machine_learning_project

December 9, 2025

1 MOD10: Machine learning

Fall 2025 By : Arthur BOUTELIER, Alexandre BRENSKI, Paul LESAGE, Adrien RIVET Instructor: Mohammed A. Shehab

GITHUB : https://github.com/Arthur-Boutelier/machineLearningCybersecurity

2 Introduction

This project implements a complete machine learning pipeline for network intrusion detection using a dataset downloaded from Kaggle. The pipeline covers: - Data acquisition - Cleaning & preprocessing - Feature analysis - Handling categorical & numerical attributes - Scaling & encoding - Train/test splitting - Model training - Evaluation (confusion matrix, classification report, heatmap visualization)

2.1 Environment Setup & Importing Dependencies

[ ]:
```
! pip install -r requirements.txt
```

```
Requirement already satisfied: alembic==1.17.2 in c:\users\arthu\onedrive\efrei\
i1\s5\machinelearning\machinelearningcybersecurity\lib\site-packages (from -r
requirements.txt (line 1)) (1.17.2)
Requirement already satisfied: annotated-doc==0.0.4 in c:\users\arthu\onedrive\e
frei\i1\s5\machinelearning\machinelearningcybersecurity\lib\site-packages (from
-r requirements.txt (line 2)) (0.0.4)
Requirement already satisfied: annotated-types==0.7.0 in c:\users\arthu\onedrive
\efrei\i1\s5\machinelearning\machinelearningcybersecurity\lib\site-packages
(from -r requirements.txt (line 3)) (0.7.0)
Requirement already satisfied: anyio==4.12.0 in c:\users\arthu\onedrive\efrei\i1
\s5\machinelearning\machinelearningcybersecurity\lib\site-packages (from -r
requirements.txt (line 4)) (4.12.0)
Requirement already satisfied: asttokens==3.0.0 in c:\users\arthu\onedrive\efrei
\i1\s5\machinelearning\machinelearningcybersecurity\lib\site-packages (from -r
requirements.txt (line 5)) (3.0.0)
Requirement already satisfied: blinker==1.9.0 in c:\users\arthu\onedrive\efrei\i
1\s5\machinelearning\machinelearningcybersecurity\lib\site-packages (from -r
requirements.txt (line 6)) (1.9.0)
Requirement already satisfied: cachetools==6.2.2 in c:\users\arthu\onedrive\efre
```

```
i\i1\s5\machinelearning\machinelearningcybersecurity\lib\site-packages (from -r
requirements.txt (line 7)) (6.2.2)
Requirement already satisfied: certifi==2025.11.12 in c:\users\arthu\onedrive\ef
rei\i1\s5\machinelearning\machinelearningcybersecurity\lib\site-packages (from
-r requirements.txt (line 8)) (2025.11.12)
Requirement already satisfied: cffi==2.0.0 in c:\users\arthu\onedrive\efrei\i1\s
5\machinelearning\machinelearningcybersecurity\lib\site-packages (from -r
requirements.txt (line 9)) (2.0.0)
Requirement already satisfied: charset-normalizer==3.4.4 in c:\users\arthu\onedr
ive\efrei\i1\s5\machinelearning\machinelearningcybersecurity\lib\site-packages
(from -r requirements.txt (line 10)) (3.4.4)
Requirement already satisfied: click==8.3.1 in c:\users\arthu\onedrive\efrei\i1\
s5\machinelearning\machinelearningcybersecurity\lib\site-packages (from -r
requirements.txt (line 11)) (8.3.1)
Requirement already satisfied: cloudpickle==3.1.2 in c:\users\arthu\onedrive\efr
ei\i1\s5\machinelearning\machinelearningcybersecurity\lib\site-packages (from -r
requirements.txt (line 12)) (3.1.2)
Requirement already satisfied: colorama==0.4.6 in c:\users\arthu\onedrive\efrei\
i1\s5\machinelearning\machinelearningcybersecurity\lib\site-packages (from -r
requirements.txt (line 13)) (0.4.6)
Requirement already satisfied: comm==0.2.3 in c:\users\arthu\onedrive\efrei\i1\s
5\machinelearning\machinelearningcybersecurity\lib\site-packages (from -r
requirements.txt (line 14)) (0.2.3)
Requirement already satisfied: contourpy==1.3.3 in c:\users\arthu\onedrive\efrei
\i1\s5\machinelearning\machinelearningcybersecurity\lib\site-packages (from -r
requirements.txt (line 15)) (1.3.3)
Requirement already satisfied: cryptography==46.0.3 in c:\users\arthu\onedrive\e
frei\i1\s5\machinelearning\machinelearningcybersecurity\lib\site-packages (from
-r requirements.txt (line 16)) (46.0.3)
Requirement already satisfied: cycler==0.12.1 in c:\users\arthu\onedrive\efrei\i
1\s5\machinelearning\machinelearningcybersecurity\lib\site-packages (from -r
requirements.txt (line 17)) (0.12.1)
Requirement already satisfied: databricks-sdk==0.73.0 in c:\users\arthu\onedrive
\efrei\i1\s5\machinelearning\machinelearningcybersecurity\lib\site-packages
(from -r requirements.txt (line 18)) (0.73.0)
Requirement already satisfied: debugpy==1.8.17 in c:\users\arthu\onedrive\efrei\
i1\s5\machinelearning\machinelearningcybersecurity\lib\site-packages (from -r
requirements.txt (line 19)) (1.8.17)
Requirement already satisfied: decorator==5.2.1 in c:\users\arthu\onedrive\efrei
\i1\s5\machinelearning\machinelearningcybersecurity\lib\site-packages (from -r
requirements.txt (line 20)) (5.2.1)
Requirement already satisfied: docker==7.1.0 in c:\users\arthu\onedrive\efrei\i1
\s5\machinelearning\machinelearningcybersecurity\lib\site-packages (from -r
requirements.txt (line 21)) (7.1.0)
Requirement already satisfied: executing==2.2.1 in c:\users\arthu\onedrive\efrei
\i1\s5\machinelearning\machinelearningcybersecurity\lib\site-packages (from -r
requirements.txt (line 22)) (2.2.1)
Requirement already satisfied: fastapi==0.124.0 in c:\users\arthu\onedrive\efrei
```

```
\i1\s5\machinelearning\machinelearningcybersecurity\lib\site-packages (from -r
requirements.txt (line 23)) (0.124.0)
Requirement already satisfied: Flask==3.1.2 in c:\users\arthu\onedrive\efrei\i1\
s5\machinelearning\machinelearningcybersecurity\lib\site-packages (from -r
requirements.txt (line 24)) (3.1.2)
Requirement already satisfied: flask-cors==6.0.1 in c:\users\arthu\onedrive\efre
i\i1\s5\machinelearning\machinelearningcybersecurity\lib\site-packages (from -r
requirements.txt (line 25)) (6.0.1)
Requirement already satisfied: fonttools==4.60.1 in c:\users\arthu\onedrive\efre
i\i1\s5\machinelearning\machinelearningcybersecurity\lib\site-packages (from -r
requirements.txt (line 26)) (4.60.1)
Requirement already satisfied: gitdb==4.0.12 in c:\users\arthu\onedrive\efrei\i1
\s5\machinelearning\machinelearningcybersecurity\lib\site-packages (from -r
requirements.txt (line 27)) (4.0.12)
Requirement already satisfied: GitPython==3.1.45 in c:\users\arthu\onedrive\efre
i\i1\s5\machinelearning\machinelearningcybersecurity\lib\site-packages (from -r
requirements.txt (line 28)) (3.1.45)
Requirement already satisfied: google-auth==2.43.0 in c:\users\arthu\onedrive\ef
rei\i1\s5\machinelearning\machinelearningcybersecurity\lib\site-packages (from
-r requirements.txt (line 29)) (2.43.0)
Requirement already satisfied: graphene==3.4.3 in c:\users\arthu\onedrive\efrei\
i1\s5\machinelearning\machinelearningcybersecurity\lib\site-packages (from -r
requirements.txt (line 30)) (3.4.3)
Requirement already satisfied: graphql-core==3.2.7 in c:\users\arthu\onedrive\ef
rei\i1\s5\machinelearning\machinelearningcybersecurity\lib\site-packages (from
-r requirements.txt (line 31)) (3.2.7)
Requirement already satisfied: graphql-relay==3.2.0 in c:\users\arthu\onedrive\e
frei\i1\s5\machinelearning\machinelearningcybersecurity\lib\site-packages (from
-r requirements.txt (line 32)) (3.2.0)
Requirement already satisfied: greenlet==3.3.0 in c:\users\arthu\onedrive\efrei\
i1\s5\machinelearning\machinelearningcybersecurity\lib\site-packages (from -r
requirements.txt (line 33)) (3.3.0)
Requirement already satisfied: h11==0.16.0 in c:\users\arthu\onedrive\efrei\i1\s
5\machinelearning\machinelearningcybersecurity\lib\site-packages (from -r
requirements.txt (line 34)) (0.16.0)
Requirement already satisfied: huey==2.5.5 in c:\users\arthu\onedrive\efrei\i1\s
5\machinelearning\machinelearningcybersecurity\lib\site-packages (from -r
requirements.txt (line 35)) (2.5.5)
Requirement already satisfied: idna==3.11 in c:\users\arthu\onedrive\efrei\i1\s5
\machinelearning\machinelearningcybersecurity\lib\site-packages (from -r
requirements.txt (line 36)) (3.11)
Requirement already satisfied: imbalanced-learn==0.14.0 in c:\users\arthu\onedri
ve\efrei\i1\s5\machinelearning\machinelearningcybersecurity\lib\site-packages
(from -r requirements.txt (line 37)) (0.14.0)
Requirement already satisfied: imblearn==0.0 in c:\users\arthu\onedrive\efrei\i1
\s5\machinelearning\machinelearningcybersecurity\lib\site-packages (from -r
requirements.txt (line 38)) (0.0)
Requirement already satisfied: importlib_metadata==8.7.0 in c:\users\arthu\onedr
```

```
ive\efrei\i1\s5\machinelearning\machinelearningcybersecurity\lib\site-packages
(from -r requirements.txt (line 39)) (8.7.0)
Requirement already satisfied: ipykernel==7.1.0 in c:\users\arthu\onedrive\efrei
\i1\s5\machinelearning\machinelearningcybersecurity\lib\site-packages (from -r
requirements.txt (line 40)) (7.1.0)
Requirement already satisfied: ipython==9.7.0 in c:\users\arthu\onedrive\efrei\i
1\s5\machinelearning\machinelearningcybersecurity\lib\site-packages (from -r
requirements.txt (line 41)) (9.7.0)
Requirement already satisfied: ipython_pygments_lexers==1.1.1 in c:\users\arthu\
onedrive\efrei\i1\s5\machinelearning\machinelearningcybersecurity\lib\site-
packages (from -r requirements.txt (line 42)) (1.1.1)
Requirement already satisfied: itsdangerous==2.2.0 in c:\users\arthu\onedrive\ef
rei\i1\s5\machinelearning\machinelearningcybersecurity\lib\site-packages (from
-r requirements.txt (line 43)) (2.2.0)
Requirement already satisfied: jedi==0.19.2 in c:\users\arthu\onedrive\efrei\i1\
s5\machinelearning\machinelearningcybersecurity\lib\site-packages (from -r
requirements.txt (line 44)) (0.19.2)
Requirement already satisfied: Jinja2==3.1.6 in c:\users\arthu\onedrive\efrei\i1
\s5\machinelearning\machinelearningcybersecurity\lib\site-packages (from -r
requirements.txt (line 45)) (3.1.6)
Requirement already satisfied: joblib==1.5.2 in c:\users\arthu\onedrive\efrei\i1
\s5\machinelearning\machinelearningcybersecurity\lib\site-packages (from -r
requirements.txt (line 46)) (1.5.2)
Requirement already satisfied: jupyter_client==8.6.3 in c:\users\arthu\onedrive\
efrei\i1\s5\machinelearning\machinelearningcybersecurity\lib\site-packages (from
-r requirements.txt (line 47)) (8.6.3)
Requirement already satisfied: jupyter_core==5.9.1 in c:\users\arthu\onedrive\ef
rei\i1\s5\machinelearning\machinelearningcybersecurity\lib\site-packages (from
-r requirements.txt (line 48)) (5.9.1)
Requirement already satisfied: kagglehub==0.3.13 in c:\users\arthu\onedrive\efre
i\i1\s5\machinelearning\machinelearningcybersecurity\lib\site-packages (from -r
requirements.txt (line 49)) (0.3.13)
Requirement already satisfied: kiwisolver==1.4.9 in c:\users\arthu\onedrive\efre
i\i1\s5\machinelearning\machinelearningcybersecurity\lib\site-packages (from -r
requirements.txt (line 50)) (1.4.9)
Requirement already satisfied: Mako==1.3.10 in c:\users\arthu\onedrive\efrei\i1\
s5\machinelearning\machinelearningcybersecurity\lib\site-packages (from -r
requirements.txt (line 51)) (1.3.10)
Requirement already satisfied: MarkupSafe==3.0.3 in c:\users\arthu\onedrive\efre
i\i1\s5\machinelearning\machinelearningcybersecurity\lib\site-packages (from -r
requirements.txt (line 52)) (3.0.3)
Requirement already satisfied: matplotlib==3.10.7 in c:\users\arthu\onedrive\efr
ei\i1\s5\machinelearning\machinelearningcybersecurity\lib\site-packages (from -r
requirements.txt (line 53)) (3.10.7)
Requirement already satisfied: matplotlib-inline==0.2.1 in c:\users\arthu\onedri
ve\efrei\i1\s5\machinelearning\machinelearningcybersecurity\lib\site-packages
(from -r requirements.txt (line 54)) (0.2.1)
Requirement already satisfied: mlflow==3.6.0 in c:\users\arthu\onedrive\efrei\i1
```

```
\s5\machinelearning\machinelearningcybersecurity\lib\site-packages (from -r
requirements.txt (line 55)) (3.6.0)
Requirement already satisfied: mlflow-skinny==3.6.0 in c:\users\arthu\onedrive\e
frei\i1\s5\machinelearning\machinelearningcybersecurity\lib\site-packages (from
-r requirements.txt (line 56)) (3.6.0)
Requirement already satisfied: mlflow-tracing==3.6.0 in c:\users\arthu\onedrive\
efrei\i1\s5\machinelearning\machinelearningcybersecurity\lib\site-packages (from
-r requirements.txt (line 57)) (3.6.0)
Requirement already satisfied: nest-asyncio==1.6.0 in c:\users\arthu\onedrive\ef
rei\i1\s5\machinelearning\machinelearningcybersecurity\lib\site-packages (from
-r requirements.txt (line 58)) (1.6.0)
Requirement already satisfied: numpy==2.3.4 in c:\users\arthu\onedrive\efrei\i1\
s5\machinelearning\machinelearningcybersecurity\lib\site-packages (from -r
requirements.txt (line 59)) (2.3.4)
Requirement already satisfied: opentelemetry-api==1.39.0 in c:\users\arthu\onedr
ive\efrei\i1\s5\machinelearning\machinelearningcybersecurity\lib\site-packages
(from -r requirements.txt (line 60)) (1.39.0)
Requirement already satisfied: opentelemetry-proto==1.39.0 in c:\users\arthu\one
drive\efrei\i1\s5\machinelearning\machinelearningcybersecurity\lib\site-packages
(from -r requirements.txt (line 61)) (1.39.0)
Requirement already satisfied: opentelemetry-sdk==1.39.0 in c:\users\arthu\onedr
ive\efrei\i1\s5\machinelearning\machinelearningcybersecurity\lib\site-packages
(from -r requirements.txt (line 62)) (1.39.0)
Requirement already satisfied: opentelemetry-semantic-conventions==0.60b0 in c:\
users\arthu\onedrive\efrei\i1\s5\machinelearning\machinelearningcybersecurity\li
b\site-packages (from -r requirements.txt (line 63)) (0.60b0)
Requirement already satisfied: packaging==25.0 in c:\users\arthu\onedrive\efrei\
i1\s5\machinelearning\machinelearningcybersecurity\lib\site-packages (from -r
requirements.txt (line 64)) (25.0)
Requirement already satisfied: pandas==2.3.3 in c:\users\arthu\onedrive\efrei\i1
\s5\machinelearning\machinelearningcybersecurity\lib\site-packages (from -r
requirements.txt (line 65)) (2.3.3)
Requirement already satisfied: parso==0.8.5 in c:\users\arthu\onedrive\efrei\i1\
s5\machinelearning\machinelearningcybersecurity\lib\site-packages (from -r
requirements.txt (line 66)) (0.8.5)
Requirement already satisfied: pillow==12.0.0 in c:\users\arthu\onedrive\efrei\i
1\s5\machinelearning\machinelearningcybersecurity\lib\site-packages (from -r
requirements.txt (line 67)) (12.0.0)
Requirement already satisfied: platformdirs==4.5.0 in c:\users\arthu\onedrive\ef
rei\i1\s5\machinelearning\machinelearningcybersecurity\lib\site-packages (from
-r requirements.txt (line 68)) (4.5.0)
Requirement already satisfied: prompt_toolkit==3.0.52 in c:\users\arthu\onedrive
\efrei\i1\s5\machinelearning\machinelearningcybersecurity\lib\site-packages
(from -r requirements.txt (line 69)) (3.0.52)
Requirement already satisfied: protobuf==6.33.2 in c:\users\arthu\onedrive\efrei
\i1\s5\machinelearning\machinelearningcybersecurity\lib\site-packages (from -r
requirements.txt (line 70)) (6.33.2)
Requirement already satisfied: psutil==7.1.3 in c:\users\arthu\onedrive\efrei\i1
```

```
\s5\machinelearning\machinelearningcybersecurity\lib\site-packages (from -r
requirements.txt (line 71)) (7.1.3)
Requirement already satisfied: pure_eval==0.2.3 in c:\users\arthu\onedrive\efrei
\i1\s5\machinelearning\machinelearningcybersecurity\lib\site-packages (from -r
requirements.txt (line 72)) (0.2.3)
Requirement already satisfied: pyarrow==22.0.0 in c:\users\arthu\onedrive\efrei\
i1\s5\machinelearning\machinelearningcybersecurity\lib\site-packages (from -r
requirements.txt (line 73)) (22.0.0)
Requirement already satisfied: pyasn1==0.6.1 in c:\users\arthu\onedrive\efrei\i1
\s5\machinelearning\machinelearningcybersecurity\lib\site-packages (from -r
requirements.txt (line 74)) (0.6.1)
Requirement already satisfied: pyasn1_modules==0.4.2 in c:\users\arthu\onedrive\
efrei\i1\s5\machinelearning\machinelearningcybersecurity\lib\site-packages (from
-r requirements.txt (line 75)) (0.4.2)
Requirement already satisfied: pycparser==2.23 in c:\users\arthu\onedrive\efrei\
i1\s5\machinelearning\machinelearningcybersecurity\lib\site-packages (from -r
requirements.txt (line 76)) (2.23)
Requirement already satisfied: pydantic==2.12.5 in c:\users\arthu\onedrive\efrei
\i1\s5\machinelearning\machinelearningcybersecurity\lib\site-packages (from -r
requirements.txt (line 77)) (2.12.5)
Requirement already satisfied: pydantic_core==2.41.5 in c:\users\arthu\onedrive\
efrei\i1\s5\machinelearning\machinelearningcybersecurity\lib\site-packages (from
-r requirements.txt (line 78)) (2.41.5)
Requirement already satisfied: Pygments==2.19.2 in c:\users\arthu\onedrive\efrei
\i1\s5\machinelearning\machinelearningcybersecurity\lib\site-packages (from -r
requirements.txt (line 79)) (2.19.2)
Requirement already satisfied: pyparsing==3.2.5 in c:\users\arthu\onedrive\efrei
\i1\s5\machinelearning\machinelearningcybersecurity\lib\site-packages (from -r
requirements.txt (line 80)) (3.2.5)
Requirement already satisfied: python-dateutil==2.9.0.post0 in c:\users\arthu\on
edrive\efrei\i1\s5\machinelearning\machinelearningcybersecurity\lib\site-
packages (from -r requirements.txt (line 81)) (2.9.0.post0)
Requirement already satisfied: python-dotenv==1.2.1 in c:\users\arthu\onedrive\e
frei\i1\s5\machinelearning\machinelearningcybersecurity\lib\site-packages (from
-r requirements.txt (line 82)) (1.2.1)
Requirement already satisfied: pytz==2025.2 in c:\users\arthu\onedrive\efrei\i1\
s5\machinelearning\machinelearningcybersecurity\lib\site-packages (from -r
requirements.txt (line 83)) (2025.2)
Requirement already satisfied: pywin32==311 in c:\users\arthu\onedrive\efrei\i1\
s5\machinelearning\machinelearningcybersecurity\lib\site-packages (from -r
requirements.txt (line 84)) (311)
Requirement already satisfied: PyYAML==6.0.3 in c:\users\arthu\onedrive\efrei\i1
\s5\machinelearning\machinelearningcybersecurity\lib\site-packages (from -r
requirements.txt (line 85)) (6.0.3)
Requirement already satisfied: pyzmq==27.1.0 in c:\users\arthu\onedrive\efrei\i1
\s5\machinelearning\machinelearningcybersecurity\lib\site-packages (from -r
requirements.txt (line 86)) (27.1.0)
Requirement already satisfied: requests==2.32.5 in c:\users\arthu\onedrive\efrei
```

```
\i1\s5\machinelearning\machinelearningcybersecurity\lib\site-packages (from -r
requirements.txt (line 87)) (2.32.5)
Requirement already satisfied: rsa==4.9.1 in c:\users\arthu\onedrive\efrei\i1\s5
\machinelearning\machinelearningcybersecurity\lib\site-packages (from -r
requirements.txt (line 88)) (4.9.1)
Requirement already satisfied: scikit-learn==1.7.2 in c:\users\arthu\onedrive\ef
rei\i1\s5\machinelearning\machinelearningcybersecurity\lib\site-packages (from
-r requirements.txt (line 89)) (1.7.2)
Requirement already satisfied: scipy==1.16.3 in c:\users\arthu\onedrive\efrei\i1
\s5\machinelearning\machinelearningcybersecurity\lib\site-packages (from -r
requirements.txt (line 90)) (1.16.3)
Requirement already satisfied: seaborn==0.13.2 in c:\users\arthu\onedrive\efrei\
i1\s5\machinelearning\machinelearningcybersecurity\lib\site-packages (from -r
requirements.txt (line 91)) (0.13.2)
Requirement already satisfied: six==1.17.0 in c:\users\arthu\onedrive\efrei\i1\s
5\machinelearning\machinelearningcybersecurity\lib\site-packages (from -r
requirements.txt (line 92)) (1.17.0)
Requirement already satisfied: smmap==5.0.2 in c:\users\arthu\onedrive\efrei\i1\
s5\machinelearning\machinelearningcybersecurity\lib\site-packages (from -r
requirements.txt (line 93)) (5.0.2)
Requirement already satisfied: SQLAlchemy==2.0.44 in c:\users\arthu\onedrive\efr
ei\i1\s5\machinelearning\machinelearningcybersecurity\lib\site-packages (from -r
requirements.txt (line 94)) (2.0.44)
Requirement already satisfied: sqlparse==0.5.4 in c:\users\arthu\onedrive\efrei\
i1\s5\machinelearning\machinelearningcybersecurity\lib\site-packages (from -r
requirements.txt (line 95)) (0.5.4)
Requirement already satisfied: stack-data==0.6.3 in c:\users\arthu\onedrive\efre
i\i1\s5\machinelearning\machinelearningcybersecurity\lib\site-packages (from -r
requirements.txt (line 96)) (0.6.3)
Requirement already satisfied: starlette==0.50.0 in c:\users\arthu\onedrive\efre
i\i1\s5\machinelearning\machinelearningcybersecurity\lib\site-packages (from -r
requirements.txt (line 97)) (0.50.0)
Requirement already satisfied: threadpoolctl==3.6.0 in c:\users\arthu\onedrive\e
frei\i1\s5\machinelearning\machinelearningcybersecurity\lib\site-packages (from
-r requirements.txt (line 98)) (3.6.0)
Requirement already satisfied: tornado==6.5.2 in c:\users\arthu\onedrive\efrei\i
1\s5\machinelearning\machinelearningcybersecurity\lib\site-packages (from -r
requirements.txt (line 99)) (6.5.2)
Requirement already satisfied: tqdm==4.67.1 in c:\users\arthu\onedrive\efrei\i1\
s5\machinelearning\machinelearningcybersecurity\lib\site-packages (from -r
requirements.txt (line 100)) (4.67.1)
Requirement already satisfied: traitlets==5.14.3 in c:\users\arthu\onedrive\efre
i\i1\s5\machinelearning\machinelearningcybersecurity\lib\site-packages (from -r
requirements.txt (line 101)) (5.14.3)
Requirement already satisfied: typing-inspection==0.4.2 in c:\users\arthu\onedri
ve\efrei\i1\s5\machinelearning\machinelearningcybersecurity\lib\site-packages
(from -r requirements.txt (line 102)) (0.4.2)
Requirement already satisfied: typing_extensions==4.15.0 in c:\users\arthu\onedr
```

```
ive\efrei\i1\s5\machinelearning\machinelearningcybersecurity\lib\site-packages
(from -r requirements.txt (line 103)) (4.15.0)
Requirement already satisfied: tzdata==2025.2 in c:\users\arthu\onedrive\efrei\i
1\s5\machinelearning\machinelearningcybersecurity\lib\site-packages (from -r
requirements.txt (line 104)) (2025.2)
Requirement already satisfied: urllib3==2.5.0 in c:\users\arthu\onedrive\efrei\i
1\s5\machinelearning\machinelearningcybersecurity\lib\site-packages (from -r
requirements.txt (line 105)) (2.5.0)
Requirement already satisfied: uvicorn==0.38.0 in c:\users\arthu\onedrive\efrei\
i1\s5\machinelearning\machinelearningcybersecurity\lib\site-packages (from -r
requirements.txt (line 106)) (0.38.0)
Requirement already satisfied: waitress==3.0.2 in c:\users\arthu\onedrive\efrei\
i1\s5\machinelearning\machinelearningcybersecurity\lib\site-packages (from -r
requirements.txt (line 107)) (3.0.2)
Requirement already satisfied: wcwidth==0.2.14 in c:\users\arthu\onedrive\efrei\
i1\s5\machinelearning\machinelearningcybersecurity\lib\site-packages (from -r
requirements.txt (line 108)) (0.2.14)
Requirement already satisfied: Werkzeug==3.1.4 in c:\users\arthu\onedrive\efrei\
i1\s5\machinelearning\machinelearningcybersecurity\lib\site-packages (from -r
requirements.txt (line 109)) (3.1.4)
Requirement already satisfied: xgboost==3.1.2 in c:\users\arthu\onedrive\efrei\i
1\s5\machinelearning\machinelearningcybersecurity\lib\site-packages (from -r
requirements.txt (line 110)) (3.1.2)
Requirement already satisfied: zipp==3.23.0 in c:\users\arthu\onedrive\efrei\i1\
s5\machinelearning\machinelearningcybersecurity\lib\site-packages (from -r
requirements.txt (line 111)) (3.23.0)


[notice] A new release of pip is available: 25.2 -> 25.3
[notice] To update, run: python.exe -m pip install --upgrade pip
```

