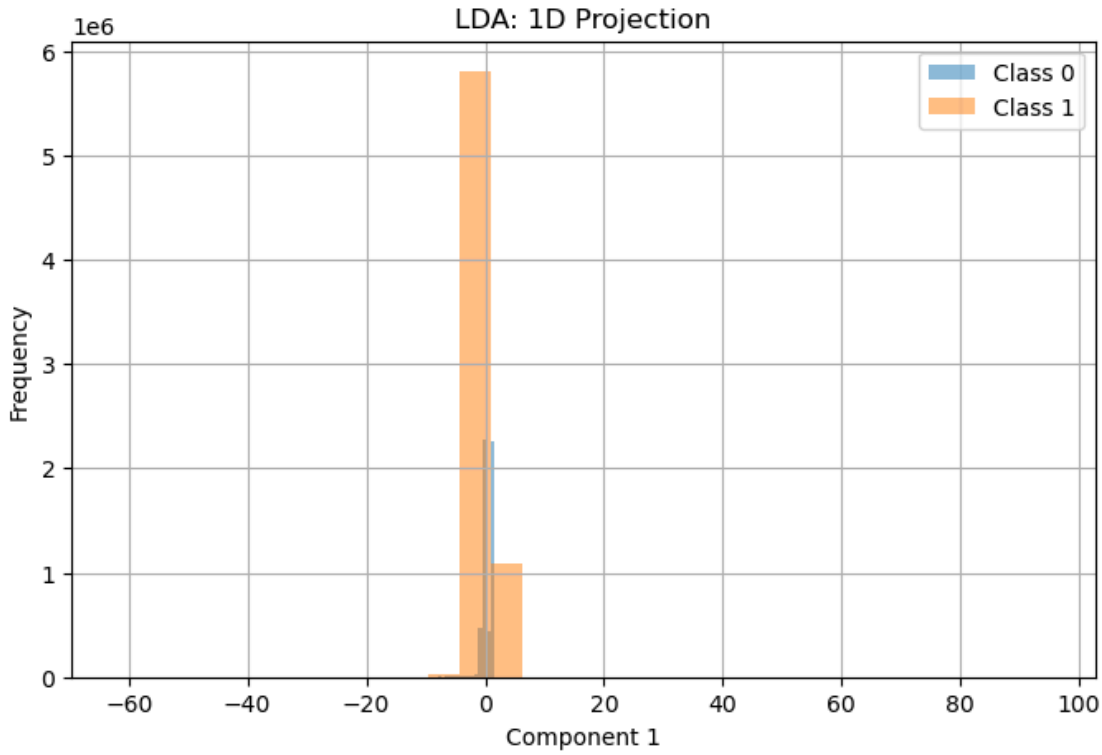
[26]:
```
# Visualizations
plot_2D(X_pca, y_train, "PCA: 2D Projection")
plot_2D(X_tsne, y_tsne, "t-SNE: 2D Projection (Sample of 1000)")
```

```
c:\Users\barro\anaconda3\Lib\site-packages\IPython\core\pylabtools.py:170:
UserWarning: Creating legend with loc="best" can be slow with large amounts of
data.
  fig.canvas.print_figure(bytes_io, **kw)
```





[27]:
```
# LDA is 1D → plot as a histogram instead
plt.figure(figsize=(8, 5))
for label in sorted(y_train.unique()):
    plt.hist(X_lda[y_train == label], bins=30, alpha=0.5, label=f"Class␣
↪{label}")
plt.title("LDA:1DProjection")
plt.xlabel("Component 1")
plt.ylabel("Frequency")
plt.legend()
plt.grid(True)
plt.show()
```



[12]:
```
# Train RCF (using IsolationForest as a local alternative, similar in behavior)
rcf_model = IsolationForest(n_estimators=100, contamination=0.05,␣
  ↪random_state=42)
rcf_model.fit(X_train_scaled)

# Predict anomaly scores
y_pred_test = rcf_model.predict(X_test_scaled)
y_pred_test = [1 if pred == -1 else 0 for pred in y_pred_test] # Convert to␣
  ↪binary (1=anomaly, 0=normal)
# Evaluation
print("Classification Report:\n", classification_report(y_test, y_pred_test))
print("Confusion Matrix:\n", confusion_matrix(y_test, y_pred_test))
```

Classification Report:

	precision	recall	f1-score	support
0	0.46	0.99	0.63	692705
1	0.91	0.08	0.15	863337
accuracy			0.49	1556042
macroavg	0.69	0.54	0.39	1556042
weightedavg	0.71	0.49	0.36	1556042

Confusion Matrix:
[[685554 7151]
[792910 70427]]

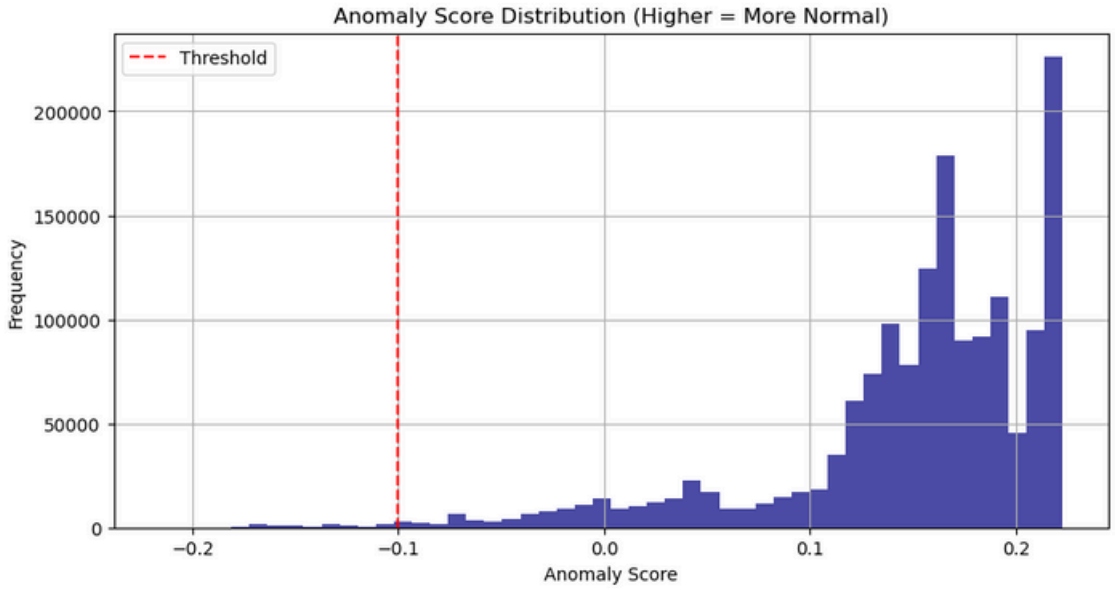
[13]:
```
# Get anomaly scores (the higher, the more "normal")
anomaly_scores = rcf_model.decision_function(X_test_scaled)
# Define a custom threshold
threshold = -0.1 # Adjust this based on your domain or precision-recall␣
↪tradeoff

y_pred_custom=[1ifscore<thresholdelse0forscoreinanomaly_scores]

# Compare with standard prediction
print("Custom Threshold Classification Report:\n",␣
↪classification_report(y_test,y_pred_custom))
print("CustomThresholdConfusionMatrix:\n",confusion_matrix(y_test,␣
↪y_pred_custom))

# Optional: visualize the distribution of anomaly scores
plt.figure(figsize=(10, 5))
plt.hist(anomaly_scores, bins=50, color='navy', alpha=0.7)
plt.axvline(threshold, color='red', linestyle='--', label='Threshold')
plt.title("Anomaly Score Distribution (Higher = More Normal)")
plt.xlabel("Anomaly Score")
plt.ylabel("Frequency")
plt.legend()
plt.grid(True)
plt.show()
```