This block imports all required Python libraries used throughout the project.

1/ General-purpose libraries - os: Interacting with the operating system (listing files, building paths). - pandas: For data loading, manipulation, DataFrame operations. - numpy: For numerical computation, array processing. - seaborn, matplotlib.pyplot: For plotting charts and visual analysis.

2/ Machine Learning - StandardScaler: Normalizes numerical features (important for models like Logistic Regression). - LabelEncoder: Converts categorical labels into integers. - train_test_split: Splits data into training/testing sets. - LogisticRegression: A baseline ML classification model. - classification_report, confusion_matrix: Evaluation metrics. - DecisionTreeClassifier: A simple tree-based ML model.

3/ Advanced ML - xgboost: Gradient boosting model, high-performance. - mlflow: Used for experiment tracking and model versioning.

4/ Purpose - This central import block prepares all tools needed for: - Data cleaning - Preprocessing - Machine learning - Visualization - Model evaluation - Experiment logging

[42]:
```python
import os
import pandas as pd
import numpy as np
import seaborn as sns
import matplotlib.pyplot as plt
from sklearn.preprocessing import LabelEncoder, StandardScaler
import kagglehub
from sklearn.model_selection import train_test_split, GridSearchCV
from imblearn.over_sampling import SMOTE
from sklearn.linear_model import LogisticRegression
from sklearn.neighbors import KNeighborsClassifier
from sklearn.metrics import classification_report, confusion_matrix,␣
↪accuracy_score, f1_score
from sklearn.svm import SVC
from sklearn.ensemble import RandomForestClassifier, VotingClassifier,␣
↪StackingClassifier, GradientBoostingClassifier
from sklearn.tree import DecisionTreeClassifier
import mlflow as mlf
import xgboost as xgb
```

3 1. Data Preprocessing and EDA

3.1 Load and clean dataset

Load and clean the dataset, handling missing values, normalizing or scaling numerical features, and encoding categorical variables.

This block downloads the network intrusion Dataset from Kaggle through kagglehub.

[48]:
```python
# Download latest version
path = kagglehub.dataset_download("chethuhn/network-intrusion-dataset")
print("Path to dataset files:", path)
models = []
```

```
Path to dataset files:
C:\Users\arthu\.cache\kagglehub\datasets\chethuhn\network-intrusion-
dataset\versions\1
```

This block lists all files downloaded from Kaggle.

[49]:
```python
# see list of downloaded dataset files
list_of_files = os.listdir(path)
print("Files in dataset:", list_of_files)
```

```
Files in dataset: ['Friday-WorkingHours-Afternoon-DDos.pcap_ISCX.csv', 'Friday-
WorkingHours-Afternoon-PortScan.pcap_ISCX.csv', 'Friday-WorkingHours-
Morning.pcap_ISCX.csv', 'Monday-WorkingHours.pcap_ISCX.csv', 'Thursday-
WorkingHours-Afternoon-Infilteration.pcap_ISCX.csv', 'Thursday-WorkingHours-
Morning-WebAttacks.pcap_ISCX.csv', 'Tuesday-WorkingHours.pcap_ISCX.csv',
```

'Wednesday-workingHours.pcap_ISCX.csv']

Loads the dataset into a pandas DataFrame. df.head() shows the first 5 rows to verify correct loading.

[50]:
```python
# Load first dataset
df = pd.read_csv(f"{path}/{list_of_files[0]}")
df.head()
```

[50]:
```
   Destination Port   Flow Duration   Total Fwd Packets  \
0             54865               3                   2
1             55054             109                   1
2             55055              52                   1
3             46236              34                   1
4             54863               3                   2

   Total Backward Packets  Total Length of Fwd Packets  \
0                       0                           12
1                       1                            6
2                       1                            6
3                       1                            6
4                       0                           12

   Total Length of Bwd Packets   Fwd Packet Length Max  \
0                            0                       6
1                            6                       6
2                            6                       6
3                            6                       6
4                            0                       6

   Fwd Packet Length Min   Fwd Packet Length Mean   Fwd Packet Length Std  \
0                      6                      6.0                     0.0
1                      6                      6.0                     0.0
2                      6                      6.0                     0.0
3                      6                      6.0                     0.0
4                      6                      6.0                     0.0

   …   min_seg_size_forward  Active Mean   Active Std   Active Max  \
0  …                     20          0.0          0.0            0
1  …                     20          0.0          0.0            0
2  …                     20          0.0          0.0            0
3  …                     20          0.0          0.0            0
4  …                     20          0.0          0.0            0

   Active Min  Idle Mean   Idle Std   Idle Max   Idle Min   Label
0           0        0.0        0.0          0          0  BENIGN
1           0        0.0        0.0          0          0  BENIGN
2           0        0.0        0.0          0          0  BENIGN
```

```
3              0        0.0        0.0          0           0  BENIGN
4              0        0.0        0.0          0           0  BENIGN

[5 rows x 79 columns]
```

This code loads every CSV file in the dataset, adds a column identifying the file of origin, stores each loaded file in a list, and finally concatenates all of them into one unified DataFrame while printing its final shape and previewing the first rows.

We concatenate all datasets because they have the same columns and represent different days of the same measurements.

We concatenate all datasets because they have the same columns and represent different days of the same measurements.

[51]:

```python
# concatenate all dataset files
dataframes = []
for i, file in enumerate(list_of_files):
    df_part = pd.read_csv(f"{path}/{file}")
    df_part['source_file'] = i  # optional: add a column to identify source file
    dataframes.append(df_part)

# concatenate all dataframes into a single dataframe
df = pd.concat(dataframes, axis=0, ignore_index=True)
print("Combined dataset shape:", df.shape)
df.head()
```

```
Combined dataset shape: (2830743, 80)
```

[51]:

```
    Destination Port   Flow Duration   Total Fwd Packets  \
0              54865               3                   2
1              55054             109                   1
2              55055              52                   1
3              46236              34                   1
4              54863               3                   2

    Total Backward Packets  Total Length of Fwd Packets  \
0                        0                           12
1                        1                            6
2                        1                            6
3                        1                            6
4                        0                           12

    Total Length of Bwd Packets   Fwd Packet Length Max  \
0                             0                       6
1                             6                       6
2                             6                       6
3                             6                       6
4                             0                       6
```

```
   Fwd Packet Length Min  Fwd Packet Length Mean  Fwd Packet Length Std  \
0                      6                     6.0                    0.0
1                      6                     6.0                    0.0
2                      6                     6.0                    0.0
3                      6                     6.0                    0.0
4                      6                     6.0                    0.0

   …  Active Mean  Active Std  Active Max  Active Min  Idle Mean  \
0  …          0.0         0.0           0           0        0.0
1  …          0.0         0.0           0           0        0.0
2  …          0.0         0.0           0           0        0.0
3  …          0.0         0.0           0           0        0.0
4  …          0.0         0.0           0           0        0.0

   Idle Std  Idle Max  Idle Min   Label  source_file
0       0.0         0         0  BENIGN            0
1       0.0         0         0  BENIGN            0
2       0.0         0         0  BENIGN            0
3       0.0         0         0  BENIGN            0
4       0.0         0         0  BENIGN            0

[5 rows x 80 columns]
```

This code enables a test mode that randomly selects a smaller 50,000-row subset of the full dataset to speed up development and experimentation, and then prints the reduced dataset's shape.

[52]:
```python
# define test mode, to work with a smaller subset during development, instead of 2.8M rows
TEST_MODE = True
if TEST_MODE:
    df = df.sample(n=50000, random_state=42)
    print("Test mode: using smaller dataset shape:", df.shape)
```

```
Test mode: using smaller dataset shape: (50000, 80)
```

This code temporarily configures pandas to display all columns so the first rows of the dataset can be fully inspected, prints them, and then restores the display setting to its default limit of 20 columns.

[9]:
```python
# print the first lines with all columns
pd.set_option('display.max_columns', None)
print(df.head())
pd.set_option('display.max_columns', 20)  # reset to default
```

```
         Destination Port  Flow Duration  Total Fwd Packets  \
746827              50545            232                  1
946912                 53          31226                  2
2216843                80       99951883                  9
```

```
699389                  53          30894                     4
1170268                 53          48943                     2

          Total Backward Packets  Total Length of Fwd Packets  \
746827                         1                            0
946912                         2                           68
2216843                        7                          317
699389                         2                          140
1170268                        2                           88

          Total Length of Bwd Packets   Fwd Packet Length Max  \
746827                              0                       0
946912                            380                      34
2216843                         11595                     317
699389                            172                      35
1170268                           166                      44

          Fwd Packet Length Min   Fwd Packet Length Mean  \
746827                        0                 0.000000
946912                       34                34.000000
2216843                       0                35.222222
699389                       35                35.000000
1170268                      44                44.000000

          Fwd Packet Length Std  Bwd Packet Length Max  \
746827                 0.000000                      0
946912                 0.000000                    190
2216843              105.666667                   5792
699389                 0.000000                     86
1170268                0.000000                     83

          Bwd Packet Length Min   Bwd Packet Length Mean  \
746827                        0                 0.000000
946912                      190               190.000000
2216843                       0              1656.428571
699389                       86                86.000000
1170268                      83                83.000000

          Bwd Packet Length Std  Flow Bytes/s   Flow Packets/s  \
746827                 0.000000      0.000000      8620.689655
946912                 0.000000  14347.018510       128.098380
2216843             2118.227235    119.177345         0.160077
699389                 0.000000  10099.048360       194.212468
1170268                0.000000   5189.710480        81.727724

          Flow IAT Mean   Flow IAT Std   Flow IAT Max   Flow IAT Min  \
746827     2.320000e+02   0.000000e+00            232            232
946912     1.040867e+04   1.802228e+04          31219              3
```

```
2216843   6.663459e+06  2.580000e+07     99900000            1
699389    6.178800e+03  8.507391e+03        17161            1
1170268   1.631433e+04  2.825206e+04        48937            3

         Fwd IAT Total   Fwd IAT Mean   Fwd IAT Std   Fwd IAT Max  \
746827               0            0.0  0.000000e+00             0
946912               3            3.0  0.000000e+00             3
2216843       99900000     12500000.0  3.530000e+07      99900000
699389           17211         5737.0  9.893503e+03         17161
1170268              3            3.0  0.000000e+00             3

          Fwd IAT Min  Bwd IAT Total   Bwd IAT Mean   Bwd IAT Std  \
746827              0              0            0.0        0.0000
946912              3              4            4.0        0.0000
2216843             1          56541         9423.5    18378.1771
699389              1             49           49.0        0.0000
1170268             3              3            3.0        0.0000

          Bwd IAT Max   Bwd IAT Min  Fwd PSH Flags   Bwd PSH Flags  \
746827              0             0              0               0
946912              4             4              0               0
2216843         46050            14              0               0
699389             49            49              0               0
1170268             3             3              0               0

          Fwd URG Flags   Bwd URG Flags   Fwd Header Length  \
746827                0               0                  32
946912                0               0                  64
2216843               0               0                 296
699389                0               0                 128
1170268               0               0                  40

          Bwd Header Length  Fwd Packets/s   Bwd Packets/s  \
746827                   32    4310.344828     4310.344828
946912                   40      64.049190       64.049190
2216843                 232       0.090043        0.070034
699389                   64     129.474979       64.737489
1170268                  64      40.863862       40.863862

          Min Packet Length   Max Packet Length   Packet Length Mean  \
746827                    0                   0             0.000000
946912                   34                 190            96.400000
2216843                   0                5792           700.705882
699389                   35                  86            49.571429
1170268                  44                  83            59.600000

          Packet Length Std   Packet Length Variance  FIN Flag Count  \
746827             0.000000             0.000000e+00               0
```

946912	85.444719	7.300800e+03	0
2216843	1538.694445	2.367581e+06	0
699389	24.885452	6.192857e+02	0
1170268	21.361180	4.563000e+02	0

	SYN Flag Count	RST Flag Count	PSH Flag Count	ACK Flag Count \
746827	0	0	0	1
946912	0	0	0	0
2216843	0	0	0	1
699389	0	0	0	0
1170268	0	0	0	0

	URG Flag Count	CWE Flag Count	ECE Flag Count	Down/Up Ratio \
746827	1	0	0	1
946912	0	0	0	1
2216843	0	0	0	0
699389	0	0	0	0
1170268	0	0	0	1

	Average Packet Size	Avg Fwd Segment Size	Avg Bwd Segment Size \
746827	0.000000	0.000000	0.000000
946912	120.500000	34.000000	190.000000
2216843	744.500000	35.222222	1656.428571
699389	57.833333	35.000000	86.000000
1170268	74.500000	44.000000	83.000000

	Fwd Header Length.1	Fwd Avg Bytes/Bulk	Fwd Avg Packets/Bulk \
746827	32	0	0
946912	64	0	0
2216843	296	0	0
699389	128	0	0
1170268	40	0	0

	Fwd Avg Bulk Rate	Bwd Avg Bytes/Bulk	Bwd Avg Packets/Bulk \
746827	0	0	0
946912	0	0	0
2216843	0	0	0
699389	0	0	0
1170268	0	0	0

	Bwd Avg Bulk Rate	Subflow Fwd Packets	Subflow Fwd Bytes \
746827	0	1	0
946912	0	2	68
2216843	0	9	317
699389	0	4	140
1170268	0	2	88

	Subflow Bwd Packets	Subflow Bwd Bytes	Init_Win_bytes_forward \

```
746827                        1               0                 357
946912                        2             380                  -1
2216843                       7           11595                 274
699389                        2             172                  -1
1170268                       2             166                  -1

          Init_Win_bytes_backward  act_data_pkt_fwd  min_seg_size_forward  \
746827                      32832                 0                    32
946912                         -1                 1                    32
2216843                       235                 1                    32
699389                         -1                 3                    32
1170268                        -1                 1                    20

         Active Mean  Active Std  Active Max  Active Min   Idle Mean  \
746827           0.0         0.0           0           0         0.0
946912           0.0         0.0           0           0         0.0
2216843        999.0         0.0         999         999  99900000.0
699389           0.0         0.0           0           0         0.0
1170268          0.0         0.0           0           0         0.0

          Idle Std   Idle Max   Idle Min     Label  source_file
746827         0.0          0          0    BENIGN            3
946912         0.0          0          0    BENIGN            3
2216843        0.0   99900000   99900000  DoS Hulk            7
699389         0.0          0          0    BENIGN            2
1170268        0.0          0          0    BENIGN            3
```

This code retrieves and displays all unique values present in the ' Label' column, allowing you to see which distinct classes or categories exist in the dataset.

[10]:
```
df[' Label'].unique()
```

[10]:
```
array(['BENIGN', 'DoS Hulk', 'DDoS', 'PortScan', 'DoS slowloris',
       'DoS GoldenEye', 'FTP-Patator', 'DoS Slowhttptest', 'Bot',
       'SSH-Patator', 'Web Attack  Brute Force', 'Web Attack  XSS'],
      dtype=object)
```

This code provides an overview of the dataset by displaying structural information, counting missing values per column and in total, and presenting descriptive statistics for all features to help understand the dataset's composition and potential issues.

[11]:
```
# view some basic info about the dataset
print("\nDataset Info:")
print(df.info())

print("\nMissing Values per Column:")
print(df.isnull().sum())
print(f"Total number of missing values: {df.isnull().sum().sum()}")
```

```python
print("\nBasic Statistics:")
display(df.describe(include='all'))
```

```
Dataset Info:
<class 'pandas.core.frame.DataFrame'>
Index: 50000 entries, 746827 to 2337454
Data columns (total 80 columns):
 #   Column                        Non-Null Count  Dtype
---  ------                        --------------  -----
 0    Destination Port             50000 non-null  int64
 1    Flow Duration                50000 non-null  int64
 2    Total Fwd Packets            50000 non-null  int64
 3    Total Backward Packets       50000 non-null  int64
 4   Total Length of Fwd Packets   50000 non-null  int64
 5    Total Length of Bwd Packets  50000 non-null  int64
 6    Fwd Packet Length Max        50000 non-null  int64
 7    Fwd Packet Length Min        50000 non-null  int64
 8    Fwd Packet Length Mean       50000 non-null  float64
 9    Fwd Packet Length Std        50000 non-null  float64
 10  Bwd Packet Length Max         50000 non-null  int64
 11   Bwd Packet Length Min        50000 non-null  int64
 12   Bwd Packet Length Mean       50000 non-null  float64
 13   Bwd Packet Length Std        50000 non-null  float64
 14  Flow Bytes/s                  49975 non-null  float64
 15   Flow Packets/s               50000 non-null  float64
 16   Flow IAT Mean                50000 non-null  float64
 17   Flow IAT Std                 50000 non-null  float64
 18   Flow IAT Max                 50000 non-null  int64
 19   Flow IAT Min                 50000 non-null  int64
 20  Fwd IAT Total                 50000 non-null  int64
 21   Fwd IAT Mean                 50000 non-null  float64
 22   Fwd IAT Std                  50000 non-null  float64
 23   Fwd IAT Max                  50000 non-null  int64
 24   Fwd IAT Min                  50000 non-null  int64
 25  Bwd IAT Total                 50000 non-null  int64
 26   Bwd IAT Mean                 50000 non-null  float64
 27   Bwd IAT Std                  50000 non-null  float64
 28   Bwd IAT Max                  50000 non-null  int64
 29   Bwd IAT Min                  50000 non-null  int64
 30  Fwd PSH Flags                 50000 non-null  int64
 31   Bwd PSH Flags                50000 non-null  int64
 32   Fwd URG Flags                50000 non-null  int64
 33   Bwd URG Flags                50000 non-null  int64
 34   Fwd Header Length            50000 non-null  int64
 35   Bwd Header Length            50000 non-null  int64
 36  Fwd Packets/s                 50000 non-null  float64
 37   Bwd Packets/s                50000 non-null  float64
```

```
 38   Min Packet Length           50000 non-null  int64
 39   Max Packet Length           50000 non-null  int64
 40   Packet Length Mean          50000 non-null  float64
 41   Packet Length Std           50000 non-null  float64
 42   Packet Length Variance      50000 non-null  float64
 43  FIN Flag Count               50000 non-null  int64
 44   SYN Flag Count              50000 non-null  int64
 45   RST Flag Count              50000 non-null  int64
 46   PSH Flag Count              50000 non-null  int64
 47   ACK Flag Count              50000 non-null  int64
 48   URG Flag Count              50000 non-null  int64
 49   CWE Flag Count              50000 non-null  int64
 50   ECE Flag Count              50000 non-null  int64
 51   Down/Up Ratio               50000 non-null  int64
 52   Average Packet Size         50000 non-null  float64
 53   Avg Fwd Segment Size        50000 non-null  float64
 54   Avg Bwd Segment Size        50000 non-null  float64
 55   Fwd Header Length.1         50000 non-null  int64
 56  Fwd Avg Bytes/Bulk           50000 non-null  int64
 57   Fwd Avg Packets/Bulk        50000 non-null  int64
 58   Fwd Avg Bulk Rate           50000 non-null  int64
 59   Bwd Avg Bytes/Bulk          50000 non-null  int64
 60   Bwd Avg Packets/Bulk        50000 non-null  int64
 61  Bwd Avg Bulk Rate            50000 non-null  int64
 62  Subflow Fwd Packets          50000 non-null  int64
 63   Subflow Fwd Bytes           50000 non-null  int64
 64   Subflow Bwd Packets         50000 non-null  int64
 65   Subflow Bwd Bytes           50000 non-null  int64
 66  Init_Win_bytes_forward       50000 non-null  int64
 67   Init_Win_bytes_backward     50000 non-null  int64
 68   act_data_pkt_fwd            50000 non-null  int64
 69   min_seg_size_forward        50000 non-null  int64
 70  Active Mean                  50000 non-null  float64
 71   Active Std                  50000 non-null  float64
 72   Active Max                  50000 non-null  int64
 73   Active Min                  50000 non-null  int64
 74  Idle Mean                    50000 non-null  float64
 75   Idle Std                    50000 non-null  float64
 76   Idle Max                    50000 non-null  int64
 77   Idle Min                    50000 non-null  int64
 78   Label                       50000 non-null  object
 79  source_file                  50000 non-null  int64
dtypes: float64(24), int64(55), object(1)
memory usage: 30.9+ MB
None

Missing Values per Column:
 Destination Port             0
```

```
 Flow Duration                0
 Total Fwd Packets            0
 Total Backward Packets       0
Total Length of Fwd Packets   0
                             ..
 Idle Std                     0
 Idle Max                     0
 Idle Min                     0
 Label                        0
source_file                   0
Length: 80, dtype: int64
Total number of missing values: 25

Basic Statistics:

c:\Users\arthu\OneDrive\Efrei\I1\S5\MachineLearning\machineLearningCybersecurity
\Lib\site-packages\pandas\core\nanops.py:1016: RuntimeWarning: invalid value
encountered in subtract
  sqr = _ensure_numeric((avg - values) ** 2)
c:\Users\arthu\OneDrive\Efrei\I1\S5\MachineLearning\machineLearningCybersecurity
\Lib\site-packages\pandas\core\nanops.py:1016: RuntimeWarning: invalid value
encountered in subtract
  sqr = _ensure_numeric((avg - values) ** 2)
         Destination Port   Flow Duration   Total Fwd Packets  \
count        50000.000000    5.000000e+04        50000.000000
unique                NaN             NaN                 NaN
top                   NaN             NaN                 NaN
freq                  NaN             NaN                 NaN
mean          8122.670040    1.467833e+07            6.404040
std          18355.671208    3.356638e+07           63.476262
min              0.000000   -1.000000e+00            1.000000
25%             53.000000    1.540000e+02            2.000000
50%             80.000000    3.128800e+04            2.000000
75%            443.000000    2.935594e+06            5.000000
max          65529.000000    1.199994e+08         9449.000000

         Total Backward Packets  Total Length of Fwd Packets  \
count              50000.000000                 50000.000000
unique                      NaN                          NaN
top                         NaN                          NaN
freq                        NaN                          NaN
mean                   6.329980                   518.673680
std                   76.536092                  2723.044996
min                    0.000000                     0.000000
25%                    1.000000                    12.000000
50%                    2.000000                    62.000000
75%                    4.000000                   193.000000
max                10063.000000                153145.000000
```