Custom Threshold Classification Report:

	precision	recall	f1-score	support
0	0.45	1.00	0.62	692705
1	0.98	0.01	0.02	863337
accuracy			0.45	1556042
macroavg	0.72	0.51	0.32	1556042
weightedavg	0.75	0.45	0.29	

stom Threshold Confusion Matrix:

[[692531 174]
[852851 10486]]



[14]:
```
# Train an Isolation Forest model for anomaly detection
isolation_forest = IsolationForest(contamination=0.1, random_state=42)
isolation_forest.fit(X_train_imputed)

# Predict anomalies (1 = anomaly, 0 = normal)
y_pred_test = isolation_forest.predict(X_test_clean)
y_pred_test = np.where(y_pred_test == -1, 1, 0)

# Evaluate model performance
print("Classification Report:\n", classification_report(y_test_clean,␣
    y_pred_test))
print("\nConfusionMatrix:\n",confusion_matrix(y_test_clean,y_pred_test))
```

Classification Report:

	precision	recall	f1-score	support
0	0.48	0.97	0.64	692705
1	0.88	0.16	0.27	863337
accuracy			0.52	1556042
macroavg	0.68	0.57	0.46	1556042
weightedavg	0.70	0.52	0.44	

Confusion Matrix:
[[673873 18832]
[726696 136641]]

[15]:
```
!pip install xgboost
```

```
Requirement already satisfied: xgboost in c:\users\barro\anaconda3\lib\site-
packages (3.0.0)
Requirement already satisfied: numpy in c:\users\barro\anaconda3\lib\site-
packages (from xgboost) (1.26.4)
Requirement already satisfied: scipy in c:\users\barro\anaconda3\lib\site-
packages (from xgboost) (1.15.2)
```

[16]:
```
import pandas as pd
import numpy as np
from sklearn.impute import SimpleImputer
from sklearn.model_selection import train_test_split
from sklearn.metrics import classification_report, confusion_matrix,␣
    precision_score,recall_score,f1_score,roc_auc_score
fromxgboostimportXGBClassifier
```

[17]:
```
# Initialize XGBoost classifier
xgb_clf = XGBClassifier(
    n_estimators=100,
    max_depth=6,
    learning_rate=0.1,
    subsample=0.8,
    colsample_bytree=0.8,
    use_label_encoder=False,
    eval_metric="logloss",
    random_state=42
)

# Train the model
xgb_clf.fit(X_train_scaled, y_train)
# Predict on the clean test set
y_pred_xgb = xgb_clf.predict(X_test_clean)
# Evaluation
print("XGBoost Classification Report:\n", classification_report(y_test_clean,␣
    y_pred_xgb))

print("\nConfusionMatrix:\n",confusion_matrix(y_test_clean,y_pred_xgb))
```

```
c:\Users\barro\anaconda3\Lib\site-packages\xgboost\training.py:183: UserWarning:
[22:53:50] WARNING: C:\actions-runner\_work\xgboost\xgboost\src\learner.cc:738:
Parameters: { "use_label_encoder" } are not used.

  bst.update(dtrain, iteration=i, fobj=obj)
XGBoost Classification Report:
              precision    recall  f1-score   support
```

0	0.9	0.0	0.01	692705
1	0	0	0.71	863337
accuracy	0.5	1.00	0.5	1556042
macro avg	6		6	1556042
weighted avg	0.7	0.5	0.3	1556042
	3	0	6	
	0.71	0.5	0.4	
		6	0	

Confusion Matrix:
[[1885 690820]
[218863119]]

[18]:
```
from keras.models import Model
from keras.layers import Input, Dense
from keras.callbacks import EarlyStopping
from sklearn.preprocessing import StandardScaler
import matplotlib.pyplot as plt
```

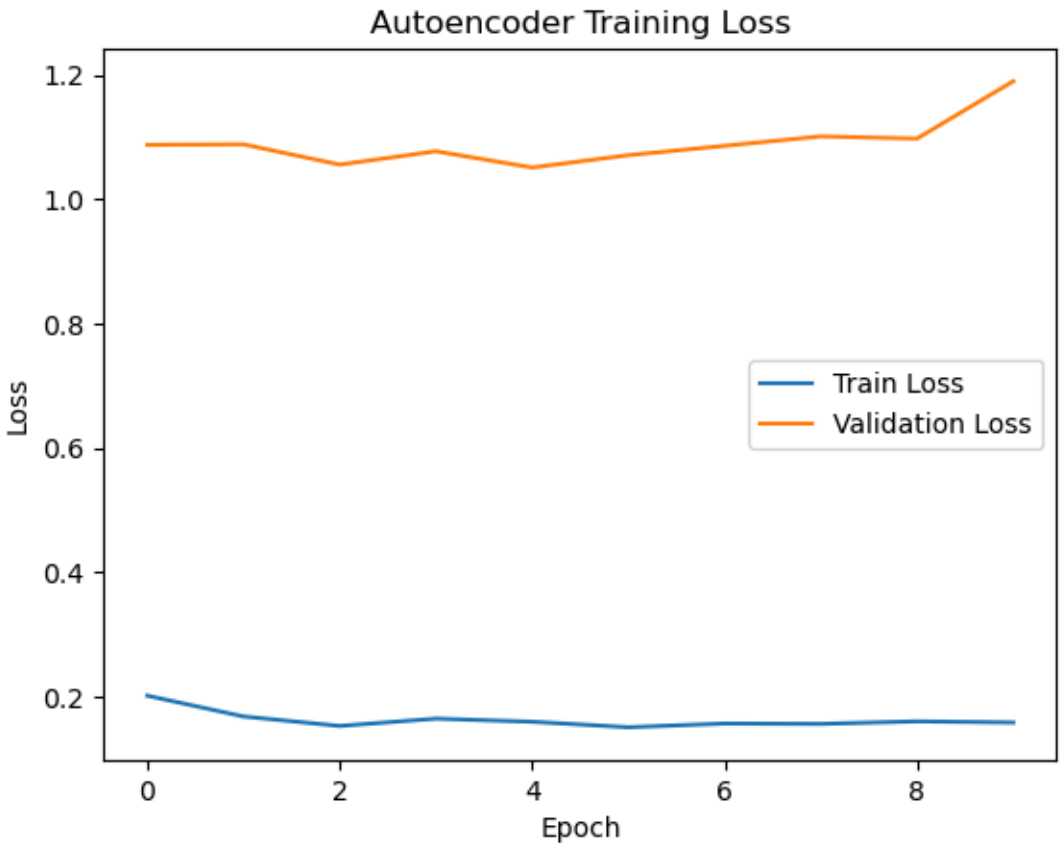
[19]:
```
from sklearn.preprocessing import StandardScaler
from tensorflow.keras.models import Model
from tensorflow.keras.layers import Input, Dense, Dropout
from tensorflow.keras.callbacks import EarlyStopping
from tensorflow.keras.optimizers import Adam

# Scale the imputed data
scaler = StandardScaler()
X_train_scaled = scaler.fit_transform(X_train_imputed)
X_val_scaled = scaler.transform(X_val_imputed)
X_test_scaled = scaler.transform(X_test_imputed)
# Define input dimension
input_dim = X_train_scaled.shape[1]
# Build the Autoencoder model
input_layer = Input(shape=(input_dim,))
encoded = Dense(32, activation='relu')(input_layer)
encoded = Dropout(0.2)(encoded) # Optional: helps prevent overfitting
encoded = Dense(16, activation='relu')(encoded)
bottleneck = Dense(8, activation='relu')(encoded)
decoded = Dense(16, activation='relu')(bottleneck)
decoded = Dropout(0.2)(decoded) # Optional: helps generalize better
decoded = Dense(32, activation='relu')(decoded)
output_layer = Dense(input_dim, activation='linear')(decoded)
# Define and compile model
autoencoder = Model(inputs=input_layer, outputs=output_layer)
optimizer = Adam(learning_rate=0.001)
```

```
autoencoder.compile(optimizer=optimizer, loss='mse')

# Set up early stopping
early_stop = EarlyStopping(
monitor='val_loss',
patience=5,
restore_best_weights=True,
verbose=1
)

# Train the model
history = autoencoder.fit(
    X_train_scaled, X_train_scaled,
    epochs=50,
    batch_size=64, # Increased for better performance with large data
    validation_data=(X_val_scaled, X_val_scaled),
    callbacks=[early_stop],
    verbose=1
)
```

```
Epoch 1/50
194506/194506 □□□□□□□□□□□□□□□□□□□□□□ 186s
950us/step - loss: 0.0893 - val_loss: 1.0873
Epoch 2/50
194506/194506 □□□□□□□□□□□□□□□□□□□□□□ 186s
957us/step - loss: 0.2209 - val_loss: 1.0881
Epoch 3/50
194506/194506 □□□□□□□□□□□□□□□□□□□□□□ 185s
950us/step - loss: 0.2165 - val_loss: 1.0555
Epoch 4/50
194506/194506 □□□□□□□□□□□□□□□□□□□□□□ 186s
954us/step - loss: 0.0618 - val_loss: 1.0769
Epoch 5/50
194506/194506 □□□□□□□□□□□□□□□□□□□□□□ 186s
954us/step - loss: 0.0356 - val_loss: 1.0508
Epoch 6/50
194506/194506 □□□□□□□□□□□□□□□□□□□□□□ 187s
958us/step - loss: 0.0385 - val_loss: 1.0707
Epoch 7/50
194506/194506 □□□□□□□□□□□□□□□□□□□□□□ 186s
953us/step - loss: 0.3211 - val_loss: 1.0857
Epoch 8/50
194506/194506 □□□□□□□□□□□□□□□□□□□□□□ 185s
948us/step - loss: 0.1941 - val_loss: 1.1009
Epoch 9/50
194506/194506 □□□□□□□□□□□□□□□□□□□□□□ 184s
946us/step - loss: 0.1401 - val_loss: 1.0973
Epoch 10/50
```