	Total Length of Bwd Packets	Fwd Packet Length Max	\
count	5.000000e+04	50000.000000	
unique	NaN	NaN	
top	NaN	NaN	
freq	NaN	NaN	
mean	6.986600e+03	207.416480	
std	1.650067e+05	707.720042	
min	0.000000e+00	0.000000	
25%	4.000000e+00	6.000000	
50%	1.230000e+02	37.000000	
75%	4.840000e+02	85.000000	
max	2.340090e+07	23360.000000	

	Fwd Packet Length Min	Fwd Packet Length Mean	\
count	50000.000000	50000.000000	
unique	NaN	NaN	
top	NaN	NaN	
freq	NaN	NaN	
mean	18.604720	58.313406	
std	57.708775	186.253930	
min	0.000000	0.000000	
25%	0.000000	6.000000	
50%	2.000000	34.000000	
75%	36.000000	50.000000	
max	1983.000000	4672.000000	

	Fwd Packet Length Std	…	Active Mean	Active Std	Active Max	\
count	50000.000000	…	5.000000e+04	5.000000e+04	5.000000e+04	
unique	NaN	…	NaN	NaN	NaN	
top	NaN	…	NaN	NaN	NaN	
freq	NaN	…	NaN	NaN	NaN	
mean	68.733582	…	7.749996e+04	4.327377e+04	1.535587e+05	
std	277.492447	…	5.408293e+05	4.194620e+05	1.012230e+06	
min	0.000000	…	0.000000e+00	0.000000e+00	0.000000e+00	
25%	0.000000	…	0.000000e+00	0.000000e+00	0.000000e+00	
50%	0.000000	…	0.000000e+00	0.000000e+00	0.000000e+00	
75%	26.162951	…	0.000000e+00	0.000000e+00	0.000000e+00	
max	6429.190773	…	2.920000e+07	5.040000e+07	8.740000e+07	

	Active Min	Idle Mean	Idle Std	Idle Max	Idle Min	\
count	5.000000e+04	5.000000e+04	5.000000e+04	5.000000e+04	5.000000e+04	
unique	NaN	NaN	NaN	NaN	NaN	
top	NaN	NaN	NaN	NaN	NaN	
freq	NaN	NaN	NaN	NaN	NaN	
mean	5.316534e+04	8.217208e+06	4.902213e+05	8.587389e+06	7.830262e+06	
std	4.593371e+05	2.350088e+07	4.552573e+06	2.422544e+07	2.323902e+07	
min	0.000000e+00	0.000000e+00	0.000000e+00	0.000000e+00	0.000000e+00	

```
25%     0.000000e+00  0.000000e+00  0.000000e+00  0.000000e+00  0.000000e+00
50%     0.000000e+00  0.000000e+00  0.000000e+00  0.000000e+00  0.000000e+00
75%     0.000000e+00  0.000000e+00  0.000000e+00  0.000000e+00  0.000000e+00
max     1.500000e+07  1.200000e+08  6.680000e+07  1.200000e+08  1.200000e+08

         Label   source_file
count    50000  50000.000000
unique      12           NaN
top     BENIGN           NaN
freq     40193           NaN
mean       NaN      4.156480
std        NaN      2.348877
min        NaN      0.000000
25%        NaN      2.000000
50%        NaN      4.000000
75%        NaN      6.000000
max        NaN      7.000000

[11 rows x 80 columns]
```

This code scans all columns and drops any column that contains only one unique value, since such features provide no useful information for machine learning and can be safely removed.

[12]:
```
# remove columns with the same value for all rows
for col in df.columns:
    if df[col].nunique() == 1:
        df.drop(columns=[col], inplace=True)
        print(f"Dropped column {col} with a single unique value.")
```

```
Dropped column  Bwd PSH Flags with a single unique value.
Dropped column  Bwd URG Flags with a single unique value.
Dropped column Fwd Avg Bytes/Bulk with a single unique value.
Dropped column  Fwd Avg Packets/Bulk with a single unique value.
Dropped column  Fwd Avg Bulk Rate with a single unique value.
Dropped column  Bwd Avg Bytes/Bulk with a single unique value.
Dropped column  Bwd Avg Packets/Bulk with a single unique value.
Dropped column Bwd Avg Bulk Rate with a single unique value.
```

This code identifies all numeric columns and all categorical (object-type) columns in the dataset, prints how many numeric features exist, and lists the names of all categorical features to prepare for later preprocessing steps.

[13]:
```
numeric_cols = df.select_dtypes(include=np.number).columns
print("\nNumber of numeric Columns:", len(numeric_cols))

categorical_cols = df.select_dtypes(include=['object']).columns
print("\nCategorical Columns:", list(categorical_cols))
```

```

Number of numeric Columns: 71
```

```
Categorical Columns: [' Label']
```

This code displays all unique values found in the ' PSH Flag Count' column to help understand the range or categories present in that feature.

[14]:
```python
df[' PSH Flag Count'].unique()
```

[14]:
```
array([0, 1])
```

Some columns contain integers, but are actually binary values 0 or 1. We will encode them as categorical variables.

[15]:
```python
# add columns containing only 0 and 1 to categorical columns
binary_cols = [col for col in numeric_cols if df[col].nunique() == 2]
print("\nBinary Columns (0/1):", binary_cols)
categorical_cols = binary_cols + ['source_file'] + categorical_cols.tolist()
print("\nUpdated number of Categorical Columns:", len(categorical_cols))
numeric_cols = [col for col in numeric_cols if col not in binary_cols]
print("\nUpdated number of Numeric Columns:", len(numeric_cols))
```

```
Binary Columns (0/1): ['Fwd PSH Flags', ' Fwd URG Flags', 'FIN Flag Count', '
SYN Flag Count', ' RST Flag Count', ' PSH Flag Count', ' ACK Flag Count', ' URG
Flag Count', ' CWE Flag Count', ' ECE Flag Count']

Updated number of Categorical Columns: 12

Updated number of Numeric Columns: 61
```

3.1.1 Handling missing values

[16]:
```python
# Handle infinite or very large values before scaling
print(f"Number of infinite values before cleaning: {np.isinf(df[numeric_cols]).
 ↪sum().sum()}")
df[numeric_cols] = df[numeric_cols].replace([np.inf, -np.inf], np.nan)


# Fill remaining missing values
for col in df.columns:
    if df[col].dtype == 'object':
        # categorical column, fill with most frequent value
        df[col] = df[col].fillna(df[col].mode()[0])
    else:
        # numerical column, fill with median value
        df[col] = df[col].fillna(df[col].median())
```

```
Number of infinite values before cleaning: 83
```

This code replaces infinite values in numeric features with NaN, then fills all remaining missing values using the most frequent value for categorical columns and the median for numerical ones to ensure the dataset contains no invalid or empty entries before scaling or modeling.

3.1.2 Normalizing or scaling numerical features

The code clips extreme numerical values to prevent overflow and applies Standard Scaling to standardize features (mean=0, std=1) for model training.

[17]:
```
# Clip extremely large values to avoid numerical overflow
df[numeric_cols] = df[numeric_cols].clip(lower=-1e10, upper=1e10)

scaler = StandardScaler()  # to get mean = 0, std = 1
df[numeric_cols] = scaler.fit_transform(df[numeric_cols])
```

3.1.3 Encoding categorical variables

This code uses LabelEncoder to convert each column in categorical_cols into numerical integers, ensuring string conversion first. It also prints the encoded classes for verification.

[18]:
```
le = LabelEncoder()
for col in categorical_cols:
    df[col] = le.fit_transform(df[col].astype(str))
    print(f"Encoded {col} with classes: {le.classes_}")
```

```
Encoded Fwd PSH Flags with classes: ['0' '1']
Encoded  Fwd URG Flags with classes: ['0' '1']
Encoded FIN Flag Count with classes: ['0' '1']
Encoded  SYN Flag Count with classes: ['0' '1']
Encoded  RST Flag Count with classes: ['0' '1']
Encoded  PSH Flag Count with classes: ['0' '1']
Encoded  ACK Flag Count with classes: ['0' '1']
Encoded  URG Flag Count with classes: ['0' '1']
Encoded  CWE Flag Count with classes: ['0' '1']
Encoded  ECE Flag Count with classes: ['0' '1']
Encoded source_file with classes: ['-0.0666197284514735' '-0.49235930188880367'
'-0.9180988753261339'
 '-1.343838448763464' '-1.7695780222007942' '0.3591198449858567'
 '0.7848594184231868' '1.2105989918605171']
Encoded  Label with classes: ['BENIGN' 'Bot' 'DDoS' 'DoS GoldenEye' 'DoS Hulk'
'DoS Slowhttptest'
 'DoS slowloris' 'FTP-Patator' 'PortScan' 'SSH-Patator'
 'Web Attack   Brute Force' 'Web Attack   XSS']
```

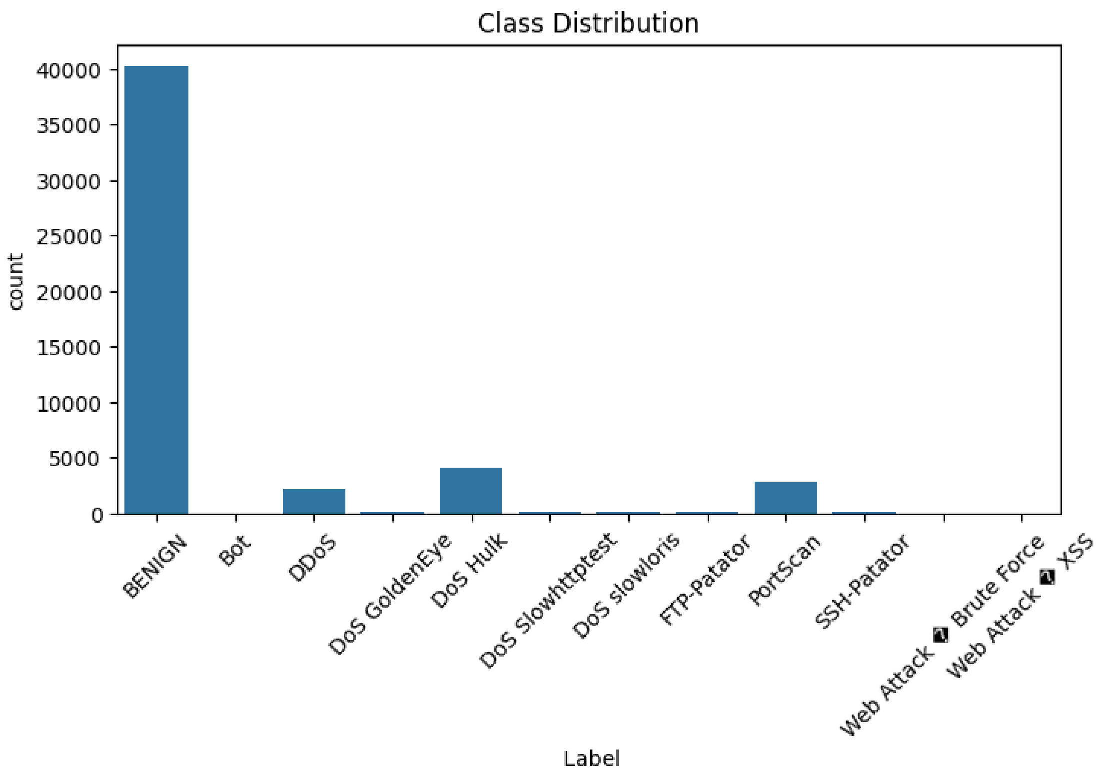
3.2 EDA - Exploratory Data Analysis

Conduct EDA, visualizing feature distributions and identifying potential relationships to guide feature engineering.

3.2.1 Visualize distibution of target variable

This block visualizes the distribution of the target variable (the 'Label' column) using a count plot. It retrieves the original class names using le.inverse_transform to correctly label the x-axis for improved readability of the class distribution.

[19]:
```
target_col = ' Label'
plt.figure(figsize=(8, 4))
sns.countplot(x=df[target_col])
plt.title("Class Distribution")
class_names = le.inverse_transform(sorted(df[target_col].unique()))
plt.xticks(ticks=range(len(class_names)), labels=class_names, rotation=45)
plt.show()
```



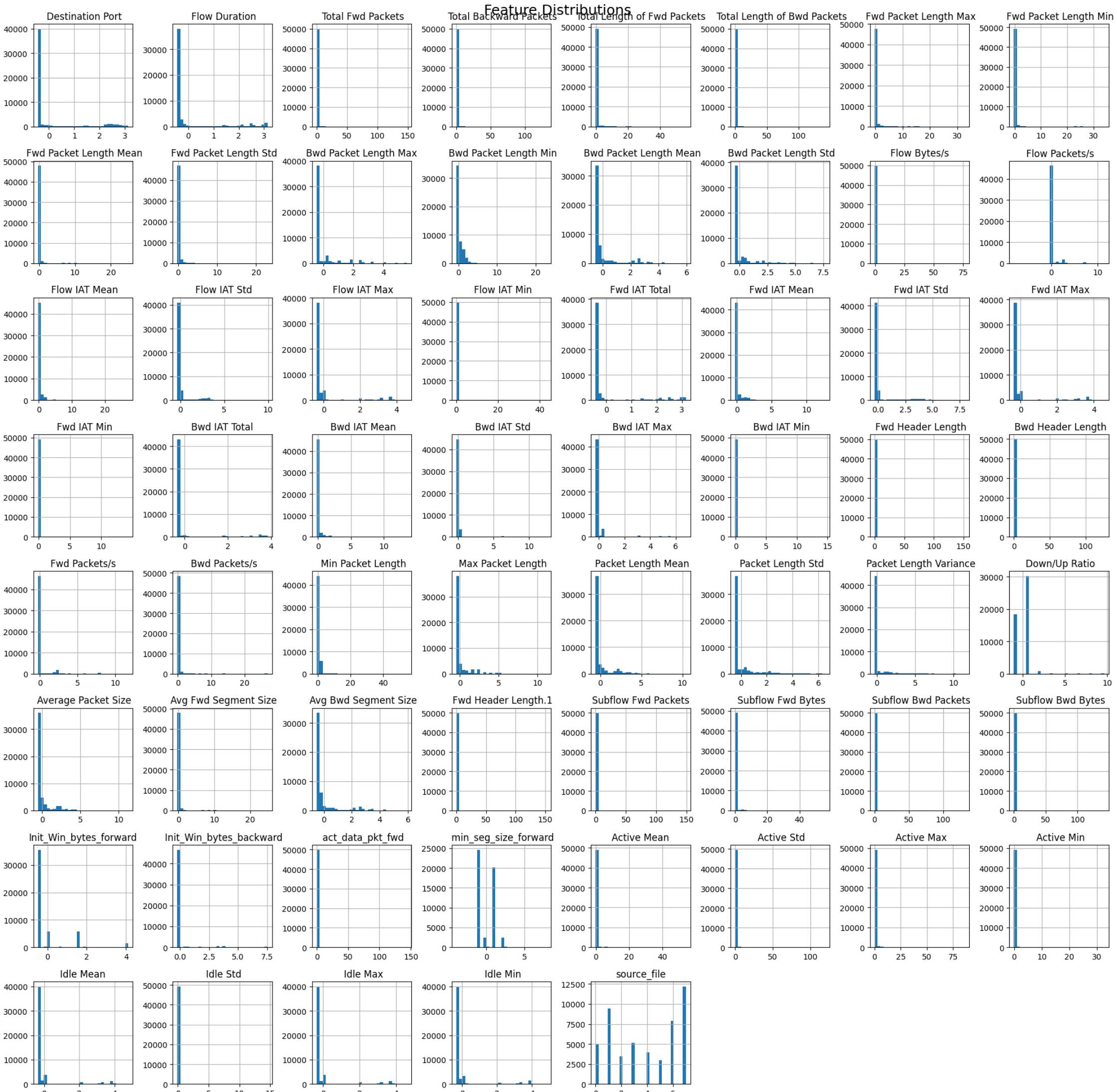
Begnign is the most frequent class. Frequent attacks are DDoS, DoS Hulk and PortScan.

3.2.2 Visualize feature distributions

This block generates histograms with 30 bins for all numerical features in numeric_cols across a large figure (20x20) to visualize their distributions, using tight_layout() to prevent overlap.

[20]:
```
df[numeric_cols].hist(bins=30, figsize=(20, 20))
plt.suptitle("Feature Distributions", fontsize=18)
plt.tight_layout()
```

```
plt.show()
```



3.2.3 Correlation analysis

This block calculates the correlation matrix and performs feature selection by dropping columns that have a near-zero absolute correlation (< 0.01) with the target 'Label'. It then prints the resulting DataFrame's head and information, and visualizes the full correlation matrix using a heatmap.texte

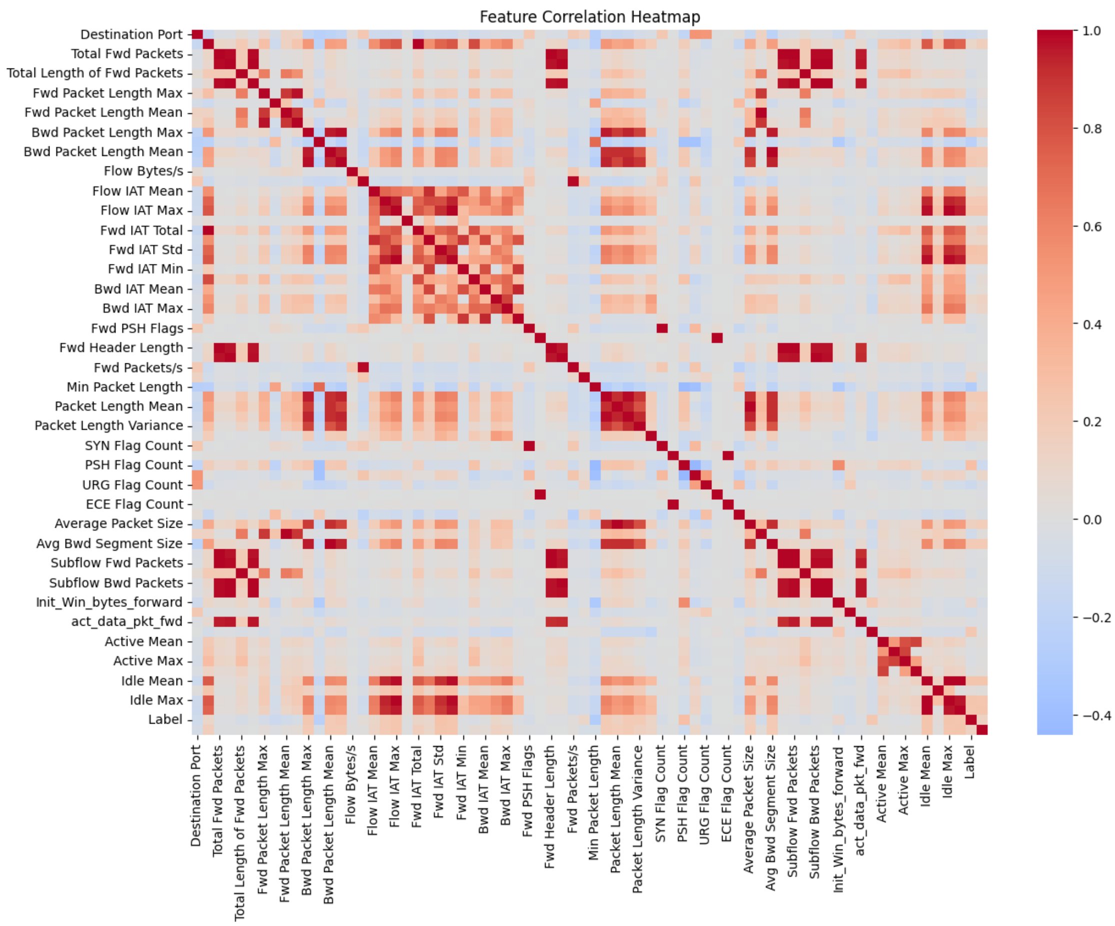
[21]:
```
corr = df.corr()
df = df.drop(corr[" Label"][abs(corr[" Label"])<0.01].index, axis=1)
print(df.head(), df.info())
plt.figure(figsize=(14, 10))
```

```python
sns.heatmap(corr, cmap="coolwarm", center=0)
plt.title("Feature Correlation Heatmap")
plt.show()
```

```
<class 'pandas.core.frame.DataFrame'>
Index: 50000 entries, 746827 to 2337454
Data columns (total 61 columns):
 #   Column                       Non-Null Count  Dtype
---  ------                       --------------  -----
 0    Destination Port            50000 non-null  float64
 1    Flow Duration               50000 non-null  float64
 2    Total Fwd Packets           50000 non-null  float64
 3    Total Backward Packets      50000 non-null  float64
 4   Total Length of Fwd Packets  50000 non-null  float64
 5    Fwd Packet Length Max       50000 non-null  float64
 6    Fwd Packet Length Min       50000 non-null  float64
 7    Fwd Packet Length Mean      50000 non-null  float64
 8    Fwd Packet Length Std       50000 non-null  float64
 9   Bwd Packet Length Max        50000 non-null  float64
 10   Bwd Packet Length Min       50000 non-null  float64
 11   Bwd Packet Length Mean      50000 non-null  float64
 12   Bwd Packet Length Std       50000 non-null  float64
 13  Flow Bytes/s                 50000 non-null  float64
 14   Flow Packets/s              50000 non-null  float64
 15   Flow IAT Mean               50000 non-null  float64
 16   Flow IAT Std                50000 non-null  float64
 17   Flow IAT Max                50000 non-null  float64
 18  Fwd IAT Total                50000 non-null  float64
 19   Fwd IAT Mean                50000 non-null  float64
 20   Fwd IAT Std                 50000 non-null  float64
 21   Fwd IAT Max                 50000 non-null  float64
 22   Fwd IAT Min                 50000 non-null  float64
 23  Bwd IAT Total                50000 non-null  float64
 24   Bwd IAT Std                 50000 non-null  float64
 25   Bwd IAT Max                 50000 non-null  float64
 26   Bwd IAT Min                 50000 non-null  float64
 27  Fwd PSH Flags                50000 non-null  int64
 28   Fwd Header Length           50000 non-null  float64
 29   Bwd Header Length           50000 non-null  float64
 30  Fwd Packets/s                50000 non-null  float64
 31   Bwd Packets/s               50000 non-null  float64
 32   Min Packet Length           50000 non-null  float64
 33   Max Packet Length           50000 non-null  float64
 34   Packet Length Mean          50000 non-null  float64
 35   Packet Length Std           50000 non-null  float64
 36   Packet Length Variance      50000 non-null  float64
 37  FIN Flag Count               50000 non-null  int64
 38   SYN Flag Count              50000 non-null  int64
```

```
 39   PSH Flag Count             50000 non-null  int64
 40   URG Flag Count             50000 non-null  int64
 41   Down/Up Ratio              50000 non-null  float64
 42   Average Packet Size        50000 non-null  float64
 43   Avg Fwd Segment Size       50000 non-null  float64
 44   Avg Bwd Segment Size       50000 non-null  float64
 45   Fwd Header Length.1        50000 non-null  float64
 46  Subflow Fwd Packets         50000 non-null  float64
 47   Subflow Fwd Bytes          50000 non-null  float64
 48   Subflow Bwd Packets        50000 non-null  float64
 49  Init_Win_bytes_forward      50000 non-null  float64
 50   Init_Win_bytes_backward    50000 non-null  float64
 51   act_data_pkt_fwd           50000 non-null  float64
 52   min_seg_size_forward       50000 non-null  float64
 53   Active Std                 50000 non-null  float64
 54   Active Max                 50000 non-null  float64
 55  Idle Mean                   50000 non-null  float64
 56   Idle Std                   50000 non-null  float64
 57   Idle Max                   50000 non-null  float64
 58   Idle Min                   50000 non-null  float64
 59   Label                      50000 non-null  int64
 60  source_file                 50000 non-null  int64
dtypes: float64(54), int64(7)
memory usage: 23.7 MB
          Destination Port   Flow Duration   Total Fwd Packets  \
746827            2.311152       -0.437290           -0.085136
946912           -0.439633       -0.436367           -0.069382
2216843          -0.438162        2.540471            0.040897
699389           -0.439633       -0.436376           -0.037873
1170268          -0.439633       -0.435839           -0.069382

          Total Backward Packets  Total Length of Fwd Packets  \
746827                 -0.069641                    -0.190478
946912                 -0.056575                    -0.165505
2216843                 0.008754                    -0.074063
699389                 -0.056575                    -0.139064
1170268                -0.056575                    -0.158160

          Fwd Packet Length Max   Fwd Packet Length Min  \
746827                -0.293080               -0.322393
946912                -0.245038                0.266778
2216843                0.154842               -0.322393
699389                -0.243625                0.284107
1170268               -0.230908                0.440064

          Fwd Packet Length Mean   Fwd Packet Length Std  \
746827                 -0.313089               -0.247698
946912                 -0.130540               -0.247698
```

```
2216843                -0.123978                0.133097
699389                 -0.125171               -0.247698
1170268                -0.076850               -0.247698

         Bwd Packet Length Max  …  act_data_pkt_fwd  min_seg_size_forward  \
746827               -0.447062  …         -0.069280              0.896237
946912               -0.348742  …         -0.048303              0.896237
2216843               2.550154  …         -0.048303              0.896237
699389               -0.402559  …         -0.006350              0.896237
1170268              -0.404112  …         -0.048303             -0.923312

          Active Std  Active Max  Idle Mean  Idle Std  Idle Max  Idle Min  \
746827     -0.103166   -0.151705  -0.349659  -0.107681  -0.354482  -0.336948
946912     -0.103166   -0.151705  -0.349659  -0.107681  -0.354482  -0.336948
2216843    -0.103166   -0.150718   3.901289  -0.107681   3.769324   3.961900
699389     -0.103166   -0.151705  -0.349659  -0.107681  -0.354482  -0.336948
1170268    -0.103166   -0.151705  -0.349659  -0.107681  -0.354482  -0.336948

          Label  source_file
746827        0            1
946912        0            1
2216843       4            7
699389        0            2
1170268       0            1

[5 rows x 61 columns] None
```



The entirely white rows and columns in the heatmap indicate features that contain only a single value, resulting in zero variance and no correlation with other features.

The heatmap analysis indicates strong correlations between some features (e.g., Total Fwd/Backward Packets), suggesting redundancy that dimensionality reduction (like PCA) could address. It also confirms that other features are strongly correlated with the target ('Label') (e.g., PSH Flag Count, Min Packet Length), validating their importance for the classification task.

3.2.4 Create train and test datasets, and balance the dataset with SMOTE

This block splits the data into training (80%) and testing (20%) sets using a fixed random state. It then addresses class imbalance by applying the SMOTE (Synthetic Minority Over-sampling Technique) algorithm to the training data (X_train, y_train), generating synthetic samples for minority classes to achieve a balanced distribution. The code prints the sizes of the sets and the class distributions before and after SMOTE

[22]:
```
X = df.drop(columns=[target_col])
y = df[target_col]
```

```python
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2,␣
 ↪random_state=42)
print("Training set size:", X_train.shape)
print("Test set size:", X_test.shape)


print("Class répartition before SMOTE :")
print(y_train.value_counts())

# Appliquer SMOTE pour équilibrer les classes sur les données d'entraînement
smote = SMOTE(sampling_strategy='auto', random_state=42)
X_train_smote, y_train_smote = smote.fit_resample(X_train, y_train)

print("\nClass répartition after SMOTE :")
print(pd.Series(y_train_smote).value_counts())
print(X_train.info(), df.info())
```

```
Training set size: (40000, 60)
Test set size: (10000, 60)
Class répartition before SMOTE :
 Label
0     32125
4      3266
8      2287
2      1785
3       142
7       107
6        86
9        77
5        75
1        23
10       16
11       11
Name: count, dtype: int64

Class répartition after SMOTE :
 Label
2     32125
0     32125
4     32125
8     32125
5     32125
7     32125
3     32125
9     32125
6     32125
1     32125
10    32125
```

```
11    32125
Name: count, dtype: int64
<class 'pandas.core.frame.DataFrame'>
Index: 40000 entries, 182468 to 895607
Data columns (total 60 columns):
 #   Column                       Non-Null Count  Dtype
---  ------                       --------------  -----
 0    Destination Port            40000 non-null  float64
 1    Flow Duration               40000 non-null  float64
 2    Total Fwd Packets           40000 non-null  float64
 3    Total Backward Packets      40000 non-null  float64
 4   Total Length of Fwd Packets  40000 non-null  float64
 5    Fwd Packet Length Max       40000 non-null  float64
 6    Fwd Packet Length Min       40000 non-null  float64
 7    Fwd Packet Length Mean      40000 non-null  float64
 8    Fwd Packet Length Std       40000 non-null  float64
 9   Bwd Packet Length Max        40000 non-null  float64
 10   Bwd Packet Length Min       40000 non-null  float64
 11   Bwd Packet Length Mean      40000 non-null  float64
 12   Bwd Packet Length Std       40000 non-null  float64
 13  Flow Bytes/s                 40000 non-null  float64
 14   Flow Packets/s              40000 non-null  float64
 15   Flow IAT Mean               40000 non-null  float64
 16   Flow IAT Std                40000 non-null  float64
 17   Flow IAT Max                40000 non-null  float64
 18  Fwd IAT Total                40000 non-null  float64
 19   Fwd IAT Mean                40000 non-null  float64
 20   Fwd IAT Std                 40000 non-null  float64
 21   Fwd IAT Max                 40000 non-null  float64
 22   Fwd IAT Min                 40000 non-null  float64
 23  Bwd IAT Total                40000 non-null  float64
 24   Bwd IAT Std                 40000 non-null  float64
 25   Bwd IAT Max                 40000 non-null  float64
 26   Bwd IAT Min                 40000 non-null  float64
 27  Fwd PSH Flags                40000 non-null  int64
 28   Fwd Header Length           40000 non-null  float64
 29   Bwd Header Length           40000 non-null  float64
 30  Fwd Packets/s                40000 non-null  float64
 31   Bwd Packets/s               40000 non-null  float64
 32   Min Packet Length           40000 non-null  float64
 33   Max Packet Length           40000 non-null  float64
 34   Packet Length Mean          40000 non-null  float64
 35   Packet Length Std           40000 non-null  float64
 36   Packet Length Variance      40000 non-null  float64
 37  FIN Flag Count               40000 non-null  int64
 38   SYN Flag Count              40000 non-null  int64
 39   PSH Flag Count              40000 non-null  int64
 40   URG Flag Count              40000 non-null  int64
```

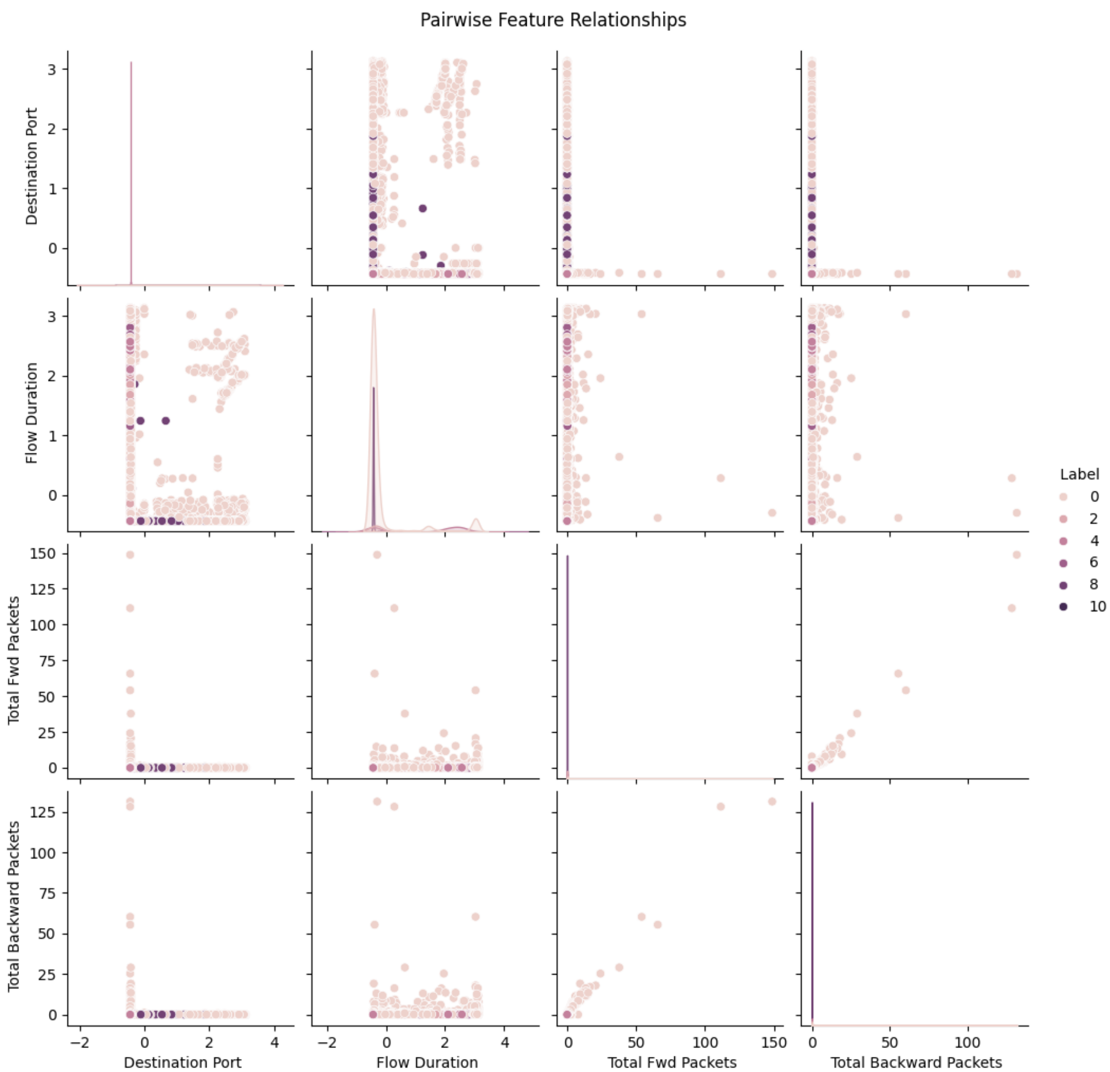
```
 41   Down/Up Ratio                40000 non-null  float64
 42   Average Packet Size          40000 non-null  float64
 43   Avg Fwd Segment Size         40000 non-null  float64
 44   Avg Bwd Segment Size         40000 non-null  float64
 45   Fwd Header Length.1          40000 non-null  float64
 46  Subflow Fwd Packets           40000 non-null  float64
 47   Subflow Fwd Bytes            40000 non-null  float64
 48   Subflow Bwd Packets          40000 non-null  float64
 49  Init_Win_bytes_forward        40000 non-null  float64
 50   Init_Win_bytes_backward      40000 non-null  float64
 51   act_data_pkt_fwd             40000 non-null  float64
 52   min_seg_size_forward         40000 non-null  float64
 53   Active Std                   40000 non-null  float64
 54   Active Max                   40000 non-null  float64
 55  Idle Mean                     40000 non-null  float64
 56   Idle Std                     40000 non-null  float64
 57   Idle Max                     40000 non-null  float64
 58   Idle Min                     40000 non-null  float64
 59  source_file                   40000 non-null  int64
dtypes: float64(54), int64(6)
memory usage: 18.6 MB
<class 'pandas.core.frame.DataFrame'>
Index: 50000 entries, 746827 to 2337454
Data columns (total 61 columns):
 #   Column                        Non-Null Count  Dtype
---  ------                        --------------  -----
 0    Destination Port             50000 non-null  float64
 1    Flow Duration                50000 non-null  float64
 2    Total Fwd Packets            50000 non-null  float64
 3    Total Backward Packets       50000 non-null  float64
 4   Total Length of Fwd Packets   50000 non-null  float64
 5    Fwd Packet Length Max        50000 non-null  float64
 6    Fwd Packet Length Min        50000 non-null  float64
 7    Fwd Packet Length Mean       50000 non-null  float64
 8    Fwd Packet Length Std        50000 non-null  float64
 9   Bwd Packet Length Max         50000 non-null  float64
 10   Bwd Packet Length Min        50000 non-null  float64
 11   Bwd Packet Length Mean       50000 non-null  float64
 12   Bwd Packet Length Std        50000 non-null  float64
 13  Flow Bytes/s                  50000 non-null  float64
 14   Flow Packets/s               50000 non-null  float64
 15   Flow IAT Mean                50000 non-null  float64
 16   Flow IAT Std                 50000 non-null  float64
 17   Flow IAT Max                 50000 non-null  float64
 18  Fwd IAT Total                 50000 non-null  float64
 19   Fwd IAT Mean                 50000 non-null  float64
 20   Fwd IAT Std                  50000 non-null  float64
 21   Fwd IAT Max                  50000 non-null  float64
```

```
 22   Fwd IAT Min                  50000 non-null  float64
 23  Bwd IAT Total                 50000 non-null  float64
 24   Bwd IAT Std                  50000 non-null  float64
 25   Bwd IAT Max                  50000 non-null  float64
 26   Bwd IAT Min                  50000 non-null  float64
 27  Fwd PSH Flags                 50000 non-null  int64
 28   Fwd Header Length            50000 non-null  float64
 29   Bwd Header Length            50000 non-null  float64
 30  Fwd Packets/s                 50000 non-null  float64
 31   Bwd Packets/s                50000 non-null  float64
 32   Min Packet Length            50000 non-null  float64
 33   Max Packet Length            50000 non-null  float64
 34   Packet Length Mean           50000 non-null  float64
 35   Packet Length Std            50000 non-null  float64
 36   Packet Length Variance       50000 non-null  float64
 37  FIN Flag Count                50000 non-null  int64
 38   SYN Flag Count               50000 non-null  int64
 39   PSH Flag Count               50000 non-null  int64
 40   URG Flag Count               50000 non-null  int64
 41   Down/Up Ratio                50000 non-null  float64
 42   Average Packet Size          50000 non-null  float64
 43   Avg Fwd Segment Size         50000 non-null  float64
 44   Avg Bwd Segment Size         50000 non-null  float64
 45   Fwd Header Length.1          50000 non-null  float64
 46  Subflow Fwd Packets           50000 non-null  float64
 47   Subflow Fwd Bytes            50000 non-null  float64
 48   Subflow Bwd Packets          50000 non-null  float64
 49  Init_Win_bytes_forward        50000 non-null  float64
 50   Init_Win_bytes_backward      50000 non-null  float64
 51   act_data_pkt_fwd             50000 non-null  float64
 52   min_seg_size_forward         50000 non-null  float64
 53   Active Std                   50000 non-null  float64
 54   Active Max                   50000 non-null  float64
 55  Idle Mean                     50000 non-null  float64
 56   Idle Std                     50000 non-null  float64
 57   Idle Max                     50000 non-null  float64
 58   Idle Min                     50000 non-null  float64
 59   Label                        50000 non-null  int64
 60  source_file                   50000 non-null  int64
dtypes: float64(54), int64(7)
memory usage: 23.7 MB
None None
```

3.2.5 Pairwise feature relationships

This block splits the data into training (80%) and testing (20%) sets using a fixed random state. It then addresses class imbalance by applying the SMOTE (Synthetic Minority Over-sampling Technique) algorithm to the training data (X_train, y_train), generating synthetic samples for minority classes to achieve a balanced distribution. The code prints the sizes of the sets and the class distributions before and after SMOTE application.texte

[23]:
```
sample_features = list(numeric_cols[:4])  # visualize only a few to keep plots␣
 ↪readable and time reasonable
plot_features = sample_features + [target_col]
sns.pairplot(df[plot_features], hue=target_col, diag_kind="kde")
plt.suptitle("Pairwise Feature Relationships", y=1.02)
plt.show()
```



4 2. Model Training, Comparison, and Ensemble

4.0.1 2.1 Logistic regression

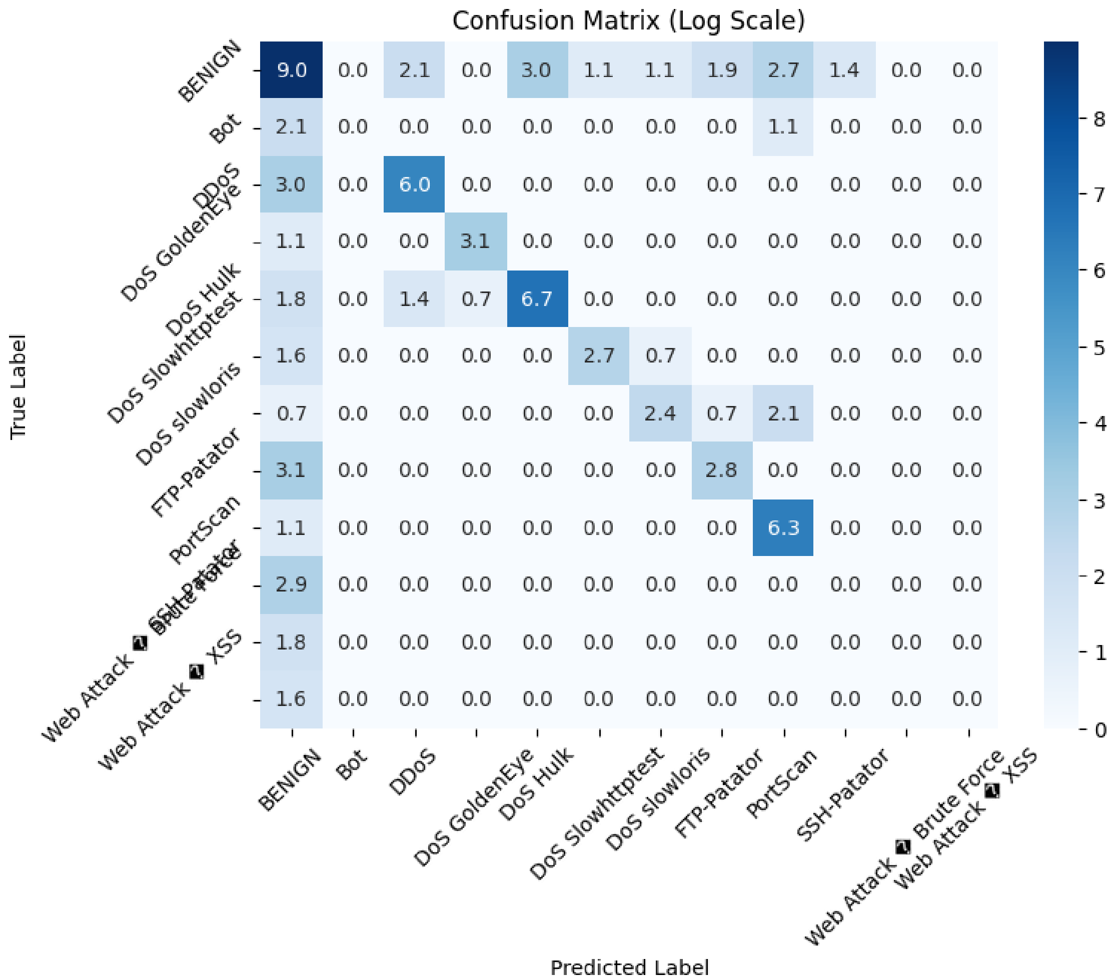
On the original train set, before SMOTE:

This block initializes and trains a Logistic Regression model on the training data. It then evaluates the model's performance on the test set, printing the raw confusion matrix and a full classification report. Finally, it visualizes the confusion matrix using a heatmap with a logarithmic scale (via np.log1p) for better visualization when classes are imbalanced.

[24]:
```
logistic_regression = LogisticRegression()
logistic_regression.fit(X_train, y_train)
models.append({"model" : logistic_regression, "name": "log_reg", "library":␣
 ↪"sklearn"})

### Evaluate the model
y_pred = logistic_regression.predict(X_test)
conf_matrix = confusion_matrix(y_test, y_pred)
print("Confusion matrix: ", conf_matrix)
print(classification_report(y_test, y_pred))

plt.figure(figsize=(8, 6))
# Use log scale for better visualization when one class is very frequent
conf_matrix_log = np.log1p(conf_matrix)  # log(1 + x) to handle zeros
sns.heatmap(conf_matrix_log, annot=True, fmt='.1f', cmap='Blues')
plt.title("Confusion Matrix (Log Scale)")
plt.xlabel("Predicted Label")
plt.xticks(ticks=np.arange(len(class_names))+0.5, labels=class_names,␣
 ↪rotation=45)
plt.ylabel("True Label")
plt.yticks(ticks=np.arange(len(class_names))+0.5, labels=class_names,␣
 ↪rotation=45)
plt.show()
```

```
c:\Users\arthu\OneDrive\Efrei\I1\S5\MachineLearning\machineLearningCybersecurity
\Lib\site-packages\sklearn\linear_model\_logistic.py:473: ConvergenceWarning:
lbfgs failed to converge after 100 iteration(s) (status=1):
STOP: TOTAL NO. OF ITERATIONS REACHED LIMIT

Increase the number of iterations to improve the convergence (max_iter=100).
You might also want to scale the data as shown in:
    https://scikit-learn.org/stable/modules/preprocessing.html
Please also refer to the documentation for alternative solver options:
    https://scikit-learn.org/stable/modules/linear_model.html#logistic-
regression
  n_iter_i = _check_optimize_result(
c:\Users\arthu\OneDrive\Efrei\I1\S5\MachineLearning\machineLearningCybersecurity
\Lib\site-packages\sklearn\metrics\_classification.py:1731:
```

```
UndefinedMetricWarning: Precision is ill-defined and being set to 0.0 in labels
with no predicted samples. Use `zero_division` parameter to control this
behavior.
  _warn_prf(average, modifier, f"{metric.capitalize()} is", result.shape[0])
c:\Users\arthu\OneDrive\Efrei\I1\S5\MachineLearning\machineLearningCybersecurity
\Lib\site-packages\sklearn\metrics\_classification.py:1731:
UndefinedMetricWarning: Precision is ill-defined and being set to 0.0 in labels
with no predicted samples. Use `zero_division` parameter to control this
behavior.
  _warn_prf(average, modifier, f"{metric.capitalize()} is", result.shape[0])
c:\Users\arthu\OneDrive\Efrei\I1\S5\MachineLearning\machineLearningCybersecurity
\Lib\site-packages\sklearn\metrics\_classification.py:1731:
UndefinedMetricWarning: Precision is ill-defined and being set to 0.0 in labels
with no predicted samples. Use `zero_division` parameter to control this
behavior.
  _warn_prf(average, modifier, f"{metric.capitalize()} is", result.shape[0])

Confusion matrix:  [[8014    0    7    0   20    2    2    6   14    3    0
0]
 [   7    0    0    0    0    0    0    0    2    0    0    0]
 [  20    0  422    0    0    0    0    0    0    0    0    0]
 [   2    0    0   22    0    0    0    0    0    0    0    0]
 [   5    0    3    1  804    0    0    0    0    0    0    0]
 [   4    0    0    0    0   14    1    0    0    0    0    0]
 [   1    0    0    0    0    0   10    1    7    0    0    0]
 [  21    0    0    0    0    0    0   16    0    0    0    0]
 [   2    0    0    0    0    0    0    0  541    0    0    0]
 [  17    0    0    0    0    0    0    0    0    0    0    0]
 [   5    0    0    0    0    0    0    0    0    0    0    0]
 [   4    0    0    0    0    0    0    0    0    0    0    0]]
              precision    recall  f1-score   support

           0       0.99      0.99      0.99      8068
           1       0.00      0.00      0.00         9
           2       0.98      0.95      0.97       442
           3       0.96      0.92      0.94        24
           4       0.98      0.99      0.98       813
           5       0.88      0.74      0.80        19
           6       0.77      0.53      0.62        19
           7       0.70      0.43      0.53        37
           8       0.96      1.00      0.98       543
           9       0.00      0.00      0.00        17
          10       0.00      0.00      0.00         5
          11       0.00      0.00      0.00         4

    accuracy                           0.98     10000
   macro avg       0.60      0.55      0.57     10000
weighted avg       0.98      0.98      0.98     10000
```



A lot of attacks are not detected by logistic regression, especially minority classes. The model tends to predict the majority class "Benign" too often. Still, we get decent f1-scores for some classes like DDoS and DoS Hulk.

On the SMOTE balanced train set:

This block trains a second Logistic Regression model, increasing max_iter to 1000 for convergence, but crucially, using the balanced training data (X_train_smote, y_train_smote) generated by SMOTE. The model is then evaluated on the original, imbalanced test set, with its performance metrics (confusion matrix and classification report) and log-scale heatmap visualized.

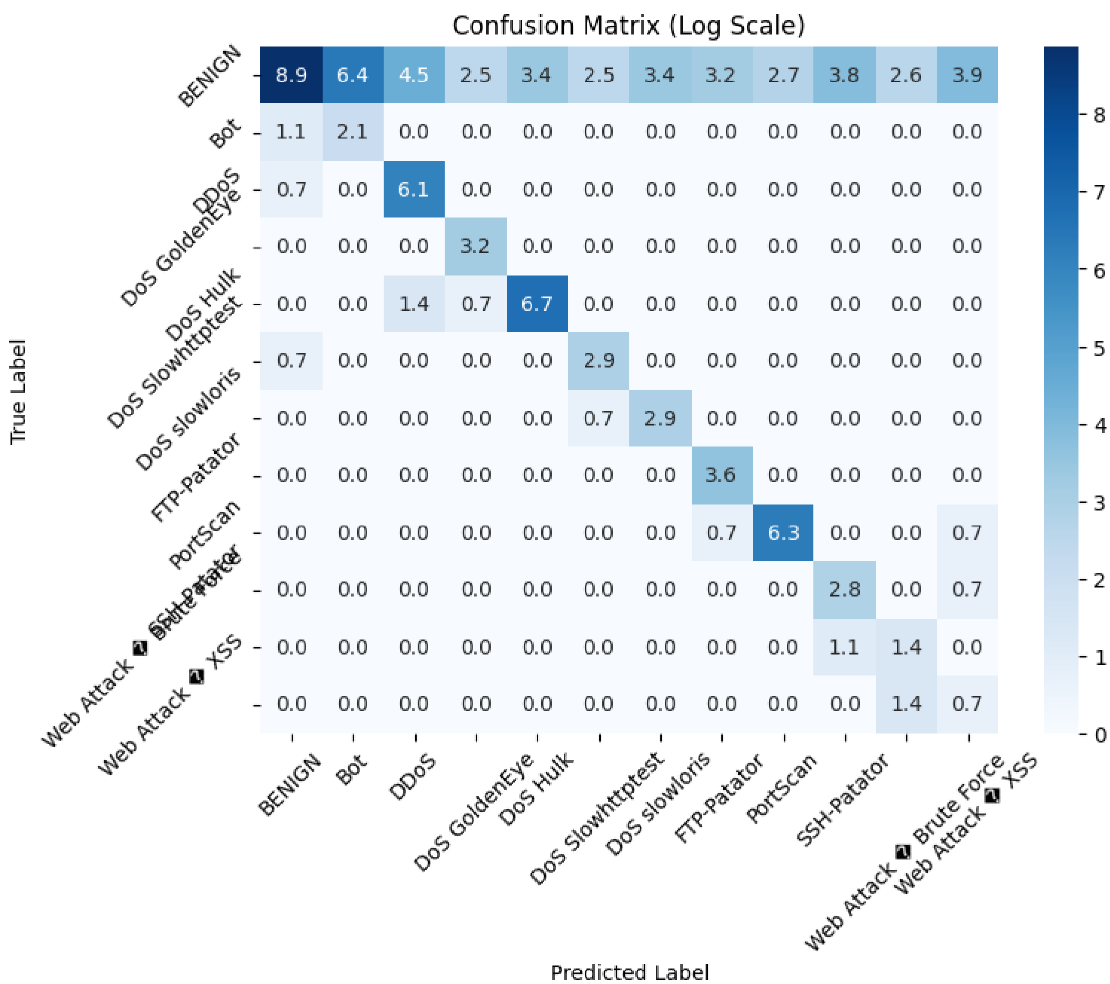
[25]:
```
logistic_regression_smote = LogisticRegression(max_iter=1000)
logistic_regression_smote.fit(X_train_smote, y_train_smote)
models.append({"model" : logistic_regression_smote, "name": "log_reg_smote",␣
 ↪"library": "sklearn"})
```

```python
### Evaluate the model
y_pred = logistic_regression_smote.predict(X_test)
conf_matrix = confusion_matrix(y_test, y_pred)
print("Confusion matrix: ", conf_matrix)
print(classification_report(y_test, y_pred))

plt.figure(figsize=(8, 6))
# Use log scale for better visualization when one class is very frequent
conf_matrix_log = np.log1p(conf_matrix)  # log(1 + x) to handle zeros
sns.heatmap(conf_matrix_log, annot=True, fmt='.1f', cmap='Blues')
plt.title("Confusion Matrix (Log Scale)")
plt.xlabel("Predicted Label")
plt.xticks(ticks=np.arange(len(class_names))+0.5, labels=class_names,␣
 ↪rotation=45)
plt.ylabel("True Label")
plt.yticks(ticks=np.arange(len(class_names))+0.5, labels=class_names,␣
 ↪rotation=45)
plt.show()
```

```
c:\Users\arthu\OneDrive\Efrei\I1\S5\MachineLearning\machineLearningCybersecurity
\Lib\site-packages\sklearn\linear_model\_logistic.py:473: ConvergenceWarning:
lbfgs failed to converge after 1000 iteration(s) (status=1):
STOP: TOTAL NO. OF ITERATIONS REACHED LIMIT

Increase the number of iterations to improve the convergence (max_iter=1000).
You might also want to scale the data as shown in:
    https://scikit-learn.org/stable/modules/preprocessing.html
Please also refer to the documentation for alternative solver options:
    https://scikit-learn.org/stable/modules/linear_model.html#logistic-
regression
  n_iter_i = _check_optimize_result(
```

```
Confusion matrix:  [[7158  599   86   11   29   11   29   24   14   45   12
50]
 [   2    7    0    0    0    0    0    0    0    0    0    0]
 [   1    0  441    0    0    0    0    0    0    0    0    0]
 [   0    0    0   24    0    0    0    0    0    0    0    0]
 [   0    0    3    1  809    0    0    0    0    0    0    0]
 [   1    0    0    0    0   18    0    0    0    0    0    0]
 [   0    0    0    0    0    1   18    0    0    0    0    0]
 [   0    0    0    0    0    0    0   37    0    0    0    0]
 [   0    0    0    0    0    0    0    1  541    0    0    1]
 [   0    0    0    0    0    0    0    0    0   16    0    1]
 [   0    0    0    0    0    0    0    0    0    2    3    0]
 [   0    0    0    0    0    0    0    0    0    0    3    1]]
              precision    recall  f1-score   support

           0       1.00      0.89      0.94      8068
```

1	0.01	0.78	0.02	9
2	0.83	1.00	0.91	442
3	0.67	1.00	0.80	24
4	0.97	1.00	0.98	813
5	0.60	0.95	0.73	19
6	0.38	0.95	0.55	19
7	0.60	1.00	0.75	37
8	0.97	1.00	0.99	543
9	0.25	0.94	0.40	17
10	0.17	0.60	0.26	5
11	0.02	0.25	0.04	4
accuracy			0.91	10000
macro avg	0.54	0.86	0.61	10000
weighted avg	0.98	0.91	0.94	10000



4.1 2.2 KNN implementation :

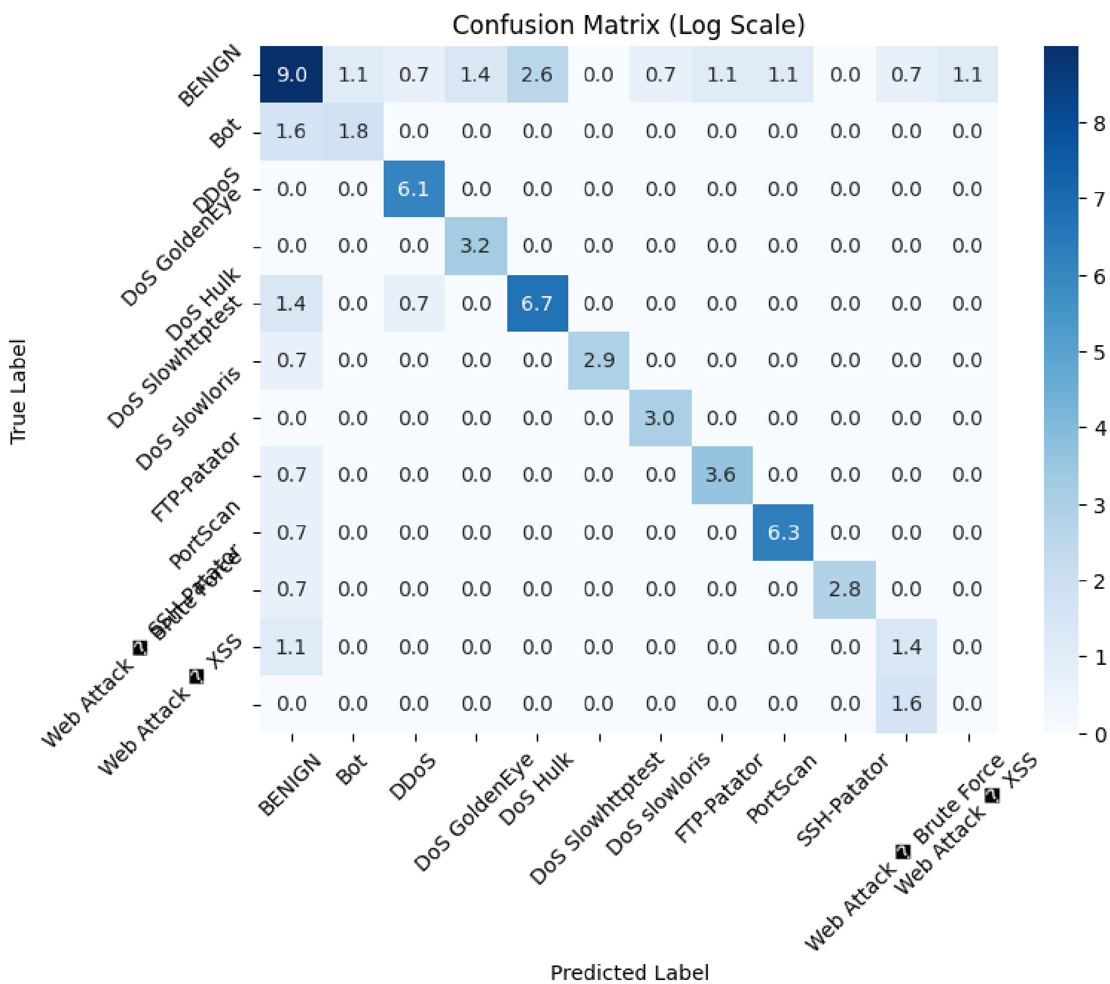
On the original train set, before SMOTE:

This block trains a K-Nearest Neighbors (KNN) classifier with $n = 5$ neighbors and 'distance' weighting on the original training data. It evaluates the model on the test set, displaying the confusion matrix, the classification report, and a log-scale heatmap visualization of the results.

[26]:
```
knn = KNeighborsClassifier(n_neighbors=5, weights='distance')
knn.fit(X_train, y_train)
models.append({"model" : knn, "name": "knn", "library": "sklearn"})

y_pred = knn.predict(X_test)
conf_matrix = confusion_matrix(y_test, y_pred)
print("Confusion matrix: ", conf_matrix)
print(classification_report(y_test, y_pred))


plt.figure(figsize=(8, 6))
# Use log scale for better visualization when one class is very frequent
conf_matrix_log = np.log1p(conf_matrix)  # log(1 + x) to handle zeros
sns.heatmap(conf_matrix_log, annot=True, fmt='.1f', cmap='Blues')
plt.title("Confusion Matrix (Log Scale)")
plt.xlabel("Predicted Label")
plt.xticks(ticks=np.arange(len(class_names))+0.5, labels=class_names,␣
    ↪rotation=45)
plt.ylabel("True Label")
plt.yticks(ticks=np.arange(len(class_names))+0.5, labels=class_names,␣
    ↪rotation=45)
plt.show()
```

```
Confusion matrix:  [[8042    2    1    3   12    0    1    2    2    0    1
2]
 [   4    5    0    0    0    0    0    0    0    0    0    0]
 [   0    0  442    0    0    0    0    0    0    0    0    0]
 [   0    0    0   24    0    0    0    0    0    0    0    0]
 [   3    0    1    0  809    0    0    0    0    0    0    0]
 [   1    0    0    0    0   18    0    0    0    0    0    0]
 [   0    0    0    0    0    0   19    0    0    0    0    0]
 [   1    0    0    0    0    0    0   36    0    0    0    0]
 [   1    0    0    0    0    0    0    0  542    0    0    0]
 [   1    0    0    0    0    0    0    0    0   16    0    0]
 [   2    0    0    0    0    0    0    0    0    0    3    0]
 [   0    0    0    0    0    0    0    0    0    0    4    0]]
              precision    recall  f1-score   support

           0       1.00      1.00      1.00      8068
           1       0.71      0.56      0.62         9
           2       1.00      1.00      1.00       442
```

```
           3       0.89      1.00      0.94        24
           4       0.99      1.00      0.99       813
           5       1.00      0.95      0.97        19
           6       0.95      1.00      0.97        19
           7       0.95      0.97      0.96        37
           8       1.00      1.00      1.00       543
           9       1.00      0.94      0.97        17
          10       0.38      0.60      0.46         5
          11       0.00      0.00      0.00         4

    accuracy                           1.00     10000
   macro avg       0.82      0.83      0.82     10000
weighted avg       1.00      1.00      1.00     10000
```



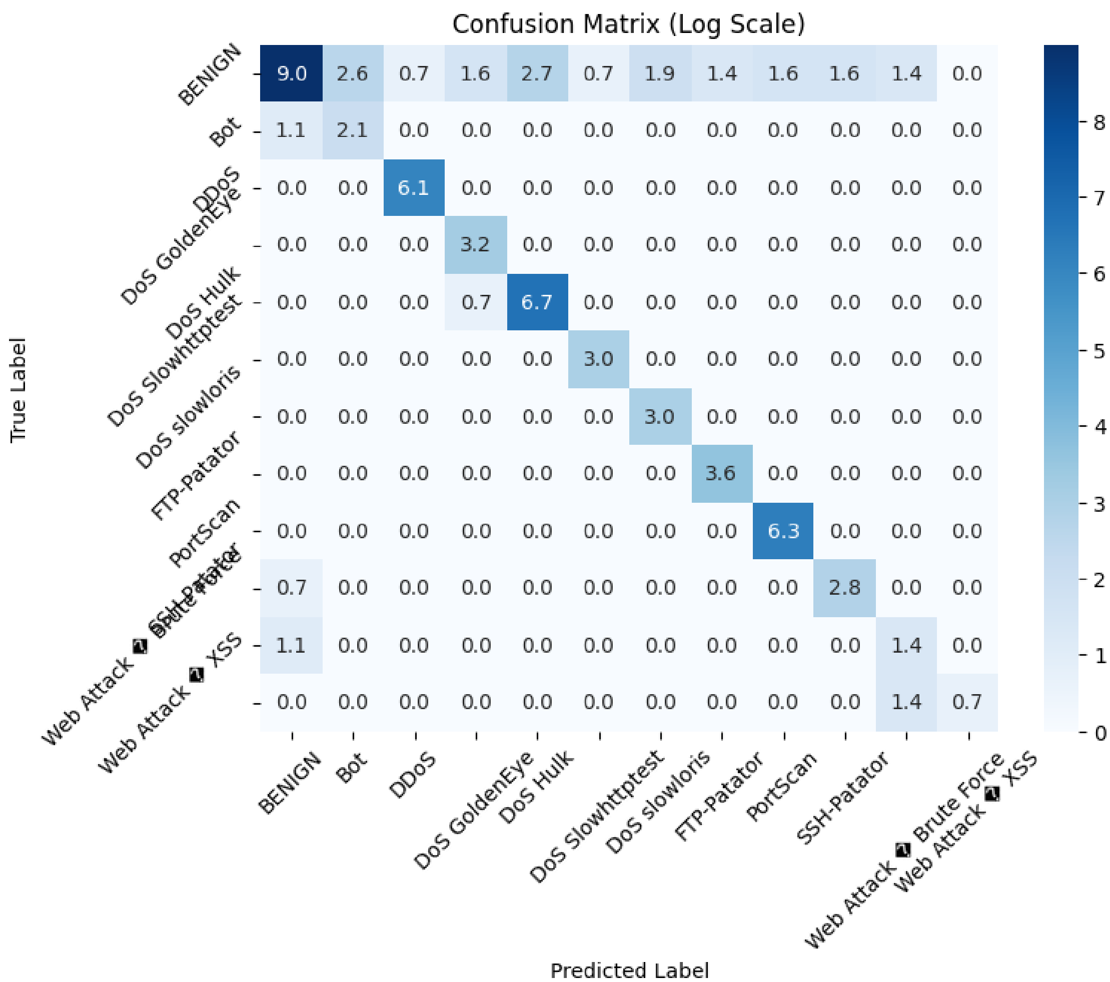
On the SMOTE balanced train set:

This block trains a second KNN model with the same hyperparameters but uses the balanced, SMOTE-resampled training data (X_train_smote, y_train_smote). It evaluates performance on the original test set, displaying the classification report and visualizing the log-scale confusion matrix heatmap for comparison.

[27]:
```
knn_smote = KNeighborsClassifier(n_neighbors=5, weights='distance')
knn_smote.fit(X_train_smote, y_train_smote)
models.append({"model" : knn_smote, "name": "knn_smote", "library": "sklearn"})
y_pred = knn_smote.predict(X_test)
conf_matrix = confusion_matrix(y_test, y_pred)
print("Confusion matrix: ", conf_matrix)
print(classification_report(y_test, y_pred))


plt.figure(figsize=(8, 6))
# Use log scale for better visualization when one class is very frequent
conf_matrix_log = np.log1p(conf_matrix)  # log(1 + x) to handle zeros
sns.heatmap(conf_matrix_log, annot=True, fmt='.1f', cmap='Blues')
plt.title("Confusion Matrix (Log Scale)")
plt.xlabel("Predicted Label")
plt.xticks(ticks=np.arange(len(class_names))+0.5, labels=class_names,␣
 ↪rotation=45)
plt.ylabel("True Label")
plt.yticks(ticks=np.arange(len(class_names))+0.5, labels=class_names,␣
 ↪rotation=45)
plt.show()
```

```
Confusion matrix:  [[8016   12    1    4   14    1    6    3    4    4    3
0]
 [   2    7    0    0    0    0    0    0    0    0    0    0]
 [   0    0  442    0    0    0    0    0    0    0    0    0]
 [   0    0    0   24    0    0    0    0    0    0    0    0]
 [   0    0    0    1  812    0    0    0    0    0    0    0]
 [   0    0    0    0    0   19    0    0    0    0    0    0]
 [   0    0    0    0    0    0   19    0    0    0    0    0]
 [   0    0    0    0    0    0    0   37    0    0    0    0]
 [   0    0    0    0    0    0    0    0  543    0    0    0]
 [   1    0    0    0    0    0    0    0    0   16    0    0]
 [   2    0    0    0    0    0    0    0    0    0    3    0]
 [   0    0    0    0    0    0    0    0    0    0    3    1]]
              precision    recall  f1-score   support

           0       1.00      0.99      1.00      8068
           1       0.37      0.78      0.50         9
           2       1.00      1.00      1.00       442
           3       0.83      1.00      0.91        24
           4       0.98      1.00      0.99       813
           5       0.95      1.00      0.97        19
           6       0.76      1.00      0.86        19
```

```
           7       0.93      1.00      0.96        37
           8       0.99      1.00      1.00       543
           9       0.80      0.94      0.86        17
          10       0.33      0.60      0.43         5
          11       1.00      0.25      0.40         4

    accuracy                           0.99     10000
   macro avg       0.83      0.88      0.82     10000
weighted avg       1.00      0.99      0.99     10000
```



4.2 2.4 Random Forest Classification

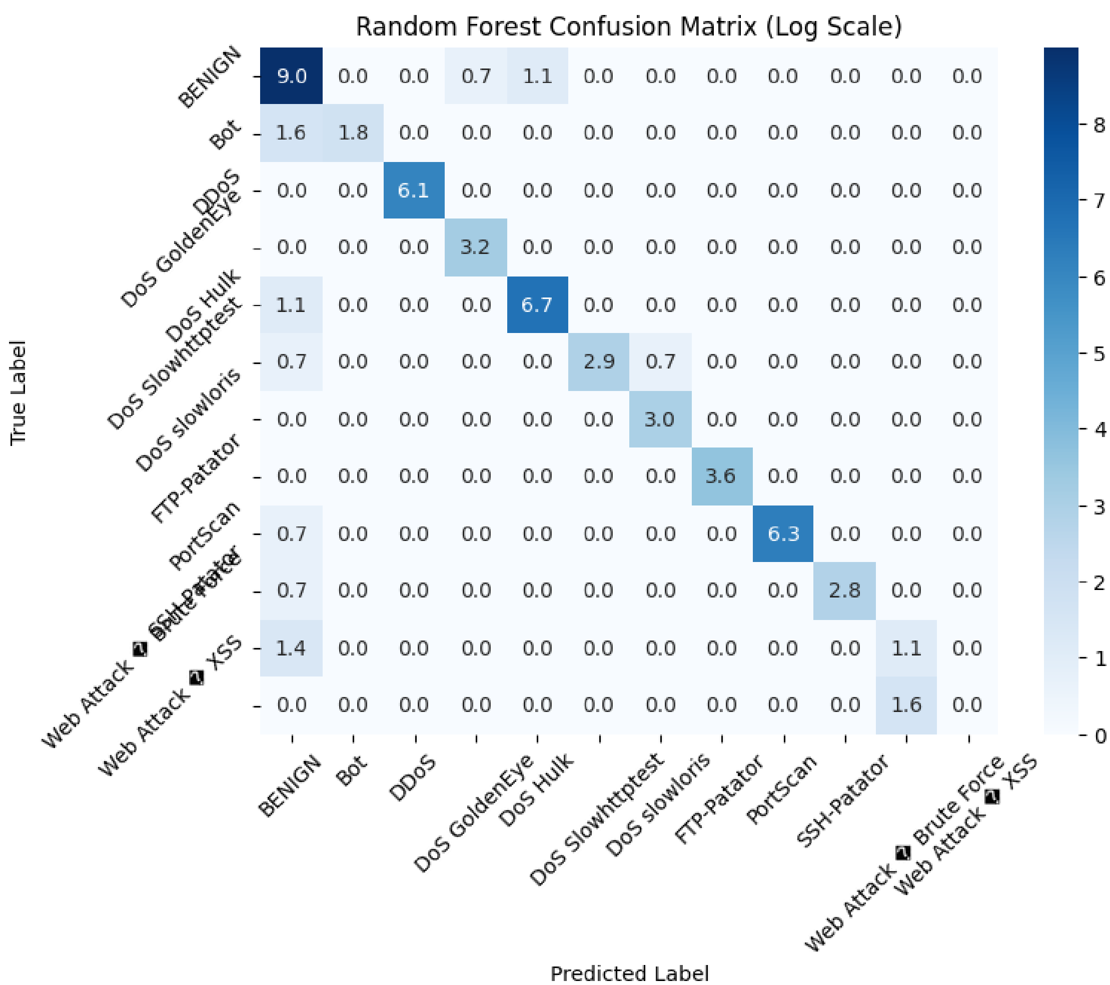
On the original train set, before SMOTE:

This block trains a Random Forest classifier with 100 estimators, using the class_weight='balanced' setting to automatically adjust weights inversely proportional to class frequencies to handle imbalance. It then evaluates the model's performance on the test set, printing the results and visualizing the log-scale confusion matrix.texte

[28]:
```
random_forest = RandomForestClassifier(n_estimators=100, random_state=42,␣
 ↪class_weight='balanced')
random_forest.fit(X_train, y_train)
models.append({"model" : random_forest, "name": "rand_forest", "library":␣
 ↪"sklearn"})
y_pred_rf = random_forest.predict(X_test)

conf_matrix = confusion_matrix(y_test, y_pred_rf)
print("Random Forest Results:")
print("Confusion matrix: ", conf_matrix)
print(classification_report(y_test, y_pred_rf))

plt.figure(figsize=(8, 6))
conf_matrix_log = np.log1p(conf_matrix)
sns.heatmap(conf_matrix_log, annot=True, fmt='.1f', cmap='Blues')
plt.title("Random Forest Confusion Matrix (Log Scale)")
plt.xlabel("Predicted Label")
plt.xticks(ticks=np.arange(len(class_names))+0.5, labels=class_names,␣
 ↪rotation=45)
plt.ylabel("True Label")
plt.yticks(ticks=np.arange(len(class_names))+0.5, labels=class_names,␣
 ↪rotation=45)
plt.show()
```

```
Random Forest Results:
Confusion matrix:  [[8065    0    0    1    2    0    0    0    0    0    0
0]
 [   4    5    0    0    0    0    0    0    0    0    0    0]
 [   0    0  442    0    0    0    0    0    0    0    0    0]
 [   0    0    0   24    0    0    0    0    0    0    0    0]
 [   2    0    0    0  811    0    0    0    0    0    0    0]
 [   1    0    0    0    0   17    1    0    0    0    0    0]
 [   0    0    0    0    0    0   19    0    0    0    0    0]
 [   0    0    0    0    0    0    0   37    0    0    0    0]
 [   1    0    0    0    0    0    0    0  542    0    0    0]
 [   1    0    0    0    0    0    0    0    0   16    0    0]
 [   3    0    0    0    0    0    0    0    0    0    2    0]
 [   0    0    0    0    0    0    0    0    0    0    4    0]]
              precision    recall  f1-score   support

           0       1.00      1.00      1.00      8068
           1       1.00      0.56      0.71         9
           2       1.00      1.00      1.00       442
           3       0.96      1.00      0.98        24
           4       1.00      1.00      1.00       813
```

```
           5       1.00      0.89      0.94        19
           6       0.95      1.00      0.97        19
           7       1.00      1.00      1.00        37
           8       1.00      1.00      1.00       543
           9       1.00      0.94      0.97        17
          10       0.33      0.40      0.36         5
          11       0.00      0.00      0.00         4

    accuracy                           1.00     10000
   macro avg       0.85      0.82      0.83     10000
weighted avg       1.00      1.00      1.00     10000

c:\Users\arthu\OneDrive\Efrei\I1\S5\MachineLearning\machineLearningCybersecurity
\Lib\site-packages\sklearn\metrics\_classification.py:1731:
UndefinedMetricWarning: Precision is ill-defined and being set to 0.0 in labels
with no predicted samples. Use `zero_division` parameter to control this
behavior.
  _warn_prf(average, modifier, f"{metric.capitalize()} is", result.shape[0])
c:\Users\arthu\OneDrive\Efrei\I1\S5\MachineLearning\machineLearningCybersecurity
\Lib\site-packages\sklearn\metrics\_classification.py:1731:
UndefinedMetricWarning: Precision is ill-defined and being set to 0.0 in labels
with no predicted samples. Use `zero_division` parameter to control this
behavior.
  _warn_prf(average, modifier, f"{metric.capitalize()} is", result.shape[0])
c:\Users\arthu\OneDrive\Efrei\I1\S5\MachineLearning\machineLearningCybersecurity
\Lib\site-packages\sklearn\metrics\_classification.py:1731:
UndefinedMetricWarning: Precision is ill-defined and being set to 0.0 in labels
with no predicted samples. Use `zero_division` parameter to control this
behavior.
  _warn_prf(average, modifier, f"{metric.capitalize()} is", result.shape[0])
```



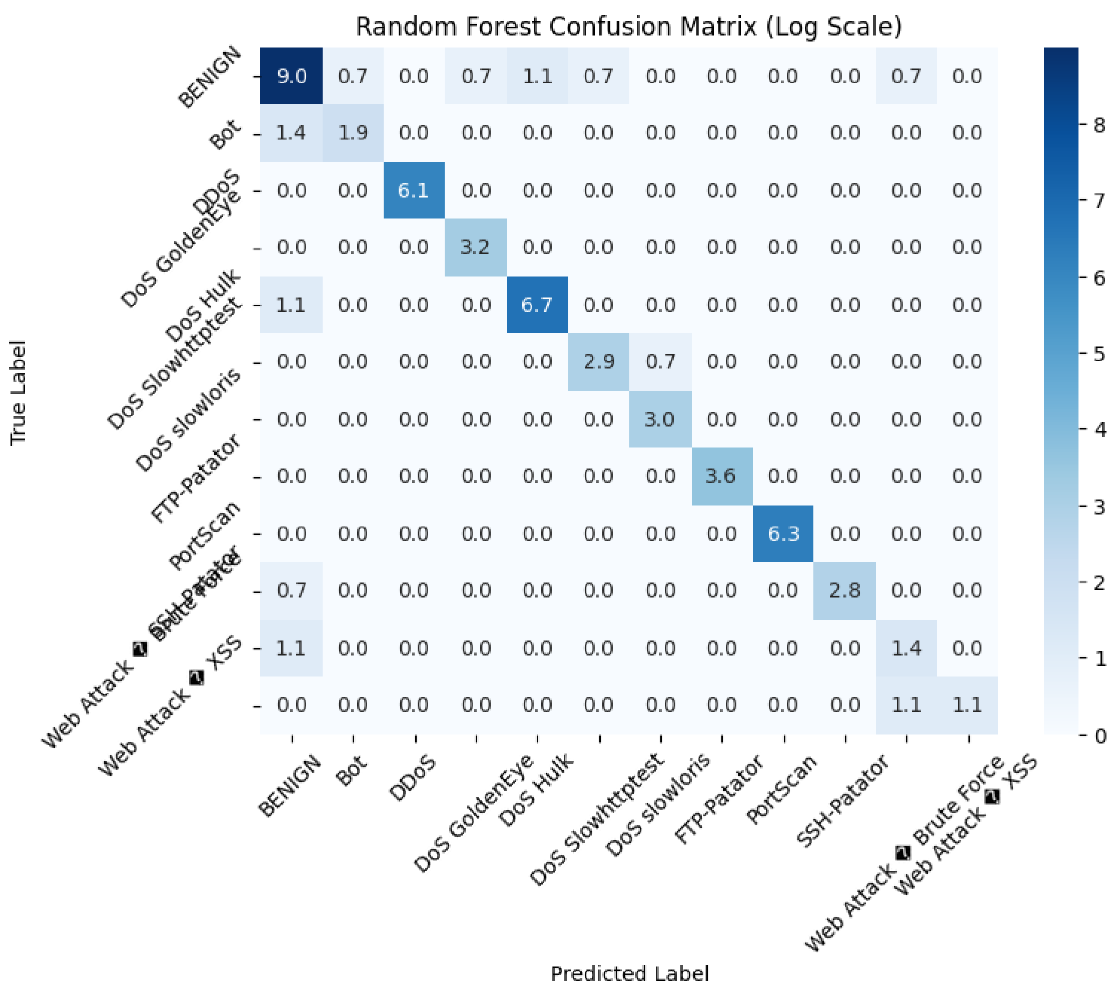
On the SMOTE balanced train set:

This block trains a second Random Forest model, retaining the class_weight='balanced' parameter but crucially fitting it using the SMOTE-resampled training data. The model is then evaluated on the original test set, with its performance (classification report and log-scale confusion matrix) printed and visualized.

[29]:
```
random_forest_smote = RandomForestClassifier(n_estimators=100, random_state=42, class_weight='balanced')
random_forest_smote.fit(X_train_smote, y_train_smote)
models.append({"model" : random_forest_smote, "name": "rand_forest_smote", "library": "sklearn"})
y_pred_rf = random_forest_smote.predict(X_test)

conf_matrix = confusion_matrix(y_test, y_pred_rf)
print("Random Forest Results:")
print("Confusion matrix: ", conf_matrix)
```

```python
print(classification_report(y_test, y_pred_rf))

plt.figure(figsize=(8, 6))
conf_matrix_log = np.log1p(conf_matrix)
sns.heatmap(conf_matrix_log, annot=True, fmt='.1f', cmap='Blues')
plt.title("Random Forest Confusion Matrix (Log Scale)")
plt.xlabel("Predicted Label")
plt.xticks(ticks=np.arange(len(class_names))+0.5, labels=class_names,␣
 ↪rotation=45)
plt.ylabel("True Label")
plt.yticks(ticks=np.arange(len(class_names))+0.5, labels=class_names,␣
 ↪rotation=45)
plt.show()
```

```
Random Forest Results:
Confusion matrix:  [[8062    1    0    1    2    1    0    0    0    0    1
0]
 [   3    6    0    0    0    0    0    0    0    0    0    0]
 [   0    0  442    0    0    0    0    0    0    0    0    0]
 [   0    0    0   24    0    0    0    0    0    0    0    0]
 [   2    0    0    0  811    0    0    0    0    0    0    0]
 [   0    0    0    0    0   18    1    0    0    0    0    0]
 [   0    0    0    0    0    0   19    0    0    0    0    0]
 [   0    0    0    0    0    0    0   37    0    0    0    0]
 [   0    0    0    0    0    0    0    0  543    0    0    0]
 [   1    0    0    0    0    0    0    0    0   16    0    0]
 [   2    0    0    0    0    0    0    0    0    0    3    0]
 [   0    0    0    0    0    0    0    0    0    0    2    2]]
              precision    recall  f1-score   support

           0       1.00      1.00      1.00      8068
           1       0.86      0.67      0.75         9
           2       1.00      1.00      1.00       442
           3       0.96      1.00      0.98        24
           4       1.00      1.00      1.00       813
           5       0.95      0.95      0.95        19
           6       0.95      1.00      0.97        19
           7       1.00      1.00      1.00        37
           8       1.00      1.00      1.00       543
           9       1.00      0.94      0.97        17
          10       0.50      0.60      0.55         5
          11       1.00      0.50      0.67         4

    accuracy                           1.00     10000
   macro avg       0.93      0.89      0.90     10000
weighted avg       1.00      1.00      1.00     10000
```



4.3 2.xg Boost Classification

On the original train set, before SMOTE:

This block trains an XGBoost classifier, utilizing the faster hist tree method and implementing early stopping (stopping after 2 rounds if performance on the test set does not improve). The model is trained on the original training data, evaluated on the test set, and its performance metrics (classification report and log-scale confusion matrix) are printed and visualized.

[30]:

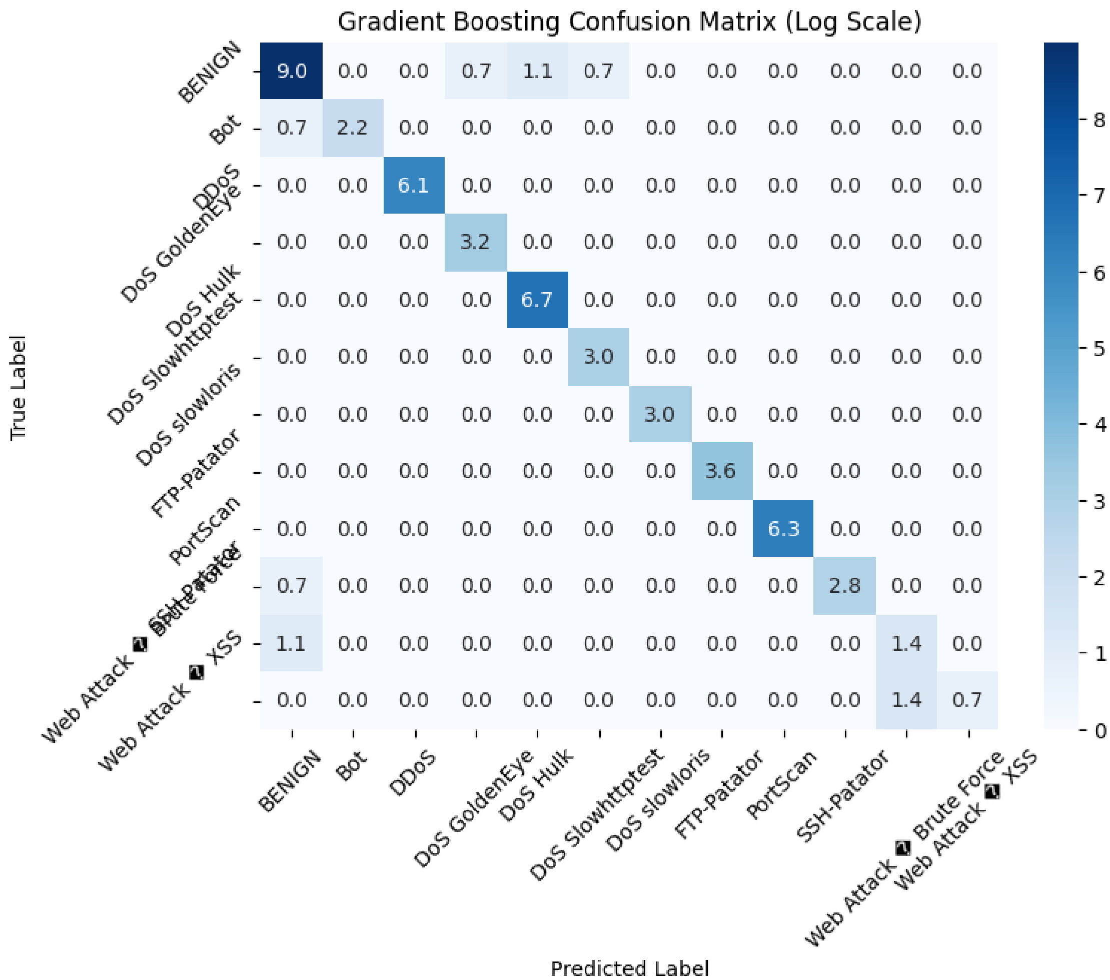
```
gradient_boosting = xgb.XGBClassifier(tree_method="hist", early_stopping_rounds=2)
gradient_boosting.fit(X_train, y_train, eval_set=[(X_test, y_test)])
models.append({"model" : gradient_boosting, "name": "XGB", "library": "xgb"})
y_pred_gb = gradient_boosting.predict(X_test)
conf_matrix = confusion_matrix(y_test, y_pred_gb)
print("Gradient Boosting Results:")
print("Confusion matrix: ", conf_matrix)
```

```python
print(classification_report(y_test, y_pred_gb))

plt.figure(figsize=(8, 6))
conf_matrix_log = np.log1p(conf_matrix)
sns.heatmap(conf_matrix_log, annot=True, fmt='.1f', cmap='Blues')
plt.title("Gradient Boosting Confusion Matrix (Log Scale)")
plt.xlabel("Predicted Label")
plt.xticks(ticks=np.arange(len(class_names))+0.5, labels=class_names,␣
 ↪rotation=45)
plt.ylabel("True Label")
plt.yticks(ticks=np.arange(len(class_names))+0.5, labels=class_names,␣
 ↪rotation=45)
plt.show()
```

```
[0]     validation_0-mlogloss:1.00574
[1]     validation_0-mlogloss:0.69199
[2]     validation_0-mlogloss:0.49702
[3]     validation_0-mlogloss:0.36451
[4]     validation_0-mlogloss:0.27057
[5]     validation_0-mlogloss:0.20249
[6]     validation_0-mlogloss:0.15279
[7]     validation_0-mlogloss:0.11627
[8]     validation_0-mlogloss:0.08944
[9]     validation_0-mlogloss:0.06951
[10]    validation_0-mlogloss:0.05452
[11]    validation_0-mlogloss:0.04334
[12]    validation_0-mlogloss:0.03460
[13]    validation_0-mlogloss:0.02785
[14]    validation_0-mlogloss:0.02265
[15]    validation_0-mlogloss:0.01871
[16]    validation_0-mlogloss:0.01561
[17]    validation_0-mlogloss:0.01334
[18]    validation_0-mlogloss:0.01158
[19]    validation_0-mlogloss:0.01026
[20]    validation_0-mlogloss:0.00927
[21]    validation_0-mlogloss:0.00844
[22]    validation_0-mlogloss:0.00773
[23]    validation_0-mlogloss:0.00720
[24]    validation_0-mlogloss:0.00680
[25]    validation_0-mlogloss:0.00631
[26]    validation_0-mlogloss:0.00606
[27]    validation_0-mlogloss:0.00588
[28]    validation_0-mlogloss:0.00578
[29]    validation_0-mlogloss:0.00560
[30]    validation_0-mlogloss:0.00552
[31]    validation_0-mlogloss:0.00544
[32]    validation_0-mlogloss:0.00540
```

```
[33]	validation_0-mlogloss:0.00529
[34]	validation_0-mlogloss:0.00522
[35]	validation_0-mlogloss:0.00519
[36]	validation_0-mlogloss:0.00521
[37]	validation_0-mlogloss:0.00516
[38]	validation_0-mlogloss:0.00515
[39]	validation_0-mlogloss:0.00511
[40]	validation_0-mlogloss:0.00508
[41]	validation_0-mlogloss:0.00503
[42]	validation_0-mlogloss:0.00502
[43]	validation_0-mlogloss:0.00502
[44]	validation_0-mlogloss:0.00498
[45]	validation_0-mlogloss:0.00495
[46]	validation_0-mlogloss:0.00494
[47]	validation_0-mlogloss:0.00490
[48]	validation_0-mlogloss:0.00491
[49]	validation_0-mlogloss:0.00489
[50]	validation_0-mlogloss:0.00489
[51]	validation_0-mlogloss:0.00489
[52]	validation_0-mlogloss:0.00488
[53]	validation_0-mlogloss:0.00491
[54]	validation_0-mlogloss:0.00491
Gradient Boosting Results:
Confusion matrix:  [[8064    0    0    1    2    1    0    0    0    0    0
0]
 [   1    8    0    0    0    0    0    0    0    0    0    0]
 [   0    0  442    0    0    0    0    0    0    0    0    0]
 [   0    0    0   24    0    0    0    0    0    0    0    0]
 [   0    0    0    0  813    0    0    0    0    0    0    0]
 [   0    0    0    0    0   19    0    0    0    0    0    0]
 [   0    0    0    0    0    0   19    0    0    0    0    0]
 [   0    0    0    0    0    0    0   37    0    0    0    0]
 [   0    0    0    0    0    0    0    0  543    0    0    0]
 [   1    0    0    0    0    0    0    0    0   16    0    0]
 [   2    0    0    0    0    0    0    0    0    0    3    0]
 [   0    0    0    0    0    0    0    0    0    0    3    1]]
              precision    recall  f1-score   support

           0       1.00      1.00      1.00      8068
           1       1.00      0.89      0.94         9
           2       1.00      1.00      1.00       442
           3       0.96      1.00      0.98        24
           4       1.00      1.00      1.00       813
           5       0.95      1.00      0.97        19
           6       1.00      1.00      1.00        19
           7       1.00      1.00      1.00        37
           8       1.00      1.00      1.00       543
           9       1.00      0.94      0.97        17
```

```
          10       0.50      0.60      0.55         5
          11       1.00      0.25      0.40         4

    accuracy                           1.00     10000
   macro avg       0.95      0.89      0.90     10000
weighted avg       1.00      1.00      1.00     10000
```



On the SMOTE balanced train set:

This block trains a second XGBoost classifier with the same parameters (hist tree method, early_stopping_rounds=2) but utilizes the SMOTE-resampled training data. It evaluates the model on the original test set, printing the full classification report and visualizing the log-scale confusion matrix to assess the impact of balancing on the model's performance.

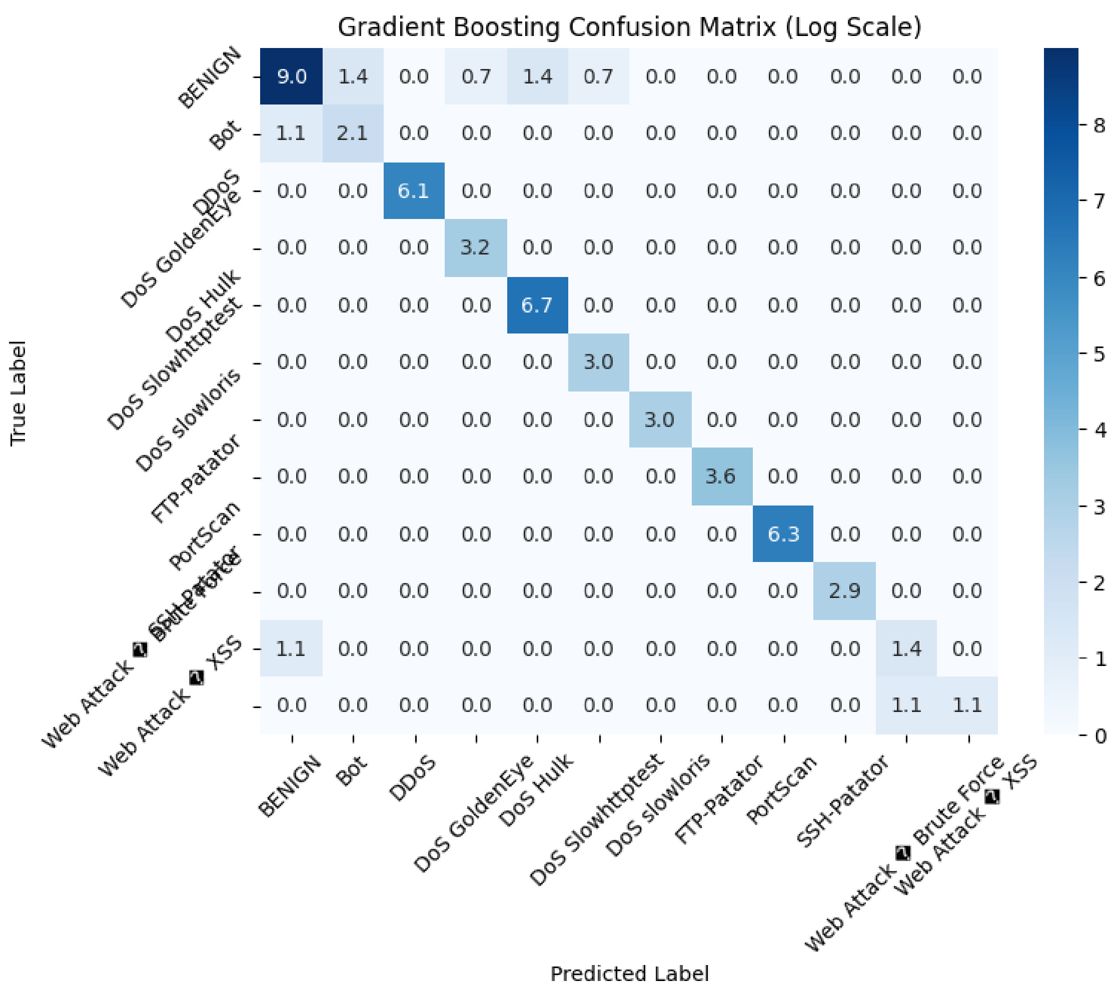
[31]:
```
gradient_boosting_smote = xgb.XGBClassifier(tree_method="hist",
    early_stopping_rounds=2)
```

```python
gradient_boosting_smote.fit(X_train_smote, y_train_smote, eval_set=[(X_test,␣
↪y_test)])
models.append({"model" : gradient_boosting_smote, "name": "XGB_smote",␣
↪"library": "xgb"})
y_pred_gb = gradient_boosting_smote.predict(X_test)

conf_matrix = confusion_matrix(y_test, y_pred_gb)
print("Gradient Boosting Results:")
print("Confusion matrix: ", conf_matrix)
print(classification_report(y_test, y_pred_gb))

plt.figure(figsize=(8, 6))
conf_matrix_log = np.log1p(conf_matrix)
sns.heatmap(conf_matrix_log, annot=True, fmt='.1f', cmap='Blues')
plt.title("Gradient Boosting Confusion Matrix (Log Scale)")
plt.xlabel("Predicted Label")
plt.xticks(ticks=np.arange(len(class_names))+0.5, labels=class_names,␣
↪rotation=45)
plt.ylabel("True Label")
plt.yticks(ticks=np.arange(len(class_names))+0.5, labels=class_names,␣
↪rotation=45)
plt.show()
```

```
[0]	validation_0-mlogloss:1.24058
[1]	validation_0-mlogloss:0.82534
[2]	validation_0-mlogloss:0.60220
[3]	validation_0-mlogloss:0.44161
[4]	validation_0-mlogloss:0.33313
[5]	validation_0-mlogloss:0.25540
[6]	validation_0-mlogloss:0.19896
[7]	validation_0-mlogloss:0.15780
[8]	validation_0-mlogloss:0.12754
[9]	validation_0-mlogloss:0.10024
[10]	validation_0-mlogloss:0.07889
[11]	validation_0-mlogloss:0.06269
[12]	validation_0-mlogloss:0.05028
[13]	validation_0-mlogloss:0.04115
[14]	validation_0-mlogloss:0.03384
[15]	validation_0-mlogloss:0.02826
[16]	validation_0-mlogloss:0.02386
[17]	validation_0-mlogloss:0.02000
[18]	validation_0-mlogloss:0.01730
[19]	validation_0-mlogloss:0.01515
[20]	validation_0-mlogloss:0.01375
[21]	validation_0-mlogloss:0.01225
[22]	validation_0-mlogloss:0.01113
[23]	validation_0-mlogloss:0.01016
```

```
[24]	validation_0-mlogloss:0.00934
[25]	validation_0-mlogloss:0.00888
[26]	validation_0-mlogloss:0.00845
[27]	validation_0-mlogloss:0.00817
[28]	validation_0-mlogloss:0.00786
[29]	validation_0-mlogloss:0.00758
[30]	validation_0-mlogloss:0.00741
[31]	validation_0-mlogloss:0.00731
[32]	validation_0-mlogloss:0.00716
[33]	validation_0-mlogloss:0.00706
[34]	validation_0-mlogloss:0.00692
[35]	validation_0-mlogloss:0.00690
[36]	validation_0-mlogloss:0.00686
[37]	validation_0-mlogloss:0.00678
[38]	validation_0-mlogloss:0.00674
[39]	validation_0-mlogloss:0.00671
[40]	validation_0-mlogloss:0.00663
[41]	validation_0-mlogloss:0.00650
[42]	validation_0-mlogloss:0.00641
[43]	validation_0-mlogloss:0.00634
[44]	validation_0-mlogloss:0.00629
[45]	validation_0-mlogloss:0.00624
[46]	validation_0-mlogloss:0.00620
[47]	validation_0-mlogloss:0.00620
[48]	validation_0-mlogloss:0.00623
[49]	validation_0-mlogloss:0.00624
Gradient Boosting Results:
Confusion matrix:  [[8060    3    0    1    3    1    0    0    0    0    0
0]
 [   2    7    0    0    0    0    0    0    0    0    0    0]
 [   0    0  442    0    0    0    0    0    0    0    0    0]
 [   0    0    0   24    0    0    0    0    0    0    0    0]
 [   0    0    0    0  813    0    0    0    0    0    0    0]
 [   0    0    0    0    0   19    0    0    0    0    0    0]
 [   0    0    0    0    0    0   19    0    0    0    0    0]
 [   0    0    0    0    0    0    0   37    0    0    0    0]
 [   0    0    0    0    0    0    0    0  543    0    0    0]
 [   0    0    0    0    0    0    0    0    0   17    0    0]
 [   2    0    0    0    0    0    0    0    0    0    3    0]
 [   0    0    0    0    0    0    0    0    0    0    2    2]]
              precision    recall  f1-score   support

           0       1.00      1.00      1.00      8068
           1       0.70      0.78      0.74         9
           2       1.00      1.00      1.00       442
           3       0.96      1.00      0.98        24
           4       1.00      1.00      1.00       813
           5       0.95      1.00      0.97        19
```

6	1.00	1.00	1.00	19
7	1.00	1.00	1.00	37
8	1.00	1.00	1.00	543
9	1.00	1.00	1.00	17
10	0.60	0.60	0.60	5
11	1.00	0.50	0.67	4
accuracy			1.00	10000
macro avg	0.93	0.91	0.91	10000
weighted avg	1.00	1.00	1.00	10000



5 2.6 Ensemble Methods

5.1 2.6.1 Voting Classifier

This block implements an Ensemble Learning approach by creating two types of VotingClassifier (Hard and Soft), leveraging Logistic Regression, KNN, and Random Forest as base models. Both ensemble models are trained on the SMOTE-resampled data and then evaluated on the test set, with their performance and classification reports printed. Hard Voting uses majority prediction, while Soft Voting aggregates predicted probabilities for the final result.

[32]:
```
# Create base models for voting ensemble
lr_voting = LogisticRegression(max_iter=1000)
knn_voting = KNeighborsClassifier(n_neighbors=5, weights='distance')
rf_voting = RandomForestClassifier(n_estimators=100, random_state=42,␣
 ↪class_weight='balanced')



# Hard Voting Classifier
hard_voting_clf = VotingClassifier(
    estimators=[
        ('lr', lr_voting),
        ('knn', knn_voting),
        ('rf', rf_voting),
    ],
    voting='hard'
)

print("Training Hard Voting Classifier...")
hard_voting_clf.fit(X_train_smote, y_train_smote)
models.append({"model" : hard_voting_clf, "name": "hard_voting", "library":␣
 ↪"sklearn"})
y_pred_hard = hard_voting_clf.predict(X_test)

print("Hard Voting Results:")
print(classification_report(y_test, y_pred_hard))

# Soft Voting Classifier
soft_voting_clf = VotingClassifier(
    estimators=[
        ('lr', lr_voting),
        ('knn', knn_voting),
        ('rf', rf_voting),
    ],
    voting='soft'
)

print("\nTraining Soft Voting Classifier...")
```

```
soft_voting_clf.fit(X_train_smote, y_train_smote)
models.append({"model" : soft_voting_clf, "name": "soft_voting", "library":
↪"sklearn"})

y_pred_soft = soft_voting_clf.predict(X_test)

print("Soft Voting Results:")
print(classification_report(y_test, y_pred_soft))
```

```
Training Hard Voting Classifier…
```

```
c:\Users\arthu\OneDrive\Efrei\I1\S5\MachineLearning\machineLearningCybersecurity
\Lib\site-packages\sklearn\linear_model\_logistic.py:473: ConvergenceWarning:
lbfgs failed to converge after 1000 iteration(s) (status=1):
STOP: TOTAL NO. OF ITERATIONS REACHED LIMIT

Increase the number of iterations to improve the convergence (max_iter=1000).
You might also want to scale the data as shown in:
    https://scikit-learn.org/stable/modules/preprocessing.html
Please also refer to the documentation for alternative solver options:
    https://scikit-learn.org/stable/modules/linear_model.html#logistic-
regression
  n_iter_i = _check_optimize_result(
```

```
Hard Voting Results:
              precision    recall  f1-score   support

           0       1.00      1.00      1.00      8068
           1       0.39      0.78      0.52         9
           2       1.00      1.00      1.00       442
           3       0.92      1.00      0.96        24
           4       0.99      1.00      0.99       813
           5       1.00      1.00      1.00        19
           6       0.90      1.00      0.95        19
           7       0.93      1.00      0.96        37
           8       1.00      1.00      1.00       543
           9       0.80      0.94      0.86        17
          10       0.50      0.60      0.55         5
          11       1.00      0.50      0.67         4

    accuracy                           1.00     10000
   macro avg       0.87      0.90      0.87     10000
weighted avg       1.00      1.00      1.00     10000
```

```
Training Soft Voting Classifier…
```

```
c:\Users\arthu\OneDrive\Efrei\I1\S5\MachineLearning\machineLearningCybersecurity
\Lib\site-packages\sklearn\linear_model\_logistic.py:473: ConvergenceWarning:
```

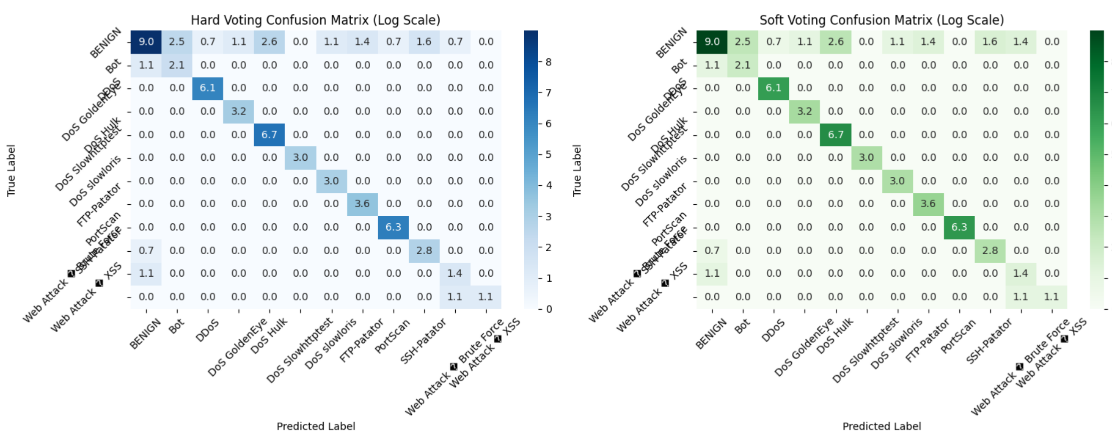
```
lbfgs failed to converge after 1000 iteration(s) (status=1):
STOP: TOTAL NO. OF ITERATIONS REACHED LIMIT

Increase the number of iterations to improve the convergence (max_iter=1000).
You might also want to scale the data as shown in:
    https://scikit-learn.org/stable/modules/preprocessing.html
Please also refer to the documentation for alternative solver options:
    https://scikit-learn.org/stable/modules/linear_model.html#logistic-
regression
  n_iter_i = _check_optimize_result(
```

```
Soft Voting Results:
              precision    recall  f1-score   support

           0       1.00      1.00      1.00      8068
           1       0.39      0.78      0.52         9
           2       1.00      1.00      1.00       442
           3       0.92      1.00      0.96        24
           4       0.99      1.00      0.99       813
           5       1.00      1.00      1.00        19
           6       0.90      1.00      0.95        19
           7       0.93      1.00      0.96        37
           8       1.00      1.00      1.00       543
           9       0.80      0.94      0.86        17
          10       0.38      0.60      0.46         5
          11       1.00      0.50      0.67         4

    accuracy                           1.00     10000
   macro avg       0.86      0.90      0.86     10000
weighted avg       1.00      1.00      1.00     10000
```

This block generates a figure with two subplots to visually compare the performance of the Hard and Soft Voting ensemble classifiers. It calculates and displays the log-scale confusion matrices for both models side-by-side, using distinct colormaps ('Blues' and 'Greens') to facilitate comparison of their misclassification patterns.

[33]:
```python
# Visualize voting results
fig, axes = plt.subplots(1, 2, figsize=(16, 6))

# Hard voting confusion matrix
conf_matrix_hard = confusion_matrix(y_test, y_pred_hard)
conf_matrix_hard_log = np.log1p(conf_matrix_hard)
sns.heatmap(conf_matrix_hard_log, annot=True, fmt='.1f', cmap='Blues',␣
 ↪ax=axes[0])
axes[0].set_title("Hard Voting Confusion Matrix (Log Scale)")
axes[0].set_xlabel("Predicted Label")
axes[0].set_ylabel("True Label")
```

```
axes[0].set_xticks(np.arange(len(class_names))+0.5)
axes[0].set_xticklabels(class_names, rotation=45)
axes[0].set_yticks(np.arange(len(class_names))+0.5)
axes[0].set_yticklabels(class_names, rotation=45)

# Soft voting confusion matrix
conf_matrix_soft = confusion_matrix(y_test, y_pred_soft)
conf_matrix_soft_log = np.log1p(conf_matrix_soft)
sns.heatmap(conf_matrix_soft_log, annot=True, fmt='.1f', cmap='Greens', ax=axes[1])
axes[1].set_title("Soft Voting Confusion Matrix (Log Scale)")
axes[1].set_xlabel("Predicted Label")
axes[1].set_ylabel("True Label")
axes[1].set_xticks(np.arange(len(class_names))+0.5)
axes[1].set_xticklabels(class_names, rotation=45)
axes[1].set_yticks(np.arange(len(class_names))+0.5)
axes[1].set_yticklabels(class_names, rotation=45)

plt.tight_layout()
plt.show()
```



5.2 2.6.2 Stacking Classifier

This block implements a Stacking ensemble, using Logistic Regression, KNN, and Random Forest as base estimators, with a separate Random Forest as the meta-learner to combine their predictions. The model is trained on the SMOTE-resampled data, using 5-fold cross-validation to generate meta-features. The final performance on the test set is evaluated via a classification report and visualized with a log-scale confusion matrix.

[34]:
```
# Create base models for stacking
base_models = [
    ('lr', LogisticRegression()),
```

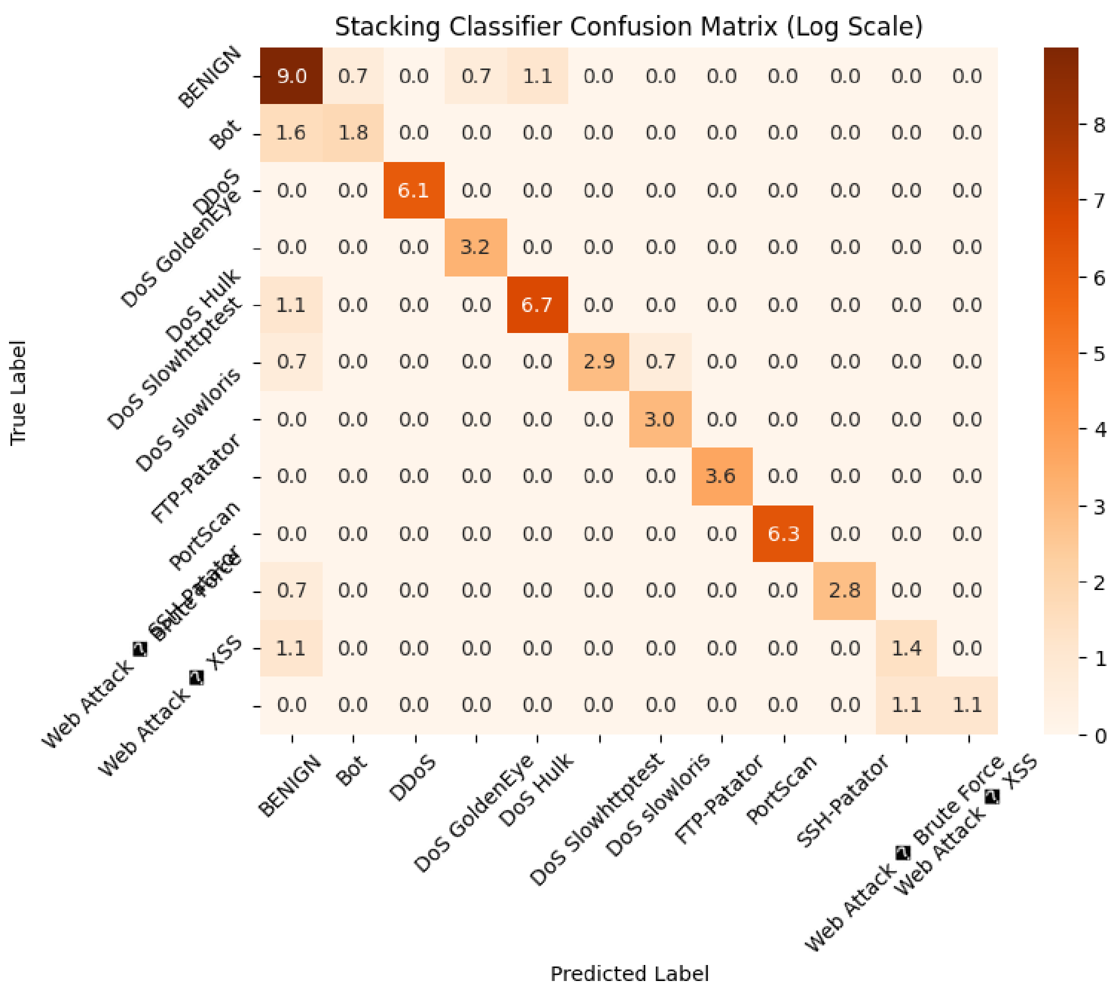
```
    ('knn', KNeighborsClassifier(n_neighbors=5, weights='distance')),
    ('rf', RandomForestClassifier(n_estimators=100, random_state=42,␣
 ↪class_weight='balanced')),
]

# Use Random Forest as meta-learner
meta_learner = RandomForestClassifier(n_estimators=50, random_state=42)

# Create stacking classifier
stacking_clf = StackingClassifier(
    estimators=base_models,
    final_estimator=meta_learner,
    cv=5  # Use 5-fold cross-validation for generating meta-features
)

print("Training Stacking Classifier...")
stacking_clf.fit(X_train_smote, y_train_smote)
models.append({"model" : stacking_clf, "name": "stacking", "library":␣
 ↪"sklearn", "name":"stacking"})
y_pred_stack = stacking_clf.predict(X_test)

print("Stacking Results:")
print(classification_report(y_test, y_pred_stack))

# Visualize stacking results
conf_matrix_stack = confusion_matrix(y_test, y_pred_stack)
plt.figure(figsize=(8, 6))
conf_matrix_stack_log = np.log1p(conf_matrix_stack)
sns.heatmap(conf_matrix_stack_log, annot=True, fmt='.1f', cmap='Oranges')
plt.title("Stacking Classifier Confusion Matrix (Log Scale)")
plt.xlabel("Predicted Label")
plt.xticks(ticks=np.arange(len(class_names))+0.5, labels=class_names,␣
 ↪rotation=45)
plt.ylabel("True Label")
plt.yticks(ticks=np.arange(len(class_names))+0.5, labels=class_names,␣
 ↪rotation=45)
plt.show()
```

```
Training Stacking Classifier…
```

```
c:\Users\arthu\OneDrive\Efrei\I1\S5\MachineLearning\machineLearningCybersecurity
\Lib\site-packages\sklearn\linear_model\_logistic.py:473: ConvergenceWarning:
lbfgs failed to converge after 100 iteration(s) (status=1):
STOP: TOTAL NO. OF ITERATIONS REACHED LIMIT

Increase the number of iterations to improve the convergence (max_iter=100).
You might also want to scale the data as shown in:
    https://scikit-learn.org/stable/modules/preprocessing.html
```

```
Please also refer to the documentation for alternative solver options:
    https://scikit-learn.org/stable/modules/linear_model.html#logistic-
regression
  n_iter_i = _check_optimize_result(
c:\Users\arthu\OneDrive\Efrei\I1\S5\MachineLearning\machineLearningCybersecurity
\Lib\site-packages\sklearn\linear_model\_logistic.py:473: ConvergenceWarning:
lbfgs failed to converge after 100 iteration(s) (status=1):
STOP: TOTAL NO. OF ITERATIONS REACHED LIMIT

Increase the number of iterations to improve the convergence (max_iter=100).
You might also want to scale the data as shown in:
    https://scikit-learn.org/stable/modules/preprocessing.html
Please also refer to the documentation for alternative solver options:
    https://scikit-learn.org/stable/modules/linear_model.html#logistic-
regression
  n_iter_i = _check_optimize_result(
c:\Users\arthu\OneDrive\Efrei\I1\S5\MachineLearning\machineLearningCybersecurity
\Lib\site-packages\sklearn\linear_model\_logistic.py:473: ConvergenceWarning:
lbfgs failed to converge after 100 iteration(s) (status=1):
STOP: TOTAL NO. OF ITERATIONS REACHED LIMIT

Increase the number of iterations to improve the convergence (max_iter=100).
You might also want to scale the data as shown in:
    https://scikit-learn.org/stable/modules/preprocessing.html
Please also refer to the documentation for alternative solver options:
    https://scikit-learn.org/stable/modules/linear_model.html#logistic-
regression
  n_iter_i = _check_optimize_result(
c:\Users\arthu\OneDrive\Efrei\I1\S5\MachineLearning\machineLearningCybersecurity
\Lib\site-packages\sklearn\linear_model\_logistic.py:473: ConvergenceWarning:
lbfgs failed to converge after 100 iteration(s) (status=1):
STOP: TOTAL NO. OF ITERATIONS REACHED LIMIT

Increase the number of iterations to improve the convergence (max_iter=100).
You might also want to scale the data as shown in:
    https://scikit-learn.org/stable/modules/preprocessing.html
Please also refer to the documentation for alternative solver options:
    https://scikit-learn.org/stable/modules/linear_model.html#logistic-
regression
  n_iter_i = _check_optimize_result(
c:\Users\arthu\OneDrive\Efrei\I1\S5\MachineLearning\machineLearningCybersecurity
\Lib\site-packages\sklearn\linear_model\_logistic.py:473: ConvergenceWarning:
lbfgs failed to converge after 100 iteration(s) (status=1):
STOP: TOTAL NO. OF ITERATIONS REACHED LIMIT

Increase the number of iterations to improve the convergence (max_iter=100).
You might also want to scale the data as shown in:
    https://scikit-learn.org/stable/modules/preprocessing.html
```

```
Please also refer to the documentation for alternative solver options:
    https://scikit-learn.org/stable/modules/linear_model.html#logistic-
regression
  n_iter_i = _check_optimize_result(
c:\Users\arthu\OneDrive\Efrei\I1\S5\MachineLearning\machineLearningCybersecurity
\Lib\site-packages\sklearn\linear_model\_logistic.py:473: ConvergenceWarning:
lbfgs failed to converge after 100 iteration(s) (status=1):
STOP: TOTAL NO. OF ITERATIONS REACHED LIMIT

Increase the number of iterations to improve the convergence (max_iter=100).
You might also want to scale the data as shown in:
    https://scikit-learn.org/stable/modules/preprocessing.html
Please also refer to the documentation for alternative solver options:
    https://scikit-learn.org/stable/modules/linear_model.html#logistic-
regression
  n_iter_i = _check_optimize_result(

Stacking Results:
              precision    recall  f1-score   support

           0       1.00      1.00      1.00      8068
           1       0.83      0.56      0.67         9
           2       1.00      1.00      1.00       442
           3       0.96      1.00      0.98        24
           4       1.00      1.00      1.00       813
           5       1.00      0.89      0.94        19
           6       0.95      1.00      0.97        19
           7       1.00      1.00      1.00        37
           8       1.00      1.00      1.00       543
           9       1.00      0.94      0.97        17
          10       0.60      0.60      0.60         5
          11       1.00      0.50      0.67         4

    accuracy                           1.00     10000
   macro avg       0.94      0.87      0.90     10000
weighted avg       1.00      1.00      1.00     10000
```



5.3 2.7 Model Performance Comparison

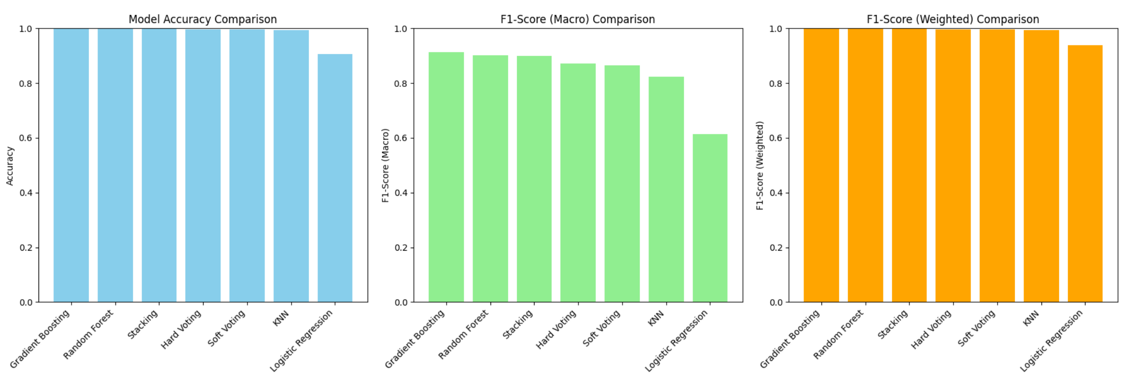
This block aggregates the performance metrics (Accuracy, F1-Macro, F1-Weighted) for all trained models (base and ensemble). It creates a comparison table, sorted by F1-Weighted score, and visualizes the results using three side-by-side bar plots to compare the models across these key metrics.

[40]:
```
# Get predictions from all models for comparison
models_to_compare = {
    'Logistic Regression': logistic_regression_smote,
    'KNN': knn_smote,
    'Random Forest': random_forest_smote,
    'Gradient Boosting': gradient_boosting_smote,
    'Hard Voting': hard_voting_clf,
    'Soft Voting': soft_voting_clf,
    'Stacking': stacking_clf
```

```
}

# Calculate metrics for each model
results = []
for name, model in models_to_compare.items():
    if name in ['Hard Voting', 'Soft Voting', 'Stacking']:
        # These models are already trained and we have their predictions
        if name == 'Hard Voting':
            y_pred = y_pred_hard
        elif name == 'Soft Voting':
            y_pred = y_pred_soft
        else:  # Stacking
            y_pred = y_pred_stack
    else:
        y_pred = model.predict(X_test)

    accuracy = accuracy_score(y_test, y_pred)
    f1_macro = f1_score(y_test, y_pred, average='macro')
    f1_weighted = f1_score(y_test, y_pred, average='weighted')

    results.append({
        'Model': name,
        'Accuracy': accuracy,
        'F1-Score (Macro)': f1_macro,
        'F1-Score (Weighted)': f1_weighted
    })

# Create comparison DataFrame
comparison_df = pd.DataFrame(results)
comparison_df = comparison_df.sort_values('F1-Score (Weighted)',␣
 ↪ascending=False)

print("Model Performance Comparison:")
print("=" * 80)
print(comparison_df.to_string(index=False, float_format='%.4f'))

# Visualize model comparison
fig, axes = plt.subplots(1, 3, figsize=(18, 6))

# Accuracy comparison
axes[0].bar(range(len(comparison_df)), comparison_df['Accuracy'],␣
 ↪color='skyblue')
axes[0].set_title('Model Accuracy Comparison')
axes[0].set_ylabel('Accuracy')
axes[0].set_xticks(range(len(comparison_df)))
axes[0].set_xticklabels(comparison_df['Model'], rotation=45, ha='right')
axes[0].set_ylim(0, 1)
```

```
# F1-Score (Macro) comparison
axes[1].bar(range(len(comparison_df)), comparison_df['F1-Score (Macro)'],␣
↪color='lightgreen')
axes[1].set_title('F1-Score (Macro) Comparison')
axes[1].set_ylabel('F1-Score (Macro)')
axes[1].set_xticks(range(len(comparison_df)))
axes[1].set_xticklabels(comparison_df['Model'], rotation=45, ha='right')
axes[1].set_ylim(0, 1)

# F1-Score (Weighted) comparison
axes[2].bar(range(len(comparison_df)), comparison_df['F1-Score (Weighted)'],␣
↪color='orange')
axes[2].set_title('F1-Score (Weighted) Comparison')
axes[2].set_ylabel('F1-Score (Weighted)')
axes[2].set_xticks(range(len(comparison_df)))
axes[2].set_xticklabels(comparison_df['Model'], rotation=45, ha='right')
axes[2].set_ylim(0, 1)

plt.tight_layout()
plt.show()
```

```
Model Performance Comparison:
================================================================================
              Model  Accuracy  F1-Score (Macro)  F1-Score (Weighted)
  Gradient Boosting    0.9986            0.9129               0.9986
      Random Forest    0.9983            0.9025               0.9983
           Stacking    0.9983            0.8998               0.9982
        Hard Voting    0.9956            0.8712               0.9958
        Soft Voting    0.9955            0.8643               0.9958
                KNN    0.9939            0.8234               0.9943
Logistic Regression    0.9073            0.6133               0.9396
```

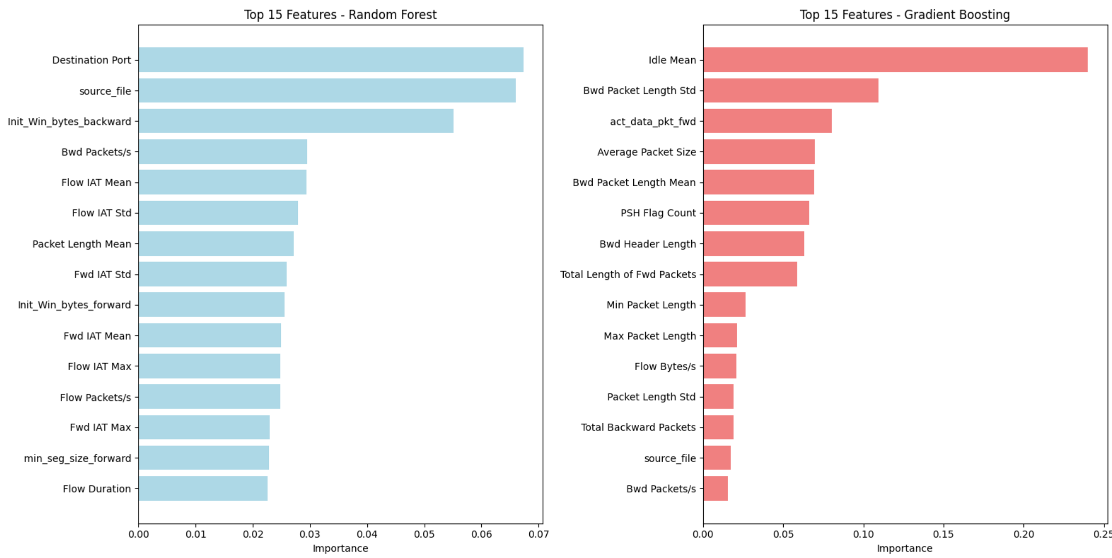


5.4 2.8 Feature Importance Analysis

This block calculates and compares feature importance scores from the Random Forest and Gradient Boosting models, ranking features for each. It then visualizes the top 15 most important features for both models using horizontal bar charts and prints the top 10 features in a table format.

[36]:
```
# Extract feature importance from tree-based models
feature_names = X.columns

# Random Forest feature importance
rf_importance = random_forest.feature_importances_
rf_feature_df = pd.DataFrame({
    'feature': feature_names,
    'importance': rf_importance
}).sort_values('importance', ascending=False)

# Gradient Boosting feature importance
gb_importance = gradient_boosting.feature_importances_
gb_feature_df = pd.DataFrame({
    'feature': feature_names,
    'importance': gb_importance
}).sort_values('importance', ascending=False)

# Plot feature importance comparison
fig, axes = plt.subplots(1, 2, figsize=(16, 8))

# Top 15 features from Random Forest
top_rf_features = rf_feature_df.head(15)
axes[0].barh(range(len(top_rf_features)), top_rf_features['importance'],␣
 ↪color='lightblue')
axes[0].set_yticks(range(len(top_rf_features)))
axes[0].set_yticklabels(top_rf_features['feature'])
axes[0].set_xlabel('Importance')
axes[0].set_title('Top 15 Features - Random Forest')
axes[0].invert_yaxis()

# Top 15 features from Gradient Boosting
top_gb_features = gb_feature_df.head(15)
axes[1].barh(range(len(top_gb_features)), top_gb_features['importance'],␣
 ↪color='lightcoral')
axes[1].set_yticks(range(len(top_gb_features)))
axes[1].set_yticklabels(top_gb_features['feature'])
axes[1].set_xlabel('Importance')
axes[1].set_title('Top 15 Features - Gradient Boosting')
axes[1].invert_yaxis()

plt.tight_layout()
plt.show()
```

```python
print("Top 10 Most Important Features (Random Forest):")
print(rf_feature_df.head(10).to_string(index=False))

print("\nTop 10 Most Important Features (Gradient Boosting):")
print(gb_feature_df.head(10).to_string(index=False))
```




```
Top 10 Most Important Features (Random Forest):
                feature  importance
       Destination Port    0.067340
            source_file    0.065975
Init_Win_bytes_backward    0.055071
          Bwd Packets/s    0.029580
          Flow IAT Mean    0.029441
           Flow IAT Std    0.027969
     Packet Length Mean    0.027132
            Fwd IAT Std    0.025926
 Init_Win_bytes_forward    0.025565
           Fwd IAT Mean    0.024942

Top 10 Most Important Features (Gradient Boosting):
               feature  importance
             Idle Mean    0.240179
 Bwd Packet Length Std    0.109400
      act_data_pkt_fwd    0.080240
   Average Packet Size    0.069569
Bwd Packet Length Mean    0.069367
        PSH Flag Count    0.066146
     Bwd Header Length    0.063291
```

```
Total Length of Fwd Packets    0.058635
          Min Packet Length    0.026591
          Max Packet Length    0.021086
```

6 6. MLflow Tracking

This block configures MLflow by setting the tracking URI and experiment name, then starts a run named "test run" to automatically log all trained models from the models list. It uses the appropriate MLflow logging function (mlf.sklearn.log_model or mlf.xgboost.log_model) to save each model as an artifact and registers it for version tracking.

[ ]:
```
mlf.set_tracking_uri(uri="http://localhost:5001")
mlf.set_experiment("Cybersecurity project")
with mlf.start_run(run_name = "test run"):
    for mod in models:
        if mod["library"] == "sklearn":
            mlf.sklearn.log_model(
                sk_model=mod["model"],
                artifact_path=mod["name"],
                registered_model_name=f"reg_{mod["name"]}"
            )
        else:
            mlf.xgboost.log_model(
                xgb_model=mod["model"],
                artifact_path=mod["name"],
                registered_model_name=f"reg_{mod["name"]}"
            )
```

```
2025/12/09 21:27:35 INFO mlflow.tracking.fluent: Experiment with name
'Cybersecurity project' does not exist. Creating a new experiment.
2025/12/09 21:27:36 WARNING mlflow.models.model: `artifact_path` is deprecated.
Please use `name` instead.
2025/12/09 21:27:49 WARNING mlflow.models.model: Model logged without a
signature and input example. Please set `input_example` parameter when logging
the model to auto infer the model signature.
Successfully registered model 'reg_log_reg'.
2025/12/09 21:27:51 INFO mlflow.store.model_registry.abstract_store: Waiting up
to 300 seconds for model version to finish creation. Model name: reg_log_reg,
version 1
Created version '1' of model 'reg_log_reg'.
2025/12/09 21:27:51 WARNING mlflow.models.model: `artifact_path` is deprecated.
Please use `name` instead.
2025/12/09 21:27:55 WARNING mlflow.models.model: Model logged without a
signature and input example. Please set `input_example` parameter when logging
the model to auto infer the model signature.
Successfully registered model 'reg_log_reg_smote'.
2025/12/09 21:27:55 INFO mlflow.store.model_registry.abstract_store: Waiting up
```

```
to 300 seconds for model version to finish creation. Model name:
reg_log_reg_smote, version 1
Created version '1' of model 'reg_log_reg_smote'.
2025/12/09 21:27:55 WARNING mlflow.models.model: `artifact_path` is deprecated.
Please use `name` instead.
2025/12/09 21:27:59 WARNING mlflow.models.model: Model logged without a
signature and input example. Please set `input_example` parameter when logging
the model to auto infer the model signature.
Successfully registered model 'reg_knn'.
2025/12/09 21:28:00 INFO mlflow.store.model_registry.abstract_store: Waiting up
to 300 seconds for model version to finish creation. Model name: reg_knn,
version 1
Created version '1' of model 'reg_knn'.
2025/12/09 21:28:00 WARNING mlflow.models.model: `artifact_path` is deprecated.
Please use `name` instead.
2025/12/09 21:28:04 WARNING mlflow.models.model: Model logged without a
signature and input example. Please set `input_example` parameter when logging
the model to auto infer the model signature.
Successfully registered model 'reg_knn_smote'.
2025/12/09 21:28:27 INFO mlflow.store.model_registry.abstract_store: Waiting up
to 300 seconds for model version to finish creation. Model name: reg_knn_smote,
version 1
Created version '1' of model 'reg_knn_smote'.
2025/12/09 21:28:27 WARNING mlflow.models.model: `artifact_path` is deprecated.
Please use `name` instead.
2025/12/09 21:28:31 WARNING mlflow.models.model: Model logged without a
signature and input example. Please set `input_example` parameter when logging
the model to auto infer the model signature.
Successfully registered model 'reg_rand_forest'.
2025/12/09 21:28:32 INFO mlflow.store.model_registry.abstract_store: Waiting up
to 300 seconds for model version to finish creation. Model name:
reg_rand_forest, version 1
Created version '1' of model 'reg_rand_forest'.
2025/12/09 21:28:32 WARNING mlflow.models.model: `artifact_path` is deprecated.
Please use `name` instead.
2025/12/09 21:28:36 WARNING mlflow.models.model: Model logged without a
signature and input example. Please set `input_example` parameter when logging
the model to auto infer the model signature.
Successfully registered model 'reg_rand_forest_smote'.
2025/12/09 21:28:41 INFO mlflow.store.model_registry.abstract_store: Waiting up
to 300 seconds for model version to finish creation. Model name:
reg_rand_forest_smote, version 1
Created version '1' of model 'reg_rand_forest_smote'.
2025/12/09 21:28:41 WARNING mlflow.models.model: `artifact_path` is deprecated.
Please use `name` instead.
2025/12/09 21:28:47 WARNING mlflow.models.model: Model logged without a
signature and input example. Please set `input_example` parameter when logging
the model to auto infer the model signature.
```

```
Successfully registered model 'reg_XGB'.
2025/12/09 21:28:49 INFO mlflow.store.model_registry.abstract_store: Waiting up
to 300 seconds for model version to finish creation. Model name: reg_XGB,
version 1
Created version '1' of model 'reg_XGB'.
2025/12/09 21:28:49 WARNING mlflow.models.model: `artifact_path` is deprecated.
Please use `name` instead.
2025/12/09 21:28:54 WARNING mlflow.models.model: Model logged without a
signature and input example. Please set `input_example` parameter when logging
the model to auto infer the model signature.
Successfully registered model 'reg_XGB_smote'.
2025/12/09 21:28:54 INFO mlflow.store.model_registry.abstract_store: Waiting up
to 300 seconds for model version to finish creation. Model name: reg_XGB_smote,
version 1
Created version '1' of model 'reg_XGB_smote'.
2025/12/09 21:28:54 WARNING mlflow.models.model: `artifact_path` is deprecated.
Please use `name` instead.
2025/12/09 21:28:59 WARNING mlflow.models.model: Model logged without a
signature and input example. Please set `input_example` parameter when logging
the model to auto infer the model signature.
Successfully registered model 'reg_hard_voting'.
2025/12/09 21:30:06 INFO mlflow.store.model_registry.abstract_store: Waiting up
to 300 seconds for model version to finish creation. Model name:
reg_hard_voting, version 1
Created version '1' of model 'reg_hard_voting'.
2025/12/09 21:30:06 WARNING mlflow.models.model: `artifact_path` is deprecated.
Please use `name` instead.
2025/12/09 21:30:25 WARNING mlflow.models.model: Model logged without a
signature and input example. Please set `input_example` parameter when logging
the model to auto infer the model signature.
Successfully registered model 'reg_soft_voting'.
2025/12/09 21:31:08 INFO mlflow.store.model_registry.abstract_store: Waiting up
to 300 seconds for model version to finish creation. Model name:
reg_soft_voting, version 1
Created version '1' of model 'reg_soft_voting'.
2025/12/09 21:31:09 WARNING mlflow.models.model: `artifact_path` is deprecated.
Please use `name` instead.
2025/12/09 21:31:13 WARNING mlflow.models.model: Model logged without a
signature and input example. Please set `input_example` parameter when logging
the model to auto infer the model signature.
Successfully registered model 'reg_stacking'.
2025/12/09 21:32:19 INFO mlflow.store.model_registry.abstract_store: Waiting up
to 300 seconds for model version to finish creation. Model name: reg_stacking,
version 1
```

```
 View run test run at: http://localhost:5001/#/experiments/312917764447112075/r
uns/5acd235c9dde4a518023b614fe5b4f86
 View experiment at: http://localhost:5001/#/experiments/312917764447112075
```

```
Created version '1' of model 'reg_stacking'.
```