194506/194506 ━━━━━━━━━━━━━━━━━━━━ 185s
949us/step - loss: 0.2675 - val_loss: 1.1895
Epoch 10: early stopping
Restoring model weights from the end of the best epoch: 5.

[20]:
```
# Plot training history
plt.plot(history.history['loss'], label='Train Loss')
plt.plot(history.history['val_loss'], label='Validation Loss')
plt.title('Autoencoder Training Loss')
plt.xlabel('Epoch')
plt.ylabel('Loss')
plt.legend()
plt.show()
# Compute reconstruction error on test set
reconstructions = autoencoder.predict(X_test_scaled)
mse = np.mean(np.power(X_test_scaled - reconstructions, 2), axis=1)

# Define threshold (e.g., 95th percentile)
threshold = np.percentile(mse, 95)
# Predict anomalies
y_pred_autoencoder = (mse > threshold).astype(int)
# Evaluate predictions
print("Autoencoder Classification Report:\n",␣

  classification_report(y_test_clean,y_pred_autoencoder))

print("\nConfusionMatrix:\n",confusion_matrix(y_test_clean,␣
  y_pred_autoencoder))
```



48627/48627
□□□□□□□□□□□□□□□□□□□□□□□□□□□□ 23s
477us/step Autoencoder
Classification Report:
Confusion Matrix:
[[677694 15011]
[800545 62792]]

	precision	recall	f1-score	support
0	0.46	0.98	0.62	692705
1	0.81	0.07	0.13	863337
accuracy			0.48	155604 2
macro avg	0.63	0.53	0.38	155604 2
weighted avg	0.65	0.48	0.35	155604 2

[]: