copy-of-adr-1

November 7, 2024

[ ]:
```
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
```

[ ]:
```
# Install RDKit in Google Colab
!pip install rdkit-pypi
```

```
Collecting rdkit-pypi
  Downloading rdkit_pypi-2022.9.5-cp310-cp310-manylinux_2_17_x86_64.manylinux201
4_x86_64.whl.metadata (3.9 kB)
Requirement already satisfied: numpy in /usr/local/lib/python3.10/dist-packages
(from rdkit-pypi) (1.26.4)
Requirement already satisfied: Pillow in /usr/local/lib/python3.10/dist-packages
(from rdkit-pypi) (10.4.0)
Downloading
rdkit_pypi-2022.9.5-cp310-cp310-manylinux_2_17_x86_64.manylinux2014_x86_64.whl
(29.4 MB)
                          29.4/29.4 MB
54.6 MB/s eta 0:00:00
Installing collected packages: rdkit-pypi
Successfully installed rdkit-pypi-2022.9.5
```

[ ]:
```
adr_df = pd.read_csv('binary adr.csv')
```

[ ]:
```
from rdkit import Chem
from rdkit.Chem import AllChem

smiles_list = adr_df['Chemical Compound']

# Convert SMILES to RDKit Mol objects
mols = [Chem.MolFromSmiles(smile) for smile in smiles_list]

# Generate molecular fingerprints (1024-bit) for each molecule
fingerprints = [AllChem.GetMorganFingerprintAsBitVect(mol, 2, nBits=1024) for
 ↪mol in mols]
```

[ ]:
```
print(fingerprints)
```

```
[<rdkit.DataStructs.cDataStructs.ExplicitBitVect object at 0x7dc7c96a2b90>,
<rdkit.DataStructs.cDataStructs.ExplicitBitVect object at 0x7dc7c96a2c00>,
<rdkit.DataStructs.cDataStructs.ExplicitBitVect object at 0x7dc7c96a2c70>,
<rdkit.DataStructs.cDataStructs.ExplicitBitVect object at 0x7dc7c96a2ce0>,
<rdkit.DataStructs.cDataStructs.ExplicitBitVect object at 0x7dc7c96a2d50>,
<rdkit.DataStructs.cDataStructs.ExplicitBitVect object at 0x7dc7c96a2dc0>,
<rdkit.DataStructs.cDataStructs.ExplicitBitVect object at 0x7dc7c96a2e30>,
<rdkit.DataStructs.cDataStructs.ExplicitBitVect object at 0x7dc7c96a2ea0>,
<rdkit.DataStructs.cDataStructs.ExplicitBitVect object at 0x7dc7c96a2f10>,
<rdkit.DataStructs.cDataStructs.ExplicitBitVect object at 0x7dc7c96a2f80>,
<rdkit.DataStructs.cDataStructs.ExplicitBitVect object at 0x7dc7c96a2ff0>,
<rdkit.DataStructs.cDataStructs.ExplicitBitVect object at 0x7dc7c96a3060>,
<rdkit.DataStructs.cDataStructs.ExplicitBitVect object at 0x7dc7c96a30d0>,
<rdkit.DataStructs.cDataStructs.ExplicitBitVect object at 0x7dc7c96a3140>,
<rdkit.DataStructs.cDataStructs.ExplicitBitVect object at 0x7dc7c96a31b0>,
<rdkit.DataStructs.cDataStructs.ExplicitBitVect object at 0x7dc7c96a3220>,
<rdkit.DataStructs.cDataStructs.ExplicitBitVect object at 0x7dc7c96a3290>,
<rdkit.DataStructs.cDataStructs.ExplicitBitVect object at 0x7dc7c96a3300>,
<rdkit.DataStructs.cDataStructs.ExplicitBitVect object at 0x7dc7c96a3370>,
<rdkit.DataStructs.cDataStructs.ExplicitBitVect object at 0x7dc7c96a33e0>,
<rdkit.DataStructs.cDataStructs.ExplicitBitVect object at 0x7dc7c96a3450>,
<rdkit.DataStructs.cDataStructs.ExplicitBitVect object at 0x7dc7c96a34c0>,
<rdkit.DataStructs.cDataStructs.ExplicitBitVect object at 0x7dc7c96a3530>,
<rdkit.DataStructs.cDataStructs.ExplicitBitVect object at 0x7dc7c96a35a0>,
<rdkit.DataStructs.cDataStructs.ExplicitBitVect object at 0x7dc7c96a3610>,
<rdkit.DataStructs.cDataStructs.ExplicitBitVect object at 0x7dc7c96a3680>,
<rdkit.DataStructs.cDataStructs.ExplicitBitVect object at 0x7dc7c96a36f0>,
<rdkit.DataStructs.cDataStructs.ExplicitBitVect object at 0x7dc7c96a3760>,
<rdkit.DataStructs.cDataStructs.ExplicitBitVect object at 0x7dc7c96a37d0>,
<rdkit.DataStructs.cDataStructs.ExplicitBitVect object at 0x7dc7c96a3840>,
<rdkit.DataStructs.cDataStructs.ExplicitBitVect object at 0x7dc7c96a38b0>,
<rdkit.DataStructs.cDataStructs.ExplicitBitVect object at 0x7dc7c96a3920>,
<rdkit.DataStructs.cDataStructs.ExplicitBitVect object at 0x7dc7c96a3990>,
<rdkit.DataStructs.cDataStructs.ExplicitBitVect object at 0x7dc7c96a3a00>,
<rdkit.DataStructs.cDataStructs.ExplicitBitVect object at 0x7dc7c96a3a70>,
<rdkit.DataStructs.cDataStructs.ExplicitBitVect object at 0x7dc7c96a3ae0>,
<rdkit.DataStructs.cDataStructs.ExplicitBitVect object at 0x7dc7c96a3b50>,
<rdkit.DataStructs.cDataStructs.ExplicitBitVect object at 0x7dc7c96a3bc0>,
<rdkit.DataStructs.cDataStructs.ExplicitBitVect object at 0x7dc7c96a3c30>,
<rdkit.DataStructs.cDataStructs.ExplicitBitVect object at 0x7dc7c96a3ca0>,
<rdkit.DataStructs.cDataStructs.ExplicitBitVect object at 0x7dc7c96a3d10>,
<rdkit.DataStructs.cDataStructs.ExplicitBitVect object at 0x7dc7c96a3d80>,
<rdkit.DataStructs.cDataStructs.ExplicitBitVect object at 0x7dc7c96a3df0>,
<rdkit.DataStructs.cDataStructs.ExplicitBitVect object at 0x7dc7c96a3e60>,
<rdkit.DataStructs.cDataStructs.ExplicitBitVect object at 0x7dc7c96a3ed0>,
<rdkit.DataStructs.cDataStructs.ExplicitBitVect object at 0x7dc7c96a3f40>,
<rdkit.DataStructs.cDataStructs.ExplicitBitVect object at 0x7dc7c96a4040>,
<rdkit.DataStructs.cDataStructs.ExplicitBitVect object at 0x7dc7c96a40b0>,
```

```
<rdkit.DataStructs.cDataStructs.ExplicitBitVect object at 0x7dc7c96a4120>,
<rdkit.DataStructs.cDataStructs.ExplicitBitVect object at 0x7dc7c96a4190>,
<rdkit.DataStructs.cDataStructs.ExplicitBitVect object at 0x7dc7c96a4200>,
<rdkit.DataStructs.cDataStructs.ExplicitBitVect object at 0x7dc7c96a4270>,
<rdkit.DataStructs.cDataStructs.ExplicitBitVect object at 0x7dc7c96a42e0>,
<rdkit.DataStructs.cDataStructs.ExplicitBitVect object at 0x7dc7c96a4350>,
<rdkit.DataStructs.cDataStructs.ExplicitBitVect object at 0x7dc7c96a43c0>,
<rdkit.DataStructs.cDataStructs.ExplicitBitVect object at 0x7dc7c96a4430>,
<rdkit.DataStructs.cDataStructs.ExplicitBitVect object at 0x7dc7c96a44a0>,
<rdkit.DataStructs.cDataStructs.ExplicitBitVect object at 0x7dc7c96a4510>,
<rdkit.DataStructs.cDataStructs.ExplicitBitVect object at 0x7dc7c96a4580>,
<rdkit.DataStructs.cDataStructs.ExplicitBitVect object at 0x7dc7c96a45f0>,
<rdkit.DataStructs.cDataStructs.ExplicitBitVect object at 0x7dc7c96a4660>,
<rdkit.DataStructs.cDataStructs.ExplicitBitVect object at 0x7dc7c96a46d0>,
<rdkit.DataStructs.cDataStructs.ExplicitBitVect object at 0x7dc7c96a4740>,
<rdkit.DataStructs.cDataStructs.ExplicitBitVect object at 0x7dc7c96a47b0>,
<rdkit.DataStructs.cDataStructs.ExplicitBitVect object at 0x7dc7c96a4820>,
<rdkit.DataStructs.cDataStructs.ExplicitBitVect object at 0x7dc7c96a4890>,
<rdkit.DataStructs.cDataStructs.ExplicitBitVect object at 0x7dc7c96a4900>,
<rdkit.DataStructs.cDataStructs.ExplicitBitVect object at 0x7dc7c96a4970>,
<rdkit.DataStructs.cDataStructs.ExplicitBitVect object at 0x7dc7c96a49e0>,
<rdkit.DataStructs.cDataStructs.ExplicitBitVect object at 0x7dc7c96a4a50>,
<rdkit.DataStructs.cDataStructs.ExplicitBitVect object at 0x7dc7c96a4ac0>,
<rdkit.DataStructs.cDataStructs.ExplicitBitVect object at 0x7dc7c96a4b30>,
<rdkit.DataStructs.cDataStructs.ExplicitBitVect object at 0x7dc7c96a4ba0>,
<rdkit.DataStructs.cDataStructs.ExplicitBitVect object at 0x7dc7c96a4c10>,
<rdkit.DataStructs.cDataStructs.ExplicitBitVect object at 0x7dc7c96a4c80>,
<rdkit.DataStructs.cDataStructs.ExplicitBitVect object at 0x7dc7c96a4cf0>,
<rdkit.DataStructs.cDataStructs.ExplicitBitVect object at 0x7dc7c96a4d60>,
<rdkit.DataStructs.cDataStructs.ExplicitBitVect object at 0x7dc7c96a4dd0>,
<rdkit.DataStructs.cDataStructs.ExplicitBitVect object at 0x7dc7c96a4e40>,
<rdkit.DataStructs.cDataStructs.ExplicitBitVect object at 0x7dc7c96a4eb0>,
<rdkit.DataStructs.cDataStructs.ExplicitBitVect object at 0x7dc7c96a4f20>,
<rdkit.DataStructs.cDataStructs.ExplicitBitVect object at 0x7dc7c96a4f90>,
<rdkit.DataStructs.cDataStructs.ExplicitBitVect object at 0x7dc7c96a5000>,
<rdkit.DataStructs.cDataStructs.ExplicitBitVect object at 0x7dc7c96a5070>,
<rdkit.DataStructs.cDataStructs.ExplicitBitVect object at 0x7dc7c96a50e0>,
<rdkit.DataStructs.cDataStructs.ExplicitBitVect object at 0x7dc7c96a5150>,
<rdkit.DataStructs.cDataStructs.ExplicitBitVect object at 0x7dc7c96a51c0>,
<rdkit.DataStructs.cDataStructs.ExplicitBitVect object at 0x7dc7c96a5230>,
<rdkit.DataStructs.cDataStructs.ExplicitBitVect object at 0x7dc7c96a52a0>,
<rdkit.DataStructs.cDataStructs.ExplicitBitVect object at 0x7dc7c96a5310>,
<rdkit.DataStructs.cDataStructs.ExplicitBitVect object at 0x7dc7c96a5380>,
<rdkit.DataStructs.cDataStructs.ExplicitBitVect object at 0x7dc7c96a53f0>,
<rdkit.DataStructs.cDataStructs.ExplicitBitVect object at 0x7dc7c96a5460>,
<rdkit.DataStructs.cDataStructs.ExplicitBitVect object at 0x7dc7c96a54d0>,
<rdkit.DataStructs.cDataStructs.ExplicitBitVect object at 0x7dc7c96a5540>,
<rdkit.DataStructs.cDataStructs.ExplicitBitVect object at 0x7dc7c96a55b0>,
```

```
<rdkit.DataStructs.cDataStructs.ExplicitBitVect object at 0x7dc7c96a5620>,
<rdkit.DataStructs.cDataStructs.ExplicitBitVect object at 0x7dc7c96a5690>,
<rdkit.DataStructs.cDataStructs.ExplicitBitVect object at 0x7dc7c96a5700>,
<rdkit.DataStructs.cDataStructs.ExplicitBitVect object at 0x7dc7c96a5770>,
<rdkit.DataStructs.cDataStructs.ExplicitBitVect object at 0x7dc7c96a57e0>,
<rdkit.DataStructs.cDataStructs.ExplicitBitVect object at 0x7dc7c96a5850>,
<rdkit.DataStructs.cDataStructs.ExplicitBitVect object at 0x7dc7c96a58c0>,
<rdkit.DataStructs.cDataStructs.ExplicitBitVect object at 0x7dc7c96a5930>,
<rdkit.DataStructs.cDataStructs.ExplicitBitVect object at 0x7dc7c96a59a0>,
<rdkit.DataStructs.cDataStructs.ExplicitBitVect object at 0x7dc7c96a5a10>,
<rdkit.DataStructs.cDataStructs.ExplicitBitVect object at 0x7dc7c96a5a80>,
<rdkit.DataStructs.cDataStructs.ExplicitBitVect object at 0x7dc7c96a5af0>,
<rdkit.DataStructs.cDataStructs.ExplicitBitVect object at 0x7dc7c96a5b60>,
<rdkit.DataStructs.cDataStructs.ExplicitBitVect object at 0x7dc7c96a5bd0>,
<rdkit.DataStructs.cDataStructs.ExplicitBitVect object at 0x7dc7c96a5c40>,
<rdkit.DataStructs.cDataStructs.ExplicitBitVect object at 0x7dc7c96a5cb0>,
<rdkit.DataStructs.cDataStructs.ExplicitBitVect object at 0x7dc7c96a5d20>,
<rdkit.DataStructs.cDataStructs.ExplicitBitVect object at 0x7dc7c96a5d90>,
<rdkit.DataStructs.cDataStructs.ExplicitBitVect object at 0x7dc7c96a5e00>,
<rdkit.DataStructs.cDataStructs.ExplicitBitVect object at 0x7dc7c96a5e70>,
<rdkit.DataStructs.cDataStructs.ExplicitBitVect object at 0x7dc7c96a5ee0>,
<rdkit.DataStructs.cDataStructs.ExplicitBitVect object at 0x7dc7c96a5f50>,
<rdkit.DataStructs.cDataStructs.ExplicitBitVect object at 0x7dc7c96a5fc0>,
<rdkit.DataStructs.cDataStructs.ExplicitBitVect object at 0x7dc7c96a6030>,
<rdkit.DataStructs.cDataStructs.ExplicitBitVect object at 0x7dc7c96a60a0>,
<rdkit.DataStructs.cDataStructs.ExplicitBitVect object at 0x7dc7c96a6110>,
<rdkit.DataStructs.cDataStructs.ExplicitBitVect object at 0x7dc7c96a6180>,
<rdkit.DataStructs.cDataStructs.ExplicitBitVect object at 0x7dc7c96a61f0>,
<rdkit.DataStructs.cDataStructs.ExplicitBitVect object at 0x7dc7c96a6260>,
<rdkit.DataStructs.cDataStructs.ExplicitBitVect object at 0x7dc7c96a62d0>,
<rdkit.DataStructs.cDataStructs.ExplicitBitVect object at 0x7dc7c96a6340>,
<rdkit.DataStructs.cDataStructs.ExplicitBitVect object at 0x7dc7c96a63b0>,
<rdkit.DataStructs.cDataStructs.ExplicitBitVect object at 0x7dc7c96a6420>,
<rdkit.DataStructs.cDataStructs.ExplicitBitVect object at 0x7dc7c96a6490>,
<rdkit.DataStructs.cDataStructs.ExplicitBitVect object at 0x7dc7c96a6500>,
<rdkit.DataStructs.cDataStructs.ExplicitBitVect object at 0x7dc7c96a6570>,
<rdkit.DataStructs.cDataStructs.ExplicitBitVect object at 0x7dc7c96a65e0>,
<rdkit.DataStructs.cDataStructs.ExplicitBitVect object at 0x7dc7c96a6650>,
<rdkit.DataStructs.cDataStructs.ExplicitBitVect object at 0x7dc7c96a66c0>,
<rdkit.DataStructs.cDataStructs.ExplicitBitVect object at 0x7dc7c96a6730>,
<rdkit.DataStructs.cDataStructs.ExplicitBitVect object at 0x7dc7c96a67a0>,
<rdkit.DataStructs.cDataStructs.ExplicitBitVect object at 0x7dc7c96a6810>,
<rdkit.DataStructs.cDataStructs.ExplicitBitVect object at 0x7dc7c96a6880>,
<rdkit.DataStructs.cDataStructs.ExplicitBitVect object at 0x7dc7c96a68f0>,
<rdkit.DataStructs.cDataStructs.ExplicitBitVect object at 0x7dc7c96a6960>,
<rdkit.DataStructs.cDataStructs.ExplicitBitVect object at 0x7dc7c96a69d0>,
<rdkit.DataStructs.cDataStructs.ExplicitBitVect object at 0x7dc7c96a6a40>,
<rdkit.DataStructs.cDataStructs.ExplicitBitVect object at 0x7dc7c96a6ab0>,
```

```
<rdkit.DataStructs.cDataStructs.ExplicitBitVect object at 0x7dc7c96a6b20>,
<rdkit.DataStructs.cDataStructs.ExplicitBitVect object at 0x7dc7c96a6b90>,
<rdkit.DataStructs.cDataStructs.ExplicitBitVect object at 0x7dc7c96a6c00>,
<rdkit.DataStructs.cDataStructs.ExplicitBitVect object at 0x7dc7c96a6c70>,
<rdkit.DataStructs.cDataStructs.ExplicitBitVect object at 0x7dc7c96a6ce0>,
<rdkit.DataStructs.cDataStructs.ExplicitBitVect object at 0x7dc7c96a6d50>,
<rdkit.DataStructs.cDataStructs.ExplicitBitVect object at 0x7dc7c96a6dc0>,
<rdkit.DataStructs.cDataStructs.ExplicitBitVect object at 0x7dc7c96a6e30>,
<rdkit.DataStructs.cDataStructs.ExplicitBitVect object at 0x7dc7c96a6ea0>,
<rdkit.DataStructs.cDataStructs.ExplicitBitVect object at 0x7dc7c96a6f10>,
<rdkit.DataStructs.cDataStructs.ExplicitBitVect object at 0x7dc7c96a6f80>,
<rdkit.DataStructs.cDataStructs.ExplicitBitVect object at 0x7dc7c96a6ff0>,
<rdkit.DataStructs.cDataStructs.ExplicitBitVect object at 0x7dc7c96a7060>,
<rdkit.DataStructs.cDataStructs.ExplicitBitVect object at 0x7dc7c96a70d0>,
<rdkit.DataStructs.cDataStructs.ExplicitBitVect object at 0x7dc7c96a7140>,
<rdkit.DataStructs.cDataStructs.ExplicitBitVect object at 0x7dc7c96a71b0>,
<rdkit.DataStructs.cDataStructs.ExplicitBitVect object at 0x7dc7c96a7220>,
<rdkit.DataStructs.cDataStructs.ExplicitBitVect object at 0x7dc7c96a7290>,
<rdkit.DataStructs.cDataStructs.ExplicitBitVect object at 0x7dc7c96a7300>,
<rdkit.DataStructs.cDataStructs.ExplicitBitVect object at 0x7dc7c96a7370>,
<rdkit.DataStructs.cDataStructs.ExplicitBitVect object at 0x7dc7c96a73e0>,
<rdkit.DataStructs.cDataStructs.ExplicitBitVect object at 0x7dc7c96a7450>,
<rdkit.DataStructs.cDataStructs.ExplicitBitVect object at 0x7dc7c96a74c0>,
<rdkit.DataStructs.cDataStructs.ExplicitBitVect object at 0x7dc7c96a7530>,
<rdkit.DataStructs.cDataStructs.ExplicitBitVect object at 0x7dc7c96a75a0>,
<rdkit.DataStructs.cDataStructs.ExplicitBitVect object at 0x7dc7c96a7610>,
<rdkit.DataStructs.cDataStructs.ExplicitBitVect object at 0x7dc7c96a7680>,
<rdkit.DataStructs.cDataStructs.ExplicitBitVect object at 0x7dc7c96a76f0>,
<rdkit.DataStructs.cDataStructs.ExplicitBitVect object at 0x7dc7c96a7760>,
<rdkit.DataStructs.cDataStructs.ExplicitBitVect object at 0x7dc7c96a77d0>,
<rdkit.DataStructs.cDataStructs.ExplicitBitVect object at 0x7dc7c96a7840>,
<rdkit.DataStructs.cDataStructs.ExplicitBitVect object at 0x7dc7c96a78b0>,
<rdkit.DataStructs.cDataStructs.ExplicitBitVect object at 0x7dc7c96a7920>,
<rdkit.DataStructs.cDataStructs.ExplicitBitVect object at 0x7dc7c96a7990>,
<rdkit.DataStructs.cDataStructs.ExplicitBitVect object at 0x7dc7c96a7a00>,
<rdkit.DataStructs.cDataStructs.ExplicitBitVect object at 0x7dc7c96a7a70>,
<rdkit.DataStructs.cDataStructs.ExplicitBitVect object at 0x7dc7c96a7ae0>,
<rdkit.DataStructs.cDataStructs.ExplicitBitVect object at 0x7dc7c96a7b50>,
<rdkit.DataStructs.cDataStructs.ExplicitBitVect object at 0x7dc7c96a7bc0>,
<rdkit.DataStructs.cDataStructs.ExplicitBitVect object at 0x7dc7c96a7c30>,
<rdkit.DataStructs.cDataStructs.ExplicitBitVect object at 0x7dc7c96a7ca0>,
<rdkit.DataStructs.cDataStructs.ExplicitBitVect object at 0x7dc7c96a7d10>,
<rdkit.DataStructs.cDataStructs.ExplicitBitVect object at 0x7dc7c96a7d80>,
<rdkit.DataStructs.cDataStructs.ExplicitBitVect object at 0x7dc7c96a7df0>,
<rdkit.DataStructs.cDataStructs.ExplicitBitVect object at 0x7dc7c96a7e60>,
<rdkit.DataStructs.cDataStructs.ExplicitBitVect object at 0x7dc7c96a7ed0>,
<rdkit.DataStructs.cDataStructs.ExplicitBitVect object at 0x7dc7c96a7f40>,
<rdkit.DataStructs.cDataStructs.ExplicitBitVect object at 0x7dc7c96a8040>,
```

```
<rdkit.DataStructs.cDataStructs.ExplicitBitVect object at 0x7dc7c96a80b0>,
<rdkit.DataStructs.cDataStructs.ExplicitBitVect object at 0x7dc7c96a8120>,
<rdkit.DataStructs.cDataStructs.ExplicitBitVect object at 0x7dc7c96a8190>,
<rdkit.DataStructs.cDataStructs.ExplicitBitVect object at 0x7dc7c96a8200>,
<rdkit.DataStructs.cDataStructs.ExplicitBitVect object at 0x7dc7c96a8270>,
<rdkit.DataStructs.cDataStructs.ExplicitBitVect object at 0x7dc7c96a82e0>,
<rdkit.DataStructs.cDataStructs.ExplicitBitVect object at 0x7dc7c96a8350>,
<rdkit.DataStructs.cDataStructs.ExplicitBitVect object at 0x7dc7c96a83c0>,
<rdkit.DataStructs.cDataStructs.ExplicitBitVect object at 0x7dc7c96a8430>,
<rdkit.DataStructs.cDataStructs.ExplicitBitVect object at 0x7dc7c96a84a0>,
<rdkit.DataStructs.cDataStructs.ExplicitBitVect object at 0x7dc7c96a8510>,
<rdkit.DataStructs.cDataStructs.ExplicitBitVect object at 0x7dc7c96a8580>,
<rdkit.DataStructs.cDataStructs.ExplicitBitVect object at 0x7dc7c96a85f0>,
<rdkit.DataStructs.cDataStructs.ExplicitBitVect object at 0x7dc7c96a8660>,
<rdkit.DataStructs.cDataStructs.ExplicitBitVect object at 0x7dc7c96a86d0>,
<rdkit.DataStructs.cDataStructs.ExplicitBitVect object at 0x7dc7c96a8740>,
<rdkit.DataStructs.cDataStructs.ExplicitBitVect object at 0x7dc7c96a87b0>,
<rdkit.DataStructs.cDataStructs.ExplicitBitVect object at 0x7dc7c96a8820>,
<rdkit.DataStructs.cDataStructs.ExplicitBitVect object at 0x7dc7c96a8890>,
<rdkit.DataStructs.cDataStructs.ExplicitBitVect object at 0x7dc7c96a8900>,
<rdkit.DataStructs.cDataStructs.ExplicitBitVect object at 0x7dc7c96a8970>,
<rdkit.DataStructs.cDataStructs.ExplicitBitVect object at 0x7dc7c96a89e0>,
<rdkit.DataStructs.cDataStructs.ExplicitBitVect object at 0x7dc7c96a8a50>,
<rdkit.DataStructs.cDataStructs.ExplicitBitVect object at 0x7dc7c96a8ac0>,
<rdkit.DataStructs.cDataStructs.ExplicitBitVect object at 0x7dc7c96a8b30>,
<rdkit.DataStructs.cDataStructs.ExplicitBitVect object at 0x7dc7c96a8ba0>,
<rdkit.DataStructs.cDataStructs.ExplicitBitVect object at 0x7dc7c96a8c10>,
<rdkit.DataStructs.cDataStructs.ExplicitBitVect object at 0x7dc7c96a8c80>,
<rdkit.DataStructs.cDataStructs.ExplicitBitVect object at 0x7dc7c96a8cf0>,
<rdkit.DataStructs.cDataStructs.ExplicitBitVect object at 0x7dc7c96a8d60>,
<rdkit.DataStructs.cDataStructs.ExplicitBitVect object at 0x7dc7c96a8dd0>,
<rdkit.DataStructs.cDataStructs.ExplicitBitVect object at 0x7dc7c96a8e40>,
<rdkit.DataStructs.cDataStructs.ExplicitBitVect object at 0x7dc7c96a8eb0>,
<rdkit.DataStructs.cDataStructs.ExplicitBitVect object at 0x7dc7c96a8f20>,
<rdkit.DataStructs.cDataStructs.ExplicitBitVect object at 0x7dc7c96a8f90>,
<rdkit.DataStructs.cDataStructs.ExplicitBitVect object at 0x7dc7c96a9000>,
<rdkit.DataStructs.cDataStructs.ExplicitBitVect object at 0x7dc7c96a9070>,
<rdkit.DataStructs.cDataStructs.ExplicitBitVect object at 0x7dc7c96a90e0>,
<rdkit.DataStructs.cDataStructs.ExplicitBitVect object at 0x7dc7c96a9150>,
<rdkit.DataStructs.cDataStructs.ExplicitBitVect object at 0x7dc7c96a91c0>,
<rdkit.DataStructs.cDataStructs.ExplicitBitVect object at 0x7dc7c96a9230>,
<rdkit.DataStructs.cDataStructs.ExplicitBitVect object at 0x7dc7c96a92a0>,
<rdkit.DataStructs.cDataStructs.ExplicitBitVect object at 0x7dc7c96a9310>,
<rdkit.DataStructs.cDataStructs.ExplicitBitVect object at 0x7dc7c96a9380>,
<rdkit.DataStructs.cDataStructs.ExplicitBitVect object at 0x7dc7c96a93f0>,
<rdkit.DataStructs.cDataStructs.ExplicitBitVect object at 0x7dc7c96a9460>,
<rdkit.DataStructs.cDataStructs.ExplicitBitVect object at 0x7dc7c96a94d0>,
<rdkit.DataStructs.cDataStructs.ExplicitBitVect object at 0x7dc7c96a9540>,
```

```
<rdkit.DataStructs.cDataStructs.ExplicitBitVect object at 0x7dc7c96a95b0>,
<rdkit.DataStructs.cDataStructs.ExplicitBitVect object at 0x7dc7c96a9620>,
<rdkit.DataStructs.cDataStructs.ExplicitBitVect object at 0x7dc7c96a9690>,
<rdkit.DataStructs.cDataStructs.ExplicitBitVect object at 0x7dc7c96a9700>,
<rdkit.DataStructs.cDataStructs.ExplicitBitVect object at 0x7dc7c96a9770>,
<rdkit.DataStructs.cDataStructs.ExplicitBitVect object at 0x7dc7c96a97e0>,
<rdkit.DataStructs.cDataStructs.ExplicitBitVect object at 0x7dc7c96a9850>,
<rdkit.DataStructs.cDataStructs.ExplicitBitVect object at 0x7dc7c96a98c0>,
<rdkit.DataStructs.cDataStructs.ExplicitBitVect object at 0x7dc7c96a9930>,
<rdkit.DataStructs.cDataStructs.ExplicitBitVect object at 0x7dc7c96a99a0>,
<rdkit.DataStructs.cDataStructs.ExplicitBitVect object at 0x7dc7c96a9a10>,
<rdkit.DataStructs.cDataStructs.ExplicitBitVect object at 0x7dc7c96a9a80>,
<rdkit.DataStructs.cDataStructs.ExplicitBitVect object at 0x7dc7c96a9af0>,
<rdkit.DataStructs.cDataStructs.ExplicitBitVect object at 0x7dc7c96a9b60>,
<rdkit.DataStructs.cDataStructs.ExplicitBitVect object at 0x7dc7c96a9bd0>,
<rdkit.DataStructs.cDataStructs.ExplicitBitVect object at 0x7dc7c96a9c40>,
<rdkit.DataStructs.cDataStructs.ExplicitBitVect object at 0x7dc7c96a9cb0>,
<rdkit.DataStructs.cDataStructs.ExplicitBitVect object at 0x7dc7c96a9d20>,
<rdkit.DataStructs.cDataStructs.ExplicitBitVect object at 0x7dc7c96a9d90>,
<rdkit.DataStructs.cDataStructs.ExplicitBitVect object at 0x7dc7c96a9e00>,
<rdkit.DataStructs.cDataStructs.ExplicitBitVect object at 0x7dc7c96a9e70>,
<rdkit.DataStructs.cDataStructs.ExplicitBitVect object at 0x7dc7c96a9ee0>,
<rdkit.DataStructs.cDataStructs.ExplicitBitVect object at 0x7dc7c96a9f50>,
<rdkit.DataStructs.cDataStructs.ExplicitBitVect object at 0x7dc7c96a9fc0>,
<rdkit.DataStructs.cDataStructs.ExplicitBitVect object at 0x7dc7c96aa030>,
<rdkit.DataStructs.cDataStructs.ExplicitBitVect object at 0x7dc7c96aa0a0>,
<rdkit.DataStructs.cDataStructs.ExplicitBitVect object at 0x7dc7c96aa110>,
<rdkit.DataStructs.cDataStructs.ExplicitBitVect object at 0x7dc7c96aa180>,
<rdkit.DataStructs.cDataStructs.ExplicitBitVect object at 0x7dc7c96aa1f0>,
<rdkit.DataStructs.cDataStructs.ExplicitBitVect object at 0x7dc7c96aa260>,
<rdkit.DataStructs.cDataStructs.ExplicitBitVect object at 0x7dc7c96aa2d0>,
<rdkit.DataStructs.cDataStructs.ExplicitBitVect object at 0x7dc7c96aa340>,
<rdkit.DataStructs.cDataStructs.ExplicitBitVect object at 0x7dc7c96aa3b0>,
<rdkit.DataStructs.cDataStructs.ExplicitBitVect object at 0x7dc7c96aa420>,
<rdkit.DataStructs.cDataStructs.ExplicitBitVect object at 0x7dc7c96aa490>,
<rdkit.DataStructs.cDataStructs.ExplicitBitVect object at 0x7dc7c96aa500>,
<rdkit.DataStructs.cDataStructs.ExplicitBitVect object at 0x7dc7c96aa570>,
<rdkit.DataStructs.cDataStructs.ExplicitBitVect object at 0x7dc7c96aa5e0>,
<rdkit.DataStructs.cDataStructs.ExplicitBitVect object at 0x7dc7c96aa650>,
<rdkit.DataStructs.cDataStructs.ExplicitBitVect object at 0x7dc7c96aa6c0>,
<rdkit.DataStructs.cDataStructs.ExplicitBitVect object at 0x7dc7c96aa730>,
<rdkit.DataStructs.cDataStructs.ExplicitBitVect object at 0x7dc7c96aa7a0>,
<rdkit.DataStructs.cDataStructs.ExplicitBitVect object at 0x7dc7c96aa810>,
<rdkit.DataStructs.cDataStructs.ExplicitBitVect object at 0x7dc7c96aa880>,
<rdkit.DataStructs.cDataStructs.ExplicitBitVect object at 0x7dc7c96aa8f0>,
<rdkit.DataStructs.cDataStructs.ExplicitBitVect object at 0x7dc7c96aa960>,
<rdkit.DataStructs.cDataStructs.ExplicitBitVect object at 0x7dc7c96aa9d0>,
<rdkit.DataStructs.cDataStructs.ExplicitBitVect object at 0x7dc7c96aaa40>,
```

```
<rdkit.DataStructs.cDataStructs.ExplicitBitVect object at 0x7dc7c96aaab0>,
<rdkit.DataStructs.cDataStructs.ExplicitBitVect object at 0x7dc7c96aab20>,
<rdkit.DataStructs.cDataStructs.ExplicitBitVect object at 0x7dc7c96aab90>,
<rdkit.DataStructs.cDataStructs.ExplicitBitVect object at 0x7dc7c96aac00>,
<rdkit.DataStructs.cDataStructs.ExplicitBitVect object at 0x7dc7c96aac70>,
<rdkit.DataStructs.cDataStructs.ExplicitBitVect object at 0x7dc7c96aace0>,
<rdkit.DataStructs.cDataStructs.ExplicitBitVect object at 0x7dc7c96aad50>,
<rdkit.DataStructs.cDataStructs.ExplicitBitVect object at 0x7dc7c96aadc0>,
<rdkit.DataStructs.cDataStructs.ExplicitBitVect object at 0x7dc7c96aae30>,
<rdkit.DataStructs.cDataStructs.ExplicitBitVect object at 0x7dc7c96aaea0>,
<rdkit.DataStructs.cDataStructs.ExplicitBitVect object at 0x7dc7c96aaf10>,
<rdkit.DataStructs.cDataStructs.ExplicitBitVect object at 0x7dc7c96aaf80>,
<rdkit.DataStructs.cDataStructs.ExplicitBitVect object at 0x7dc7c96aaff0>,
<rdkit.DataStructs.cDataStructs.ExplicitBitVect object at 0x7dc7c96ab060>,
<rdkit.DataStructs.cDataStructs.ExplicitBitVect object at 0x7dc7c96ab0d0>,
<rdkit.DataStructs.cDataStructs.ExplicitBitVect object at 0x7dc7c96ab140>,
<rdkit.DataStructs.cDataStructs.ExplicitBitVect object at 0x7dc7c96ab1b0>,
<rdkit.DataStructs.cDataStructs.ExplicitBitVect object at 0x7dc7c96ab220>,
<rdkit.DataStructs.cDataStructs.ExplicitBitVect object at 0x7dc7c96ab290>,
<rdkit.DataStructs.cDataStructs.ExplicitBitVect object at 0x7dc7c96ab300>,
<rdkit.DataStructs.cDataStructs.ExplicitBitVect object at 0x7dc7c96ab370>,
<rdkit.DataStructs.cDataStructs.ExplicitBitVect object at 0x7dc7c96ab3e0>,
<rdkit.DataStructs.cDataStructs.ExplicitBitVect object at 0x7dc7c96ab450>,
<rdkit.DataStructs.cDataStructs.ExplicitBitVect object at 0x7dc7c96ab4c0>,
<rdkit.DataStructs.cDataStructs.ExplicitBitVect object at 0x7dc7c96ab530>,
<rdkit.DataStructs.cDataStructs.ExplicitBitVect object at 0x7dc7c96ab5a0>,
<rdkit.DataStructs.cDataStructs.ExplicitBitVect object at 0x7dc7c96ab610>,
<rdkit.DataStructs.cDataStructs.ExplicitBitVect object at 0x7dc7c96ab680>,
<rdkit.DataStructs.cDataStructs.ExplicitBitVect object at 0x7dc7c96ab6f0>,
<rdkit.DataStructs.cDataStructs.ExplicitBitVect object at 0x7dc7c96ab760>,
<rdkit.DataStructs.cDataStructs.ExplicitBitVect object at 0x7dc7c96ab7d0>,
<rdkit.DataStructs.cDataStructs.ExplicitBitVect object at 0x7dc7c96ab840>,
<rdkit.DataStructs.cDataStructs.ExplicitBitVect object at 0x7dc7c96ab8b0>,
<rdkit.DataStructs.cDataStructs.ExplicitBitVect object at 0x7dc7c96ab920>,
<rdkit.DataStructs.cDataStructs.ExplicitBitVect object at 0x7dc7c96ab990>,
<rdkit.DataStructs.cDataStructs.ExplicitBitVect object at 0x7dc7c96aba00>,
<rdkit.DataStructs.cDataStructs.ExplicitBitVect object at 0x7dc7c96aba70>,
<rdkit.DataStructs.cDataStructs.ExplicitBitVect object at 0x7dc7c96abae0>,
<rdkit.DataStructs.cDataStructs.ExplicitBitVect object at 0x7dc7c96abb50>,
<rdkit.DataStructs.cDataStructs.ExplicitBitVect object at 0x7dc7c96abbc0>,
<rdkit.DataStructs.cDataStructs.ExplicitBitVect object at 0x7dc7c96abc30>,
<rdkit.DataStructs.cDataStructs.ExplicitBitVect object at 0x7dc7c96abca0>,
<rdkit.DataStructs.cDataStructs.ExplicitBitVect object at 0x7dc7c96abd10>,
<rdkit.DataStructs.cDataStructs.ExplicitBitVect object at 0x7dc7c96abd80>,
<rdkit.DataStructs.cDataStructs.ExplicitBitVect object at 0x7dc7c96abdf0>,
<rdkit.DataStructs.cDataStructs.ExplicitBitVect object at 0x7dc7c96abe60>,
<rdkit.DataStructs.cDataStructs.ExplicitBitVect object at 0x7dc7c96abed0>,
<rdkit.DataStructs.cDataStructs.ExplicitBitVect object at 0x7dc7c96abf40>,
```

```
<rdkit.DataStructs.cDataStructs.ExplicitBitVect object at 0x7dc7c96ac040>,
<rdkit.DataStructs.cDataStructs.ExplicitBitVect object at 0x7dc7c96ac0b0>,
<rdkit.DataStructs.cDataStructs.ExplicitBitVect object at 0x7dc7c96ac120>,
<rdkit.DataStructs.cDataStructs.ExplicitBitVect object at 0x7dc7c96ac190>,
<rdkit.DataStructs.cDataStructs.ExplicitBitVect object at 0x7dc7c96ac200>,
<rdkit.DataStructs.cDataStructs.ExplicitBitVect object at 0x7dc7c96ac270>,
<rdkit.DataStructs.cDataStructs.ExplicitBitVect object at 0x7dc7c96ac2e0>,
<rdkit.DataStructs.cDataStructs.ExplicitBitVect object at 0x7dc7c96ac350>,
<rdkit.DataStructs.cDataStructs.ExplicitBitVect object at 0x7dc7c96ac3c0>,
<rdkit.DataStructs.cDataStructs.ExplicitBitVect object at 0x7dc7c96ac430>,
<rdkit.DataStructs.cDataStructs.ExplicitBitVect object at 0x7dc7c96ac4a0>,
<rdkit.DataStructs.cDataStructs.ExplicitBitVect object at 0x7dc7c96ac510>,
<rdkit.DataStructs.cDataStructs.ExplicitBitVect object at 0x7dc7c96ac580>,
<rdkit.DataStructs.cDataStructs.ExplicitBitVect object at 0x7dc7c96ac5f0>,
<rdkit.DataStructs.cDataStructs.ExplicitBitVect object at 0x7dc7c96ac660>,
<rdkit.DataStructs.cDataStructs.ExplicitBitVect object at 0x7dc7c96ac6d0>,
<rdkit.DataStructs.cDataStructs.ExplicitBitVect object at 0x7dc7c96ac740>,
<rdkit.DataStructs.cDataStructs.ExplicitBitVect object at 0x7dc7c96ac7b0>,
<rdkit.DataStructs.cDataStructs.ExplicitBitVect object at 0x7dc7c96ac820>,
<rdkit.DataStructs.cDataStructs.ExplicitBitVect object at 0x7dc7c96ac890>,
<rdkit.DataStructs.cDataStructs.ExplicitBitVect object at 0x7dc7c96ac900>,
<rdkit.DataStructs.cDataStructs.ExplicitBitVect object at 0x7dc7c96ac970>,
<rdkit.DataStructs.cDataStructs.ExplicitBitVect object at 0x7dc7c96ac9e0>,
<rdkit.DataStructs.cDataStructs.ExplicitBitVect object at 0x7dc7c96aca50>,
<rdkit.DataStructs.cDataStructs.ExplicitBitVect object at 0x7dc7c96acac0>,
<rdkit.DataStructs.cDataStructs.ExplicitBitVect object at 0x7dc7c96acb30>,
<rdkit.DataStructs.cDataStructs.ExplicitBitVect object at 0x7dc7c96acba0>,
<rdkit.DataStructs.cDataStructs.ExplicitBitVect object at 0x7dc7c96acc10>,
<rdkit.DataStructs.cDataStructs.ExplicitBitVect object at 0x7dc7c96acc80>,
<rdkit.DataStructs.cDataStructs.ExplicitBitVect object at 0x7dc7c96accf0>,
<rdkit.DataStructs.cDataStructs.ExplicitBitVect object at 0x7dc7c96acd60>,
<rdkit.DataStructs.cDataStructs.ExplicitBitVect object at 0x7dc7c96acdd0>,
<rdkit.DataStructs.cDataStructs.ExplicitBitVect object at 0x7dc7c96ace40>,
<rdkit.DataStructs.cDataStructs.ExplicitBitVect object at 0x7dc7c96aceb0>,
<rdkit.DataStructs.cDataStructs.ExplicitBitVect object at 0x7dc7c96acf20>,
<rdkit.DataStructs.cDataStructs.ExplicitBitVect object at 0x7dc7c96acf90>,
<rdkit.DataStructs.cDataStructs.ExplicitBitVect object at 0x7dc7c96ad000>,
<rdkit.DataStructs.cDataStructs.ExplicitBitVect object at 0x7dc7c96ad070>,
<rdkit.DataStructs.cDataStructs.ExplicitBitVect object at 0x7dc7c96ad0e0>,
<rdkit.DataStructs.cDataStructs.ExplicitBitVect object at 0x7dc7c96ad150>,
<rdkit.DataStructs.cDataStructs.ExplicitBitVect object at 0x7dc7c96ad1c0>,
<rdkit.DataStructs.cDataStructs.ExplicitBitVect object at 0x7dc7c96ad230>,
<rdkit.DataStructs.cDataStructs.ExplicitBitVect object at 0x7dc7c96ad2a0>,
<rdkit.DataStructs.cDataStructs.ExplicitBitVect object at 0x7dc7c96ad310>,
<rdkit.DataStructs.cDataStructs.ExplicitBitVect object at 0x7dc7c96ad380>,
<rdkit.DataStructs.cDataStructs.ExplicitBitVect object at 0x7dc7c96ad3f0>,
<rdkit.DataStructs.cDataStructs.ExplicitBitVect object at 0x7dc7c96ad460>,
<rdkit.DataStructs.cDataStructs.ExplicitBitVect object at 0x7dc7c96ad4d0>,
```

```
<rdkit.DataStructs.cDataStructs.ExplicitBitVect object at 0x7dc7c96ad540>,
<rdkit.DataStructs.cDataStructs.ExplicitBitVect object at 0x7dc7c96ad5b0>,
<rdkit.DataStructs.cDataStructs.ExplicitBitVect object at 0x7dc7c96ad620>,
<rdkit.DataStructs.cDataStructs.ExplicitBitVect object at 0x7dc7c96ad690>,
<rdkit.DataStructs.cDataStructs.ExplicitBitVect object at 0x7dc7c96ad700>,
<rdkit.DataStructs.cDataStructs.ExplicitBitVect object at 0x7dc7c96ad770>,
<rdkit.DataStructs.cDataStructs.ExplicitBitVect object at 0x7dc7c96ad7e0>,
<rdkit.DataStructs.cDataStructs.ExplicitBitVect object at 0x7dc7c96ad850>,
<rdkit.DataStructs.cDataStructs.ExplicitBitVect object at 0x7dc7c96ad8c0>,
<rdkit.DataStructs.cDataStructs.ExplicitBitVect object at 0x7dc7c96ad930>,
<rdkit.DataStructs.cDataStructs.ExplicitBitVect object at 0x7dc7c96ad9a0>,
<rdkit.DataStructs.cDataStructs.ExplicitBitVect object at 0x7dc7c96ada10>,
<rdkit.DataStructs.cDataStructs.ExplicitBitVect object at 0x7dc7c96ada80>,
<rdkit.DataStructs.cDataStructs.ExplicitBitVect object at 0x7dc7c96adaf0>,
<rdkit.DataStructs.cDataStructs.ExplicitBitVect object at 0x7dc7c96adb60>,
<rdkit.DataStructs.cDataStructs.ExplicitBitVect object at 0x7dc7c96adbd0>,
<rdkit.DataStructs.cDataStructs.ExplicitBitVect object at 0x7dc7c96adc40>,
<rdkit.DataStructs.cDataStructs.ExplicitBitVect object at 0x7dc7c96adcb0>,
<rdkit.DataStructs.cDataStructs.ExplicitBitVect object at 0x7dc7c96add20>,
<rdkit.DataStructs.cDataStructs.ExplicitBitVect object at 0x7dc7c96add90>,
<rdkit.DataStructs.cDataStructs.ExplicitBitVect object at 0x7dc7c96ade00>,
<rdkit.DataStructs.cDataStructs.ExplicitBitVect object at 0x7dc7c96ade70>,
<rdkit.DataStructs.cDataStructs.ExplicitBitVect object at 0x7dc7c96adee0>,
<rdkit.DataStructs.cDataStructs.ExplicitBitVect object at 0x7dc7c96adf50>,
<rdkit.DataStructs.cDataStructs.ExplicitBitVect object at 0x7dc7c96adfc0>,
<rdkit.DataStructs.cDataStructs.ExplicitBitVect object at 0x7dc7c96ae030>,
<rdkit.DataStructs.cDataStructs.ExplicitBitVect object at 0x7dc7c96ae0a0>,
<rdkit.DataStructs.cDataStructs.ExplicitBitVect object at 0x7dc7c96ae110>,
<rdkit.DataStructs.cDataStructs.ExplicitBitVect object at 0x7dc7c96ae180>,
<rdkit.DataStructs.cDataStructs.ExplicitBitVect object at 0x7dc7c96ae1f0>,
<rdkit.DataStructs.cDataStructs.ExplicitBitVect object at 0x7dc7c96ae260>,
<rdkit.DataStructs.cDataStructs.ExplicitBitVect object at 0x7dc7c96ae2d0>,
<rdkit.DataStructs.cDataStructs.ExplicitBitVect object at 0x7dc7c96ae340>,
<rdkit.DataStructs.cDataStructs.ExplicitBitVect object at 0x7dc7c96ae3b0>,
<rdkit.DataStructs.cDataStructs.ExplicitBitVect object at 0x7dc7c96ae420>,
<rdkit.DataStructs.cDataStructs.ExplicitBitVect object at 0x7dc7c96ae490>,
<rdkit.DataStructs.cDataStructs.ExplicitBitVect object at 0x7dc7c96ae500>,
<rdkit.DataStructs.cDataStructs.ExplicitBitVect object at 0x7dc7c96ae570>,
<rdkit.DataStructs.cDataStructs.ExplicitBitVect object at 0x7dc7c96ae5e0>,
<rdkit.DataStructs.cDataStructs.ExplicitBitVect object at 0x7dc7c96ae650>,
<rdkit.DataStructs.cDataStructs.ExplicitBitVect object at 0x7dc7c96ae6c0>,
<rdkit.DataStructs.cDataStructs.ExplicitBitVect object at 0x7dc7c96ae730>,
<rdkit.DataStructs.cDataStructs.ExplicitBitVect object at 0x7dc7c96ae7a0>,
<rdkit.DataStructs.cDataStructs.ExplicitBitVect object at 0x7dc7c96ae810>,
<rdkit.DataStructs.cDataStructs.ExplicitBitVect object at 0x7dc7c96ae880>,
<rdkit.DataStructs.cDataStructs.ExplicitBitVect object at 0x7dc7c96ae8f0>,
<rdkit.DataStructs.cDataStructs.ExplicitBitVect object at 0x7dc7c96ae960>,
<rdkit.DataStructs.cDataStructs.ExplicitBitVect object at 0x7dc7c96ae9d0>,
```

```
<rdkit.DataStructs.cDataStructs.ExplicitBitVect object at 0x7dc7c96aea40>,
<rdkit.DataStructs.cDataStructs.ExplicitBitVect object at 0x7dc7c96aeab0>,
<rdkit.DataStructs.cDataStructs.ExplicitBitVect object at 0x7dc7c96aeb20>,
<rdkit.DataStructs.cDataStructs.ExplicitBitVect object at 0x7dc7c96aeb90>,
<rdkit.DataStructs.cDataStructs.ExplicitBitVect object at 0x7dc7c96aec00>,
<rdkit.DataStructs.cDataStructs.ExplicitBitVect object at 0x7dc7c96aec70>,
<rdkit.DataStructs.cDataStructs.ExplicitBitVect object at 0x7dc7c96aece0>,
<rdkit.DataStructs.cDataStructs.ExplicitBitVect object at 0x7dc7c96aed50>,
<rdkit.DataStructs.cDataStructs.ExplicitBitVect object at 0x7dc7c96aedc0>,
<rdkit.DataStructs.cDataStructs.ExplicitBitVect object at 0x7dc7c96aee30>,
<rdkit.DataStructs.cDataStructs.ExplicitBitVect object at 0x7dc7c96aeea0>,
<rdkit.DataStructs.cDataStructs.ExplicitBitVect object at 0x7dc7c96aef10>,
<rdkit.DataStructs.cDataStructs.ExplicitBitVect object at 0x7dc7c96aef80>,
<rdkit.DataStructs.cDataStructs.ExplicitBitVect object at 0x7dc7c96aeff0>,
<rdkit.DataStructs.cDataStructs.ExplicitBitVect object at 0x7dc7c96af060>,
<rdkit.DataStructs.cDataStructs.ExplicitBitVect object at 0x7dc7c96af0d0>,
<rdkit.DataStructs.cDataStructs.ExplicitBitVect object at 0x7dc7c96af140>,
<rdkit.DataStructs.cDataStructs.ExplicitBitVect object at 0x7dc7c96af1b0>,
<rdkit.DataStructs.cDataStructs.ExplicitBitVect object at 0x7dc7c96af220>,
<rdkit.DataStructs.cDataStructs.ExplicitBitVect object at 0x7dc7c96af290>,
<rdkit.DataStructs.cDataStructs.ExplicitBitVect object at 0x7dc7c96af300>,
<rdkit.DataStructs.cDataStructs.ExplicitBitVect object at 0x7dc7c96af370>,
<rdkit.DataStructs.cDataStructs.ExplicitBitVect object at 0x7dc7c96af3e0>,
<rdkit.DataStructs.cDataStructs.ExplicitBitVect object at 0x7dc7c96af450>,
<rdkit.DataStructs.cDataStructs.ExplicitBitVect object at 0x7dc7c96af4c0>,
<rdkit.DataStructs.cDataStructs.ExplicitBitVect object at 0x7dc7c96af530>,
<rdkit.DataStructs.cDataStructs.ExplicitBitVect object at 0x7dc7c96af5a0>,
<rdkit.DataStructs.cDataStructs.ExplicitBitVect object at 0x7dc7c96af610>,
<rdkit.DataStructs.cDataStructs.ExplicitBitVect object at 0x7dc7c96af680>,
<rdkit.DataStructs.cDataStructs.ExplicitBitVect object at 0x7dc7c96af6f0>,
<rdkit.DataStructs.cDataStructs.ExplicitBitVect object at 0x7dc7c96af760>,
<rdkit.DataStructs.cDataStructs.ExplicitBitVect object at 0x7dc7c96af7d0>,
<rdkit.DataStructs.cDataStructs.ExplicitBitVect object at 0x7dc7c96af840>,
<rdkit.DataStructs.cDataStructs.ExplicitBitVect object at 0x7dc7c96af8b0>,
<rdkit.DataStructs.cDataStructs.ExplicitBitVect object at 0x7dc7c96af920>,
<rdkit.DataStructs.cDataStructs.ExplicitBitVect object at 0x7dc7c96af990>,
<rdkit.DataStructs.cDataStructs.ExplicitBitVect object at 0x7dc7c96afa00>,
<rdkit.DataStructs.cDataStructs.ExplicitBitVect object at 0x7dc7c96afa70>,
<rdkit.DataStructs.cDataStructs.ExplicitBitVect object at 0x7dc7c96afae0>,
<rdkit.DataStructs.cDataStructs.ExplicitBitVect object at 0x7dc7c96afb50>,
<rdkit.DataStructs.cDataStructs.ExplicitBitVect object at 0x7dc7c96afbc0>,
<rdkit.DataStructs.cDataStructs.ExplicitBitVect object at 0x7dc7c96afc30>,
<rdkit.DataStructs.cDataStructs.ExplicitBitVect object at 0x7dc7c96afca0>,
<rdkit.DataStructs.cDataStructs.ExplicitBitVect object at 0x7dc7c96afd10>,
<rdkit.DataStructs.cDataStructs.ExplicitBitVect object at 0x7dc7c96afd80>,
<rdkit.DataStructs.cDataStructs.ExplicitBitVect object at 0x7dc7c96afdf0>,
<rdkit.DataStructs.cDataStructs.ExplicitBitVect object at 0x7dc7c96afe60>,
<rdkit.DataStructs.cDataStructs.ExplicitBitVect object at 0x7dc7c96afed0>,
```

```
<rdkit.DataStructs.cDataStructs.ExplicitBitVect object at 0x7dc7c96aff40>,
<rdkit.DataStructs.cDataStructs.ExplicitBitVect object at 0x7dc7c96b0040>,
<rdkit.DataStructs.cDataStructs.ExplicitBitVect object at 0x7dc7c96b00b0>,
<rdkit.DataStructs.cDataStructs.ExplicitBitVect object at 0x7dc7c96b0120>,
<rdkit.DataStructs.cDataStructs.ExplicitBitVect object at 0x7dc7c96b0190>,
<rdkit.DataStructs.cDataStructs.ExplicitBitVect object at 0x7dc7c96b0200>,
<rdkit.DataStructs.cDataStructs.ExplicitBitVect object at 0x7dc7c96b0270>,
<rdkit.DataStructs.cDataStructs.ExplicitBitVect object at 0x7dc7c96b02e0>,
<rdkit.DataStructs.cDataStructs.ExplicitBitVect object at 0x7dc7c96b0350>,
<rdkit.DataStructs.cDataStructs.ExplicitBitVect object at 0x7dc7c96b03c0>,
<rdkit.DataStructs.cDataStructs.ExplicitBitVect object at 0x7dc7c96b0430>,
<rdkit.DataStructs.cDataStructs.ExplicitBitVect object at 0x7dc7c96b04a0>,
<rdkit.DataStructs.cDataStructs.ExplicitBitVect object at 0x7dc7c96b0510>,
<rdkit.DataStructs.cDataStructs.ExplicitBitVect object at 0x7dc7c96b0580>,
<rdkit.DataStructs.cDataStructs.ExplicitBitVect object at 0x7dc7c96b05f0>,
<rdkit.DataStructs.cDataStructs.ExplicitBitVect object at 0x7dc7c96b0660>,
<rdkit.DataStructs.cDataStructs.ExplicitBitVect object at 0x7dc7c96b06d0>,
<rdkit.DataStructs.cDataStructs.ExplicitBitVect object at 0x7dc7c96b0740>,
<rdkit.DataStructs.cDataStructs.ExplicitBitVect object at 0x7dc7c96b07b0>,
<rdkit.DataStructs.cDataStructs.ExplicitBitVect object at 0x7dc7c96b0820>,
<rdkit.DataStructs.cDataStructs.ExplicitBitVect object at 0x7dc7c96b0890>,
<rdkit.DataStructs.cDataStructs.ExplicitBitVect object at 0x7dc7c96b0900>,
<rdkit.DataStructs.cDataStructs.ExplicitBitVect object at 0x7dc7c96b0970>,
<rdkit.DataStructs.cDataStructs.ExplicitBitVect object at 0x7dc7c96b09e0>,
<rdkit.DataStructs.cDataStructs.ExplicitBitVect object at 0x7dc7c96b0a50>,
<rdkit.DataStructs.cDataStructs.ExplicitBitVect object at 0x7dc7c96b0ac0>,
<rdkit.DataStructs.cDataStructs.ExplicitBitVect object at 0x7dc7c96b0b30>,
<rdkit.DataStructs.cDataStructs.ExplicitBitVect object at 0x7dc7c96b0ba0>,
<rdkit.DataStructs.cDataStructs.ExplicitBitVect object at 0x7dc7c96b0c10>,
<rdkit.DataStructs.cDataStructs.ExplicitBitVect object at 0x7dc7c96b0c80>,
<rdkit.DataStructs.cDataStructs.ExplicitBitVect object at 0x7dc7c96b0cf0>,
<rdkit.DataStructs.cDataStructs.ExplicitBitVect object at 0x7dc7c96b0d60>,
<rdkit.DataStructs.cDataStructs.ExplicitBitVect object at 0x7dc7c96b0dd0>,
<rdkit.DataStructs.cDataStructs.ExplicitBitVect object at 0x7dc7c96b0e40>,
<rdkit.DataStructs.cDataStructs.ExplicitBitVect object at 0x7dc7c96b0eb0>,
<rdkit.DataStructs.cDataStructs.ExplicitBitVect object at 0x7dc7c96b0f20>,
<rdkit.DataStructs.cDataStructs.ExplicitBitVect object at 0x7dc7c96b0f90>,
<rdkit.DataStructs.cDataStructs.ExplicitBitVect object at 0x7dc7c96b1000>,
<rdkit.DataStructs.cDataStructs.ExplicitBitVect object at 0x7dc7c96b1070>,
<rdkit.DataStructs.cDataStructs.ExplicitBitVect object at 0x7dc7c96b10e0>,
<rdkit.DataStructs.cDataStructs.ExplicitBitVect object at 0x7dc7c96b1150>,
<rdkit.DataStructs.cDataStructs.ExplicitBitVect object at 0x7dc7c96b11c0>,
<rdkit.DataStructs.cDataStructs.ExplicitBitVect object at 0x7dc7c96b1230>,
<rdkit.DataStructs.cDataStructs.ExplicitBitVect object at 0x7dc7c96b12a0>,
<rdkit.DataStructs.cDataStructs.ExplicitBitVect object at 0x7dc7c96b1310>,
<rdkit.DataStructs.cDataStructs.ExplicitBitVect object at 0x7dc7c96b1380>,
<rdkit.DataStructs.cDataStructs.ExplicitBitVect object at 0x7dc7c96b13f0>,
<rdkit.DataStructs.cDataStructs.ExplicitBitVect object at 0x7dc7c96b1460>,
```

```
<rdkit.DataStructs.cDataStructs.ExplicitBitVect object at 0x7dc7c96b14d0>,
<rdkit.DataStructs.cDataStructs.ExplicitBitVect object at 0x7dc7c96b1540>,
<rdkit.DataStructs.cDataStructs.ExplicitBitVect object at 0x7dc7c96b15b0>,
<rdkit.DataStructs.cDataStructs.ExplicitBitVect object at 0x7dc7c96b1620>,
<rdkit.DataStructs.cDataStructs.ExplicitBitVect object at 0x7dc7c96b1690>,
<rdkit.DataStructs.cDataStructs.ExplicitBitVect object at 0x7dc7c96b1700>,
<rdkit.DataStructs.cDataStructs.ExplicitBitVect object at 0x7dc7c96b1770>,
<rdkit.DataStructs.cDataStructs.ExplicitBitVect object at 0x7dc7c96b17e0>,
<rdkit.DataStructs.cDataStructs.ExplicitBitVect object at 0x7dc7c96b1850>,
<rdkit.DataStructs.cDataStructs.ExplicitBitVect object at 0x7dc7c96b18c0>,
<rdkit.DataStructs.cDataStructs.ExplicitBitVect object at 0x7dc7c96b1930>,
<rdkit.DataStructs.cDataStructs.ExplicitBitVect object at 0x7dc7c96b19a0>,
<rdkit.DataStructs.cDataStructs.ExplicitBitVect object at 0x7dc7c96b1a10>,
<rdkit.DataStructs.cDataStructs.ExplicitBitVect object at 0x7dc7c96b1a80>,
<rdkit.DataStructs.cDataStructs.ExplicitBitVect object at 0x7dc7c96b1af0>,
<rdkit.DataStructs.cDataStructs.ExplicitBitVect object at 0x7dc7c96b1b60>,
<rdkit.DataStructs.cDataStructs.ExplicitBitVect object at 0x7dc7c96b1bd0>,
<rdkit.DataStructs.cDataStructs.ExplicitBitVect object at 0x7dc7c96b1c40>,
<rdkit.DataStructs.cDataStructs.ExplicitBitVect object at 0x7dc7c96b1cb0>,
<rdkit.DataStructs.cDataStructs.ExplicitBitVect object at 0x7dc7c96b1d20>,
<rdkit.DataStructs.cDataStructs.ExplicitBitVect object at 0x7dc7c96b1d90>,
<rdkit.DataStructs.cDataStructs.ExplicitBitVect object at 0x7dc7c96b1e00>,
<rdkit.DataStructs.cDataStructs.ExplicitBitVect object at 0x7dc7c96b1e70>,
<rdkit.DataStructs.cDataStructs.ExplicitBitVect object at 0x7dc7c96b1ee0>,
<rdkit.DataStructs.cDataStructs.ExplicitBitVect object at 0x7dc7c96b1f50>,
<rdkit.DataStructs.cDataStructs.ExplicitBitVect object at 0x7dc7c96b1fc0>,
<rdkit.DataStructs.cDataStructs.ExplicitBitVect object at 0x7dc7c96b2030>,
<rdkit.DataStructs.cDataStructs.ExplicitBitVect object at 0x7dc7c96b20a0>,
<rdkit.DataStructs.cDataStructs.ExplicitBitVect object at 0x7dc7c96b2110>,
<rdkit.DataStructs.cDataStructs.ExplicitBitVect object at 0x7dc7c96b2180>,
<rdkit.DataStructs.cDataStructs.ExplicitBitVect object at 0x7dc7c96b21f0>,
<rdkit.DataStructs.cDataStructs.ExplicitBitVect object at 0x7dc7c96b2260>,
<rdkit.DataStructs.cDataStructs.ExplicitBitVect object at 0x7dc7c96b22d0>,
<rdkit.DataStructs.cDataStructs.ExplicitBitVect object at 0x7dc7c96b2340>,
<rdkit.DataStructs.cDataStructs.ExplicitBitVect object at 0x7dc7c96b23b0>,
<rdkit.DataStructs.cDataStructs.ExplicitBitVect object at 0x7dc7c96b2420>,
<rdkit.DataStructs.cDataStructs.ExplicitBitVect object at 0x7dc7c96b2490>,
<rdkit.DataStructs.cDataStructs.ExplicitBitVect object at 0x7dc7c96b2500>,
<rdkit.DataStructs.cDataStructs.ExplicitBitVect object at 0x7dc7c96b2570>,
<rdkit.DataStructs.cDataStructs.ExplicitBitVect object at 0x7dc7c96b25e0>,
<rdkit.DataStructs.cDataStructs.ExplicitBitVect object at 0x7dc7c96b2650>,
<rdkit.DataStructs.cDataStructs.ExplicitBitVect object at 0x7dc7c96b26c0>,
<rdkit.DataStructs.cDataStructs.ExplicitBitVect object at 0x7dc7c96b2730>,
<rdkit.DataStructs.cDataStructs.ExplicitBitVect object at 0x7dc7c96b27a0>,
<rdkit.DataStructs.cDataStructs.ExplicitBitVect object at 0x7dc7c96b2810>,
<rdkit.DataStructs.cDataStructs.ExplicitBitVect object at 0x7dc7c96b2880>,
<rdkit.DataStructs.cDataStructs.ExplicitBitVect object at 0x7dc7c96b28f0>,
<rdkit.DataStructs.cDataStructs.ExplicitBitVect object at 0x7dc7c96b2960>,
```

```
<rdkit.DataStructs.cDataStructs.ExplicitBitVect object at 0x7dc7c96b29d0>,
<rdkit.DataStructs.cDataStructs.ExplicitBitVect object at 0x7dc7c96b2a40>,
<rdkit.DataStructs.cDataStructs.ExplicitBitVect object at 0x7dc7c96b2ab0>,
<rdkit.DataStructs.cDataStructs.ExplicitBitVect object at 0x7dc7c96b2b20>,
<rdkit.DataStructs.cDataStructs.ExplicitBitVect object at 0x7dc7c96b2b90>,
<rdkit.DataStructs.cDataStructs.ExplicitBitVect object at 0x7dc7c96b2c00>,
<rdkit.DataStructs.cDataStructs.ExplicitBitVect object at 0x7dc7c96b2c70>,
<rdkit.DataStructs.cDataStructs.ExplicitBitVect object at 0x7dc7c96b2ce0>,
<rdkit.DataStructs.cDataStructs.ExplicitBitVect object at 0x7dc7c96b2d50>,
<rdkit.DataStructs.cDataStructs.ExplicitBitVect object at 0x7dc7c96b2dc0>,
<rdkit.DataStructs.cDataStructs.ExplicitBitVect object at 0x7dc7c96b2e30>,
<rdkit.DataStructs.cDataStructs.ExplicitBitVect object at 0x7dc7c96b2ea0>,
<rdkit.DataStructs.cDataStructs.ExplicitBitVect object at 0x7dc7c96b2f10>,
<rdkit.DataStructs.cDataStructs.ExplicitBitVect object at 0x7dc7c96b2f80>,
<rdkit.DataStructs.cDataStructs.ExplicitBitVect object at 0x7dc7c96b2ff0>,
<rdkit.DataStructs.cDataStructs.ExplicitBitVect object at 0x7dc7c96b3060>,
<rdkit.DataStructs.cDataStructs.ExplicitBitVect object at 0x7dc7c96b30d0>,
<rdkit.DataStructs.cDataStructs.ExplicitBitVect object at 0x7dc7c96b3140>,
<rdkit.DataStructs.cDataStructs.ExplicitBitVect object at 0x7dc7c96b31b0>,
<rdkit.DataStructs.cDataStructs.ExplicitBitVect object at 0x7dc7c96b3220>,
<rdkit.DataStructs.cDataStructs.ExplicitBitVect object at 0x7dc7c96b3290>,
<rdkit.DataStructs.cDataStructs.ExplicitBitVect object at 0x7dc7c96b3300>,
<rdkit.DataStructs.cDataStructs.ExplicitBitVect object at 0x7dc7c96b3370>,
<rdkit.DataStructs.cDataStructs.ExplicitBitVect object at 0x7dc7c96b33e0>,
<rdkit.DataStructs.cDataStructs.ExplicitBitVect object at 0x7dc7c96b3450>,
<rdkit.DataStructs.cDataStructs.ExplicitBitVect object at 0x7dc7c96b34c0>,
<rdkit.DataStructs.cDataStructs.ExplicitBitVect object at 0x7dc7c96b3530>,
<rdkit.DataStructs.cDataStructs.ExplicitBitVect object at 0x7dc7c96b35a0>,
<rdkit.DataStructs.cDataStructs.ExplicitBitVect object at 0x7dc7c96b3610>,
<rdkit.DataStructs.cDataStructs.ExplicitBitVect object at 0x7dc7c96b3680>,
<rdkit.DataStructs.cDataStructs.ExplicitBitVect object at 0x7dc7c96b36f0>,
<rdkit.DataStructs.cDataStructs.ExplicitBitVect object at 0x7dc7c96b3760>,
<rdkit.DataStructs.cDataStructs.ExplicitBitVect object at 0x7dc7c96b37d0>,
<rdkit.DataStructs.cDataStructs.ExplicitBitVect object at 0x7dc7c96b3840>,
<rdkit.DataStructs.cDataStructs.ExplicitBitVect object at 0x7dc7c96b38b0>,
<rdkit.DataStructs.cDataStructs.ExplicitBitVect object at 0x7dc7c96b3920>,
<rdkit.DataStructs.cDataStructs.ExplicitBitVect object at 0x7dc7c96b3990>,
<rdkit.DataStructs.cDataStructs.ExplicitBitVect object at 0x7dc7c96b3a00>,
<rdkit.DataStructs.cDataStructs.ExplicitBitVect object at 0x7dc7c96b3a70>,
<rdkit.DataStructs.cDataStructs.ExplicitBitVect object at 0x7dc7c96b3ae0>,
<rdkit.DataStructs.cDataStructs.ExplicitBitVect object at 0x7dc7c96b3b50>,
<rdkit.DataStructs.cDataStructs.ExplicitBitVect object at 0x7dc7c96b3bc0>,
<rdkit.DataStructs.cDataStructs.ExplicitBitVect object at 0x7dc7c96b3c30>,
<rdkit.DataStructs.cDataStructs.ExplicitBitVect object at 0x7dc7c96b3ca0>,
<rdkit.DataStructs.cDataStructs.ExplicitBitVect object at 0x7dc7c96b3d10>,
<rdkit.DataStructs.cDataStructs.ExplicitBitVect object at 0x7dc7c96b3d80>,
<rdkit.DataStructs.cDataStructs.ExplicitBitVect object at 0x7dc7c96b3df0>,
<rdkit.DataStructs.cDataStructs.ExplicitBitVect object at 0x7dc7c96b3e60>,
```

```
<rdkit.DataStructs.cDataStructs.ExplicitBitVect object at 0x7dc7c96b3ed0>,
<rdkit.DataStructs.cDataStructs.ExplicitBitVect object at 0x7dc7c96b3f40>,
<rdkit.DataStructs.cDataStructs.ExplicitBitVect object at 0x7dc7c96b4040>,
<rdkit.DataStructs.cDataStructs.ExplicitBitVect object at 0x7dc7c96b40b0>,
<rdkit.DataStructs.cDataStructs.ExplicitBitVect object at 0x7dc7c96b4120>,
<rdkit.DataStructs.cDataStructs.ExplicitBitVect object at 0x7dc7c96b4190>,
<rdkit.DataStructs.cDataStructs.ExplicitBitVect object at 0x7dc7c96b4200>,
<rdkit.DataStructs.cDataStructs.ExplicitBitVect object at 0x7dc7c96b4270>,
<rdkit.DataStructs.cDataStructs.ExplicitBitVect object at 0x7dc7c96b42e0>,
<rdkit.DataStructs.cDataStructs.ExplicitBitVect object at 0x7dc7c96b4350>,
<rdkit.DataStructs.cDataStructs.ExplicitBitVect object at 0x7dc7c96b43c0>,
<rdkit.DataStructs.cDataStructs.ExplicitBitVect object at 0x7dc7c96b4430>,
<rdkit.DataStructs.cDataStructs.ExplicitBitVect object at 0x7dc7cceb25e0>,
<rdkit.DataStructs.cDataStructs.ExplicitBitVect object at 0x7dc7cceb2650>,
<rdkit.DataStructs.cDataStructs.ExplicitBitVect object at 0x7dc7cceb26c0>,
<rdkit.DataStructs.cDataStructs.ExplicitBitVect object at 0x7dc7cceb2730>,
<rdkit.DataStructs.cDataStructs.ExplicitBitVect object at 0x7dc7cceb27a0>,
<rdkit.DataStructs.cDataStructs.ExplicitBitVect object at 0x7dc7cceb2810>,
<rdkit.DataStructs.cDataStructs.ExplicitBitVect object at 0x7dc7cceb2880>,
<rdkit.DataStructs.cDataStructs.ExplicitBitVect object at 0x7dc7cceb28f0>,
<rdkit.DataStructs.cDataStructs.ExplicitBitVect object at 0x7dc7cceb2960>,
<rdkit.DataStructs.cDataStructs.ExplicitBitVect object at 0x7dc7cceb29d0>,
<rdkit.DataStructs.cDataStructs.ExplicitBitVect object at 0x7dc7cceb2a40>,
<rdkit.DataStructs.cDataStructs.ExplicitBitVect object at 0x7dc7cceb2ab0>,
<rdkit.DataStructs.cDataStructs.ExplicitBitVect object at 0x7dc7cceb2b20>,
<rdkit.DataStructs.cDataStructs.ExplicitBitVect object at 0x7dc7cceb2b90>,
<rdkit.DataStructs.cDataStructs.ExplicitBitVect object at 0x7dc7cceb2c00>,
<rdkit.DataStructs.cDataStructs.ExplicitBitVect object at 0x7dc7cceb2c70>,
<rdkit.DataStructs.cDataStructs.ExplicitBitVect object at 0x7dc7cceb2ce0>,
<rdkit.DataStructs.cDataStructs.ExplicitBitVect object at 0x7dc7cceb2d50>,
<rdkit.DataStructs.cDataStructs.ExplicitBitVect object at 0x7dc7cceb2dc0>,
<rdkit.DataStructs.cDataStructs.ExplicitBitVect object at 0x7dc7cceb2e30>,
<rdkit.DataStructs.cDataStructs.ExplicitBitVect object at 0x7dc7cceb2ea0>,
<rdkit.DataStructs.cDataStructs.ExplicitBitVect object at 0x7dc7cceb2f10>,
<rdkit.DataStructs.cDataStructs.ExplicitBitVect object at 0x7dc7cceb2f80>,
<rdkit.DataStructs.cDataStructs.ExplicitBitVect object at 0x7dc7cceb2ff0>,
<rdkit.DataStructs.cDataStructs.ExplicitBitVect object at 0x7dc7cceb3060>,
<rdkit.DataStructs.cDataStructs.ExplicitBitVect object at 0x7dc7cceb30d0>,
<rdkit.DataStructs.cDataStructs.ExplicitBitVect object at 0x7dc7cceb3140>,
<rdkit.DataStructs.cDataStructs.ExplicitBitVect object at 0x7dc7cceb31b0>,
<rdkit.DataStructs.cDataStructs.ExplicitBitVect object at 0x7dc7cceb3220>,
<rdkit.DataStructs.cDataStructs.ExplicitBitVect object at 0x7dc7cceb3290>,
<rdkit.DataStructs.cDataStructs.ExplicitBitVect object at 0x7dc7cceb3300>,
<rdkit.DataStructs.cDataStructs.ExplicitBitVect object at 0x7dc7cceb3370>,
<rdkit.DataStructs.cDataStructs.ExplicitBitVect object at 0x7dc7cceb33e0>,
<rdkit.DataStructs.cDataStructs.ExplicitBitVect object at 0x7dc7cceb3450>,
<rdkit.DataStructs.cDataStructs.ExplicitBitVect object at 0x7dc7cceb34c0>,
<rdkit.DataStructs.cDataStructs.ExplicitBitVect object at 0x7dc7cceb3530>,
```

```
<rdkit.DataStructs.cDataStructs.ExplicitBitVect object at 0x7dc7cceb35a0>,
<rdkit.DataStructs.cDataStructs.ExplicitBitVect object at 0x7dc7cceb3610>,
<rdkit.DataStructs.cDataStructs.ExplicitBitVect object at 0x7dc7cceb3680>,
<rdkit.DataStructs.cDataStructs.ExplicitBitVect object at 0x7dc7cceb36f0>,
<rdkit.DataStructs.cDataStructs.ExplicitBitVect object at 0x7dc7cceb3760>,
<rdkit.DataStructs.cDataStructs.ExplicitBitVect object at 0x7dc7cceb37d0>,
<rdkit.DataStructs.cDataStructs.ExplicitBitVect object at 0x7dc7cceb3840>,
<rdkit.DataStructs.cDataStructs.ExplicitBitVect object at 0x7dc7cceb38b0>,
<rdkit.DataStructs.cDataStructs.ExplicitBitVect object at 0x7dc7cceb3920>,
<rdkit.DataStructs.cDataStructs.ExplicitBitVect object at 0x7dc7cceb3990>,
<rdkit.DataStructs.cDataStructs.ExplicitBitVect object at 0x7dc7cceb3a00>,
<rdkit.DataStructs.cDataStructs.ExplicitBitVect object at 0x7dc7cceb3a70>,
<rdkit.DataStructs.cDataStructs.ExplicitBitVect object at 0x7dc7cceb3ae0>,
<rdkit.DataStructs.cDataStructs.ExplicitBitVect object at 0x7dc7cceb3b50>,
<rdkit.DataStructs.cDataStructs.ExplicitBitVect object at 0x7dc7cceb3bc0>,
<rdkit.DataStructs.cDataStructs.ExplicitBitVect object at 0x7dc7cceb3c30>,
<rdkit.DataStructs.cDataStructs.ExplicitBitVect object at 0x7dc7cceb3ca0>,
<rdkit.DataStructs.cDataStructs.ExplicitBitVect object at 0x7dc7cceb3d10>,
<rdkit.DataStructs.cDataStructs.ExplicitBitVect object at 0x7dc7cceb3d80>,
<rdkit.DataStructs.cDataStructs.ExplicitBitVect object at 0x7dc7cceb3df0>,
<rdkit.DataStructs.cDataStructs.ExplicitBitVect object at 0x7dc7cceb3e60>,
<rdkit.DataStructs.cDataStructs.ExplicitBitVect object at 0x7dc7cceb3ed0>,
<rdkit.DataStructs.cDataStructs.ExplicitBitVect object at 0x7dc7cceb3f40>,
<rdkit.DataStructs.cDataStructs.ExplicitBitVect object at 0x7dc7c9634040>,
<rdkit.DataStructs.cDataStructs.ExplicitBitVect object at 0x7dc7c96340b0>,
<rdkit.DataStructs.cDataStructs.ExplicitBitVect object at 0x7dc7c9634120>,
<rdkit.DataStructs.cDataStructs.ExplicitBitVect object at 0x7dc7c9634190>,
<rdkit.DataStructs.cDataStructs.ExplicitBitVect object at 0x7dc7c9634200>,
<rdkit.DataStructs.cDataStructs.ExplicitBitVect object at 0x7dc7c9634270>,
<rdkit.DataStructs.cDataStructs.ExplicitBitVect object at 0x7dc7c96342e0>,
<rdkit.DataStructs.cDataStructs.ExplicitBitVect object at 0x7dc7c9634350>,
<rdkit.DataStructs.cDataStructs.ExplicitBitVect object at 0x7dc7c96343c0>,
<rdkit.DataStructs.cDataStructs.ExplicitBitVect object at 0x7dc7c9634430>,
<rdkit.DataStructs.cDataStructs.ExplicitBitVect object at 0x7dc7c96344a0>,
<rdkit.DataStructs.cDataStructs.ExplicitBitVect object at 0x7dc7c9634510>,
<rdkit.DataStructs.cDataStructs.ExplicitBitVect object at 0x7dc7c9634580>,
<rdkit.DataStructs.cDataStructs.ExplicitBitVect object at 0x7dc7c96345f0>,
<rdkit.DataStructs.cDataStructs.ExplicitBitVect object at 0x7dc7c9634660>,
<rdkit.DataStructs.cDataStructs.ExplicitBitVect object at 0x7dc7c96346d0>,
<rdkit.DataStructs.cDataStructs.ExplicitBitVect object at 0x7dc7c9634740>,
<rdkit.DataStructs.cDataStructs.ExplicitBitVect object at 0x7dc7c96347b0>,
<rdkit.DataStructs.cDataStructs.ExplicitBitVect object at 0x7dc7c9634820>,
<rdkit.DataStructs.cDataStructs.ExplicitBitVect object at 0x7dc7c9634890>,
<rdkit.DataStructs.cDataStructs.ExplicitBitVect object at 0x7dc7c9634900>,
<rdkit.DataStructs.cDataStructs.ExplicitBitVect object at 0x7dc7c9634970>,
<rdkit.DataStructs.cDataStructs.ExplicitBitVect object at 0x7dc7c96349e0>,
<rdkit.DataStructs.cDataStructs.ExplicitBitVect object at 0x7dc7c9634a50>,
<rdkit.DataStructs.cDataStructs.ExplicitBitVect object at 0x7dc7c9634ac0>,
```

```
<rdkit.DataStructs.cDataStructs.ExplicitBitVect object at 0x7dc7c9634b30>,
<rdkit.DataStructs.cDataStructs.ExplicitBitVect object at 0x7dc7c9634ba0>,
<rdkit.DataStructs.cDataStructs.ExplicitBitVect object at 0x7dc7c9634c10>,
<rdkit.DataStructs.cDataStructs.ExplicitBitVect object at 0x7dc7c9634c80>,
<rdkit.DataStructs.cDataStructs.ExplicitBitVect object at 0x7dc7c9634cf0>,
<rdkit.DataStructs.cDataStructs.ExplicitBitVect object at 0x7dc7c9634d60>,
<rdkit.DataStructs.cDataStructs.ExplicitBitVect object at 0x7dc7c9634dd0>,
<rdkit.DataStructs.cDataStructs.ExplicitBitVect object at 0x7dc7c9634e40>,
<rdkit.DataStructs.cDataStructs.ExplicitBitVect object at 0x7dc7c9634eb0>,
<rdkit.DataStructs.cDataStructs.ExplicitBitVect object at 0x7dc7c9634f20>,
<rdkit.DataStructs.cDataStructs.ExplicitBitVect object at 0x7dc7c9634f90>,
<rdkit.DataStructs.cDataStructs.ExplicitBitVect object at 0x7dc7c9635000>,
<rdkit.DataStructs.cDataStructs.ExplicitBitVect object at 0x7dc7c9635070>,
<rdkit.DataStructs.cDataStructs.ExplicitBitVect object at 0x7dc7c96350e0>,
<rdkit.DataStructs.cDataStructs.ExplicitBitVect object at 0x7dc7c9635150>,
<rdkit.DataStructs.cDataStructs.ExplicitBitVect object at 0x7dc7c96351c0>,
<rdkit.DataStructs.cDataStructs.ExplicitBitVect object at 0x7dc7c9635230>,
<rdkit.DataStructs.cDataStructs.ExplicitBitVect object at 0x7dc7c96352a0>,
<rdkit.DataStructs.cDataStructs.ExplicitBitVect object at 0x7dc7c9635310>,
<rdkit.DataStructs.cDataStructs.ExplicitBitVect object at 0x7dc7c9635380>,
<rdkit.DataStructs.cDataStructs.ExplicitBitVect object at 0x7dc7c96353f0>,
<rdkit.DataStructs.cDataStructs.ExplicitBitVect object at 0x7dc7c9635460>,
<rdkit.DataStructs.cDataStructs.ExplicitBitVect object at 0x7dc7c96354d0>,
<rdkit.DataStructs.cDataStructs.ExplicitBitVect object at 0x7dc7c9635540>,
<rdkit.DataStructs.cDataStructs.ExplicitBitVect object at 0x7dc7c96355b0>,
<rdkit.DataStructs.cDataStructs.ExplicitBitVect object at 0x7dc7c9635620>,
<rdkit.DataStructs.cDataStructs.ExplicitBitVect object at 0x7dc7c9635690>,
<rdkit.DataStructs.cDataStructs.ExplicitBitVect object at 0x7dc7c9635700>,
<rdkit.DataStructs.cDataStructs.ExplicitBitVect object at 0x7dc7c9635770>,
<rdkit.DataStructs.cDataStructs.ExplicitBitVect object at 0x7dc7c96357e0>,
<rdkit.DataStructs.cDataStructs.ExplicitBitVect object at 0x7dc7c9635850>,
<rdkit.DataStructs.cDataStructs.ExplicitBitVect object at 0x7dc7c96358c0>,
<rdkit.DataStructs.cDataStructs.ExplicitBitVect object at 0x7dc7c9635930>,
<rdkit.DataStructs.cDataStructs.ExplicitBitVect object at 0x7dc7c96359a0>,
<rdkit.DataStructs.cDataStructs.ExplicitBitVect object at 0x7dc7c9635a10>,
<rdkit.DataStructs.cDataStructs.ExplicitBitVect object at 0x7dc7c9635a80>,
<rdkit.DataStructs.cDataStructs.ExplicitBitVect object at 0x7dc7c9635af0>,
<rdkit.DataStructs.cDataStructs.ExplicitBitVect object at 0x7dc7c9635b60>,
<rdkit.DataStructs.cDataStructs.ExplicitBitVect object at 0x7dc7c9635bd0>,
<rdkit.DataStructs.cDataStructs.ExplicitBitVect object at 0x7dc7c9635c40>,
<rdkit.DataStructs.cDataStructs.ExplicitBitVect object at 0x7dc7c9635cb0>,
<rdkit.DataStructs.cDataStructs.ExplicitBitVect object at 0x7dc7c9635d20>,
<rdkit.DataStructs.cDataStructs.ExplicitBitVect object at 0x7dc7c9635d90>,
<rdkit.DataStructs.cDataStructs.ExplicitBitVect object at 0x7dc7c9635e00>,
<rdkit.DataStructs.cDataStructs.ExplicitBitVect object at 0x7dc7c9635e70>,
<rdkit.DataStructs.cDataStructs.ExplicitBitVect object at 0x7dc7c9635ee0>,
<rdkit.DataStructs.cDataStructs.ExplicitBitVect object at 0x7dc7c9635f50>,
<rdkit.DataStructs.cDataStructs.ExplicitBitVect object at 0x7dc7c9635fc0>,
```

```
<rdkit.DataStructs.cDataStructs.ExplicitBitVect object at 0x7dc7c9636030>,
<rdkit.DataStructs.cDataStructs.ExplicitBitVect object at 0x7dc7c96360a0>,
<rdkit.DataStructs.cDataStructs.ExplicitBitVect object at 0x7dc7c9636110>,
<rdkit.DataStructs.cDataStructs.ExplicitBitVect object at 0x7dc7c9636180>,
<rdkit.DataStructs.cDataStructs.ExplicitBitVect object at 0x7dc7c96362d0>,
<rdkit.DataStructs.cDataStructs.ExplicitBitVect object at 0x7dc7c9636340>,
<rdkit.DataStructs.cDataStructs.ExplicitBitVect object at 0x7dc7c96363b0>,
<rdkit.DataStructs.cDataStructs.ExplicitBitVect object at 0x7dc7c9636420>,
<rdkit.DataStructs.cDataStructs.ExplicitBitVect object at 0x7dc7c9636490>,
<rdkit.DataStructs.cDataStructs.ExplicitBitVect object at 0x7dc7c9636500>,
<rdkit.DataStructs.cDataStructs.ExplicitBitVect object at 0x7dc7c9636570>,
<rdkit.DataStructs.cDataStructs.ExplicitBitVect object at 0x7dc7c96365e0>,
<rdkit.DataStructs.cDataStructs.ExplicitBitVect object at 0x7dc7c9636650>,
<rdkit.DataStructs.cDataStructs.ExplicitBitVect object at 0x7dc7c96366c0>,
<rdkit.DataStructs.cDataStructs.ExplicitBitVect object at 0x7dc7c9636730>,
<rdkit.DataStructs.cDataStructs.ExplicitBitVect object at 0x7dc7c96367a0>,
<rdkit.DataStructs.cDataStructs.ExplicitBitVect object at 0x7dc7c9636810>,
<rdkit.DataStructs.cDataStructs.ExplicitBitVect object at 0x7dc7c9636880>,
<rdkit.DataStructs.cDataStructs.ExplicitBitVect object at 0x7dc7c96368f0>,
<rdkit.DataStructs.cDataStructs.ExplicitBitVect object at 0x7dc7c9636960>,
<rdkit.DataStructs.cDataStructs.ExplicitBitVect object at 0x7dc7c96369d0>,
<rdkit.DataStructs.cDataStructs.ExplicitBitVect object at 0x7dc7c9636a40>,
<rdkit.DataStructs.cDataStructs.ExplicitBitVect object at 0x7dc7c9636ab0>,
<rdkit.DataStructs.cDataStructs.ExplicitBitVect object at 0x7dc7c9636b20>,
<rdkit.DataStructs.cDataStructs.ExplicitBitVect object at 0x7dc7c9636b90>,
<rdkit.DataStructs.cDataStructs.ExplicitBitVect object at 0x7dc7c9636c00>,
<rdkit.DataStructs.cDataStructs.ExplicitBitVect object at 0x7dc7c9636c70>,
<rdkit.DataStructs.cDataStructs.ExplicitBitVect object at 0x7dc7c9636ce0>,
<rdkit.DataStructs.cDataStructs.ExplicitBitVect object at 0x7dc7c9636d50>,
<rdkit.DataStructs.cDataStructs.ExplicitBitVect object at 0x7dc7c9636dc0>,
<rdkit.DataStructs.cDataStructs.ExplicitBitVect object at 0x7dc7c9636e30>,
<rdkit.DataStructs.cDataStructs.ExplicitBitVect object at 0x7dc7c9636ea0>,
<rdkit.DataStructs.cDataStructs.ExplicitBitVect object at 0x7dc7c9636f10>,
<rdkit.DataStructs.cDataStructs.ExplicitBitVect object at 0x7dc7c9636f80>,
<rdkit.DataStructs.cDataStructs.ExplicitBitVect object at 0x7dc7c9637060>,
<rdkit.DataStructs.cDataStructs.ExplicitBitVect object at 0x7dc7c96370d0>,
<rdkit.DataStructs.cDataStructs.ExplicitBitVect object at 0x7dc7c9637140>,
<rdkit.DataStructs.cDataStructs.ExplicitBitVect object at 0x7dc7c96371b0>,
<rdkit.DataStructs.cDataStructs.ExplicitBitVect object at 0x7dc7c9637220>,
<rdkit.DataStructs.cDataStructs.ExplicitBitVect object at 0x7dc7c9637290>,
<rdkit.DataStructs.cDataStructs.ExplicitBitVect object at 0x7dc7c9637300>,
<rdkit.DataStructs.cDataStructs.ExplicitBitVect object at 0x7dc7c9637370>,
<rdkit.DataStructs.cDataStructs.ExplicitBitVect object at 0x7dc7c96373e0>,
<rdkit.DataStructs.cDataStructs.ExplicitBitVect object at 0x7dc7c9637450>,
<rdkit.DataStructs.cDataStructs.ExplicitBitVect object at 0x7dc7c96374c0>,
<rdkit.DataStructs.cDataStructs.ExplicitBitVect object at 0x7dc7c9637530>,
<rdkit.DataStructs.cDataStructs.ExplicitBitVect object at 0x7dc7c96375a0>,
<rdkit.DataStructs.cDataStructs.ExplicitBitVect object at 0x7dc7c9637610>,
```

```
<rdkit.DataStructs.cDataStructs.ExplicitBitVect object at 0x7dc7c9637680>,
<rdkit.DataStructs.cDataStructs.ExplicitBitVect object at 0x7dc7c96376f0>,
<rdkit.DataStructs.cDataStructs.ExplicitBitVect object at 0x7dc7c9637760>,
<rdkit.DataStructs.cDataStructs.ExplicitBitVect object at 0x7dc7c96377d0>,
<rdkit.DataStructs.cDataStructs.ExplicitBitVect object at 0x7dc7c9637840>,
<rdkit.DataStructs.cDataStructs.ExplicitBitVect object at 0x7dc7c96378b0>,
<rdkit.DataStructs.cDataStructs.ExplicitBitVect object at 0x7dc7c9637920>,
<rdkit.DataStructs.cDataStructs.ExplicitBitVect object at 0x7dc7c9637990>,
<rdkit.DataStructs.cDataStructs.ExplicitBitVect object at 0x7dc7c9637a00>,
<rdkit.DataStructs.cDataStructs.ExplicitBitVect object at 0x7dc7c9637a70>,
<rdkit.DataStructs.cDataStructs.ExplicitBitVect object at 0x7dc7c9637ae0>,
<rdkit.DataStructs.cDataStructs.ExplicitBitVect object at 0x7dc7c9637b50>,
<rdkit.DataStructs.cDataStructs.ExplicitBitVect object at 0x7dc7c9637bc0>,
<rdkit.DataStructs.cDataStructs.ExplicitBitVect object at 0x7dc7c9637c30>,
<rdkit.DataStructs.cDataStructs.ExplicitBitVect object at 0x7dc7c9637ca0>,
<rdkit.DataStructs.cDataStructs.ExplicitBitVect object at 0x7dc7c9637d10>,
<rdkit.DataStructs.cDataStructs.ExplicitBitVect object at 0x7dc7c9636ff0>,
<rdkit.DataStructs.cDataStructs.ExplicitBitVect object at 0x7dc7c9637df0>,
<rdkit.DataStructs.cDataStructs.ExplicitBitVect object at 0x7dc7c9637e60>,
<rdkit.DataStructs.cDataStructs.ExplicitBitVect object at 0x7dc7c9637ed0>,
<rdkit.DataStructs.cDataStructs.ExplicitBitVect object at 0x7dc7c9637f40>,
<rdkit.DataStructs.cDataStructs.ExplicitBitVect object at 0x7dc7c9637d80>,
<rdkit.DataStructs.cDataStructs.ExplicitBitVect object at 0x7dc7c963c040>,
<rdkit.DataStructs.cDataStructs.ExplicitBitVect object at 0x7dc7c963c0b0>,
<rdkit.DataStructs.cDataStructs.ExplicitBitVect object at 0x7dc7c963c120>,
<rdkit.DataStructs.cDataStructs.ExplicitBitVect object at 0x7dc7c963c190>,
<rdkit.DataStructs.cDataStructs.ExplicitBitVect object at 0x7dc7c963c200>,
<rdkit.DataStructs.cDataStructs.ExplicitBitVect object at 0x7dc7c963c2e0>,
<rdkit.DataStructs.cDataStructs.ExplicitBitVect object at 0x7dc7c963c350>,
<rdkit.DataStructs.cDataStructs.ExplicitBitVect object at 0x7dc7c963c3c0>,
<rdkit.DataStructs.cDataStructs.ExplicitBitVect object at 0x7dc7c963c430>,
<rdkit.DataStructs.cDataStructs.ExplicitBitVect object at 0x7dc7c963c4a0>,
<rdkit.DataStructs.cDataStructs.ExplicitBitVect object at 0x7dc7c963c510>,
<rdkit.DataStructs.cDataStructs.ExplicitBitVect object at 0x7dc7c963c580>,
<rdkit.DataStructs.cDataStructs.ExplicitBitVect object at 0x7dc7c963c5f0>,
<rdkit.DataStructs.cDataStructs.ExplicitBitVect object at 0x7dc7c963c660>,
<rdkit.DataStructs.cDataStructs.ExplicitBitVect object at 0x7dc7c963c6d0>,
<rdkit.DataStructs.cDataStructs.ExplicitBitVect object at 0x7dc7c963c740>,
<rdkit.DataStructs.cDataStructs.ExplicitBitVect object at 0x7dc7c963c7b0>,
<rdkit.DataStructs.cDataStructs.ExplicitBitVect object at 0x7dc7c963c820>,
<rdkit.DataStructs.cDataStructs.ExplicitBitVect object at 0x7dc7c963c890>,
<rdkit.DataStructs.cDataStructs.ExplicitBitVect object at 0x7dc7c963c900>,
<rdkit.DataStructs.cDataStructs.ExplicitBitVect object at 0x7dc7c963c970>,
<rdkit.DataStructs.cDataStructs.ExplicitBitVect object at 0x7dc7c963c9e0>,
<rdkit.DataStructs.cDataStructs.ExplicitBitVect object at 0x7dc7c963ca50>,
<rdkit.DataStructs.cDataStructs.ExplicitBitVect object at 0x7dc7c963cac0>,
<rdkit.DataStructs.cDataStructs.ExplicitBitVect object at 0x7dc7c963cb30>,
<rdkit.DataStructs.cDataStructs.ExplicitBitVect object at 0x7dc7c963cba0>,
```

```
<rdkit.DataStructs.cDataStructs.ExplicitBitVect object at 0x7dc7c963c270>,
<rdkit.DataStructs.cDataStructs.ExplicitBitVect object at 0x7dc7c963cc10>,
<rdkit.DataStructs.cDataStructs.ExplicitBitVect object at 0x7dc7c963cc80>,
<rdkit.DataStructs.cDataStructs.ExplicitBitVect object at 0x7dc7c963ccf0>,
<rdkit.DataStructs.cDataStructs.ExplicitBitVect object at 0x7dc7c963cd60>,
<rdkit.DataStructs.cDataStructs.ExplicitBitVect object at 0x7dc7c963cdd0>,
<rdkit.DataStructs.cDataStructs.ExplicitBitVect object at 0x7dc7c963ce40>,
<rdkit.DataStructs.cDataStructs.ExplicitBitVect object at 0x7dc7c963ceb0>,
<rdkit.DataStructs.cDataStructs.ExplicitBitVect object at 0x7dc7c963cf20>,
<rdkit.DataStructs.cDataStructs.ExplicitBitVect object at 0x7dc7c963cf90>,
<rdkit.DataStructs.cDataStructs.ExplicitBitVect object at 0x7dc7c963d000>,
<rdkit.DataStructs.cDataStructs.ExplicitBitVect object at 0x7dc7c963d070>,
<rdkit.DataStructs.cDataStructs.ExplicitBitVect object at 0x7dc7c963d0e0>,
<rdkit.DataStructs.cDataStructs.ExplicitBitVect object at 0x7dc7c963d150>,
<rdkit.DataStructs.cDataStructs.ExplicitBitVect object at 0x7dc7c963d1c0>,
<rdkit.DataStructs.cDataStructs.ExplicitBitVect object at 0x7dc7c963d230>,
<rdkit.DataStructs.cDataStructs.ExplicitBitVect object at 0x7dc7c963d2a0>,
<rdkit.DataStructs.cDataStructs.ExplicitBitVect object at 0x7dc7c963d310>,
<rdkit.DataStructs.cDataStructs.ExplicitBitVect object at 0x7dc7c963d380>,
<rdkit.DataStructs.cDataStructs.ExplicitBitVect object at 0x7dc7c963d3f0>,
<rdkit.DataStructs.cDataStructs.ExplicitBitVect object at 0x7dc7c963d460>,
<rdkit.DataStructs.cDataStructs.ExplicitBitVect object at 0x7dc7c963d4d0>,
<rdkit.DataStructs.cDataStructs.ExplicitBitVect object at 0x7dc7c963d540>,
<rdkit.DataStructs.cDataStructs.ExplicitBitVect object at 0x7dc7c963d5b0>,
<rdkit.DataStructs.cDataStructs.ExplicitBitVect object at 0x7dc7c963d620>,
<rdkit.DataStructs.cDataStructs.ExplicitBitVect object at 0x7dc7c963d690>,
<rdkit.DataStructs.cDataStructs.ExplicitBitVect object at 0x7dc7c963d700>,
<rdkit.DataStructs.cDataStructs.ExplicitBitVect object at 0x7dc7c963d770>,
<rdkit.DataStructs.cDataStructs.ExplicitBitVect object at 0x7dc7c963d7e0>,
<rdkit.DataStructs.cDataStructs.ExplicitBitVect object at 0x7dc7c963d850>,
<rdkit.DataStructs.cDataStructs.ExplicitBitVect object at 0x7dc7c963d8c0>,
<rdkit.DataStructs.cDataStructs.ExplicitBitVect object at 0x7dc7c963d930>,
<rdkit.DataStructs.cDataStructs.ExplicitBitVect object at 0x7dc7c963d9a0>,
<rdkit.DataStructs.cDataStructs.ExplicitBitVect object at 0x7dc7c963da10>,
<rdkit.DataStructs.cDataStructs.ExplicitBitVect object at 0x7dc7c963da80>,
<rdkit.DataStructs.cDataStructs.ExplicitBitVect object at 0x7dc7c963daf0>,
<rdkit.DataStructs.cDataStructs.ExplicitBitVect object at 0x7dc7c963db60>,
<rdkit.DataStructs.cDataStructs.ExplicitBitVect object at 0x7dc7c963dbd0>,
<rdkit.DataStructs.cDataStructs.ExplicitBitVect object at 0x7dc7c963dc40>,
<rdkit.DataStructs.cDataStructs.ExplicitBitVect object at 0x7dc7c963dcb0>,
<rdkit.DataStructs.cDataStructs.ExplicitBitVect object at 0x7dc7c963dd20>,
<rdkit.DataStructs.cDataStructs.ExplicitBitVect object at 0x7dc7c963dd90>,
<rdkit.DataStructs.cDataStructs.ExplicitBitVect object at 0x7dc7c963de00>,
<rdkit.DataStructs.cDataStructs.ExplicitBitVect object at 0x7dc7c963de70>,
<rdkit.DataStructs.cDataStructs.ExplicitBitVect object at 0x7dc7c963dee0>,
<rdkit.DataStructs.cDataStructs.ExplicitBitVect object at 0x7dc7c963df50>,
<rdkit.DataStructs.cDataStructs.ExplicitBitVect object at 0x7dc7c963dfc0>,
<rdkit.DataStructs.cDataStructs.ExplicitBitVect object at 0x7dc7c963e030>,
```

```
<rdkit.DataStructs.cDataStructs.ExplicitBitVect object at 0x7dc7c963e0a0>,
<rdkit.DataStructs.cDataStructs.ExplicitBitVect object at 0x7dc7c963e110>,
<rdkit.DataStructs.cDataStructs.ExplicitBitVect object at 0x7dc7c963e180>,
<rdkit.DataStructs.cDataStructs.ExplicitBitVect object at 0x7dc7c963e1f0>,
<rdkit.DataStructs.cDataStructs.ExplicitBitVect object at 0x7dc7c963e260>,
<rdkit.DataStructs.cDataStructs.ExplicitBitVect object at 0x7dc7c963e340>,
<rdkit.DataStructs.cDataStructs.ExplicitBitVect object at 0x7dc7c963e3b0>,
<rdkit.DataStructs.cDataStructs.ExplicitBitVect object at 0x7dc7c963e420>,
<rdkit.DataStructs.cDataStructs.ExplicitBitVect object at 0x7dc7c963e490>,
<rdkit.DataStructs.cDataStructs.ExplicitBitVect object at 0x7dc7c963e500>,
<rdkit.DataStructs.cDataStructs.ExplicitBitVect object at 0x7dc7c963e570>,
<rdkit.DataStructs.cDataStructs.ExplicitBitVect object at 0x7dc7c963e5e0>,
<rdkit.DataStructs.cDataStructs.ExplicitBitVect object at 0x7dc7c963e650>,
<rdkit.DataStructs.cDataStructs.ExplicitBitVect object at 0x7dc7c963e6c0>,
<rdkit.DataStructs.cDataStructs.ExplicitBitVect object at 0x7dc7c963e730>,
<rdkit.DataStructs.cDataStructs.ExplicitBitVect object at 0x7dc7c963e7a0>,
<rdkit.DataStructs.cDataStructs.ExplicitBitVect object at 0x7dc7c963e810>,
<rdkit.DataStructs.cDataStructs.ExplicitBitVect object at 0x7dc7c963e880>,
<rdkit.DataStructs.cDataStructs.ExplicitBitVect object at 0x7dc7c963e8f0>,
<rdkit.DataStructs.cDataStructs.ExplicitBitVect object at 0x7dc7c963e960>,
<rdkit.DataStructs.cDataStructs.ExplicitBitVect object at 0x7dc7c963e9d0>,
<rdkit.DataStructs.cDataStructs.ExplicitBitVect object at 0x7dc7c963ea40>,
<rdkit.DataStructs.cDataStructs.ExplicitBitVect object at 0x7dc7c963eab0>,
<rdkit.DataStructs.cDataStructs.ExplicitBitVect object at 0x7dc7c963eb20>,
<rdkit.DataStructs.cDataStructs.ExplicitBitVect object at 0x7dc7c963eb90>,
<rdkit.DataStructs.cDataStructs.ExplicitBitVect object at 0x7dc7c963ec00>,
<rdkit.DataStructs.cDataStructs.ExplicitBitVect object at 0x7dc7c963ec70>,
<rdkit.DataStructs.cDataStructs.ExplicitBitVect object at 0x7dc7c963ece0>,
<rdkit.DataStructs.cDataStructs.ExplicitBitVect object at 0x7dc7c963ed50>,
<rdkit.DataStructs.cDataStructs.ExplicitBitVect object at 0x7dc7c963edc0>,
<rdkit.DataStructs.cDataStructs.ExplicitBitVect object at 0x7dc7c963ee30>,
<rdkit.DataStructs.cDataStructs.ExplicitBitVect object at 0x7dc7c963eea0>,
<rdkit.DataStructs.cDataStructs.ExplicitBitVect object at 0x7dc7c963e2d0>,
<rdkit.DataStructs.cDataStructs.ExplicitBitVect object at 0x7dc7c963ef80>,
<rdkit.DataStructs.cDataStructs.ExplicitBitVect object at 0x7dc7c963eff0>,
<rdkit.DataStructs.cDataStructs.ExplicitBitVect object at 0x7dc7c963f060>,
<rdkit.DataStructs.cDataStructs.ExplicitBitVect object at 0x7dc7c963f0d0>,
<rdkit.DataStructs.cDataStructs.ExplicitBitVect object at 0x7dc7c963f140>,
<rdkit.DataStructs.cDataStructs.ExplicitBitVect object at 0x7dc7c963f1b0>,
<rdkit.DataStructs.cDataStructs.ExplicitBitVect object at 0x7dc7c963f220>,
<rdkit.DataStructs.cDataStructs.ExplicitBitVect object at 0x7dc7c963f290>,
<rdkit.DataStructs.cDataStructs.ExplicitBitVect object at 0x7dc7c963f300>,
<rdkit.DataStructs.cDataStructs.ExplicitBitVect object at 0x7dc7c963f370>,
<rdkit.DataStructs.cDataStructs.ExplicitBitVect object at 0x7dc7c963f3e0>,
<rdkit.DataStructs.cDataStructs.ExplicitBitVect object at 0x7dc7c963f450>,
<rdkit.DataStructs.cDataStructs.ExplicitBitVect object at 0x7dc7c963f4c0>,
<rdkit.DataStructs.cDataStructs.ExplicitBitVect object at 0x7dc7c963f530>,
<rdkit.DataStructs.cDataStructs.ExplicitBitVect object at 0x7dc7c963f5a0>,
```

```
<rdkit.DataStructs.cDataStructs.ExplicitBitVect object at 0x7dc7c963f610>,
<rdkit.DataStructs.cDataStructs.ExplicitBitVect object at 0x7dc7c963f680>,
<rdkit.DataStructs.cDataStructs.ExplicitBitVect object at 0x7dc7c963f6f0>,
<rdkit.DataStructs.cDataStructs.ExplicitBitVect object at 0x7dc7c963f760>,
<rdkit.DataStructs.cDataStructs.ExplicitBitVect object at 0x7dc7c963f7d0>,
<rdkit.DataStructs.cDataStructs.ExplicitBitVect object at 0x7dc7c963f840>,
<rdkit.DataStructs.cDataStructs.ExplicitBitVect object at 0x7dc7c963f8b0>,
<rdkit.DataStructs.cDataStructs.ExplicitBitVect object at 0x7dc7c963f920>,
<rdkit.DataStructs.cDataStructs.ExplicitBitVect object at 0x7dc7c963f990>,
<rdkit.DataStructs.cDataStructs.ExplicitBitVect object at 0x7dc7c963fa00>,
<rdkit.DataStructs.cDataStructs.ExplicitBitVect object at 0x7dc7c963fa70>,
<rdkit.DataStructs.cDataStructs.ExplicitBitVect object at 0x7dc7c963fae0>,
<rdkit.DataStructs.cDataStructs.ExplicitBitVect object at 0x7dc7c963fb50>,
<rdkit.DataStructs.cDataStructs.ExplicitBitVect object at 0x7dc7c963fbc0>,
<rdkit.DataStructs.cDataStructs.ExplicitBitVect object at 0x7dc7c963fc30>,
<rdkit.DataStructs.cDataStructs.ExplicitBitVect object at 0x7dc7c963fca0>,
<rdkit.DataStructs.cDataStructs.ExplicitBitVect object at 0x7dc7c963fd10>,
<rdkit.DataStructs.cDataStructs.ExplicitBitVect object at 0x7dc7c963fd80>,
<rdkit.DataStructs.cDataStructs.ExplicitBitVect object at 0x7dc7c963fdf0>,
<rdkit.DataStructs.cDataStructs.ExplicitBitVect object at 0x7dc7c963fe60>,
<rdkit.DataStructs.cDataStructs.ExplicitBitVect object at 0x7dc7c963fed0>,
<rdkit.DataStructs.cDataStructs.ExplicitBitVect object at 0x7dc7c963ff40>,
<rdkit.DataStructs.cDataStructs.ExplicitBitVect object at 0x7dc7c9644040>,
<rdkit.DataStructs.cDataStructs.ExplicitBitVect object at 0x7dc7c96440b0>,
<rdkit.DataStructs.cDataStructs.ExplicitBitVect object at 0x7dc7c9644120>,
<rdkit.DataStructs.cDataStructs.ExplicitBitVect object at 0x7dc7c9644190>,
<rdkit.DataStructs.cDataStructs.ExplicitBitVect object at 0x7dc7c9644200>,
<rdkit.DataStructs.cDataStructs.ExplicitBitVect object at 0x7dc7c9644270>,
<rdkit.DataStructs.cDataStructs.ExplicitBitVect object at 0x7dc7c96442e0>,
<rdkit.DataStructs.cDataStructs.ExplicitBitVect object at 0x7dc7c9644350>,
<rdkit.DataStructs.cDataStructs.ExplicitBitVect object at 0x7dc7c96443c0>,
<rdkit.DataStructs.cDataStructs.ExplicitBitVect object at 0x7dc7c9644430>,
<rdkit.DataStructs.cDataStructs.ExplicitBitVect object at 0x7dc7c96444a0>,
<rdkit.DataStructs.cDataStructs.ExplicitBitVect object at 0x7dc7c9644510>,
<rdkit.DataStructs.cDataStructs.ExplicitBitVect object at 0x7dc7c9644580>,
<rdkit.DataStructs.cDataStructs.ExplicitBitVect object at 0x7dc7c96445f0>,
<rdkit.DataStructs.cDataStructs.ExplicitBitVect object at 0x7dc7c9644660>,
<rdkit.DataStructs.cDataStructs.ExplicitBitVect object at 0x7dc7c96446d0>,
<rdkit.DataStructs.cDataStructs.ExplicitBitVect object at 0x7dc7c9644740>,
<rdkit.DataStructs.cDataStructs.ExplicitBitVect object at 0x7dc7c96447b0>,
<rdkit.DataStructs.cDataStructs.ExplicitBitVect object at 0x7dc7c9644820>,
<rdkit.DataStructs.cDataStructs.ExplicitBitVect object at 0x7dc7c9644890>,
<rdkit.DataStructs.cDataStructs.ExplicitBitVect object at 0x7dc7c9644900>,
<rdkit.DataStructs.cDataStructs.ExplicitBitVect object at 0x7dc7c9644970>,
<rdkit.DataStructs.cDataStructs.ExplicitBitVect object at 0x7dc7c96449e0>,
<rdkit.DataStructs.cDataStructs.ExplicitBitVect object at 0x7dc7c9644a50>,
<rdkit.DataStructs.cDataStructs.ExplicitBitVect object at 0x7dc7c9644ac0>,
<rdkit.DataStructs.cDataStructs.ExplicitBitVect object at 0x7dc7c9644b30>,
```

```
<rdkit.DataStructs.cDataStructs.ExplicitBitVect object at 0x7dc7c9644ba0>,
<rdkit.DataStructs.cDataStructs.ExplicitBitVect object at 0x7dc7c9644c10>,
<rdkit.DataStructs.cDataStructs.ExplicitBitVect object at 0x7dc7c9644c80>,
<rdkit.DataStructs.cDataStructs.ExplicitBitVect object at 0x7dc7c9644cf0>,
<rdkit.DataStructs.cDataStructs.ExplicitBitVect object at 0x7dc7c9644d60>,
<rdkit.DataStructs.cDataStructs.ExplicitBitVect object at 0x7dc7c9644dd0>,
<rdkit.DataStructs.cDataStructs.ExplicitBitVect object at 0x7dc7c9644e40>,
<rdkit.DataStructs.cDataStructs.ExplicitBitVect object at 0x7dc7c9644eb0>,
<rdkit.DataStructs.cDataStructs.ExplicitBitVect object at 0x7dc7c9644f20>,
<rdkit.DataStructs.cDataStructs.ExplicitBitVect object at 0x7dc7c9644f90>,
<rdkit.DataStructs.cDataStructs.ExplicitBitVect object at 0x7dc7c9645000>,
<rdkit.DataStructs.cDataStructs.ExplicitBitVect object at 0x7dc7c9645070>,
<rdkit.DataStructs.cDataStructs.ExplicitBitVect object at 0x7dc7c96450e0>,
<rdkit.DataStructs.cDataStructs.ExplicitBitVect object at 0x7dc7c96451c0>,
<rdkit.DataStructs.cDataStructs.ExplicitBitVect object at 0x7dc7c9645230>,
<rdkit.DataStructs.cDataStructs.ExplicitBitVect object at 0x7dc7c96452a0>,
<rdkit.DataStructs.cDataStructs.ExplicitBitVect object at 0x7dc7c9645310>,
<rdkit.DataStructs.cDataStructs.ExplicitBitVect object at 0x7dc7c9645380>,
<rdkit.DataStructs.cDataStructs.ExplicitBitVect object at 0x7dc7c96453f0>,
<rdkit.DataStructs.cDataStructs.ExplicitBitVect object at 0x7dc7c9645460>,
<rdkit.DataStructs.cDataStructs.ExplicitBitVect object at 0x7dc7c96454d0>,
<rdkit.DataStructs.cDataStructs.ExplicitBitVect object at 0x7dc7c9645540>,
<rdkit.DataStructs.cDataStructs.ExplicitBitVect object at 0x7dc7c96455b0>,
<rdkit.DataStructs.cDataStructs.ExplicitBitVect object at 0x7dc7c9645620>,
<rdkit.DataStructs.cDataStructs.ExplicitBitVect object at 0x7dc7c9645690>,
<rdkit.DataStructs.cDataStructs.ExplicitBitVect object at 0x7dc7c9645700>,
<rdkit.DataStructs.cDataStructs.ExplicitBitVect object at 0x7dc7c9645770>,
<rdkit.DataStructs.cDataStructs.ExplicitBitVect object at 0x7dc7c96457e0>,
<rdkit.DataStructs.cDataStructs.ExplicitBitVect object at 0x7dc7c9645850>,
<rdkit.DataStructs.cDataStructs.ExplicitBitVect object at 0x7dc7c96458c0>,
<rdkit.DataStructs.cDataStructs.ExplicitBitVect object at 0x7dc7c9645930>,
<rdkit.DataStructs.cDataStructs.ExplicitBitVect object at 0x7dc7c96459a0>,
<rdkit.DataStructs.cDataStructs.ExplicitBitVect object at 0x7dc7c9645a10>,
<rdkit.DataStructs.cDataStructs.ExplicitBitVect object at 0x7dc7c9645a80>,
<rdkit.DataStructs.cDataStructs.ExplicitBitVect object at 0x7dc7c9645af0>,
<rdkit.DataStructs.cDataStructs.ExplicitBitVect object at 0x7dc7c9645b60>,
<rdkit.DataStructs.cDataStructs.ExplicitBitVect object at 0x7dc7c9645c40>,
<rdkit.DataStructs.cDataStructs.ExplicitBitVect object at 0x7dc7c9645cb0>,
<rdkit.DataStructs.cDataStructs.ExplicitBitVect object at 0x7dc7c9645d20>,
<rdkit.DataStructs.cDataStructs.ExplicitBitVect object at 0x7dc7c9645d90>,
<rdkit.DataStructs.cDataStructs.ExplicitBitVect object at 0x7dc7c9645e00>,
<rdkit.DataStructs.cDataStructs.ExplicitBitVect object at 0x7dc7c9645e70>,
<rdkit.DataStructs.cDataStructs.ExplicitBitVect object at 0x7dc7c9645ee0>,
<rdkit.DataStructs.cDataStructs.ExplicitBitVect object at 0x7dc7c9645f50>,
<rdkit.DataStructs.cDataStructs.ExplicitBitVect object at 0x7dc7c9645fc0>,
<rdkit.DataStructs.cDataStructs.ExplicitBitVect object at 0x7dc7c9646030>,
<rdkit.DataStructs.cDataStructs.ExplicitBitVect object at 0x7dc7c96460a0>,
<rdkit.DataStructs.cDataStructs.ExplicitBitVect object at 0x7dc7c9646110>,
```

```
<rdkit.DataStructs.cDataStructs.ExplicitBitVect object at 0x7dc7c9646180>,
<rdkit.DataStructs.cDataStructs.ExplicitBitVect object at 0x7dc7c96461f0>,
<rdkit.DataStructs.cDataStructs.ExplicitBitVect object at 0x7dc7c9646260>,
<rdkit.DataStructs.cDataStructs.ExplicitBitVect object at 0x7dc7c96462d0>,
<rdkit.DataStructs.cDataStructs.ExplicitBitVect object at 0x7dc7c9646340>,
<rdkit.DataStructs.cDataStructs.ExplicitBitVect object at 0x7dc7c96463b0>,
<rdkit.DataStructs.cDataStructs.ExplicitBitVect object at 0x7dc7c9645bd0>,
<rdkit.DataStructs.cDataStructs.ExplicitBitVect object at 0x7dc7c9646490>,
<rdkit.DataStructs.cDataStructs.ExplicitBitVect object at 0x7dc7c9646500>,
<rdkit.DataStructs.cDataStructs.ExplicitBitVect object at 0x7dc7c9646570>,
<rdkit.DataStructs.cDataStructs.ExplicitBitVect object at 0x7dc7c96465e0>,
<rdkit.DataStructs.cDataStructs.ExplicitBitVect object at 0x7dc7c9646650>,
<rdkit.DataStructs.cDataStructs.ExplicitBitVect object at 0x7dc7c96466c0>,
<rdkit.DataStructs.cDataStructs.ExplicitBitVect object at 0x7dc7c9646730>,
<rdkit.DataStructs.cDataStructs.ExplicitBitVect object at 0x7dc7c96467a0>,
<rdkit.DataStructs.cDataStructs.ExplicitBitVect object at 0x7dc7c9646810>,
<rdkit.DataStructs.cDataStructs.ExplicitBitVect object at 0x7dc7c9646880>,
<rdkit.DataStructs.cDataStructs.ExplicitBitVect object at 0x7dc7c96468f0>,
<rdkit.DataStructs.cDataStructs.ExplicitBitVect object at 0x7dc7c9646960>,
<rdkit.DataStructs.cDataStructs.ExplicitBitVect object at 0x7dc7c96469d0>,
<rdkit.DataStructs.cDataStructs.ExplicitBitVect object at 0x7dc7c9646a40>,
<rdkit.DataStructs.cDataStructs.ExplicitBitVect object at 0x7dc7c9646ab0>,
<rdkit.DataStructs.cDataStructs.ExplicitBitVect object at 0x7dc7c9646b20>,
<rdkit.DataStructs.cDataStructs.ExplicitBitVect object at 0x7dc7c9646b90>,
<rdkit.DataStructs.cDataStructs.ExplicitBitVect object at 0x7dc7c9646c00>,
<rdkit.DataStructs.cDataStructs.ExplicitBitVect object at 0x7dc7c9646c70>,
<rdkit.DataStructs.cDataStructs.ExplicitBitVect object at 0x7dc7c9646ce0>,
<rdkit.DataStructs.cDataStructs.ExplicitBitVect object at 0x7dc7c9646d50>,
<rdkit.DataStructs.cDataStructs.ExplicitBitVect object at 0x7dc7c9646dc0>,
<rdkit.DataStructs.cDataStructs.ExplicitBitVect object at 0x7dc7c9646e30>,
<rdkit.DataStructs.cDataStructs.ExplicitBitVect object at 0x7dc7c9646ea0>,
<rdkit.DataStructs.cDataStructs.ExplicitBitVect object at 0x7dc7c9646f10>,
<rdkit.DataStructs.cDataStructs.ExplicitBitVect object at 0x7dc7c9646f80>,
<rdkit.DataStructs.cDataStructs.ExplicitBitVect object at 0x7dc7c9646ff0>,
<rdkit.DataStructs.cDataStructs.ExplicitBitVect object at 0x7dc7c9647060>,
<rdkit.DataStructs.cDataStructs.ExplicitBitVect object at 0x7dc7c96470d0>,
<rdkit.DataStructs.cDataStructs.ExplicitBitVect object at 0x7dc7c9647140>,
<rdkit.DataStructs.cDataStructs.ExplicitBitVect object at 0x7dc7c96471b0>,
<rdkit.DataStructs.cDataStructs.ExplicitBitVect object at 0x7dc7c9647220>,
<rdkit.DataStructs.cDataStructs.ExplicitBitVect object at 0x7dc7c9647290>,
<rdkit.DataStructs.cDataStructs.ExplicitBitVect object at 0x7dc7c9647300>,
<rdkit.DataStructs.cDataStructs.ExplicitBitVect object at 0x7dc7c9647370>,
<rdkit.DataStructs.cDataStructs.ExplicitBitVect object at 0x7dc7c96473e0>,
<rdkit.DataStructs.cDataStructs.ExplicitBitVect object at 0x7dc7c9647450>,
<rdkit.DataStructs.cDataStructs.ExplicitBitVect object at 0x7dc7c96474c0>,
<rdkit.DataStructs.cDataStructs.ExplicitBitVect object at 0x7dc7c9647530>,
<rdkit.DataStructs.cDataStructs.ExplicitBitVect object at 0x7dc7c96475a0>,
<rdkit.DataStructs.cDataStructs.ExplicitBitVect object at 0x7dc7c9647610>,
```

```
<rdkit.DataStructs.cDataStructs.ExplicitBitVect object at 0x7dc7c9647680>,
<rdkit.DataStructs.cDataStructs.ExplicitBitVect object at 0x7dc7c96476f0>,
<rdkit.DataStructs.cDataStructs.ExplicitBitVect object at 0x7dc7c9647760>,
<rdkit.DataStructs.cDataStructs.ExplicitBitVect object at 0x7dc7c96477d0>,
<rdkit.DataStructs.cDataStructs.ExplicitBitVect object at 0x7dc7c9647840>,
<rdkit.DataStructs.cDataStructs.ExplicitBitVect object at 0x7dc7c96478b0>,
<rdkit.DataStructs.cDataStructs.ExplicitBitVect object at 0x7dc7c9647920>,
<rdkit.DataStructs.cDataStructs.ExplicitBitVect object at 0x7dc7c9647990>,
<rdkit.DataStructs.cDataStructs.ExplicitBitVect object at 0x7dc7c9647a00>,
<rdkit.DataStructs.cDataStructs.ExplicitBitVect object at 0x7dc7c9647a70>,
<rdkit.DataStructs.cDataStructs.ExplicitBitVect object at 0x7dc7c9647ae0>,
<rdkit.DataStructs.cDataStructs.ExplicitBitVect object at 0x7dc7c9647b50>,
<rdkit.DataStructs.cDataStructs.ExplicitBitVect object at 0x7dc7c9647bc0>,
<rdkit.DataStructs.cDataStructs.ExplicitBitVect object at 0x7dc7c9647c30>,
<rdkit.DataStructs.cDataStructs.ExplicitBitVect object at 0x7dc7c9647ca0>,
<rdkit.DataStructs.cDataStructs.ExplicitBitVect object at 0x7dc7c9647d10>,
<rdkit.DataStructs.cDataStructs.ExplicitBitVect object at 0x7dc7c9647d80>,
<rdkit.DataStructs.cDataStructs.ExplicitBitVect object at 0x7dc7c9647df0>,
<rdkit.DataStructs.cDataStructs.ExplicitBitVect object at 0x7dc7c9647e60>,
<rdkit.DataStructs.cDataStructs.ExplicitBitVect object at 0x7dc7c9647ed0>,
<rdkit.DataStructs.cDataStructs.ExplicitBitVect object at 0x7dc7c9647f40>,
<rdkit.DataStructs.cDataStructs.ExplicitBitVect object at 0x7dc7c9648040>,
<rdkit.DataStructs.cDataStructs.ExplicitBitVect object at 0x7dc7c96480b0>,
<rdkit.DataStructs.cDataStructs.ExplicitBitVect object at 0x7dc7c9648120>,
<rdkit.DataStructs.cDataStructs.ExplicitBitVect object at 0x7dc7c9648190>,
<rdkit.DataStructs.cDataStructs.ExplicitBitVect object at 0x7dc7c9648200>,
<rdkit.DataStructs.cDataStructs.ExplicitBitVect object at 0x7dc7c96482e0>,
<rdkit.DataStructs.cDataStructs.ExplicitBitVect object at 0x7dc7c9648350>,
<rdkit.DataStructs.cDataStructs.ExplicitBitVect object at 0x7dc7c96483c0>,
<rdkit.DataStructs.cDataStructs.ExplicitBitVect object at 0x7dc7c9648430>,
<rdkit.DataStructs.cDataStructs.ExplicitBitVect object at 0x7dc7c96484a0>,
<rdkit.DataStructs.cDataStructs.ExplicitBitVect object at 0x7dc7c9648510>,
<rdkit.DataStructs.cDataStructs.ExplicitBitVect object at 0x7dc7c9648580>,
<rdkit.DataStructs.cDataStructs.ExplicitBitVect object at 0x7dc7c96485f0>,
<rdkit.DataStructs.cDataStructs.ExplicitBitVect object at 0x7dc7c9648660>,
<rdkit.DataStructs.cDataStructs.ExplicitBitVect object at 0x7dc7c96486d0>,
<rdkit.DataStructs.cDataStructs.ExplicitBitVect object at 0x7dc7c9648740>,
<rdkit.DataStructs.cDataStructs.ExplicitBitVect object at 0x7dc7c96487b0>,
<rdkit.DataStructs.cDataStructs.ExplicitBitVect object at 0x7dc7c9648820>,
<rdkit.DataStructs.cDataStructs.ExplicitBitVect object at 0x7dc7c9648890>,
<rdkit.DataStructs.cDataStructs.ExplicitBitVect object at 0x7dc7c9648900>,
<rdkit.DataStructs.cDataStructs.ExplicitBitVect object at 0x7dc7c9648970>,
<rdkit.DataStructs.cDataStructs.ExplicitBitVect object at 0x7dc7c96489e0>,
<rdkit.DataStructs.cDataStructs.ExplicitBitVect object at 0x7dc7c9648a50>,
<rdkit.DataStructs.cDataStructs.ExplicitBitVect object at 0x7dc7c9648ac0>,
<rdkit.DataStructs.cDataStructs.ExplicitBitVect object at 0x7dc7c9648b30>,
<rdkit.DataStructs.cDataStructs.ExplicitBitVect object at 0x7dc7c9648ba0>,
<rdkit.DataStructs.cDataStructs.ExplicitBitVect object at 0x7dc7c9648c10>,
```

```
<rdkit.DataStructs.cDataStructs.ExplicitBitVect object at 0x7dc7c9648c80>,
<rdkit.DataStructs.cDataStructs.ExplicitBitVect object at 0x7dc7c9648d60>,
<rdkit.DataStructs.cDataStructs.ExplicitBitVect object at 0x7dc7c9648dd0>,
<rdkit.DataStructs.cDataStructs.ExplicitBitVect object at 0x7dc7c9648e40>,
<rdkit.DataStructs.cDataStructs.ExplicitBitVect object at 0x7dc7c9648eb0>,
<rdkit.DataStructs.cDataStructs.ExplicitBitVect object at 0x7dc7c9648f20>,
<rdkit.DataStructs.cDataStructs.ExplicitBitVect object at 0x7dc7c9648f90>,
<rdkit.DataStructs.cDataStructs.ExplicitBitVect object at 0x7dc7c9649000>,
<rdkit.DataStructs.cDataStructs.ExplicitBitVect object at 0x7dc7c9649070>,
<rdkit.DataStructs.cDataStructs.ExplicitBitVect object at 0x7dc7c96490e0>,
<rdkit.DataStructs.cDataStructs.ExplicitBitVect object at 0x7dc7c9649150>,
<rdkit.DataStructs.cDataStructs.ExplicitBitVect object at 0x7dc7c96491c0>,
<rdkit.DataStructs.cDataStructs.ExplicitBitVect object at 0x7dc7c9649230>,
<rdkit.DataStructs.cDataStructs.ExplicitBitVect object at 0x7dc7c96492a0>,
<rdkit.DataStructs.cDataStructs.ExplicitBitVect object at 0x7dc7c9649310>,
<rdkit.DataStructs.cDataStructs.ExplicitBitVect object at 0x7dc7c9649380>,
<rdkit.DataStructs.cDataStructs.ExplicitBitVect object at 0x7dc7c96493f0>,
<rdkit.DataStructs.cDataStructs.ExplicitBitVect object at 0x7dc7c9649460>,
<rdkit.DataStructs.cDataStructs.ExplicitBitVect object at 0x7dc7c96494d0>,
<rdkit.DataStructs.cDataStructs.ExplicitBitVect object at 0x7dc7c9649540>,
<rdkit.DataStructs.cDataStructs.ExplicitBitVect object at 0x7dc7c96495b0>,
<rdkit.DataStructs.cDataStructs.ExplicitBitVect object at 0x7dc7c9649620>,
<rdkit.DataStructs.cDataStructs.ExplicitBitVect object at 0x7dc7c9648cf0>,
<rdkit.DataStructs.cDataStructs.ExplicitBitVect object at 0x7dc7c9648270>,
<rdkit.DataStructs.cDataStructs.ExplicitBitVect object at 0x7dc7c9649700>,
<rdkit.DataStructs.cDataStructs.ExplicitBitVect object at 0x7dc7c9649770>,
<rdkit.DataStructs.cDataStructs.ExplicitBitVect object at 0x7dc7c96497e0>,
<rdkit.DataStructs.cDataStructs.ExplicitBitVect object at 0x7dc7c9649850>,
<rdkit.DataStructs.cDataStructs.ExplicitBitVect object at 0x7dc7c96498c0>,
<rdkit.DataStructs.cDataStructs.ExplicitBitVect object at 0x7dc7c9649930>,
<rdkit.DataStructs.cDataStructs.ExplicitBitVect object at 0x7dc7c96499a0>,
<rdkit.DataStructs.cDataStructs.ExplicitBitVect object at 0x7dc7c9649a10>,
<rdkit.DataStructs.cDataStructs.ExplicitBitVect object at 0x7dc7c9649a80>,
<rdkit.DataStructs.cDataStructs.ExplicitBitVect object at 0x7dc7c9649af0>,
<rdkit.DataStructs.cDataStructs.ExplicitBitVect object at 0x7dc7c9649b60>,
<rdkit.DataStructs.cDataStructs.ExplicitBitVect object at 0x7dc7c9649bd0>,
<rdkit.DataStructs.cDataStructs.ExplicitBitVect object at 0x7dc7c9649c40>,
<rdkit.DataStructs.cDataStructs.ExplicitBitVect object at 0x7dc7c9649cb0>,
<rdkit.DataStructs.cDataStructs.ExplicitBitVect object at 0x7dc7c9649d20>,
<rdkit.DataStructs.cDataStructs.ExplicitBitVect object at 0x7dc7c9649d90>,
<rdkit.DataStructs.cDataStructs.ExplicitBitVect object at 0x7dc7c9649e00>,
<rdkit.DataStructs.cDataStructs.ExplicitBitVect object at 0x7dc7c9649e70>,
<rdkit.DataStructs.cDataStructs.ExplicitBitVect object at 0x7dc7c9649ee0>,
<rdkit.DataStructs.cDataStructs.ExplicitBitVect object at 0x7dc7c9649f50>,
<rdkit.DataStructs.cDataStructs.ExplicitBitVect object at 0x7dc7c9649fc0>,
<rdkit.DataStructs.cDataStructs.ExplicitBitVect object at 0x7dc7c964a030>,
<rdkit.DataStructs.cDataStructs.ExplicitBitVect object at 0x7dc7c964a0a0>,
<rdkit.DataStructs.cDataStructs.ExplicitBitVect object at 0x7dc7c964a110>,
```

```
<rdkit.DataStructs.cDataStructs.ExplicitBitVect object at 0x7dc7c964a180>,
<rdkit.DataStructs.cDataStructs.ExplicitBitVect object at 0x7dc7c964a1f0>,
<rdkit.DataStructs.cDataStructs.ExplicitBitVect object at 0x7dc7c964a260>,
<rdkit.DataStructs.cDataStructs.ExplicitBitVect object at 0x7dc7c964a2d0>,
<rdkit.DataStructs.cDataStructs.ExplicitBitVect object at 0x7dc7c964a340>,
<rdkit.DataStructs.cDataStructs.ExplicitBitVect object at 0x7dc7c964a3b0>,
<rdkit.DataStructs.cDataStructs.ExplicitBitVect object at 0x7dc7c964a420>,
<rdkit.DataStructs.cDataStructs.ExplicitBitVect object at 0x7dc7c964a490>,
<rdkit.DataStructs.cDataStructs.ExplicitBitVect object at 0x7dc7c964a500>,
<rdkit.DataStructs.cDataStructs.ExplicitBitVect object at 0x7dc7c964a570>,
<rdkit.DataStructs.cDataStructs.ExplicitBitVect object at 0x7dc7c964a5e0>,
<rdkit.DataStructs.cDataStructs.ExplicitBitVect object at 0x7dc7c964a650>,
<rdkit.DataStructs.cDataStructs.ExplicitBitVect object at 0x7dc7c964a6c0>,
<rdkit.DataStructs.cDataStructs.ExplicitBitVect object at 0x7dc7c964a730>,
<rdkit.DataStructs.cDataStructs.ExplicitBitVect object at 0x7dc7c964a7a0>,
<rdkit.DataStructs.cDataStructs.ExplicitBitVect object at 0x7dc7c964a810>,
<rdkit.DataStructs.cDataStructs.ExplicitBitVect object at 0x7dc7c964a880>,
<rdkit.DataStructs.cDataStructs.ExplicitBitVect object at 0x7dc7c964a8f0>,
<rdkit.DataStructs.cDataStructs.ExplicitBitVect object at 0x7dc7c964a960>,
<rdkit.DataStructs.cDataStructs.ExplicitBitVect object at 0x7dc7c964a9d0>,
<rdkit.DataStructs.cDataStructs.ExplicitBitVect object at 0x7dc7c964aa40>,
<rdkit.DataStructs.cDataStructs.ExplicitBitVect object at 0x7dc7c964aab0>,
<rdkit.DataStructs.cDataStructs.ExplicitBitVect object at 0x7dc7c964ab20>,
<rdkit.DataStructs.cDataStructs.ExplicitBitVect object at 0x7dc7c964ab90>,
<rdkit.DataStructs.cDataStructs.ExplicitBitVect object at 0x7dc7c964ac00>,
<rdkit.DataStructs.cDataStructs.ExplicitBitVect object at 0x7dc7c964ac70>,
<rdkit.DataStructs.cDataStructs.ExplicitBitVect object at 0x7dc7c964ace0>,
<rdkit.DataStructs.cDataStructs.ExplicitBitVect object at 0x7dc7c964ad50>,
<rdkit.DataStructs.cDataStructs.ExplicitBitVect object at 0x7dc7c964adc0>,
<rdkit.DataStructs.cDataStructs.ExplicitBitVect object at 0x7dc7c964ae30>,
<rdkit.DataStructs.cDataStructs.ExplicitBitVect object at 0x7dc7c964aea0>,
<rdkit.DataStructs.cDataStructs.ExplicitBitVect object at 0x7dc7c964af10>,
<rdkit.DataStructs.cDataStructs.ExplicitBitVect object at 0x7dc7c964af80>,
<rdkit.DataStructs.cDataStructs.ExplicitBitVect object at 0x7dc7c964aff0>,
<rdkit.DataStructs.cDataStructs.ExplicitBitVect object at 0x7dc7c964b060>,
<rdkit.DataStructs.cDataStructs.ExplicitBitVect object at 0x7dc7c964b0d0>,
<rdkit.DataStructs.cDataStructs.ExplicitBitVect object at 0x7dc7c964b140>,
<rdkit.DataStructs.cDataStructs.ExplicitBitVect object at 0x7dc7c964b1b0>,
<rdkit.DataStructs.cDataStructs.ExplicitBitVect object at 0x7dc7c964b220>,
<rdkit.DataStructs.cDataStructs.ExplicitBitVect object at 0x7dc7c964b290>,
<rdkit.DataStructs.cDataStructs.ExplicitBitVect object at 0x7dc7c964b300>,
<rdkit.DataStructs.cDataStructs.ExplicitBitVect object at 0x7dc7c964b370>,
<rdkit.DataStructs.cDataStructs.ExplicitBitVect object at 0x7dc7c964b3e0>,
<rdkit.DataStructs.cDataStructs.ExplicitBitVect object at 0x7dc7c96b44a0>,
<rdkit.DataStructs.cDataStructs.ExplicitBitVect object at 0x7dc7c96b4510>,
<rdkit.DataStructs.cDataStructs.ExplicitBitVect object at 0x7dc7c96b4580>,
<rdkit.DataStructs.cDataStructs.ExplicitBitVect object at 0x7dc7c96b45f0>,
<rdkit.DataStructs.cDataStructs.ExplicitBitVect object at 0x7dc7c96b4660>,
```

```
<rdkit.DataStructs.cDataStructs.ExplicitBitVect object at 0x7dc7c96b46d0>,
<rdkit.DataStructs.cDataStructs.ExplicitBitVect object at 0x7dc7c96b4740>,
<rdkit.DataStructs.cDataStructs.ExplicitBitVect object at 0x7dc7c96b47b0>,
<rdkit.DataStructs.cDataStructs.ExplicitBitVect object at 0x7dc7c96b4820>,
<rdkit.DataStructs.cDataStructs.ExplicitBitVect object at 0x7dc7c96b4890>,
<rdkit.DataStructs.cDataStructs.ExplicitBitVect object at 0x7dc7c96b4900>,
<rdkit.DataStructs.cDataStructs.ExplicitBitVect object at 0x7dc7c96b4970>,
<rdkit.DataStructs.cDataStructs.ExplicitBitVect object at 0x7dc7c96b49e0>,
<rdkit.DataStructs.cDataStructs.ExplicitBitVect object at 0x7dc7c96b4a50>,
<rdkit.DataStructs.cDataStructs.ExplicitBitVect object at 0x7dc7c96b4ac0>,
<rdkit.DataStructs.cDataStructs.ExplicitBitVect object at 0x7dc7c96b4b30>,
<rdkit.DataStructs.cDataStructs.ExplicitBitVect object at 0x7dc7c96b4ba0>,
<rdkit.DataStructs.cDataStructs.ExplicitBitVect object at 0x7dc7c96b4c10>,
<rdkit.DataStructs.cDataStructs.ExplicitBitVect object at 0x7dc7c96b4c80>,
<rdkit.DataStructs.cDataStructs.ExplicitBitVect object at 0x7dc7c96b4cf0>,
<rdkit.DataStructs.cDataStructs.ExplicitBitVect object at 0x7dc7c96b4d60>,
<rdkit.DataStructs.cDataStructs.ExplicitBitVect object at 0x7dc7c96b4dd0>,
<rdkit.DataStructs.cDataStructs.ExplicitBitVect object at 0x7dc7c96b4e40>,
<rdkit.DataStructs.cDataStructs.ExplicitBitVect object at 0x7dc7c96b4eb0>,
<rdkit.DataStructs.cDataStructs.ExplicitBitVect object at 0x7dc7c96b4f20>,
<rdkit.DataStructs.cDataStructs.ExplicitBitVect object at 0x7dc7c96b4f90>,
<rdkit.DataStructs.cDataStructs.ExplicitBitVect object at 0x7dc7c96b5000>,
<rdkit.DataStructs.cDataStructs.ExplicitBitVect object at 0x7dc7c96b5070>,
<rdkit.DataStructs.cDataStructs.ExplicitBitVect object at 0x7dc7c96b50e0>,
<rdkit.DataStructs.cDataStructs.ExplicitBitVect object at 0x7dc7c96b5150>,
<rdkit.DataStructs.cDataStructs.ExplicitBitVect object at 0x7dc7c96b51c0>,
<rdkit.DataStructs.cDataStructs.ExplicitBitVect object at 0x7dc7c96b5230>,
<rdkit.DataStructs.cDataStructs.ExplicitBitVect object at 0x7dc7c96b52a0>,
<rdkit.DataStructs.cDataStructs.ExplicitBitVect object at 0x7dc7c96b5310>,
<rdkit.DataStructs.cDataStructs.ExplicitBitVect object at 0x7dc7c96b5380>,
<rdkit.DataStructs.cDataStructs.ExplicitBitVect object at 0x7dc7c96b53f0>,
<rdkit.DataStructs.cDataStructs.ExplicitBitVect object at 0x7dc7c96b5460>,
<rdkit.DataStructs.cDataStructs.ExplicitBitVect object at 0x7dc7c96b54d0>,
<rdkit.DataStructs.cDataStructs.ExplicitBitVect object at 0x7dc7c96b5540>,
<rdkit.DataStructs.cDataStructs.ExplicitBitVect object at 0x7dc7c96b55b0>,
<rdkit.DataStructs.cDataStructs.ExplicitBitVect object at 0x7dc7c96b5620>,
<rdkit.DataStructs.cDataStructs.ExplicitBitVect object at 0x7dc7c96b5690>,
<rdkit.DataStructs.cDataStructs.ExplicitBitVect object at 0x7dc7c96b5700>,
<rdkit.DataStructs.cDataStructs.ExplicitBitVect object at 0x7dc7c96b5770>,
<rdkit.DataStructs.cDataStructs.ExplicitBitVect object at 0x7dc7c96b57e0>,
<rdkit.DataStructs.cDataStructs.ExplicitBitVect object at 0x7dc7c96b5850>,
<rdkit.DataStructs.cDataStructs.ExplicitBitVect object at 0x7dc7c96b58c0>,
<rdkit.DataStructs.cDataStructs.ExplicitBitVect object at 0x7dc7c96b5930>,
<rdkit.DataStructs.cDataStructs.ExplicitBitVect object at 0x7dc7c96b59a0>,
<rdkit.DataStructs.cDataStructs.ExplicitBitVect object at 0x7dc7c96b5a10>,
<rdkit.DataStructs.cDataStructs.ExplicitBitVect object at 0x7dc7c96b5a80>,
<rdkit.DataStructs.cDataStructs.ExplicitBitVect object at 0x7dc7c96b5af0>,
<rdkit.DataStructs.cDataStructs.ExplicitBitVect object at 0x7dc7c96b5b60>,
```

```
<rdkit.DataStructs.cDataStructs.ExplicitBitVect object at 0x7dc7c96b5bd0>,
<rdkit.DataStructs.cDataStructs.ExplicitBitVect object at 0x7dc7c96b5c40>,
<rdkit.DataStructs.cDataStructs.ExplicitBitVect object at 0x7dc7c96b5cb0>,
<rdkit.DataStructs.cDataStructs.ExplicitBitVect object at 0x7dc7c96b5d20>,
<rdkit.DataStructs.cDataStructs.ExplicitBitVect object at 0x7dc7c96b5d90>,
<rdkit.DataStructs.cDataStructs.ExplicitBitVect object at 0x7dc7c96b5e00>,
<rdkit.DataStructs.cDataStructs.ExplicitBitVect object at 0x7dc7c96b5e70>,
<rdkit.DataStructs.cDataStructs.ExplicitBitVect object at 0x7dc7c96b5ee0>,
<rdkit.DataStructs.cDataStructs.ExplicitBitVect object at 0x7dc7c96b5f50>,
<rdkit.DataStructs.cDataStructs.ExplicitBitVect object at 0x7dc7c96b5fc0>,
<rdkit.DataStructs.cDataStructs.ExplicitBitVect object at 0x7dc7c96b6030>,
<rdkit.DataStructs.cDataStructs.ExplicitBitVect object at 0x7dc7c96b60a0>,
<rdkit.DataStructs.cDataStructs.ExplicitBitVect object at 0x7dc7c96b6110>,
<rdkit.DataStructs.cDataStructs.ExplicitBitVect object at 0x7dc7c96b6180>,
<rdkit.DataStructs.cDataStructs.ExplicitBitVect object at 0x7dc7c96b61f0>,
<rdkit.DataStructs.cDataStructs.ExplicitBitVect object at 0x7dc7c96b6260>,
<rdkit.DataStructs.cDataStructs.ExplicitBitVect object at 0x7dc7c96b62d0>,
<rdkit.DataStructs.cDataStructs.ExplicitBitVect object at 0x7dc7c96b6340>,
<rdkit.DataStructs.cDataStructs.ExplicitBitVect object at 0x7dc7c96b63b0>,
<rdkit.DataStructs.cDataStructs.ExplicitBitVect object at 0x7dc7c96b6420>,
<rdkit.DataStructs.cDataStructs.ExplicitBitVect object at 0x7dc7c96b6490>,
<rdkit.DataStructs.cDataStructs.ExplicitBitVect object at 0x7dc7c96b6500>,
<rdkit.DataStructs.cDataStructs.ExplicitBitVect object at 0x7dc7c96b6570>,
<rdkit.DataStructs.cDataStructs.ExplicitBitVect object at 0x7dc7c96b65e0>,
<rdkit.DataStructs.cDataStructs.ExplicitBitVect object at 0x7dc7c96b6650>,
<rdkit.DataStructs.cDataStructs.ExplicitBitVect object at 0x7dc7c96b66c0>,
<rdkit.DataStructs.cDataStructs.ExplicitBitVect object at 0x7dc7c96b6730>,
<rdkit.DataStructs.cDataStructs.ExplicitBitVect object at 0x7dc7c96b67a0>,
<rdkit.DataStructs.cDataStructs.ExplicitBitVect object at 0x7dc7c96b6810>,
<rdkit.DataStructs.cDataStructs.ExplicitBitVect object at 0x7dc7c96b6880>,
<rdkit.DataStructs.cDataStructs.ExplicitBitVect object at 0x7dc7c96b68f0>,
<rdkit.DataStructs.cDataStructs.ExplicitBitVect object at 0x7dc7c96b6960>,
<rdkit.DataStructs.cDataStructs.ExplicitBitVect object at 0x7dc7c96b69d0>,
<rdkit.DataStructs.cDataStructs.ExplicitBitVect object at 0x7dc7c96b6a40>,
<rdkit.DataStructs.cDataStructs.ExplicitBitVect object at 0x7dc7c96b6ab0>,
<rdkit.DataStructs.cDataStructs.ExplicitBitVect object at 0x7dc7c96b6b20>]
```

[ ]:
```python
#Convert Fingerprints to Arrays
import numpy as np
from rdkit import DataStructs

# Convert each fingerprint into a NumPy array
fingerprint_array = []
for fp in fingerprints:
    arr = np.zeros((1,))
    DataStructs.ConvertToNumpyArray(fp, arr)
    fingerprint_array.append(arr)
```

```
# Convert to a NumPy array for easier manipulation (e.g., for machine learning)
X = np.array(fingerprint_array)

# Check the shape of the resulting feature matrix
print(X.shape)  # Should be (number_of_compounds, 1024) if using 1024-bit␣
↪fingerprints
```

```
(1332, 1024)
```

[ ]:
```
# Prepare Target (Adverse Reaction Labels)

# Assuming 'adr_df' contains columns for each adverse reaction
reaction_columns = ['Hepatobiliary disorders', 'Metabolism and nutrition␣
↪disorders', 'Eye disorders',
                    'Musculoskeletal and connective tissue disorders',␣
↪'Gastrointestinal disorders',
                    'Immune system disorders', 'Reproductive system and breast␣
↪disorders',
                    'Neoplasms benign, malignant and unspecified (incl cysts␣
↪and polyps)',
                    'General disorders and administration site conditions',␣
↪'Endocrine disorders',
                    'Surgical and medical procedures', 'Vascular disorders',␣
↪'Blood and lymphatic system disorders',
                    'Skin and subcutaneous tissue disorders', 'Congenital,␣
↪familial and genetic disorders',
                    'Infections and infestations', 'Respiratory, thoracic and␣
↪mediastinal disorders',
                    'Psychiatric disorders', 'Renal and urinary disorders',␣
↪'Pregnancy, puerperium and perinatal conditions',
                    'Ear and labyrinth disorders', 'Cardiac disorders',␣
↪'Nervous system disorders',
                    'Injury, poisoning and procedural complications']

# Extract the target labels
y = adr_df[reaction_columns].values

# Check the shape of the target matrix
print(y.shape)  # Should be (number_of_compounds, number_of_reactions)
```

```
(1332, 24)
```

[ ]:
```
# Split the Data

from sklearn.model_selection import train_test_split
```

```
# Split the data into training and testing sets
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2,␣
↪random_state=42)

# Check the shape of the training and test sets
print(X_train.shape, X_test.shape)
```

```
(1065, 1024) (267, 1024)
```

[ ]:
```
# 1. MLP with hypparameter tuning = 10,15,20,25,30,35,40,50

import torch
import torch.nn.functional as F
from sklearn.metrics import accuracy_score, f1_score
from sklearn.model_selection import train_test_split
import numpy as np

# Use your fingerprint array `X` from your previous code
# X represents the features, and y represents the labels
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2,␣
↪random_state=42)

# Convert X_train and y_train to PyTorch tensors
X_train_tensor = torch.tensor(X_train, dtype=torch.float)
X_test_tensor = torch.tensor(X_test, dtype=torch.float)
y_train_tensor = torch.tensor(y_train, dtype=torch.float)
y_test_tensor = torch.tensor(y_test, dtype=torch.float)

# Define a simple MLP model
class MLP(torch.nn.Module):
    def __init__(self, input_dim, hidden_dim, output_dim, dropout=0.5,␣
↪num_layers=2, activation=F.relu):
        super(MLP, self).__init__()
        self.layers = torch.nn.ModuleList()
        self.dropout = dropout
        self.activation = activation

        # Add hidden layers
        self.layers.append(torch.nn.Linear(input_dim, hidden_dim))
        for _ in range(num_layers - 1):
            self.layers.append(torch.nn.Linear(hidden_dim, hidden_dim))

        # Output layer
        self.fc = torch.nn.Linear(hidden_dim, output_dim)

    def forward(self, x):
        for layer in self.layers:
```

```
            x = self.activation(layer(x))
            x = F.dropout(x, p=self.dropout, training=self.training)
        x = self.fc(x)
        return torch.sigmoid(x)  # Multi-label classification

# Hyperparameter grid
param_grid = {
    'hidden_dim': [64, 128],  # Number of hidden units
    'dropout': [0.3, 0.5],  # Dropout rate
    'epochs': [10,20,25,15,30,35,40,50],  # Number of epochs
    'batch_size': [32, 64],  # Batch size
    'learning_rate': [0.01, 0.001],  # Learning rates
    'num_layers': [2, 3],  # Number of hidden layers
    'activation': ['relu', 'tanh'],  # Activation functions
}

# Variables to track the best model
best_binary_acc = 0
best_f1 = 0
best_normal_acc = 0
best_params = {}

# Perform hyperparameter tuning
for hidden_dim in param_grid['hidden_dim']:
    for dropout in param_grid['dropout']:
        for lr in param_grid['learning_rate']:
            for batch_size in param_grid['batch_size']:
                for num_layers in param_grid['num_layers']:
                    for activation in param_grid['activation']:
                      for epochs in param_grid['epochs']:
                        print(f'Training with hidden_dim={hidden_dim}, dropout={dropout}, lr={lr}, batch_size={batch_size}, num_layers={num_layers}, activation={activation}, epochs={epochs}')

                        # Initialize activation function
                        if activation == 'relu':
                            activation_fn = F.relu
                        elif activation == 'tanh':
                            activation_fn = torch.tanh

                        # Initialize model
                        model = MLP(input_dim=X_train.shape[1], hidden_dim=hidden_dim, output_dim=y_train.shape[1], dropout=dropout, num_layers=num_layers, activation=activation_fn)

                        # Loss and optimizer
                        optimizer = torch.optim.Adam(model.parameters(), lr=lr)
```

```
                        criterion = torch.nn.BCELoss()

                        # Train the model
                        for epoch in range(param_grid['epochs'][0]):
                            model.train()
                            optimizer.zero_grad()
                            out = model(X_train_tensor)
                            loss = criterion(out, y_train_tensor)
                            loss.backward()
                            optimizer.step()

                        # Evaluate the model
                        model.eval()
                        with torch.no_grad():
                            y_pred = model(X_test_tensor)
                            y_pred_binary = (y_pred > 0.5).float()

                        # Calculate binary accuracy
                        binary_acc = (y_pred_binary == y_test_tensor).float().
 ↪mean().item()

                        # Calculate normal accuracy
                        normal_acc = accuracy_score(y_test, y_pred_binary.
 ↪numpy())

                        # Calculate F1 score
                        f1 = f1_score(y_test, y_pred_binary.numpy(),␣
 ↪average='micro')

                        # Update best params if this model is better
                        if binary_acc > best_binary_acc:
                            best_binary_acc = binary_acc
                            best_f1 = f1
                            best_normal_acc = normal_acc
                            best_params = {
                                'hidden_dim': hidden_dim,
                                'dropout': dropout,
                                'lr': lr,
                                'batch_size': batch_size,
                                'num_layers': num_layers,
                                'activation': activation,
                            }

# Print the best results
print(f'Best Params: {best_params}')
print(f'Best Binary Accuracy: {best_binary_acc:.4f}')
print(f'Best Normal Accuracy: {best_normal_acc:.4f}')
```

```python
print(f'Best F1 Score: {best_f1:.4f}')
```

```
Training with hidden_dim=64, dropout=0.3, lr=0.01, batch_size=32, num_layers=2,
activation=relu, epochs=10
Training with hidden_dim=64, dropout=0.3, lr=0.01, batch_size=32, num_layers=2,
activation=relu, epochs=20
Training with hidden_dim=64, dropout=0.3, lr=0.01, batch_size=32, num_layers=2,
activation=relu, epochs=25
Training with hidden_dim=64, dropout=0.3, lr=0.01, batch_size=32, num_layers=2,
activation=relu, epochs=15
Training with hidden_dim=64, dropout=0.3, lr=0.01, batch_size=32, num_layers=2,
activation=relu, epochs=30
Training with hidden_dim=64, dropout=0.3, lr=0.01, batch_size=32, num_layers=2,
activation=relu, epochs=35
Training with hidden_dim=64, dropout=0.3, lr=0.01, batch_size=32, num_layers=2,
activation=relu, epochs=40
Training with hidden_dim=64, dropout=0.3, lr=0.01, batch_size=32, num_layers=2,
activation=relu, epochs=50
Training with hidden_dim=64, dropout=0.3, lr=0.01, batch_size=32, num_layers=2,
activation=tanh, epochs=10
Training with hidden_dim=64, dropout=0.3, lr=0.01, batch_size=32, num_layers=2,
activation=tanh, epochs=20
Training with hidden_dim=64, dropout=0.3, lr=0.01, batch_size=32, num_layers=2,
activation=tanh, epochs=25
Training with hidden_dim=64, dropout=0.3, lr=0.01, batch_size=32, num_layers=2,
activation=tanh, epochs=15
Training with hidden_dim=64, dropout=0.3, lr=0.01, batch_size=32, num_layers=2,
activation=tanh, epochs=30
Training with hidden_dim=64, dropout=0.3, lr=0.01, batch_size=32, num_layers=2,
activation=tanh, epochs=35
Training with hidden_dim=64, dropout=0.3, lr=0.01, batch_size=32, num_layers=2,
activation=tanh, epochs=40
Training with hidden_dim=64, dropout=0.3, lr=0.01, batch_size=32, num_layers=2,
activation=tanh, epochs=50
Training with hidden_dim=64, dropout=0.3, lr=0.01, batch_size=32, num_layers=3,
activation=relu, epochs=10
Training with hidden_dim=64, dropout=0.3, lr=0.01, batch_size=32, num_layers=3,
activation=relu, epochs=20
Training with hidden_dim=64, dropout=0.3, lr=0.01, batch_size=32, num_layers=3,
activation=relu, epochs=25
Training with hidden_dim=64, dropout=0.3, lr=0.01, batch_size=32, num_layers=3,
activation=relu, epochs=15
Training with hidden_dim=64, dropout=0.3, lr=0.01, batch_size=32, num_layers=3,
activation=relu, epochs=30
Training with hidden_dim=64, dropout=0.3, lr=0.01, batch_size=32, num_layers=3,
activation=relu, epochs=35
Training with hidden_dim=64, dropout=0.3, lr=0.01, batch_size=32, num_layers=3,
```

```
activation=relu, epochs=40
Training with hidden_dim=64, dropout=0.3, lr=0.01, batch_size=32, num_layers=3,
activation=relu, epochs=50
Training with hidden_dim=64, dropout=0.3, lr=0.01, batch_size=32, num_layers=3,
activation=tanh, epochs=10
Training with hidden_dim=64, dropout=0.3, lr=0.01, batch_size=32, num_layers=3,
activation=tanh, epochs=20
Training with hidden_dim=64, dropout=0.3, lr=0.01, batch_size=32, num_layers=3,
activation=tanh, epochs=25
Training with hidden_dim=64, dropout=0.3, lr=0.01, batch_size=32, num_layers=3,
activation=tanh, epochs=15
Training with hidden_dim=64, dropout=0.3, lr=0.01, batch_size=32, num_layers=3,
activation=tanh, epochs=30
Training with hidden_dim=64, dropout=0.3, lr=0.01, batch_size=32, num_layers=3,
activation=tanh, epochs=35
Training with hidden_dim=64, dropout=0.3, lr=0.01, batch_size=32, num_layers=3,
activation=tanh, epochs=40
Training with hidden_dim=64, dropout=0.3, lr=0.01, batch_size=32, num_layers=3,
activation=tanh, epochs=50
Training with hidden_dim=64, dropout=0.3, lr=0.01, batch_size=64, num_layers=2,
activation=relu, epochs=10
Training with hidden_dim=64, dropout=0.3, lr=0.01, batch_size=64, num_layers=2,
activation=relu, epochs=20
Training with hidden_dim=64, dropout=0.3, lr=0.01, batch_size=64, num_layers=2,
activation=relu, epochs=25
Training with hidden_dim=64, dropout=0.3, lr=0.01, batch_size=64, num_layers=2,
activation=relu, epochs=15
Training with hidden_dim=64, dropout=0.3, lr=0.01, batch_size=64, num_layers=2,
activation=relu, epochs=30
Training with hidden_dim=64, dropout=0.3, lr=0.01, batch_size=64, num_layers=2,
activation=relu, epochs=35
Training with hidden_dim=64, dropout=0.3, lr=0.01, batch_size=64, num_layers=2,
activation=relu, epochs=40
Training with hidden_dim=64, dropout=0.3, lr=0.01, batch_size=64, num_layers=2,
activation=relu, epochs=50
Training with hidden_dim=64, dropout=0.3, lr=0.01, batch_size=64, num_layers=2,
activation=tanh, epochs=10
Training with hidden_dim=64, dropout=0.3, lr=0.01, batch_size=64, num_layers=2,
activation=tanh, epochs=20
Training with hidden_dim=64, dropout=0.3, lr=0.01, batch_size=64, num_layers=2,
activation=tanh, epochs=25
Training with hidden_dim=64, dropout=0.3, lr=0.01, batch_size=64, num_layers=2,
activation=tanh, epochs=15
Training with hidden_dim=64, dropout=0.3, lr=0.01, batch_size=64, num_layers=2,
activation=tanh, epochs=30
Training with hidden_dim=64, dropout=0.3, lr=0.01, batch_size=64, num_layers=2,
activation=tanh, epochs=35
Training with hidden_dim=64, dropout=0.3, lr=0.01, batch_size=64, num_layers=2,
```

```
activation=tanh, epochs=40
Training with hidden_dim=64, dropout=0.3, lr=0.01, batch_size=64, num_layers=2,
activation=tanh, epochs=50
Training with hidden_dim=64, dropout=0.3, lr=0.01, batch_size=64, num_layers=3,
activation=relu, epochs=10
Training with hidden_dim=64, dropout=0.3, lr=0.01, batch_size=64, num_layers=3,
activation=relu, epochs=20
Training with hidden_dim=64, dropout=0.3, lr=0.01, batch_size=64, num_layers=3,
activation=relu, epochs=25
Training with hidden_dim=64, dropout=0.3, lr=0.01, batch_size=64, num_layers=3,
activation=relu, epochs=15
Training with hidden_dim=64, dropout=0.3, lr=0.01, batch_size=64, num_layers=3,
activation=relu, epochs=30
Training with hidden_dim=64, dropout=0.3, lr=0.01, batch_size=64, num_layers=3,
activation=relu, epochs=35
Training with hidden_dim=64, dropout=0.3, lr=0.01, batch_size=64, num_layers=3,
activation=relu, epochs=40
Training with hidden_dim=64, dropout=0.3, lr=0.01, batch_size=64, num_layers=3,
activation=relu, epochs=50
Training with hidden_dim=64, dropout=0.3, lr=0.01, batch_size=64, num_layers=3,
activation=tanh, epochs=10
Training with hidden_dim=64, dropout=0.3, lr=0.01, batch_size=64, num_layers=3,
activation=tanh, epochs=20
Training with hidden_dim=64, dropout=0.3, lr=0.01, batch_size=64, num_layers=3,
activation=tanh, epochs=25
Training with hidden_dim=64, dropout=0.3, lr=0.01, batch_size=64, num_layers=3,
activation=tanh, epochs=15
Training with hidden_dim=64, dropout=0.3, lr=0.01, batch_size=64, num_layers=3,
activation=tanh, epochs=30
Training with hidden_dim=64, dropout=0.3, lr=0.01, batch_size=64, num_layers=3,
activation=tanh, epochs=35
Training with hidden_dim=64, dropout=0.3, lr=0.01, batch_size=64, num_layers=3,
activation=tanh, epochs=40
Training with hidden_dim=64, dropout=0.3, lr=0.01, batch_size=64, num_layers=3,
activation=tanh, epochs=50
Training with hidden_dim=64, dropout=0.3, lr=0.001, batch_size=32, num_layers=2,
activation=relu, epochs=10
Training with hidden_dim=64, dropout=0.3, lr=0.001, batch_size=32, num_layers=2,
activation=relu, epochs=20
Training with hidden_dim=64, dropout=0.3, lr=0.001, batch_size=32, num_layers=2,
activation=relu, epochs=25
Training with hidden_dim=64, dropout=0.3, lr=0.001, batch_size=32, num_layers=2,
activation=relu, epochs=15
Training with hidden_dim=64, dropout=0.3, lr=0.001, batch_size=32, num_layers=2,
activation=relu, epochs=30
Training with hidden_dim=64, dropout=0.3, lr=0.001, batch_size=32, num_layers=2,
activation=relu, epochs=35
Training with hidden_dim=64, dropout=0.3, lr=0.001, batch_size=32, num_layers=2,
```

```
activation=relu, epochs=40
Training with hidden_dim=64, dropout=0.3, lr=0.001, batch_size=32, num_layers=2,
activation=relu, epochs=50
Training with hidden_dim=64, dropout=0.3, lr=0.001, batch_size=32, num_layers=2,
activation=tanh, epochs=10
Training with hidden_dim=64, dropout=0.3, lr=0.001, batch_size=32, num_layers=2,
activation=tanh, epochs=20
Training with hidden_dim=64, dropout=0.3, lr=0.001, batch_size=32, num_layers=2,
activation=tanh, epochs=25
Training with hidden_dim=64, dropout=0.3, lr=0.001, batch_size=32, num_layers=2,
activation=tanh, epochs=15
Training with hidden_dim=64, dropout=0.3, lr=0.001, batch_size=32, num_layers=2,
activation=tanh, epochs=30
Training with hidden_dim=64, dropout=0.3, lr=0.001, batch_size=32, num_layers=2,
activation=tanh, epochs=35
Training with hidden_dim=64, dropout=0.3, lr=0.001, batch_size=32, num_layers=2,
activation=tanh, epochs=40
Training with hidden_dim=64, dropout=0.3, lr=0.001, batch_size=32, num_layers=2,
activation=tanh, epochs=50
Training with hidden_dim=64, dropout=0.3, lr=0.001, batch_size=32, num_layers=3,
activation=relu, epochs=10
Training with hidden_dim=64, dropout=0.3, lr=0.001, batch_size=32, num_layers=3,
activation=relu, epochs=20
Training with hidden_dim=64, dropout=0.3, lr=0.001, batch_size=32, num_layers=3,
activation=relu, epochs=25
Training with hidden_dim=64, dropout=0.3, lr=0.001, batch_size=32, num_layers=3,
activation=relu, epochs=15
Training with hidden_dim=64, dropout=0.3, lr=0.001, batch_size=32, num_layers=3,
activation=relu, epochs=30
Training with hidden_dim=64, dropout=0.3, lr=0.001, batch_size=32, num_layers=3,
activation=relu, epochs=35
Training with hidden_dim=64, dropout=0.3, lr=0.001, batch_size=32, num_layers=3,
activation=relu, epochs=40
Training with hidden_dim=64, dropout=0.3, lr=0.001, batch_size=32, num_layers=3,
activation=relu, epochs=50
Training with hidden_dim=64, dropout=0.3, lr=0.001, batch_size=32, num_layers=3,
activation=tanh, epochs=10
Training with hidden_dim=64, dropout=0.3, lr=0.001, batch_size=32, num_layers=3,
activation=tanh, epochs=20
Training with hidden_dim=64, dropout=0.3, lr=0.001, batch_size=32, num_layers=3,
activation=tanh, epochs=25
Training with hidden_dim=64, dropout=0.3, lr=0.001, batch_size=32, num_layers=3,
activation=tanh, epochs=15
Training with hidden_dim=64, dropout=0.3, lr=0.001, batch_size=32, num_layers=3,
activation=tanh, epochs=30
Training with hidden_dim=64, dropout=0.3, lr=0.001, batch_size=32, num_layers=3,
activation=tanh, epochs=35
Training with hidden_dim=64, dropout=0.3, lr=0.001, batch_size=32, num_layers=3,
```

```
activation=tanh, epochs=40
Training with hidden_dim=64, dropout=0.3, lr=0.001, batch_size=32, num_layers=3,
activation=tanh, epochs=50
Training with hidden_dim=64, dropout=0.3, lr=0.001, batch_size=64, num_layers=2,
activation=relu, epochs=10
Training with hidden_dim=64, dropout=0.3, lr=0.001, batch_size=64, num_layers=2,
activation=relu, epochs=20
Training with hidden_dim=64, dropout=0.3, lr=0.001, batch_size=64, num_layers=2,
activation=relu, epochs=25
Training with hidden_dim=64, dropout=0.3, lr=0.001, batch_size=64, num_layers=2,
activation=relu, epochs=15
Training with hidden_dim=64, dropout=0.3, lr=0.001, batch_size=64, num_layers=2,
activation=relu, epochs=30
Training with hidden_dim=64, dropout=0.3, lr=0.001, batch_size=64, num_layers=2,
activation=relu, epochs=35
Training with hidden_dim=64, dropout=0.3, lr=0.001, batch_size=64, num_layers=2,
activation=relu, epochs=40
Training with hidden_dim=64, dropout=0.3, lr=0.001, batch_size=64, num_layers=2,
activation=relu, epochs=50
Training with hidden_dim=64, dropout=0.3, lr=0.001, batch_size=64, num_layers=2,
activation=tanh, epochs=10
Training with hidden_dim=64, dropout=0.3, lr=0.001, batch_size=64, num_layers=2,
activation=tanh, epochs=20
Training with hidden_dim=64, dropout=0.3, lr=0.001, batch_size=64, num_layers=2,
activation=tanh, epochs=25
Training with hidden_dim=64, dropout=0.3, lr=0.001, batch_size=64, num_layers=2,
activation=tanh, epochs=15
Training with hidden_dim=64, dropout=0.3, lr=0.001, batch_size=64, num_layers=2,
activation=tanh, epochs=30
Training with hidden_dim=64, dropout=0.3, lr=0.001, batch_size=64, num_layers=2,
activation=tanh, epochs=35
Training with hidden_dim=64, dropout=0.3, lr=0.001, batch_size=64, num_layers=2,
activation=tanh, epochs=40
Training with hidden_dim=64, dropout=0.3, lr=0.001, batch_size=64, num_layers=2,
activation=tanh, epochs=50
Training with hidden_dim=64, dropout=0.3, lr=0.001, batch_size=64, num_layers=3,
activation=relu, epochs=10
Training with hidden_dim=64, dropout=0.3, lr=0.001, batch_size=64, num_layers=3,
activation=relu, epochs=20
Training with hidden_dim=64, dropout=0.3, lr=0.001, batch_size=64, num_layers=3,
activation=relu, epochs=25
Training with hidden_dim=64, dropout=0.3, lr=0.001, batch_size=64, num_layers=3,
activation=relu, epochs=15
Training with hidden_dim=64, dropout=0.3, lr=0.001, batch_size=64, num_layers=3,
activation=relu, epochs=30
Training with hidden_dim=64, dropout=0.3, lr=0.001, batch_size=64, num_layers=3,
activation=relu, epochs=35
Training with hidden_dim=64, dropout=0.3, lr=0.001, batch_size=64, num_layers=3,
```

```
activation=relu, epochs=40
Training with hidden_dim=64, dropout=0.3, lr=0.001, batch_size=64, num_layers=3,
activation=relu, epochs=50
Training with hidden_dim=64, dropout=0.3, lr=0.001, batch_size=64, num_layers=3,
activation=tanh, epochs=10
Training with hidden_dim=64, dropout=0.3, lr=0.001, batch_size=64, num_layers=3,
activation=tanh, epochs=20
Training with hidden_dim=64, dropout=0.3, lr=0.001, batch_size=64, num_layers=3,
activation=tanh, epochs=25
Training with hidden_dim=64, dropout=0.3, lr=0.001, batch_size=64, num_layers=3,
activation=tanh, epochs=15
Training with hidden_dim=64, dropout=0.3, lr=0.001, batch_size=64, num_layers=3,
activation=tanh, epochs=30
Training with hidden_dim=64, dropout=0.3, lr=0.001, batch_size=64, num_layers=3,
activation=tanh, epochs=35
Training with hidden_dim=64, dropout=0.3, lr=0.001, batch_size=64, num_layers=3,
activation=tanh, epochs=40
Training with hidden_dim=64, dropout=0.3, lr=0.001, batch_size=64, num_layers=3,
activation=tanh, epochs=50
Training with hidden_dim=64, dropout=0.5, lr=0.01, batch_size=32, num_layers=2,
activation=relu, epochs=10
Training with hidden_dim=64, dropout=0.5, lr=0.01, batch_size=32, num_layers=2,
activation=relu, epochs=20
Training with hidden_dim=64, dropout=0.5, lr=0.01, batch_size=32, num_layers=2,
activation=relu, epochs=25
Training with hidden_dim=64, dropout=0.5, lr=0.01, batch_size=32, num_layers=2,
activation=relu, epochs=15
Training with hidden_dim=64, dropout=0.5, lr=0.01, batch_size=32, num_layers=2,
activation=relu, epochs=30
Training with hidden_dim=64, dropout=0.5, lr=0.01, batch_size=32, num_layers=2,
activation=relu, epochs=35
Training with hidden_dim=64, dropout=0.5, lr=0.01, batch_size=32, num_layers=2,
activation=relu, epochs=40
Training with hidden_dim=64, dropout=0.5, lr=0.01, batch_size=32, num_layers=2,
activation=relu, epochs=50
Training with hidden_dim=64, dropout=0.5, lr=0.01, batch_size=32, num_layers=2,
activation=tanh, epochs=10
Training with hidden_dim=64, dropout=0.5, lr=0.01, batch_size=32, num_layers=2,
activation=tanh, epochs=20
Training with hidden_dim=64, dropout=0.5, lr=0.01, batch_size=32, num_layers=2,
activation=tanh, epochs=25
Training with hidden_dim=64, dropout=0.5, lr=0.01, batch_size=32, num_layers=2,
activation=tanh, epochs=15
Training with hidden_dim=64, dropout=0.5, lr=0.01, batch_size=32, num_layers=2,
activation=tanh, epochs=30
Training with hidden_dim=64, dropout=0.5, lr=0.01, batch_size=32, num_layers=2,
activation=tanh, epochs=35
Training with hidden_dim=64, dropout=0.5, lr=0.01, batch_size=32, num_layers=2,
```

```
activation=tanh, epochs=40
Training with hidden_dim=64, dropout=0.5, lr=0.01, batch_size=32, num_layers=2,
activation=tanh, epochs=50
Training with hidden_dim=64, dropout=0.5, lr=0.01, batch_size=32, num_layers=3,
activation=relu, epochs=10
Training with hidden_dim=64, dropout=0.5, lr=0.01, batch_size=32, num_layers=3,
activation=relu, epochs=20
Training with hidden_dim=64, dropout=0.5, lr=0.01, batch_size=32, num_layers=3,
activation=relu, epochs=25
Training with hidden_dim=64, dropout=0.5, lr=0.01, batch_size=32, num_layers=3,
activation=relu, epochs=15
Training with hidden_dim=64, dropout=0.5, lr=0.01, batch_size=32, num_layers=3,
activation=relu, epochs=30
Training with hidden_dim=64, dropout=0.5, lr=0.01, batch_size=32, num_layers=3,
activation=relu, epochs=35
Training with hidden_dim=64, dropout=0.5, lr=0.01, batch_size=32, num_layers=3,
activation=relu, epochs=40
Training with hidden_dim=64, dropout=0.5, lr=0.01, batch_size=32, num_layers=3,
activation=relu, epochs=50
Training with hidden_dim=64, dropout=0.5, lr=0.01, batch_size=32, num_layers=3,
activation=tanh, epochs=10
Training with hidden_dim=64, dropout=0.5, lr=0.01, batch_size=32, num_layers=3,
activation=tanh, epochs=20
Training with hidden_dim=64, dropout=0.5, lr=0.01, batch_size=32, num_layers=3,
activation=tanh, epochs=25
Training with hidden_dim=64, dropout=0.5, lr=0.01, batch_size=32, num_layers=3,
activation=tanh, epochs=15
Training with hidden_dim=64, dropout=0.5, lr=0.01, batch_size=32, num_layers=3,
activation=tanh, epochs=30
Training with hidden_dim=64, dropout=0.5, lr=0.01, batch_size=32, num_layers=3,
activation=tanh, epochs=35
Training with hidden_dim=64, dropout=0.5, lr=0.01, batch_size=32, num_layers=3,
activation=tanh, epochs=40
Training with hidden_dim=64, dropout=0.5, lr=0.01, batch_size=32, num_layers=3,
activation=tanh, epochs=50
Training with hidden_dim=64, dropout=0.5, lr=0.01, batch_size=64, num_layers=2,
activation=relu, epochs=10
Training with hidden_dim=64, dropout=0.5, lr=0.01, batch_size=64, num_layers=2,
activation=relu, epochs=20
Training with hidden_dim=64, dropout=0.5, lr=0.01, batch_size=64, num_layers=2,
activation=relu, epochs=25
Training with hidden_dim=64, dropout=0.5, lr=0.01, batch_size=64, num_layers=2,
activation=relu, epochs=15
Training with hidden_dim=64, dropout=0.5, lr=0.01, batch_size=64, num_layers=2,
activation=relu, epochs=30
Training with hidden_dim=64, dropout=0.5, lr=0.01, batch_size=64, num_layers=2,
activation=relu, epochs=35
Training with hidden_dim=64, dropout=0.5, lr=0.01, batch_size=64, num_layers=2,
```

```
activation=relu, epochs=40
Training with hidden_dim=64, dropout=0.5, lr=0.01, batch_size=64, num_layers=2,
activation=relu, epochs=50
Training with hidden_dim=64, dropout=0.5, lr=0.01, batch_size=64, num_layers=2,
activation=tanh, epochs=10
Training with hidden_dim=64, dropout=0.5, lr=0.01, batch_size=64, num_layers=2,
activation=tanh, epochs=20
Training with hidden_dim=64, dropout=0.5, lr=0.01, batch_size=64, num_layers=2,
activation=tanh, epochs=25
Training with hidden_dim=64, dropout=0.5, lr=0.01, batch_size=64, num_layers=2,
activation=tanh, epochs=15
Training with hidden_dim=64, dropout=0.5, lr=0.01, batch_size=64, num_layers=2,
activation=tanh, epochs=30
Training with hidden_dim=64, dropout=0.5, lr=0.01, batch_size=64, num_layers=2,
activation=tanh, epochs=35
Training with hidden_dim=64, dropout=0.5, lr=0.01, batch_size=64, num_layers=2,
activation=tanh, epochs=40
Training with hidden_dim=64, dropout=0.5, lr=0.01, batch_size=64, num_layers=2,
activation=tanh, epochs=50
Training with hidden_dim=64, dropout=0.5, lr=0.01, batch_size=64, num_layers=3,
activation=relu, epochs=10
Training with hidden_dim=64, dropout=0.5, lr=0.01, batch_size=64, num_layers=3,
activation=relu, epochs=20
Training with hidden_dim=64, dropout=0.5, lr=0.01, batch_size=64, num_layers=3,
activation=relu, epochs=25
Training with hidden_dim=64, dropout=0.5, lr=0.01, batch_size=64, num_layers=3,
activation=relu, epochs=15
Training with hidden_dim=64, dropout=0.5, lr=0.01, batch_size=64, num_layers=3,
activation=relu, epochs=30
Training with hidden_dim=64, dropout=0.5, lr=0.01, batch_size=64, num_layers=3,
activation=relu, epochs=35
Training with hidden_dim=64, dropout=0.5, lr=0.01, batch_size=64, num_layers=3,
activation=relu, epochs=40
Training with hidden_dim=64, dropout=0.5, lr=0.01, batch_size=64, num_layers=3,
activation=relu, epochs=50
Training with hidden_dim=64, dropout=0.5, lr=0.01, batch_size=64, num_layers=3,
activation=tanh, epochs=10
Training with hidden_dim=64, dropout=0.5, lr=0.01, batch_size=64, num_layers=3,
activation=tanh, epochs=20
Training with hidden_dim=64, dropout=0.5, lr=0.01, batch_size=64, num_layers=3,
activation=tanh, epochs=25
Training with hidden_dim=64, dropout=0.5, lr=0.01, batch_size=64, num_layers=3,
activation=tanh, epochs=15
Training with hidden_dim=64, dropout=0.5, lr=0.01, batch_size=64, num_layers=3,
activation=tanh, epochs=30
Training with hidden_dim=64, dropout=0.5, lr=0.01, batch_size=64, num_layers=3,
activation=tanh, epochs=35
Training with hidden_dim=64, dropout=0.5, lr=0.01, batch_size=64, num_layers=3,
```

```
activation=tanh, epochs=40
Training with hidden_dim=64, dropout=0.5, lr=0.01, batch_size=64, num_layers=3,
activation=tanh, epochs=50
Training with hidden_dim=64, dropout=0.5, lr=0.001, batch_size=32, num_layers=2,
activation=relu, epochs=10
Training with hidden_dim=64, dropout=0.5, lr=0.001, batch_size=32, num_layers=2,
activation=relu, epochs=20
Training with hidden_dim=64, dropout=0.5, lr=0.001, batch_size=32, num_layers=2,
activation=relu, epochs=25
Training with hidden_dim=64, dropout=0.5, lr=0.001, batch_size=32, num_layers=2,
activation=relu, epochs=15
Training with hidden_dim=64, dropout=0.5, lr=0.001, batch_size=32, num_layers=2,
activation=relu, epochs=30
Training with hidden_dim=64, dropout=0.5, lr=0.001, batch_size=32, num_layers=2,
activation=relu, epochs=35
Training with hidden_dim=64, dropout=0.5, lr=0.001, batch_size=32, num_layers=2,
activation=relu, epochs=40
Training with hidden_dim=64, dropout=0.5, lr=0.001, batch_size=32, num_layers=2,
activation=relu, epochs=50
Training with hidden_dim=64, dropout=0.5, lr=0.001, batch_size=32, num_layers=2,
activation=tanh, epochs=10
Training with hidden_dim=64, dropout=0.5, lr=0.001, batch_size=32, num_layers=2,
activation=tanh, epochs=20
Training with hidden_dim=64, dropout=0.5, lr=0.001, batch_size=32, num_layers=2,
activation=tanh, epochs=25
Training with hidden_dim=64, dropout=0.5, lr=0.001, batch_size=32, num_layers=2,
activation=tanh, epochs=15
Training with hidden_dim=64, dropout=0.5, lr=0.001, batch_size=32, num_layers=2,
activation=tanh, epochs=30
Training with hidden_dim=64, dropout=0.5, lr=0.001, batch_size=32, num_layers=2,
activation=tanh, epochs=35
Training with hidden_dim=64, dropout=0.5, lr=0.001, batch_size=32, num_layers=2,
activation=tanh, epochs=40
Training with hidden_dim=64, dropout=0.5, lr=0.001, batch_size=32, num_layers=2,
activation=tanh, epochs=50
Training with hidden_dim=64, dropout=0.5, lr=0.001, batch_size=32, num_layers=3,
activation=relu, epochs=10
Training with hidden_dim=64, dropout=0.5, lr=0.001, batch_size=32, num_layers=3,
activation=relu, epochs=20
Training with hidden_dim=64, dropout=0.5, lr=0.001, batch_size=32, num_layers=3,
activation=relu, epochs=25
Training with hidden_dim=64, dropout=0.5, lr=0.001, batch_size=32, num_layers=3,
activation=relu, epochs=15
Training with hidden_dim=64, dropout=0.5, lr=0.001, batch_size=32, num_layers=3,
activation=relu, epochs=30
Training with hidden_dim=64, dropout=0.5, lr=0.001, batch_size=32, num_layers=3,
activation=relu, epochs=35
Training with hidden_dim=64, dropout=0.5, lr=0.001, batch_size=32, num_layers=3,
```

```
activation=relu, epochs=40
Training with hidden_dim=64, dropout=0.5, lr=0.001, batch_size=32, num_layers=3,
activation=relu, epochs=50
Training with hidden_dim=64, dropout=0.5, lr=0.001, batch_size=32, num_layers=3,
activation=tanh, epochs=10
Training with hidden_dim=64, dropout=0.5, lr=0.001, batch_size=32, num_layers=3,
activation=tanh, epochs=20
Training with hidden_dim=64, dropout=0.5, lr=0.001, batch_size=32, num_layers=3,
activation=tanh, epochs=25
Training with hidden_dim=64, dropout=0.5, lr=0.001, batch_size=32, num_layers=3,
activation=tanh, epochs=15
Training with hidden_dim=64, dropout=0.5, lr=0.001, batch_size=32, num_layers=3,
activation=tanh, epochs=30
Training with hidden_dim=64, dropout=0.5, lr=0.001, batch_size=32, num_layers=3,
activation=tanh, epochs=35
Training with hidden_dim=64, dropout=0.5, lr=0.001, batch_size=32, num_layers=3,
activation=tanh, epochs=40
Training with hidden_dim=64, dropout=0.5, lr=0.001, batch_size=32, num_layers=3,
activation=tanh, epochs=50
Training with hidden_dim=64, dropout=0.5, lr=0.001, batch_size=64, num_layers=2,
activation=relu, epochs=10
Training with hidden_dim=64, dropout=0.5, lr=0.001, batch_size=64, num_layers=2,
activation=relu, epochs=20
Training with hidden_dim=64, dropout=0.5, lr=0.001, batch_size=64, num_layers=2,
activation=relu, epochs=25
Training with hidden_dim=64, dropout=0.5, lr=0.001, batch_size=64, num_layers=2,
activation=relu, epochs=15
Training with hidden_dim=64, dropout=0.5, lr=0.001, batch_size=64, num_layers=2,
activation=relu, epochs=30
Training with hidden_dim=64, dropout=0.5, lr=0.001, batch_size=64, num_layers=2,
activation=relu, epochs=35
Training with hidden_dim=64, dropout=0.5, lr=0.001, batch_size=64, num_layers=2,
activation=relu, epochs=40
Training with hidden_dim=64, dropout=0.5, lr=0.001, batch_size=64, num_layers=2,
activation=relu, epochs=50
Training with hidden_dim=64, dropout=0.5, lr=0.001, batch_size=64, num_layers=2,
activation=tanh, epochs=10
Training with hidden_dim=64, dropout=0.5, lr=0.001, batch_size=64, num_layers=2,
activation=tanh, epochs=20
Training with hidden_dim=64, dropout=0.5, lr=0.001, batch_size=64, num_layers=2,
activation=tanh, epochs=25
Training with hidden_dim=64, dropout=0.5, lr=0.001, batch_size=64, num_layers=2,
activation=tanh, epochs=15
Training with hidden_dim=64, dropout=0.5, lr=0.001, batch_size=64, num_layers=2,
activation=tanh, epochs=30
Training with hidden_dim=64, dropout=0.5, lr=0.001, batch_size=64, num_layers=2,
activation=tanh, epochs=35
Training with hidden_dim=64, dropout=0.5, lr=0.001, batch_size=64, num_layers=2,
```

```
activation=tanh, epochs=40
Training with hidden_dim=64, dropout=0.5, lr=0.001, batch_size=64, num_layers=2,
activation=tanh, epochs=50
Training with hidden_dim=64, dropout=0.5, lr=0.001, batch_size=64, num_layers=3,
activation=relu, epochs=10
Training with hidden_dim=64, dropout=0.5, lr=0.001, batch_size=64, num_layers=3,
activation=relu, epochs=20
Training with hidden_dim=64, dropout=0.5, lr=0.001, batch_size=64, num_layers=3,
activation=relu, epochs=25
Training with hidden_dim=64, dropout=0.5, lr=0.001, batch_size=64, num_layers=3,
activation=relu, epochs=15
Training with hidden_dim=64, dropout=0.5, lr=0.001, batch_size=64, num_layers=3,
activation=relu, epochs=30
Training with hidden_dim=64, dropout=0.5, lr=0.001, batch_size=64, num_layers=3,
activation=relu, epochs=35
Training with hidden_dim=64, dropout=0.5, lr=0.001, batch_size=64, num_layers=3,
activation=relu, epochs=40
Training with hidden_dim=64, dropout=0.5, lr=0.001, batch_size=64, num_layers=3,
activation=relu, epochs=50
Training with hidden_dim=64, dropout=0.5, lr=0.001, batch_size=64, num_layers=3,
activation=tanh, epochs=10
Training with hidden_dim=64, dropout=0.5, lr=0.001, batch_size=64, num_layers=3,
activation=tanh, epochs=20
Training with hidden_dim=64, dropout=0.5, lr=0.001, batch_size=64, num_layers=3,
activation=tanh, epochs=25
Training with hidden_dim=64, dropout=0.5, lr=0.001, batch_size=64, num_layers=3,
activation=tanh, epochs=15
Training with hidden_dim=64, dropout=0.5, lr=0.001, batch_size=64, num_layers=3,
activation=tanh, epochs=30
Training with hidden_dim=64, dropout=0.5, lr=0.001, batch_size=64, num_layers=3,
activation=tanh, epochs=35
Training with hidden_dim=64, dropout=0.5, lr=0.001, batch_size=64, num_layers=3,
activation=tanh, epochs=40
Training with hidden_dim=64, dropout=0.5, lr=0.001, batch_size=64, num_layers=3,
activation=tanh, epochs=50
Training with hidden_dim=128, dropout=0.3, lr=0.01, batch_size=32, num_layers=2,
activation=relu, epochs=10
Training with hidden_dim=128, dropout=0.3, lr=0.01, batch_size=32, num_layers=2,
activation=relu, epochs=20
Training with hidden_dim=128, dropout=0.3, lr=0.01, batch_size=32, num_layers=2,
activation=relu, epochs=25
Training with hidden_dim=128, dropout=0.3, lr=0.01, batch_size=32, num_layers=2,
activation=relu, epochs=15
Training with hidden_dim=128, dropout=0.3, lr=0.01, batch_size=32, num_layers=2,
activation=relu, epochs=30
Training with hidden_dim=128, dropout=0.3, lr=0.01, batch_size=32, num_layers=2,
activation=relu, epochs=35
Training with hidden_dim=128, dropout=0.3, lr=0.01, batch_size=32, num_layers=2,
```

```
activation=relu, epochs=40
Training with hidden_dim=128, dropout=0.3, lr=0.01, batch_size=32, num_layers=2,
activation=relu, epochs=50
Training with hidden_dim=128, dropout=0.3, lr=0.01, batch_size=32, num_layers=2,
activation=tanh, epochs=10
Training with hidden_dim=128, dropout=0.3, lr=0.01, batch_size=32, num_layers=2,
activation=tanh, epochs=20
Training with hidden_dim=128, dropout=0.3, lr=0.01, batch_size=32, num_layers=2,
activation=tanh, epochs=25
Training with hidden_dim=128, dropout=0.3, lr=0.01, batch_size=32, num_layers=2,
activation=tanh, epochs=15
Training with hidden_dim=128, dropout=0.3, lr=0.01, batch_size=32, num_layers=2,
activation=tanh, epochs=30
Training with hidden_dim=128, dropout=0.3, lr=0.01, batch_size=32, num_layers=2,
activation=tanh, epochs=35
Training with hidden_dim=128, dropout=0.3, lr=0.01, batch_size=32, num_layers=2,
activation=tanh, epochs=40
Training with hidden_dim=128, dropout=0.3, lr=0.01, batch_size=32, num_layers=2,
activation=tanh, epochs=50
Training with hidden_dim=128, dropout=0.3, lr=0.01, batch_size=32, num_layers=3,
activation=relu, epochs=10
Training with hidden_dim=128, dropout=0.3, lr=0.01, batch_size=32, num_layers=3,
activation=relu, epochs=20
Training with hidden_dim=128, dropout=0.3, lr=0.01, batch_size=32, num_layers=3,
activation=relu, epochs=25
Training with hidden_dim=128, dropout=0.3, lr=0.01, batch_size=32, num_layers=3,
activation=relu, epochs=15
Training with hidden_dim=128, dropout=0.3, lr=0.01, batch_size=32, num_layers=3,
activation=relu, epochs=30
Training with hidden_dim=128, dropout=0.3, lr=0.01, batch_size=32, num_layers=3,
activation=relu, epochs=35
Training with hidden_dim=128, dropout=0.3, lr=0.01, batch_size=32, num_layers=3,
activation=relu, epochs=40
Training with hidden_dim=128, dropout=0.3, lr=0.01, batch_size=32, num_layers=3,
activation=relu, epochs=50
Training with hidden_dim=128, dropout=0.3, lr=0.01, batch_size=32, num_layers=3,
activation=tanh, epochs=10
Training with hidden_dim=128, dropout=0.3, lr=0.01, batch_size=32, num_layers=3,
activation=tanh, epochs=20
Training with hidden_dim=128, dropout=0.3, lr=0.01, batch_size=32, num_layers=3,
activation=tanh, epochs=25
Training with hidden_dim=128, dropout=0.3, lr=0.01, batch_size=32, num_layers=3,
activation=tanh, epochs=15
Training with hidden_dim=128, dropout=0.3, lr=0.01, batch_size=32, num_layers=3,
activation=tanh, epochs=30
Training with hidden_dim=128, dropout=0.3, lr=0.01, batch_size=32, num_layers=3,
activation=tanh, epochs=35
Training with hidden_dim=128, dropout=0.3, lr=0.01, batch_size=32, num_layers=3,
```

```
activation=tanh, epochs=40
Training with hidden_dim=128, dropout=0.3, lr=0.01, batch_size=32, num_layers=3,
activation=tanh, epochs=50
Training with hidden_dim=128, dropout=0.3, lr=0.01, batch_size=64, num_layers=2,
activation=relu, epochs=10
Training with hidden_dim=128, dropout=0.3, lr=0.01, batch_size=64, num_layers=2,
activation=relu, epochs=20
Training with hidden_dim=128, dropout=0.3, lr=0.01, batch_size=64, num_layers=2,
activation=relu, epochs=25
Training with hidden_dim=128, dropout=0.3, lr=0.01, batch_size=64, num_layers=2,
activation=relu, epochs=15
Training with hidden_dim=128, dropout=0.3, lr=0.01, batch_size=64, num_layers=2,
activation=relu, epochs=30
Training with hidden_dim=128, dropout=0.3, lr=0.01, batch_size=64, num_layers=2,
activation=relu, epochs=35
Training with hidden_dim=128, dropout=0.3, lr=0.01, batch_size=64, num_layers=2,
activation=relu, epochs=40
Training with hidden_dim=128, dropout=0.3, lr=0.01, batch_size=64, num_layers=2,
activation=relu, epochs=50
Training with hidden_dim=128, dropout=0.3, lr=0.01, batch_size=64, num_layers=2,
activation=tanh, epochs=10
Training with hidden_dim=128, dropout=0.3, lr=0.01, batch_size=64, num_layers=2,
activation=tanh, epochs=20
Training with hidden_dim=128, dropout=0.3, lr=0.01, batch_size=64, num_layers=2,
activation=tanh, epochs=25
Training with hidden_dim=128, dropout=0.3, lr=0.01, batch_size=64, num_layers=2,
activation=tanh, epochs=15
Training with hidden_dim=128, dropout=0.3, lr=0.01, batch_size=64, num_layers=2,
activation=tanh, epochs=30
Training with hidden_dim=128, dropout=0.3, lr=0.01, batch_size=64, num_layers=2,
activation=tanh, epochs=35
Training with hidden_dim=128, dropout=0.3, lr=0.01, batch_size=64, num_layers=2,
activation=tanh, epochs=40
Training with hidden_dim=128, dropout=0.3, lr=0.01, batch_size=64, num_layers=2,
activation=tanh, epochs=50
Training with hidden_dim=128, dropout=0.3, lr=0.01, batch_size=64, num_layers=3,
activation=relu, epochs=10
Training with hidden_dim=128, dropout=0.3, lr=0.01, batch_size=64, num_layers=3,
activation=relu, epochs=20
Training with hidden_dim=128, dropout=0.3, lr=0.01, batch_size=64, num_layers=3,
activation=relu, epochs=25
Training with hidden_dim=128, dropout=0.3, lr=0.01, batch_size=64, num_layers=3,
activation=relu, epochs=15
Training with hidden_dim=128, dropout=0.3, lr=0.01, batch_size=64, num_layers=3,
activation=relu, epochs=30
Training with hidden_dim=128, dropout=0.3, lr=0.01, batch_size=64, num_layers=3,
activation=relu, epochs=35
Training with hidden_dim=128, dropout=0.3, lr=0.01, batch_size=64, num_layers=3,
```

```
activation=relu, epochs=40
Training with hidden_dim=128, dropout=0.3, lr=0.01, batch_size=64, num_layers=3,
activation=relu, epochs=50
Training with hidden_dim=128, dropout=0.3, lr=0.01, batch_size=64, num_layers=3,
activation=tanh, epochs=10
Training with hidden_dim=128, dropout=0.3, lr=0.01, batch_size=64, num_layers=3,
activation=tanh, epochs=20
Training with hidden_dim=128, dropout=0.3, lr=0.01, batch_size=64, num_layers=3,
activation=tanh, epochs=25
Training with hidden_dim=128, dropout=0.3, lr=0.01, batch_size=64, num_layers=3,
activation=tanh, epochs=15
Training with hidden_dim=128, dropout=0.3, lr=0.01, batch_size=64, num_layers=3,
activation=tanh, epochs=30
Training with hidden_dim=128, dropout=0.3, lr=0.01, batch_size=64, num_layers=3,
activation=tanh, epochs=35
Training with hidden_dim=128, dropout=0.3, lr=0.01, batch_size=64, num_layers=3,
activation=tanh, epochs=40
Training with hidden_dim=128, dropout=0.3, lr=0.01, batch_size=64, num_layers=3,
activation=tanh, epochs=50
Training with hidden_dim=128, dropout=0.3, lr=0.001, batch_size=32,
num_layers=2, activation=relu, epochs=10
Training with hidden_dim=128, dropout=0.3, lr=0.001, batch_size=32,
num_layers=2, activation=relu, epochs=20
Training with hidden_dim=128, dropout=0.3, lr=0.001, batch_size=32,
num_layers=2, activation=relu, epochs=25
Training with hidden_dim=128, dropout=0.3, lr=0.001, batch_size=32,
num_layers=2, activation=relu, epochs=15
Training with hidden_dim=128, dropout=0.3, lr=0.001, batch_size=32,
num_layers=2, activation=relu, epochs=30
Training with hidden_dim=128, dropout=0.3, lr=0.001, batch_size=32,
num_layers=2, activation=relu, epochs=35
Training with hidden_dim=128, dropout=0.3, lr=0.001, batch_size=32,
num_layers=2, activation=relu, epochs=40
Training with hidden_dim=128, dropout=0.3, lr=0.001, batch_size=32,
num_layers=2, activation=relu, epochs=50
Training with hidden_dim=128, dropout=0.3, lr=0.001, batch_size=32,
num_layers=2, activation=tanh, epochs=10
Training with hidden_dim=128, dropout=0.3, lr=0.001, batch_size=32,
num_layers=2, activation=tanh, epochs=20
Training with hidden_dim=128, dropout=0.3, lr=0.001, batch_size=32,
num_layers=2, activation=tanh, epochs=25
Training with hidden_dim=128, dropout=0.3, lr=0.001, batch_size=32,
num_layers=2, activation=tanh, epochs=15
Training with hidden_dim=128, dropout=0.3, lr=0.001, batch_size=32,
num_layers=2, activation=tanh, epochs=30
Training with hidden_dim=128, dropout=0.3, lr=0.001, batch_size=32,
num_layers=2, activation=tanh, epochs=35
Training with hidden_dim=128, dropout=0.3, lr=0.001, batch_size=32,
```

```
num_layers=2, activation=tanh, epochs=40
Training with hidden_dim=128, dropout=0.3, lr=0.001, batch_size=32,
num_layers=2, activation=tanh, epochs=50
Training with hidden_dim=128, dropout=0.3, lr=0.001, batch_size=32,
num_layers=3, activation=relu, epochs=10
Training with hidden_dim=128, dropout=0.3, lr=0.001, batch_size=32,
num_layers=3, activation=relu, epochs=20
Training with hidden_dim=128, dropout=0.3, lr=0.001, batch_size=32,
num_layers=3, activation=relu, epochs=25
Training with hidden_dim=128, dropout=0.3, lr=0.001, batch_size=32,
num_layers=3, activation=relu, epochs=15
Training with hidden_dim=128, dropout=0.3, lr=0.001, batch_size=32,
num_layers=3, activation=relu, epochs=30
Training with hidden_dim=128, dropout=0.3, lr=0.001, batch_size=32,
num_layers=3, activation=relu, epochs=35
Training with hidden_dim=128, dropout=0.3, lr=0.001, batch_size=32,
num_layers=3, activation=relu, epochs=40
Training with hidden_dim=128, dropout=0.3, lr=0.001, batch_size=32,
num_layers=3, activation=relu, epochs=50
Training with hidden_dim=128, dropout=0.3, lr=0.001, batch_size=32,
num_layers=3, activation=tanh, epochs=10
Training with hidden_dim=128, dropout=0.3, lr=0.001, batch_size=32,
num_layers=3, activation=tanh, epochs=20
Training with hidden_dim=128, dropout=0.3, lr=0.001, batch_size=32,
num_layers=3, activation=tanh, epochs=25
Training with hidden_dim=128, dropout=0.3, lr=0.001, batch_size=32,
num_layers=3, activation=tanh, epochs=15
Training with hidden_dim=128, dropout=0.3, lr=0.001, batch_size=32,
num_layers=3, activation=tanh, epochs=30
Training with hidden_dim=128, dropout=0.3, lr=0.001, batch_size=32,
num_layers=3, activation=tanh, epochs=35
Training with hidden_dim=128, dropout=0.3, lr=0.001, batch_size=32,
num_layers=3, activation=tanh, epochs=40
Training with hidden_dim=128, dropout=0.3, lr=0.001, batch_size=32,
num_layers=3, activation=tanh, epochs=50
Training with hidden_dim=128, dropout=0.3, lr=0.001, batch_size=64,
num_layers=2, activation=relu, epochs=10
Training with hidden_dim=128, dropout=0.3, lr=0.001, batch_size=64,
num_layers=2, activation=relu, epochs=20
Training with hidden_dim=128, dropout=0.3, lr=0.001, batch_size=64,
num_layers=2, activation=relu, epochs=25
Training with hidden_dim=128, dropout=0.3, lr=0.001, batch_size=64,
num_layers=2, activation=relu, epochs=15
Training with hidden_dim=128, dropout=0.3, lr=0.001, batch_size=64,
num_layers=2, activation=relu, epochs=30
Training with hidden_dim=128, dropout=0.3, lr=0.001, batch_size=64,
num_layers=2, activation=relu, epochs=35
Training with hidden_dim=128, dropout=0.3, lr=0.001, batch_size=64,
```

```
num_layers=2, activation=relu, epochs=40
Training with hidden_dim=128, dropout=0.3, lr=0.001, batch_size=64,
num_layers=2, activation=relu, epochs=50
Training with hidden_dim=128, dropout=0.3, lr=0.001, batch_size=64,
num_layers=2, activation=tanh, epochs=10
Training with hidden_dim=128, dropout=0.3, lr=0.001, batch_size=64,
num_layers=2, activation=tanh, epochs=20
Training with hidden_dim=128, dropout=0.3, lr=0.001, batch_size=64,
num_layers=2, activation=tanh, epochs=25
Training with hidden_dim=128, dropout=0.3, lr=0.001, batch_size=64,
num_layers=2, activation=tanh, epochs=15
Training with hidden_dim=128, dropout=0.3, lr=0.001, batch_size=64,
num_layers=2, activation=tanh, epochs=30
Training with hidden_dim=128, dropout=0.3, lr=0.001, batch_size=64,
num_layers=2, activation=tanh, epochs=35
Training with hidden_dim=128, dropout=0.3, lr=0.001, batch_size=64,
num_layers=2, activation=tanh, epochs=40
Training with hidden_dim=128, dropout=0.3, lr=0.001, batch_size=64,
num_layers=2, activation=tanh, epochs=50
Training with hidden_dim=128, dropout=0.3, lr=0.001, batch_size=64,
num_layers=3, activation=relu, epochs=10
Training with hidden_dim=128, dropout=0.3, lr=0.001, batch_size=64,
num_layers=3, activation=relu, epochs=20
Training with hidden_dim=128, dropout=0.3, lr=0.001, batch_size=64,
num_layers=3, activation=relu, epochs=25
Training with hidden_dim=128, dropout=0.3, lr=0.001, batch_size=64,
num_layers=3, activation=relu, epochs=15
Training with hidden_dim=128, dropout=0.3, lr=0.001, batch_size=64,
num_layers=3, activation=relu, epochs=30
Training with hidden_dim=128, dropout=0.3, lr=0.001, batch_size=64,
num_layers=3, activation=relu, epochs=35
Training with hidden_dim=128, dropout=0.3, lr=0.001, batch_size=64,
num_layers=3, activation=relu, epochs=40
Training with hidden_dim=128, dropout=0.3, lr=0.001, batch_size=64,
num_layers=3, activation=relu, epochs=50
Training with hidden_dim=128, dropout=0.3, lr=0.001, batch_size=64,
num_layers=3, activation=tanh, epochs=10
Training with hidden_dim=128, dropout=0.3, lr=0.001, batch_size=64,
num_layers=3, activation=tanh, epochs=20
Training with hidden_dim=128, dropout=0.3, lr=0.001, batch_size=64,
num_layers=3, activation=tanh, epochs=25
Training with hidden_dim=128, dropout=0.3, lr=0.001, batch_size=64,
num_layers=3, activation=tanh, epochs=15
Training with hidden_dim=128, dropout=0.3, lr=0.001, batch_size=64,
num_layers=3, activation=tanh, epochs=30
Training with hidden_dim=128, dropout=0.3, lr=0.001, batch_size=64,
num_layers=3, activation=tanh, epochs=35
Training with hidden_dim=128, dropout=0.3, lr=0.001, batch_size=64,
```

```
num_layers=3, activation=tanh, epochs=40
Training with hidden_dim=128, dropout=0.3, lr=0.001, batch_size=64,
num_layers=3, activation=tanh, epochs=50
Training with hidden_dim=128, dropout=0.5, lr=0.01, batch_size=32, num_layers=2,
activation=relu, epochs=10
Training with hidden_dim=128, dropout=0.5, lr=0.01, batch_size=32, num_layers=2,
activation=relu, epochs=20
Training with hidden_dim=128, dropout=0.5, lr=0.01, batch_size=32, num_layers=2,
activation=relu, epochs=25
Training with hidden_dim=128, dropout=0.5, lr=0.01, batch_size=32, num_layers=2,
activation=relu, epochs=15
Training with hidden_dim=128, dropout=0.5, lr=0.01, batch_size=32, num_layers=2,
activation=relu, epochs=30
Training with hidden_dim=128, dropout=0.5, lr=0.01, batch_size=32, num_layers=2,
activation=relu, epochs=35
Training with hidden_dim=128, dropout=0.5, lr=0.01, batch_size=32, num_layers=2,
activation=relu, epochs=40
Training with hidden_dim=128, dropout=0.5, lr=0.01, batch_size=32, num_layers=2,
activation=relu, epochs=50
Training with hidden_dim=128, dropout=0.5, lr=0.01, batch_size=32, num_layers=2,
activation=tanh, epochs=10
Training with hidden_dim=128, dropout=0.5, lr=0.01, batch_size=32, num_layers=2,
activation=tanh, epochs=20
Training with hidden_dim=128, dropout=0.5, lr=0.01, batch_size=32, num_layers=2,
activation=tanh, epochs=25
Training with hidden_dim=128, dropout=0.5, lr=0.01, batch_size=32, num_layers=2,
activation=tanh, epochs=15
Training with hidden_dim=128, dropout=0.5, lr=0.01, batch_size=32, num_layers=2,
activation=tanh, epochs=30
Training with hidden_dim=128, dropout=0.5, lr=0.01, batch_size=32, num_layers=2,
activation=tanh, epochs=35
Training with hidden_dim=128, dropout=0.5, lr=0.01, batch_size=32, num_layers=2,
activation=tanh, epochs=40
Training with hidden_dim=128, dropout=0.5, lr=0.01, batch_size=32, num_layers=2,
activation=tanh, epochs=50
Training with hidden_dim=128, dropout=0.5, lr=0.01, batch_size=32, num_layers=3,
activation=relu, epochs=10
Training with hidden_dim=128, dropout=0.5, lr=0.01, batch_size=32, num_layers=3,
activation=relu, epochs=20
Training with hidden_dim=128, dropout=0.5, lr=0.01, batch_size=32, num_layers=3,
activation=relu, epochs=25
Training with hidden_dim=128, dropout=0.5, lr=0.01, batch_size=32, num_layers=3,
activation=relu, epochs=15
Training with hidden_dim=128, dropout=0.5, lr=0.01, batch_size=32, num_layers=3,
activation=relu, epochs=30
Training with hidden_dim=128, dropout=0.5, lr=0.01, batch_size=32, num_layers=3,
activation=relu, epochs=35
Training with hidden_dim=128, dropout=0.5, lr=0.01, batch_size=32, num_layers=3,
```

```
activation=relu, epochs=40
Training with hidden_dim=128, dropout=0.5, lr=0.01, batch_size=32, num_layers=3,
activation=relu, epochs=50
Training with hidden_dim=128, dropout=0.5, lr=0.01, batch_size=32, num_layers=3,
activation=tanh, epochs=10
Training with hidden_dim=128, dropout=0.5, lr=0.01, batch_size=32, num_layers=3,
activation=tanh, epochs=20
Training with hidden_dim=128, dropout=0.5, lr=0.01, batch_size=32, num_layers=3,
activation=tanh, epochs=25
Training with hidden_dim=128, dropout=0.5, lr=0.01, batch_size=32, num_layers=3,
activation=tanh, epochs=15
Training with hidden_dim=128, dropout=0.5, lr=0.01, batch_size=32, num_layers=3,
activation=tanh, epochs=30
Training with hidden_dim=128, dropout=0.5, lr=0.01, batch_size=32, num_layers=3,
activation=tanh, epochs=35
Training with hidden_dim=128, dropout=0.5, lr=0.01, batch_size=32, num_layers=3,
activation=tanh, epochs=40
Training with hidden_dim=128, dropout=0.5, lr=0.01, batch_size=32, num_layers=3,
activation=tanh, epochs=50
Training with hidden_dim=128, dropout=0.5, lr=0.01, batch_size=64, num_layers=2,
activation=relu, epochs=10
Training with hidden_dim=128, dropout=0.5, lr=0.01, batch_size=64, num_layers=2,
activation=relu, epochs=20
Training with hidden_dim=128, dropout=0.5, lr=0.01, batch_size=64, num_layers=2,
activation=relu, epochs=25
Training with hidden_dim=128, dropout=0.5, lr=0.01, batch_size=64, num_layers=2,
activation=relu, epochs=15
Training with hidden_dim=128, dropout=0.5, lr=0.01, batch_size=64, num_layers=2,
activation=relu, epochs=30
Training with hidden_dim=128, dropout=0.5, lr=0.01, batch_size=64, num_layers=2,
activation=relu, epochs=35
Training with hidden_dim=128, dropout=0.5, lr=0.01, batch_size=64, num_layers=2,
activation=relu, epochs=40
Training with hidden_dim=128, dropout=0.5, lr=0.01, batch_size=64, num_layers=2,
activation=relu, epochs=50
Training with hidden_dim=128, dropout=0.5, lr=0.01, batch_size=64, num_layers=2,
activation=tanh, epochs=10
Training with hidden_dim=128, dropout=0.5, lr=0.01, batch_size=64, num_layers=2,
activation=tanh, epochs=20
Training with hidden_dim=128, dropout=0.5, lr=0.01, batch_size=64, num_layers=2,
activation=tanh, epochs=25
Training with hidden_dim=128, dropout=0.5, lr=0.01, batch_size=64, num_layers=2,
activation=tanh, epochs=15
Training with hidden_dim=128, dropout=0.5, lr=0.01, batch_size=64, num_layers=2,
activation=tanh, epochs=30
Training with hidden_dim=128, dropout=0.5, lr=0.01, batch_size=64, num_layers=2,
activation=tanh, epochs=35
Training with hidden_dim=128, dropout=0.5, lr=0.01, batch_size=64, num_layers=2,
```

```
activation=tanh, epochs=40
Training with hidden_dim=128, dropout=0.5, lr=0.01, batch_size=64, num_layers=2,
activation=tanh, epochs=50
Training with hidden_dim=128, dropout=0.5, lr=0.01, batch_size=64, num_layers=3,
activation=relu, epochs=10
Training with hidden_dim=128, dropout=0.5, lr=0.01, batch_size=64, num_layers=3,
activation=relu, epochs=20
Training with hidden_dim=128, dropout=0.5, lr=0.01, batch_size=64, num_layers=3,
activation=relu, epochs=25
Training with hidden_dim=128, dropout=0.5, lr=0.01, batch_size=64, num_layers=3,
activation=relu, epochs=15
Training with hidden_dim=128, dropout=0.5, lr=0.01, batch_size=64, num_layers=3,
activation=relu, epochs=30
Training with hidden_dim=128, dropout=0.5, lr=0.01, batch_size=64, num_layers=3,
activation=relu, epochs=35
Training with hidden_dim=128, dropout=0.5, lr=0.01, batch_size=64, num_layers=3,
activation=relu, epochs=40
Training with hidden_dim=128, dropout=0.5, lr=0.01, batch_size=64, num_layers=3,
activation=relu, epochs=50
Training with hidden_dim=128, dropout=0.5, lr=0.01, batch_size=64, num_layers=3,
activation=tanh, epochs=10
Training with hidden_dim=128, dropout=0.5, lr=0.01, batch_size=64, num_layers=3,
activation=tanh, epochs=20
Training with hidden_dim=128, dropout=0.5, lr=0.01, batch_size=64, num_layers=3,
activation=tanh, epochs=25
Training with hidden_dim=128, dropout=0.5, lr=0.01, batch_size=64, num_layers=3,
activation=tanh, epochs=15
Training with hidden_dim=128, dropout=0.5, lr=0.01, batch_size=64, num_layers=3,
activation=tanh, epochs=30
Training with hidden_dim=128, dropout=0.5, lr=0.01, batch_size=64, num_layers=3,
activation=tanh, epochs=35
Training with hidden_dim=128, dropout=0.5, lr=0.01, batch_size=64, num_layers=3,
activation=tanh, epochs=40
Training with hidden_dim=128, dropout=0.5, lr=0.01, batch_size=64, num_layers=3,
activation=tanh, epochs=50
Training with hidden_dim=128, dropout=0.5, lr=0.001, batch_size=32,
num_layers=2, activation=relu, epochs=10
Training with hidden_dim=128, dropout=0.5, lr=0.001, batch_size=32,
num_layers=2, activation=relu, epochs=20
Training with hidden_dim=128, dropout=0.5, lr=0.001, batch_size=32,
num_layers=2, activation=relu, epochs=25
Training with hidden_dim=128, dropout=0.5, lr=0.001, batch_size=32,
num_layers=2, activation=relu, epochs=15
Training with hidden_dim=128, dropout=0.5, lr=0.001, batch_size=32,
num_layers=2, activation=relu, epochs=30
Training with hidden_dim=128, dropout=0.5, lr=0.001, batch_size=32,
num_layers=2, activation=relu, epochs=35
Training with hidden_dim=128, dropout=0.5, lr=0.001, batch_size=32,
```

```
num_layers=2, activation=relu, epochs=40
Training with hidden_dim=128, dropout=0.5, lr=0.001, batch_size=32,
num_layers=2, activation=relu, epochs=50
Training with hidden_dim=128, dropout=0.5, lr=0.001, batch_size=32,
num_layers=2, activation=tanh, epochs=10
Training with hidden_dim=128, dropout=0.5, lr=0.001, batch_size=32,
num_layers=2, activation=tanh, epochs=20
Training with hidden_dim=128, dropout=0.5, lr=0.001, batch_size=32,
num_layers=2, activation=tanh, epochs=25
Training with hidden_dim=128, dropout=0.5, lr=0.001, batch_size=32,
num_layers=2, activation=tanh, epochs=15
Training with hidden_dim=128, dropout=0.5, lr=0.001, batch_size=32,
num_layers=2, activation=tanh, epochs=30
Training with hidden_dim=128, dropout=0.5, lr=0.001, batch_size=32,
num_layers=2, activation=tanh, epochs=35
Training with hidden_dim=128, dropout=0.5, lr=0.001, batch_size=32,
num_layers=2, activation=tanh, epochs=40
Training with hidden_dim=128, dropout=0.5, lr=0.001, batch_size=32,
num_layers=2, activation=tanh, epochs=50
Training with hidden_dim=128, dropout=0.5, lr=0.001, batch_size=32,
num_layers=3, activation=relu, epochs=10
Training with hidden_dim=128, dropout=0.5, lr=0.001, batch_size=32,
num_layers=3, activation=relu, epochs=20
Training with hidden_dim=128, dropout=0.5, lr=0.001, batch_size=32,
num_layers=3, activation=relu, epochs=25
Training with hidden_dim=128, dropout=0.5, lr=0.001, batch_size=32,
num_layers=3, activation=relu, epochs=15
Training with hidden_dim=128, dropout=0.5, lr=0.001, batch_size=32,
num_layers=3, activation=relu, epochs=30
Training with hidden_dim=128, dropout=0.5, lr=0.001, batch_size=32,
num_layers=3, activation=relu, epochs=35
Training with hidden_dim=128, dropout=0.5, lr=0.001, batch_size=32,
num_layers=3, activation=relu, epochs=40
Training with hidden_dim=128, dropout=0.5, lr=0.001, batch_size=32,
num_layers=3, activation=relu, epochs=50
Training with hidden_dim=128, dropout=0.5, lr=0.001, batch_size=32,
num_layers=3, activation=tanh, epochs=10
Training with hidden_dim=128, dropout=0.5, lr=0.001, batch_size=32,
num_layers=3, activation=tanh, epochs=20
Training with hidden_dim=128, dropout=0.5, lr=0.001, batch_size=32,
num_layers=3, activation=tanh, epochs=25
Training with hidden_dim=128, dropout=0.5, lr=0.001, batch_size=32,
num_layers=3, activation=tanh, epochs=15
Training with hidden_dim=128, dropout=0.5, lr=0.001, batch_size=32,
num_layers=3, activation=tanh, epochs=30
Training with hidden_dim=128, dropout=0.5, lr=0.001, batch_size=32,
num_layers=3, activation=tanh, epochs=35
Training with hidden_dim=128, dropout=0.5, lr=0.001, batch_size=32,
```

```
num_layers=3, activation=tanh, epochs=40
Training with hidden_dim=128, dropout=0.5, lr=0.001, batch_size=32,
num_layers=3, activation=tanh, epochs=50
Training with hidden_dim=128, dropout=0.5, lr=0.001, batch_size=64,
num_layers=2, activation=relu, epochs=10
Training with hidden_dim=128, dropout=0.5, lr=0.001, batch_size=64,
num_layers=2, activation=relu, epochs=20
Training with hidden_dim=128, dropout=0.5, lr=0.001, batch_size=64,
num_layers=2, activation=relu, epochs=25
Training with hidden_dim=128, dropout=0.5, lr=0.001, batch_size=64,
num_layers=2, activation=relu, epochs=15
Training with hidden_dim=128, dropout=0.5, lr=0.001, batch_size=64,
num_layers=2, activation=relu, epochs=30
Training with hidden_dim=128, dropout=0.5, lr=0.001, batch_size=64,
num_layers=2, activation=relu, epochs=35
Training with hidden_dim=128, dropout=0.5, lr=0.001, batch_size=64,
num_layers=2, activation=relu, epochs=40
Training with hidden_dim=128, dropout=0.5, lr=0.001, batch_size=64,
num_layers=2, activation=relu, epochs=50
Training with hidden_dim=128, dropout=0.5, lr=0.001, batch_size=64,
num_layers=2, activation=tanh, epochs=10
Training with hidden_dim=128, dropout=0.5, lr=0.001, batch_size=64,
num_layers=2, activation=tanh, epochs=20
Training with hidden_dim=128, dropout=0.5, lr=0.001, batch_size=64,
num_layers=2, activation=tanh, epochs=25
Training with hidden_dim=128, dropout=0.5, lr=0.001, batch_size=64,
num_layers=2, activation=tanh, epochs=15
Training with hidden_dim=128, dropout=0.5, lr=0.001, batch_size=64,
num_layers=2, activation=tanh, epochs=30
Training with hidden_dim=128, dropout=0.5, lr=0.001, batch_size=64,
num_layers=2, activation=tanh, epochs=35
Training with hidden_dim=128, dropout=0.5, lr=0.001, batch_size=64,
num_layers=2, activation=tanh, epochs=40
Training with hidden_dim=128, dropout=0.5, lr=0.001, batch_size=64,
num_layers=2, activation=tanh, epochs=50
Training with hidden_dim=128, dropout=0.5, lr=0.001, batch_size=64,
num_layers=3, activation=relu, epochs=10
Training with hidden_dim=128, dropout=0.5, lr=0.001, batch_size=64,
num_layers=3, activation=relu, epochs=20
Training with hidden_dim=128, dropout=0.5, lr=0.001, batch_size=64,
num_layers=3, activation=relu, epochs=25
Training with hidden_dim=128, dropout=0.5, lr=0.001, batch_size=64,
num_layers=3, activation=relu, epochs=15
Training with hidden_dim=128, dropout=0.5, lr=0.001, batch_size=64,
num_layers=3, activation=relu, epochs=30
Training with hidden_dim=128, dropout=0.5, lr=0.001, batch_size=64,
num_layers=3, activation=relu, epochs=35
Training with hidden_dim=128, dropout=0.5, lr=0.001, batch_size=64,
```

```
num_layers=3, activation=relu, epochs=40
Training with hidden_dim=128, dropout=0.5, lr=0.001, batch_size=64,
num_layers=3, activation=relu, epochs=50
Training with hidden_dim=128, dropout=0.5, lr=0.001, batch_size=64,
num_layers=3, activation=tanh, epochs=10
Training with hidden_dim=128, dropout=0.5, lr=0.001, batch_size=64,
num_layers=3, activation=tanh, epochs=20
Training with hidden_dim=128, dropout=0.5, lr=0.001, batch_size=64,
num_layers=3, activation=tanh, epochs=25
Training with hidden_dim=128, dropout=0.5, lr=0.001, batch_size=64,
num_layers=3, activation=tanh, epochs=15
Training with hidden_dim=128, dropout=0.5, lr=0.001, batch_size=64,
num_layers=3, activation=tanh, epochs=30
Training with hidden_dim=128, dropout=0.5, lr=0.001, batch_size=64,
num_layers=3, activation=tanh, epochs=35
Training with hidden_dim=128, dropout=0.5, lr=0.001, batch_size=64,
num_layers=3, activation=tanh, epochs=40
Training with hidden_dim=128, dropout=0.5, lr=0.001, batch_size=64,
num_layers=3, activation=tanh, epochs=50
Best Params: {'hidden_dim': 128, 'dropout': 0.5, 'lr': 0.001, 'batch_size': 64,
'num_layers': 3, 'activation': 'tanh'}
Best Binary Accuracy: 0.9115
Best Normal Accuracy: 0.2884
Best F1 Score: 0.9537
```

[ ]:
```python
# 2. CNN

from tensorflow.keras.models import Sequential
from tensorflow.keras.layers import Conv1D, MaxPooling1D, Dense, Flatten,␣
↪Dropout
from sklearn.metrics import f1_score, accuracy_score
import numpy as np

# Define the CNN model
def build_cnn_model(filters=64, kernel_size=3, dropout=0.5):
    model = Sequential()

    # 1D Convolutional Layer
    model.add(Conv1D(filters=filters, kernel_size=kernel_size,␣
↪activation='relu', input_shape=(X_train.shape[1], 1)))

    # MaxPooling Layer
    model.add(MaxPooling1D(pool_size=2))

    # Dropout Layer
    model.add(Dropout(dropout))
```

```
    # Flatten the output for dense layers
    model.add(Flatten())

    # Dense Layers
    model.add(Dense(256, activation='relu'))
    model.add(Dropout(dropout))

    # Output layer for multi-label classification
    model.add(Dense(y_train.shape[1], activation='sigmoid'))

    # Compile the model
    model.compile(optimizer='adam', loss='binary_crossentropy',␣
↪metrics=['binary_accuracy'])

    return model

# Reshape your data for CNN input
X_train = X_train.reshape((X_train.shape[0], X_train.shape[1], 1))
X_test = X_test.reshape((X_test.shape[0], X_test.shape[1], 1))

# Hyperparameters grid
param_grid = {
    'filters': [32, 64],  # Number of filters for Conv1D
    'kernel_size': [3, 5],  # Kernel size for Conv1D
    'dropout': [0.3, 0.5],  # Dropout rate
    'epochs': [10],  # Number of epochs
    'batch_size': [32],  # Batch size
}

# Variables to track the best model
best_binary_acc = 0
best_params = {}
best_f1 = 0
best_normal_acc = 0

# Perform the manual hyperparameter search
for filters in param_grid['filters']:
    for kernel_size in param_grid['kernel_size']:  # Now this will work
        for dropout in param_grid['dropout']:
            for epochs in param_grid['epochs']:
                for batch_size in param_grid['batch_size']:
                    print(f'Training with filters={filters},␣
↪kernel_size={kernel_size}, dropout={dropout}, epochs={epochs},␣
↪batch_size={batch_size}')

                    # Build the model
```

```
                    model = build_cnn_model(filters=filters,␣
↪kernel_size=kernel_size, dropout=dropout)

                    # Train the model
                    model.fit(X_train, y_train, epochs=epochs,␣
↪batch_size=batch_size, validation_split=0.1, verbose=1)

                    # Predict on the test set
                    y_pred = model.predict(X_test)
                    y_pred_binary = (y_pred > 0.5).astype(int)

                    # Calculate binary accuracy
                    binary_acc = np.mean(np.equal(y_test, y_pred_binary).
↪astype(int))

                    # Calculate normal accuracy
                    normal_acc = accuracy_score(y_test, y_pred_binary)

                    # Calculate F1 Score
                    f1 = f1_score(y_test, y_pred_binary, average='micro')

                    # Update best params if this model is better
                    if binary_acc > best_binary_acc:
                        best_binary_acc = binary_acc
                        best_f1 = f1
                        best_normal_acc = normal_acc
                        best_params = {
                            'filters': filters,
                            'kernel_size': kernel_size,
                            'dropout': dropout,
                            'epochs': epochs,
                            'batch_size': batch_size,
                        }

# Print the best results
print(f'Best Params: {best_params}')
print(f'Best Binary Accuracy: {best_binary_acc:.4f}')
print(f'Best Normal Accuracy: {best_normal_acc:.4f}')
print(f'Best F1 Score: {best_f1:.4f}')
```

```
Training with filters=32, kernel_size=3, dropout=0.3, epochs=10, batch_size=32
Epoch 1/10
30/30 [==============================] - 13s 42ms/step - loss: 0.3576 -
binary_accuracy: 0.8693 - val_loss: 0.2657 - val_binary_accuracy: 0.9124
Epoch 2/10
30/30 [==============================] - 0s 11ms/step - loss: 0.2807 -
binary_accuracy: 0.8937 - val_loss: 0.2467 - val_binary_accuracy: 0.9116
```

```
Epoch 3/10
30/30 [==============================] - 0s 14ms/step - loss: 0.2522 -
binary_accuracy: 0.9002 - val_loss: 0.2450 - val_binary_accuracy: 0.9143
Epoch 4/10
30/30 [==============================] - 0s 15ms/step - loss: 0.2301 -
binary_accuracy: 0.9054 - val_loss: 0.2431 - val_binary_accuracy: 0.9112
Epoch 5/10
30/30 [==============================] - 0s 10ms/step - loss: 0.2119 -
binary_accuracy: 0.9106 - val_loss: 0.2436 - val_binary_accuracy: 0.9104
Epoch 6/10
30/30 [==============================] - 0s 13ms/step - loss: 0.1964 -
binary_accuracy: 0.9168 - val_loss: 0.2573 - val_binary_accuracy: 0.9100
Epoch 7/10
30/30 [==============================] - 0s 10ms/step - loss: 0.1817 -
binary_accuracy: 0.9231 - val_loss: 0.2642 - val_binary_accuracy: 0.8945
Epoch 8/10
30/30 [==============================] - 0s 7ms/step - loss: 0.1689 -
binary_accuracy: 0.9287 - val_loss: 0.2754 - val_binary_accuracy: 0.8917
Epoch 9/10
30/30 [==============================] - 0s 6ms/step - loss: 0.1568 -
binary_accuracy: 0.9327 - val_loss: 0.2829 - val_binary_accuracy: 0.8921
Epoch 10/10
30/30 [==============================] - 0s 7ms/step - loss: 0.1433 -
binary_accuracy: 0.9407 - val_loss: 0.2987 - val_binary_accuracy: 0.8906
9/9 [==============================] - 0s 5ms/step
Training with filters=32, kernel_size=3, dropout=0.5, epochs=10, batch_size=32
Epoch 1/10
30/30 [==============================] - 2s 12ms/step - loss: 0.3652 -
binary_accuracy: 0.8637 - val_loss: 0.2602 - val_binary_accuracy: 0.9124
Epoch 2/10
30/30 [==============================] - 0s 6ms/step - loss: 0.2914 -
binary_accuracy: 0.8888 - val_loss: 0.2499 - val_binary_accuracy: 0.9120
Epoch 3/10
30/30 [==============================] - 0s 7ms/step - loss: 0.2649 -
binary_accuracy: 0.8967 - val_loss: 0.2412 - val_binary_accuracy: 0.9120
Epoch 4/10
30/30 [==============================] - 0s 7ms/step - loss: 0.2453 -
binary_accuracy: 0.9011 - val_loss: 0.2407 - val_binary_accuracy: 0.9147
Epoch 5/10
30/30 [==============================] - 0s 7ms/step - loss: 0.2351 -
binary_accuracy: 0.9038 - val_loss: 0.2432 - val_binary_accuracy: 0.9104
Epoch 6/10
30/30 [==============================] - 0s 7ms/step - loss: 0.2233 -
binary_accuracy: 0.9074 - val_loss: 0.2435 - val_binary_accuracy: 0.9100
Epoch 7/10
30/30 [==============================] - 0s 7ms/step - loss: 0.2095 -
binary_accuracy: 0.9116 - val_loss: 0.2465 - val_binary_accuracy: 0.9093
Epoch 8/10
```

```
30/30 [==============================] - 0s 7ms/step - loss: 0.1986 -
binary_accuracy: 0.9162 - val_loss: 0.2553 - val_binary_accuracy: 0.9065
Epoch 9/10
30/30 [==============================] - 0s 6ms/step - loss: 0.1907 -
binary_accuracy: 0.9177 - val_loss: 0.2580 - val_binary_accuracy: 0.9046
Epoch 10/10
30/30 [==============================] - 0s 7ms/step - loss: 0.1783 -
binary_accuracy: 0.9220 - val_loss: 0.2663 - val_binary_accuracy: 0.9011
9/9 [==============================] - 0s 2ms/step
Training with filters=32, kernel_size=5, dropout=0.3, epochs=10, batch_size=32
Epoch 1/10
30/30 [==============================] - 2s 16ms/step - loss: 0.3430 -
binary_accuracy: 0.8689 - val_loss: 0.2659 - val_binary_accuracy: 0.9085
Epoch 2/10
30/30 [==============================] - 0s 6ms/step - loss: 0.2731 -
binary_accuracy: 0.8953 - val_loss: 0.2461 - val_binary_accuracy: 0.9124
Epoch 3/10
30/30 [==============================] - 0s 7ms/step - loss: 0.2459 -
binary_accuracy: 0.9017 - val_loss: 0.2458 - val_binary_accuracy: 0.9112
Epoch 4/10
30/30 [==============================] - 0s 6ms/step - loss: 0.2287 -
binary_accuracy: 0.9080 - val_loss: 0.2441 - val_binary_accuracy: 0.9124
Epoch 5/10
30/30 [==============================] - 0s 7ms/step - loss: 0.2083 -
binary_accuracy: 0.9123 - val_loss: 0.2484 - val_binary_accuracy: 0.9023
Epoch 6/10
30/30 [==============================] - 0s 7ms/step - loss: 0.1941 -
binary_accuracy: 0.9171 - val_loss: 0.2555 - val_binary_accuracy: 0.9026
Epoch 7/10
30/30 [==============================] - 0s 6ms/step - loss: 0.1795 -
binary_accuracy: 0.9214 - val_loss: 0.2658 - val_binary_accuracy: 0.8949
Epoch 8/10
30/30 [==============================] - 0s 6ms/step - loss: 0.1678 -
binary_accuracy: 0.9280 - val_loss: 0.2757 - val_binary_accuracy: 0.8980
Epoch 9/10
30/30 [==============================] - 0s 7ms/step - loss: 0.1533 -
binary_accuracy: 0.9343 - val_loss: 0.2875 - val_binary_accuracy: 0.8988
Epoch 10/10
30/30 [==============================] - 0s 8ms/step - loss: 0.1412 -
binary_accuracy: 0.9394 - val_loss: 0.2936 - val_binary_accuracy: 0.8976
9/9 [==============================] - 0s 3ms/step
Training with filters=32, kernel_size=5, dropout=0.5, epochs=10, batch_size=32
Epoch 1/10
30/30 [==============================] - 2s 18ms/step - loss: 0.3682 -
binary_accuracy: 0.8619 - val_loss: 0.2697 - val_binary_accuracy: 0.9124
Epoch 2/10
30/30 [==============================] - 0s 9ms/step - loss: 0.2930 -
binary_accuracy: 0.8926 - val_loss: 0.2506 - val_binary_accuracy: 0.9128
```

```
Epoch 3/10
30/30 [==============================] - 0s 10ms/step - loss: 0.2638 -
binary_accuracy: 0.8966 - val_loss: 0.2412 - val_binary_accuracy: 0.9116
Epoch 4/10
30/30 [==============================] - 0s 9ms/step - loss: 0.2464 -
binary_accuracy: 0.9010 - val_loss: 0.2391 - val_binary_accuracy: 0.9155
Epoch 5/10
30/30 [==============================] - 0s 9ms/step - loss: 0.2315 -
binary_accuracy: 0.9047 - val_loss: 0.2416 - val_binary_accuracy: 0.9100
Epoch 6/10
30/30 [==============================] - 0s 9ms/step - loss: 0.2203 -
binary_accuracy: 0.9077 - val_loss: 0.2430 - val_binary_accuracy: 0.9085
Epoch 7/10
30/30 [==============================] - 0s 7ms/step - loss: 0.2102 -
binary_accuracy: 0.9124 - val_loss: 0.2471 - val_binary_accuracy: 0.9081
Epoch 8/10
30/30 [==============================] - 0s 7ms/step - loss: 0.2000 -
binary_accuracy: 0.9134 - val_loss: 0.2500 - val_binary_accuracy: 0.9077
Epoch 9/10
30/30 [==============================] - 0s 6ms/step - loss: 0.1927 -
binary_accuracy: 0.9162 - val_loss: 0.2578 - val_binary_accuracy: 0.9026
Epoch 10/10
30/30 [==============================] - 0s 6ms/step - loss: 0.1825 -
binary_accuracy: 0.9221 - val_loss: 0.2577 - val_binary_accuracy: 0.9042
9/9 [==============================] - 0s 3ms/step
Training with filters=64, kernel_size=3, dropout=0.3, epochs=10, batch_size=32
Epoch 1/10
30/30 [==============================] - 2s 18ms/step - loss: 0.3594 -
binary_accuracy: 0.8699 - val_loss: 0.2651 - val_binary_accuracy: 0.9097
Epoch 2/10
30/30 [==============================] - 0s 8ms/step - loss: 0.2723 -
binary_accuracy: 0.8961 - val_loss: 0.2497 - val_binary_accuracy: 0.9124
Epoch 3/10
30/30 [==============================] - 0s 8ms/step - loss: 0.2381 -
binary_accuracy: 0.9050 - val_loss: 0.2480 - val_binary_accuracy: 0.9065
Epoch 4/10
30/30 [==============================] - 0s 8ms/step - loss: 0.2143 -
binary_accuracy: 0.9108 - val_loss: 0.2500 - val_binary_accuracy: 0.9007
Epoch 5/10
30/30 [==============================] - 0s 9ms/step - loss: 0.1980 -
binary_accuracy: 0.9161 - val_loss: 0.2670 - val_binary_accuracy: 0.8949
Epoch 6/10
30/30 [==============================] - 0s 8ms/step - loss: 0.1800 -
binary_accuracy: 0.9221 - val_loss: 0.2725 - val_binary_accuracy: 0.8972
Epoch 7/10
30/30 [==============================] - 0s 8ms/step - loss: 0.1639 -
binary_accuracy: 0.9303 - val_loss: 0.2891 - val_binary_accuracy: 0.8898
Epoch 8/10
```

```
30/30 [==============================] - 0s 8ms/step - loss: 0.1512 -
binary_accuracy: 0.9357 - val_loss: 0.3007 - val_binary_accuracy: 0.8952
Epoch 9/10
30/30 [==============================] - 0s 9ms/step - loss: 0.1379 -
binary_accuracy: 0.9414 - val_loss: 0.3085 - val_binary_accuracy: 0.8949
Epoch 10/10
30/30 [==============================] - 0s 8ms/step - loss: 0.1258 -
binary_accuracy: 0.9467 - val_loss: 0.3232 - val_binary_accuracy: 0.8910
9/9 [==============================] - 0s 2ms/step
Training with filters=64, kernel_size=3, dropout=0.5, epochs=10, batch_size=32
Epoch 1/10
30/30 [==============================] - 2s 14ms/step - loss: 0.3760 -
binary_accuracy: 0.8584 - val_loss: 0.2683 - val_binary_accuracy: 0.9116
Epoch 2/10
30/30 [==============================] - 0s 8ms/step - loss: 0.2861 -
binary_accuracy: 0.8946 - val_loss: 0.2469 - val_binary_accuracy: 0.9132
Epoch 3/10
30/30 [==============================] - 0s 8ms/step - loss: 0.2598 -
binary_accuracy: 0.8983 - val_loss: 0.2382 - val_binary_accuracy: 0.9151
Epoch 4/10
30/30 [==============================] - 0s 8ms/step - loss: 0.2389 -
binary_accuracy: 0.9030 - val_loss: 0.2420 - val_binary_accuracy: 0.9108
Epoch 5/10
30/30 [==============================] - 0s 8ms/step - loss: 0.2239 -
binary_accuracy: 0.9071 - val_loss: 0.2429 - val_binary_accuracy: 0.9093
Epoch 6/10
30/30 [==============================] - 0s 8ms/step - loss: 0.2100 -
binary_accuracy: 0.9121 - val_loss: 0.2505 - val_binary_accuracy: 0.9081
Epoch 7/10
30/30 [==============================] - 0s 8ms/step - loss: 0.1963 -
binary_accuracy: 0.9168 - val_loss: 0.2587 - val_binary_accuracy: 0.9042
Epoch 8/10
30/30 [==============================] - 0s 8ms/step - loss: 0.1871 -
binary_accuracy: 0.9194 - val_loss: 0.2636 - val_binary_accuracy: 0.9019
Epoch 9/10
30/30 [==============================] - 0s 8ms/step - loss: 0.1785 -
binary_accuracy: 0.9248 - val_loss: 0.2745 - val_binary_accuracy: 0.9015
Epoch 10/10
30/30 [==============================] - 0s 8ms/step - loss: 0.1680 -
binary_accuracy: 0.9269 - val_loss: 0.2773 - val_binary_accuracy: 0.9011
9/9 [==============================] - 0s 2ms/step
Training with filters=64, kernel_size=5, dropout=0.3, epochs=10, batch_size=32
Epoch 1/10
30/30 [==============================] - 2s 18ms/step - loss: 0.3592 -
binary_accuracy: 0.8668 - val_loss: 0.2653 - val_binary_accuracy: 0.9097
Epoch 2/10
30/30 [==============================] - 0s 8ms/step - loss: 0.2761 -
binary_accuracy: 0.8933 - val_loss: 0.2461 - val_binary_accuracy: 0.9128
```

```
Epoch 3/10
30/30 [==============================] - 0s 8ms/step - loss: 0.2451 -
binary_accuracy: 0.9007 - val_loss: 0.2428 - val_binary_accuracy: 0.9081
Epoch 4/10
30/30 [==============================] - 0s 8ms/step - loss: 0.2262 -
binary_accuracy: 0.9064 - val_loss: 0.2463 - val_binary_accuracy: 0.9112
Epoch 5/10
30/30 [==============================] - 0s 8ms/step - loss: 0.2044 -
binary_accuracy: 0.9139 - val_loss: 0.2600 - val_binary_accuracy: 0.9054
Epoch 6/10
30/30 [==============================] - 0s 8ms/step - loss: 0.1880 -
binary_accuracy: 0.9199 - val_loss: 0.2634 - val_binary_accuracy: 0.9058
Epoch 7/10
30/30 [==============================] - 0s 8ms/step - loss: 0.1721 -
binary_accuracy: 0.9256 - val_loss: 0.2733 - val_binary_accuracy: 0.9034
Epoch 8/10
30/30 [==============================] - 0s 9ms/step - loss: 0.1571 -
binary_accuracy: 0.9340 - val_loss: 0.2878 - val_binary_accuracy: 0.8964
Epoch 9/10
30/30 [==============================] - 0s 8ms/step - loss: 0.1441 -
binary_accuracy: 0.9381 - val_loss: 0.2973 - val_binary_accuracy: 0.8902
Epoch 10/10
30/30 [==============================] - 0s 8ms/step - loss: 0.1339 -
binary_accuracy: 0.9445 - val_loss: 0.3206 - val_binary_accuracy: 0.8956
9/9 [==============================] - 0s 2ms/step
Training with filters=64, kernel_size=5, dropout=0.5, epochs=10, batch_size=32
Epoch 1/10
30/30 [==============================] - 2s 14ms/step - loss: 0.3722 -
binary_accuracy: 0.8620 - val_loss: 0.2689 - val_binary_accuracy: 0.9108
Epoch 2/10
30/30 [==============================] - 0s 8ms/step - loss: 0.2875 -
binary_accuracy: 0.8921 - val_loss: 0.2474 - val_binary_accuracy: 0.9136
Epoch 3/10
30/30 [==============================] - 0s 9ms/step - loss: 0.2591 -
binary_accuracy: 0.8954 - val_loss: 0.2436 - val_binary_accuracy: 0.9136
Epoch 4/10
30/30 [==============================] - 0s 8ms/step - loss: 0.2416 -
binary_accuracy: 0.9018 - val_loss: 0.2423 - val_binary_accuracy: 0.9116
Epoch 5/10
30/30 [==============================] - 0s 9ms/step - loss: 0.2244 -
binary_accuracy: 0.9079 - val_loss: 0.2458 - val_binary_accuracy: 0.9085
Epoch 6/10
30/30 [==============================] - 0s 8ms/step - loss: 0.2122 -
binary_accuracy: 0.9099 - val_loss: 0.2524 - val_binary_accuracy: 0.9108
Epoch 7/10
30/30 [==============================] - 0s 8ms/step - loss: 0.1989 -
binary_accuracy: 0.9148 - val_loss: 0.2594 - val_binary_accuracy: 0.9093
Epoch 8/10
```

```
30/30 [==============================] - 0s 8ms/step - loss: 0.1890 -
binary_accuracy: 0.9191 - val_loss: 0.2614 - val_binary_accuracy: 0.9026
Epoch 9/10
30/30 [==============================] - 0s 8ms/step - loss: 0.1794 -
binary_accuracy: 0.9226 - val_loss: 0.2772 - val_binary_accuracy: 0.9046
Epoch 10/10
30/30 [==============================] - 0s 8ms/step - loss: 0.1738 -
binary_accuracy: 0.9258 - val_loss: 0.2760 - val_binary_accuracy: 0.8937
9/9 [==============================] - 0s 3ms/step
Best Params: {'filters': 64, 'kernel_size': 3, 'dropout': 0.5, 'epochs': 10,
'batch_size': 32}
Best Binary Accuracy: 0.9076
Best Normal Accuracy: 0.2472
Best F1 Score: 0.9510
```

[ ]:
```python
# CNN - continued with 15 epochs

# 2. CNN

from tensorflow.keras.models import Sequential
from tensorflow.keras.layers import Conv1D, MaxPooling1D, Dense, Flatten,␣
 ↪Dropout
from sklearn.metrics import f1_score, accuracy_score
import numpy as np

# Define the CNN model
def build_cnn_model(filters=64, kernel_size=3, dropout=0.5):
    model = Sequential()

    # 1D Convolutional Layer
    model.add(Conv1D(filters=filters, kernel_size=kernel_size,␣
 ↪activation='relu', input_shape=(X_train.shape[1], 1)))

    # MaxPooling Layer
    model.add(MaxPooling1D(pool_size=2))

    # Dropout Layer
    model.add(Dropout(dropout))

    # Flatten the output for dense layers
    model.add(Flatten())

    # Dense Layers
    model.add(Dense(256, activation='relu'))
    model.add(Dropout(dropout))

    # Output layer for multi-label classification
```

```python
    model.add(Dense(y_train.shape[1], activation='sigmoid'))

    # Compile the model
    model.compile(optimizer='adam', loss='binary_crossentropy', 
 ↪metrics=['binary_accuracy'])

    return model

# Reshape your data for CNN input
X_train = X_train.reshape((X_train.shape[0], X_train.shape[1], 1))
X_test = X_test.reshape((X_test.shape[0], X_test.shape[1], 1))

# Hyperparameters grid
param_grid = {
    'filters': [32, 64],  # Number of filters for Conv1D
    'kernel_size': [3],  # Kernel size for Conv1D
    'dropout': [0.3, 0.5],  # Dropout rate
    'epochs': [15],  # Number of epochs
    'batch_size': [32,64],  # Batch size
}

# Variables to track the best model
best_binary_acc = 0
best_params = {}
best_f1 = 0
best_normal_acc = 0

# Perform the manual hyperparameter search
for filters in param_grid['filters']:
    for kernel_size in param_grid['kernel_size']:  # Now this will work
        for dropout in param_grid['dropout']:
            for epochs in param_grid['epochs']:
                for batch_size in param_grid['batch_size']:
                    print(f'Training with filters={filters}, 
 ↪kernel_size={kernel_size}, dropout={dropout}, epochs={epochs}, 
 ↪batch_size={batch_size}')

                    # Build the model
                    model = build_cnn_model(filters=filters, 
 ↪kernel_size=kernel_size, dropout=dropout)

                    # Train the model
                    model.fit(X_train, y_train, epochs=epochs, 
 ↪batch_size=batch_size, validation_split=0.1, verbose=1)

                    # Predict on the test set
                    y_pred = model.predict(X_test)
```

```
                    y_pred_binary = (y_pred > 0.5).astype(int)

                    # Calculate binary accuracy
                    binary_acc = np.mean(np.equal(y_test, y_pred_binary).
↪astype(int))

                    # Calculate normal accuracy
                    normal_acc = accuracy_score(y_test, y_pred_binary)

                    # Calculate F1 Score
                    f1 = f1_score(y_test, y_pred_binary, average='micro')

                    # Update best params if this model is better
                    if binary_acc > best_binary_acc:
                        best_binary_acc = binary_acc
                        best_f1 = f1
                        best_normal_acc = normal_acc
                        best_params = {
                            'filters': filters,
                            'kernel_size': kernel_size,
                            'dropout': dropout,
                            'epochs': epochs,
                            'batch_size': batch_size,
                        }

# Print the best results
print(f'Best Params: {best_params}')
print(f'Best Binary Accuracy: {best_binary_acc:.4f}')
print(f'Best Normal Accuracy: {best_normal_acc:.4f}')
print(f'Best F1 Score: {best_f1:.4f}')
```

```
Training with filters=32, kernel_size=3, dropout=0.3, epochs=15, batch_size=32
Epoch 1/15
30/30 [==============================] - 2s 12ms/step - loss: 0.3543 -
binary_accuracy: 0.8660 - val_loss: 0.2638 - val_binary_accuracy: 0.9065
Epoch 2/15
30/30 [==============================] - 0s 7ms/step - loss: 0.2697 -
binary_accuracy: 0.8977 - val_loss: 0.2476 - val_binary_accuracy: 0.9108
Epoch 3/15
30/30 [==============================] - 0s 6ms/step - loss: 0.2404 -
binary_accuracy: 0.9050 - val_loss: 0.2463 - val_binary_accuracy: 0.9108
Epoch 4/15
30/30 [==============================] - 0s 6ms/step - loss: 0.2183 -
binary_accuracy: 0.9110 - val_loss: 0.2493 - val_binary_accuracy: 0.9073
Epoch 5/15
30/30 [==============================] - 0s 6ms/step - loss: 0.2039 -
binary_accuracy: 0.9149 - val_loss: 0.2597 - val_binary_accuracy: 0.9062
```

```
Epoch 6/15
30/30 [==============================] - 0s 6ms/step - loss: 0.1874 -
binary_accuracy: 0.9202 - val_loss: 0.2647 - val_binary_accuracy: 0.8968
Epoch 7/15
30/30 [==============================] - 0s 7ms/step - loss: 0.1728 -
binary_accuracy: 0.9245 - val_loss: 0.2740 - val_binary_accuracy: 0.9030
Epoch 8/15
30/30 [==============================] - 0s 6ms/step - loss: 0.1635 -
binary_accuracy: 0.9307 - val_loss: 0.2812 - val_binary_accuracy: 0.9023
Epoch 9/15
30/30 [==============================] - 0s 6ms/step - loss: 0.1502 -
binary_accuracy: 0.9363 - val_loss: 0.2911 - val_binary_accuracy: 0.8941
Epoch 10/15
30/30 [==============================] - 0s 7ms/step - loss: 0.1363 -
binary_accuracy: 0.9430 - val_loss: 0.3056 - val_binary_accuracy: 0.8824
Epoch 11/15
30/30 [==============================] - 0s 7ms/step - loss: 0.1280 -
binary_accuracy: 0.9461 - val_loss: 0.3257 - val_binary_accuracy: 0.8933
Epoch 12/15
30/30 [==============================] - 0s 7ms/step - loss: 0.1194 -
binary_accuracy: 0.9498 - val_loss: 0.3351 - val_binary_accuracy: 0.8882
Epoch 13/15
30/30 [==============================] - 0s 6ms/step - loss: 0.1079 -
binary_accuracy: 0.9548 - val_loss: 0.3511 - val_binary_accuracy: 0.8906
Epoch 14/15
30/30 [==============================] - 0s 6ms/step - loss: 0.1022 -
binary_accuracy: 0.9582 - val_loss: 0.3650 - val_binary_accuracy: 0.8863
Epoch 15/15
30/30 [==============================] - 0s 7ms/step - loss: 0.0935 -
binary_accuracy: 0.9634 - val_loss: 0.3727 - val_binary_accuracy: 0.8816
9/9 [==============================] - 0s 2ms/step
Training with filters=32, kernel_size=3, dropout=0.3, epochs=15, batch_size=64
Epoch 1/15
15/15 [==============================] - 2s 29ms/step - loss: 0.3761 -
binary_accuracy: 0.8652 - val_loss: 0.2705 - val_binary_accuracy: 0.8949
Epoch 2/15
15/15 [==============================] - 0s 9ms/step - loss: 0.3001 -
binary_accuracy: 0.8874 - val_loss: 0.2602 - val_binary_accuracy: 0.9112
Epoch 3/15
15/15 [==============================] - 0s 8ms/step - loss: 0.2683 -
binary_accuracy: 0.8987 - val_loss: 0.2529 - val_binary_accuracy: 0.9120
Epoch 4/15
15/15 [==============================] - 0s 9ms/step - loss: 0.2469 -
binary_accuracy: 0.9050 - val_loss: 0.2466 - val_binary_accuracy: 0.9112
Epoch 5/15
15/15 [==============================] - 0s 8ms/step - loss: 0.2296 -
binary_accuracy: 0.9084 - val_loss: 0.2455 - val_binary_accuracy: 0.9128
Epoch 6/15
```

```
15/15 [==============================] - 0s 8ms/step - loss: 0.2167 -
binary_accuracy: 0.9100 - val_loss: 0.2481 - val_binary_accuracy: 0.9054
Epoch 7/15
15/15 [==============================] - 0s 10ms/step - loss: 0.2020 -
binary_accuracy: 0.9158 - val_loss: 0.2528 - val_binary_accuracy: 0.9034
Epoch 8/15
15/15 [==============================] - 0s 8ms/step - loss: 0.1920 -
binary_accuracy: 0.9181 - val_loss: 0.2566 - val_binary_accuracy: 0.9038
Epoch 9/15
15/15 [==============================] - 0s 10ms/step - loss: 0.1813 -
binary_accuracy: 0.9231 - val_loss: 0.2651 - val_binary_accuracy: 0.8995
Epoch 10/15
15/15 [==============================] - 0s 9ms/step - loss: 0.1705 -
binary_accuracy: 0.9277 - val_loss: 0.2695 - val_binary_accuracy: 0.9007
Epoch 11/15
15/15 [==============================] - 0s 10ms/step - loss: 0.1624 -
binary_accuracy: 0.9278 - val_loss: 0.2708 - val_binary_accuracy: 0.8956
Epoch 12/15
15/15 [==============================] - 0s 8ms/step - loss: 0.1541 -
binary_accuracy: 0.9348 - val_loss: 0.2826 - val_binary_accuracy: 0.8960
Epoch 13/15
15/15 [==============================] - 0s 8ms/step - loss: 0.1445 -
binary_accuracy: 0.9387 - val_loss: 0.2928 - val_binary_accuracy: 0.8902
Epoch 14/15
15/15 [==============================] - 0s 8ms/step - loss: 0.1372 -
binary_accuracy: 0.9436 - val_loss: 0.3012 - val_binary_accuracy: 0.8910
Epoch 15/15
15/15 [==============================] - 0s 11ms/step - loss: 0.1304 -
binary_accuracy: 0.9442 - val_loss: 0.3099 - val_binary_accuracy: 0.8843
9/9 [==============================] - 0s 2ms/step
Training with filters=32, kernel_size=3, dropout=0.5, epochs=15, batch_size=32
Epoch 1/15
30/30 [==============================] - 2s 13ms/step - loss: 0.3731 -
binary_accuracy: 0.8620 - val_loss: 0.2635 - val_binary_accuracy: 0.9124
Epoch 2/15
30/30 [==============================] - 0s 6ms/step - loss: 0.2902 -
binary_accuracy: 0.8897 - val_loss: 0.2508 - val_binary_accuracy: 0.9100
Epoch 3/15
30/30 [==============================] - 0s 7ms/step - loss: 0.2612 -
binary_accuracy: 0.8952 - val_loss: 0.2500 - val_binary_accuracy: 0.9093
Epoch 4/15
30/30 [==============================] - 0s 6ms/step - loss: 0.2477 -
binary_accuracy: 0.9022 - val_loss: 0.2396 - val_binary_accuracy: 0.9124
Epoch 5/15
30/30 [==============================] - 0s 7ms/step - loss: 0.2294 -
binary_accuracy: 0.9061 - val_loss: 0.2436 - val_binary_accuracy: 0.9097
Epoch 6/15
30/30 [==============================] - 0s 6ms/step - loss: 0.2197 -
```

```
binary_accuracy: 0.9072 - val_loss: 0.2465 - val_binary_accuracy: 0.9085
Epoch 7/15
30/30 [==============================] - 0s 6ms/step - loss: 0.2099 -
binary_accuracy: 0.9114 - val_loss: 0.2486 - val_binary_accuracy: 0.9085
Epoch 8/15
30/30 [==============================] - 0s 7ms/step - loss: 0.1984 -
binary_accuracy: 0.9151 - val_loss: 0.2563 - val_binary_accuracy: 0.9089
Epoch 9/15
30/30 [==============================] - 0s 6ms/step - loss: 0.1890 -
binary_accuracy: 0.9193 - val_loss: 0.2588 - val_binary_accuracy: 0.9054
Epoch 10/15
30/30 [==============================] - 0s 7ms/step - loss: 0.1859 -
binary_accuracy: 0.9195 - val_loss: 0.2653 - val_binary_accuracy: 0.8995
Epoch 11/15
30/30 [==============================] - 0s 7ms/step - loss: 0.1764 -
binary_accuracy: 0.9248 - val_loss: 0.2716 - val_binary_accuracy: 0.8991
Epoch 12/15
30/30 [==============================] - 0s 6ms/step - loss: 0.1696 -
binary_accuracy: 0.9264 - val_loss: 0.2767 - val_binary_accuracy: 0.8956
Epoch 13/15
30/30 [==============================] - 0s 7ms/step - loss: 0.1612 -
binary_accuracy: 0.9309 - val_loss: 0.2821 - val_binary_accuracy: 0.8980
Epoch 14/15
30/30 [==============================] - 0s 6ms/step - loss: 0.1517 -
binary_accuracy: 0.9350 - val_loss: 0.2931 - val_binary_accuracy: 0.8964
Epoch 15/15
30/30 [==============================] - 0s 7ms/step - loss: 0.1460 -
binary_accuracy: 0.9382 - val_loss: 0.3011 - val_binary_accuracy: 0.8937
9/9 [==============================] - 0s 2ms/step
Training with filters=32, kernel_size=3, dropout=0.5, epochs=15, batch_size=64
Epoch 1/15
15/15 [==============================] - 1s 21ms/step - loss: 0.4010 -
binary_accuracy: 0.8578 - val_loss: 0.2674 - val_binary_accuracy: 0.8941
Epoch 2/15
15/15 [==============================] - 0s 9ms/step - loss: 0.3179 -
binary_accuracy: 0.8818 - val_loss: 0.2632 - val_binary_accuracy: 0.9112
Epoch 3/15
15/15 [==============================] - 0s 9ms/step - loss: 0.2827 -
binary_accuracy: 0.8948 - val_loss: 0.2499 - val_binary_accuracy: 0.9097
Epoch 4/15
15/15 [==============================] - 0s 8ms/step - loss: 0.2612 -
binary_accuracy: 0.8978 - val_loss: 0.2461 - val_binary_accuracy: 0.9100
Epoch 5/15
15/15 [==============================] - 0s 8ms/step - loss: 0.2467 -
binary_accuracy: 0.9030 - val_loss: 0.2438 - val_binary_accuracy: 0.9097
Epoch 6/15
15/15 [==============================] - 0s 8ms/step - loss: 0.2302 -
binary_accuracy: 0.9063 - val_loss: 0.2413 - val_binary_accuracy: 0.9065
```

```
Epoch 7/15
15/15 [==============================] - 0s 9ms/step - loss: 0.2227 -
binary_accuracy: 0.9089 - val_loss: 0.2441 - val_binary_accuracy: 0.9069
Epoch 8/15
15/15 [==============================] - 0s 9ms/step - loss: 0.2144 -
binary_accuracy: 0.9123 - val_loss: 0.2457 - val_binary_accuracy: 0.9104
Epoch 9/15
15/15 [==============================] - 0s 8ms/step - loss: 0.2043 -
binary_accuracy: 0.9140 - val_loss: 0.2504 - val_binary_accuracy: 0.9038
Epoch 10/15
15/15 [==============================] - 0s 10ms/step - loss: 0.1981 -
binary_accuracy: 0.9170 - val_loss: 0.2524 - val_binary_accuracy: 0.9011
Epoch 11/15
15/15 [==============================] - 0s 9ms/step - loss: 0.1886 -
binary_accuracy: 0.9195 - val_loss: 0.2564 - val_binary_accuracy: 0.9007
Epoch 12/15
15/15 [==============================] - 0s 10ms/step - loss: 0.1850 -
binary_accuracy: 0.9216 - val_loss: 0.2631 - val_binary_accuracy: 0.8976
Epoch 13/15
15/15 [==============================] - 0s 13ms/step - loss: 0.1780 -
binary_accuracy: 0.9241 - val_loss: 0.2668 - val_binary_accuracy: 0.9007
Epoch 14/15
15/15 [==============================] - 0s 12ms/step - loss: 0.1691 -
binary_accuracy: 0.9269 - val_loss: 0.2713 - val_binary_accuracy: 0.8991
Epoch 15/15
15/15 [==============================] - 0s 12ms/step - loss: 0.1654 -
binary_accuracy: 0.9279 - val_loss: 0.2733 - val_binary_accuracy: 0.8964
9/9 [==============================] - 0s 3ms/step
Training with filters=64, kernel_size=3, dropout=0.3, epochs=15, batch_size=32
Epoch 1/15
30/30 [==============================] - 2s 15ms/step - loss: 0.3526 -
binary_accuracy: 0.8754 - val_loss: 0.2600 - val_binary_accuracy: 0.9128
Epoch 2/15
30/30 [==============================] - 0s 8ms/step - loss: 0.2657 -
binary_accuracy: 0.8981 - val_loss: 0.2446 - val_binary_accuracy: 0.9108
Epoch 3/15
30/30 [==============================] - 0s 8ms/step - loss: 0.2376 -
binary_accuracy: 0.9021 - val_loss: 0.2467 - val_binary_accuracy: 0.9100
Epoch 4/15
30/30 [==============================] - 0s 9ms/step - loss: 0.2156 -
binary_accuracy: 0.9111 - val_loss: 0.2484 - val_binary_accuracy: 0.9081
Epoch 5/15
30/30 [==============================] - 0s 9ms/step - loss: 0.1952 -
binary_accuracy: 0.9177 - val_loss: 0.2605 - val_binary_accuracy: 0.8952
Epoch 6/15
30/30 [==============================] - 0s 8ms/step - loss: 0.1800 -
binary_accuracy: 0.9237 - val_loss: 0.2710 - val_binary_accuracy: 0.9034
Epoch 7/15
```

```
30/30 [==============================] - 0s 8ms/step - loss: 0.1662 -
binary_accuracy: 0.9279 - val_loss: 0.2794 - val_binary_accuracy: 0.8991
Epoch 8/15
30/30 [==============================] - 0s 9ms/step - loss: 0.1466 -
binary_accuracy: 0.9369 - val_loss: 0.2979 - val_binary_accuracy: 0.8890
Epoch 9/15
30/30 [==============================] - 0s 9ms/step - loss: 0.1324 -
binary_accuracy: 0.9440 - val_loss: 0.3166 - val_binary_accuracy: 0.8910
Epoch 10/15
30/30 [==============================] - 0s 8ms/step - loss: 0.1175 -
binary_accuracy: 0.9511 - val_loss: 0.3319 - val_binary_accuracy: 0.8890
Epoch 11/15
30/30 [==============================] - 0s 9ms/step - loss: 0.1083 -
binary_accuracy: 0.9564 - val_loss: 0.3496 - val_binary_accuracy: 0.8812
Epoch 12/15
30/30 [==============================] - 0s 9ms/step - loss: 0.1002 -
binary_accuracy: 0.9589 - val_loss: 0.3591 - val_binary_accuracy: 0.8859
Epoch 13/15
30/30 [==============================] - 0s 9ms/step - loss: 0.0906 -
binary_accuracy: 0.9647 - val_loss: 0.3819 - val_binary_accuracy: 0.8871
Epoch 14/15
30/30 [==============================] - 0s 8ms/step - loss: 0.0801 -
binary_accuracy: 0.9695 - val_loss: 0.4096 - val_binary_accuracy: 0.8867
Epoch 15/15
30/30 [==============================] - 0s 8ms/step - loss: 0.0737 -
binary_accuracy: 0.9704 - val_loss: 0.4126 - val_binary_accuracy: 0.8812
9/9 [==============================] - 0s 2ms/step
Training with filters=64, kernel_size=3, dropout=0.3, epochs=15, batch_size=64
Epoch 1/15
15/15 [==============================] - 2s 52ms/step - loss: 0.3764 -
binary_accuracy: 0.8580 - val_loss: 0.2553 - val_binary_accuracy: 0.9124
Epoch 2/15
15/15 [==============================] - 0s 13ms/step - loss: 0.2912 -
binary_accuracy: 0.8925 - val_loss: 0.2547 - val_binary_accuracy: 0.9128
Epoch 3/15
15/15 [==============================] - 0s 13ms/step - loss: 0.2600 -
binary_accuracy: 0.8999 - val_loss: 0.2509 - val_binary_accuracy: 0.9073
Epoch 4/15
15/15 [==============================] - 0s 11ms/step - loss: 0.2395 -
binary_accuracy: 0.9036 - val_loss: 0.2410 - val_binary_accuracy: 0.9116
Epoch 5/15
15/15 [==============================] - 0s 11ms/step - loss: 0.2235 -
binary_accuracy: 0.9072 - val_loss: 0.2468 - val_binary_accuracy: 0.9116
Epoch 6/15
15/15 [==============================] - 0s 13ms/step - loss: 0.2050 -
binary_accuracy: 0.9148 - val_loss: 0.2508 - val_binary_accuracy: 0.9089
Epoch 7/15
15/15 [==============================] - 0s 14ms/step - loss: 0.1905 -
```

```
binary_accuracy: 0.9201 - val_loss: 0.2525 - val_binary_accuracy: 0.9026
Epoch 8/15
15/15 [==============================] - 0s 16ms/step - loss: 0.1776 -
binary_accuracy: 0.9248 - val_loss: 0.2622 - val_binary_accuracy: 0.9015
Epoch 9/15
15/15 [==============================] - 0s 15ms/step - loss: 0.1653 -
binary_accuracy: 0.9298 - val_loss: 0.2692 - val_binary_accuracy: 0.8945
Epoch 10/15
15/15 [==============================] - 0s 14ms/step - loss: 0.1578 -
binary_accuracy: 0.9327 - val_loss: 0.2801 - val_binary_accuracy: 0.8991
Epoch 11/15
15/15 [==============================] - 0s 14ms/step - loss: 0.1424 -
binary_accuracy: 0.9402 - val_loss: 0.2955 - val_binary_accuracy: 0.8941
Epoch 12/15
15/15 [==============================] - 0s 13ms/step - loss: 0.1355 -
binary_accuracy: 0.9437 - val_loss: 0.3054 - val_binary_accuracy: 0.8995
Epoch 13/15
15/15 [==============================] - 0s 17ms/step - loss: 0.1237 -
binary_accuracy: 0.9485 - val_loss: 0.3118 - val_binary_accuracy: 0.8964
Epoch 14/15
15/15 [==============================] - 0s 14ms/step - loss: 0.1157 -
binary_accuracy: 0.9533 - val_loss: 0.3236 - val_binary_accuracy: 0.8902
Epoch 15/15
15/15 [==============================] - 0s 15ms/step - loss: 0.1091 -
binary_accuracy: 0.9552 - val_loss: 0.3438 - val_binary_accuracy: 0.8875
9/9 [==============================] - 0s 4ms/step
Training with filters=64, kernel_size=3, dropout=0.5, epochs=15, batch_size=32
Epoch 1/15
30/30 [==============================] - 2s 14ms/step - loss: 0.3756 -
binary_accuracy: 0.8627 - val_loss: 0.2673 - val_binary_accuracy: 0.9124
Epoch 2/15
30/30 [==============================] - 0s 9ms/step - loss: 0.2856 -
binary_accuracy: 0.8947 - val_loss: 0.2477 - val_binary_accuracy: 0.9120
Epoch 3/15
30/30 [==============================] - 0s 8ms/step - loss: 0.2552 -
binary_accuracy: 0.8991 - val_loss: 0.2411 - val_binary_accuracy: 0.9120
Epoch 4/15
30/30 [==============================] - 0s 9ms/step - loss: 0.2341 -
binary_accuracy: 0.9049 - val_loss: 0.2433 - val_binary_accuracy: 0.9073
Epoch 5/15
30/30 [==============================] - 0s 9ms/step - loss: 0.2202 -
binary_accuracy: 0.9087 - val_loss: 0.2509 - val_binary_accuracy: 0.9038
Epoch 6/15
30/30 [==============================] - 0s 9ms/step - loss: 0.2052 -
binary_accuracy: 0.9127 - val_loss: 0.2513 - val_binary_accuracy: 0.9038
Epoch 7/15
30/30 [==============================] - 0s 8ms/step - loss: 0.1941 -
binary_accuracy: 0.9181 - val_loss: 0.2604 - val_binary_accuracy: 0.8972
```

```
Epoch 8/15
30/30 [==============================] - 0s 8ms/step - loss: 0.1842 -
binary_accuracy: 0.9198 - val_loss: 0.2650 - val_binary_accuracy: 0.8991
Epoch 9/15
30/30 [==============================] - 0s 8ms/step - loss: 0.1756 -
binary_accuracy: 0.9251 - val_loss: 0.2744 - val_binary_accuracy: 0.8956
Epoch 10/15
30/30 [==============================] - 0s 8ms/step - loss: 0.1678 -
binary_accuracy: 0.9283 - val_loss: 0.2802 - val_binary_accuracy: 0.9011
Epoch 11/15
30/30 [==============================] - 0s 8ms/step - loss: 0.1580 -
binary_accuracy: 0.9325 - val_loss: 0.2854 - val_binary_accuracy: 0.9015
Epoch 12/15
30/30 [==============================] - 0s 8ms/step - loss: 0.1499 -
binary_accuracy: 0.9348 - val_loss: 0.2983 - val_binary_accuracy: 0.8984
Epoch 13/15
30/30 [==============================] - 0s 9ms/step - loss: 0.1392 -
binary_accuracy: 0.9409 - val_loss: 0.3075 - val_binary_accuracy: 0.8999
Epoch 14/15
30/30 [==============================] - 0s 8ms/step - loss: 0.1305 -
binary_accuracy: 0.9447 - val_loss: 0.3109 - val_binary_accuracy: 0.8921
Epoch 15/15
30/30 [==============================] - 0s 8ms/step - loss: 0.1303 -
binary_accuracy: 0.9454 - val_loss: 0.3276 - val_binary_accuracy: 0.8960
9/9 [==============================] - 0s 3ms/step
Training with filters=64, kernel_size=3, dropout=0.5, epochs=15, batch_size=64
Epoch 1/15
15/15 [==============================] - 2s 24ms/step - loss: 0.4033 -
binary_accuracy: 0.8484 - val_loss: 0.2599 - val_binary_accuracy: 0.9120
Epoch 2/15
15/15 [==============================] - 0s 12ms/step - loss: 0.3130 -
binary_accuracy: 0.8847 - val_loss: 0.2606 - val_binary_accuracy: 0.9124
Epoch 3/15
15/15 [==============================] - 0s 12ms/step - loss: 0.2773 -
binary_accuracy: 0.8946 - val_loss: 0.2484 - val_binary_accuracy: 0.9124
Epoch 4/15
15/15 [==============================] - 0s 12ms/step - loss: 0.2502 -
binary_accuracy: 0.8998 - val_loss: 0.2471 - val_binary_accuracy: 0.9136
Epoch 5/15
15/15 [==============================] - 0s 12ms/step - loss: 0.2410 -
binary_accuracy: 0.9042 - val_loss: 0.2447 - val_binary_accuracy: 0.9128
Epoch 6/15
15/15 [==============================] - 0s 12ms/step - loss: 0.2249 -
binary_accuracy: 0.9066 - val_loss: 0.2488 - val_binary_accuracy: 0.9089
Epoch 7/15
15/15 [==============================] - 0s 11ms/step - loss: 0.2123 -
binary_accuracy: 0.9118 - val_loss: 0.2490 - val_binary_accuracy: 0.9116
Epoch 8/15
```

```
15/15 [==============================] - 0s 11ms/step - loss: 0.2015 -
binary_accuracy: 0.9144 - val_loss: 0.2529 - val_binary_accuracy: 0.9077
Epoch 9/15
15/15 [==============================] - 0s 12ms/step - loss: 0.1919 -
binary_accuracy: 0.9183 - val_loss: 0.2576 - val_binary_accuracy: 0.9038
Epoch 10/15
15/15 [==============================] - 0s 13ms/step - loss: 0.1839 -
binary_accuracy: 0.9220 - val_loss: 0.2686 - val_binary_accuracy: 0.8999
Epoch 11/15
15/15 [==============================] - 0s 11ms/step - loss: 0.1748 -
binary_accuracy: 0.9245 - val_loss: 0.2702 - val_binary_accuracy: 0.8980
Epoch 12/15
15/15 [==============================] - 0s 12ms/step - loss: 0.1675 -
binary_accuracy: 0.9292 - val_loss: 0.2767 - val_binary_accuracy: 0.8988
Epoch 13/15
15/15 [==============================] - 0s 11ms/step - loss: 0.1612 -
binary_accuracy: 0.9309 - val_loss: 0.2813 - val_binary_accuracy: 0.9011
Epoch 14/15
15/15 [==============================] - 0s 12ms/step - loss: 0.1523 -
binary_accuracy: 0.9363 - val_loss: 0.2849 - val_binary_accuracy: 0.8960
Epoch 15/15
15/15 [==============================] - 0s 11ms/step - loss: 0.1479 -
binary_accuracy: 0.9366 - val_loss: 0.2959 - val_binary_accuracy: 0.8902
9/9 [==============================] - 0s 3ms/step
Best Params: {'filters': 32, 'kernel_size': 3, 'dropout': 0.5, 'epochs': 15,
'batch_size': 64}
Best Binary Accuracy: 0.9034
Best Normal Accuracy: 0.2097
Best F1 Score: 0.9485
```

[ ]:

```python
# CNN - continued with 30 epochs

# 2. CNN

from tensorflow.keras.models import Sequential
from tensorflow.keras.layers import Conv1D, MaxPooling1D, Dense, Flatten,␣
 ↪Dropout
from sklearn.metrics import f1_score, accuracy_score
import numpy as np

# Define the CNN model
def build_cnn_model(filters=64, kernel_size=3, dropout=0.5):
    model = Sequential()

    # 1D Convolutional Layer
    model.add(Conv1D(filters=filters, kernel_size=kernel_size,␣
 ↪activation='relu', input_shape=(X_train.shape[1], 1)))
```

```python
    # MaxPooling Layer
    model.add(MaxPooling1D(pool_size=2))

    # Dropout Layer
    model.add(Dropout(dropout))

    # Flatten the output for dense layers
    model.add(Flatten())

    # Dense Layers
    model.add(Dense(256, activation='relu'))
    model.add(Dropout(dropout))

    # Output layer for multi-label classification
    model.add(Dense(y_train.shape[1], activation='sigmoid'))

    # Compile the model
    model.compile(optimizer='adam', loss='binary_crossentropy',␣
↪metrics=['binary_accuracy'])

    return model

# Reshape your data for CNN input
X_train = X_train.reshape((X_train.shape[0], X_train.shape[1], 1))
X_test = X_test.reshape((X_test.shape[0], X_test.shape[1], 1))

# Hyperparameters grid
param_grid = {
    'filters': [32, 64],  # Number of filters for Conv1D
    'kernel_size': [3],  # Kernel size for Conv1D
    'dropout': [0.3, 0.5],  # Dropout rate
    'epochs': [30],  # Number of epochs
    'batch_size': [32,64],  # Batch size
}

# Variables to track the best model
best_binary_acc = 0
best_params = {}
best_f1 = 0
best_normal_acc = 0

# Perform the manual hyperparameter search
for filters in param_grid['filters']:
    for kernel_size in param_grid['kernel_size']:  # Now this will work
        for dropout in param_grid['dropout']:
            for epochs in param_grid['epochs']:
```

```
                for batch_size in param_grid['batch_size']:
                    print(f'Training with filters={filters},␣
↪kernel_size={kernel_size}, dropout={dropout}, epochs={epochs},␣
↪batch_size={batch_size}')

                    # Build the model
                    model = build_cnn_model(filters=filters,␣
↪kernel_size=kernel_size, dropout=dropout)

                    # Train the model
                    model.fit(X_train, y_train, epochs=epochs,␣
↪batch_size=batch_size, validation_split=0.1, verbose=0)

                    # Predict on the test set
                    y_pred = model.predict(X_test)
                    y_pred_binary = (y_pred > 0.5).astype(int)

                    # Calculate binary accuracy
                    binary_acc = np.mean(np.equal(y_test, y_pred_binary).
↪astype(int))

                    # Calculate normal accuracy
                    normal_acc = accuracy_score(y_test, y_pred_binary)

                    # Calculate F1 Score
                    f1 = f1_score(y_test, y_pred_binary, average='micro')

                    # Update best params if this model is better
                    if binary_acc > best_binary_acc:
                        best_binary_acc = binary_acc
                        best_f1 = f1
                        best_normal_acc = normal_acc
                        best_params = {
                            'filters': filters,
                            'kernel_size': kernel_size,
                            'dropout': dropout,
                            'epochs': epochs,
                            'batch_size': batch_size,
                        }

# Print the best results
print(f'Best Params: {best_params}')
print(f'Best Binary Accuracy: {best_binary_acc:.4f}')
print(f'Best Normal Accuracy: {best_normal_acc:.4f}')
print(f'Best F1 Score: {best_f1:.4f}')
```

```
Training with filters=32, kernel_size=3, dropout=0.3, epochs=30, batch_size=32
```

```
9/9 [==============================] - 0s 3ms/step
Training with filters=32, kernel_size=3, dropout=0.3, epochs=30, batch_size=64
9/9 [==============================] - 0s 3ms/step
Training with filters=32, kernel_size=3, dropout=0.5, epochs=30, batch_size=32
9/9 [==============================] - 0s 3ms/step
Training with filters=32, kernel_size=3, dropout=0.5, epochs=30, batch_size=64
9/9 [==============================] - 0s 2ms/step
Training with filters=64, kernel_size=3, dropout=0.3, epochs=30, batch_size=32
9/9 [==============================] - 0s 2ms/step
Training with filters=64, kernel_size=3, dropout=0.3, epochs=30, batch_size=64
9/9 [==============================] - 0s 3ms/step
Training with filters=64, kernel_size=3, dropout=0.5, epochs=30, batch_size=32
9/9 [==============================] - 0s 2ms/step
Training with filters=64, kernel_size=3, dropout=0.5, epochs=30, batch_size=64
9/9 [==============================] - 0s 2ms/step
Best Params: {'filters': 32, 'kernel_size': 3, 'dropout': 0.5, 'epochs': 30,
'batch_size': 64}
Best Binary Accuracy: 0.8951
Best Normal Accuracy: 0.1985
Best F1 Score: 0.9437
```

[ ]:

```
# CNN - continued with 5 epochs

# 2. CNN

from tensorflow.keras.models import Sequential
from tensorflow.keras.layers import Conv1D, MaxPooling1D, Dense, Flatten,␣
 ↪Dropout
from sklearn.metrics import f1_score, accuracy_score
import numpy as np

# Define the CNN model
def build_cnn_model(filters=64, kernel_size=3, dropout=0.5):
    model = Sequential()

    # 1D Convolutional Layer
    model.add(Conv1D(filters=filters, kernel_size=kernel_size,␣
 ↪activation='relu', input_shape=(X_train.shape[1], 1)))

    # MaxPooling Layer
    model.add(MaxPooling1D(pool_size=2))

    # Dropout Layer
    model.add(Dropout(dropout))

    # Flatten the output for dense layers
    model.add(Flatten())
```

```
    # Dense Layers
    model.add(Dense(256, activation='relu'))
    model.add(Dropout(dropout))

    # Output layer for multi-label classification
    model.add(Dense(y_train.shape[1], activation='sigmoid'))

    # Compile the model
    model.compile(optimizer='adam', loss='binary_crossentropy',␣
↪metrics=['binary_accuracy'])

    return model

# Reshape your data for CNN input
X_train = X_train.reshape((X_train.shape[0], X_train.shape[1], 1))
X_test = X_test.reshape((X_test.shape[0], X_test.shape[1], 1))

# Hyperparameters grid
param_grid = {
    'filters': [32, 64],  # Number of filters for Conv1D
    'kernel_size': [3],  # Kernel size for Conv1D
    'dropout': [0.3, 0.5],  # Dropout rate
    'epochs': [5],  # Number of epochs
    'batch_size': [32,64],  # Batch size
}

# Variables to track the best model
best_binary_acc = 0
best_params = {}
best_f1 = 0
best_normal_acc = 0

# Perform the manual hyperparameter search
for filters in param_grid['filters']:
    for kernel_size in param_grid['kernel_size']:  # Now this will work
        for dropout in param_grid['dropout']:
            for epochs in param_grid['epochs']:
                for batch_size in param_grid['batch_size']:
                    print(f'Training with filters={filters},␣
↪kernel_size={kernel_size}, dropout={dropout}, epochs={epochs},␣
↪batch_size={batch_size}')

                    # Build the model
                    model = build_cnn_model(filters=filters,␣
↪kernel_size=kernel_size, dropout=dropout)
```

```
                    # Train the model
                    model.fit(X_train, y_train, epochs=epochs,␣
↪batch_size=batch_size, validation_split=0.1, verbose=1)

                    # Predict on the test set
                    y_pred = model.predict(X_test)
                    y_pred_binary = (y_pred > 0.5).astype(int)

                    # Calculate binary accuracy
                    binary_acc = np.mean(np.equal(y_test, y_pred_binary).
↪astype(int))

                    # Calculate normal accuracy
                    normal_acc = accuracy_score(y_test, y_pred_binary)

                    # Calculate F1 Score
                    f1 = f1_score(y_test, y_pred_binary, average='micro')

                    # Update best params if this model is better
                    if binary_acc > best_binary_acc:
                        best_binary_acc = binary_acc
                        best_f1 = f1
                        best_normal_acc = normal_acc
                        best_params = {
                            'filters': filters,
                            'kernel_size': kernel_size,
                            'dropout': dropout,
                            'epochs': epochs,
                            'batch_size': batch_size,
                        }

# Print the best results
print(f'Best Params: {best_params}')
print(f'Best Binary Accuracy: {best_binary_acc:.4f}')
print(f'Best Normal Accuracy: {best_normal_acc:.4f}')
print(f'Best F1 Score: {best_f1:.4f}')
```

```
Training with filters=32, kernel_size=3, dropout=0.3, epochs=5, batch_size=32
Epoch 1/5
30/30 [==============================] - 3s 20ms/step - loss: 0.3456 -
binary_accuracy: 0.8697 - val_loss: 0.2565 - val_binary_accuracy: 0.9128
Epoch 2/5
30/30 [==============================] - 0s 9ms/step - loss: 0.2738 -
binary_accuracy: 0.8953 - val_loss: 0.2487 - val_binary_accuracy: 0.9128
Epoch 3/5
30/30 [==============================] - 0s 10ms/step - loss: 0.2494 -
binary_accuracy: 0.9017 - val_loss: 0.2486 - val_binary_accuracy: 0.9062
```

```
Epoch 4/5
30/30 [==============================] - 0s 9ms/step - loss: 0.2270 -
binary_accuracy: 0.9074 - val_loss: 0.2492 - val_binary_accuracy: 0.9077
Epoch 5/5
30/30 [==============================] - 0s 9ms/step - loss: 0.2098 -
binary_accuracy: 0.9133 - val_loss: 0.2510 - val_binary_accuracy: 0.9030
9/9 [==============================] - 0s 3ms/step
Training with filters=32, kernel_size=3, dropout=0.3, epochs=5, batch_size=64
Epoch 1/5
15/15 [==============================] - 3s 31ms/step - loss: 0.3908 -
binary_accuracy: 0.8576 - val_loss: 0.2667 - val_binary_accuracy: 0.9069
Epoch 2/5
15/15 [==============================] - 0s 9ms/step - loss: 0.2963 -
binary_accuracy: 0.8896 - val_loss: 0.2575 - val_binary_accuracy: 0.9124
Epoch 3/5
15/15 [==============================] - 0s 9ms/step - loss: 0.2672 -
binary_accuracy: 0.8984 - val_loss: 0.2452 - val_binary_accuracy: 0.9143
Epoch 4/5
15/15 [==============================] - 0s 8ms/step - loss: 0.2487 -
binary_accuracy: 0.9004 - val_loss: 0.2445 - val_binary_accuracy: 0.9124
Epoch 5/5
15/15 [==============================] - 0s 10ms/step - loss: 0.2330 -
binary_accuracy: 0.9057 - val_loss: 0.2451 - val_binary_accuracy: 0.9058
9/9 [==============================] - 0s 2ms/step
Training with filters=32, kernel_size=3, dropout=0.5, epochs=5, batch_size=32
Epoch 1/5
30/30 [==============================] - 2s 12ms/step - loss: 0.3669 -
binary_accuracy: 0.8662 - val_loss: 0.2622 - val_binary_accuracy: 0.9120
Epoch 2/5
30/30 [==============================] - 0s 7ms/step - loss: 0.2891 -
binary_accuracy: 0.8934 - val_loss: 0.2489 - val_binary_accuracy: 0.9097
Epoch 3/5
30/30 [==============================] - 0s 6ms/step - loss: 0.2588 -
binary_accuracy: 0.8974 - val_loss: 0.2403 - val_binary_accuracy: 0.9108
Epoch 4/5
30/30 [==============================] - 0s 6ms/step - loss: 0.2435 -
binary_accuracy: 0.9016 - val_loss: 0.2427 - val_binary_accuracy: 0.9104
Epoch 5/5
30/30 [==============================] - 0s 7ms/step - loss: 0.2283 -
binary_accuracy: 0.9045 - val_loss: 0.2424 - val_binary_accuracy: 0.9085
9/9 [==============================] - 0s 2ms/step
Training with filters=32, kernel_size=3, dropout=0.5, epochs=5, batch_size=64
Epoch 1/5
15/15 [==============================] - 2s 32ms/step - loss: 0.4199 -
binary_accuracy: 0.8422 - val_loss: 0.2619 - val_binary_accuracy: 0.9124
Epoch 2/5
15/15 [==============================] - 0s 13ms/step - loss: 0.3219 -
binary_accuracy: 0.8793 - val_loss: 0.2649 - val_binary_accuracy: 0.9120
```

```
Epoch 3/5
15/15 [==============================] - 0s 12ms/step - loss: 0.2861 -
binary_accuracy: 0.8916 - val_loss: 0.2528 - val_binary_accuracy: 0.9112
Epoch 4/5
15/15 [==============================] - 0s 13ms/step - loss: 0.2667 -
binary_accuracy: 0.8968 - val_loss: 0.2469 - val_binary_accuracy: 0.9128
Epoch 5/5
15/15 [==============================] - 0s 9ms/step - loss: 0.2519 -
binary_accuracy: 0.9000 - val_loss: 0.2432 - val_binary_accuracy: 0.9120
9/9 [==============================] - 0s 2ms/step
Training with filters=64, kernel_size=3, dropout=0.3, epochs=5, batch_size=32
Epoch 1/5
30/30 [==============================] - 2s 14ms/step - loss: 0.3512 -
binary_accuracy: 0.8702 - val_loss: 0.2626 - val_binary_accuracy: 0.9100
Epoch 2/5
30/30 [==============================] - 0s 9ms/step - loss: 0.2676 -
binary_accuracy: 0.8971 - val_loss: 0.2475 - val_binary_accuracy: 0.9116
Epoch 3/5
30/30 [==============================] - 0s 9ms/step - loss: 0.2401 -
binary_accuracy: 0.9027 - val_loss: 0.2497 - val_binary_accuracy: 0.9026
Epoch 4/5
30/30 [==============================] - 0s 8ms/step - loss: 0.2175 -
binary_accuracy: 0.9099 - val_loss: 0.2511 - val_binary_accuracy: 0.9073
Epoch 5/5
30/30 [==============================] - 0s 8ms/step - loss: 0.1990 -
binary_accuracy: 0.9151 - val_loss: 0.2564 - val_binary_accuracy: 0.9007
9/9 [==============================] - 0s 2ms/step
Training with filters=64, kernel_size=3, dropout=0.3, epochs=5, batch_size=64
Epoch 1/5
15/15 [==============================] - 2s 24ms/step - loss: 0.3853 -
binary_accuracy: 0.8499 - val_loss: 0.2649 - val_binary_accuracy: 0.9124
Epoch 2/5
15/15 [==============================] - 0s 11ms/step - loss: 0.2935 -
binary_accuracy: 0.8895 - val_loss: 0.2592 - val_binary_accuracy: 0.9104
Epoch 3/5
15/15 [==============================] - 0s 12ms/step - loss: 0.2674 -
binary_accuracy: 0.8960 - val_loss: 0.2457 - val_binary_accuracy: 0.9104
Epoch 4/5
15/15 [==============================] - 0s 11ms/step - loss: 0.2452 -
binary_accuracy: 0.9034 - val_loss: 0.2419 - val_binary_accuracy: 0.9136
Epoch 5/5
15/15 [==============================] - 0s 11ms/step - loss: 0.2263 -
binary_accuracy: 0.9089 - val_loss: 0.2414 - val_binary_accuracy: 0.9139
9/9 [==============================] - 0s 2ms/step
Training with filters=64, kernel_size=3, dropout=0.5, epochs=5, batch_size=32
Epoch 1/5
30/30 [==============================] - 2s 14ms/step - loss: 0.3740 -
binary_accuracy: 0.8560 - val_loss: 0.2687 - val_binary_accuracy: 0.9128
```

```
Epoch 2/5
30/30 [==============================] - 0s 8ms/step - loss: 0.2837 -
binary_accuracy: 0.8924 - val_loss: 0.2468 - val_binary_accuracy: 0.9128
Epoch 3/5
30/30 [==============================] - 0s 8ms/step - loss: 0.2536 -
binary_accuracy: 0.9001 - val_loss: 0.2447 - val_binary_accuracy: 0.9116
Epoch 4/5
30/30 [==============================] - 0s 8ms/step - loss: 0.2340 -
binary_accuracy: 0.9049 - val_loss: 0.2455 - val_binary_accuracy: 0.9112
Epoch 5/5
30/30 [==============================] - 0s 11ms/step - loss: 0.2198 -
binary_accuracy: 0.9064 - val_loss: 0.2514 - val_binary_accuracy: 0.9050
9/9 [==============================] - 0s 3ms/step
Training with filters=64, kernel_size=3, dropout=0.5, epochs=5, batch_size=64
Epoch 1/5
15/15 [==============================] - 2s 23ms/step - loss: 0.3877 -
binary_accuracy: 0.8522 - val_loss: 0.2601 - val_binary_accuracy: 0.9120
Epoch 2/5
15/15 [==============================] - 0s 12ms/step - loss: 0.3057 -
binary_accuracy: 0.8858 - val_loss: 0.2572 - val_binary_accuracy: 0.9124
Epoch 3/5
15/15 [==============================] - 0s 11ms/step - loss: 0.2774 -
binary_accuracy: 0.8944 - val_loss: 0.2469 - val_binary_accuracy: 0.9136
Epoch 4/5
15/15 [==============================] - 0s 12ms/step - loss: 0.2545 -
binary_accuracy: 0.8991 - val_loss: 0.2435 - val_binary_accuracy: 0.9116
Epoch 5/5
15/15 [==============================] - 0s 11ms/step - loss: 0.2332 -
binary_accuracy: 0.9051 - val_loss: 0.2429 - val_binary_accuracy: 0.9112
9/9 [==============================] - 0s 2ms/step
Best Params: {'filters': 64, 'kernel_size': 3, 'dropout': 0.3, 'epochs': 5,
'batch_size': 64}
Best Binary Accuracy: 0.9114
Best Normal Accuracy: 0.2772
Best F1 Score: 0.9535
```

[ ]:
```python
# LTSM with parameter tuning

from tensorflow.keras.models import Sequential
from tensorflow.keras.layers import LSTM, Dense, Dropout, Flatten,␣
↪BatchNormalization
from tensorflow.keras.regularizers import l2
from sklearn.metrics import f1_score, accuracy_score
import numpy as np

# Define the LSTM model with L2 Regularization and Batch Normalization
def build_lstm_model(units=100, dropout=0.5, l2_reg=0.01):
```

```
    model = Sequential()

    # LSTM Layer
    model.add(LSTM(units, return_sequences=False, input_shape=(X_train.
 ↪shape[1], 1)))

    # Dropout Layer
    model.add(Dropout(dropout))

    # Batch Normalization
    model.add(BatchNormalization())

    # Dense Layer with L2 regularization
    model.add(Dense(256, activation='relu', kernel_regularizer=l2(l2_reg)))

    # Output layer for multi-label classification
    model.add(Dense(y_train.shape[1], activation='sigmoid'))

    # Compile the model
    model.compile(optimizer='adam', loss='binary_crossentropy',␣
 ↪metrics=['binary_accuracy'])

    return model

# Reshape your data for LSTM input
X_train = X_train.reshape((X_train.shape[0], X_train.shape[1], 1))
X_test = X_test.reshape((X_test.shape[0], X_test.shape[1], 1))

# Hyperparameters grid
param_grid = {
    'units': [64, 128],  # Number of LSTM units
    'dropout': [0.3,0.5],  # Dropout rate
    'l2_reg': [0.01, 0.001],  # L2 regularization
    'epochs': [10, 15, 20, 30],  # Multiple numbers of epochs
    'batch_size': [32, 64],  # Batch size
}

# Variables to track the best model
best_binary_acc = 0
best_params = {}
best_f1 = 0
best_normal_acc = 0

# Perform the manual hyperparameter search
for units in param_grid['units']:
    for dropout in param_grid['dropout']:
        for l2_reg in param_grid['l2_reg']:
```

```
            for epochs in param_grid['epochs']:
                for batch_size in param_grid['batch_size']:
                    print(f'Training with units={units}, dropout={dropout},␣
↪l2_reg={l2_reg}, epochs={epochs}, batch_size={batch_size}')

                    # Build the model
                    model = build_lstm_model(units=units, dropout=dropout,␣
↪l2_reg=l2_reg)

                    # Train the model
                    model.fit(X_train, y_train, epochs=epochs,␣
↪batch_size=batch_size, validation_split=0.1, verbose=0)

                    # Predict on the test set
                    y_pred = model.predict(X_test)
                    y_pred_binary = (y_pred > 0.5).astype(int)

                    # Calculate binary accuracy
                    binary_acc = np.mean(np.equal(y_test, y_pred_binary).
↪astype(int))

                    # Calculate normal accuracy
                    normal_acc = accuracy_score(y_test, y_pred_binary)

                    # Calculate F1 Score
                    f1 = f1_score(y_test, y_pred_binary, average='micro')

                    # Update best params if this model is better
                    if binary_acc > best_binary_acc:
                        best_binary_acc = binary_acc
                        best_f1 = f1
                        best_normal_acc = normal_acc
                        best_params = {
                            'units': units,
                            'dropout': dropout,
                            'l2_reg': l2_reg,
                            'epochs': epochs,
                            'batch_size': batch_size,
                        }

# Print the best results
print(f'Best Params: {best_params}')
print(f'Best Binary Accuracy: {best_binary_acc:.4f}')
print(f'Best Normal Accuracy: {best_normal_acc:.4f}')
print(f'Best F1 Score: {best_f1:.4f}')
```

```
Training with units=64, dropout=0.3, l2_reg=0.01, epochs=10, batch_size=32
```

```
/usr/local/lib/python3.10/dist-packages/keras/src/layers/rnn/rnn.py:204:
UserWarning: Do not pass an `input_shape`/`input_dim` argument to a layer. When
using Sequential models, prefer using an `Input(shape)` object as the first
layer in the model instead.
  super().__init__(**kwargs)

9/9           0s 25ms/step
Training with units=64, dropout=0.3, l2_reg=0.01, epochs=10, batch_size=64

/usr/local/lib/python3.10/dist-packages/keras/src/layers/rnn/rnn.py:204:
UserWarning: Do not pass an `input_shape`/`input_dim` argument to a layer. When
using Sequential models, prefer using an `Input(shape)` object as the first
layer in the model instead.
  super().__init__(**kwargs)

9/9           0s 26ms/step
Training with units=64, dropout=0.3, l2_reg=0.01, epochs=15, batch_size=32

/usr/local/lib/python3.10/dist-packages/keras/src/layers/rnn/rnn.py:204:
UserWarning: Do not pass an `input_shape`/`input_dim` argument to a layer. When
using Sequential models, prefer using an `Input(shape)` object as the first
layer in the model instead.
  super().__init__(**kwargs)

9/9           0s 25ms/step
Training with units=64, dropout=0.3, l2_reg=0.01, epochs=15, batch_size=64

/usr/local/lib/python3.10/dist-packages/keras/src/layers/rnn/rnn.py:204:
UserWarning: Do not pass an `input_shape`/`input_dim` argument to a layer. When
using Sequential models, prefer using an `Input(shape)` object as the first
layer in the model instead.
  super().__init__(**kwargs)

9/9           0s 25ms/step
Training with units=64, dropout=0.3, l2_reg=0.01, epochs=20, batch_size=32

/usr/local/lib/python3.10/dist-packages/keras/src/layers/rnn/rnn.py:204:
UserWarning: Do not pass an `input_shape`/`input_dim` argument to a layer. When
using Sequential models, prefer using an `Input(shape)` object as the first
layer in the model instead.
  super().__init__(**kwargs)

9/9           0s 27ms/step
Training with units=64, dropout=0.3, l2_reg=0.01, epochs=20, batch_size=64

/usr/local/lib/python3.10/dist-packages/keras/src/layers/rnn/rnn.py:204:
UserWarning: Do not pass an `input_shape`/`input_dim` argument to a layer. When
using Sequential models, prefer using an `Input(shape)` object as the first
layer in the model instead.
  super().__init__(**kwargs)

9/9           0s 36ms/step
Training with units=64, dropout=0.3, l2_reg=0.01, epochs=30, batch_size=32
```

```
/usr/local/lib/python3.10/dist-packages/keras/src/layers/rnn/rnn.py:204:
UserWarning: Do not pass an `input_shape`/`input_dim` argument to a layer. When
using Sequential models, prefer using an `Input(shape)` object as the first
layer in the model instead.
  super().__init__(**kwargs)

9/9           0s 38ms/step
Training with units=64, dropout=0.3, l2_reg=0.01, epochs=30, batch_size=64

/usr/local/lib/python3.10/dist-packages/keras/src/layers/rnn/rnn.py:204:
UserWarning: Do not pass an `input_shape`/`input_dim` argument to a layer. When
using Sequential models, prefer using an `Input(shape)` object as the first
layer in the model instead.
  super().__init__(**kwargs)

9/9           0s 26ms/step
Training with units=64, dropout=0.3, l2_reg=0.001, epochs=10, batch_size=32

/usr/local/lib/python3.10/dist-packages/keras/src/layers/rnn/rnn.py:204:
UserWarning: Do not pass an `input_shape`/`input_dim` argument to a layer. When
using Sequential models, prefer using an `Input(shape)` object as the first
layer in the model instead.
  super().__init__(**kwargs)

9/9           0s 25ms/step
Training with units=64, dropout=0.3, l2_reg=0.001, epochs=10, batch_size=64

/usr/local/lib/python3.10/dist-packages/keras/src/layers/rnn/rnn.py:204:
UserWarning: Do not pass an `input_shape`/`input_dim` argument to a layer. When
using Sequential models, prefer using an `Input(shape)` object as the first
layer in the model instead.
  super().__init__(**kwargs)

9/9           0s 26ms/step
Training with units=64, dropout=0.3, l2_reg=0.001, epochs=15, batch_size=32

/usr/local/lib/python3.10/dist-packages/keras/src/layers/rnn/rnn.py:204:
UserWarning: Do not pass an `input_shape`/`input_dim` argument to a layer. When
using Sequential models, prefer using an `Input(shape)` object as the first
layer in the model instead.
  super().__init__(**kwargs)

9/9           0s 33ms/step
Training with units=64, dropout=0.3, l2_reg=0.001, epochs=15, batch_size=64

/usr/local/lib/python3.10/dist-packages/keras/src/layers/rnn/rnn.py:204:
UserWarning: Do not pass an `input_shape`/`input_dim` argument to a layer. When
using Sequential models, prefer using an `Input(shape)` object as the first
layer in the model instead.
  super().__init__(**kwargs)

9/9           1s 39ms/step
Training with units=64, dropout=0.3, l2_reg=0.001, epochs=20, batch_size=32
```

```
/usr/local/lib/python3.10/dist-packages/keras/src/layers/rnn/rnn.py:204:
UserWarning: Do not pass an `input_shape`/`input_dim` argument to a layer. When
using Sequential models, prefer using an `Input(shape)` object as the first
layer in the model instead.
  super().__init__(**kwargs)

9/9            0s 25ms/step
Training with units=64, dropout=0.3, l2_reg=0.001, epochs=20, batch_size=64

/usr/local/lib/python3.10/dist-packages/keras/src/layers/rnn/rnn.py:204:
UserWarning: Do not pass an `input_shape`/`input_dim` argument to a layer. When
using Sequential models, prefer using an `Input(shape)` object as the first
layer in the model instead.
  super().__init__(**kwargs)

9/9            1s 40ms/step
Training with units=64, dropout=0.3, l2_reg=0.001, epochs=30, batch_size=32

/usr/local/lib/python3.10/dist-packages/keras/src/layers/rnn/rnn.py:204:
UserWarning: Do not pass an `input_shape`/`input_dim` argument to a layer. When
using Sequential models, prefer using an `Input(shape)` object as the first
layer in the model instead.
  super().__init__(**kwargs)

9/9            0s 25ms/step
Training with units=64, dropout=0.3, l2_reg=0.001, epochs=30, batch_size=64

/usr/local/lib/python3.10/dist-packages/keras/src/layers/rnn/rnn.py:204:
UserWarning: Do not pass an `input_shape`/`input_dim` argument to a layer. When
using Sequential models, prefer using an `Input(shape)` object as the first
layer in the model instead.
  super().__init__(**kwargs)

9/9            0s 25ms/step
Training with units=64, dropout=0.5, l2_reg=0.01, epochs=10, batch_size=32

/usr/local/lib/python3.10/dist-packages/keras/src/layers/rnn/rnn.py:204:
UserWarning: Do not pass an `input_shape`/`input_dim` argument to a layer. When
using Sequential models, prefer using an `Input(shape)` object as the first
layer in the model instead.
  super().__init__(**kwargs)

9/9            0s 25ms/step
Training with units=64, dropout=0.5, l2_reg=0.01, epochs=10, batch_size=64

/usr/local/lib/python3.10/dist-packages/keras/src/layers/rnn/rnn.py:204:
UserWarning: Do not pass an `input_shape`/`input_dim` argument to a layer. When
using Sequential models, prefer using an `Input(shape)` object as the first
layer in the model instead.
  super().__init__(**kwargs)

9/9            0s 25ms/step
Training with units=64, dropout=0.5, l2_reg=0.01, epochs=15, batch_size=32
```

```
/usr/local/lib/python3.10/dist-packages/keras/src/layers/rnn/rnn.py:204:
UserWarning: Do not pass an `input_shape`/`input_dim` argument to a layer. When
using Sequential models, prefer using an `Input(shape)` object as the first
layer in the model instead.
  super().__init__(**kwargs)

9/9            0s 26ms/step
Training with units=64, dropout=0.5, l2_reg=0.01, epochs=15, batch_size=64

/usr/local/lib/python3.10/dist-packages/keras/src/layers/rnn/rnn.py:204:
UserWarning: Do not pass an `input_shape`/`input_dim` argument to a layer. When
using Sequential models, prefer using an `Input(shape)` object as the first
layer in the model instead.
  super().__init__(**kwargs)

9/9            0s 37ms/step
Training with units=64, dropout=0.5, l2_reg=0.01, epochs=20, batch_size=32

/usr/local/lib/python3.10/dist-packages/keras/src/layers/rnn/rnn.py:204:
UserWarning: Do not pass an `input_shape`/`input_dim` argument to a layer. When
using Sequential models, prefer using an `Input(shape)` object as the first
layer in the model instead.
  super().__init__(**kwargs)

9/9            0s 39ms/step
Training with units=64, dropout=0.5, l2_reg=0.01, epochs=20, batch_size=64

/usr/local/lib/python3.10/dist-packages/keras/src/layers/rnn/rnn.py:204:
UserWarning: Do not pass an `input_shape`/`input_dim` argument to a layer. When
using Sequential models, prefer using an `Input(shape)` object as the first
layer in the model instead.
  super().__init__(**kwargs)

9/9            0s 26ms/step
Training with units=64, dropout=0.5, l2_reg=0.01, epochs=30, batch_size=32

/usr/local/lib/python3.10/dist-packages/keras/src/layers/rnn/rnn.py:204:
UserWarning: Do not pass an `input_shape`/`input_dim` argument to a layer. When
using Sequential models, prefer using an `Input(shape)` object as the first
layer in the model instead.
  super().__init__(**kwargs)

9/9            0s 37ms/step
Training with units=64, dropout=0.5, l2_reg=0.01, epochs=30, batch_size=64

/usr/local/lib/python3.10/dist-packages/keras/src/layers/rnn/rnn.py:204:
UserWarning: Do not pass an `input_shape`/`input_dim` argument to a layer. When
using Sequential models, prefer using an `Input(shape)` object as the first
layer in the model instead.
  super().__init__(**kwargs)

9/9            0s 25ms/step
Training with units=64, dropout=0.5, l2_reg=0.001, epochs=10, batch_size=32
```

```
/usr/local/lib/python3.10/dist-packages/keras/src/layers/rnn/rnn.py:204:
UserWarning: Do not pass an `input_shape`/`input_dim` argument to a layer. When
using Sequential models, prefer using an `Input(shape)` object as the first
layer in the model instead.
  super().__init__(**kwargs)

9/9           0s 25ms/step
Training with units=64, dropout=0.5, l2_reg=0.001, epochs=10, batch_size=64

/usr/local/lib/python3.10/dist-packages/keras/src/layers/rnn/rnn.py:204:
UserWarning: Do not pass an `input_shape`/`input_dim` argument to a layer. When
using Sequential models, prefer using an `Input(shape)` object as the first
layer in the model instead.
  super().__init__(**kwargs)

9/9           0s 26ms/step
Training with units=64, dropout=0.5, l2_reg=0.001, epochs=15, batch_size=32

/usr/local/lib/python3.10/dist-packages/keras/src/layers/rnn/rnn.py:204:
UserWarning: Do not pass an `input_shape`/`input_dim` argument to a layer. When
using Sequential models, prefer using an `Input(shape)` object as the first
layer in the model instead.
  super().__init__(**kwargs)

9/9           0s 25ms/step
Training with units=64, dropout=0.5, l2_reg=0.001, epochs=15, batch_size=64

/usr/local/lib/python3.10/dist-packages/keras/src/layers/rnn/rnn.py:204:
UserWarning: Do not pass an `input_shape`/`input_dim` argument to a layer. When
using Sequential models, prefer using an `Input(shape)` object as the first
layer in the model instead.
  super().__init__(**kwargs)

9/9           0s 26ms/step
Training with units=64, dropout=0.5, l2_reg=0.001, epochs=20, batch_size=32

/usr/local/lib/python3.10/dist-packages/keras/src/layers/rnn/rnn.py:204:
UserWarning: Do not pass an `input_shape`/`input_dim` argument to a layer. When
using Sequential models, prefer using an `Input(shape)` object as the first
layer in the model instead.
  super().__init__(**kwargs)

9/9           0s 27ms/step
Training with units=64, dropout=0.5, l2_reg=0.001, epochs=20, batch_size=64

/usr/local/lib/python3.10/dist-packages/keras/src/layers/rnn/rnn.py:204:
UserWarning: Do not pass an `input_shape`/`input_dim` argument to a layer. When
using Sequential models, prefer using an `Input(shape)` object as the first
layer in the model instead.
  super().__init__(**kwargs)

9/9           0s 25ms/step
Training with units=64, dropout=0.5, l2_reg=0.001, epochs=30, batch_size=32
```

```
/usr/local/lib/python3.10/dist-packages/keras/src/layers/rnn/rnn.py:204:
UserWarning: Do not pass an `input_shape`/`input_dim` argument to a layer. When
using Sequential models, prefer using an `Input(shape)` object as the first
layer in the model instead.
  super().__init__(**kwargs)

9/9            0s 25ms/step
Training with units=64, dropout=0.5, l2_reg=0.001, epochs=30, batch_size=64

/usr/local/lib/python3.10/dist-packages/keras/src/layers/rnn/rnn.py:204:
UserWarning: Do not pass an `input_shape`/`input_dim` argument to a layer. When
using Sequential models, prefer using an `Input(shape)` object as the first
layer in the model instead.
  super().__init__(**kwargs)

9/9            0s 30ms/step
Training with units=128, dropout=0.3, l2_reg=0.01, epochs=10, batch_size=32

/usr/local/lib/python3.10/dist-packages/keras/src/layers/rnn/rnn.py:204:
UserWarning: Do not pass an `input_shape`/`input_dim` argument to a layer. When
using Sequential models, prefer using an `Input(shape)` object as the first
layer in the model instead.
  super().__init__(**kwargs)

9/9            0s 32ms/step
Training with units=128, dropout=0.3, l2_reg=0.01, epochs=10, batch_size=64

/usr/local/lib/python3.10/dist-packages/keras/src/layers/rnn/rnn.py:204:
UserWarning: Do not pass an `input_shape`/`input_dim` argument to a layer. When
using Sequential models, prefer using an `Input(shape)` object as the first
layer in the model instead.
  super().__init__(**kwargs)

9/9            0s 25ms/step
Training with units=128, dropout=0.3, l2_reg=0.01, epochs=15, batch_size=32

/usr/local/lib/python3.10/dist-packages/keras/src/layers/rnn/rnn.py:204:
UserWarning: Do not pass an `input_shape`/`input_dim` argument to a layer. When
using Sequential models, prefer using an `Input(shape)` object as the first
layer in the model instead.
  super().__init__(**kwargs)

9/9            0s 25ms/step
Training with units=128, dropout=0.3, l2_reg=0.01, epochs=15, batch_size=64

/usr/local/lib/python3.10/dist-packages/keras/src/layers/rnn/rnn.py:204:
UserWarning: Do not pass an `input_shape`/`input_dim` argument to a layer. When
using Sequential models, prefer using an `Input(shape)` object as the first
layer in the model instead.
  super().__init__(**kwargs)

9/9            0s 25ms/step
Training with units=128, dropout=0.3, l2_reg=0.01, epochs=20, batch_size=32
```

```
/usr/local/lib/python3.10/dist-packages/keras/src/layers/rnn/rnn.py:204:
UserWarning: Do not pass an `input_shape`/`input_dim` argument to a layer. When
using Sequential models, prefer using an `Input(shape)` object as the first
layer in the model instead.
  super().__init__(**kwargs)

9/9           0s 25ms/step
Training with units=128, dropout=0.3, l2_reg=0.01, epochs=20, batch_size=64

/usr/local/lib/python3.10/dist-packages/keras/src/layers/rnn/rnn.py:204:
UserWarning: Do not pass an `input_shape`/`input_dim` argument to a layer. When
using Sequential models, prefer using an `Input(shape)` object as the first
layer in the model instead.
  super().__init__(**kwargs)

9/9           0s 24ms/step
Training with units=128, dropout=0.3, l2_reg=0.01, epochs=30, batch_size=32

/usr/local/lib/python3.10/dist-packages/keras/src/layers/rnn/rnn.py:204:
UserWarning: Do not pass an `input_shape`/`input_dim` argument to a layer. When
using Sequential models, prefer using an `Input(shape)` object as the first
layer in the model instead.
  super().__init__(**kwargs)

9/9           1s 41ms/step
Training with units=128, dropout=0.3, l2_reg=0.01, epochs=30, batch_size=64

/usr/local/lib/python3.10/dist-packages/keras/src/layers/rnn/rnn.py:204:
UserWarning: Do not pass an `input_shape`/`input_dim` argument to a layer. When
using Sequential models, prefer using an `Input(shape)` object as the first
layer in the model instead.
  super().__init__(**kwargs)

9/9           0s 25ms/step
Training with units=128, dropout=0.3, l2_reg=0.001, epochs=10, batch_size=32

/usr/local/lib/python3.10/dist-packages/keras/src/layers/rnn/rnn.py:204:
UserWarning: Do not pass an `input_shape`/`input_dim` argument to a layer. When
using Sequential models, prefer using an `Input(shape)` object as the first
layer in the model instead.
  super().__init__(**kwargs)

9/9           1s 39ms/step
Training with units=128, dropout=0.3, l2_reg=0.001, epochs=10, batch_size=64

/usr/local/lib/python3.10/dist-packages/keras/src/layers/rnn/rnn.py:204:
UserWarning: Do not pass an `input_shape`/`input_dim` argument to a layer. When
using Sequential models, prefer using an `Input(shape)` object as the first
layer in the model instead.
  super().__init__(**kwargs)

9/9           0s 27ms/step
Training with units=128, dropout=0.3, l2_reg=0.001, epochs=15, batch_size=32
```

```
/usr/local/lib/python3.10/dist-packages/keras/src/layers/rnn/rnn.py:204:
UserWarning: Do not pass an `input_shape`/`input_dim` argument to a layer. When
using Sequential models, prefer using an `Input(shape)` object as the first
layer in the model instead.
  super().__init__(**kwargs)

9/9           0s 25ms/step
Training with units=128, dropout=0.3, l2_reg=0.001, epochs=15, batch_size=64

/usr/local/lib/python3.10/dist-packages/keras/src/layers/rnn/rnn.py:204:
UserWarning: Do not pass an `input_shape`/`input_dim` argument to a layer. When
using Sequential models, prefer using an `Input(shape)` object as the first
layer in the model instead.
  super().__init__(**kwargs)

9/9           0s 25ms/step
Training with units=128, dropout=0.3, l2_reg=0.001, epochs=20, batch_size=32

/usr/local/lib/python3.10/dist-packages/keras/src/layers/rnn/rnn.py:204:
UserWarning: Do not pass an `input_shape`/`input_dim` argument to a layer. When
using Sequential models, prefer using an `Input(shape)` object as the first
layer in the model instead.
  super().__init__(**kwargs)

9/9           0s 25ms/step
Training with units=128, dropout=0.3, l2_reg=0.001, epochs=20, batch_size=64

/usr/local/lib/python3.10/dist-packages/keras/src/layers/rnn/rnn.py:204:
UserWarning: Do not pass an `input_shape`/`input_dim` argument to a layer. When
using Sequential models, prefer using an `Input(shape)` object as the first
layer in the model instead.
  super().__init__(**kwargs)

9/9           0s 26ms/step
Training with units=128, dropout=0.3, l2_reg=0.001, epochs=30, batch_size=32

/usr/local/lib/python3.10/dist-packages/keras/src/layers/rnn/rnn.py:204:
UserWarning: Do not pass an `input_shape`/`input_dim` argument to a layer. When
using Sequential models, prefer using an `Input(shape)` object as the first
layer in the model instead.
  super().__init__(**kwargs)

9/9           0s 25ms/step
Training with units=128, dropout=0.3, l2_reg=0.001, epochs=30, batch_size=64

/usr/local/lib/python3.10/dist-packages/keras/src/layers/rnn/rnn.py:204:
UserWarning: Do not pass an `input_shape`/`input_dim` argument to a layer. When
using Sequential models, prefer using an `Input(shape)` object as the first
layer in the model instead.
  super().__init__(**kwargs)

9/9           0s 24ms/step
Training with units=128, dropout=0.5, l2_reg=0.01, epochs=10, batch_size=32
```

```
/usr/local/lib/python3.10/dist-packages/keras/src/layers/rnn/rnn.py:204:
UserWarning: Do not pass an `input_shape`/`input_dim` argument to a layer. When
using Sequential models, prefer using an `Input(shape)` object as the first
layer in the model instead.
  super().__init__(**kwargs)

9/9           0s 25ms/step
Training with units=128, dropout=0.5, l2_reg=0.01, epochs=10, batch_size=64

/usr/local/lib/python3.10/dist-packages/keras/src/layers/rnn/rnn.py:204:
UserWarning: Do not pass an `input_shape`/`input_dim` argument to a layer. When
using Sequential models, prefer using an `Input(shape)` object as the first
layer in the model instead.
  super().__init__(**kwargs)

9/9           0s 25ms/step
Training with units=128, dropout=0.5, l2_reg=0.01, epochs=15, batch_size=32

/usr/local/lib/python3.10/dist-packages/keras/src/layers/rnn/rnn.py:204:
UserWarning: Do not pass an `input_shape`/`input_dim` argument to a layer. When
using Sequential models, prefer using an `Input(shape)` object as the first
layer in the model instead.
  super().__init__(**kwargs)

9/9           0s 37ms/step
Training with units=128, dropout=0.5, l2_reg=0.01, epochs=15, batch_size=64

/usr/local/lib/python3.10/dist-packages/keras/src/layers/rnn/rnn.py:204:
UserWarning: Do not pass an `input_shape`/`input_dim` argument to a layer. When
using Sequential models, prefer using an `Input(shape)` object as the first
layer in the model instead.
  super().__init__(**kwargs)

9/9           0s 31ms/step
Training with units=128, dropout=0.5, l2_reg=0.01, epochs=20, batch_size=32

/usr/local/lib/python3.10/dist-packages/keras/src/layers/rnn/rnn.py:204:
UserWarning: Do not pass an `input_shape`/`input_dim` argument to a layer. When
using Sequential models, prefer using an `Input(shape)` object as the first
layer in the model instead.
  super().__init__(**kwargs)

9/9           0s 27ms/step
Training with units=128, dropout=0.5, l2_reg=0.01, epochs=20, batch_size=64

/usr/local/lib/python3.10/dist-packages/keras/src/layers/rnn/rnn.py:204:
UserWarning: Do not pass an `input_shape`/`input_dim` argument to a layer. When
using Sequential models, prefer using an `Input(shape)` object as the first
layer in the model instead.
  super().__init__(**kwargs)

9/9           0s 25ms/step
Training with units=128, dropout=0.5, l2_reg=0.01, epochs=30, batch_size=32
```

```
/usr/local/lib/python3.10/dist-packages/keras/src/layers/rnn/rnn.py:204:
UserWarning: Do not pass an `input_shape`/`input_dim` argument to a layer. When
using Sequential models, prefer using an `Input(shape)` object as the first
layer in the model instead.
  super().__init__(**kwargs)

9/9            0s 25ms/step
Training with units=128, dropout=0.5, l2_reg=0.01, epochs=30, batch_size=64

/usr/local/lib/python3.10/dist-packages/keras/src/layers/rnn/rnn.py:204:
UserWarning: Do not pass an `input_shape`/`input_dim` argument to a layer. When
using Sequential models, prefer using an `Input(shape)` object as the first
layer in the model instead.
  super().__init__(**kwargs)

9/9            0s 26ms/step
Training with units=128, dropout=0.5, l2_reg=0.001, epochs=10, batch_size=32

/usr/local/lib/python3.10/dist-packages/keras/src/layers/rnn/rnn.py:204:
UserWarning: Do not pass an `input_shape`/`input_dim` argument to a layer. When
using Sequential models, prefer using an `Input(shape)` object as the first
layer in the model instead.
  super().__init__(**kwargs)

9/9            0s 26ms/step
Training with units=128, dropout=0.5, l2_reg=0.001, epochs=10, batch_size=64

/usr/local/lib/python3.10/dist-packages/keras/src/layers/rnn/rnn.py:204:
UserWarning: Do not pass an `input_shape`/`input_dim` argument to a layer. When
using Sequential models, prefer using an `Input(shape)` object as the first
layer in the model instead.
  super().__init__(**kwargs)

9/9            0s 26ms/step
Training with units=128, dropout=0.5, l2_reg=0.001, epochs=15, batch_size=32

/usr/local/lib/python3.10/dist-packages/keras/src/layers/rnn/rnn.py:204:
UserWarning: Do not pass an `input_shape`/`input_dim` argument to a layer. When
using Sequential models, prefer using an `Input(shape)` object as the first
layer in the model instead.
  super().__init__(**kwargs)

9/9            0s 28ms/step
Training with units=128, dropout=0.5, l2_reg=0.001, epochs=15, batch_size=64

/usr/local/lib/python3.10/dist-packages/keras/src/layers/rnn/rnn.py:204:
UserWarning: Do not pass an `input_shape`/`input_dim` argument to a layer. When
using Sequential models, prefer using an `Input(shape)` object as the first
layer in the model instead.
  super().__init__(**kwargs)

9/9            0s 25ms/step
Training with units=128, dropout=0.5, l2_reg=0.001, epochs=20, batch_size=32
```

```
/usr/local/lib/python3.10/dist-packages/keras/src/layers/rnn/rnn.py:204:
UserWarning: Do not pass an `input_shape`/`input_dim` argument to a layer. When
using Sequential models, prefer using an `Input(shape)` object as the first
layer in the model instead.
  super().__init__(**kwargs)
```

```
9/9              0s 25ms/step
Training with units=128, dropout=0.5, l2_reg=0.001, epochs=20, batch_size=64
```

```
/usr/local/lib/python3.10/dist-packages/keras/src/layers/rnn/rnn.py:204:
UserWarning: Do not pass an `input_shape`/`input_dim` argument to a layer. When
using Sequential models, prefer using an `Input(shape)` object as the first
layer in the model instead.
  super().__init__(**kwargs)
```

```
9/9              0s 25ms/step
Training with units=128, dropout=0.5, l2_reg=0.001, epochs=30, batch_size=32
```

```
/usr/local/lib/python3.10/dist-packages/keras/src/layers/rnn/rnn.py:204:
UserWarning: Do not pass an `input_shape`/`input_dim` argument to a layer. When
using Sequential models, prefer using an `Input(shape)` object as the first
layer in the model instead.
  super().__init__(**kwargs)
```

```
9/9              0s 27ms/step
Training with units=128, dropout=0.5, l2_reg=0.001, epochs=30, batch_size=64
```

```
/usr/local/lib/python3.10/dist-packages/keras/src/layers/rnn/rnn.py:204:
UserWarning: Do not pass an `input_shape`/`input_dim` argument to a layer. When
using Sequential models, prefer using an `Input(shape)` object as the first
layer in the model instead.
  super().__init__(**kwargs)
```

```
9/9              0s 26ms/step
Best Params: {'units': 64, 'dropout': 0.3, 'l2_reg': 0.01, 'epochs': 10,
'batch_size': 32}
Best Binary Accuracy: 0.9114
Best Normal Accuracy: 0.2884
Best F1 Score: 0.9536
```

[ ]:
```python
# 4. TRANSFORMER BASED MODEL
```

[ ]:
```python
# 10 epochs

import numpy as np
from tensorflow.keras.models import Model
from tensorflow.keras.layers import Input, Dense, LayerNormalization, Dropout,
    MultiHeadAttention, Flatten
from sklearn.metrics import f1_score, accuracy_score
from sklearn.model_selection import train_test_split
```

```
# Define Transformer-based model
def build_transformer_model(num_heads=4, hidden_dim=64, dropout=0.3):
    input_layer = Input(shape=(X_train.shape[1], 1))

    # Multi-Head Attention Layer
    attention_output = MultiHeadAttention(num_heads=num_heads,␣
↪key_dim=hidden_dim)(input_layer, input_layer)
    attention_output = LayerNormalization(epsilon=1e-6)(attention_output)

    # Feed-forward Network
    ff_output = Dense(hidden_dim, activation='relu')(attention_output)
    ff_output = Dropout(dropout)(ff_output)
    ff_output = Dense(hidden_dim, activation='relu')(ff_output)
    ff_output = LayerNormalization(epsilon=1e-6)(ff_output)

    # Flatten and Dense layers for output
    flattened = Flatten()(ff_output)
    dense_output = Dense(hidden_dim, activation='relu')(flattened)
    dense_output = Dropout(dropout)(dense_output)

    # Output Layer (sigmoid for multi-label classification)
    output_layer = Dense(y_train.shape[1], activation='sigmoid')(dense_output)

    # Build Model
    model = Model(inputs=input_layer, outputs=output_layer)
    model.compile(optimizer='adam', loss='binary_crossentropy',␣
↪metrics=['binary_accuracy'])

    return model

# Reshape your data for transformer input
X_train = X_train.reshape((X_train.shape[0], X_train.shape[1], 1))
X_test = X_test.reshape((X_test.shape[0], X_test.shape[1], 1))

# Hyperparameters grid
param_grid = {
    'num_heads': [4],         # Number of attention heads
    'hidden_dim': [64, 128],     # Hidden layer dimension
    'dropout': [0.3, 0.5],       # Dropout rate
    'epochs': [10],              # Number of epochs
    'batch_size': [32, 64],      # Batch size
}

# Variables to track the best model
best_binary_acc = 0
best_params = {}
best_f1 = 0
```

```
best_normal_acc = 0

# Perform the manual hyperparameter search
for num_heads in param_grid['num_heads']:
    for hidden_dim in param_grid['hidden_dim']:
        for dropout in param_grid['dropout']:
            for epochs in param_grid['epochs']:
                for batch_size in param_grid['batch_size']:
                    print(f'Training with num_heads={num_heads}, 
 ↪hidden_dim={hidden_dim}, dropout={dropout}, epochs={epochs}, 
 ↪batch_size={batch_size}')

                    # Build the model
                    model = build_transformer_model(num_heads=num_heads, 
 ↪hidden_dim=hidden_dim, dropout=dropout)

                    # Train the model
                    model.fit(X_train, y_train, epochs=epochs, 
 ↪batch_size=batch_size, validation_split=0.1, verbose=0)

                    # Predict on the test set
                    y_pred = model.predict(X_test)
                    y_pred_binary = (y_pred > 0.5).astype(int)

                    # Calculate binary accuracy
                    binary_acc = np.mean(np.equal(y_test, y_pred_binary).
 ↪astype(int))

                    # Calculate normal accuracy
                    normal_acc = accuracy_score(y_test, y_pred_binary)

                    # Calculate F1 Score
                    f1 = f1_score(y_test, y_pred_binary, average='micro')

                    # Update best params if this model is better
                    if binary_acc > best_binary_acc:
                        best_binary_acc = binary_acc
                        best_f1 = f1
                        best_normal_acc = normal_acc
                        best_params = {
                            'num_heads': num_heads,
                            'hidden_dim': hidden_dim,
                            'dropout': dropout,
                            'epochs': epochs,
                            'batch_size': batch_size,
                        }
```

```python
# Print the best results
print(f'Best Params: {best_params}')
print(f'Best Binary Accuracy: {best_binary_acc:.4f}')
print(f'Best Normal Accuracy: {best_normal_acc:.4f}')
print(f'Best F1 Score: {best_f1:.4f}')
```

```
Training with num_heads=4, hidden_dim=64, dropout=0.3, epochs=10, batch_size=32
9/9              1s 83ms/step
Training with num_heads=4, hidden_dim=64, dropout=0.3, epochs=10, batch_size=64
9/9              1s 104ms/step
Training with num_heads=4, hidden_dim=64, dropout=0.5, epochs=10, batch_size=32
9/9              1s 73ms/step
Training with num_heads=4, hidden_dim=64, dropout=0.5, epochs=10, batch_size=64
9/9              1s 100ms/step
Training with num_heads=4, hidden_dim=128, dropout=0.3, epochs=10, batch_size=32
9/9              1s 99ms/step
Training with num_heads=4, hidden_dim=128, dropout=0.3, epochs=10, batch_size=64
9/9              1s 78ms/step
Training with num_heads=4, hidden_dim=128, dropout=0.5, epochs=10, batch_size=32
9/9              1s 78ms/step
Training with num_heads=4, hidden_dim=128, dropout=0.5, epochs=10, batch_size=64
9/9              1s 80ms/step
Best Params: {'num_heads': 4, 'hidden_dim': 64, 'dropout': 0.3, 'epochs': 10,
'batch_size': 32}
Best Binary Accuracy: 0.9114
Best Normal Accuracy: 0.2884
Best F1 Score: 0.9536
```

[ ]:
```python
# num_heads = 8, epochs = 15
```

[ ]:
```python
# 5. GNN
```

[ ]:
```python
!pip install torch-geometric
```

```
Collecting torch-geometric
  Downloading torch_geometric-2.6.0-py3-none-any.whl.metadata (63 kB)
                           63.1/63.1 kB
2.4 MB/s eta 0:00:00
Requirement already satisfied: aiohttp in /usr/local/lib/python3.10/dist-
packages (from torch-geometric) (3.10.5)
Requirement already satisfied: fsspec in /usr/local/lib/python3.10/dist-packages
(from torch-geometric) (2024.6.1)
Requirement already satisfied: jinja2 in /usr/local/lib/python3.10/dist-packages
(from torch-geometric) (3.1.4)
Requirement already satisfied: numpy in /usr/local/lib/python3.10/dist-packages
(from torch-geometric) (1.26.4)
Requirement already satisfied: psutil>=5.8.0 in /usr/local/lib/python3.10/dist-
```

```
packages (from torch-geometric) (5.9.5)
Requirement already satisfied: pyparsing in /usr/local/lib/python3.10/dist-
packages (from torch-geometric) (3.1.4)
Requirement already satisfied: requests in /usr/local/lib/python3.10/dist-
packages (from torch-geometric) (2.32.3)
Requirement already satisfied: tqdm in /usr/local/lib/python3.10/dist-packages
(from torch-geometric) (4.66.5)
Requirement already satisfied: aiohappyeyeballs>=2.3.0 in
/usr/local/lib/python3.10/dist-packages (from aiohttp->torch-geometric) (2.4.0)
Requirement already satisfied: aiosignal>=1.1.2 in
/usr/local/lib/python3.10/dist-packages (from aiohttp->torch-geometric) (1.3.1)
Requirement already satisfied: attrs>=17.3.0 in /usr/local/lib/python3.10/dist-
packages (from aiohttp->torch-geometric) (24.2.0)
Requirement already satisfied: frozenlist>=1.1.1 in
/usr/local/lib/python3.10/dist-packages (from aiohttp->torch-geometric) (1.4.1)
Requirement already satisfied: multidict<7.0,>=4.5 in
/usr/local/lib/python3.10/dist-packages (from aiohttp->torch-geometric) (6.1.0)
Requirement already satisfied: yarl<2.0,>=1.0 in /usr/local/lib/python3.10/dist-
packages (from aiohttp->torch-geometric) (1.11.1)
Requirement already satisfied: async-timeout<5.0,>=4.0 in
/usr/local/lib/python3.10/dist-packages (from aiohttp->torch-geometric) (4.0.3)
Requirement already satisfied: MarkupSafe>=2.0 in
/usr/local/lib/python3.10/dist-packages (from jinja2->torch-geometric) (2.1.5)
Requirement already satisfied: charset-normalizer<4,>=2 in
/usr/local/lib/python3.10/dist-packages (from requests->torch-geometric) (3.3.2)
Requirement already satisfied: idna<4,>=2.5 in /usr/local/lib/python3.10/dist-
packages (from requests->torch-geometric) (3.10)
Requirement already satisfied: urllib3<3,>=1.21.1 in
/usr/local/lib/python3.10/dist-packages (from requests->torch-geometric) (2.2.3)
Requirement already satisfied: certifi>=2017.4.17 in
/usr/local/lib/python3.10/dist-packages (from requests->torch-geometric)
(2024.8.30)
Requirement already satisfied: typing-extensions>=4.1.0 in
/usr/local/lib/python3.10/dist-packages (from
multidict<7.0,>=4.5->aiohttp->torch-geometric) (4.12.2)
Downloading torch_geometric-2.6.0-py3-none-any.whl (1.1 MB)
                         1.1/1.1 MB
25.3 MB/s eta 0:00:00
Installing collected packages: torch-geometric
Successfully installed torch-geometric-2.6.0
```

[ ]:
```python
import torch
import torch.nn.functional as F
from torch_geometric.nn import GCNConv
from sklearn.metrics import accuracy_score, f1_score
from sklearn.model_selection import train_test_split
import numpy as np
```

```python
# Use your fingerprint array `X` from your previous code
# X represents the features, and y represents the labels
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=42)

# Convert X_train and y_train to PyTorch tensors
X_train_tensor = torch.tensor(X_train, dtype=torch.float)
X_test_tensor = torch.tensor(X_test, dtype=torch.float)
y_train_tensor = torch.tensor(y_train, dtype=torch.float)
y_test_tensor = torch.tensor(y_test, dtype=torch.float)

# GNN model definition
class GCN(torch.nn.Module):
    def __init__(self, input_dim, hidden_dim, output_dim, dropout=0.5):
        super(GCN, self).__init__()
        self.conv1 = GCNConv(input_dim, hidden_dim)
        self.conv2 = GCNConv(hidden_dim, hidden_dim)
        self.fc = torch.nn.Linear(hidden_dim, output_dim)
        self.dropout = dropout

    def forward(self, x, edge_index):
        x = self.conv1(x, edge_index)
        x = F.relu(x)
        x = self.conv2(x, edge_index)
        x = F.relu(x)
        x = F.dropout(x, p=self.dropout, training=self.training)
        x = self.fc(x)
        return torch.sigmoid(x)  # For multi-label classification

# Hyperparameter grid
param_grid = {
    'hidden_dim': [64, 128],  # Number of hidden units
    'dropout': [0.3, 0.5],  # Dropout rate
    'epochs': [10,15,20],  # Number of epochs
    'batch_size': [32, 64],  # Batch size
    'learning_rate': [0.01, 0.001]  # Learning rates
}

# Variables to track the best model
best_binary_acc = 0
best_f1 = 0
best_normal_acc = 0
best_params = {}

# Simulate edge connections for GCN (simplified for fingerprints, no real graph structure available)
```

```
edge_index = torch.tensor([[0, 1], [1, 0]], dtype=torch.long)  # Placeholder␣
↪edge connections for GCN

# Perform hyperparameter tuning
for hidden_dim in param_grid['hidden_dim']:
    for dropout in param_grid['dropout']:
        for lr in param_grid['learning_rate']:
            for batch_size in param_grid['batch_size']:
                print(f'Training with hidden_dim={hidden_dim},␣
↪dropout={dropout}, lr={lr}, batch_size={batch_size}')

                # Initialize model
                model = GCN(input_dim=X_train.shape[1], hidden_dim=hidden_dim,␣
↪output_dim=y_train.shape[1], dropout=dropout)

                # Loss and optimizer
                optimizer = torch.optim.Adam(model.parameters(), lr=lr)
                criterion = torch.nn.BCELoss()

                # Train the model
                for epoch in range(param_grid['epochs'][0]):
                    model.train()
                    optimizer.zero_grad()
                    out = model(X_train_tensor, edge_index)
                    loss = criterion(out, y_train_tensor)
                    loss.backward()
                    optimizer.step()
                    print(f"Epoch {epoch+1}/{param_grid['epochs'][0]}, Loss:␣
↪{loss.item()}")

                # Evaluate the model
                model.eval()
                with torch.no_grad():
                    y_pred = model(X_test_tensor, edge_index)
                    y_pred_binary = (y_pred > 0.5).float()

                # Calculate binary accuracy
                binary_acc = (y_pred_binary == y_test_tensor).float().mean().
↪item()

                # Calculate normal accuracy
                normal_acc = accuracy_score(y_test, y_pred_binary.numpy())

                # Calculate F1 score
                f1 = f1_score(y_test, y_pred_binary.numpy(), average='micro')

                # Update best params if this model is better
```

```python
                if binary_acc > best_binary_acc:
                    best_binary_acc = binary_acc
                    best_f1 = f1
                    best_normal_acc = normal_acc
                    best_params = {
                        'hidden_dim': hidden_dim,
                        'dropout': dropout,
                        'lr': lr,
                        'batch_size': batch_size
                    }

# Print the best results
print(f'Best Params: {best_params}')
print(f'Best Binary Accuracy: {best_binary_acc:.4f}')
print(f'Best Normal Accuracy: {best_normal_acc:.4f}')
print(f'Best F1 Score: {best_f1:.4f}')
```

```
Training with hidden_dim=64, dropout=0.3, lr=0.01, batch_size=32
Epoch 1/10, Loss: 0.709597110748291
Epoch 2/10, Loss: 0.6415501832962036
Epoch 3/10, Loss: 0.5371586084365845
Epoch 4/10, Loss: 0.4165983200073242
Epoch 5/10, Loss: 0.3561471402645111
Epoch 6/10, Loss: 0.35802194476127625
Epoch 7/10, Loss: 0.3422446548938751
Epoch 8/10, Loss: 0.3302128314971924
Epoch 9/10, Loss: 0.33071452379226685
Epoch 10/10, Loss: 0.3300503194332123
Training with hidden_dim=64, dropout=0.3, lr=0.01, batch_size=64
Epoch 1/10, Loss: 0.6885817646980286
Epoch 2/10, Loss: 0.5982992649078369
Epoch 3/10, Loss: 0.47013750672340393
Epoch 4/10, Loss: 0.36258840560913086
Epoch 5/10, Loss: 0.3617899715900421
Epoch 6/10, Loss: 0.35540688037872314
Epoch 7/10, Loss: 0.34128352999687195
Epoch 8/10, Loss: 0.33353498578071594
Epoch 9/10, Loss: 0.3304072618484497
Epoch 10/10, Loss: 0.3054121136665344
Training with hidden_dim=64, dropout=0.3, lr=0.001, batch_size=32
Epoch 1/10, Loss: 0.7070016264915466
Epoch 2/10, Loss: 0.6985868215560913
Epoch 3/10, Loss: 0.6910017132759094
Epoch 4/10, Loss: 0.6830345392227173
Epoch 5/10, Loss: 0.674662709236145
Epoch 6/10, Loss: 0.6660047173500061
Epoch 7/10, Loss: 0.6564385890960693
```

```
Epoch 8/10, Loss: 0.6457278728485107
Epoch 9/10, Loss: 0.6339900493621826
Epoch 10/10, Loss: 0.6205863356590271
Training with hidden_dim=64, dropout=0.3, lr=0.001, batch_size=64
Epoch 1/10, Loss: 0.6994835138320923
Epoch 2/10, Loss: 0.6906312108039856
Epoch 3/10, Loss: 0.6824186444282532
Epoch 4/10, Loss: 0.6741432547569275
Epoch 5/10, Loss: 0.6649523973464966
Epoch 6/10, Loss: 0.6558994054794312
Epoch 7/10, Loss: 0.6441252827644348
Epoch 8/10, Loss: 0.633897602558136
Epoch 9/10, Loss: 0.6206861138343811
Epoch 10/10, Loss: 0.6066814064979553
Training with hidden_dim=64, dropout=0.5, lr=0.01, batch_size=32
Epoch 1/10, Loss: 0.6883083581924438
Epoch 2/10, Loss: 0.6086488962173462
Epoch 3/10, Loss: 0.4917351305484772
Epoch 4/10, Loss: 0.3852728307247162
Epoch 5/10, Loss: 0.3716936707496643
Epoch 6/10, Loss: 0.3797025680541992
Epoch 7/10, Loss: 0.3605228364467621
Epoch 8/10, Loss: 0.3535187542438507
Epoch 9/10, Loss: 0.33475977182388306
Epoch 10/10, Loss: 0.3031013011932373
Training with hidden_dim=64, dropout=0.5, lr=0.01, batch_size=64
Epoch 1/10, Loss: 0.6837955713272095
Epoch 2/10, Loss: 0.6130130887031555
Epoch 3/10, Loss: 0.5039457678794861
Epoch 4/10, Loss: 0.40949174761772156
Epoch 5/10, Loss: 0.3780607283115387
Epoch 6/10, Loss: 0.3632622957229614
Epoch 7/10, Loss: 0.34651923179626465
Epoch 8/10, Loss: 0.33340340852737427
Epoch 9/10, Loss: 0.3266436457633972
Epoch 10/10, Loss: 0.30676063895225525
Training with hidden_dim=64, dropout=0.5, lr=0.001, batch_size=32
Epoch 1/10, Loss: 0.6869210004806519
Epoch 2/10, Loss: 0.6782388091087341
Epoch 3/10, Loss: 0.6700339913368225
Epoch 4/10, Loss: 0.6610857248306274
Epoch 5/10, Loss: 0.6522855162620544
Epoch 6/10, Loss: 0.6425086855888367
Epoch 7/10, Loss: 0.6305216550827026
Epoch 8/10, Loss: 0.6191814541816711
Epoch 9/10, Loss: 0.606719970703125
Epoch 10/10, Loss: 0.5933386087417603
Training with hidden_dim=64, dropout=0.5, lr=0.001, batch_size=64
```

```
Epoch 1/10, Loss: 0.6958237290382385
Epoch 2/10, Loss: 0.6901788711547852
Epoch 3/10, Loss: 0.6850414872169495
Epoch 4/10, Loss: 0.6796642541885376
Epoch 5/10, Loss: 0.6730794906616211
Epoch 6/10, Loss: 0.6666602492332458
Epoch 7/10, Loss: 0.6587667465209961
Epoch 8/10, Loss: 0.650916576385498
Epoch 9/10, Loss: 0.6414874792098999
Epoch 10/10, Loss: 0.6299852728843689
Training with hidden_dim=128, dropout=0.3, lr=0.01, batch_size=32
Epoch 1/10, Loss: 0.6866666078567505
Epoch 2/10, Loss: 0.5021674633026123
Epoch 3/10, Loss: 0.35043177008628845
Epoch 4/10, Loss: 0.3630755543708801
Epoch 5/10, Loss: 0.33068397641181946
Epoch 6/10, Loss: 0.3221912086009979
Epoch 7/10, Loss: 0.30208665132522583
Epoch 8/10, Loss: 0.2855725884437561
Epoch 9/10, Loss: 0.2744961380958557
Epoch 10/10, Loss: 0.2614966034889221
Training with hidden_dim=128, dropout=0.3, lr=0.01, batch_size=64
Epoch 1/10, Loss: 0.6873682141304016
Epoch 2/10, Loss: 0.5006620287895203
Epoch 3/10, Loss: 0.3505573868751526
Epoch 4/10, Loss: 0.3546064496040344
Epoch 5/10, Loss: 0.3181251287460327
Epoch 6/10, Loss: 0.32015225291252136
Epoch 7/10, Loss: 0.3082484304904938
Epoch 8/10, Loss: 0.28546085953712463
Epoch 9/10, Loss: 0.2734510898590088
Epoch 10/10, Loss: 0.2627168893814087
Training with hidden_dim=128, dropout=0.3, lr=0.001, batch_size=32
Epoch 1/10, Loss: 0.6873036026954651
Epoch 2/10, Loss: 0.6764920949935913
Epoch 3/10, Loss: 0.6656936407089233
Epoch 4/10, Loss: 0.6537812948226929
Epoch 5/10, Loss: 0.639602541923523
Epoch 6/10, Loss: 0.6232268214225769
Epoch 7/10, Loss: 0.6041022539138794
Epoch 8/10, Loss: 0.5826297998428345
Epoch 9/10, Loss: 0.5572234392166138
Epoch 10/10, Loss: 0.5309958457946777
Training with hidden_dim=128, dropout=0.3, lr=0.001, batch_size=64
Epoch 1/10, Loss: 0.6876800060272217
Epoch 2/10, Loss: 0.6736128926277161
Epoch 3/10, Loss: 0.6599327921867371
Epoch 4/10, Loss: 0.6450079679489136
```

```
Epoch 5/10, Loss: 0.6280670166015625
Epoch 6/10, Loss: 0.6088924407958984
Epoch 7/10, Loss: 0.5874664783477783
Epoch 8/10, Loss: 0.5634155869483948
Epoch 9/10, Loss: 0.5374200940132141
Epoch 10/10, Loss: 0.5099051594734192
Training with hidden_dim=128, dropout=0.5, lr=0.01, batch_size=32
Epoch 1/10, Loss: 0.6915625929832458
Epoch 2/10, Loss: 0.5237178206443787
Epoch 3/10, Loss: 0.3717024326324463
Epoch 4/10, Loss: 0.3740563988685608
Epoch 5/10, Loss: 0.3378613591194153
Epoch 6/10, Loss: 0.3174165189266205
Epoch 7/10, Loss: 0.31205984950065613
Epoch 8/10, Loss: 0.2967338562011719
Epoch 9/10, Loss: 0.27938246726989746
Epoch 10/10, Loss: 0.26713234186172485
Training with hidden_dim=128, dropout=0.5, lr=0.01, batch_size=64
Epoch 1/10, Loss: 0.6962990760803223
Epoch 2/10, Loss: 0.5339031219482422
Epoch 3/10, Loss: 0.36193016171455383
Epoch 4/10, Loss: 0.36410513520240784
Epoch 5/10, Loss: 0.346571683883667
Epoch 6/10, Loss: 0.3508358895778656
Epoch 7/10, Loss: 0.31922414898872375
Epoch 8/10, Loss: 0.29260289669036865
Epoch 9/10, Loss: 0.2768450081348419
Epoch 10/10, Loss: 0.266574501991272
Training with hidden_dim=128, dropout=0.5, lr=0.001, batch_size=32
Epoch 1/10, Loss: 0.6906462907791138
Epoch 2/10, Loss: 0.6779081225395203
Epoch 3/10, Loss: 0.6650395393371582
Epoch 4/10, Loss: 0.651150643825531
Epoch 5/10, Loss: 0.6342793107032776
Epoch 6/10, Loss: 0.6168862581253052
Epoch 7/10, Loss: 0.5972729325294495
Epoch 8/10, Loss: 0.5753123164176941
Epoch 9/10, Loss: 0.5508957505226135
Epoch 10/10, Loss: 0.5242833495140076
Training with hidden_dim=128, dropout=0.5, lr=0.001, batch_size=64
Epoch 1/10, Loss: 0.6975699663162231
Epoch 2/10, Loss: 0.6850435137748718
Epoch 3/10, Loss: 0.6736516356468201
Epoch 4/10, Loss: 0.6612989902496338
Epoch 5/10, Loss: 0.6475942134857178
Epoch 6/10, Loss: 0.6322312355041504
Epoch 7/10, Loss: 0.6135097742080688
Epoch 8/10, Loss: 0.5938735008239746
```

```
Epoch 9/10, Loss: 0.570262610912323
Epoch 10/10, Loss: 0.5439286231994629
Best Params: {'hidden_dim': 64, 'dropout': 0.5, 'lr': 0.01, 'batch_size': 32}
Best Binary Accuracy: 0.9114
Best Normal Accuracy: 0.2884
Best F1 Score: 0.9536
```

[ ]:
```
# 15 epochs

import torch
import torch.nn.functional as F
from torch_geometric.nn import GCNConv
from sklearn.metrics import accuracy_score, f1_score
from sklearn.model_selection import train_test_split
import numpy as np

# Use your fingerprint array `X` from your previous code
# X represents the features, and y represents the labels
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2,␣
↪random_state=42)

# Convert X_train and y_train to PyTorch tensors
X_train_tensor = torch.tensor(X_train, dtype=torch.float)
X_test_tensor = torch.tensor(X_test, dtype=torch.float)
y_train_tensor = torch.tensor(y_train, dtype=torch.float)
y_test_tensor = torch.tensor(y_test, dtype=torch.float)

# GNN model definition
class GCN(torch.nn.Module):
    def __init__(self, input_dim, hidden_dim, output_dim, dropout=0.5):
        super(GCN, self).__init__()
        self.conv1 = GCNConv(input_dim, hidden_dim)
        self.conv2 = GCNConv(hidden_dim, hidden_dim)
        self.fc = torch.nn.Linear(hidden_dim, output_dim)
        self.dropout = dropout

    def forward(self, x, edge_index):
        x = self.conv1(x, edge_index)
        x = F.relu(x)
        x = self.conv2(x, edge_index)
        x = F.relu(x)
        x = F.dropout(x, p=self.dropout, training=self.training)
        x = self.fc(x)
        return torch.sigmoid(x)  # For multi-label classification

# Hyperparameter grid
param_grid = {
```

```
    'hidden_dim': [64, 128],  # Number of hidden units
    'dropout': [0.3, 0.5],  # Dropout rate
    'epochs': [15,20],  # Number of epochs
    'batch_size': [32, 64],  # Batch size
    'learning_rate': [0.01, 0.001]  # Learning rates
}

# Variables to track the best model
best_binary_acc = 0
best_f1 = 0
best_normal_acc = 0
best_params = {}

# Simulate edge connections for GCN (simplified for fingerprints, no real graph␣
↪structure available)
edge_index = torch.tensor([[0, 1], [1, 0]], dtype=torch.long)  # Placeholder␣
↪edge connections for GCN

# Perform hyperparameter tuning
for hidden_dim in param_grid['hidden_dim']:
    for dropout in param_grid['dropout']:
        for lr in param_grid['learning_rate']:
            for batch_size in param_grid['batch_size']:
                print(f'Training with hidden_dim={hidden_dim},␣
↪dropout={dropout}, lr={lr}, batch_size={batch_size}')

                # Initialize model
                model = GCN(input_dim=X_train.shape[1], hidden_dim=hidden_dim,␣
↪output_dim=y_train.shape[1], dropout=dropout)

                # Loss and optimizer
                optimizer = torch.optim.Adam(model.parameters(), lr=lr)
                criterion = torch.nn.BCELoss()

                # Train the model
                for epoch in range(param_grid['epochs'][0]):
                    model.train()
                    optimizer.zero_grad()
                    out = model(X_train_tensor, edge_index)
                    loss = criterion(out, y_train_tensor)
                    loss.backward()
                    optimizer.step()
                    print(f"Epoch {epoch+1}/{param_grid['epochs'][0]}, Loss:␣
↪{loss.item()}")

                # Evaluate the model
                model.eval()
```

```
                with torch.no_grad():
                    y_pred = model(X_test_tensor, edge_index)
                    y_pred_binary = (y_pred > 0.5).float()

                # Calculate binary accuracy
                binary_acc = (y_pred_binary == y_test_tensor).float().mean().
 ↪item()

                # Calculate normal accuracy
                normal_acc = accuracy_score(y_test, y_pred_binary.numpy())

                # Calculate F1 score
                f1 = f1_score(y_test, y_pred_binary.numpy(), average='micro')

                # Update best params if this model is better
                if binary_acc > best_binary_acc:
                    best_binary_acc = binary_acc
                    best_f1 = f1
                    best_normal_acc = normal_acc
                    best_params = {
                        'hidden_dim': hidden_dim,
                        'dropout': dropout,
                        'lr': lr,
                        'batch_size': batch_size
                    }

# Print the best results
print(f'Best Params: {best_params}')
print(f'Best Binary Accuracy: {best_binary_acc:.4f}')
print(f'Best Normal Accuracy: {best_normal_acc:.4f}')
print(f'Best F1 Score: {best_f1:.4f}')
```

```
Training with hidden_dim=64, dropout=0.3, lr=0.01, batch_size=32
Epoch 1/15, Loss: 0.6952621936798096
Epoch 2/15, Loss: 0.6270509362220764
Epoch 3/15, Loss: 0.5159099698066711
Epoch 4/15, Loss: 0.4039263129234314
Epoch 5/15, Loss: 0.35897353291511536
Epoch 6/15, Loss: 0.35700345039367676
Epoch 7/15, Loss: 0.33971405029296875
Epoch 8/15, Loss: 0.3220519721508026
Epoch 9/15, Loss: 0.30620330572128296
Epoch 10/15, Loss: 0.29612797498703003
Epoch 11/15, Loss: 0.2941589951515198
Epoch 12/15, Loss: 0.2867872416973114
Epoch 13/15, Loss: 0.27960288524627686
Epoch 14/15, Loss: 0.27185094356536865
```

```
Epoch 15/15, Loss: 0.2621895968914032
Training with hidden_dim=64, dropout=0.3, lr=0.01, batch_size=64
Epoch 1/15, Loss: 0.6914840340614319
Epoch 2/15, Loss: 0.5877062082290649
Epoch 3/15, Loss: 0.45496731996536255
Epoch 4/15, Loss: 0.36330509185791016
Epoch 5/15, Loss: 0.3593505620956421
Epoch 6/15, Loss: 0.35855719447135925
Epoch 7/15, Loss: 0.3380836844444275
Epoch 8/15, Loss: 0.3227978050708771
Epoch 9/15, Loss: 0.3131084144115448
Epoch 10/15, Loss: 0.3077828586101532
Epoch 11/15, Loss: 0.29653123021125793
Epoch 12/15, Loss: 0.2870234549045563
Epoch 13/15, Loss: 0.2783140540122986
Epoch 14/15, Loss: 0.2743372619152069
Epoch 15/15, Loss: 0.26745304465293884
Training with hidden_dim=64, dropout=0.3, lr=0.001, batch_size=32
Epoch 1/15, Loss: 0.702755868434906
Epoch 2/15, Loss: 0.6954046487808228
Epoch 3/15, Loss: 0.6881657242774963
Epoch 4/15, Loss: 0.6816341876983643
Epoch 5/15, Loss: 0.6737716794013977
Epoch 6/15, Loss: 0.665929913520813
Epoch 7/15, Loss: 0.6572540998458862
Epoch 8/15, Loss: 0.6486310362815857
Epoch 9/15, Loss: 0.6388293504714966
Epoch 10/15, Loss: 0.6286277770996094
Epoch 11/15, Loss: 0.6166297793388367
Epoch 12/15, Loss: 0.6044055223464966
Epoch 13/15, Loss: 0.5907407999038696
Epoch 14/15, Loss: 0.5772386789321899
Epoch 15/15, Loss: 0.562836229801178
Training with hidden_dim=64, dropout=0.3, lr=0.001, batch_size=64
Epoch 1/15, Loss: 0.696432888507843
Epoch 2/15, Loss: 0.6891416907310486
Epoch 3/15, Loss: 0.6812499761581421
Epoch 4/15, Loss: 0.6732641458511353
Epoch 5/15, Loss: 0.6636177897453308
Epoch 6/15, Loss: 0.653079628944397
Epoch 7/15, Loss: 0.641034722328186
Epoch 8/15, Loss: 0.6281229853630066
Epoch 9/15, Loss: 0.6139667630195618
Epoch 10/15, Loss: 0.5996797680854797
Epoch 11/15, Loss: 0.582539975643158
Epoch 12/15, Loss: 0.565375566482544
Epoch 13/15, Loss: 0.547389030456543
Epoch 14/15, Loss: 0.529052197933197
```

```
Epoch 15/15, Loss: 0.5103024840354919
Training with hidden_dim=64, dropout=0.5, lr=0.01, batch_size=32
Epoch 1/15, Loss: 0.6947649717330933
Epoch 2/15, Loss: 0.6070510149002075
Epoch 3/15, Loss: 0.48495668172836304
Epoch 4/15, Loss: 0.39044877886772156
Epoch 5/15, Loss: 0.3640161156654358
Epoch 6/15, Loss: 0.3784691393375397
Epoch 7/15, Loss: 0.3711124062538147
Epoch 8/15, Loss: 0.34947407245635986
Epoch 9/15, Loss: 0.3223731517791748
Epoch 10/15, Loss: 0.3064509928226471
Epoch 11/15, Loss: 0.293600469827652
Epoch 12/15, Loss: 0.2847106158733368
Epoch 13/15, Loss: 0.2806759774684906
Epoch 14/15, Loss: 0.27583757042884827
Epoch 15/15, Loss: 0.2674733102321625
Training with hidden_dim=64, dropout=0.5, lr=0.01, batch_size=64
Epoch 1/15, Loss: 0.7033355236053467
Epoch 2/15, Loss: 0.6540586352348328
Epoch 3/15, Loss: 0.5690011978149414
Epoch 4/15, Loss: 0.45475414395332336
Epoch 5/15, Loss: 0.3819302022457123
Epoch 6/15, Loss: 0.3742135465145111
Epoch 7/15, Loss: 0.3624228537082672
Epoch 8/15, Loss: 0.3378712832927704
Epoch 9/15, Loss: 0.3384837806224823
Epoch 10/15, Loss: 0.32551366090774536
Epoch 11/15, Loss: 0.3183295428752899
Epoch 12/15, Loss: 0.30532941222190857
Epoch 13/15, Loss: 0.2939428687095642
Epoch 14/15, Loss: 0.2852615714073181
Epoch 15/15, Loss: 0.27718284726142883
Training with hidden_dim=64, dropout=0.5, lr=0.001, batch_size=32
Epoch 1/15, Loss: 0.6957007646560669
Epoch 2/15, Loss: 0.6888988018035889
Epoch 3/15, Loss: 0.6822246313095093
Epoch 4/15, Loss: 0.6756710410118103
Epoch 5/15, Loss: 0.6685109734535217
Epoch 6/15, Loss: 0.6610754132270813
Epoch 7/15, Loss: 0.6521624326705933
Epoch 8/15, Loss: 0.642544150352478
Epoch 9/15, Loss: 0.632664680480957
Epoch 10/15, Loss: 0.620508074760437
Epoch 11/15, Loss: 0.6088703274726868
Epoch 12/15, Loss: 0.5949753522872925
Epoch 13/15, Loss: 0.5826719999313354
Epoch 14/15, Loss: 0.5675411224365234
```

```
Epoch 15/15, Loss: 0.5528921484947205
Training with hidden_dim=64, dropout=0.5, lr=0.001, batch_size=64
Epoch 1/15, Loss: 0.6945118308067322
Epoch 2/15, Loss: 0.6879046559333801
Epoch 3/15, Loss: 0.6814534664154053
Epoch 4/15, Loss: 0.6744557023048401
Epoch 5/15, Loss: 0.6674869656562805
Epoch 6/15, Loss: 0.6597470045089722
Epoch 7/15, Loss: 0.6507701873779297
Epoch 8/15, Loss: 0.6401440501213074
Epoch 9/15, Loss: 0.6306188106536865
Epoch 10/15, Loss: 0.6196376085281372
Epoch 11/15, Loss: 0.6070877313613892
Epoch 12/15, Loss: 0.5922108292579651
Epoch 13/15, Loss: 0.5794215798377991
Epoch 14/15, Loss: 0.566223680973053
Epoch 15/15, Loss: 0.5492606163024902
Training with hidden_dim=128, dropout=0.3, lr=0.01, batch_size=32
Epoch 1/15, Loss: 0.6888718008995056
Epoch 2/15, Loss: 0.5243257880210876
Epoch 3/15, Loss: 0.37243324518203735
Epoch 4/15, Loss: 0.3653971254825592
Epoch 5/15, Loss: 0.32846546173095703
Epoch 6/15, Loss: 0.32312098145484924
Epoch 7/15, Loss: 0.31686633825302124
Epoch 8/15, Loss: 0.2964753210544586
Epoch 9/15, Loss: 0.27938830852508545
Epoch 10/15, Loss: 0.2678970694541931
Epoch 11/15, Loss: 0.26006126403808594
Epoch 12/15, Loss: 0.25500231981277466
Epoch 13/15, Loss: 0.24978049099445343
Epoch 14/15, Loss: 0.24291157722473145
Epoch 15/15, Loss: 0.23726727068424225
Training with hidden_dim=128, dropout=0.3, lr=0.01, batch_size=64
Epoch 1/15, Loss: 0.6895471215248108
Epoch 2/15, Loss: 0.49223464727401733
Epoch 3/15, Loss: 0.35082191228866577
Epoch 4/15, Loss: 0.3517342805862427
Epoch 5/15, Loss: 0.3347811698913574
Epoch 6/15, Loss: 0.31298282742500305
Epoch 7/15, Loss: 0.2931625247001648
Epoch 8/15, Loss: 0.28444793820381165
Epoch 9/15, Loss: 0.2777138352394104
Epoch 10/15, Loss: 0.2675612270832062
Epoch 11/15, Loss: 0.25444501638412476
Epoch 12/15, Loss: 0.24776823818683624
Epoch 13/15, Loss: 0.2400817573070526
Epoch 14/15, Loss: 0.23304712772369385
```

```
Epoch 15/15, Loss: 0.22763872146606445
Training with hidden_dim=128, dropout=0.3, lr=0.001, batch_size=32
Epoch 1/15, Loss: 0.7088435292243958
Epoch 2/15, Loss: 0.6967198848724365
Epoch 3/15, Loss: 0.6847159266471863
Epoch 4/15, Loss: 0.671735405921936
Epoch 5/15, Loss: 0.6577410697937012
Epoch 6/15, Loss: 0.6414211988449097
Epoch 7/15, Loss: 0.6229097843170166
Epoch 8/15, Loss: 0.6021108031272888
Epoch 9/15, Loss: 0.5787609219551086
Epoch 10/15, Loss: 0.5521972179412842
Epoch 11/15, Loss: 0.5242392420768738
Epoch 12/15, Loss: 0.49602723121643066
Epoch 13/15, Loss: 0.46506765484809875
Epoch 14/15, Loss: 0.435176283121109
Epoch 15/15, Loss: 0.4073580205440521
Training with hidden_dim=128, dropout=0.3, lr=0.001, batch_size=64
Epoch 1/15, Loss: 0.6804156303405762
Epoch 2/15, Loss: 0.6646252274513245
Epoch 3/15, Loss: 0.6480933427810669
Epoch 4/15, Loss: 0.6302090883255005
Epoch 5/15, Loss: 0.6105372905731201
Epoch 6/15, Loss: 0.5887525677680969
Epoch 7/15, Loss: 0.5648030638694763
Epoch 8/15, Loss: 0.538982629776001
Epoch 9/15, Loss: 0.5099431872367859
Epoch 10/15, Loss: 0.4816553592681885
Epoch 11/15, Loss: 0.45217645168304443
Epoch 12/15, Loss: 0.4215465486049652
Epoch 13/15, Loss: 0.3979380130767822
Epoch 14/15, Loss: 0.3748511075973511
Epoch 15/15, Loss: 0.3561728894710541
Training with hidden_dim=128, dropout=0.5, lr=0.01, batch_size=32
Epoch 1/15, Loss: 0.6960592865943909
Epoch 2/15, Loss: 0.5291829109191895
Epoch 3/15, Loss: 0.3792456388473511
Epoch 4/15, Loss: 0.36259040236473083
Epoch 5/15, Loss: 0.3265916407108307
Epoch 6/15, Loss: 0.32525113224983215
Epoch 7/15, Loss: 0.30931514501571655
Epoch 8/15, Loss: 0.29500746726989746
Epoch 9/15, Loss: 0.27984681725502014
Epoch 10/15, Loss: 0.26834261417388916
Epoch 11/15, Loss: 0.2586096227169037
Epoch 12/15, Loss: 0.2531587481498718
Epoch 13/15, Loss: 0.24863970279693604
Epoch 14/15, Loss: 0.24157246947288513
```

```
Epoch 15/15, Loss: 0.23506829142570496
Training with hidden_dim=128, dropout=0.5, lr=0.01, batch_size=64
Epoch 1/15, Loss: 0.6962944269180298
Epoch 2/15, Loss: 0.5241618752479553
Epoch 3/15, Loss: 0.3604148328304291
Epoch 4/15, Loss: 0.3768053650856018
Epoch 5/15, Loss: 0.357745498418808
Epoch 6/15, Loss: 0.34560859203338623
Epoch 7/15, Loss: 0.3325866162776947
Epoch 8/15, Loss: 0.30219602584838867
Epoch 9/15, Loss: 0.2831186354160309
Epoch 10/15, Loss: 0.2781851887702942
Epoch 11/15, Loss: 0.2718324661254883
Epoch 12/15, Loss: 0.26350200176239014
Epoch 13/15, Loss: 0.2581109404563904
Epoch 14/15, Loss: 0.25290772318840027
Epoch 15/15, Loss: 0.2442295104265213
Training with hidden_dim=128, dropout=0.5, lr=0.001, batch_size=32
Epoch 1/15, Loss: 0.6976119875907898
Epoch 2/15, Loss: 0.6853035688400269
Epoch 3/15, Loss: 0.6744401454925537
Epoch 4/15, Loss: 0.6618966460227966
Epoch 5/15, Loss: 0.6475056409835815
Epoch 6/15, Loss: 0.6316348314285278
Epoch 7/15, Loss: 0.6129761934280396
Epoch 8/15, Loss: 0.5919350981712341
Epoch 9/15, Loss: 0.5683477520942688
Epoch 10/15, Loss: 0.5433096885681152
Epoch 11/15, Loss: 0.5153184533119202
Epoch 12/15, Loss: 0.4874979257583618
Epoch 13/15, Loss: 0.4613899290561676
Epoch 14/15, Loss: 0.43263110518455505
Epoch 15/15, Loss: 0.4104588031768799
Training with hidden_dim=128, dropout=0.5, lr=0.001, batch_size=64
Epoch 1/15, Loss: 0.6979619264602661
Epoch 2/15, Loss: 0.6862553954124451
Epoch 3/15, Loss: 0.6747465133666992
Epoch 4/15, Loss: 0.6624504327774048
Epoch 5/15, Loss: 0.6475896239280701
Epoch 6/15, Loss: 0.6316494345664978
Epoch 7/15, Loss: 0.6139592528343201
Epoch 8/15, Loss: 0.5932042002677917
Epoch 9/15, Loss: 0.571079671382904
Epoch 10/15, Loss: 0.5459352135658264
Epoch 11/15, Loss: 0.5209935307502747
Epoch 12/15, Loss: 0.4932430386543274
Epoch 13/15, Loss: 0.4669094681739807
Epoch 14/15, Loss: 0.43961241841316223
```

```
Epoch 15/15, Loss: 0.4179519712924957
Best Params: {'hidden_dim': 64, 'dropout': 0.3, 'lr': 0.01, 'batch_size': 32}
Best Binary Accuracy: 0.9114
Best Normal Accuracy: 0.2884
Best F1 Score: 0.9536
```

[ ]:
```
# 20 epochs

import torch
import torch.nn.functional as F
from torch_geometric.nn import GCNConv
from sklearn.metrics import accuracy_score, f1_score
from sklearn.model_selection import train_test_split
import numpy as np

# Use your fingerprint array `X` from your previous code
# X represents the features, and y represents the labels
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2,␣
 ↪random_state=42)

# Convert X_train and y_train to PyTorch tensors
X_train_tensor = torch.tensor(X_train, dtype=torch.float)
X_test_tensor = torch.tensor(X_test, dtype=torch.float)
y_train_tensor = torch.tensor(y_train, dtype=torch.float)
y_test_tensor = torch.tensor(y_test, dtype=torch.float)

# GNN model definition
class GCN(torch.nn.Module):
    def __init__(self, input_dim, hidden_dim, output_dim, dropout=0.5):
        super(GCN, self).__init__()
        self.conv1 = GCNConv(input_dim, hidden_dim)
        self.conv2 = GCNConv(hidden_dim, hidden_dim)
        self.fc = torch.nn.Linear(hidden_dim, output_dim)
        self.dropout = dropout

    def forward(self, x, edge_index):
        x = self.conv1(x, edge_index)
        x = F.relu(x)
        x = self.conv2(x, edge_index)
        x = F.relu(x)
        x = F.dropout(x, p=self.dropout, training=self.training)
        x = self.fc(x)
        return torch.sigmoid(x)  # For multi-label classification

# Hyperparameter grid
param_grid = {
    'hidden_dim': [64, 128],  # Number of hidden units
```

```
    'dropout': [0.3, 0.5],  # Dropout rate
    'epochs': [20],  # Number of epochs
    'batch_size': [32, 64],  # Batch size
    'learning_rate': [0.01, 0.001]  # Learning rates
}

# Variables to track the best model
best_binary_acc = 0
best_f1 = 0
best_normal_acc = 0
best_params = {}

# Simulate edge connections for GCN (simplified for fingerprints, no real graph␣
↪structure available)
edge_index = torch.tensor([[0, 1], [1, 0]], dtype=torch.long)  # Placeholder␣
↪edge connections for GCN

# Perform hyperparameter tuning
for hidden_dim in param_grid['hidden_dim']:
    for dropout in param_grid['dropout']:
        for lr in param_grid['learning_rate']:
            for batch_size in param_grid['batch_size']:
                print(f'Training with hidden_dim={hidden_dim},␣
↪dropout={dropout}, lr={lr}, batch_size={batch_size}')

                # Initialize model
                model = GCN(input_dim=X_train.shape[1], hidden_dim=hidden_dim,␣
↪output_dim=y_train.shape[1], dropout=dropout)

                # Loss and optimizer
                optimizer = torch.optim.Adam(model.parameters(), lr=lr)
                criterion = torch.nn.BCELoss()

                # Train the model
                for epoch in range(param_grid['epochs'][0]):
                    model.train()
                    optimizer.zero_grad()
                    out = model(X_train_tensor, edge_index)
                    loss = criterion(out, y_train_tensor)
                    loss.backward()
                    optimizer.step()
                    print(f"Epoch {epoch+1}/{param_grid['epochs'][0]}, Loss:␣
↪{loss.item()}")

                # Evaluate the model
                model.eval()
                with torch.no_grad():
```

```
                y_pred = model(X_test_tensor, edge_index)
                y_pred_binary = (y_pred > 0.5).float()

            # Calculate binary accuracy
            binary_acc = (y_pred_binary == y_test_tensor).float().mean().
↪item()

            # Calculate normal accuracy
            normal_acc = accuracy_score(y_test, y_pred_binary.numpy())

            # Calculate F1 score
            f1 = f1_score(y_test, y_pred_binary.numpy(), average='micro')

            # Update best params if this model is better
            if binary_acc > best_binary_acc:
                best_binary_acc = binary_acc
                best_f1 = f1
                best_normal_acc = normal_acc
                best_params = {
                    'hidden_dim': hidden_dim,
                    'dropout': dropout,
                    'lr': lr,
                    'batch_size': batch_size
                }

# Print the best results
print(f'Best Params: {best_params}')
print(f'Best Binary Accuracy: {best_binary_acc:.4f}')
print(f'Best Normal Accuracy: {best_normal_acc:.4f}')
print(f'Best F1 Score: {best_f1:.4f}')
```

```
Training with hidden_dim=64, dropout=0.3, lr=0.01, batch_size=32
Epoch 1/20, Loss: 0.699658215045929
Epoch 2/20, Loss: 0.6155330538749695
Epoch 3/20, Loss: 0.49272221326828003
Epoch 4/20, Loss: 0.3868323266506195
Epoch 5/20, Loss: 0.3517073094844818
Epoch 6/20, Loss: 0.34631210565567017
Epoch 7/20, Loss: 0.3485317528247833
Epoch 8/20, Loss: 0.3484322726726532
Epoch 9/20, Loss: 0.326820433139801
Epoch 10/20, Loss: 0.30933257937431335
Epoch 11/20, Loss: 0.29509687423706055
Epoch 12/20, Loss: 0.28475749492645264
Epoch 13/20, Loss: 0.27821609377861023
Epoch 14/20, Loss: 0.27134689688682556
Epoch 15/20, Loss: 0.26733213663101196
```

```
Epoch 16/20, Loss: 0.2629132866859436
Epoch 17/20, Loss: 0.2571866810321808
Epoch 18/20, Loss: 0.2497851848602295
Epoch 19/20, Loss: 0.2458299845457077
Epoch 20/20, Loss: 0.23911352455615997
Training with hidden_dim=64, dropout=0.3, lr=0.01, batch_size=64
Epoch 1/20, Loss: 0.689071536064148
Epoch 2/20, Loss: 0.6114605665206909
Epoch 3/20, Loss: 0.493852823972702
Epoch 4/20, Loss: 0.3913193345069885
Epoch 5/20, Loss: 0.35648322105407715
Epoch 6/20, Loss: 0.3593050539493561
Epoch 7/20, Loss: 0.3385007977485657
Epoch 8/20, Loss: 0.32776835560798645
Epoch 9/20, Loss: 0.31240102648735046
Epoch 10/20, Loss: 0.3068794310092926
Epoch 11/20, Loss: 0.2983551621437073
Epoch 12/20, Loss: 0.28837481141090393
Epoch 13/20, Loss: 0.2841404378414154
Epoch 14/20, Loss: 0.27849772572517395
Epoch 15/20, Loss: 0.2714681029319763
Epoch 16/20, Loss: 0.26540112495422363
Epoch 17/20, Loss: 0.25614210963249207
Epoch 18/20, Loss: 0.25432994961738586
Epoch 19/20, Loss: 0.24874615669250488
Epoch 20/20, Loss: 0.24394464492797852
Training with hidden_dim=64, dropout=0.3, lr=0.001, batch_size=32
Epoch 1/20, Loss: 0.6948914527893066
Epoch 2/20, Loss: 0.687730610370636
Epoch 3/20, Loss: 0.680662989616394
Epoch 4/20, Loss: 0.6732134819030762
Epoch 5/20, Loss: 0.6651691198348999
Epoch 6/20, Loss: 0.6568726301193237
Epoch 7/20, Loss: 0.6458554863929749
Epoch 8/20, Loss: 0.6349275708198547
Epoch 9/20, Loss: 0.6235425472259521
Epoch 10/20, Loss: 0.6095243096351624
Epoch 11/20, Loss: 0.5942180156707764
Epoch 12/20, Loss: 0.5785388946533203
Epoch 13/20, Loss: 0.5620080232620239
Epoch 14/20, Loss: 0.5418387651443481
Epoch 15/20, Loss: 0.5247167348861694
Epoch 16/20, Loss: 0.5045859217643738
Epoch 17/20, Loss: 0.4861680865287781
Epoch 18/20, Loss: 0.4683403968811035
Epoch 19/20, Loss: 0.45041289925575256
Epoch 20/20, Loss: 0.4320085644721985
Training with hidden_dim=64, dropout=0.3, lr=0.001, batch_size=64
```

```
Epoch 1/20, Loss: 0.7194058299064636
Epoch 2/20, Loss: 0.710715115070343
Epoch 3/20, Loss: 0.7026896476745605
Epoch 4/20, Loss: 0.6951087117195129
Epoch 5/20, Loss: 0.6870121955871582
Epoch 6/20, Loss: 0.678348958492279
Epoch 7/20, Loss: 0.6694285273551941
Epoch 8/20, Loss: 0.6597533822059631
Epoch 9/20, Loss: 0.648698627948761
Epoch 10/20, Loss: 0.6363694667816162
Epoch 11/20, Loss: 0.6230244040489197
Epoch 12/20, Loss: 0.6089291572570801
Epoch 13/20, Loss: 0.5925900340080261
Epoch 14/20, Loss: 0.5761035680770874
Epoch 15/20, Loss: 0.556637704372406
Epoch 16/20, Loss: 0.538627564907074
Epoch 17/20, Loss: 0.5190118551254272
Epoch 18/20, Loss: 0.49880486726760864
Epoch 19/20, Loss: 0.47925859689712524
Epoch 20/20, Loss: 0.46012991666793823
Training with hidden_dim=64, dropout=0.5, lr=0.01, batch_size=32
Epoch 1/20, Loss: 0.7025178074836731
Epoch 2/20, Loss: 0.6372708082199097
Epoch 3/20, Loss: 0.5434900522232056
Epoch 4/20, Loss: 0.4404847025871277
Epoch 5/20, Loss: 0.38116493821144104
Epoch 6/20, Loss: 0.36800429224967957
Epoch 7/20, Loss: 0.3541412949562073
Epoch 8/20, Loss: 0.36018747091293335
Epoch 9/20, Loss: 0.33891749382019043
Epoch 10/20, Loss: 0.3219658434391022
Epoch 11/20, Loss: 0.3021220564842224
Epoch 12/20, Loss: 0.29172712564468384
Epoch 13/20, Loss: 0.28107649087905884
Epoch 14/20, Loss: 0.2755497395992279
Epoch 15/20, Loss: 0.27361568808555603
Epoch 16/20, Loss: 0.2650458514690399
Epoch 17/20, Loss: 0.25991135835647583
Epoch 18/20, Loss: 0.25569507479667664
Epoch 19/20, Loss: 0.2511289417743683
Epoch 20/20, Loss: 0.24631927907466888
Training with hidden_dim=64, dropout=0.5, lr=0.01, batch_size=64
Epoch 1/20, Loss: 0.694157600402832
Epoch 2/20, Loss: 0.6093996167182922
Epoch 3/20, Loss: 0.4871779680252075
Epoch 4/20, Loss: 0.38617488741874695
Epoch 5/20, Loss: 0.3694049119949341
Epoch 6/20, Loss: 0.3787538707256317
```

```
Epoch 7/20, Loss: 0.36230239272117615
Epoch 8/20, Loss: 0.34602871537208557
Epoch 9/20, Loss: 0.333862841129303
Epoch 10/20, Loss: 0.32130396366119385
Epoch 11/20, Loss: 0.3040819466114044
Epoch 12/20, Loss: 0.2941168546676636
Epoch 13/20, Loss: 0.28447604179382324
Epoch 14/20, Loss: 0.27676495909690857
Epoch 15/20, Loss: 0.26904061436653137
Epoch 16/20, Loss: 0.2661812901496887
Epoch 17/20, Loss: 0.25640058517456055
Epoch 18/20, Loss: 0.2539396286010742
Epoch 19/20, Loss: 0.2509534955024719
Epoch 20/20, Loss: 0.24531027674674988
Training with hidden_dim=64, dropout=0.5, lr=0.001, batch_size=32
Epoch 1/20, Loss: 0.6998839974403381
Epoch 2/20, Loss: 0.6922768950462341
Epoch 3/20, Loss: 0.685097873210907
Epoch 4/20, Loss: 0.678838849067688
Epoch 5/20, Loss: 0.6724370121955872
Epoch 6/20, Loss: 0.664829671382904
Epoch 7/20, Loss: 0.6577200293540955
Epoch 8/20, Loss: 0.6492687463760376
Epoch 9/20, Loss: 0.6397125720977783
Epoch 10/20, Loss: 0.6312083601951599
Epoch 11/20, Loss: 0.6204937696456909
Epoch 12/20, Loss: 0.6085951924324036
Epoch 13/20, Loss: 0.5971465706825256
Epoch 14/20, Loss: 0.583692729473114
Epoch 15/20, Loss: 0.5693644285202026
Epoch 16/20, Loss: 0.5568744540214539
Epoch 17/20, Loss: 0.5416817665100098
Epoch 18/20, Loss: 0.5258736610412598
Epoch 19/20, Loss: 0.5113489627838135
Epoch 20/20, Loss: 0.4953991770744324
Training with hidden_dim=64, dropout=0.5, lr=0.001, batch_size=64
Epoch 1/20, Loss: 0.6916524767875671
Epoch 2/20, Loss: 0.6814015507698059
Epoch 3/20, Loss: 0.6711322069168091
Epoch 4/20, Loss: 0.6617146134376526
Epoch 5/20, Loss: 0.6502377986907959
Epoch 6/20, Loss: 0.639369785785675
Epoch 7/20, Loss: 0.6287029385566711
Epoch 8/20, Loss: 0.6143694519996643
Epoch 9/20, Loss: 0.600808322429657
Epoch 10/20, Loss: 0.5860493183135986
Epoch 11/20, Loss: 0.5688501000404358
Epoch 12/20, Loss: 0.5529199242591858
```

```
Epoch 13/20, Loss: 0.5347387790679932
Epoch 14/20, Loss: 0.5159860253334045
Epoch 15/20, Loss: 0.49905380606651306
Epoch 16/20, Loss: 0.480751097202301
Epoch 17/20, Loss: 0.46414583921432495
Epoch 18/20, Loss: 0.44476085901260376
Epoch 19/20, Loss: 0.4280952513217926
Epoch 20/20, Loss: 0.4164988100528717
Training with hidden_dim=128, dropout=0.3, lr=0.01, batch_size=32
Epoch 1/20, Loss: 0.6897670030593872
Epoch 2/20, Loss: 0.523259162902832
Epoch 3/20, Loss: 0.362259179353714
Epoch 4/20, Loss: 0.35802575945854187
Epoch 5/20, Loss: 0.33104443550109863
Epoch 6/20, Loss: 0.315444678068161
Epoch 7/20, Loss: 0.30620211362838745
Epoch 8/20, Loss: 0.29395565390586853
Epoch 9/20, Loss: 0.2809019684791565
Epoch 10/20, Loss: 0.27081596851348877
Epoch 11/20, Loss: 0.26138633489608765
Epoch 12/20, Loss: 0.25596171617507935
Epoch 13/20, Loss: 0.25002631545066833
Epoch 14/20, Loss: 0.24552980065345764
Epoch 15/20, Loss: 0.2378195822238922
Epoch 16/20, Loss: 0.23236191272735596
Epoch 17/20, Loss: 0.22776545584201813
Epoch 18/20, Loss: 0.22492577135562897
Epoch 19/20, Loss: 0.21984989941120148
Epoch 20/20, Loss: 0.21754394471645355
Training with hidden_dim=128, dropout=0.3, lr=0.01, batch_size=64
Epoch 1/20, Loss: 0.6903532147407532
Epoch 2/20, Loss: 0.5243184566497803
Epoch 3/20, Loss: 0.36493411660194397
Epoch 4/20, Loss: 0.3774659335613251
Epoch 5/20, Loss: 0.3320021629333496
Epoch 6/20, Loss: 0.31105828285217285
Epoch 7/20, Loss: 0.30745986104011536
Epoch 8/20, Loss: 0.29489219188690186
Epoch 9/20, Loss: 0.286286324262619
Epoch 10/20, Loss: 0.27538734674453735
Epoch 11/20, Loss: 0.26334893703460693
Epoch 12/20, Loss: 0.2570473551750183
Epoch 13/20, Loss: 0.2521562874317169
Epoch 14/20, Loss: 0.24833105504512787
Epoch 15/20, Loss: 0.2399148941040039
Epoch 16/20, Loss: 0.234702929854393
Epoch 17/20, Loss: 0.23042643070220947
Epoch 18/20, Loss: 0.2286936640739441
```

```
Epoch 19/20, Loss: 0.22438377141952515
Epoch 20/20, Loss: 0.22136300802230835
Training with hidden_dim=128, dropout=0.3, lr=0.001, batch_size=32
Epoch 1/20, Loss: 0.688434362411499
Epoch 2/20, Loss: 0.6762621402740479
Epoch 3/20, Loss: 0.6636173725128174
Epoch 4/20, Loss: 0.6503449082374573
Epoch 5/20, Loss: 0.6352349519729614
Epoch 6/20, Loss: 0.6177411079406738
Epoch 7/20, Loss: 0.5982077121734619
Epoch 8/20, Loss: 0.576682448387146
Epoch 9/20, Loss: 0.5533820390701294
Epoch 10/20, Loss: 0.5282666087150574
Epoch 11/20, Loss: 0.49981507658958435
Epoch 12/20, Loss: 0.4722060263156891
Epoch 13/20, Loss: 0.44530409574508667
Epoch 14/20, Loss: 0.41847658157348633
Epoch 15/20, Loss: 0.3944931626319885
Epoch 16/20, Loss: 0.3737901747226715
Epoch 17/20, Loss: 0.35796058177948
Epoch 18/20, Loss: 0.34394049644470215
Epoch 19/20, Loss: 0.3358052968978882
Epoch 20/20, Loss: 0.32949161529541016
Training with hidden_dim=128, dropout=0.3, lr=0.001, batch_size=64
Epoch 1/20, Loss: 0.691110372543335
Epoch 2/20, Loss: 0.6797335147857666
Epoch 3/20, Loss: 0.6682026386260986
Epoch 4/20, Loss: 0.6569870710372925
Epoch 5/20, Loss: 0.6440669894218445
Epoch 6/20, Loss: 0.6291279792785645
Epoch 7/20, Loss: 0.6128146052360535
Epoch 8/20, Loss: 0.5940009355545044
Epoch 9/20, Loss: 0.5730326771736145
Epoch 10/20, Loss: 0.5492037534713745
Epoch 11/20, Loss: 0.5255206823348999
Epoch 12/20, Loss: 0.4990648627281189
Epoch 13/20, Loss: 0.470781147480011
Epoch 14/20, Loss: 0.4430924952030182
Epoch 15/20, Loss: 0.4164915680885315
Epoch 16/20, Loss: 0.39286819100379944
Epoch 17/20, Loss: 0.37142014503479004
Epoch 18/20, Loss: 0.3544582724571228
Epoch 19/20, Loss: 0.3415623605251312
Epoch 20/20, Loss: 0.3336162269115448
Training with hidden_dim=128, dropout=0.5, lr=0.01, batch_size=32
Epoch 1/20, Loss: 0.7092819213867188
Epoch 2/20, Loss: 0.5725641846656799
Epoch 3/20, Loss: 0.4039178788661957
```

```
Epoch 4/20, Loss: 0.3725810647010803
Epoch 5/20, Loss: 0.36250993609428406
Epoch 6/20, Loss: 0.33197858929634094
Epoch 7/20, Loss: 0.31752538681030273
Epoch 8/20, Loss: 0.3088107705116272
Epoch 9/20, Loss: 0.2950946092605591
Epoch 10/20, Loss: 0.285712331533432
Epoch 11/20, Loss: 0.27566009759902954
Epoch 12/20, Loss: 0.26707518100738525
Epoch 13/20, Loss: 0.2612447440624237
Epoch 14/20, Loss: 0.255196213722229
Epoch 15/20, Loss: 0.24966809153556824
Epoch 16/20, Loss: 0.24377015233039856
Epoch 17/20, Loss: 0.2388983815908432
Epoch 18/20, Loss: 0.23604774475097656
Epoch 19/20, Loss: 0.23312272131443024
Epoch 20/20, Loss: 0.228593647480011
Training with hidden_dim=128, dropout=0.5, lr=0.01, batch_size=64
Epoch 1/20, Loss: 0.7010146975517273
Epoch 2/20, Loss: 0.5799434185028076
Epoch 3/20, Loss: 0.40602782368659973
Epoch 4/20, Loss: 0.369476318359375
Epoch 5/20, Loss: 0.35787224769592285
Epoch 6/20, Loss: 0.33343306183815
Epoch 7/20, Loss: 0.3145439326763153
Epoch 8/20, Loss: 0.30883821845054626
Epoch 9/20, Loss: 0.3034120798110962
Epoch 10/20, Loss: 0.29011261463165283
Epoch 11/20, Loss: 0.27633577585220337
Epoch 12/20, Loss: 0.2643585205078125
Epoch 13/20, Loss: 0.2587999701499939
Epoch 14/20, Loss: 0.2566339671611786
Epoch 15/20, Loss: 0.24680820107460022
Epoch 16/20, Loss: 0.24079933762550354
Epoch 17/20, Loss: 0.2369673252105713
Epoch 18/20, Loss: 0.23411229252815247
Epoch 19/20, Loss: 0.22890222072601318
Epoch 20/20, Loss: 0.22622136771678925
Training with hidden_dim=128, dropout=0.5, lr=0.001, batch_size=32
Epoch 1/20, Loss: 0.6892096996307373
Epoch 2/20, Loss: 0.6777986884117126
Epoch 3/20, Loss: 0.6656076312065125
Epoch 4/20, Loss: 0.6524375081062317
Epoch 5/20, Loss: 0.6380916833877563
Epoch 6/20, Loss: 0.6220588684082031
Epoch 7/20, Loss: 0.6038244366645813
Epoch 8/20, Loss: 0.5831056833267212
Epoch 9/20, Loss: 0.5596638917922974
```

```
Epoch 10/20, Loss: 0.5336142182350159
Epoch 11/20, Loss: 0.5063611268997192
Epoch 12/20, Loss: 0.4779794216156006
Epoch 13/20, Loss: 0.4496092200279236
Epoch 14/20, Loss: 0.4231898784637451
Epoch 15/20, Loss: 0.39845648407936096
Epoch 16/20, Loss: 0.37828829884529114
Epoch 17/20, Loss: 0.36275947093963623
Epoch 18/20, Loss: 0.35127347707748413
Epoch 19/20, Loss: 0.342928946018219
Epoch 20/20, Loss: 0.33968785405158997
Training with hidden_dim=128, dropout=0.5, lr=0.001, batch_size=64
Epoch 1/20, Loss: 0.6854851245880127
Epoch 2/20, Loss: 0.6727800369262695
Epoch 3/20, Loss: 0.6594692468643188
Epoch 4/20, Loss: 0.6460111737251282
Epoch 5/20, Loss: 0.629231333732605
Epoch 6/20, Loss: 0.6117150187492371
Epoch 7/20, Loss: 0.5924291610717773
Epoch 8/20, Loss: 0.5726961493492126
Epoch 9/20, Loss: 0.5484854578971863
Epoch 10/20, Loss: 0.5237650275230408
Epoch 11/20, Loss: 0.4974948763847351
Epoch 12/20, Loss: 0.47382792830467224
Epoch 13/20, Loss: 0.44789350032806396
Epoch 14/20, Loss: 0.42338332533836365
Epoch 15/20, Loss: 0.40014541149139404
Epoch 16/20, Loss: 0.3826206624507904
Epoch 17/20, Loss: 0.3689213991165161
Epoch 18/20, Loss: 0.3575296103954315
Epoch 19/20, Loss: 0.3470417857170105
Epoch 20/20, Loss: 0.33902236819267273
Best Params: {'hidden_dim': 128, 'dropout': 0.3, 'lr': 0.01, 'batch_size': 64}
Best Binary Accuracy: 0.9118
Best Normal Accuracy: 0.2884
Best F1 Score: 0.9539
```

[ ]:
```
# 50 epochs (same in 35)



import torch
import torch.nn.functional as F
from torch_geometric.nn import GCNConv
from sklearn.metrics import accuracy_score, f1_score
from sklearn.model_selection import train_test_split
import numpy as np
```

```python
# Use your fingerprint array `X` from your previous code
# X represents the features, and y represents the labels
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2,␣
 ↪random_state=42)

# Convert X_train and y_train to PyTorch tensors
X_train_tensor = torch.tensor(X_train, dtype=torch.float)
X_test_tensor = torch.tensor(X_test, dtype=torch.float)
y_train_tensor = torch.tensor(y_train, dtype=torch.float)
y_test_tensor = torch.tensor(y_test, dtype=torch.float)

# GNN model definition
class GCN(torch.nn.Module):
    def __init__(self, input_dim, hidden_dim, output_dim, dropout=0.5):
        super(GCN, self).__init__()
        self.conv1 = GCNConv(input_dim, hidden_dim)
        self.conv2 = GCNConv(hidden_dim, hidden_dim)
        self.fc = torch.nn.Linear(hidden_dim, output_dim)
        self.dropout = dropout

    def forward(self, x, edge_index):
        x = self.conv1(x, edge_index)
        x = F.relu(x)
        x = self.conv2(x, edge_index)
        x = F.relu(x)
        x = F.dropout(x, p=self.dropout, training=self.training)
        x = self.fc(x)
        return torch.sigmoid(x)  # For multi-label classification

# Hyperparameter grid
param_grid = {
    'hidden_dim': [64, 128],  # Number of hidden units
    'dropout': [0.3, 0.5],  # Dropout rate
    'epochs': [50],  # Number of epochs
    'batch_size': [32, 64],  # Batch size
    'learning_rate': [0.01, 0.001]  # Learning rates
}

# Variables to track the best model
best_binary_acc = 0
best_f1 = 0
best_normal_acc = 0
best_params = {}

# Simulate edge connections for GCN (simplified for fingerprints, no real graph␣
 ↪structure available)
```

```python
edge_index = torch.tensor([[0, 1], [1, 0]], dtype=torch.long)  # Placeholder␣
↪edge connections for GCN

# Perform hyperparameter tuning
for hidden_dim in param_grid['hidden_dim']:
    for dropout in param_grid['dropout']:
        for lr in param_grid['learning_rate']:
            for batch_size in param_grid['batch_size']:
                print(f'Training with hidden_dim={hidden_dim},␣
↪dropout={dropout}, lr={lr}, batch_size={batch_size}')

                # Initialize model
                model = GCN(input_dim=X_train.shape[1], hidden_dim=hidden_dim,␣
↪output_dim=y_train.shape[1], dropout=dropout)

                # Loss and optimizer
                optimizer = torch.optim.Adam(model.parameters(), lr=lr)
                criterion = torch.nn.BCELoss()

                # Train the model
                for epoch in range(param_grid['epochs'][0]):
                    model.train()
                    optimizer.zero_grad()
                    out = model(X_train_tensor, edge_index)
                    loss = criterion(out, y_train_tensor)
                    loss.backward()
                    optimizer.step()
                    #print(f"Epoch {epoch+1}/{param_grid['epochs'][0]}, Loss:␣
↪{loss.item()}")

                # Evaluate the model
                model.eval()
                with torch.no_grad():
                    y_pred = model(X_test_tensor, edge_index)
                    y_pred_binary = (y_pred > 0.5).float()

                # Calculate binary accuracy
                binary_acc = (y_pred_binary == y_test_tensor).float().mean().
↪item()

                # Calculate normal accuracy
                normal_acc = accuracy_score(y_test, y_pred_binary.numpy())

                # Calculate F1 score
                f1 = f1_score(y_test, y_pred_binary.numpy(), average='micro')

                # Update best params if this model is better
```

```python
                if binary_acc > best_binary_acc:
                    best_binary_acc = binary_acc
                    best_f1 = f1
                    best_normal_acc = normal_acc
                    best_params = {
                        'hidden_dim': hidden_dim,
                        'dropout': dropout,
                        'lr': lr,
                        'batch_size': batch_size
                    }

# Print the best results
print(f'Best Params: {best_params}')
print(f'Best Binary Accuracy: {best_binary_acc:.4f}')
print(f'Best Normal Accuracy: {best_normal_acc:.4f}')
print(f'Best F1 Score: {best_f1:.4f}')
```

```
Training with hidden_dim=64, dropout=0.3, lr=0.01, batch_size=32
Training with hidden_dim=64, dropout=0.3, lr=0.01, batch_size=64
Training with hidden_dim=64, dropout=0.3, lr=0.001, batch_size=32
Training with hidden_dim=64, dropout=0.3, lr=0.001, batch_size=64
Training with hidden_dim=64, dropout=0.5, lr=0.01, batch_size=32
Training with hidden_dim=64, dropout=0.5, lr=0.01, batch_size=64
Training with hidden_dim=64, dropout=0.5, lr=0.001, batch_size=32
Training with hidden_dim=64, dropout=0.5, lr=0.001, batch_size=64
Training with hidden_dim=128, dropout=0.3, lr=0.01, batch_size=32
Training with hidden_dim=128, dropout=0.3, lr=0.01, batch_size=64
Training with hidden_dim=128, dropout=0.3, lr=0.001, batch_size=32
Training with hidden_dim=128, dropout=0.3, lr=0.001, batch_size=64
Training with hidden_dim=128, dropout=0.5, lr=0.01, batch_size=32
Training with hidden_dim=128, dropout=0.5, lr=0.01, batch_size=64
Training with hidden_dim=128, dropout=0.5, lr=0.001, batch_size=32
Training with hidden_dim=128, dropout=0.5, lr=0.001, batch_size=64
Best Params: {'hidden_dim': 128, 'dropout': 0.5, 'lr': 0.001, 'batch_size': 32}
Best Binary Accuracy: 0.9115
Best Normal Accuracy: 0.2884
Best F1 Score: 0.9537
```

[ ]:

```python
# 100 epochs

# 20 epochs

import torch
import torch.nn.functional as F
from torch_geometric.nn import GCNConv
from sklearn.metrics import accuracy_score, f1_score
from sklearn.model_selection import train_test_split
```

```python
import numpy as np

# Use your fingerprint array `X` from your previous code
# X represents the features, and y represents the labels
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2,
    random_state=42)

# Convert X_train and y_train to PyTorch tensors
X_train_tensor = torch.tensor(X_train, dtype=torch.float)
X_test_tensor = torch.tensor(X_test, dtype=torch.float)
y_train_tensor = torch.tensor(y_train, dtype=torch.float)
y_test_tensor = torch.tensor(y_test, dtype=torch.float)

# GNN model definition
class GCN(torch.nn.Module):
    def __init__(self, input_dim, hidden_dim, output_dim, dropout=0.5):
        super(GCN, self).__init__()
        self.conv1 = GCNConv(input_dim, hidden_dim)
        self.conv2 = GCNConv(hidden_dim, hidden_dim)
        self.fc = torch.nn.Linear(hidden_dim, output_dim)
        self.dropout = dropout

    def forward(self, x, edge_index):
        x = self.conv1(x, edge_index)
        x = F.relu(x)
        x = self.conv2(x, edge_index)
        x = F.relu(x)
        x = F.dropout(x, p=self.dropout, training=self.training)
        x = self.fc(x)
        return torch.sigmoid(x)  # For multi-label classification

# Hyperparameter grid
param_grid = {
    'hidden_dim': [64, 128],  # Number of hidden units
    'dropout': [0.3, 0.5],  # Dropout rate
    'epochs': [100],  # Number of epochs
    'batch_size': [32, 64],  # Batch size
    'learning_rate': [0.01, 0.001]  # Learning rates
}

# Variables to track the best model
best_binary_acc = 0
best_f1 = 0
best_normal_acc = 0
best_params = {}
```

```
# Simulate edge connections for GCN (simplified for fingerprints, no real graph␣
↪structure available)
edge_index = torch.tensor([[0, 1], [1, 0]], dtype=torch.long)  # Placeholder␣
↪edge connections for GCN

# Perform hyperparameter tuning
for hidden_dim in param_grid['hidden_dim']:
    for dropout in param_grid['dropout']:
        for lr in param_grid['learning_rate']:
            for batch_size in param_grid['batch_size']:
                print(f'Training with hidden_dim={hidden_dim},␣
↪dropout={dropout}, lr={lr}, batch_size={batch_size}')

                # Initialize model
                model = GCN(input_dim=X_train.shape[1], hidden_dim=hidden_dim,␣
↪output_dim=y_train.shape[1], dropout=dropout)

                # Loss and optimizer
                optimizer = torch.optim.Adam(model.parameters(), lr=lr)
                criterion = torch.nn.BCELoss()

                # Train the model
                for epoch in range(param_grid['epochs'][0]):
                    model.train()
                    optimizer.zero_grad()
                    out = model(X_train_tensor, edge_index)
                    loss = criterion(out, y_train_tensor)
                    loss.backward()
                    optimizer.step()

                # Evaluate the model
                model.eval()
                with torch.no_grad():
                    y_pred = model(X_test_tensor, edge_index)
                    y_pred_binary = (y_pred > 0.5).float()

                # Calculate binary accuracy
                binary_acc = (y_pred_binary == y_test_tensor).float().mean().
↪item()

                # Calculate normal accuracy
                normal_acc = accuracy_score(y_test, y_pred_binary.numpy())

                # Calculate F1 score
                f1 = f1_score(y_test, y_pred_binary.numpy(), average='micro')

                # Update best params if this model is better
```

```python
                if binary_acc > best_binary_acc:
                    best_binary_acc = binary_acc
                    best_f1 = f1
                    best_normal_acc = normal_acc
                    best_params = {
                        'hidden_dim': hidden_dim,
                        'dropout': dropout,
                        'lr': lr,
                        'batch_size': batch_size
                    }

# Print the best results
print(f'Best Params: {best_params}')
print(f'Best Binary Accuracy: {best_binary_acc:.4f}')
print(f'Best Normal Accuracy: {best_normal_acc:.4f}')
print(f'Best F1 Score: {best_f1:.4f}')
```

```
Training with hidden_dim=64, dropout=0.3, lr=0.01, batch_size=32
Training with hidden_dim=64, dropout=0.3, lr=0.01, batch_size=64
Training with hidden_dim=64, dropout=0.3, lr=0.001, batch_size=32
Training with hidden_dim=64, dropout=0.3, lr=0.001, batch_size=64
Training with hidden_dim=64, dropout=0.5, lr=0.01, batch_size=32
Training with hidden_dim=64, dropout=0.5, lr=0.01, batch_size=64
Training with hidden_dim=64, dropout=0.5, lr=0.001, batch_size=32
Training with hidden_dim=64, dropout=0.5, lr=0.001, batch_size=64
Training with hidden_dim=128, dropout=0.3, lr=0.01, batch_size=32
Training with hidden_dim=128, dropout=0.3, lr=0.01, batch_size=64
Training with hidden_dim=128, dropout=0.3, lr=0.001, batch_size=32
Training with hidden_dim=128, dropout=0.3, lr=0.001, batch_size=64
Training with hidden_dim=128, dropout=0.5, lr=0.01, batch_size=32
Training with hidden_dim=128, dropout=0.5, lr=0.01, batch_size=64
Training with hidden_dim=128, dropout=0.5, lr=0.001, batch_size=32
Training with hidden_dim=128, dropout=0.5, lr=0.001, batch_size=64
Best Params: {'hidden_dim': 64, 'dropout': 0.5, 'lr': 0.001, 'batch_size': 32}
Best Binary Accuracy: 0.9115
Best Normal Accuracy: 0.2884
Best F1 Score: 0.9537
```

[ ]:
```python
# 25 epochs

import torch
import torch.nn.functional as F
from torch_geometric.nn import GCNConv
from sklearn.metrics import accuracy_score, f1_score
from sklearn.model_selection import train_test_split
import numpy as np
```

```python
# Use your fingerprint array `X` from your previous code
# X represents the features, and y represents the labels
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2,␣
↪random_state=42)

# Convert X_train and y_train to PyTorch tensors
X_train_tensor = torch.tensor(X_train, dtype=torch.float)
X_test_tensor = torch.tensor(X_test, dtype=torch.float)
y_train_tensor = torch.tensor(y_train, dtype=torch.float)
y_test_tensor = torch.tensor(y_test, dtype=torch.float)

# GNN model definition
class GCN(torch.nn.Module):
    def __init__(self, input_dim, hidden_dim, output_dim, dropout=0.5):
        super(GCN, self).__init__()
        self.conv1 = GCNConv(input_dim, hidden_dim)
        self.conv2 = GCNConv(hidden_dim, hidden_dim)
        self.fc = torch.nn.Linear(hidden_dim, output_dim)
        self.dropout = dropout

    def forward(self, x, edge_index):
        x = self.conv1(x, edge_index)
        x = F.relu(x)
        x = self.conv2(x, edge_index)
        x = F.relu(x)
        x = F.dropout(x, p=self.dropout, training=self.training)
        x = self.fc(x)
        return torch.sigmoid(x)  # For multi-label classification

# Hyperparameter grid
param_grid = {
    'hidden_dim': [64, 128],  # Number of hidden units
    'dropout': [0.3, 0.5],  # Dropout rate
    'epochs': [20],  # Number of epochs
    'batch_size': [32, 64],  # Batch size
    'learning_rate': [0.01, 0.001]  # Learning rates
}

# Variables to track the best model
best_binary_acc = 0
best_f1 = 0
best_normal_acc = 0
best_params = {}

# Simulate edge connections for GCN (simplified for fingerprints, no real graph␣
↪structure available)
```

```
edge_index = torch.tensor([[0, 1], [1, 0]], dtype=torch.long)  # Placeholder␣
↪edge connections for GCN

# Perform hyperparameter tuning
for hidden_dim in param_grid['hidden_dim']:
    for dropout in param_grid['dropout']:
        for lr in param_grid['learning_rate']:
            for batch_size in param_grid['batch_size']:
                print(f'Training with hidden_dim={hidden_dim},␣
↪dropout={dropout}, lr={lr}, batch_size={batch_size}')

                # Initialize model
                model = GCN(input_dim=X_train.shape[1], hidden_dim=hidden_dim,␣
↪output_dim=y_train.shape[1], dropout=dropout)

                # Loss and optimizer
                optimizer = torch.optim.Adam(model.parameters(), lr=lr)
                criterion = torch.nn.BCELoss()

                # Train the model
                for epoch in range(param_grid['epochs'][0]):
                    model.train()
                    optimizer.zero_grad()
                    out = model(X_train_tensor, edge_index)
                    loss = criterion(out, y_train_tensor)
                    loss.backward()
                    optimizer.step()

                # Evaluate the model
                model.eval()
                with torch.no_grad():
                    y_pred = model(X_test_tensor, edge_index)
                    y_pred_binary = (y_pred > 0.5).float()

                # Calculate binary accuracy
                binary_acc = (y_pred_binary == y_test_tensor).float().mean().
↪item()

                # Calculate normal accuracy
                normal_acc = accuracy_score(y_test, y_pred_binary.numpy())

                # Calculate F1 score
                f1 = f1_score(y_test, y_pred_binary.numpy(), average='micro')

                # Update best params if this model is better
                if binary_acc > best_binary_acc:
                    best_binary_acc = binary_acc
```

```
                    best_f1 = f1
                    best_normal_acc = normal_acc
                    best_params = {
                        'hidden_dim': hidden_dim,
                        'dropout': dropout,
                        'lr': lr,
                        'batch_size': batch_size
                    }

# Print the best results
print(f'Best Params: {best_params}')
print(f'Best Binary Accuracy: {best_binary_acc:.4f}')
print(f'Best Normal Accuracy: {best_normal_acc:.4f}')
print(f'Best F1 Score: {best_f1:.4f}')
```

```
Training with hidden_dim=64, dropout=0.3, lr=0.01, batch_size=32
Training with hidden_dim=64, dropout=0.3, lr=0.01, batch_size=64
Training with hidden_dim=64, dropout=0.3, lr=0.001, batch_size=32
Training with hidden_dim=64, dropout=0.3, lr=0.001, batch_size=64
Training with hidden_dim=64, dropout=0.5, lr=0.01, batch_size=32
Training with hidden_dim=64, dropout=0.5, lr=0.01, batch_size=64
Training with hidden_dim=64, dropout=0.5, lr=0.001, batch_size=32
Training with hidden_dim=64, dropout=0.5, lr=0.001, batch_size=64
Training with hidden_dim=128, dropout=0.3, lr=0.01, batch_size=32
Training with hidden_dim=128, dropout=0.3, lr=0.01, batch_size=64
Training with hidden_dim=128, dropout=0.3, lr=0.001, batch_size=32
Training with hidden_dim=128, dropout=0.3, lr=0.001, batch_size=64
Training with hidden_dim=128, dropout=0.5, lr=0.01, batch_size=32
Training with hidden_dim=128, dropout=0.5, lr=0.01, batch_size=64
Training with hidden_dim=128, dropout=0.5, lr=0.001, batch_size=32
Training with hidden_dim=128, dropout=0.5, lr=0.001, batch_size=64
Best Params: {'hidden_dim': 128, 'dropout': 0.3, 'lr': 0.01, 'batch_size': 64}
Best Binary Accuracy: 0.9120
Best Normal Accuracy: 0.2921
Best F1 Score: 0.9539
```

[ ]:

```
# 30 epochs

import torch
import torch.nn.functional as F
from torch_geometric.nn import GCNConv
from sklearn.metrics import accuracy_score, f1_score
from sklearn.model_selection import train_test_split
import numpy as np

# Use your fingerprint array `X` from your previous code
# X represents the features, and y represents the labels
```

```python
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2,␣
 ↪random_state=42)

# Convert X_train and y_train to PyTorch tensors
X_train_tensor = torch.tensor(X_train, dtype=torch.float)
X_test_tensor = torch.tensor(X_test, dtype=torch.float)
y_train_tensor = torch.tensor(y_train, dtype=torch.float)
y_test_tensor = torch.tensor(y_test, dtype=torch.float)

# GNN model definition
class GCN(torch.nn.Module):
    def __init__(self, input_dim, hidden_dim, output_dim, dropout=0.5):
        super(GCN, self).__init__()
        self.conv1 = GCNConv(input_dim, hidden_dim)
        self.conv2 = GCNConv(hidden_dim, hidden_dim)
        self.fc = torch.nn.Linear(hidden_dim, output_dim)
        self.dropout = dropout

    def forward(self, x, edge_index):
        x = self.conv1(x, edge_index)
        x = F.relu(x)
        x = self.conv2(x, edge_index)
        x = F.relu(x)
        x = F.dropout(x, p=self.dropout, training=self.training)
        x = self.fc(x)
        return torch.sigmoid(x)  # For multi-label classification

# Hyperparameter grid
param_grid = {
    'hidden_dim': [64, 128],  # Number of hidden units
    'dropout': [0.3, 0.5],  # Dropout rate
    'epochs': [30],  # Number of epochs
    'batch_size': [32, 64],  # Batch size
    'learning_rate': [0.01, 0.001]  # Learning rates
}

# Variables to track the best model
best_binary_acc = 0
best_f1 = 0
best_normal_acc = 0
best_params = {}

# Simulate edge connections for GCN (simplified for fingerprints, no real graph␣
 ↪structure available)
edge_index = torch.tensor([[0, 1], [1, 0]], dtype=torch.long)  # Placeholder␣
 ↪edge connections for GCN
```

```
# Perform hyperparameter tuning
for hidden_dim in param_grid['hidden_dim']:
    for dropout in param_grid['dropout']:
        for lr in param_grid['learning_rate']:
            for batch_size in param_grid['batch_size']:
                print(f'Training with hidden_dim={hidden_dim},␣
↪dropout={dropout}, lr={lr}, batch_size={batch_size}')

                # Initialize model
                model = GCN(input_dim=X_train.shape[1], hidden_dim=hidden_dim,␣
↪output_dim=y_train.shape[1], dropout=dropout)

                # Loss and optimizer
                optimizer = torch.optim.Adam(model.parameters(), lr=lr)
                criterion = torch.nn.BCELoss()

                # Train the model
                for epoch in range(param_grid['epochs'][0]):
                    model.train()
                    optimizer.zero_grad()
                    out = model(X_train_tensor, edge_index)
                    loss = criterion(out, y_train_tensor)
                    loss.backward()
                    optimizer.step()

                # Evaluate the model
                model.eval()
                with torch.no_grad():
                    y_pred = model(X_test_tensor, edge_index)
                    y_pred_binary = (y_pred > 0.5).float()

                # Calculate binary accuracy
                binary_acc = (y_pred_binary == y_test_tensor).float().mean().
↪item()

                # Calculate normal accuracy
                normal_acc = accuracy_score(y_test, y_pred_binary.numpy())

                # Calculate F1 score
                f1 = f1_score(y_test, y_pred_binary.numpy(), average='micro')

                # Update best params if this model is better
                if binary_acc > best_binary_acc:
                    best_binary_acc = binary_acc
                    best_f1 = f1
                    best_normal_acc = normal_acc
                    best_params = {
```

```
                        'hidden_dim': hidden_dim,
                        'dropout': dropout,
                        'lr': lr,
                        'batch_size': batch_size
                    }

# Print the best results
print(f'Best Params: {best_params}')
print(f'Best Binary Accuracy: {best_binary_acc:.4f}')
print(f'Best Normal Accuracy: {best_normal_acc:.4f}')
print(f'Best F1 Score: {best_f1:.4f}')
```

```
Training with hidden_dim=64, dropout=0.3, lr=0.01, batch_size=32
Training with hidden_dim=64, dropout=0.3, lr=0.01, batch_size=64
Training with hidden_dim=64, dropout=0.3, lr=0.001, batch_size=32
Training with hidden_dim=64, dropout=0.3, lr=0.001, batch_size=64
Training with hidden_dim=64, dropout=0.5, lr=0.01, batch_size=32
Training with hidden_dim=64, dropout=0.5, lr=0.01, batch_size=64
Training with hidden_dim=64, dropout=0.5, lr=0.001, batch_size=32
Training with hidden_dim=64, dropout=0.5, lr=0.001, batch_size=64
Training with hidden_dim=128, dropout=0.3, lr=0.01, batch_size=32
Training with hidden_dim=128, dropout=0.3, lr=0.01, batch_size=64
Training with hidden_dim=128, dropout=0.3, lr=0.001, batch_size=32
Training with hidden_dim=128, dropout=0.3, lr=0.001, batch_size=64
Training with hidden_dim=128, dropout=0.5, lr=0.01, batch_size=32
Training with hidden_dim=128, dropout=0.5, lr=0.01, batch_size=64
Training with hidden_dim=128, dropout=0.5, lr=0.001, batch_size=32
Training with hidden_dim=128, dropout=0.5, lr=0.001, batch_size=64
Best Params: {'hidden_dim': 64, 'dropout': 0.5, 'lr': 0.01, 'batch_size': 32}
Best Binary Accuracy: 0.9132
Best Normal Accuracy: 0.2921
Best F1 Score: 0.9545
```

[ ]:
```
# 33 epochs

import torch
import torch.nn.functional as F
from torch_geometric.nn import GCNConv
from sklearn.metrics import accuracy_score, f1_score
from sklearn.model_selection import train_test_split
import numpy as np

# Use your fingerprint array `X` from your previous code
# X represents the features, and y represents the labels
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2,␣
 ↪random_state=42)
```

```python
# Convert X_train and y_train to PyTorch tensors
X_train_tensor = torch.tensor(X_train, dtype=torch.float)
X_test_tensor = torch.tensor(X_test, dtype=torch.float)
y_train_tensor = torch.tensor(y_train, dtype=torch.float)
y_test_tensor = torch.tensor(y_test, dtype=torch.float)

# GNN model definition
class GCN(torch.nn.Module):
    def __init__(self, input_dim, hidden_dim, output_dim, dropout=0.5):
        super(GCN, self).__init__()
        self.conv1 = GCNConv(input_dim, hidden_dim)
        self.conv2 = GCNConv(hidden_dim, hidden_dim)
        self.fc = torch.nn.Linear(hidden_dim, output_dim)
        self.dropout = dropout

    def forward(self, x, edge_index):
        x = self.conv1(x, edge_index)
        x = F.relu(x)
        x = self.conv2(x, edge_index)
        x = F.relu(x)
        x = F.dropout(x, p=self.dropout, training=self.training)
        x = self.fc(x)
        return torch.sigmoid(x)  # For multi-label classification

# Hyperparameter grid
param_grid = {
    'hidden_dim': [64, 128],  # Number of hidden units
    'dropout': [0.3, 0.5],  # Dropout rate
    'epochs': [30],  # Number of epochs
    'batch_size': [32, 64],  # Batch size
    'learning_rate': [0.01, 0.001],  # Learning rates
    'weight_decay': [0, 1e-4, 1e-5],  # L2 regularization
    'activation': ['relu', 'leaky_relu', 'tanh'],  # Activation functions
    'num_layers': [2, 3],  # Number of GCNConv layers
    'lr_decay': [0.95, 0.99],  # Learning rate decay
    'batch_norm': [True, False],  # Batch normalization
    'aggregation': ['mean', 'sum', 'max'],  # Aggregation method
    'residual': [True, False]  # Residual connections
}


# Variables to track the best model
best_binary_acc = 0
best_f1 = 0
best_normal_acc = 0
best_params = {}
```

```python
# Simulate edge connections for GCN (simplified for fingerprints, no real graph structure available)
edge_index = torch.tensor([[0, 1], [1, 0]], dtype=torch.long)  # Placeholder edge connections for GCN

# Perform hyperparameter tuning
for hidden_dim in param_grid['hidden_dim']:
    for dropout in param_grid['dropout']:
        for lr in param_grid['learning_rate']:
            for batch_size in param_grid['batch_size']:
                print(f'Training with hidden_dim={hidden_dim}, dropout={dropout}, lr={lr}, batch_size={batch_size}')

                # Initialize model
                model = GCN(input_dim=X_train.shape[1], hidden_dim=hidden_dim, output_dim=y_train.shape[1], dropout=dropout)

                # Loss and optimizer
                optimizer = torch.optim.Adam(model.parameters(), lr=lr)
                criterion = torch.nn.BCELoss()

                # Train the model
                for epoch in range(param_grid['epochs'][0]):
                    model.train()
                    optimizer.zero_grad()
                    out = model(X_train_tensor, edge_index)
                    loss = criterion(out, y_train_tensor)
                    loss.backward()
                    optimizer.step()

                # Evaluate the model
                model.eval()
                with torch.no_grad():
                    y_pred = model(X_test_tensor, edge_index)
                    y_pred_binary = (y_pred > 0.5).float()

                # Calculate binary accuracy
                binary_acc = (y_pred_binary == y_test_tensor).float().mean().item()

                # Calculate normal accuracy
                normal_acc = accuracy_score(y_test, y_pred_binary.numpy())

                # Calculate F1 score
                f1 = f1_score(y_test, y_pred_binary.numpy(), average='micro')

                # Update best params if this model is better
```

```python
                if binary_acc > best_binary_acc:
                    best_binary_acc = binary_acc
                    best_f1 = f1
                    best_normal_acc = normal_acc
                    best_params = {
                        'hidden_dim': hidden_dim,
                        'dropout': dropout,
                        'lr': lr,
                        'batch_size': batch_size
                    }

# Print the best results
print(f'Best Params: {best_params}')
print(f'Best Binary Accuracy: {best_binary_acc:.4f}')
print(f'Best Normal Accuracy: {best_normal_acc:.4f}')
print(f'Best F1 Score: {best_f1:.4f}')
```

```
Training with hidden_dim=64, dropout=0.3, lr=0.01, batch_size=32
Training with hidden_dim=64, dropout=0.3, lr=0.01, batch_size=64
Training with hidden_dim=64, dropout=0.3, lr=0.001, batch_size=32
Training with hidden_dim=64, dropout=0.3, lr=0.001, batch_size=64
Training with hidden_dim=64, dropout=0.5, lr=0.01, batch_size=32
Training with hidden_dim=64, dropout=0.5, lr=0.01, batch_size=64
Training with hidden_dim=64, dropout=0.5, lr=0.001, batch_size=32
Training with hidden_dim=64, dropout=0.5, lr=0.001, batch_size=64
Training with hidden_dim=128, dropout=0.3, lr=0.01, batch_size=32
Training with hidden_dim=128, dropout=0.3, lr=0.01, batch_size=64
Training with hidden_dim=128, dropout=0.3, lr=0.001, batch_size=32
Training with hidden_dim=128, dropout=0.3, lr=0.001, batch_size=64
Training with hidden_dim=128, dropout=0.5, lr=0.01, batch_size=32
Training with hidden_dim=128, dropout=0.5, lr=0.01, batch_size=64
Training with hidden_dim=128, dropout=0.5, lr=0.001, batch_size=32
Training with hidden_dim=128, dropout=0.5, lr=0.001, batch_size=64
Best Params: {'hidden_dim': 128, 'dropout': 0.5, 'lr': 0.01, 'batch_size': 64}
Best Binary Accuracy: 0.9129
Best Normal Accuracy: 0.2959
Best F1 Score: 0.9544
```

[ ]:

```python
# Full GNN code with hyperparameter tuning and self-loop edge_index

import torch
import torch.nn.functional as F
from torch_geometric.nn import GCNConv
from sklearn.metrics import accuracy_score, f1_score
from sklearn.model_selection import train_test_split
import numpy as np
```

```python
# Assuming X and y are defined as molecular fingerprints (X) and labels (y)
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=42)

# Convert X_train and y_train to PyTorch tensors
X_train_tensor = torch.tensor(X_train, dtype=torch.float)
X_test_tensor = torch.tensor(X_test, dtype=torch.float)
y_train_tensor = torch.tensor(y_train, dtype=torch.float)
y_test_tensor = torch.tensor(y_test, dtype=torch.float)

# GNN model definition
class GCN(torch.nn.Module):
    def __init__(self, input_dim, hidden_dim, output_dim, dropout=0.5):
        super(GCN, self).__init__()
        self.conv1 = GCNConv(input_dim, hidden_dim)
        self.conv2 = GCNConv(hidden_dim, hidden_dim)
        self.fc = torch.nn.Linear(hidden_dim, output_dim)
        self.dropout = dropout

    def forward(self, x, edge_index):
        x = self.conv1(x, edge_index)
        x = F.relu(x)
        x = self.conv2(x, edge_index)
        x = F.relu(x)
        x = F.dropout(x, p=self.dropout, training=self.training)
        x = self.fc(x)
        return torch.sigmoid(x)  # For multi-label classification

# Hyperparameter grid
param_grid = {
    'hidden_dim': [64, 128],  # Number of hidden units
    'dropout': [0.3, 0.5],  # Dropout rate
    'epochs': [10,15, 20,25, 30,35,50,100],  # Different epoch values
    'batch_size': [32, 64],  # Batch size
    'learning_rate': [0.01, 0.001]  # Learning rates
}

# Create a self-loop graph (each node connected to itself)
num_nodes = X_train_tensor.size(0)
edge_index = torch.arange(0, num_nodes, dtype=torch.long).unsqueeze(0).repeat(2, 1)

# Variables to track the best model
best_binary_acc = 0
best_f1 = 0
best_normal_acc = 0
best_params = {}
```

```
# Perform hyperparameter tuning
for hidden_dim in param_grid['hidden_dim']:
    for dropout in param_grid['dropout']:
        for lr in param_grid['learning_rate']:
            for batch_size in param_grid['batch_size']:
                for epochs in param_grid['epochs']:  # Loop for different epoch␣
↪values
                    print(f'Training with hidden_dim={hidden_dim},␣
↪dropout={dropout}, lr={lr}, batch_size={batch_size}, epochs={epochs}')

                    # Initialize model
                    model = GCN(input_dim=X_train.shape[1],␣
↪hidden_dim=hidden_dim, output_dim=y_train.shape[1], dropout=dropout)

                    # Loss and optimizer
                    optimizer = torch.optim.Adam(model.parameters(), lr=lr)
                    criterion = torch.nn.BCELoss()

                    # Train the model
                    for epoch in range(epochs):
                        model.train()
                        optimizer.zero_grad()
                        out = model(X_train_tensor, edge_index)
                        loss = criterion(out, y_train_tensor)
                        loss.backward()
                        optimizer.step()

                    # Evaluate the model
                    model.eval()
                    with torch.no_grad():
                        y_pred = model(X_test_tensor, edge_index)
                        y_pred_binary = (y_pred > 0.5).float()

                    # Calculate binary accuracy
                    binary_acc = (y_pred_binary == y_test_tensor).float().
↪mean().item()

                    # Calculate normal accuracy
                    normal_acc = accuracy_score(y_test, y_pred_binary.numpy())

                    # Calculate F1 score
                    f1 = f1_score(y_test, y_pred_binary.numpy(),␣
↪average='micro')

                    # Update best params if this model is better
                    if binary_acc > best_binary_acc:
```

```
                        best_binary_acc = binary_acc
                        best_f1 = f1
                        best_normal_acc = normal_acc
                        best_params = {
                            'hidden_dim': hidden_dim,
                            'dropout': dropout,
                            'lr': lr,
                            'batch_size': batch_size,
                            'epochs': epochs
                        }

# Print the best results
print(f'Best Params: {best_params}')
print(f'Best Binary Accuracy: {best_binary_acc:.4f}')
print(f'Best Normal Accuracy: {best_normal_acc:.4f}')
print(f'Best F1 Score: {best_f1:.4f}')
```

```
Training with hidden_dim=64, dropout=0.3, lr=0.01, batch_size=32, epochs=10
Training with hidden_dim=64, dropout=0.3, lr=0.01, batch_size=32, epochs=15
Training with hidden_dim=64, dropout=0.3, lr=0.01, batch_size=32, epochs=20
Training with hidden_dim=64, dropout=0.3, lr=0.01, batch_size=32, epochs=25
Training with hidden_dim=64, dropout=0.3, lr=0.01, batch_size=32, epochs=30
Training with hidden_dim=64, dropout=0.3, lr=0.01, batch_size=32, epochs=35
Training with hidden_dim=64, dropout=0.3, lr=0.01, batch_size=32, epochs=50
Training with hidden_dim=64, dropout=0.3, lr=0.01, batch_size=32, epochs=100
Training with hidden_dim=64, dropout=0.3, lr=0.01, batch_size=64, epochs=10
Training with hidden_dim=64, dropout=0.3, lr=0.01, batch_size=64, epochs=15
Training with hidden_dim=64, dropout=0.3, lr=0.01, batch_size=64, epochs=20
Training with hidden_dim=64, dropout=0.3, lr=0.01, batch_size=64, epochs=25
Training with hidden_dim=64, dropout=0.3, lr=0.01, batch_size=64, epochs=30
Training with hidden_dim=64, dropout=0.3, lr=0.01, batch_size=64, epochs=35
Training with hidden_dim=64, dropout=0.3, lr=0.01, batch_size=64, epochs=50
Training with hidden_dim=64, dropout=0.3, lr=0.01, batch_size=64, epochs=100
Training with hidden_dim=64, dropout=0.3, lr=0.001, batch_size=32, epochs=10
Training with hidden_dim=64, dropout=0.3, lr=0.001, batch_size=32, epochs=15
Training with hidden_dim=64, dropout=0.3, lr=0.001, batch_size=32, epochs=20
Training with hidden_dim=64, dropout=0.3, lr=0.001, batch_size=32, epochs=25
Training with hidden_dim=64, dropout=0.3, lr=0.001, batch_size=32, epochs=30
Training with hidden_dim=64, dropout=0.3, lr=0.001, batch_size=32, epochs=35
Training with hidden_dim=64, dropout=0.3, lr=0.001, batch_size=32, epochs=50
Training with hidden_dim=64, dropout=0.3, lr=0.001, batch_size=32, epochs=100
Training with hidden_dim=64, dropout=0.3, lr=0.001, batch_size=64, epochs=10
Training with hidden_dim=64, dropout=0.3, lr=0.001, batch_size=64, epochs=15
Training with hidden_dim=64, dropout=0.3, lr=0.001, batch_size=64, epochs=20
Training with hidden_dim=64, dropout=0.3, lr=0.001, batch_size=64, epochs=25
Training with hidden_dim=64, dropout=0.3, lr=0.001, batch_size=64, epochs=30
Training with hidden_dim=64, dropout=0.3, lr=0.001, batch_size=64, epochs=35
```

```
Training with hidden_dim=64, dropout=0.3, lr=0.001, batch_size=64, epochs=50
Training with hidden_dim=64, dropout=0.3, lr=0.001, batch_size=64, epochs=100
Training with hidden_dim=64, dropout=0.5, lr=0.01, batch_size=32, epochs=10
Training with hidden_dim=64, dropout=0.5, lr=0.01, batch_size=32, epochs=15
Training with hidden_dim=64, dropout=0.5, lr=0.01, batch_size=32, epochs=20
Training with hidden_dim=64, dropout=0.5, lr=0.01, batch_size=32, epochs=25
Training with hidden_dim=64, dropout=0.5, lr=0.01, batch_size=32, epochs=30
Training with hidden_dim=64, dropout=0.5, lr=0.01, batch_size=32, epochs=35
Training with hidden_dim=64, dropout=0.5, lr=0.01, batch_size=32, epochs=50
Training with hidden_dim=64, dropout=0.5, lr=0.01, batch_size=32, epochs=100
Training with hidden_dim=64, dropout=0.5, lr=0.01, batch_size=64, epochs=10
Training with hidden_dim=64, dropout=0.5, lr=0.01, batch_size=64, epochs=15
Training with hidden_dim=64, dropout=0.5, lr=0.01, batch_size=64, epochs=20
Training with hidden_dim=64, dropout=0.5, lr=0.01, batch_size=64, epochs=25
Training with hidden_dim=64, dropout=0.5, lr=0.01, batch_size=64, epochs=30
Training with hidden_dim=64, dropout=0.5, lr=0.01, batch_size=64, epochs=35
Training with hidden_dim=64, dropout=0.5, lr=0.01, batch_size=64, epochs=50
Training with hidden_dim=64, dropout=0.5, lr=0.01, batch_size=64, epochs=100
Training with hidden_dim=64, dropout=0.5, lr=0.001, batch_size=32, epochs=10
Training with hidden_dim=64, dropout=0.5, lr=0.001, batch_size=32, epochs=15
Training with hidden_dim=64, dropout=0.5, lr=0.001, batch_size=32, epochs=20
Training with hidden_dim=64, dropout=0.5, lr=0.001, batch_size=32, epochs=25
Training with hidden_dim=64, dropout=0.5, lr=0.001, batch_size=32, epochs=30
Training with hidden_dim=64, dropout=0.5, lr=0.001, batch_size=32, epochs=35
Training with hidden_dim=64, dropout=0.5, lr=0.001, batch_size=32, epochs=50
Training with hidden_dim=64, dropout=0.5, lr=0.001, batch_size=32, epochs=100
Training with hidden_dim=64, dropout=0.5, lr=0.001, batch_size=64, epochs=10
Training with hidden_dim=64, dropout=0.5, lr=0.001, batch_size=64, epochs=15
Training with hidden_dim=64, dropout=0.5, lr=0.001, batch_size=64, epochs=20
Training with hidden_dim=64, dropout=0.5, lr=0.001, batch_size=64, epochs=25
Training with hidden_dim=64, dropout=0.5, lr=0.001, batch_size=64, epochs=30
Training with hidden_dim=64, dropout=0.5, lr=0.001, batch_size=64, epochs=35
Training with hidden_dim=64, dropout=0.5, lr=0.001, batch_size=64, epochs=50
Training with hidden_dim=64, dropout=0.5, lr=0.001, batch_size=64, epochs=100
Training with hidden_dim=128, dropout=0.3, lr=0.01, batch_size=32, epochs=10
Training with hidden_dim=128, dropout=0.3, lr=0.01, batch_size=32, epochs=15
Training with hidden_dim=128, dropout=0.3, lr=0.01, batch_size=32, epochs=20
Training with hidden_dim=128, dropout=0.3, lr=0.01, batch_size=32, epochs=25
Training with hidden_dim=128, dropout=0.3, lr=0.01, batch_size=32, epochs=30
Training with hidden_dim=128, dropout=0.3, lr=0.01, batch_size=32, epochs=35
Training with hidden_dim=128, dropout=0.3, lr=0.01, batch_size=32, epochs=50
Training with hidden_dim=128, dropout=0.3, lr=0.01, batch_size=32, epochs=100
Training with hidden_dim=128, dropout=0.3, lr=0.01, batch_size=64, epochs=10
Training with hidden_dim=128, dropout=0.3, lr=0.01, batch_size=64, epochs=15
Training with hidden_dim=128, dropout=0.3, lr=0.01, batch_size=64, epochs=20
Training with hidden_dim=128, dropout=0.3, lr=0.01, batch_size=64, epochs=25
Training with hidden_dim=128, dropout=0.3, lr=0.01, batch_size=64, epochs=30
Training with hidden_dim=128, dropout=0.3, lr=0.01, batch_size=64, epochs=35
```

```
Training with hidden_dim=128, dropout=0.3, lr=0.01, batch_size=64, epochs=50
Training with hidden_dim=128, dropout=0.3, lr=0.01, batch_size=64, epochs=100
Training with hidden_dim=128, dropout=0.3, lr=0.001, batch_size=32, epochs=10
Training with hidden_dim=128, dropout=0.3, lr=0.001, batch_size=32, epochs=15
Training with hidden_dim=128, dropout=0.3, lr=0.001, batch_size=32, epochs=20
Training with hidden_dim=128, dropout=0.3, lr=0.001, batch_size=32, epochs=25
Training with hidden_dim=128, dropout=0.3, lr=0.001, batch_size=32, epochs=30
Training with hidden_dim=128, dropout=0.3, lr=0.001, batch_size=32, epochs=35
Training with hidden_dim=128, dropout=0.3, lr=0.001, batch_size=32, epochs=50
Training with hidden_dim=128, dropout=0.3, lr=0.001, batch_size=32, epochs=100
Training with hidden_dim=128, dropout=0.3, lr=0.001, batch_size=64, epochs=10
Training with hidden_dim=128, dropout=0.3, lr=0.001, batch_size=64, epochs=15
Training with hidden_dim=128, dropout=0.3, lr=0.001, batch_size=64, epochs=20
Training with hidden_dim=128, dropout=0.3, lr=0.001, batch_size=64, epochs=25
Training with hidden_dim=128, dropout=0.3, lr=0.001, batch_size=64, epochs=30
Training with hidden_dim=128, dropout=0.3, lr=0.001, batch_size=64, epochs=35
Training with hidden_dim=128, dropout=0.3, lr=0.001, batch_size=64, epochs=50
Training with hidden_dim=128, dropout=0.3, lr=0.001, batch_size=64, epochs=100
Training with hidden_dim=128, dropout=0.5, lr=0.01, batch_size=32, epochs=10
Training with hidden_dim=128, dropout=0.5, lr=0.01, batch_size=32, epochs=15
Training with hidden_dim=128, dropout=0.5, lr=0.01, batch_size=32, epochs=20
Training with hidden_dim=128, dropout=0.5, lr=0.01, batch_size=32, epochs=25
Training with hidden_dim=128, dropout=0.5, lr=0.01, batch_size=32, epochs=30
Training with hidden_dim=128, dropout=0.5, lr=0.01, batch_size=32, epochs=35
Training with hidden_dim=128, dropout=0.5, lr=0.01, batch_size=32, epochs=50
Training with hidden_dim=128, dropout=0.5, lr=0.01, batch_size=32, epochs=100
Training with hidden_dim=128, dropout=0.5, lr=0.01, batch_size=64, epochs=10
Training with hidden_dim=128, dropout=0.5, lr=0.01, batch_size=64, epochs=15
Training with hidden_dim=128, dropout=0.5, lr=0.01, batch_size=64, epochs=20
Training with hidden_dim=128, dropout=0.5, lr=0.01, batch_size=64, epochs=25
Training with hidden_dim=128, dropout=0.5, lr=0.01, batch_size=64, epochs=30
Training with hidden_dim=128, dropout=0.5, lr=0.01, batch_size=64, epochs=35
Training with hidden_dim=128, dropout=0.5, lr=0.01, batch_size=64, epochs=50
Training with hidden_dim=128, dropout=0.5, lr=0.01, batch_size=64, epochs=100
Training with hidden_dim=128, dropout=0.5, lr=0.001, batch_size=32, epochs=10
Training with hidden_dim=128, dropout=0.5, lr=0.001, batch_size=32, epochs=15
Training with hidden_dim=128, dropout=0.5, lr=0.001, batch_size=32, epochs=20
Training with hidden_dim=128, dropout=0.5, lr=0.001, batch_size=32, epochs=25
Training with hidden_dim=128, dropout=0.5, lr=0.001, batch_size=32, epochs=30
Training with hidden_dim=128, dropout=0.5, lr=0.001, batch_size=32, epochs=35
Training with hidden_dim=128, dropout=0.5, lr=0.001, batch_size=32, epochs=50
Training with hidden_dim=128, dropout=0.5, lr=0.001, batch_size=32, epochs=100
Training with hidden_dim=128, dropout=0.5, lr=0.001, batch_size=64, epochs=10
Training with hidden_dim=128, dropout=0.5, lr=0.001, batch_size=64, epochs=15
Training with hidden_dim=128, dropout=0.5, lr=0.001, batch_size=64, epochs=20
Training with hidden_dim=128, dropout=0.5, lr=0.001, batch_size=64, epochs=25
Training with hidden_dim=128, dropout=0.5, lr=0.001, batch_size=64, epochs=30
Training with hidden_dim=128, dropout=0.5, lr=0.001, batch_size=64, epochs=35
```

```
Training with hidden_dim=128, dropout=0.5, lr=0.001, batch_size=64, epochs=50
Training with hidden_dim=128, dropout=0.5, lr=0.001, batch_size=64, epochs=100
Best Params: {'hidden_dim': 64, 'dropout': 0.3, 'lr': 0.01, 'batch_size': 64,
'epochs': 35}
Best Binary Accuracy: 0.9126
Best Normal Accuracy: 0.2884
Best F1 Score: 0.9541
```

[ ]:
```
# OPTIMIZATION APPROACHES
```

[ ]:
```
# 1. SOFT VOTING

# Import necessary libraries
import numpy as np
import torch
import torch.nn.functional as F
from torch_geometric.nn import GCNConv
from sklearn.model_selection import train_test_split
from sklearn.metrics import accuracy_score, f1_score
from tensorflow.keras.models import Sequential
from tensorflow.keras.layers import Conv1D, MaxPooling1D, Dense, Flatten,␣
↪Dropout, LSTM
from tensorflow.keras.optimizers import Adam
from tensorflow.keras.regularizers import l2

# Assuming you already have X and y prepared
# X represents molecular fingerprints, y represents the reaction labels
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2,␣
↪random_state=42)

# Convert data to PyTorch tensors for GNN
X_train_tensor = torch.tensor(X_train, dtype=torch.float)
X_test_tensor = torch.tensor(X_test, dtype=torch.float)
y_train_tensor = torch.tensor(y_train, dtype=torch.float)

# Simulate edge connections for GNN (as a placeholder)
edge_index = torch.tensor([[0, 1], [1, 0]], dtype=torch.long)

###############################################
# CNN Model Definition and Training
###############################################
def build_cnn_model():
    model = Sequential()
    model.add(Conv1D(filters=64, kernel_size=3, activation='relu',␣
↪input_shape=(X_train.shape[1], 1)))
    model.add(MaxPooling1D(pool_size=2))
    model.add(Dropout(0.5))
```

```
    model.add(Flatten())
    model.add(Dense(256, activation='relu'))
    model.add(Dropout(0.5))
    model.add(Dense(y_train.shape[1], activation='sigmoid'))  # Multi-label␣
↪classification
    model.compile(optimizer=Adam(), loss='binary_crossentropy',␣
↪metrics=['binary_accuracy'])
    return model

# Reshape data for CNN input
X_train_cnn = X_train.reshape((X_train.shape[0], X_train.shape[1], 1))
X_test_cnn = X_test.reshape((X_test.shape[0], X_test.shape[1], 1))

# Train CNN model
cnn_model = build_cnn_model()
cnn_model.fit(X_train_cnn, y_train, epochs=10, batch_size=64,␣
↪validation_split=0.1)
cnn_pred = cnn_model.predict(X_test_cnn)

###############################################
# LSTM Model Definition and Training
###############################################
def build_lstm_model():
    model = Sequential()
    model.add(LSTM(64, return_sequences=False, input_shape=(X_train.shape[1],␣
↪1), kernel_regularizer=l2(0.01)))
    model.add(Dropout(0.3))
    model.add(Dense(256, activation='relu'))
    model.add(Dense(y_train.shape[1], activation='sigmoid'))  # Multi-label␣
↪classification
    model.compile(optimizer=Adam(), loss='binary_crossentropy',␣
↪metrics=['binary_accuracy'])
    return model

# Reshape data for LSTM input
X_train_lstm = X_train.reshape((X_train.shape[0], X_train.shape[1], 1))
X_test_lstm = X_test.reshape((X_test.shape[0], X_test.shape[1], 1))

# Train LSTM model
lstm_model = build_lstm_model()
lstm_model.fit(X_train_lstm, y_train, epochs=10, batch_size=32,␣
↪validation_split=0.1)
lstm_pred = lstm_model.predict(X_test_lstm)

###############################################
# GNN Model Definition and Training
```

```python
###############################################
class GCN(torch.nn.Module):
    def __init__(self, input_dim, hidden_dim, output_dim, dropout=0.5):
        super(GCN, self).__init__()
        self.conv1 = GCNConv(input_dim, hidden_dim)
        self.conv2 = GCNConv(hidden_dim, hidden_dim)
        self.fc = torch.nn.Linear(hidden_dim, output_dim)
        self.dropout = dropout

    def forward(self, x, edge_index):
        x = self.conv1(x, edge_index)
        x = F.relu(x)
        x = self.conv2(x, edge_index)
        x = F.relu(x)
        x = F.dropout(x, p=self.dropout, training=self.training)
        x = self.fc(x)
        return torch.sigmoid(x)

# Initialize GNN model
gnn_model = GCN(input_dim=X_train.shape[1], hidden_dim=128, output_dim=y_train.
 ↪shape[1], dropout=0.5)
optimizer = torch.optim.Adam(gnn_model.parameters(), lr=0.01)
criterion = torch.nn.BCELoss()

# Train GNN model
for epoch in range(30):
    gnn_model.train()
    optimizer.zero_grad()
    out = gnn_model(X_train_tensor, edge_index)
    loss = criterion(out, y_train_tensor)
    loss.backward()
    optimizer.step()

# Get GNN predictions
gnn_model.eval()
with torch.no_grad():
    gnn_pred = gnn_model(X_test_tensor, edge_index).numpy()

###############################################
# Soft Voting (Combining Predictions)
###############################################
# Perform Soft Voting
combined_pred = (cnn_pred + lstm_pred + gnn_pred) / 3

# Convert to binary predictions
combined_pred_binary = (combined_pred > 0.5).astype(int)
```

```python
# Evaluate the combined predictions
binary_acc = np.mean(np.equal(y_test, combined_pred_binary).astype(int))
f1 = f1_score(y_test, combined_pred_binary, average='micro')
normal_acc = accuracy_score(y_test, combined_pred_binary)

# Print the results
print(f'Soft Voting Binary Accuracy: {binary_acc:.4f}')
print(f'Soft Voting F1 Score: {f1:.4f}')
print(f'Soft Voting Normal Accuracy: {normal_acc:.4f}')
```

```
Epoch 1/10
```

```
/usr/local/lib/python3.10/dist-
packages/keras/src/layers/convolutional/base_conv.py:107: UserWarning: Do not
pass an `input_shape`/`input_dim` argument to a layer. When using Sequential
models, prefer using an `Input(shape)` object as the first layer in the model
instead.
  super().__init__(activity_regularizer=activity_regularizer, **kwargs)
```

```
15/15           7s 194ms/step -
binary_accuracy: 0.7868 - loss: 0.4658 - val_binary_accuracy: 0.9124 - val_loss:
0.2611
Epoch 2/10
15/15           4s 16ms/step -
binary_accuracy: 0.8816 - loss: 0.3234 - val_binary_accuracy: 0.9108 - val_loss:
0.2626
Epoch 3/10
15/15           0s 15ms/step -
binary_accuracy: 0.8891 - loss: 0.2858 - val_binary_accuracy: 0.9116 - val_loss:
0.2499
Epoch 4/10
15/15           0s 11ms/step -
binary_accuracy: 0.9004 - loss: 0.2517 - val_binary_accuracy: 0.9116 - val_loss:
0.2447
Epoch 5/10
15/15           0s 12ms/step -
binary_accuracy: 0.9049 - loss: 0.2353 - val_binary_accuracy: 0.9065 - val_loss:
0.2464
Epoch 6/10
15/15           0s 12ms/step -
binary_accuracy: 0.9114 - loss: 0.2159 - val_binary_accuracy: 0.9054 - val_loss:
0.2493
Epoch 7/10
15/15           0s 11ms/step -
binary_accuracy: 0.9154 - loss: 0.2037 - val_binary_accuracy: 0.9065 - val_loss:
0.2514
Epoch 8/10
15/15           0s 11ms/step -
```

```
binary_accuracy: 0.9179 - loss: 0.1963 - val_binary_accuracy: 0.9054 - val_loss:
0.2558
Epoch 9/10
15/15            0s 10ms/step -
binary_accuracy: 0.9148 - loss: 0.1965 - val_binary_accuracy: 0.9003 - val_loss:
0.2634
Epoch 10/10
15/15            0s 11ms/step -
binary_accuracy: 0.9225 - loss: 0.1788 - val_binary_accuracy: 0.9034 - val_loss:
0.2724
9/9            1s 39ms/step
```

```
/usr/local/lib/python3.10/dist-packages/keras/src/layers/rnn/rnn.py:204:
UserWarning: Do not pass an `input_shape`/`input_dim` argument to a layer. When
using Sequential models, prefer using an `Input(shape)` object as the first
layer in the model instead.
  super().__init__(**kwargs)
```

```
Epoch 1/10
30/30            6s 50ms/step -
binary_accuracy: 0.8266 - loss: 0.6305 - val_binary_accuracy: 0.9124 - val_loss:
0.2562
Epoch 2/10
30/30            1s 40ms/step -
binary_accuracy: 0.8949 - loss: 0.2823 - val_binary_accuracy: 0.9124 - val_loss:
0.2429
Epoch 3/10
30/30            1s 40ms/step -
binary_accuracy: 0.8965 - loss: 0.2745 - val_binary_accuracy: 0.9124 - val_loss:
0.2383
Epoch 4/10
30/30            1s 31ms/step -
binary_accuracy: 0.8962 - loss: 0.2688 - val_binary_accuracy: 0.9124 - val_loss:
0.2377
Epoch 5/10
30/30            1s 32ms/step -
binary_accuracy: 0.9044 - loss: 0.2522 - val_binary_accuracy: 0.9124 - val_loss:
0.2395
Epoch 6/10
30/30            1s 32ms/step -
binary_accuracy: 0.9011 - loss: 0.2591 - val_binary_accuracy: 0.9124 - val_loss:
0.2436
Epoch 7/10
30/30            1s 32ms/step -
binary_accuracy: 0.8990 - loss: 0.2669 - val_binary_accuracy: 0.9124 - val_loss:
0.2365
Epoch 8/10
30/30            1s 32ms/step -
binary_accuracy: 0.9030 - loss: 0.2561 - val_binary_accuracy: 0.9124 - val_loss:
```

```
0.2387
Epoch 9/10
30/30                  1s 32ms/step -
binary_accuracy: 0.9026 - loss: 0.2528 - val_binary_accuracy: 0.9124 - val_loss:
0.2394
Epoch 10/10
30/30                  1s 31ms/step -
binary_accuracy: 0.9013 - loss: 0.2542 - val_binary_accuracy: 0.9124 - val_loss:
0.2378
9/9                0s 22ms/step
Soft Voting Binary Accuracy: 0.9131
Soft Voting F1 Score: 0.9544
Soft Voting Normal Accuracy: 0.2884
```

[ ]:

```python
# WEIGHTED SOFT VOTING

from sklearn.metrics import accuracy_score, f1_score
from itertools import product
import numpy as np

# Define ranges for the weights (can fine-tune these ranges based on your␣
↪preference)
weight_range = np.arange(0.1, 1.1, 0.1)

# Initialize variables to track the best performance
best_binary_acc = 0
best_f1 = 0
best_normal_acc = 0
best_weights = None

# Perform grid search for weights w1 (CNN), w2 (LSTM), w3 (GNN)
for w1, w2, w3 in product(weight_range, repeat=3):
    # Weighted soft voting
    combined_pred = (w1 * cnn_pred + w2 * lstm_pred + w3 * gnn_pred) / (w1 + w2␣
↪+ w3)

    # Convert to binary predictions
    combined_pred_binary = (combined_pred > 0.5).astype(int)

    # Calculate binary accuracy
    binary_acc = np.mean(np.equal(y_test, combined_pred_binary).astype(int))

    # Calculate normal accuracy
    normal_acc = accuracy_score(y_test, combined_pred_binary)

    # Calculate F1 score
    f1 = f1_score(y_test, combined_pred_binary, average='micro')
```

```
    # Update the best weights if this configuration performs better
    if binary_acc > best_binary_acc:
        best_binary_acc = binary_acc
        best_f1 = f1
        best_normal_acc = normal_acc
        best_weights = (w1, w2, w3)

# Print the best results
print(f'Best Weights: CNN={best_weights[0]}, LSTM={best_weights[1]}, GNN={best_weights[2]}')
print(f'Best Binary Accuracy: {best_binary_acc:.4f}')
print(f'Best Normal Accuracy: {best_normal_acc:.4f}')
print(f'Best F1 Score: {best_f1:.4f}')
```

```
Best Weights: CNN=0.4, LSTM=0.30000000000000004, GNN=0.5
Best Binary Accuracy: 0.9137
Best Normal Accuracy: 0.2921
Best F1 Score: 0.9547
```

[ ]:
```
# NEW START
```

[ ]:
```
# 1. ResNet CNN - with hyperparameter tuning
```

[ ]:
```
!pip install keras-tuner tensorflow
```

```
Collecting keras-tuner
  Downloading keras_tuner-1.4.7-py3-none-any.whl.metadata (5.4 kB)
Requirement already satisfied: tensorflow in /usr/local/lib/python3.10/dist-
packages (2.17.0)
Requirement already satisfied: keras in /usr/local/lib/python3.10/dist-packages
(from keras-tuner) (3.4.1)
Requirement already satisfied: packaging in /usr/local/lib/python3.10/dist-
packages (from keras-tuner) (24.1)
Requirement already satisfied: requests in /usr/local/lib/python3.10/dist-
packages (from keras-tuner) (2.32.3)
Collecting kt-legacy (from keras-tuner)
  Downloading kt_legacy-1.0.5-py3-none-any.whl.metadata (221 bytes)
Requirement already satisfied: absl-py>=1.0.0 in /usr/local/lib/python3.10/dist-
packages (from tensorflow) (1.4.0)
Requirement already satisfied: astunparse>=1.6.0 in
/usr/local/lib/python3.10/dist-packages (from tensorflow) (1.6.3)
Requirement already satisfied: flatbuffers>=24.3.25 in
/usr/local/lib/python3.10/dist-packages (from tensorflow) (24.3.25)
Requirement already satisfied: gast!=0.5.0,!=0.5.1,!=0.5.2,>=0.2.1 in
/usr/local/lib/python3.10/dist-packages (from tensorflow) (0.6.0)
Requirement already satisfied: google-pasta>=0.1.1 in
```

```
/usr/local/lib/python3.10/dist-packages (from tensorflow) (0.2.0)
Requirement already satisfied: h5py>=3.10.0 in /usr/local/lib/python3.10/dist-
packages (from tensorflow) (3.11.0)
Requirement already satisfied: libclang>=13.0.0 in
/usr/local/lib/python3.10/dist-packages (from tensorflow) (18.1.1)
Requirement already satisfied: ml-dtypes<0.5.0,>=0.3.1 in
/usr/local/lib/python3.10/dist-packages (from tensorflow) (0.4.0)
Requirement already satisfied: opt-einsum>=2.3.2 in
/usr/local/lib/python3.10/dist-packages (from tensorflow) (3.3.0)
Requirement already satisfied:
protobuf!=4.21.0,!=4.21.1,!=4.21.2,!=4.21.3,!=4.21.4,!=4.21.5,<5.0.0dev,>=3.20.3
in /usr/local/lib/python3.10/dist-packages (from tensorflow) (3.20.3)
Requirement already satisfied: setuptools in /usr/local/lib/python3.10/dist-
packages (from tensorflow) (71.0.4)
Requirement already satisfied: six>=1.12.0 in /usr/local/lib/python3.10/dist-
packages (from tensorflow) (1.16.0)
Requirement already satisfied: termcolor>=1.1.0 in
/usr/local/lib/python3.10/dist-packages (from tensorflow) (2.4.0)
Requirement already satisfied: typing-extensions>=3.6.6 in
/usr/local/lib/python3.10/dist-packages (from tensorflow) (4.12.2)
Requirement already satisfied: wrapt>=1.11.0 in /usr/local/lib/python3.10/dist-
packages (from tensorflow) (1.16.0)
Requirement already satisfied: grpcio<2.0,>=1.24.3 in
/usr/local/lib/python3.10/dist-packages (from tensorflow) (1.64.1)
Requirement already satisfied: tensorboard<2.18,>=2.17 in
/usr/local/lib/python3.10/dist-packages (from tensorflow) (2.17.0)
Requirement already satisfied: tensorflow-io-gcs-filesystem>=0.23.1 in
/usr/local/lib/python3.10/dist-packages (from tensorflow) (0.37.1)
Requirement already satisfied: numpy<2.0.0,>=1.23.5 in
/usr/local/lib/python3.10/dist-packages (from tensorflow) (1.26.4)
Requirement already satisfied: wheel<1.0,>=0.23.0 in
/usr/local/lib/python3.10/dist-packages (from astunparse>=1.6.0->tensorflow)
(0.44.0)
Requirement already satisfied: rich in /usr/local/lib/python3.10/dist-packages
(from keras->keras-tuner) (13.8.1)
Requirement already satisfied: namex in /usr/local/lib/python3.10/dist-packages
(from keras->keras-tuner) (0.0.8)
Requirement already satisfied: optree in /usr/local/lib/python3.10/dist-packages
(from keras->keras-tuner) (0.12.1)
Requirement already satisfied: charset-normalizer<4,>=2 in
/usr/local/lib/python3.10/dist-packages (from requests->keras-tuner) (3.3.2)
Requirement already satisfied: idna<4,>=2.5 in /usr/local/lib/python3.10/dist-
packages (from requests->keras-tuner) (3.8)
Requirement already satisfied: urllib3<3,>=1.21.1 in
/usr/local/lib/python3.10/dist-packages (from requests->keras-tuner) (2.0.7)
Requirement already satisfied: certifi>=2017.4.17 in
/usr/local/lib/python3.10/dist-packages (from requests->keras-tuner) (2024.8.30)
Requirement already satisfied: markdown>=2.6.8 in
```

```
/usr/local/lib/python3.10/dist-packages (from
tensorboard<2.18,>=2.17->tensorflow) (3.7)
Requirement already satisfied: tensorboard-data-server<0.8.0,>=0.7.0 in
/usr/local/lib/python3.10/dist-packages (from
tensorboard<2.18,>=2.17->tensorflow) (0.7.2)
Requirement already satisfied: werkzeug>=1.0.1 in
/usr/local/lib/python3.10/dist-packages (from
tensorboard<2.18,>=2.17->tensorflow) (3.0.4)
Requirement already satisfied: MarkupSafe>=2.1.1 in
/usr/local/lib/python3.10/dist-packages (from
werkzeug>=1.0.1->tensorboard<2.18,>=2.17->tensorflow) (2.1.5)
Requirement already satisfied: markdown-it-py>=2.2.0 in
/usr/local/lib/python3.10/dist-packages (from rich->keras->keras-tuner) (3.0.0)
Requirement already satisfied: pygments<3.0.0,>=2.13.0 in
/usr/local/lib/python3.10/dist-packages (from rich->keras->keras-tuner) (2.16.1)
Requirement already satisfied: mdurl~=0.1 in /usr/local/lib/python3.10/dist-
packages (from markdown-it-py>=2.2.0->rich->keras->keras-tuner) (0.1.2)
Downloading keras_tuner-1.4.7-py3-none-any.whl (129 kB)
                         129.1/129.1 kB
4.1 MB/s eta 0:00:00
Downloading kt_legacy-1.0.5-py3-none-any.whl (9.6 kB)
Installing collected packages: kt-legacy, keras-tuner
Successfully installed keras-tuner-1.4.7 kt-legacy-1.0.5
```

[ ]:

```python
import tensorflow as tf
from tensorflow.keras.layers import Conv1D, MaxPooling1D,␣
↪GlobalAveragePooling1D, Dense, Dropout, BatchNormalization
from keras_tuner import RandomSearch
from sklearn.metrics import accuracy_score, f1_score

# Custom ResNet-inspired 1D architecture
def build_resnet_1d_model(hp):
    model = tf.keras.Sequential()

    # First convolution block
    model.add(Conv1D(filters=hp.Int('filters', min_value=32, max_value=128,␣
↪step=32),
                     kernel_size=hp.Int('kernel_size', min_value=3,␣
↪max_value=7, step=2),
                     activation='relu', input_shape=(X_train.shape[1], 1)))
    model.add(BatchNormalization())
    model.add(MaxPooling1D(pool_size=2))

    # Second convolution block
    model.add(Conv1D(filters=hp.Int('filters_2', min_value=32, max_value=128,␣
↪step=32),
```

```
                      kernel_size=hp.Int('kernel_size_2', min_value=3,␣
↪max_value=7, step=2),
                      activation='relu'))
    model.add(BatchNormalization())
    model.add(MaxPooling1D(pool_size=2))

    # Global Average Pooling and Dense layers
    model.add(GlobalAveragePooling1D())
    model.add(Dense(units=hp.Int('units', min_value=128, max_value=512,␣
↪step=64), activation='relu'))
    model.add(Dropout(rate=hp.Float('dropout', min_value=0.1, max_value=0.5,␣
↪step=0.1)))

    # Output layer
    model.add(Dense(y_train.shape[1], activation='sigmoid'))  # Multi-label␣
↪classification

    # Compile the model
    model.compile(optimizer=tf.keras.optimizers.Adam(
        hp.Choice('learning_rate', values=[1e-2, 1e-3, 1e-4])),
        loss='binary_crossentropy',
        metrics=['binary_accuracy'])

    return model

# Reshape the data for Conv1D input
X_train_resnet = X_train.reshape((X_train.shape[0], X_train.shape[1], 1))
X_test_resnet = X_test.reshape((X_test.shape[0], X_test.shape[1], 1))

# Set up Keras Tuner for hyperparameter tuning
tuner_resnet_1d = RandomSearch(
    build_resnet_1d_model,
    objective='val_binary_accuracy',
    max_trials=10,  # Increase this for more extensive tuning
    executions_per_trial=1,
    directory='resnet_1d_tuner',
    project_name='adr_resnet_1d')

# Search for the best hyperparameters
tuner_resnet_1d.search(X_train_resnet, y_train, epochs=10, validation_split=0.2)

# Get the best hyperparameters and model
best_resnet_1d_model = tuner_resnet_1d.get_best_models(num_models=1)[0]

# Train the best model further on the dataset
best_resnet_1d_model.fit(X_train_resnet, y_train, epochs=10,␣
↪validation_data=(X_test_resnet, y_test))
```

```python
# Evaluate ResNet 1D
resnet_1d_pred = best_resnet_1d_model.predict(X_test_resnet)
resnet_1d_pred_binary = (resnet_1d_pred > 0.5).astype(int)

resnet_1d_binary_acc = np.mean(np.equal(y_test, resnet_1d_pred_binary).
 ↪astype(int))
resnet_1d_f1 = f1_score(y_test, resnet_1d_pred_binary, average='micro')
resnet_1d_normal_acc = accuracy_score(y_test, resnet_1d_pred_binary)

print(f'ResNet 1D Binary Accuracy: {resnet_1d_binary_acc:.4f}')
print(f'ResNet 1D Normal Accuracy: {resnet_1d_normal_acc:.4f}')
print(f'ResNet 1D F1 Score: {resnet_1d_f1:.4f}')
```

```
Trial 10 Complete [00h 01m 47s]
val_binary_accuracy: 0.9021908640861511

Best val_binary_accuracy So Far: 0.9021908640861511
Total elapsed time: 00h 15m 01s
Epoch 1/10
```

```
/usr/local/lib/python3.10/dist-packages/keras/src/saving/saving_lib.py:576:
UserWarning: Skipping variable loading for optimizer 'adam', because it has 2
variables whereas the saved optimizer has 26 variables.
  saveable.load_own_variables(weights_store.get(inner_path))
```

```
34/34              20s 511ms/step -
binary_accuracy: 0.8987 - loss: 0.2773 - val_binary_accuracy: 0.8830 - val_loss:
0.6108
Epoch 2/10
34/34              25s 639ms/step -
binary_accuracy: 0.8993 - loss: 0.2641 - val_binary_accuracy: 0.8407 - val_loss:
0.6043
Epoch 3/10
34/34              31s 355ms/step -
binary_accuracy: 0.8932 - loss: 0.2692 - val_binary_accuracy: 0.8366 - val_loss:
0.5691
Epoch 4/10
34/34              21s 358ms/step -
binary_accuracy: 0.8924 - loss: 0.2755 - val_binary_accuracy: 0.8258 - val_loss:
0.5258
Epoch 5/10
34/34              10s 309ms/step -
binary_accuracy: 0.8983 - loss: 0.2617 - val_binary_accuracy: 0.8204 - val_loss:
0.4950
Epoch 6/10
34/34              22s 368ms/step -
binary_accuracy: 0.9032 - loss: 0.2546 - val_binary_accuracy: 0.8297 - val_loss:
```

```
0.4888
Epoch 7/10
34/34            12s 357ms/step -
binary_accuracy: 0.8948 - loss: 0.2657 - val_binary_accuracy: 0.8407 - val_loss:
0.4485
Epoch 8/10
34/34            20s 336ms/step -
binary_accuracy: 0.9030 - loss: 0.2571 - val_binary_accuracy: 0.8641 - val_loss:
0.4398
Epoch 9/10
34/34            12s 352ms/step -
binary_accuracy: 0.9039 - loss: 0.2531 - val_binary_accuracy: 0.8628 - val_loss:
0.4327
Epoch 10/10
34/34            20s 330ms/step -
binary_accuracy: 0.8957 - loss: 0.2718 - val_binary_accuracy: 0.8619 - val_loss:
0.4446
9/9            1s 123ms/step
ResNet 1D Binary Accuracy: 0.8619
ResNet 1D Normal Accuracy: 0.0037
ResNet 1D F1 Score: 0.9240
```

[ ]:
```
# 2. GCN, GRAPHSAGE AND GAT
!pip install torch-scatter torch-sparse torch-geometric rdkit scikit-optimize
```

```
Collecting torch-scatter
  Using cached torch_scatter-2.1.2.tar.gz (108 kB)
  Preparing metadata (setup.py) … done
Collecting torch-sparse
  Using cached torch_sparse-0.6.18.tar.gz (209 kB)
  Preparing metadata (setup.py) … done
Requirement already satisfied: torch-geometric in
/usr/local/lib/python3.10/dist-packages (2.6.0)
Collecting rdkit
  Downloading rdkit-2024.3.5-cp310-cp310-manylinux_2_28_x86_64.whl.metadata (3.9
kB)
Collecting scikit-optimize
  Downloading scikit_optimize-0.10.2-py2.py3-none-any.whl.metadata (9.7 kB)
Requirement already satisfied: scipy in /usr/local/lib/python3.10/dist-packages
(from torch-sparse) (1.13.1)
Requirement already satisfied: aiohttp in /usr/local/lib/python3.10/dist-
packages (from torch-geometric) (3.10.5)
Requirement already satisfied: fsspec in /usr/local/lib/python3.10/dist-packages
(from torch-geometric) (2024.6.1)
Requirement already satisfied: jinja2 in /usr/local/lib/python3.10/dist-packages
(from torch-geometric) (3.1.4)
Requirement already satisfied: numpy in /usr/local/lib/python3.10/dist-packages
(from torch-geometric) (1.26.4)
```

```
Requirement already satisfied: psutil>=5.8.0 in /usr/local/lib/python3.10/dist-
packages (from torch-geometric) (5.9.5)
Requirement already satisfied: pyparsing in /usr/local/lib/python3.10/dist-
packages (from torch-geometric) (3.1.4)
Requirement already satisfied: requests in /usr/local/lib/python3.10/dist-
packages (from torch-geometric) (2.32.3)
Requirement already satisfied: tqdm in /usr/local/lib/python3.10/dist-packages
(from torch-geometric) (4.66.5)
Requirement already satisfied: Pillow in /usr/local/lib/python3.10/dist-packages
(from rdkit) (9.4.0)
Requirement already satisfied: joblib>=0.11 in /usr/local/lib/python3.10/dist-
packages (from scikit-optimize) (1.4.2)
Collecting pyaml>=16.9 (from scikit-optimize)
  Downloading pyaml-24.7.0-py3-none-any.whl.metadata (11 kB)
Requirement already satisfied: scikit-learn>=1.0.0 in
/usr/local/lib/python3.10/dist-packages (from scikit-optimize) (1.3.2)
Requirement already satisfied: packaging>=21.3 in
/usr/local/lib/python3.10/dist-packages (from scikit-optimize) (24.1)
Requirement already satisfied: PyYAML in /usr/local/lib/python3.10/dist-packages
(from pyaml>=16.9->scikit-optimize) (6.0.2)
Requirement already satisfied: threadpoolctl>=2.0.0 in
/usr/local/lib/python3.10/dist-packages (from scikit-learn>=1.0.0->scikit-
optimize) (3.5.0)
Requirement already satisfied: aiohappyeyeballs>=2.3.0 in
/usr/local/lib/python3.10/dist-packages (from aiohttp->torch-geometric) (2.4.0)
Requirement already satisfied: aiosignal>=1.1.2 in
/usr/local/lib/python3.10/dist-packages (from aiohttp->torch-geometric) (1.3.1)
Requirement already satisfied: attrs>=17.3.0 in /usr/local/lib/python3.10/dist-
packages (from aiohttp->torch-geometric) (24.2.0)
Requirement already satisfied: frozenlist>=1.1.1 in
/usr/local/lib/python3.10/dist-packages (from aiohttp->torch-geometric) (1.4.1)
Requirement already satisfied: multidict<7.0,>=4.5 in
/usr/local/lib/python3.10/dist-packages (from aiohttp->torch-geometric) (6.1.0)
Requirement already satisfied: yarl<2.0,>=1.0 in /usr/local/lib/python3.10/dist-
packages (from aiohttp->torch-geometric) (1.11.1)
Requirement already satisfied: async-timeout<5.0,>=4.0 in
/usr/local/lib/python3.10/dist-packages (from aiohttp->torch-geometric) (4.0.3)
Requirement already satisfied: MarkupSafe>=2.0 in
/usr/local/lib/python3.10/dist-packages (from jinja2->torch-geometric) (2.1.5)
Requirement already satisfied: charset-normalizer<4,>=2 in
/usr/local/lib/python3.10/dist-packages (from requests->torch-geometric) (3.3.2)
Requirement already satisfied: idna<4,>=2.5 in /usr/local/lib/python3.10/dist-
packages (from requests->torch-geometric) (3.8)
Requirement already satisfied: urllib3<3,>=1.21.1 in
/usr/local/lib/python3.10/dist-packages (from requests->torch-geometric) (2.0.7)
Requirement already satisfied: certifi>=2017.4.17 in
/usr/local/lib/python3.10/dist-packages (from requests->torch-geometric)
(2024.8.30)
```

```
Requirement already satisfied: typing-extensions>=4.1.0 in
/usr/local/lib/python3.10/dist-packages (from
multidict<7.0,>=4.5->aiohttp->torch-geometric) (4.12.2)
Downloading rdkit-2024.3.5-cp310-cp310-manylinux_2_28_x86_64.whl (33.1 MB)
                          33.1/33.1 MB
43.6 MB/s eta 0:00:00
Downloading scikit_optimize-0.10.2-py2.py3-none-any.whl (107 kB)
                          107.8/107.8 kB
5.7 MB/s eta 0:00:00
Downloading pyaml-24.7.0-py3-none-any.whl (24 kB)
Building wheels for collected packages: torch-scatter, torch-sparse
  Building wheel for torch-scatter (setup.py) … canceled
ERROR: Operation cancelled by user
```

[ ]:
```
#2.1 GCN
import torch
import torch.nn.functional as F
from torch_geometric.data import Data
from torch_geometric.loader import DataLoader  # Corrected loader import
from torch_geometric.nn import GCNConv, global_mean_pool
from rdkit import Chem
from rdkit.Chem import rdmolops
from sklearn.model_selection import train_test_split
from sklearn.metrics import f1_score, accuracy_score
import numpy as np

# Function to convert SMILES to a PyTorch Geometric graph with labels
def smiles_to_graph(smiles, labels):
    mol = Chem.MolFromSmiles(smiles)
    adj = rdmolops.GetAdjacencyMatrix(mol)
    atoms = [atom.GetAtomicNum() for atom in mol.GetAtoms()]
    edge_index = torch.tensor(np.array(adj.nonzero()), dtype=torch.long)  #␣
 ↪Converted to NumPy array first
    x = torch.tensor(atoms, dtype=torch.float).view(-1, 1)

    # Add labels to the graph as a 2D tensor (batch_size, num_classes)
    data = Data(x=x, edge_index=edge_index)
    data.y = torch.tensor(labels, dtype=torch.float).view(1, -1)  # Convert␣
 ↪labels into a 2D tensor
    return data

# Convert SMILES to graphs with correct labels
graphs = [smiles_to_graph(smile, label) for smile, label in zip(smiles_list, y)]

# Prepare train/test split
```

```
train_graphs, test_graphs = train_test_split(graphs, test_size=0.2,␣
↪random_state=42)

# Create PyTorch Geometric DataLoaders
train_loader = DataLoader(train_graphs, batch_size=32, shuffle=True)
test_loader = DataLoader(test_graphs, batch_size=32, shuffle=False)

# Define the GCN Model
class GCN(torch.nn.Module):
    def __init__(self, num_node_features, num_classes):
        super(GCN, self).__init__()
        self.conv1 = GCNConv(num_node_features, 128)
        self.conv2 = GCNConv(128, 64)
        self.fc = torch.nn.Linear(64, num_classes)

    def forward(self, data):
        x, edge_index = data.x, data.edge_index
        x = self.conv1(x, edge_index)
        x = F.relu(x)
        x = self.conv2(x, edge_index)
        x = F.relu(x)
        x = global_mean_pool(x, data.batch)  # Global mean pooling
        x = self.fc(x)
        return torch.sigmoid(x)

# Train the GCN Model
def train_gnn_model(model, train_loader, optimizer, criterion, num_epochs=100):
    model.train()
    for epoch in range(num_epochs):
        total_loss = 0
        for data in train_loader:
            optimizer.zero_grad()
            output = model(data)
            loss = criterion(output, data.y.view(output.size()))  # Ensure both␣
↪target and output have the same shape
            loss.backward()
            optimizer.step()
            total_loss += loss.item()

# Initialize and Train the Model
gcn_model = GCN(num_node_features=1, num_classes=y.shape[1])  # num_classes set␣
↪to the number of reactions (24)
optimizer = torch.optim.Adam(gcn_model.parameters(), lr=0.001)
criterion = torch.nn.BCELoss()
train_gnn_model(gcn_model, train_loader, optimizer, criterion)

# Evaluate the GCN Model
```

```
def evaluate_gnn_model(model, test_loader):
    model.eval()
    total_acc = 0
    total_f1 = 0
    total_normal_acc = 0  # To track normal accuracy
    num_batches = len(test_loader)

    for data in test_loader:
        with torch.no_grad():
            output = model(data)
            preds = (output > 0.5).float()

            # Binary Accuracy (Micro-average of true/false classifications)
            total_acc += (preds == data.y).sum().item() / data.y.numel()

            # F1 Score (Micro-average)
            total_f1 += f1_score(data.y.cpu(), preds.cpu(), average='micro')

            # Normal Accuracy (Per-sample average accuracy)
            total_normal_acc += accuracy_score(data.y.cpu(), preds.cpu())

    print(f'Binary Accuracy: {total_acc / num_batches:.4f}')
    print(f'F1 Score: {total_f1 / num_batches:.4f}')
    print(f'Normal Accuracy: {total_normal_acc / num_batches:.4f}')  # Print␣
↪normal accuracy

evaluate_gnn_model(gcn_model, test_loader)
```

```
Binary Accuracy: 0.9101
F1 Score: 0.9529
Normal Accuracy: 0.2872
```

[ ]:
```
# 2.1.1 bayesian optimization with skopts for GCN

import torch
import torch.nn.functional as F
from torch_geometric.nn import GCNConv, global_mean_pool
from torch_geometric.data import DataLoader
from sklearn.model_selection import train_test_split
from sklearn.metrics import f1_score, accuracy_score
from skopt import gp_minimize
from skopt.space import Integer, Real
from skopt.utils import use_named_args

# GCN Model Definition
class GCN(torch.nn.Module):
```

```python
    def __init__(self, hidden_dim1, hidden_dim2, num_node_features,␣
 ↪num_classes):
        super(GCN, self).__init__()
        self.conv1 = GCNConv(num_node_features, hidden_dim1)
        self.conv2 = GCNConv(hidden_dim1, hidden_dim2)
        self.fc = torch.nn.Linear(hidden_dim2, num_classes)

    def forward(self, data):
        x, edge_index = data.x, data.edge_index
        x = self.conv1(x, edge_index)
        x = F.relu(x)
        x = self.conv2(x, edge_index)
        x = F.relu(x)
        x = global_mean_pool(x, data.batch)
        x = self.fc(x)
        return torch.sigmoid(x)

# Function to train the GCN model
def train_gnn_model(model, train_loader, optimizer, criterion, num_epochs=50):
    model.train()
    for epoch in range(num_epochs):
        total_loss = 0
        for data in train_loader:
            optimizer.zero_grad()
            output = model(data)
            loss = criterion(output, data.y)
            loss.backward()
            optimizer.step()
            total_loss += loss.item()
        print(f'Epoch {epoch + 1}/{num_epochs}, Loss: {total_loss /␣
 ↪len(train_loader)}')

# Function to evaluate the GCN model
def evaluate_gnn_model(model, test_loader):
    model.eval()
    total_acc, total_f1, total_normal_acc = 0, 0, 0
    num_batches = len(test_loader)

    for data in test_loader:
        with torch.no_grad():
            output = model(data)
            preds = (output > 0.5).float()
            total_acc += (preds == data.y).sum().item() / data.y.numel()
            total_f1 += f1_score(data.y.cpu(), preds.cpu(), average='micro')
            total_normal_acc += accuracy_score(data.y.cpu(), preds.cpu())
```

```python
    return total_acc / num_batches, total_f1 / num_batches, total_normal_acc /␣
    ↪num_batches

# Define the search space for hyperparameters
space = [
    Integer(64, 256, name='hidden_dim1'),  # First GCN hidden layer
    Integer(64, 256, name='hidden_dim2'),  # Second GCN hidden layer
    Real(0.0001, 0.01, name='learning_rate', prior='log-uniform'),
    Real(0.2, 0.6, name='dropout_rate'),
    Integer(32, 128, name='batch_size'),
    Integer(30, 100, name='epochs')
]

# Objective function to minimize (we want to maximize accuracy, so return␣
↪negative accuracy)
@use_named_args(space)
def objective(hidden_dim1, hidden_dim2, learning_rate, dropout_rate,␣
↪batch_size, epochs):
    model = GCN(hidden_dim1, hidden_dim2, num_node_features=1,␣
↪num_classes=y_train.shape[1])
    optimizer = torch.optim.Adam(model.parameters(), lr=learning_rate)
    criterion = torch.nn.BCELoss()

    # Train the model with current hyperparameters
    train_gnn_model(model, train_loader, optimizer, criterion,␣
↪num_epochs=epochs)

    # Evaluate the model
    acc, f1, normal_acc = evaluate_gnn_model(model, test_loader)

    # We want to maximize accuracy, so we return the negative of accuracy
    return -acc

# Run the Bayesian optimization
res = gp_minimize(objective, space, n_calls=20, random_state=42)

# Print the best hyperparameters found
print("Best hyperparameters: ", res.x)

# Train the final model with the best parameters found
best_hidden_dim1 = res.x[0]
best_hidden_dim2 = res.x[1]
best_learning_rate = res.x[2]
best_dropout_rate = res.x[3]
best_batch_size = res.x[4]
best_epochs = res.x[5]
```

```python
# Build and train the final model
best_model = GCN(best_hidden_dim1, best_hidden_dim2, num_node_features=1,␣
↪num_classes=y_train.shape[1])
optimizer = torch.optim.Adam(best_model.parameters(), lr=best_learning_rate)
criterion = torch.nn.BCELoss()

# Train the best model
train_gnn_model(best_model, train_loader, optimizer, criterion,␣
↪num_epochs=best_epochs)

# Evaluate the final model
final_acc, final_f1, final_normal_acc = evaluate_gnn_model(best_model,␣
↪test_loader)

# Print the final results
print(f'Best Binary Accuracy: {final_acc:.4f}')
print(f'Best Normal Accuracy: {final_normal_acc:.4f}')
print(f'Best F1 Score: {final_f1:.4f}')
```

```
Epoch 1/37, Loss: 0.3430831204442417
Epoch 2/37, Loss: 0.2868456257616772
Epoch 3/37, Loss: 0.27612609503900304
Epoch 4/37, Loss: 0.2812717219485956
Epoch 5/37, Loss: 0.2717820778489113
Epoch 6/37, Loss: 0.26814550205188636
Epoch 7/37, Loss: 0.27417977285735745
Epoch 8/37, Loss: 0.2682061309323591
Epoch 9/37, Loss: 0.2648399765877163
Epoch 10/37, Loss: 0.26334266408401374
Epoch 11/37, Loss: 0.2602413677993943
Epoch 12/37, Loss: 0.26033105438246446
Epoch 13/37, Loss: 0.2624366822488168
Epoch 14/37, Loss: 0.2602762058377266
Epoch 15/37, Loss: 0.25850275947767143
Epoch 16/37, Loss: 0.25924517564913807
Epoch 17/37, Loss: 0.26168611251256046
Epoch 18/37, Loss: 0.259558292434496
Epoch 19/37, Loss: 0.2576629440574085
Epoch 20/37, Loss: 0.2584920652648982
Epoch 21/37, Loss: 0.2581867575645447
Epoch 22/37, Loss: 0.25775720003773184
Epoch 23/37, Loss: 0.2621363804620855
Epoch 24/37, Loss: 0.2593674247755724
Epoch 25/37, Loss: 0.25843332488747206
Epoch 26/37, Loss: 0.26215114313013416
Epoch 27/37, Loss: 0.259267220602316
Epoch 28/37, Loss: 0.2618049711865537
```

```
Epoch 29/37, Loss: 0.25892717172117796
Epoch 30/37, Loss: 0.26012926504892464
Epoch 31/37, Loss: 0.25861654167666154
Epoch 32/37, Loss: 0.2609308383920613
Epoch 33/37, Loss: 0.2582799998276374
Epoch 34/37, Loss: 0.2605587791870622
Epoch 35/37, Loss: 0.25798699478892717
Epoch 36/37, Loss: 0.25750523586483565
Epoch 37/37, Loss: 0.25758246332407
Epoch 1/81, Loss: 0.6237863915808061
Epoch 2/81, Loss: 0.4807470844072454
Epoch 3/81, Loss: 0.3673459387877408
Epoch 4/81, Loss: 0.310341918731437
Epoch 5/81, Loss: 0.28195609327624827
Epoch 6/81, Loss: 0.27850084927152186
Epoch 7/81, Loss: 0.2796533252386486
Epoch 8/81, Loss: 0.2815023599302067
Epoch 9/81, Loss: 0.280095513252651
Epoch 10/81, Loss: 0.27880709092406664
Epoch 11/81, Loss: 0.27814475315458637
Epoch 12/81, Loss: 0.2823852024534169
Epoch 13/81, Loss: 0.2834721334716853
Epoch 14/81, Loss: 0.2814818484818234
Epoch 15/81, Loss: 0.2827170513132039
Epoch 16/81, Loss: 0.2788054452222936
Epoch 17/81, Loss: 0.28315943873980465
Epoch 18/81, Loss: 0.2807959340074483
Epoch 19/81, Loss: 0.28047311831923094
Epoch 20/81, Loss: 0.28094656546326247
Epoch 21/81, Loss: 0.2838657813913682
Epoch 22/81, Loss: 0.2741783747778219
Epoch 23/81, Loss: 0.2788602516931646
Epoch 24/81, Loss: 0.2785802106646931
Epoch 25/81, Loss: 0.27846774590366025
Epoch 26/81, Loss: 0.2764144925510182
Epoch 27/81, Loss: 0.2769543331335573
Epoch 28/81, Loss: 0.27675584133933573
Epoch 29/81, Loss: 0.27837738876833634
Epoch 30/81, Loss: 0.27661100862657323
Epoch 31/81, Loss: 0.2791844483684091
Epoch 32/81, Loss: 0.2800212555071887
Epoch 33/81, Loss: 0.2790853893932174
Epoch 34/81, Loss: 0.2784337169107269
Epoch 35/81, Loss: 0.27704163684564476
Epoch 36/81, Loss: 0.2784632115679629
Epoch 37/81, Loss: 0.27527349354589686
Epoch 38/81, Loss: 0.27773767885039835
Epoch 39/81, Loss: 0.2755471375935218
```

```
Epoch 40/81, Loss: 0.2769031546571675
Epoch 41/81, Loss: 0.275619791711078
Epoch 42/81, Loss: 0.2750450249980478
Epoch 43/81, Loss: 0.27363745692898245
Epoch 44/81, Loss: 0.2736103784512071
Epoch 45/81, Loss: 0.2744917965987149
Epoch 46/81, Loss: 0.27179035237606836
Epoch 47/81, Loss: 0.2748243208317196
Epoch 48/81, Loss: 0.27505860170897317
Epoch 49/81, Loss: 0.2724383942344609
Epoch 50/81, Loss: 0.26986180070568533
Epoch 51/81, Loss: 0.2708342973800266
Epoch 52/81, Loss: 0.2716854915899389
Epoch 53/81, Loss: 0.27010775620446487
Epoch 54/81, Loss: 0.27631722960401983
Epoch 55/81, Loss: 0.27136779050616655
Epoch 56/81, Loss: 0.2710251317304723
Epoch 57/81, Loss: 0.27138421114753275
Epoch 58/81, Loss: 0.27231793456217823
Epoch 59/81, Loss: 0.27222635728471417
Epoch 60/81, Loss: 0.2777118634651689
Epoch 61/81, Loss: 0.2679266425616601
Epoch 62/81, Loss: 0.26626042671063366
Epoch 63/81, Loss: 0.2668374294743818
Epoch 64/81, Loss: 0.266323467826142
Epoch 65/81, Loss: 0.26999419708462324
Epoch 66/81, Loss: 0.26831912512288375
Epoch 67/81, Loss: 0.2655480889713063
Epoch 68/81, Loss: 0.26808676824850197
Epoch 69/81, Loss: 0.2654309316593058
Epoch 70/81, Loss: 0.26699314397924084
Epoch 71/81, Loss: 0.26641863409210653
Epoch 72/81, Loss: 0.2656763822716825
Epoch 73/81, Loss: 0.2655571089947925
Epoch 74/81, Loss: 0.2656956367632922
Epoch 75/81, Loss: 0.2658946102156359
Epoch 76/81, Loss: 0.26589371001019196
Epoch 77/81, Loss: 0.26925941206076565
Epoch 78/81, Loss: 0.2674551185439615
Epoch 79/81, Loss: 0.26763997314607396
Epoch 80/81, Loss: 0.2663636965786709
Epoch 81/81, Loss: 0.26583825227092295
Epoch 1/30, Loss: 0.3516288518029101
Epoch 2/30, Loss: 0.2849210198311245
Epoch 3/30, Loss: 0.26959404962904315
Epoch 4/30, Loss: 0.2657143754117629
Epoch 5/30, Loss: 0.2706308535793248
Epoch 6/30, Loss: 0.2656286648091148
```

```
Epoch 7/30, Loss: 0.26349180980640297
Epoch 8/30, Loss: 0.2576467890073271
Epoch 9/30, Loss: 0.26037487212349386
Epoch 10/30, Loss: 0.264702472178375
Epoch 11/30, Loss: 0.2594585392405005
Epoch 12/30, Loss: 0.26073626341188655
Epoch 13/30, Loss: 0.25870079065070434
Epoch 14/30, Loss: 0.25830889274092284
Epoch 15/30, Loss: 0.2579215554630055
Epoch 16/30, Loss: 0.26146263410063353
Epoch 17/30, Loss: 0.2573276105172494
Epoch 18/30, Loss: 0.2568947826238239
Epoch 19/30, Loss: 0.2609951228779905
Epoch 20/30, Loss: 0.258510881487061
Epoch 21/30, Loss: 0.2614632634555592
Epoch 22/30, Loss: 0.2599470085080932
Epoch 23/30, Loss: 0.2585375677136814
Epoch 24/30, Loss: 0.261518661151914
Epoch 25/30, Loss: 0.2579406876774395
Epoch 26/30, Loss: 0.258163959664457
Epoch 27/30, Loss: 0.2572706950938
Epoch 28/30, Loss: 0.2578907306579983
Epoch 29/30, Loss: 0.25772295760757785
Epoch 30/30, Loss: 0.257291246424703
Epoch 1/46, Loss: 0.5120057323399712
Epoch 2/46, Loss: 0.29698360141585856
Epoch 3/46, Loss: 0.2853832341292325
Epoch 4/46, Loss: 0.287257581949234
Epoch 5/46, Loss: 0.27720146626234055
Epoch 6/46, Loss: 0.2759988180854741
Epoch 7/46, Loss: 0.27894147134879055
Epoch 8/46, Loss: 0.2826379252707257
Epoch 9/46, Loss: 0.280584099538186
Epoch 10/46, Loss: 0.2821182726937182
Epoch 11/46, Loss: 0.274013678817188
Epoch 12/46, Loss: 0.28015028553850513
Epoch 13/46, Loss: 0.27290454726008806
Epoch 14/46, Loss: 0.2783452973646276
Epoch 15/46, Loss: 0.27653418306042166
Epoch 16/46, Loss: 0.2802034335977891
Epoch 17/46, Loss: 0.2770811591078253
Epoch 18/46, Loss: 0.27392299999209013
Epoch 19/46, Loss: 0.2805766077602611
Epoch 20/46, Loss: 0.27639621715335283
Epoch 21/46, Loss: 0.2707962976659046
Epoch 22/46, Loss: 0.2805701840449782
Epoch 23/46, Loss: 0.2691033202059129
Epoch 24/46, Loss: 0.26591474519056435
```

```
Epoch 25/46, Loss: 0.27086875675355687
Epoch 26/46, Loss: 0.27058591781293645
Epoch 27/46, Loss: 0.2657439305501826
Epoch 28/46, Loss: 0.2721793208928669
Epoch 29/46, Loss: 0.26719782720593843
Epoch 30/46, Loss: 0.2688606206108542
Epoch 31/46, Loss: 0.2668287276345141
Epoch 32/46, Loss: 0.2653474308112088
Epoch 33/46, Loss: 0.2661015452707515
Epoch 34/46, Loss: 0.2650333371232538
Epoch 35/46, Loss: 0.265191129025291
Epoch 36/46, Loss: 0.2638290217694114
Epoch 37/46, Loss: 0.2655296347597066
Epoch 38/46, Loss: 0.2654041516430238
Epoch 39/46, Loss: 0.264371782541275
Epoch 40/46, Loss: 0.264512090560268
Epoch 41/46, Loss: 0.26426617625881643
Epoch 42/46, Loss: 0.26392680669532104
Epoch 43/46, Loss: 0.26391713715651455
Epoch 44/46, Loss: 0.2650991275030024
Epoch 45/46, Loss: 0.2677015022319906
Epoch 46/46, Loss: 0.26342156573253517
Epoch 1/90, Loss: 0.511732973596629
Epoch 2/90, Loss: 0.30273904914365096
Epoch 3/90, Loss: 0.28682210647008
Epoch 4/90, Loss: 0.28453006244757595
Epoch 5/90, Loss: 0.2821871121140087
Epoch 6/90, Loss: 0.28221885553177667
Epoch 7/90, Loss: 0.2789309669066878
Epoch 8/90, Loss: 0.27985598015434604
Epoch 9/90, Loss: 0.28346331373733635
Epoch 10/90, Loss: 0.279766560477369
Epoch 11/90, Loss: 0.2810178048470441
Epoch 12/90, Loss: 0.27754296669188666
Epoch 13/90, Loss: 0.2756958652068587
Epoch 14/90, Loss: 0.2745235308128245
Epoch 15/90, Loss: 0.27702487654545727
Epoch 16/90, Loss: 0.2739867313819773
Epoch 17/90, Loss: 0.2750195107039283
Epoch 18/90, Loss: 0.2756977839504971
Epoch 19/90, Loss: 0.2704415268757764
Epoch 20/90, Loss: 0.2725648349698852
Epoch 21/90, Loss: 0.2711848086294006
Epoch 22/90, Loss: 0.2737342215636197
Epoch 23/90, Loss: 0.26632187208708596
Epoch 24/90, Loss: 0.2696016820914605
Epoch 25/90, Loss: 0.2677253809045343
Epoch 26/90, Loss: 0.26954763952423544
```

```
Epoch 27/90, Loss: 0.26665082530063744
Epoch 28/90, Loss: 0.2694043223472202
Epoch 29/90, Loss: 0.2668997150133638
Epoch 30/90, Loss: 0.2662936957443462
Epoch 31/90, Loss: 0.2661149409763953
Epoch 32/90, Loss: 0.2667243761174819
Epoch 33/90, Loss: 0.2676033355733928
Epoch 34/90, Loss: 0.26647280419574065
Epoch 35/90, Loss: 0.27096553453627753
Epoch 36/90, Loss: 0.26495346383136864
Epoch 37/90, Loss: 0.2639555150971693
Epoch 38/90, Loss: 0.26422104677733255
Epoch 39/90, Loss: 0.26378194768639174
Epoch 40/90, Loss: 0.26173634827136993
Epoch 41/90, Loss: 0.26474560753387566
Epoch 42/90, Loss: 0.2622404558693661
Epoch 43/90, Loss: 0.2634820048423374
Epoch 44/90, Loss: 0.26304539117742987
Epoch 45/90, Loss: 0.26493169498794217
Epoch 46/90, Loss: 0.26066051730338263
Epoch 47/90, Loss: 0.26081953679814057
Epoch 48/90, Loss: 0.26512038269463706
Epoch 49/90, Loss: 0.26282357906593995
Epoch 50/90, Loss: 0.2628311672631432
Epoch 51/90, Loss: 0.25973376587909813
Epoch 52/90, Loss: 0.26678116093663606
Epoch 53/90, Loss: 0.2597174105398795
Epoch 54/90, Loss: 0.2594480111318476
Epoch 55/90, Loss: 0.26393967165666465
Epoch 56/90, Loss: 0.25980227878865075
Epoch 57/90, Loss: 0.2584671080112457
Epoch 58/90, Loss: 0.2588415496489581
Epoch 59/90, Loss: 0.2577693988295162
Epoch 60/90, Loss: 0.2580985939678024
Epoch 61/90, Loss: 0.25882507334737215
Epoch 62/90, Loss: 0.26016760179225135
Epoch 63/90, Loss: 0.2589297088630059
Epoch 64/90, Loss: 0.2577337938196519
Epoch 65/90, Loss: 0.25660283162313346
Epoch 66/90, Loss: 0.2596064854194136
Epoch 67/90, Loss: 0.2591237320619471
Epoch 68/90, Loss: 0.25665783356217775
Epoch 69/90, Loss: 0.25663530651260824
Epoch 70/90, Loss: 0.25903550316305723
Epoch 71/90, Loss: 0.26048608168083076
Epoch 72/90, Loss: 0.25565175010877494
Epoch 73/90, Loss: 0.2569577378385207
Epoch 74/90, Loss: 0.2598072084433892
```

```
Epoch 75/90, Loss: 0.25780224931590695
Epoch 76/90, Loss: 0.2572178016690647
Epoch 77/90, Loss: 0.2579485436572748
Epoch 78/90, Loss: 0.2583482800161137
Epoch 79/90, Loss: 0.25493691773975596
Epoch 80/90, Loss: 0.25634851043715196
Epoch 81/90, Loss: 0.26018815838238774
Epoch 82/90, Loss: 0.2586413836654495
Epoch 83/90, Loss: 0.2574258591322338
Epoch 84/90, Loss: 0.2556450682527879
Epoch 85/90, Loss: 0.2611992516938378
Epoch 86/90, Loss: 0.25866564319414254
Epoch 87/90, Loss: 0.2595418358550352
Epoch 88/90, Loss: 0.2574254712637733
Epoch 89/90, Loss: 0.25739025149275274
Epoch 90/90, Loss: 0.25880594946005764
Epoch 1/57, Loss: 0.6583513354553896
Epoch 2/57, Loss: 0.570824451306287
Epoch 3/57, Loss: 0.4833807208958794
Epoch 4/57, Loss: 0.4035383024636437
Epoch 5/57, Loss: 0.34576892852783203
Epoch 6/57, Loss: 0.3134484746876885
Epoch 7/57, Loss: 0.2930199122604202
Epoch 8/57, Loss: 0.2904348233166863
Epoch 9/57, Loss: 0.2852667153758161
Epoch 10/57, Loss: 0.2930032146327636
Epoch 11/57, Loss: 0.27690571415073734
Epoch 12/57, Loss: 0.2792159774724175
Epoch 13/57, Loss: 0.2802438455469468
Epoch 14/57, Loss: 0.2815482143093558
Epoch 15/57, Loss: 0.28053362360771966
Epoch 16/57, Loss: 0.27896059479783564
Epoch 17/57, Loss: 0.28164003570290175
Epoch 18/57, Loss: 0.2753646549056558
Epoch 19/57, Loss: 0.27599824702038483
Epoch 20/57, Loss: 0.27658013122923236
Epoch 21/57, Loss: 0.2742946932421011
Epoch 22/57, Loss: 0.2755507931989782
Epoch 23/57, Loss: 0.27814946674248753
Epoch 24/57, Loss: 0.2773505515035461
Epoch 25/57, Loss: 0.2749929248410113
Epoch 26/57, Loss: 0.2758807798518854
Epoch 27/57, Loss: 0.2743653782150325
Epoch 28/57, Loss: 0.27532414829029755
Epoch 29/57, Loss: 0.27504324387101564
Epoch 30/57, Loss: 0.2737976899918388
Epoch 31/57, Loss: 0.2753347022568478
Epoch 32/57, Loss: 0.2742757801623905
```

```
Epoch 33/57, Loss: 0.27452070704277826
Epoch 34/57, Loss: 0.2746366380768664
Epoch 35/57, Loss: 0.2749834845171255
Epoch 36/57, Loss: 0.2754245645859662
Epoch 37/57, Loss: 0.2750367421437712
Epoch 38/57, Loss: 0.27328088704277487
Epoch 39/57, Loss: 0.2750913137022187
Epoch 40/57, Loss: 0.2766783618751694
Epoch 41/57, Loss: 0.27539364379995007
Epoch 42/57, Loss: 0.2759014433797668
Epoch 43/57, Loss: 0.27417177619302974
Epoch 44/57, Loss: 0.27483922330772176
Epoch 45/57, Loss: 0.2747659854152623
Epoch 46/57, Loss: 0.2744820700848804
Epoch 47/57, Loss: 0.27768255068975334
Epoch 48/57, Loss: 0.2786157455514459
Epoch 49/57, Loss: 0.2737356538281721
Epoch 50/57, Loss: 0.27912824732415814
Epoch 51/57, Loss: 0.2766757340115659
Epoch 52/57, Loss: 0.27315424514167447
Epoch 53/57, Loss: 0.27758798266158385
Epoch 54/57, Loss: 0.2729850867215325
Epoch 55/57, Loss: 0.2788333783254904
Epoch 56/57, Loss: 0.27602753262309465
Epoch 57/57, Loss: 0.2742364704608917
Epoch 1/88, Loss: 0.5965042280800202
Epoch 2/88, Loss: 0.42869508617064533
Epoch 3/88, Loss: 0.3207391759051996
Epoch 4/88, Loss: 0.2891157620093402
Epoch 5/88, Loss: 0.2791697374161552
Epoch 6/88, Loss: 0.2825067104662166
Epoch 7/88, Loss: 0.2789502770585172
Epoch 8/88, Loss: 0.27499497813336987
Epoch 9/88, Loss: 0.27796407569857207
Epoch 10/88, Loss: 0.28268008766805425
Epoch 11/88, Loss: 0.2739901963402243
Epoch 12/88, Loss: 0.2842569412554012
Epoch 13/88, Loss: 0.2790319288478178
Epoch 14/88, Loss: 0.279829026583363
Epoch 15/88, Loss: 0.279609550448025
Epoch 16/88, Loss: 0.2897398870657472
Epoch 17/88, Loss: 0.2929975705988267
Epoch 18/88, Loss: 0.27409469949848514
Epoch 19/88, Loss: 0.2780158138450454
Epoch 20/88, Loss: 0.274844068376457
Epoch 21/88, Loss: 0.2789492826251423
Epoch 22/88, Loss: 0.2785259520306307
Epoch 23/88, Loss: 0.2739758815835504
```

```
Epoch 24/88, Loss: 0.27693573736092625
Epoch 25/88, Loss: 0.2780534055303125
Epoch 26/88, Loss: 0.2765185561250238
Epoch 27/88, Loss: 0.2750208044753355
Epoch 28/88, Loss: 0.2766653053900775
Epoch 29/88, Loss: 0.2746096051791135
Epoch 30/88, Loss: 0.2748774570577285
Epoch 31/88, Loss: 0.27571234107017517
Epoch 32/88, Loss: 0.2736345441902385
Epoch 33/88, Loss: 0.275751947918359
Epoch 34/88, Loss: 0.2779308478621876
Epoch 35/88, Loss: 0.270069428226527
Epoch 36/88, Loss: 0.2716299308573498
Epoch 37/88, Loss: 0.2754347819615813
Epoch 38/88, Loss: 0.2703072249012835
Epoch 39/88, Loss: 0.2754305855316274
Epoch 40/88, Loss: 0.2748006385038881
Epoch 41/88, Loss: 0.27403710125123754
Epoch 42/88, Loss: 0.2743656459976645
Epoch 43/88, Loss: 0.27379099805565443
Epoch 44/88, Loss: 0.2727320290663663
Epoch 45/88, Loss: 0.2742646371617037
Epoch 46/88, Loss: 0.2753380729871638
Epoch 47/88, Loss: 0.27235036124201384
Epoch 48/88, Loss: 0.2720194051370901
Epoch 49/88, Loss: 0.27059222582508535
Epoch 50/88, Loss: 0.269897258895285
Epoch 51/88, Loss: 0.2696406985906994
Epoch 52/88, Loss: 0.2703826458138578
Epoch 53/88, Loss: 0.2695537209510803
Epoch 54/88, Loss: 0.27303071802153306
Epoch 55/88, Loss: 0.26917030048720975
Epoch 56/88, Loss: 0.26770339643254
Epoch 57/88, Loss: 0.2675635989974527
Epoch 58/88, Loss: 0.2682224231607774
Epoch 59/88, Loss: 0.2658932791913257
Epoch 60/88, Loss: 0.26730068641550403
Epoch 61/88, Loss: 0.26761553743306327
Epoch 62/88, Loss: 0.26614779628374996
Epoch 63/88, Loss: 0.26562131678356843
Epoch 64/88, Loss: 0.2665162774569848
Epoch 65/88, Loss: 0.26610825649079156
Epoch 66/88, Loss: 0.26563675526310415
Epoch 67/88, Loss: 0.2677425952518688
Epoch 68/88, Loss: 0.26623162276604595
Epoch 69/88, Loss: 0.2668135126723963
Epoch 70/88, Loss: 0.2647487700862043
Epoch 71/88, Loss: 0.2658349691944964
```

```
Epoch 72/88, Loss: 0.2656729940105887
Epoch 73/88, Loss: 0.26848623156547546
Epoch 74/88, Loss: 0.26456059121033726
Epoch 75/88, Loss: 0.2645766805200016
Epoch 76/88, Loss: 0.2650959180558429
Epoch 77/88, Loss: 0.26822930004666834
Epoch 78/88, Loss: 0.26465413324973164
Epoch 79/88, Loss: 0.26518116759903293
Epoch 80/88, Loss: 0.26392439007759094
Epoch 81/88, Loss: 0.265213883098434
Epoch 82/88, Loss: 0.2648977716179455
Epoch 83/88, Loss: 0.26309295787530784
Epoch 84/88, Loss: 0.26589715699939165
Epoch 85/88, Loss: 0.26535776301341896
Epoch 86/88, Loss: 0.26612817145445766
Epoch 87/88, Loss: 0.26408882526790395
Epoch 88/88, Loss: 0.2643824964761734
Epoch 1/45, Loss: 0.6352252592058742
Epoch 2/45, Loss: 0.4989793177913217
Epoch 3/45, Loss: 0.3735007591107312
Epoch 4/45, Loss: 0.306058856056017
Epoch 5/45, Loss: 0.28275220534380746
Epoch 6/45, Loss: 0.2856014839866582
Epoch 7/45, Loss: 0.27970889166874047
Epoch 8/45, Loss: 0.27675535091582465
Epoch 9/45, Loss: 0.27567440665820064
Epoch 10/45, Loss: 0.2799903754802311
Epoch 11/45, Loss: 0.2817036820685162
Epoch 12/45, Loss: 0.28149406918708014
Epoch 13/45, Loss: 0.2798803952686927
Epoch 14/45, Loss: 0.27798764889731126
Epoch 15/45, Loss: 0.27893271165735584
Epoch 16/45, Loss: 0.27696455620667515
Epoch 17/45, Loss: 0.27835816320251017
Epoch 18/45, Loss: 0.27766453211798386
Epoch 19/45, Loss: 0.2774870860226014
Epoch 20/45, Loss: 0.2786518471205936
Epoch 21/45, Loss: 0.27935001780005064
Epoch 22/45, Loss: 0.2767748674925636
Epoch 23/45, Loss: 0.2776255397235646
Epoch 24/45, Loss: 0.2770647410960758
Epoch 25/45, Loss: 0.27674972265958786
Epoch 26/45, Loss: 0.27448640325490165
Epoch 27/45, Loss: 0.28160906889859366
Epoch 28/45, Loss: 0.2745995872160968
Epoch 29/45, Loss: 0.2768266582313706
Epoch 30/45, Loss: 0.2711732812664088
Epoch 31/45, Loss: 0.2693389316692072
```

```
Epoch 32/45, Loss: 0.27213611585252423
Epoch 33/45, Loss: 0.2718777406741591
Epoch 34/45, Loss: 0.2739655542023042
Epoch 35/45, Loss: 0.2742205310393782
Epoch 36/45, Loss: 0.27477849669316234
Epoch 37/45, Loss: 0.2699806024046505
Epoch 38/45, Loss: 0.2737137736643062
Epoch 39/45, Loss: 0.2741282481481047
Epoch 40/45, Loss: 0.2706730229889645
Epoch 41/45, Loss: 0.27089449631817203
Epoch 42/45, Loss: 0.2733282928957659
Epoch 43/45, Loss: 0.2791173804332228
Epoch 44/45, Loss: 0.27271513816188364
Epoch 45/45, Loss: 0.279780371224179
Epoch 1/95, Loss: 0.35218094727572274
Epoch 2/95, Loss: 0.2798655765021549
Epoch 3/95, Loss: 0.2765262486303554
Epoch 4/95, Loss: 0.2789907126742251
Epoch 5/95, Loss: 0.2680308924001806
Epoch 6/95, Loss: 0.2672775221221587
Epoch 7/95, Loss: 0.2688947512822993
Epoch 8/95, Loss: 0.2666267226724064
Epoch 9/95, Loss: 0.26317232496598186
Epoch 10/95, Loss: 0.26145943429540186
Epoch 11/95, Loss: 0.26072006059043545
Epoch 12/95, Loss: 0.2578773402115878
Epoch 13/95, Loss: 0.2589091496432529
Epoch 14/95, Loss: 0.2594597865553463
Epoch 15/95, Loss: 0.257465144728913
Epoch 16/95, Loss: 0.2597524593858158
Epoch 17/95, Loss: 0.25963076642330957
Epoch 18/95, Loss: 0.2597721433814834
Epoch 19/95, Loss: 0.2583018160041641
Epoch 20/95, Loss: 0.2574457364047275
Epoch 21/95, Loss: 0.25790347947793846
Epoch 22/95, Loss: 0.2578558251261711
Epoch 23/95, Loss: 0.25901462400660796
Epoch 24/95, Loss: 0.26000643302412596
Epoch 25/95, Loss: 0.25899910707684126
Epoch 26/95, Loss: 0.26151904639075785
Epoch 27/95, Loss: 0.2593955660567564
Epoch 28/95, Loss: 0.2583315394380513
Epoch 29/95, Loss: 0.2654862732571714
Epoch 30/95, Loss: 0.26143376704524546
Epoch 31/95, Loss: 0.25974753586684957
Epoch 32/95, Loss: 0.2572426173616858
Epoch 33/95, Loss: 0.2578214709373081
Epoch 34/95, Loss: 0.25996240140760646
```

```
Epoch 35/95, Loss: 0.26124878403018503
Epoch 36/95, Loss: 0.2593494763269144
Epoch 37/95, Loss: 0.25769951588967266
Epoch 38/95, Loss: 0.26172975538408055
Epoch 39/95, Loss: 0.2593921260798679
Epoch 40/95, Loss: 0.25753535242641673
Epoch 41/95, Loss: 0.26018083446166096
Epoch 42/95, Loss: 0.2566336529219852
Epoch 43/95, Loss: 0.25863224136478763
Epoch 44/95, Loss: 0.2582619326079593
Epoch 45/95, Loss: 0.25783073682995405
Epoch 46/95, Loss: 0.2583292056532467
Epoch 47/95, Loss: 0.25790567696094513
Epoch 48/95, Loss: 0.2573974294697537
Epoch 49/95, Loss: 0.2588649563053075
Epoch 50/95, Loss: 0.25804835952380123
Epoch 51/95, Loss: 0.2595046605257427
Epoch 52/95, Loss: 0.2567184650722672
Epoch 53/95, Loss: 0.25663729888551373
Epoch 54/95, Loss: 0.25557492366608453
Epoch 55/95, Loss: 0.2578890656723696
Epoch 56/95, Loss: 0.26473874817876253
Epoch 57/95, Loss: 0.258534798727316
Epoch 58/95, Loss: 0.2567431571729043
Epoch 59/95, Loss: 0.25715966347385855
Epoch 60/95, Loss: 0.2656802307156956
Epoch 61/95, Loss: 0.2603461041170008
Epoch 62/95, Loss: 0.25888583721483455
Epoch 63/95, Loss: 0.257665972061017
Epoch 64/95, Loss: 0.25642872338785844
Epoch 65/95, Loss: 0.2571756782777169
Epoch 66/95, Loss: 0.2579093457144849
Epoch 67/95, Loss: 0.2580590633785023
Epoch 68/95, Loss: 0.2580855462480994
Epoch 69/95, Loss: 0.2572474431465654
Epoch 70/95, Loss: 0.25606439963859673
Epoch 71/95, Loss: 0.2583914504331701
Epoch 72/95, Loss: 0.2565670723424238
Epoch 73/95, Loss: 0.2559618866618942
Epoch 74/95, Loss: 0.2592875317615621
Epoch 75/95, Loss: 0.25697242863038006
Epoch 76/95, Loss: 0.25816788699697046
Epoch 77/95, Loss: 0.25707703697330814
Epoch 78/95, Loss: 0.2572665920152384
Epoch 79/95, Loss: 0.2561369384912884
Epoch 80/95, Loss: 0.2568994076812969
Epoch 81/95, Loss: 0.2586880720713559
Epoch 82/95, Loss: 0.25572220279889946
```

```
Epoch 83/95, Loss: 0.2562391657163115
Epoch 84/95, Loss: 0.2558722320724936
Epoch 85/95, Loss: 0.2562992349267006
Epoch 86/95, Loss: 0.25947083576637153
Epoch 87/95, Loss: 0.2575455369318233
Epoch 88/95, Loss: 0.2564496752970359
Epoch 89/95, Loss: 0.256699639646446
Epoch 90/95, Loss: 0.25997770709149975
Epoch 91/95, Loss: 0.2571143148576512
Epoch 92/95, Loss: 0.2566731791285908
Epoch 93/95, Loss: 0.25688019614009294
Epoch 94/95, Loss: 0.2565729854738011
Epoch 95/95, Loss: 0.2577045218032949
Epoch 1/89, Loss: 0.39115744434735356
Epoch 2/89, Loss: 0.2869081834659857
Epoch 3/89, Loss: 0.28324610170196085
Epoch 4/89, Loss: 0.2780856424394776
Epoch 5/89, Loss: 0.27642927231157527
Epoch 6/89, Loss: 0.27333439579781366
Epoch 7/89, Loss: 0.28144274389042573
Epoch 8/89, Loss: 0.2737090841812246
Epoch 9/89, Loss: 0.27317507302059846
Epoch 10/89, Loss: 0.2720404096386012
Epoch 11/89, Loss: 0.2718973453430569
Epoch 12/89, Loss: 0.2666752509334508
Epoch 13/89, Loss: 0.2662394620916423
Epoch 14/89, Loss: 0.2655533946612302
Epoch 15/89, Loss: 0.265841601526036
Epoch 16/89, Loss: 0.26898466795682907
Epoch 17/89, Loss: 0.2640474276507602
Epoch 18/89, Loss: 0.26745642020421867
Epoch 19/89, Loss: 0.2630841061472893
Epoch 20/89, Loss: 0.262542939361404
Epoch 21/89, Loss: 0.2650875300168991
Epoch 22/89, Loss: 0.26257770683835535
Epoch 23/89, Loss: 0.2618868526290445
Epoch 24/89, Loss: 0.2598448772640789
Epoch 25/89, Loss: 0.2612939113203217
Epoch 26/89, Loss: 0.26082006885724907
Epoch 27/89, Loss: 0.26054593745399923
Epoch 28/89, Loss: 0.2586032169706681
Epoch 29/89, Loss: 0.25747643323505626
Epoch 30/89, Loss: 0.2573033961303094
Epoch 31/89, Loss: 0.25811003586825204
Epoch 32/89, Loss: 0.25885045791373534
Epoch 33/89, Loss: 0.2584933572832276
Epoch 34/89, Loss: 0.2563488763921401
Epoch 35/89, Loss: 0.2568472970057936
```

```
Epoch 36/89, Loss: 0.2561609153361881
Epoch 37/89, Loss: 0.2570442070855814
Epoch 38/89, Loss: 0.2602296432151514
Epoch 39/89, Loss: 0.2574002690175
Epoch 40/89, Loss: 0.2564324425423847
Epoch 41/89, Loss: 0.2572936924064861
Epoch 42/89, Loss: 0.25668830293066364
Epoch 43/89, Loss: 0.2569433003664017
Epoch 44/89, Loss: 0.25937474299879637
Epoch 45/89, Loss: 0.25746380537748337
Epoch 46/89, Loss: 0.2574062772533473
Epoch 47/89, Loss: 0.2556089246097733
Epoch 48/89, Loss: 0.25773169932996526
Epoch 49/89, Loss: 0.25752898787750916
Epoch 50/89, Loss: 0.257564815966522
Epoch 51/89, Loss: 0.25977295067380457
Epoch 52/89, Loss: 0.25965587926261563
Epoch 53/89, Loss: 0.257280166973086
Epoch 54/89, Loss: 0.2563311646089834
Epoch 55/89, Loss: 0.2564105575575548
Epoch 56/89, Loss: 0.26083067979882746
Epoch 57/89, Loss: 0.26193587262840834
Epoch 58/89, Loss: 0.25859753393075047
Epoch 59/89, Loss: 0.2569532547803486
Epoch 60/89, Loss: 0.257123424726374
Epoch 61/89, Loss: 0.25716350008459654
Epoch 62/89, Loss: 0.2568598705179551
Epoch 63/89, Loss: 0.2557475251310012
Epoch 64/89, Loss: 0.25751657564850416
Epoch 65/89, Loss: 0.2584050731623874
Epoch 66/89, Loss: 0.2573191825957859
Epoch 67/89, Loss: 0.2583378939067616
Epoch 68/89, Loss: 0.2588359814356355
Epoch 69/89, Loss: 0.2569851231049089
Epoch 70/89, Loss: 0.2574391807703411
Epoch 71/89, Loss: 0.2574505065293873
Epoch 72/89, Loss: 0.2555677623433225
Epoch 73/89, Loss: 0.25646210549508824
Epoch 74/89, Loss: 0.2563001456506112
Epoch 75/89, Loss: 0.2581165218177964
Epoch 76/89, Loss: 0.2575482449987355
Epoch 77/89, Loss: 0.25772086005000505
Epoch 78/89, Loss: 0.2571152168161729
Epoch 79/89, Loss: 0.256893552839756
Epoch 80/89, Loss: 0.25819653404109616
Epoch 81/89, Loss: 0.2580673361525816
Epoch 82/89, Loss: 0.2567667023223989
Epoch 83/89, Loss: 0.25707515986526713
```

```
Epoch 84/89, Loss: 0.26635681443354664
Epoch 85/89, Loss: 0.26058564247453914
Epoch 86/89, Loss: 0.25764649273718104
Epoch 87/89, Loss: 0.25757158153197346
Epoch 88/89, Loss: 0.25851184436503577
Epoch 89/89, Loss: 0.255995235460646
Epoch 1/95, Loss: 0.3326559610226575
Epoch 2/95, Loss: 0.28643341318649407
Epoch 3/95, Loss: 0.27176039315321865
Epoch 4/95, Loss: 0.27273129627985115
Epoch 5/95, Loss: 0.2659054630819489
Epoch 6/95, Loss: 0.2616981377496439
Epoch 7/95, Loss: 0.26239389794714313
Epoch 8/95, Loss: 0.26040613037698407
Epoch 9/95, Loss: 0.2576545639073147
Epoch 10/95, Loss: 0.26428209858782153
Epoch 11/95, Loss: 0.2599891047267353
Epoch 12/95, Loss: 0.26007107601446267
Epoch 13/95, Loss: 0.259887654115172
Epoch 14/95, Loss: 0.2590394659953959
Epoch 15/95, Loss: 0.2576100366080509
Epoch 16/95, Loss: 0.2592792125309215
Epoch 17/95, Loss: 0.2609236775075688
Epoch 18/95, Loss: 0.25912628410493627
Epoch 19/95, Loss: 0.25796451989342184
Epoch 20/95, Loss: 0.25712526414324255
Epoch 21/95, Loss: 0.2613903025493902
Epoch 22/95, Loss: 0.26080455718671575
Epoch 23/95, Loss: 0.2578355021336499
Epoch 24/95, Loss: 0.25977713001124997
Epoch 25/95, Loss: 0.257201731643256
Epoch 26/95, Loss: 0.259227681247627
Epoch 27/95, Loss: 0.25914330473717523
Epoch 28/95, Loss: 0.2602281969259767
Epoch 29/95, Loss: 0.2589555167976548
Epoch 30/95, Loss: 0.25732023181284175
Epoch 31/95, Loss: 0.2611958555438939
Epoch 32/95, Loss: 0.2577645384213504
Epoch 33/95, Loss: 0.2576202307553852
Epoch 34/95, Loss: 0.2581391040893162
Epoch 35/95, Loss: 0.25740908524569345
Epoch 36/95, Loss: 0.25737108114887686
Epoch 37/95, Loss: 0.2582761807476773
Epoch 38/95, Loss: 0.2562750056385994
Epoch 39/95, Loss: 0.2562665869207943
Epoch 40/95, Loss: 0.2584764172925669
Epoch 41/95, Loss: 0.25610607687164755
Epoch 42/95, Loss: 0.2579655095058329
```

```
Epoch 43/95, Loss: 0.257156079306322
Epoch 44/95, Loss: 0.25760218413437114
Epoch 45/95, Loss: 0.256487213075161
Epoch 46/95, Loss: 0.2563491555697778
Epoch 47/95, Loss: 0.25809448913616295
Epoch 48/95, Loss: 0.2585620428709423
Epoch 49/95, Loss: 0.2573504263863844
Epoch 50/95, Loss: 0.2616653074236477
Epoch 51/95, Loss: 0.2578679472208023
Epoch 52/95, Loss: 0.2561298810383853
Epoch 53/95, Loss: 0.2564544090453316
Epoch 54/95, Loss: 0.2580668014638564
Epoch 55/95, Loss: 0.2575097811572692
Epoch 56/95, Loss: 0.25776463922332316
Epoch 57/95, Loss: 0.2575048913850504
Epoch 58/95, Loss: 0.25814209264867444
Epoch 59/95, Loss: 0.25587205939433155
Epoch 60/95, Loss: 0.2582551972830997
Epoch 61/95, Loss: 0.25880370464395075
Epoch 62/95, Loss: 0.25614353970569725
Epoch 63/95, Loss: 0.2562740957912277
Epoch 64/95, Loss: 0.2583339034634478
Epoch 65/95, Loss: 0.2591768468127531
Epoch 66/95, Loss: 0.25708428025245667
Epoch 67/95, Loss: 0.2623140671673943
Epoch 68/95, Loss: 0.2558992794331382
Epoch 69/95, Loss: 0.2580154099885155
Epoch 70/95, Loss: 0.2556557891999974
Epoch 71/95, Loss: 0.25870997414869423
Epoch 72/95, Loss: 0.25780263806090636
Epoch 73/95, Loss: 0.25803887581124024
Epoch 74/95, Loss: 0.2561763963278602
Epoch 75/95, Loss: 0.2579983849735821
Epoch 76/95, Loss: 0.25639700451317954
Epoch 77/95, Loss: 0.25982000371989084
Epoch 78/95, Loss: 0.2583468354800168
Epoch 79/95, Loss: 0.2563903024967979
Epoch 80/95, Loss: 0.261795770157786
Epoch 81/95, Loss: 0.25622982224997354
Epoch 82/95, Loss: 0.2567785794244093
Epoch 83/95, Loss: 0.2586748117909712
Epoch 84/95, Loss: 0.2577577207018347
Epoch 85/95, Loss: 0.25632838510415135
Epoch 86/95, Loss: 0.2579904877964188
Epoch 87/95, Loss: 0.25662792167242837
Epoch 88/95, Loss: 0.2593959548017558
Epoch 89/95, Loss: 0.25612223498961506
Epoch 90/95, Loss: 0.2581501476028386
```

```
Epoch 91/95, Loss: 0.25583283296402765
Epoch 92/95, Loss: 0.2573179243242039
Epoch 93/95, Loss: 0.2591007667429307
Epoch 94/95, Loss: 0.25729316604488034
Epoch 95/95, Loss: 0.2584751533234821
Epoch 1/54, Loss: 0.380375912084299
Epoch 2/54, Loss: 0.28204120082013745
Epoch 3/54, Loss: 0.28386113442042293
Epoch 4/54, Loss: 0.28236815421020284
Epoch 5/54, Loss: 0.2814457271905506
Epoch 6/54, Loss: 0.2844947738682522
Epoch 7/54, Loss: 0.2691158604095964
Epoch 8/54, Loss: 0.2680790161385256
Epoch 9/54, Loss: 0.27212535064010057
Epoch 10/54, Loss: 0.2751971115084255
Epoch 11/54, Loss: 0.2680703142986578
Epoch 12/54, Loss: 0.2650872823946616
Epoch 13/54, Loss: 0.26585452170932994
Epoch 14/54, Loss: 0.26364300619153413
Epoch 15/54, Loss: 0.2674655270050554
Epoch 16/54, Loss: 0.2666870956035221
Epoch 17/54, Loss: 0.26432235626613393
Epoch 18/54, Loss: 0.2632666217053638
Epoch 19/54, Loss: 0.2609465959317544
Epoch 20/54, Loss: 0.26061474663369794
Epoch 21/54, Loss: 0.2638506271383342
Epoch 22/54, Loss: 0.2626206936205135
Epoch 23/54, Loss: 0.2618796400287572
Epoch 24/54, Loss: 0.2602012823609745
Epoch 25/54, Loss: 0.25730833279735904
Epoch 26/54, Loss: 0.25743913080762415
Epoch 27/54, Loss: 0.25818133398013954
Epoch 28/54, Loss: 0.25797849893569946
Epoch 29/54, Loss: 0.2576768174767494
Epoch 30/54, Loss: 0.2594566424103344
Epoch 31/54, Loss: 0.25759100125116463
Epoch 32/54, Loss: 0.2562377978773678
Epoch 33/54, Loss: 0.2598079140571987
Epoch 34/54, Loss: 0.25628551358685775
Epoch 35/54, Loss: 0.2637347557088908
Epoch 36/54, Loss: 0.2568429655888501
Epoch 37/54, Loss: 0.2562098042929874
Epoch 38/54, Loss: 0.2576974082519026
Epoch 39/54, Loss: 0.2581775223507601
Epoch 40/54, Loss: 0.25901321014937234
Epoch 41/54, Loss: 0.2586892768740654
Epoch 42/54, Loss: 0.25920607719351263
Epoch 43/54, Loss: 0.25879832883091536
```

```
Epoch 44/54, Loss: 0.2600160473409821
Epoch 45/54, Loss: 0.2574441744124188
Epoch 46/54, Loss: 0.2590729037628454
Epoch 47/54, Loss: 0.2594422220307238
Epoch 48/54, Loss: 0.2570780822459389
Epoch 49/54, Loss: 0.25903315228574414
Epoch 50/54, Loss: 0.256692540119676
Epoch 51/54, Loss: 0.2575143163695055
Epoch 52/54, Loss: 0.2572448253631592
Epoch 53/54, Loss: 0.2570572802249123
Epoch 54/54, Loss: 0.25752336488050576
Epoch 1/46, Loss: 0.3654412342345013
Epoch 2/46, Loss: 0.2875844757346546
Epoch 3/46, Loss: 0.28344088924281735
Epoch 4/46, Loss: 0.28769102999392676
Epoch 5/46, Loss: 0.28095581952263327
Epoch 6/46, Loss: 0.2822728029945317
Epoch 7/46, Loss: 0.2831425539710942
Epoch 8/46, Loss: 0.2791733154479195
Epoch 9/46, Loss: 0.2729643166941755
Epoch 10/46, Loss: 0.2689959512037389
Epoch 11/46, Loss: 0.27092034062918496
Epoch 12/46, Loss: 0.26925716505331154
Epoch 13/46, Loss: 0.26630835015984144
Epoch 14/46, Loss: 0.26681238106068444
Epoch 15/46, Loss: 0.26646171904662075
Epoch 16/46, Loss: 0.2643090184120571
Epoch 17/46, Loss: 0.26592278655837565
Epoch 18/46, Loss: 0.2618267685174942
Epoch 19/46, Loss: 0.26049090132993813
Epoch 20/46, Loss: 0.2620768739896662
Epoch 21/46, Loss: 0.2643744261825786
Epoch 22/46, Loss: 0.2605428415186265
Epoch 23/46, Loss: 0.26096108029870424
Epoch 24/46, Loss: 0.25843818398082957
Epoch 25/46, Loss: 0.25841307114152345
Epoch 26/46, Loss: 0.25856022229965997
Epoch 27/46, Loss: 0.25709284579052644
Epoch 28/46, Loss: 0.2585061854299377
Epoch 29/46, Loss: 0.2629420901922619
Epoch 30/46, Loss: 0.2586070518283283
Epoch 31/46, Loss: 0.25845949702403126
Epoch 32/46, Loss: 0.25738234598847
Epoch 33/46, Loss: 0.26021996359614763
Epoch 34/46, Loss: 0.2576749018009971
Epoch 35/46, Loss: 0.258249617236502
Epoch 36/46, Loss: 0.259333717472413
Epoch 37/46, Loss: 0.26088991059976463
```

```
Epoch 38/46, Loss: 0.2598315319594215
Epoch 39/46, Loss: 0.25720686246367064
Epoch 40/46, Loss: 0.25683978988843803
Epoch 41/46, Loss: 0.2583346287993824
Epoch 42/46, Loss: 0.25800596659674363
Epoch 43/46, Loss: 0.25679948487702536
Epoch 44/46, Loss: 0.2598065006382325
Epoch 45/46, Loss: 0.25996507090680737
Epoch 46/46, Loss: 0.2616917149108999
Epoch 1/77, Loss: 0.34531217112260704
Epoch 2/77, Loss: 0.2839575195137192
Epoch 3/77, Loss: 0.284247343592784
Epoch 4/77, Loss: 0.2775595030363868
Epoch 5/77, Loss: 0.2796734922072467
Epoch 6/77, Loss: 0.2716023330302799
Epoch 7/77, Loss: 0.26708662729052934
Epoch 8/77, Loss: 0.271916241330259
Epoch 9/77, Loss: 0.26812486525844126
Epoch 10/77, Loss: 0.2658146711833337
Epoch 11/77, Loss: 0.26726019996054035
Epoch 12/77, Loss: 0.26134299837491093
Epoch 13/77, Loss: 0.2615569602040684
Epoch 14/77, Loss: 0.2592098997796283
Epoch 15/77, Loss: 0.2640875840888304
Epoch 16/77, Loss: 0.26183967379962697
Epoch 17/77, Loss: 0.26236293377245173
Epoch 18/77, Loss: 0.2627718417960055
Epoch 19/77, Loss: 0.25865525094901815
Epoch 20/77, Loss: 0.2585969694397029
Epoch 21/77, Loss: 0.2602872603079852
Epoch 22/77, Loss: 0.25835325656568303
Epoch 23/77, Loss: 0.26274557849940133
Epoch 24/77, Loss: 0.263417248340214
Epoch 25/77, Loss: 0.25983145394745993
Epoch 26/77, Loss: 0.25816697451998205
Epoch 27/77, Loss: 0.25958147469688864
Epoch 28/77, Loss: 0.25701578487368193
Epoch 29/77, Loss: 0.2571096954976811
Epoch 30/77, Loss: 0.2579299632240744
Epoch 31/77, Loss: 0.25608010414768667
Epoch 32/77, Loss: 0.2586235531112727
Epoch 33/77, Loss: 0.258211010519196
Epoch 34/77, Loss: 0.2595254511517637
Epoch 35/77, Loss: 0.2627958168878275
Epoch 36/77, Loss: 0.2580826142254998
Epoch 37/77, Loss: 0.25943753386245055
Epoch 38/77, Loss: 0.2583411848720382
Epoch 39/77, Loss: 0.25832561841782403
```

```
Epoch 40/77, Loss: 0.2594277429230073
Epoch 41/77, Loss: 0.25844430660500245
Epoch 42/77, Loss: 0.2624232488520005
Epoch 43/77, Loss: 0.2588414115940823
Epoch 44/77, Loss: 0.25735942274332047
Epoch 45/77, Loss: 0.25842491931775036
Epoch 46/77, Loss: 0.2593737463740742
Epoch 47/77, Loss: 0.26027484325801625
Epoch 48/77, Loss: 0.26194551850066466
Epoch 49/77, Loss: 0.2592057392877691
Epoch 50/77, Loss: 0.25638456554973826
Epoch 51/77, Loss: 0.25723042821182923
Epoch 52/77, Loss: 0.2566400113351205
Epoch 53/77, Loss: 0.2573145058225183
Epoch 54/77, Loss: 0.2569368626264965
Epoch 55/77, Loss: 0.2568005305879256
Epoch 56/77, Loss: 0.2580556387410444
Epoch 57/77, Loss: 0.25761709800537896
Epoch 58/77, Loss: 0.2559325857197537
Epoch 59/77, Loss: 0.2574927859446582
Epoch 60/77, Loss: 0.25923488332944755
Epoch 61/77, Loss: 0.25730465703150807
Epoch 62/77, Loss: 0.2587362869697459
Epoch 63/77, Loss: 0.2561690899379113
Epoch 64/77, Loss: 0.2579759990467745
Epoch 65/77, Loss: 0.26063071103656993
Epoch 66/77, Loss: 0.2579526664579616
Epoch 67/77, Loss: 0.25763209646239
Epoch 68/77, Loss: 0.26120562456986485
Epoch 69/77, Loss: 0.25868146998040814
Epoch 70/77, Loss: 0.25771590760525537
Epoch 71/77, Loss: 0.25729399831856
Epoch 72/77, Loss: 0.25759803196963144
Epoch 73/77, Loss: 0.2563952071701779
Epoch 74/77, Loss: 0.2569834458477357
Epoch 75/77, Loss: 0.2557552369201885
Epoch 76/77, Loss: 0.25686322371749315
Epoch 77/77, Loss: 0.2594624030239442
Epoch 1/88, Loss: 0.3523083136362188
Epoch 2/88, Loss: 0.275856857352397
Epoch 3/88, Loss: 0.2808494467069121
Epoch 4/88, Loss: 0.28974389065714445
Epoch 5/88, Loss: 0.2733323679250829
Epoch 6/88, Loss: 0.26750795981463266
Epoch 7/88, Loss: 0.26564931562718225
Epoch 8/88, Loss: 0.2657832665478482
Epoch 9/88, Loss: 0.2642484245931401
Epoch 10/88, Loss: 0.2613281331518117
```

```
Epoch 11/88, Loss: 0.2609909632626702
Epoch 12/88, Loss: 0.25901902072569905
Epoch 13/88, Loss: 0.25811635483713713
Epoch 14/88, Loss: 0.25761546764303656
Epoch 15/88, Loss: 0.2592400860260515
Epoch 16/88, Loss: 0.2585037004421739
Epoch 17/88, Loss: 0.2641719334265765
Epoch 18/88, Loss: 0.2626854632707203
Epoch 19/88, Loss: 0.2582782436819637
Epoch 20/88, Loss: 0.2582400042344542
Epoch 21/88, Loss: 0.25988510338699117
Epoch 22/88, Loss: 0.25758231518899694
Epoch 23/88, Loss: 0.25939010565771775
Epoch 24/88, Loss: 0.25804830649319815
Epoch 25/88, Loss: 0.2592047641382498
Epoch 26/88, Loss: 0.2596530405914082
Epoch 27/88, Loss: 0.25804805887096066
Epoch 28/88, Loss: 0.25639766805312214
Epoch 29/88, Loss: 0.2595878177705933
Epoch 30/88, Loss: 0.2642588032519116
Epoch 31/88, Loss: 0.2602187934167245
Epoch 32/88, Loss: 0.25662031068521385
Epoch 33/88, Loss: 0.2578548587420407
Epoch 34/88, Loss: 0.2586044520139694
Epoch 35/88, Loss: 0.25835852149654837
Epoch 36/88, Loss: 0.25728993950521245
Epoch 37/88, Loss: 0.25693635949317145
Epoch 38/88, Loss: 0.25846608289900946
Epoch 39/88, Loss: 0.26179778444416385
Epoch 40/88, Loss: 0.2603049488628612
Epoch 41/88, Loss: 0.25860753874568376
Epoch 42/88, Loss: 0.2574538365006447
Epoch 43/88, Loss: 0.2594594911617391
Epoch 44/88, Loss: 0.2568671265069176
Epoch 45/88, Loss: 0.25635626079405055
Epoch 46/88, Loss: 0.25677015588564034
Epoch 47/88, Loss: 0.25849633663892746
Epoch 48/88, Loss: 0.2578962554826456
Epoch 49/88, Loss: 0.2580518227289705
Epoch 50/88, Loss: 0.25762637573129993
Epoch 51/88, Loss: 0.2593998225296245
Epoch 52/88, Loss: 0.25723764431827206
Epoch 53/88, Loss: 0.2569874091183438
Epoch 54/88, Loss: 0.2606802367988755
Epoch 55/88, Loss: 0.25784938650972705
Epoch 56/88, Loss: 0.256706825512297
Epoch 57/88, Loss: 0.25897286876159553
Epoch 58/88, Loss: 0.2572199877570657
```

```
Epoch 59/88, Loss: 0.25791214757105885
Epoch 60/88, Loss: 0.2564193448599647
Epoch 61/88, Loss: 0.25855065356282625
Epoch 62/88, Loss: 0.2590681266258745
Epoch 63/88, Loss: 0.2571376373662668
Epoch 64/88, Loss: 0.256528876721859
Epoch 65/88, Loss: 0.2553456885849728
Epoch 66/88, Loss: 0.2580268426853068
Epoch 67/88, Loss: 0.25636039192185683
Epoch 68/88, Loss: 0.2558739365023725
Epoch 69/88, Loss: 0.2664632459773737
Epoch 70/88, Loss: 0.25739311996628256
Epoch 71/88, Loss: 0.2563255521304467
Epoch 72/88, Loss: 0.25571035243132534
Epoch 73/88, Loss: 0.25718293339014053
Epoch 74/88, Loss: 0.2566426422666101
Epoch 75/88, Loss: 0.26077500161002665
Epoch 76/88, Loss: 0.25736912953502994
Epoch 77/88, Loss: 0.2565006622496773
Epoch 78/88, Loss: 0.25748666200567694
Epoch 79/88, Loss: 0.25535154123516646
Epoch 80/88, Loss: 0.2579994736348881
Epoch 81/88, Loss: 0.258583689437193
Epoch 82/88, Loss: 0.25532614308245044
Epoch 83/88, Loss: 0.25529149525305805
Epoch 84/88, Loss: 0.25568382792613087
Epoch 85/88, Loss: 0.2549855310250731
Epoch 86/88, Loss: 0.25655410701737685
Epoch 87/88, Loss: 0.25580363194732103
Epoch 88/88, Loss: 0.2564062164986835
Epoch 1/92, Loss: 0.6471839389380287
Epoch 2/92, Loss: 0.5776530662003685
Epoch 3/92, Loss: 0.503771507564713
Epoch 4/92, Loss: 0.4310476367964464
Epoch 5/92, Loss: 0.36862697057864247
Epoch 6/92, Loss: 0.32903068205889535
Epoch 7/92, Loss: 0.30190009536112056
Epoch 8/92, Loss: 0.29268571515293684
Epoch 9/92, Loss: 0.28009168409249363
Epoch 10/92, Loss: 0.2785845937974313
Epoch 11/92, Loss: 0.27420954011818943
Epoch 12/92, Loss: 0.27904695710715127
Epoch 13/92, Loss: 0.27694885651854906
Epoch 14/92, Loss: 0.2807302746702643
Epoch 15/92, Loss: 0.295533434871365
Epoch 16/92, Loss: 0.2778138890862465
Epoch 17/92, Loss: 0.27832721787340503
Epoch 18/92, Loss: 0.28104641113211126
```

```
Epoch 19/92, Loss: 0.275393607423586
Epoch 20/92, Loss: 0.2782246544957161
Epoch 21/92, Loss: 0.2798420111922657
Epoch 22/92, Loss: 0.2806289651814629
Epoch 23/92, Loss: 0.2774532446089913
Epoch 24/92, Loss: 0.2774205317392069
Epoch 25/92, Loss: 0.2785237738314797
Epoch 26/92, Loss: 0.2775718946667278
Epoch 27/92, Loss: 0.2777813532773186
Epoch 28/92, Loss: 0.27703774457468705
Epoch 29/92, Loss: 0.27820905867744894
Epoch 30/92, Loss: 0.2796553991296712
Epoch 31/92, Loss: 0.2767624136279611
Epoch 32/92, Loss: 0.2809002465184997
Epoch 33/92, Loss: 0.2781602280104862
Epoch 34/92, Loss: 0.2779223248362541
Epoch 35/92, Loss: 0.27699508035884185
Epoch 36/92, Loss: 0.27766840510508595
Epoch 37/92, Loss: 0.2847238906166133
Epoch 38/92, Loss: 0.27872905178981666
Epoch 39/92, Loss: 0.28003132737734737
Epoch 40/92, Loss: 0.2771023450528874
Epoch 41/92, Loss: 0.2779088370940265
Epoch 42/92, Loss: 0.27878644317388535
Epoch 43/92, Loss: 0.2771194612278658
Epoch 44/92, Loss: 0.27682819217443466
Epoch 45/92, Loss: 0.2770465266178636
Epoch 46/92, Loss: 0.2778370367253528
Epoch 47/92, Loss: 0.2948714713839924
Epoch 48/92, Loss: 0.27749062680146275
Epoch 49/92, Loss: 0.2774728567284696
Epoch 50/92, Loss: 0.27619532495737076
Epoch 51/92, Loss: 0.27971114831812244
Epoch 52/92, Loss: 0.27859338285291896
Epoch 53/92, Loss: 0.27979783626163707
Epoch 54/92, Loss: 0.27707064064110026
Epoch 55/92, Loss: 0.27804754060857434
Epoch 56/92, Loss: 0.2763952367446002
Epoch 57/92, Loss: 0.27688282026964073
Epoch 58/92, Loss: 0.27665650669266195
Epoch 59/92, Loss: 0.27617953410919976
Epoch 60/92, Loss: 0.2781627967077143
Epoch 61/92, Loss: 0.276657277608619
Epoch 62/92, Loss: 0.2796400697792278
Epoch 63/92, Loss: 0.28215578680529313
Epoch 64/92, Loss: 0.2784734828507199
Epoch 65/92, Loss: 0.2787729050306713
Epoch 66/92, Loss: 0.2777431664221427
```

```
Epoch 67/92, Loss: 0.27935952605570064
Epoch 68/92, Loss: 0.27644623684532504
Epoch 69/92, Loss: 0.27877169891315345
Epoch 70/92, Loss: 0.2771475731449969
Epoch 71/92, Loss: 0.28220798907911077
Epoch 72/92, Loss: 0.2762136165710056
Epoch 73/92, Loss: 0.27558956952656016
Epoch 74/92, Loss: 0.2784123907194418
Epoch 75/92, Loss: 0.27666676132118
Epoch 76/92, Loss: 0.2775290314765537
Epoch 77/92, Loss: 0.2772951389060301
Epoch 78/92, Loss: 0.2764018896748038
Epoch 79/92, Loss: 0.27970361315152226
Epoch 80/92, Loss: 0.27505551716860605
Epoch 81/92, Loss: 0.2760155108921668
Epoch 82/92, Loss: 0.27637779099099774
Epoch 83/92, Loss: 0.27267173370894265
Epoch 84/92, Loss: 0.27281295990242677
Epoch 85/92, Loss: 0.274813769494786
Epoch 86/92, Loss: 0.2751917335040429
Epoch 87/92, Loss: 0.27371034508242326
Epoch 88/92, Loss: 0.2749754589270143
Epoch 89/92, Loss: 0.27368927834665074
Epoch 90/92, Loss: 0.2735914005076184
Epoch 91/92, Loss: 0.27268219081794515
Epoch 92/92, Loss: 0.273443547241828
Epoch 1/50, Loss: 0.3472925266798805
Epoch 2/50, Loss: 0.29263875019900937
Epoch 3/50, Loss: 0.2796617115245146
Epoch 4/50, Loss: 0.28333008902914386
Epoch 5/50, Loss: 0.27332615282605677
Epoch 6/50, Loss: 0.2722254974000594
Epoch 7/50, Loss: 0.2676156364819583
Epoch 8/50, Loss: 0.2666147364413037
Epoch 9/50, Loss: 0.26895179205081043
Epoch 10/50, Loss: 0.26687905104721293
Epoch 11/50, Loss: 0.2606630101799965
Epoch 12/50, Loss: 0.26162295218776255
Epoch 13/50, Loss: 0.258848880581996
Epoch 14/50, Loss: 0.26240671732846427
Epoch 15/50, Loss: 0.2588687569779508
Epoch 16/50, Loss: 0.25967371726737304
Epoch 17/50, Loss: 0.2590338539551286
Epoch 18/50, Loss: 0.2636450682492817
Epoch 19/50, Loss: 0.2580957285621587
Epoch 20/50, Loss: 0.2598329383660765
Epoch 21/50, Loss: 0.2577785779448116
Epoch 22/50, Loss: 0.25782986949471864
```

```
Epoch 23/50, Loss: 0.2576594155500917
Epoch 24/50, Loss: 0.2608074821970042
Epoch 25/50, Loss: 0.26243699122877684
Epoch 26/50, Loss: 0.26079656183719635
Epoch 27/50, Loss: 0.26204144165796395
Epoch 28/50, Loss: 0.2608478165724698
Epoch 29/50, Loss: 0.2579352811855428
Epoch 30/50, Loss: 0.25671369100318236
Epoch 31/50, Loss: 0.2602868715629858
Epoch 32/50, Loss: 0.25782504327156963
Epoch 33/50, Loss: 0.25725402551538806
Epoch 34/50, Loss: 0.257300903691965
Epoch 35/50, Loss: 0.25678516442284866
Epoch 36/50, Loss: 0.25856374335639615
Epoch 37/50, Loss: 0.2573821939089719
Epoch 38/50, Loss: 0.25828589279862013
Epoch 39/50, Loss: 0.25949111142579245
Epoch 40/50, Loss: 0.25884078530704274
Epoch 41/50, Loss: 0.25906040782437606
Epoch 42/50, Loss: 0.2578340150854167
Epoch 43/50, Loss: 0.2608337538207279
Epoch 44/50, Loss: 0.25916219152071895
Epoch 45/50, Loss: 0.25837575205985236
Epoch 46/50, Loss: 0.25725021914524193
Epoch 47/50, Loss: 0.2580132699188064
Epoch 48/50, Loss: 0.2580694674569018
Epoch 49/50, Loss: 0.2582973864148645
Epoch 50/50, Loss: 0.2577510204385309
Epoch 1/54, Loss: 0.45343958148184943
Epoch 2/54, Loss: 0.29000428932554584
Epoch 3/54, Loss: 0.28154344199334874
Epoch 4/54, Loss: 0.2821093381327741
Epoch 5/54, Loss: 0.2751236676293261
Epoch 6/54, Loss: 0.2819397668628132
Epoch 7/54, Loss: 0.2805289293036741
Epoch 8/54, Loss: 0.28112976384513516
Epoch 9/54, Loss: 0.28028982045019374
Epoch 10/54, Loss: 0.28042782799286003
Epoch 11/54, Loss: 0.2743808324722683
Epoch 12/54, Loss: 0.27510836808120503
Epoch 13/54, Loss: 0.2895660352180986
Epoch 14/54, Loss: 0.2729920560822767
Epoch 15/54, Loss: 0.2733631059527397
Epoch 16/54, Loss: 0.286709903355907
Epoch 17/54, Loss: 0.27155017239205975
Epoch 18/54, Loss: 0.2760668011272655
Epoch 19/54, Loss: 0.2675730450188412
Epoch 20/54, Loss: 0.267260264386149
```

```
Epoch 21/54, Loss: 0.26610130948178906
Epoch 22/54, Loss: 0.2670102150124662
Epoch 23/54, Loss: 0.2697811884915127
Epoch 24/54, Loss: 0.26845483306576223
Epoch 25/54, Loss: 0.26856377063428655
Epoch 26/54, Loss: 0.26723261177539825
Epoch 27/54, Loss: 0.2684856679509668
Epoch 28/54, Loss: 0.2671234791769701
Epoch 29/54, Loss: 0.2657447132994147
Epoch 30/54, Loss: 0.26561054105267806
Epoch 31/54, Loss: 0.2667656834511196
Epoch 32/54, Loss: 0.2754650479730438
Epoch 33/54, Loss: 0.2647649452966802
Epoch 34/54, Loss: 0.27283137204015956
Epoch 35/54, Loss: 0.2656917738563874
Epoch 36/54, Loss: 0.2671697744551827
Epoch 37/54, Loss: 0.26602479245732813
Epoch 38/54, Loss: 0.26360296019736457
Epoch 39/54, Loss: 0.26260432206532536
Epoch 40/54, Loss: 0.2633124256835264
Epoch 41/54, Loss: 0.2630431270774673
Epoch 42/54, Loss: 0.2608683087369975
Epoch 43/54, Loss: 0.2617461738340995
Epoch 44/54, Loss: 0.27050381723572225
Epoch 45/54, Loss: 0.2649926205768305
Epoch 46/54, Loss: 0.2606356845182531
Epoch 47/54, Loss: 0.26132872043287053
Epoch 48/54, Loss: 0.2609984195407699
Epoch 49/54, Loss: 0.2580026659895392
Epoch 50/54, Loss: 0.2606593611485818
Epoch 51/54, Loss: 0.25826943578088984
Epoch 52/54, Loss: 0.25945624533821554
Epoch 53/54, Loss: 0.2581210697398466
Epoch 54/54, Loss: 0.26201051517444496
Epoch 1/98, Loss: 0.3671688599621548
Epoch 2/98, Loss: 0.28037888556718826
Epoch 3/98, Loss: 0.2732788683737026
Epoch 4/98, Loss: 0.278383245362955
Epoch 5/98, Loss: 0.27905276943655577
Epoch 6/98, Loss: 0.2686499348458122
Epoch 7/98, Loss: 0.2714366785743657
Epoch 8/98, Loss: 0.2687480454059208
Epoch 9/98, Loss: 0.26652860115556154
Epoch 10/98, Loss: 0.2634766693500912
Epoch 11/98, Loss: 0.262813562856001
Epoch 12/98, Loss: 0.26015886006986394
Epoch 13/98, Loss: 0.26174550766454024
Epoch 14/98, Loss: 0.26070619637475295
```

```
Epoch 15/98, Loss: 0.2588798000532038
Epoch 16/98, Loss: 0.26073071930338354
Epoch 17/98, Loss: 0.25719743134344325
Epoch 18/98, Loss: 0.25910987310549793
Epoch 19/98, Loss: 0.25892741496072097
Epoch 20/98, Loss: 0.2567922928754021
Epoch 21/98, Loss: 0.2604347436743624
Epoch 22/98, Loss: 0.2601618639686528
Epoch 23/98, Loss: 0.2589425799601218
Epoch 24/98, Loss: 0.2580373449360623
Epoch 25/98, Loss: 0.259591685498462
Epoch 26/98, Loss: 0.25776244217858596
Epoch 27/98, Loss: 0.2604787020998843
Epoch 28/98, Loss: 0.26302558578112545
Epoch 29/98, Loss: 0.25917620474801345
Epoch 30/98, Loss: 0.2600132340894026
Epoch 31/98, Loss: 0.25799165008699193
Epoch 32/98, Loss: 0.25818869909819436
Epoch 33/98, Loss: 0.25812314538394704
Epoch 34/98, Loss: 0.2618228043703472
Epoch 35/98, Loss: 0.25856679677963257
Epoch 36/98, Loss: 0.2610721163013402
Epoch 37/98, Loss: 0.25923937734435587
Epoch 38/98, Loss: 0.2574864599634619
Epoch 39/98, Loss: 0.2632451671011308
Epoch 40/98, Loss: 0.25734226142658906
Epoch 41/98, Loss: 0.2570779034320046
Epoch 42/98, Loss: 0.2597135489477831
Epoch 43/98, Loss: 0.2586127778186518
Epoch 44/98, Loss: 0.2566501476308879
Epoch 45/98, Loss: 0.2569672763347626
Epoch 46/98, Loss: 0.258411230848116
Epoch 47/98, Loss: 0.2584922787021188
Epoch 48/98, Loss: 0.2614578911486794
Epoch 49/98, Loss: 0.25889231615206776
Epoch 50/98, Loss: 0.2566081364365185
Epoch 51/98, Loss: 0.25894343940650716
Epoch 52/98, Loss: 0.25797125258866477
Epoch 53/98, Loss: 0.26060220993617
Epoch 54/98, Loss: 0.25937776530490203
Epoch 55/98, Loss: 0.25920362288461013
Epoch 56/98, Loss: 0.25740989209974513
Epoch 57/98, Loss: 0.2565600569633877
Epoch 58/98, Loss: 0.2570183746078435
Epoch 59/98, Loss: 0.25856639532481923
Epoch 60/98, Loss: 0.258944756406195
Epoch 61/98, Loss: 0.25856951930943656
Epoch 62/98, Loss: 0.2613958212382653
```

```
Epoch 63/98, Loss: 0.2580718954696375
Epoch 64/98, Loss: 0.2591259444461149
Epoch 65/98, Loss: 0.257216132739011
Epoch 66/98, Loss: 0.2558747236342991
Epoch 67/98, Loss: 0.25764205745037866
Epoch 68/98, Loss: 0.25659384359331694
Epoch 69/98, Loss: 0.2572044239324682
Epoch 70/98, Loss: 0.2586436490802204
Epoch 71/98, Loss: 0.25815850277157393
Epoch 72/98, Loss: 0.2573608458042145
Epoch 73/98, Loss: 0.26123074398321267
Epoch 74/98, Loss: 0.2593574353000697
Epoch 75/98, Loss: 0.2566813725759001
Epoch 76/98, Loss: 0.2575815135941786
Epoch 77/98, Loss: 0.2588781350675751
Epoch 78/98, Loss: 0.25758341393050027
Epoch 79/98, Loss: 0.2582486938027775
Epoch 80/98, Loss: 0.2561750376925749
Epoch 81/98, Loss: 0.2569317664293682
Epoch 82/98, Loss: 0.25714873128077564
Epoch 83/98, Loss: 0.2570611115764169
Epoch 84/98, Loss: 0.2575600835330346
Epoch 85/98, Loss: 0.2567279295009725
Epoch 86/98, Loss: 0.25768261227537603
Epoch 87/98, Loss: 0.2564363343750729
Epoch 88/98, Loss: 0.25798164833994475
Epoch 89/98, Loss: 0.2562793659813264
Epoch 90/98, Loss: 0.25630495609606013
Epoch 91/98, Loss: 0.25698640898746605
Epoch 92/98, Loss: 0.2575920268016703
Epoch 93/98, Loss: 0.2564646286122939
Epoch 94/98, Loss: 0.25625433071571235
Epoch 95/98, Loss: 0.25771833561799107
Epoch 96/98, Loss: 0.2583060970201212
Epoch 97/98, Loss: 0.25553521250977235
Epoch 98/98, Loss: 0.2551121847594486
Epoch 1/30, Loss: 0.38987260048880296
Epoch 2/30, Loss: 0.28644001440090294
Epoch 3/30, Loss: 0.2840774572070907
Epoch 4/30, Loss: 0.2800211152609657
Epoch 5/30, Loss: 0.2862597007085295
Epoch 6/30, Loss: 0.278452023425523
Epoch 7/30, Loss: 0.279772063388544
Epoch 8/30, Loss: 0.27927649152629513
Epoch 9/30, Loss: 0.28119754309163375
Epoch 10/30, Loss: 0.2671806435374653
Epoch 11/30, Loss: 0.28517599009415684
Epoch 12/30, Loss: 0.2685929230030845
```

```
Epoch 13/30, Loss: 0.2674425533589195
Epoch 14/30, Loss: 0.2682940951165031
Epoch 15/30, Loss: 0.2684032299062785
Epoch 16/30, Loss: 0.27120276496690865
Epoch 17/30, Loss: 0.2670919724247035
Epoch 18/30, Loss: 0.266064113992102
Epoch 19/30, Loss: 0.2638118218849687
Epoch 20/30, Loss: 0.2636488527059555
Epoch 21/30, Loss: 0.26156008287387733
Epoch 22/30, Loss: 0.2607739913989516
Epoch 23/30, Loss: 0.26210372588213754
Epoch 24/30, Loss: 0.26190361818846536
Epoch 25/30, Loss: 0.25767629812745485
Epoch 26/30, Loss: 0.25860946493990283
Epoch 27/30, Loss: 0.2582666764364523
Epoch 28/30, Loss: 0.2591743903125034
Epoch 29/30, Loss: 0.2595633680329603
Epoch 30/30, Loss: 0.25665098823168697
Best hyperparameters:  [152, 128, 0.0001930783753654713, 0.46035538917954116,
37, 81]
Epoch 1/81, Loss: 0.6438096253310933
Epoch 2/81, Loss: 0.5118430958074682
Epoch 3/81, Loss: 0.39146261180148406
Epoch 4/81, Loss: 0.32357874293537703
Epoch 5/81, Loss: 0.286073270966025
Epoch 6/81, Loss: 0.2777792416951236
Epoch 7/81, Loss: 0.2789551650776583
Epoch 8/81, Loss: 0.27718352263464646
Epoch 9/81, Loss: 0.28259100502028184
Epoch 10/81, Loss: 0.2855961287722868
Epoch 11/81, Loss: 0.28486348250333
Epoch 12/81, Loss: 0.28271739710779753
Epoch 13/81, Loss: 0.28143367916345596
Epoch 14/81, Loss: 0.2812917101032594
Epoch 15/81, Loss: 0.2819855725940536
Epoch 16/81, Loss: 0.2820233934942414
Epoch 17/81, Loss: 0.2925506221020923
Epoch 18/81, Loss: 0.2934553123572293
Epoch 19/81, Loss: 0.28385331437868233
Epoch 20/81, Loss: 0.27930624783039093
Epoch 21/81, Loss: 0.28466113043182034
Epoch 22/81, Loss: 0.28003669091883826
Epoch 23/81, Loss: 0.2782539659563233
Epoch 24/81, Loss: 0.2771615495576578
Epoch 25/81, Loss: 0.2793461141340873
Epoch 26/81, Loss: 0.2795071119771284
Epoch 27/81, Loss: 0.27829935270197254
Epoch 28/81, Loss: 0.28061799924163255
```

```
Epoch 29/81, Loss: 0.27705031107453737
Epoch 30/81, Loss: 0.278854980626527
Epoch 31/81, Loss: 0.2766479481669033
Epoch 32/81, Loss: 0.2778523586252156
Epoch 33/81, Loss: 0.27913130907451406
Epoch 34/81, Loss: 0.2779468582833515
Epoch 35/81, Loss: 0.275718839291264
Epoch 36/81, Loss: 0.27598927170038223
Epoch 37/81, Loss: 0.28753337307887916
Epoch 38/81, Loss: 0.2730820003677817
Epoch 39/81, Loss: 0.27398384932209463
Epoch 40/81, Loss: 0.273350320318166
Epoch 41/81, Loss: 0.27520616001942577
Epoch 42/81, Loss: 0.27479635924100876
Epoch 43/81, Loss: 0.27370432501330094
Epoch 44/81, Loss: 0.27541320639498096
Epoch 45/81, Loss: 0.27358991828034906
Epoch 46/81, Loss: 0.2761620204238331
Epoch 47/81, Loss: 0.26958464524325204
Epoch 48/81, Loss: 0.2721760939149296
Epoch 49/81, Loss: 0.2719789116698153
Epoch 50/81, Loss: 0.27038850810597925
Epoch 51/81, Loss: 0.2745180541978163
Epoch 52/81, Loss: 0.2874753137721735
Epoch 53/81, Loss: 0.276582468958462
Epoch 54/81, Loss: 0.26953492006834817
Epoch 55/81, Loss: 0.27472493841367607
Epoch 56/81, Loss: 0.2703366012257688
Epoch 57/81, Loss: 0.27179477802094293
Epoch 58/81, Loss: 0.2736628169522566
Epoch 59/81, Loss: 0.26917685612159614
Epoch 60/81, Loss: 0.2729307689210948
Epoch 61/81, Loss: 0.27202916101497765
Epoch 62/81, Loss: 0.2685103780206512
Epoch 63/81, Loss: 0.26725369954810424
Epoch 64/81, Loss: 0.26784923672676086
Epoch 65/81, Loss: 0.26716573536396027
Epoch 66/81, Loss: 0.2681339646086973
Epoch 67/81, Loss: 0.269324854016304
Epoch 68/81, Loss: 0.2699620412553058
Epoch 69/81, Loss: 0.2674934381947798
Epoch 70/81, Loss: 0.27274490629925446
Epoch 71/81, Loss: 0.26669962835662503
Epoch 72/81, Loss: 0.2668500809985049
Epoch 73/81, Loss: 0.26729664995389824
Epoch 74/81, Loss: 0.2651229838238043
Epoch 75/81, Loss: 0.26646827950197105
Epoch 76/81, Loss: 0.2662987945710911
```

```
Epoch 77/81, Loss: 0.26722948428462534
Epoch 78/81, Loss: 0.2649303749203682
Epoch 79/81, Loss: 0.27054643280365887
Epoch 80/81, Loss: 0.26596878907259774
Epoch 81/81, Loss: 0.2642576681340442
Best Binary Accuracy: 0.9101
Best Normal Accuracy: 0.2872
Best F1 Score: 0.9529
```

[ ]:
```
# 2.1.2 Deeper GCN with Residual Connections

import torch
import torch.nn.functional as F
from torch_geometric.nn import GCNConv, global_mean_pool
from torch_geometric.data import DataLoader
from sklearn.model_selection import train_test_split
from sklearn.metrics import f1_score, accuracy_score

# Define the Residual GCN Model
class ResidualGCN(torch.nn.Module):
    def __init__(self, num_node_features, num_classes):
        super(ResidualGCN, self).__init__()
        self.conv1 = GCNConv(num_node_features, 128)
        self.conv2 = GCNConv(128, 128)  # Residual connection from input to␣
↪output
        self.conv3 = GCNConv(128, 64)
        self.fc = torch.nn.Linear(64, num_classes)

    def forward(self, data):
        x, edge_index = data.x, data.edge_index
        x1 = self.conv1(x, edge_index)
        x1 = F.relu(x1)
        x2 = self.conv2(x1, edge_index) + x1  # Add residual connection
        x2 = F.relu(x2)
        x3 = self.conv3(x2, edge_index)
        x3 = F.relu(x3)
        x = global_mean_pool(x3, data.batch)
        return torch.sigmoid(self.fc(x))

# Function to train the GCN model
def train_gnn_model(model, train_loader, optimizer, criterion, num_epochs=50):
    model.train()
    for epoch in range(num_epochs):
        total_loss = 0
        for data in train_loader:
            optimizer.zero_grad()
            output = model(data)
```

```python
            loss = criterion(output, data.y)
            loss.backward()
            optimizer.step()
            total_loss += loss.item()
        print(f'Epoch {epoch + 1}/{num_epochs}, Loss: {total_loss / 
 ↪len(train_loader)}')

# Function to evaluate the GCN model
def evaluate_gnn_model(model, test_loader):
    model.eval()
    total_acc, total_f1, total_normal_acc = 0, 0, 0
    num_batches = len(test_loader)

    for data in test_loader:
        with torch.no_grad():
            output = model(data)
            preds = (output > 0.5).float()
            total_acc += (preds == data.y).sum().item() / data.y.numel()
            total_f1 += f1_score(data.y.cpu(), preds.cpu(), average='micro')
            total_normal_acc += accuracy_score(data.y.cpu(), preds.cpu())

    print(f'Binary Accuracy: {total_acc / num_batches:.4f}')
    print(f'F1 Score: {total_f1 / num_batches:.4f}')
    print(f'Normal Accuracy: {total_normal_acc / num_batches:.4f}')

# Initialize the Residual GCN Model and Optimizer
residual_gcn_model = ResidualGCN(num_node_features=1, num_classes=y_train.
 ↪shape[1])
optimizer = torch.optim.Adam(residual_gcn_model.parameters(), lr=0.001)
criterion = torch.nn.BCELoss()

# Train and Evaluate the Residual GCN Model
train_gnn_model(residual_gcn_model, train_loader, optimizer, criterion)
evaluate_gnn_model(residual_gcn_model, test_loader)
```

```
Epoch 1/50, Loss: 0.40977052599191666
Epoch 2/50, Loss: 0.2966996466412264
Epoch 3/50, Loss: 0.2852067767697222
Epoch 4/50, Loss: 0.28455961977734284
Epoch 5/50, Loss: 0.28143780853818445
Epoch 6/50, Loss: 0.28575728351578994
Epoch 7/50, Loss: 0.2870054731474203
Epoch 8/50, Loss: 0.28114549903308644
Epoch 9/50, Loss: 0.27869916575796466
Epoch 10/50, Loss: 0.2874220434357138
Epoch 11/50, Loss: 0.27915699034929276
Epoch 12/50, Loss: 0.27465080864289226
```

```
Epoch 13/50, Loss: 0.27165277828188505
Epoch 14/50, Loss: 0.2811236149247955
Epoch 15/50, Loss: 0.27249398783725853
Epoch 16/50, Loss: 0.2691582656081985
Epoch 17/50, Loss: 0.2711702418677947
Epoch 18/50, Loss: 0.2708361920188455
Epoch 19/50, Loss: 0.26770998844329047
Epoch 20/50, Loss: 0.26767198741436005
Epoch 21/50, Loss: 0.2668007103835835
Epoch 22/50, Loss: 0.26665472370736737
Epoch 23/50, Loss: 0.26569752088364434
Epoch 24/50, Loss: 0.2658405014697243
Epoch 25/50, Loss: 0.2636105869622791
Epoch 26/50, Loss: 0.2629723228952464
Epoch 27/50, Loss: 0.2620156626490986
Epoch 28/50, Loss: 0.26151294874794345
Epoch 29/50, Loss: 0.2618579995982787
Epoch 30/50, Loss: 0.2621135426794781
Epoch 31/50, Loss: 0.2605519093134824
Epoch 32/50, Loss: 0.26194778172408834
Epoch 33/50, Loss: 0.26001938344801173
Epoch 34/50, Loss: 0.25827040742425356
Epoch 35/50, Loss: 0.2584061009042403
Epoch 36/50, Loss: 0.25973284332191243
Epoch 37/50, Loss: 0.2594396520186873
Epoch 38/50, Loss: 0.26110191818545847
Epoch 39/50, Loss: 0.25800127886673985
Epoch 40/50, Loss: 0.2610445228569648
Epoch 41/50, Loss: 0.2594033418332829
Epoch 42/50, Loss: 0.26014680722180533
Epoch 43/50, Loss: 0.25907038809622035
Epoch 44/50, Loss: 0.26095925052376356
Epoch 45/50, Loss: 0.2617171073661131
Epoch 46/50, Loss: 0.25717610208427205
Epoch 47/50, Loss: 0.25822924340472503
Epoch 48/50, Loss: 0.25924597855876474
Epoch 49/50, Loss: 0.2572482828708256
Epoch 50/50, Loss: 0.2587040754802087
Binary Accuracy: 0.9098
F1 Score: 0.9527
Normal Accuracy: 0.2872
```

[ ]:
```
# GAT
import torch
import torch.nn.functional as F
from torch_geometric.nn import GATConv, global_mean_pool
from torch_geometric.data import DataLoader
```

```python
from sklearn.metrics import f1_score, accuracy_score

# GAT Model Definition
class GAT(torch.nn.Module):
    def __init__(self, num_node_features, num_classes, hidden_dim1,␣
↪hidden_dim2, heads):
        super(GAT, self).__init__()
        # GATConv layers, in_channels is a scalar representing the number of␣
↪node features
        self.conv1 = GATConv(in_channels=num_node_features,␣
↪out_channels=hidden_dim1, heads=heads)
        self.conv2 = GATConv(in_channels=hidden_dim1 * heads,␣
↪out_channels=hidden_dim2, heads=1)  # Single head for the second layer
        self.fc = torch.nn.Linear(hidden_dim2, num_classes)

    def forward(self, data):
        x, edge_index = data.x, data.edge_index
        x = self.conv1(x, edge_index)
        x = F.elu(x)
        x = self.conv2(x, edge_index)
        x = F.elu(x)
        x = global_mean_pool(x, data.batch)
        return torch.sigmoid(self.fc(x))

# Function to train the GAT model
def train_gnn_model(model, train_loader, optimizer, criterion, num_epochs=50):
    model.train()
    for epoch in range(num_epochs):
        total_loss = 0
        for data in train_loader:
            optimizer.zero_grad()
            output = model(data)
            loss = criterion(output, data.y)
            loss.backward()
            optimizer.step()
            total_loss += loss.item()
        print(f'Epoch {epoch + 1}/{num_epochs}, Loss: {total_loss /␣
↪len(train_loader)}')

# Function to evaluate the GAT model
def evaluate_gnn_model(model, test_loader):
    model.eval()
    total_acc, total_f1, total_normal_acc = 0, 0, 0
    num_batches = len(test_loader)

    for data in test_loader:
        with torch.no_grad():
```

```
            output = model(data)
            preds = (output > 0.5).float()
            total_acc += (preds == data.y).sum().item() / data.y.numel()
            total_f1 += f1_score(data.y.cpu(), preds.cpu(), average='micro')
            total_normal_acc += accuracy_score(data.y.cpu(), preds.cpu())

    print(f'Binary Accuracy: {total_acc / num_batches:.4f}')
    print(f'F1 Score: {total_f1 / num_batches:.4f}')
    print(f'Normal Accuracy: {total_normal_acc / num_batches:.4f}')

# Train and evaluate the GAT model
def run_gat_experiment():
    # Initialize the GAT model
    model = GAT(num_node_features=1, num_classes=y_train.shape[1],
 ↪hidden_dim1=128, hidden_dim2=64, heads=4)

    # Set up optimizer and criterion
    optimizer = torch.optim.Adam(model.parameters(), lr=0.001)
    criterion = torch.nn.BCELoss()

    # Create data loaders
    train_loader = DataLoader(train_graphs, batch_size=32, shuffle=True)
    test_loader = DataLoader(test_graphs, batch_size=32, shuffle=False)

    # Train the model
    train_gnn_model(model, train_loader, optimizer, criterion, num_epochs=50)

    # Evaluate the model
    evaluate_gnn_model(model, test_loader)

# Run the experiment
run_gat_experiment()
```

```
/usr/local/lib/python3.10/dist-packages/torch_geometric/deprecation.py:26:
UserWarning: 'data.DataLoader' is deprecated, use 'loader.DataLoader' instead
  warnings.warn(out)
```

```
Epoch 1/50, Loss: 0.35357677585938396
Epoch 2/50, Loss: 0.2659758837784038
Epoch 3/50, Loss: 0.26330991220824856
Epoch 4/50, Loss: 0.2618640428956817
Epoch 5/50, Loss: 0.26475968474850936
Epoch 6/50, Loss: 0.2615349752938046
Epoch 7/50, Loss: 0.2616846263408661
Epoch 8/50, Loss: 0.26836807850529165
Epoch 9/50, Loss: 0.2659115234718603
Epoch 10/50, Loss: 0.26005418993094387
Epoch 11/50, Loss: 0.2604568666395019
```

```
Epoch 12/50, Loss: 0.2614471960593672
Epoch 13/50, Loss: 0.2606346764985253
Epoch 14/50, Loss: 0.25947807290974784
Epoch 15/50, Loss: 0.2615856103160802
Epoch 16/50, Loss: 0.2603291838484652
Epoch 17/50, Loss: 0.26111758544164543
Epoch 18/50, Loss: 0.25819076247075023
Epoch 19/50, Loss: 0.25865598680341945
Epoch 20/50, Loss: 0.2590899918885792
Epoch 21/50, Loss: 0.26015544375952554
Epoch 22/50, Loss: 0.25740544103524265
Epoch 23/50, Loss: 0.26005468166926327
Epoch 24/50, Loss: 0.2572503940147512
Epoch 25/50, Loss: 0.2608996165149352
Epoch 26/50, Loss: 0.26254732424722
Epoch 27/50, Loss: 0.2582198333214311
Epoch 28/50, Loss: 0.2576981550630401
Epoch 29/50, Loss: 0.2587903220863903
Epoch 30/50, Loss: 0.2568041853168431
Epoch 31/50, Loss: 0.2612692095777568
Epoch 32/50, Loss: 0.257519414757981
Epoch 33/50, Loss: 0.25787246139610515
Epoch 34/50, Loss: 0.25730639389332605
Epoch 35/50, Loss: 0.25954180342309613
Epoch 36/50, Loss: 0.258446880561464
Epoch 37/50, Loss: 0.2601133926826365
Epoch 38/50, Loss: 0.25731691937236223
Epoch 39/50, Loss: 0.2584214591804673
Epoch 40/50, Loss: 0.2564886887283886
Epoch 41/50, Loss: 0.26155946929665175
Epoch 42/50, Loss: 0.25797222204068126
Epoch 43/50, Loss: 0.2586630797561477
Epoch 44/50, Loss: 0.25918686696711707
Epoch 45/50, Loss: 0.25712920856826443
Epoch 46/50, Loss: 0.25571835829931144
Epoch 47/50, Loss: 0.2564181602176498
Epoch 48/50, Loss: 0.25667040488299203
Epoch 49/50, Loss: 0.25807638247223463
Epoch 50/50, Loss: 0.25848092270248074
Binary Accuracy: 0.9097
F1 Score: 0.9527
Normal Accuracy: 0.2872
```

[ ]:
```
!pip install keras
```

```
Requirement already satisfied: keras in /usr/local/lib/python3.10/dist-packages
(3.4.1)
Requirement already satisfied: absl-py in /usr/local/lib/python3.10/dist-
```

```
packages (from keras) (1.4.0)
Requirement already satisfied: numpy in /usr/local/lib/python3.10/dist-packages
(from keras) (1.26.4)
Requirement already satisfied: rich in /usr/local/lib/python3.10/dist-packages
(from keras) (13.8.1)
Requirement already satisfied: namex in /usr/local/lib/python3.10/dist-packages
(from keras) (0.0.8)
Requirement already satisfied: h5py in /usr/local/lib/python3.10/dist-packages
(from keras) (3.11.0)
Requirement already satisfied: optree in /usr/local/lib/python3.10/dist-packages
(from keras) (0.12.1)
Requirement already satisfied: ml-dtypes in /usr/local/lib/python3.10/dist-
packages (from keras) (0.4.0)
Requirement already satisfied: packaging in /usr/local/lib/python3.10/dist-
packages (from keras) (24.1)
Requirement already satisfied: typing-extensions>=4.5.0 in
/usr/local/lib/python3.10/dist-packages (from optree->keras) (4.12.2)
Requirement already satisfied: markdown-it-py>=2.2.0 in
/usr/local/lib/python3.10/dist-packages (from rich->keras) (3.0.0)
Requirement already satisfied: pygments<3.0.0,>=2.13.0 in
/usr/local/lib/python3.10/dist-packages (from rich->keras) (2.16.1)
Requirement already satisfied: mdurl~=0.1 in /usr/local/lib/python3.10/dist-
packages (from markdown-it-py>=2.2.0->rich->keras) (0.1.2)
```

[ ]:
```
!pip install torch-scatter torch-sparse torch-geometric scikit-learn
```

```
Collecting torch-scatter
  Using cached torch_scatter-2.1.2.tar.gz (108 kB)
  Preparing metadata (setup.py) … done
Collecting torch-sparse
  Using cached torch_sparse-0.6.18.tar.gz (209 kB)
  Preparing metadata (setup.py) … done
Requirement already satisfied: torch-geometric in
/usr/local/lib/python3.10/dist-packages (2.6.0)
Requirement already satisfied: scikit-learn in /usr/local/lib/python3.10/dist-
packages (1.3.2)
Requirement already satisfied: scipy in /usr/local/lib/python3.10/dist-packages
(from torch-sparse) (1.13.1)
Requirement already satisfied: aiohttp in /usr/local/lib/python3.10/dist-
packages (from torch-geometric) (3.10.5)
Requirement already satisfied: fsspec in /usr/local/lib/python3.10/dist-packages
(from torch-geometric) (2024.6.1)
Requirement already satisfied: jinja2 in /usr/local/lib/python3.10/dist-packages
(from torch-geometric) (3.1.4)
Requirement already satisfied: numpy in /usr/local/lib/python3.10/dist-packages
(from torch-geometric) (1.26.4)
Requirement already satisfied: psutil>=5.8.0 in /usr/local/lib/python3.10/dist-
packages (from torch-geometric) (5.9.5)
```

```
Requirement already satisfied: pyparsing in /usr/local/lib/python3.10/dist-
packages (from torch-geometric) (3.1.4)
Requirement already satisfied: requests in /usr/local/lib/python3.10/dist-
packages (from torch-geometric) (2.32.3)
Requirement already satisfied: tqdm in /usr/local/lib/python3.10/dist-packages
(from torch-geometric) (4.66.5)
Requirement already satisfied: joblib>=1.1.1 in /usr/local/lib/python3.10/dist-
packages (from scikit-learn) (1.4.2)
Requirement already satisfied: threadpoolctl>=2.0.0 in
/usr/local/lib/python3.10/dist-packages (from scikit-learn) (3.5.0)
Requirement already satisfied: aiohappyeyeballs>=2.3.0 in
/usr/local/lib/python3.10/dist-packages (from aiohttp->torch-geometric) (2.4.0)
Requirement already satisfied: aiosignal>=1.1.2 in
/usr/local/lib/python3.10/dist-packages (from aiohttp->torch-geometric) (1.3.1)
Requirement already satisfied: attrs>=17.3.0 in /usr/local/lib/python3.10/dist-
packages (from aiohttp->torch-geometric) (24.2.0)
Requirement already satisfied: frozenlist>=1.1.1 in
/usr/local/lib/python3.10/dist-packages (from aiohttp->torch-geometric) (1.4.1)
Requirement already satisfied: multidict<7.0,>=4.5 in
/usr/local/lib/python3.10/dist-packages (from aiohttp->torch-geometric) (6.1.0)
Requirement already satisfied: yarl<2.0,>=1.0 in /usr/local/lib/python3.10/dist-
packages (from aiohttp->torch-geometric) (1.11.1)
Requirement already satisfied: async-timeout<5.0,>=4.0 in
/usr/local/lib/python3.10/dist-packages (from aiohttp->torch-geometric) (4.0.3)
Requirement already satisfied: MarkupSafe>=2.0 in
/usr/local/lib/python3.10/dist-packages (from jinja2->torch-geometric) (2.1.5)
Requirement already satisfied: charset-normalizer<4,>=2 in
/usr/local/lib/python3.10/dist-packages (from requests->torch-geometric) (3.3.2)
Requirement already satisfied: idna<4,>=2.5 in /usr/local/lib/python3.10/dist-
packages (from requests->torch-geometric) (3.8)
Requirement already satisfied: urllib3<3,>=1.21.1 in
/usr/local/lib/python3.10/dist-packages (from requests->torch-geometric) (2.0.7)
Requirement already satisfied: certifi>=2017.4.17 in
/usr/local/lib/python3.10/dist-packages (from requests->torch-geometric)
(2024.8.30)
Requirement already satisfied: typing-extensions>=4.1.0 in
/usr/local/lib/python3.10/dist-packages (from
multidict<7.0,>=4.5->aiohttp->torch-geometric) (4.12.2)
Building wheels for collected packages: torch-scatter, torch-sparse
  Building wheel for torch-scatter (setup.py) … done
  Created wheel for torch-scatter:
filename=torch_scatter-2.1.2-cp310-cp310-linux_x86_64.whl size=3658840
sha256=b35659ef244cb0df069834186c45267e4d52d45f83a4de88e4d7785194a4cb84
  Stored in directory: /root/.cache/pip/wheels/92/f1/2b/3b46d54b134259f58c836356
8569053248040859b1a145b3ce
  Building wheel for torch-sparse (setup.py) … done
  Created wheel for torch-sparse:
filename=torch_sparse-0.6.18-cp310-cp310-linux_x86_64.whl size=2809239
```

```
sha256=1ebd200f8e1ff5b16bf80a750e7435624fec26fcaab1462b6b73c6a3ddb3c72e
  Stored in directory: /root/.cache/pip/wheels/c9/dd/0f/a6a16f9f3b0236733d257b4b
4ea91b548b984a341ed3b8f38c
Successfully built torch-scatter torch-sparse
Installing collected packages: torch-scatter, torch-sparse
Successfully installed torch-scatter-2.1.2 torch-sparse-0.6.18
```

[ ]:
```
# Install Keras Tuner
!pip install keras-tuner
```

```
Requirement already satisfied: keras-tuner in /usr/local/lib/python3.10/dist-
packages (1.4.7)
Requirement already satisfied: keras in /usr/local/lib/python3.10/dist-packages
(from keras-tuner) (3.4.1)
Requirement already satisfied: packaging in /usr/local/lib/python3.10/dist-
packages (from keras-tuner) (24.1)
Requirement already satisfied: requests in /usr/local/lib/python3.10/dist-
packages (from keras-tuner) (2.32.3)
Requirement already satisfied: kt-legacy in /usr/local/lib/python3.10/dist-
packages (from keras-tuner) (1.0.5)
Requirement already satisfied: absl-py in /usr/local/lib/python3.10/dist-
packages (from keras->keras-tuner) (1.4.0)
Requirement already satisfied: numpy in /usr/local/lib/python3.10/dist-packages
(from keras->keras-tuner) (1.26.4)
Requirement already satisfied: rich in /usr/local/lib/python3.10/dist-packages
(from keras->keras-tuner) (13.8.1)
Requirement already satisfied: namex in /usr/local/lib/python3.10/dist-packages
(from keras->keras-tuner) (0.0.8)
Requirement already satisfied: h5py in /usr/local/lib/python3.10/dist-packages
(from keras->keras-tuner) (3.11.0)
Requirement already satisfied: optree in /usr/local/lib/python3.10/dist-packages
(from keras->keras-tuner) (0.12.1)
Requirement already satisfied: ml-dtypes in /usr/local/lib/python3.10/dist-
packages (from keras->keras-tuner) (0.4.0)
Requirement already satisfied: charset-normalizer<4,>=2 in
/usr/local/lib/python3.10/dist-packages (from requests->keras-tuner) (3.3.2)
Requirement already satisfied: idna<4,>=2.5 in /usr/local/lib/python3.10/dist-
packages (from requests->keras-tuner) (3.8)
Requirement already satisfied: urllib3<3,>=1.21.1 in
/usr/local/lib/python3.10/dist-packages (from requests->keras-tuner) (2.0.7)
Requirement already satisfied: certifi>=2017.4.17 in
/usr/local/lib/python3.10/dist-packages (from requests->keras-tuner) (2024.8.30)
Requirement already satisfied: typing-extensions>=4.5.0 in
/usr/local/lib/python3.10/dist-packages (from optree->keras->keras-tuner)
(4.12.2)
Requirement already satisfied: markdown-it-py>=2.2.0 in
/usr/local/lib/python3.10/dist-packages (from rich->keras->keras-tuner) (3.0.0)
Requirement already satisfied: pygments<3.0.0,>=2.13.0 in
```

```
/usr/local/lib/python3.10/dist-packages (from rich->keras->keras-tuner) (2.16.1)
Requirement already satisfied: mdurl~=0.1 in /usr/local/lib/python3.10/dist-
packages (from markdown-it-py>=2.2.0->rich->keras->keras-tuner) (0.1.2)
```

[ ]:
```
!pip install scikeras
```

```
Collecting scikeras
  Downloading scikeras-0.13.0-py3-none-any.whl.metadata (3.1 kB)
Requirement already satisfied: keras>=3.2.0 in /usr/local/lib/python3.10/dist-
packages (from scikeras) (3.4.1)
Collecting scikit-learn>=1.4.2 (from scikeras)
  Downloading scikit_learn-1.5.2-cp310-cp310-manylinux_2_17_x86_64.manylinux2014
_x86_64.whl.metadata (13 kB)
Requirement already satisfied: absl-py in /usr/local/lib/python3.10/dist-
packages (from keras>=3.2.0->scikeras) (1.4.0)
Requirement already satisfied: numpy in /usr/local/lib/python3.10/dist-packages
(from keras>=3.2.0->scikeras) (1.26.4)
Requirement already satisfied: rich in /usr/local/lib/python3.10/dist-packages
(from keras>=3.2.0->scikeras) (13.8.1)
Requirement already satisfied: namex in /usr/local/lib/python3.10/dist-packages
(from keras>=3.2.0->scikeras) (0.0.8)
Requirement already satisfied: h5py in /usr/local/lib/python3.10/dist-packages
(from keras>=3.2.0->scikeras) (3.11.0)
Requirement already satisfied: optree in /usr/local/lib/python3.10/dist-packages
(from keras>=3.2.0->scikeras) (0.12.1)
Requirement already satisfied: ml-dtypes in /usr/local/lib/python3.10/dist-
packages (from keras>=3.2.0->scikeras) (0.4.0)
Requirement already satisfied: packaging in /usr/local/lib/python3.10/dist-
packages (from keras>=3.2.0->scikeras) (24.1)
Requirement already satisfied: scipy>=1.6.0 in /usr/local/lib/python3.10/dist-
packages (from scikit-learn>=1.4.2->scikeras) (1.13.1)
Requirement already satisfied: joblib>=1.2.0 in /usr/local/lib/python3.10/dist-
packages (from scikit-learn>=1.4.2->scikeras) (1.4.2)
Requirement already satisfied: threadpoolctl>=3.1.0 in
/usr/local/lib/python3.10/dist-packages (from scikit-learn>=1.4.2->scikeras)
(3.5.0)
Requirement already satisfied: typing-extensions>=4.5.0 in
/usr/local/lib/python3.10/dist-packages (from optree->keras>=3.2.0->scikeras)
(4.12.2)
Requirement already satisfied: markdown-it-py>=2.2.0 in
/usr/local/lib/python3.10/dist-packages (from rich->keras>=3.2.0->scikeras)
(3.0.0)
Requirement already satisfied: pygments<3.0.0,>=2.13.0 in
/usr/local/lib/python3.10/dist-packages (from rich->keras>=3.2.0->scikeras)
(2.16.1)
Requirement already satisfied: mdurl~=0.1 in /usr/local/lib/python3.10/dist-
packages (from markdown-it-py>=2.2.0->rich->keras>=3.2.0->scikeras) (0.1.2)
Downloading scikeras-0.13.0-py3-none-any.whl (26 kB)
```

```
Downloading
scikit_learn-1.5.2-cp310-cp310-manylinux_2_17_x86_64.manylinux2014_x86_64.whl
(13.3 MB)
                         13.3/13.3 MB
71.9 MB/s eta 0:00:00
Installing collected packages: scikit-learn, scikeras
  Attempting uninstall: scikit-learn
    Found existing installation: scikit-learn 1.3.2
    Uninstalling scikit-learn-1.3.2:
      Successfully uninstalled scikit-learn-1.3.2
Successfully installed scikeras-0.13.0 scikit-learn-1.5.2
```

[ ]:
```
!pip install tensorflow keras
```

```
Requirement already satisfied: tensorflow in /usr/local/lib/python3.10/dist-
packages (2.17.0)
Requirement already satisfied: keras in /usr/local/lib/python3.10/dist-packages
(3.4.1)
Requirement already satisfied: absl-py>=1.0.0 in /usr/local/lib/python3.10/dist-
packages (from tensorflow) (1.4.0)
Requirement already satisfied: astunparse>=1.6.0 in
/usr/local/lib/python3.10/dist-packages (from tensorflow) (1.6.3)
Requirement already satisfied: flatbuffers>=24.3.25 in
/usr/local/lib/python3.10/dist-packages (from tensorflow) (24.3.25)
Requirement already satisfied: gast!=0.5.0,!=0.5.1,!=0.5.2,>=0.2.1 in
/usr/local/lib/python3.10/dist-packages (from tensorflow) (0.6.0)
Requirement already satisfied: google-pasta>=0.1.1 in
/usr/local/lib/python3.10/dist-packages (from tensorflow) (0.2.0)
Requirement already satisfied: h5py>=3.10.0 in /usr/local/lib/python3.10/dist-
packages (from tensorflow) (3.11.0)
Requirement already satisfied: libclang>=13.0.0 in
/usr/local/lib/python3.10/dist-packages (from tensorflow) (18.1.1)
Requirement already satisfied: ml-dtypes<0.5.0,>=0.3.1 in
/usr/local/lib/python3.10/dist-packages (from tensorflow) (0.4.0)
Requirement already satisfied: opt-einsum>=2.3.2 in
/usr/local/lib/python3.10/dist-packages (from tensorflow) (3.3.0)
Requirement already satisfied: packaging in /usr/local/lib/python3.10/dist-
packages (from tensorflow) (24.1)
Requirement already satisfied:
protobuf!=4.21.0,!=4.21.1,!=4.21.2,!=4.21.3,!=4.21.4,!=4.21.5,<5.0.0dev,>=3.20.3
in /usr/local/lib/python3.10/dist-packages (from tensorflow) (3.20.3)
Requirement already satisfied: requests<3,>=2.21.0 in
/usr/local/lib/python3.10/dist-packages (from tensorflow) (2.32.3)
Requirement already satisfied: setuptools in /usr/local/lib/python3.10/dist-
packages (from tensorflow) (71.0.4)
Requirement already satisfied: six>=1.12.0 in /usr/local/lib/python3.10/dist-
packages (from tensorflow) (1.16.0)
Requirement already satisfied: termcolor>=1.1.0 in
```

```
/usr/local/lib/python3.10/dist-packages (from tensorflow) (2.4.0)
Requirement already satisfied: typing-extensions>=3.6.6 in
/usr/local/lib/python3.10/dist-packages (from tensorflow) (4.12.2)
Requirement already satisfied: wrapt>=1.11.0 in /usr/local/lib/python3.10/dist-
packages (from tensorflow) (1.16.0)
Requirement already satisfied: grpcio<2.0,>=1.24.3 in
/usr/local/lib/python3.10/dist-packages (from tensorflow) (1.64.1)
Requirement already satisfied: tensorboard<2.18,>=2.17 in
/usr/local/lib/python3.10/dist-packages (from tensorflow) (2.17.0)
Requirement already satisfied: tensorflow-io-gcs-filesystem>=0.23.1 in
/usr/local/lib/python3.10/dist-packages (from tensorflow) (0.37.1)
Requirement already satisfied: numpy<2.0.0,>=1.23.5 in
/usr/local/lib/python3.10/dist-packages (from tensorflow) (1.26.4)
Requirement already satisfied: rich in /usr/local/lib/python3.10/dist-packages
(from keras) (13.8.1)
Requirement already satisfied: namex in /usr/local/lib/python3.10/dist-packages
(from keras) (0.0.8)
Requirement already satisfied: optree in /usr/local/lib/python3.10/dist-packages
(from keras) (0.12.1)
Requirement already satisfied: wheel<1.0,>=0.23.0 in
/usr/local/lib/python3.10/dist-packages (from astunparse>=1.6.0->tensorflow)
(0.44.0)
Requirement already satisfied: charset-normalizer<4,>=2 in
/usr/local/lib/python3.10/dist-packages (from requests<3,>=2.21.0->tensorflow)
(3.3.2)
Requirement already satisfied: idna<4,>=2.5 in /usr/local/lib/python3.10/dist-
packages (from requests<3,>=2.21.0->tensorflow) (3.8)
Requirement already satisfied: urllib3<3,>=1.21.1 in
/usr/local/lib/python3.10/dist-packages (from requests<3,>=2.21.0->tensorflow)
(2.0.7)
Requirement already satisfied: certifi>=2017.4.17 in
/usr/local/lib/python3.10/dist-packages (from requests<3,>=2.21.0->tensorflow)
(2024.8.30)
Requirement already satisfied: markdown>=2.6.8 in
/usr/local/lib/python3.10/dist-packages (from
tensorboard<2.18,>=2.17->tensorflow) (3.7)
Requirement already satisfied: tensorboard-data-server<0.8.0,>=0.7.0 in
/usr/local/lib/python3.10/dist-packages (from
tensorboard<2.18,>=2.17->tensorflow) (0.7.2)
Requirement already satisfied: werkzeug>=1.0.1 in
/usr/local/lib/python3.10/dist-packages (from
tensorboard<2.18,>=2.17->tensorflow) (3.0.4)
Requirement already satisfied: markdown-it-py>=2.2.0 in
/usr/local/lib/python3.10/dist-packages (from rich->keras) (3.0.0)
Requirement already satisfied: pygments<3.0.0,>=2.13.0 in
/usr/local/lib/python3.10/dist-packages (from rich->keras) (2.16.1)
Requirement already satisfied: mdurl~=0.1 in /usr/local/lib/python3.10/dist-
packages (from markdown-it-py>=2.2.0->rich->keras) (0.1.2)
```

```
Requirement already satisfied: MarkupSafe>=2.1.1 in
/usr/local/lib/python3.10/dist-packages (from
werkzeug>=1.0.1->tensorboard<2.18,>=2.17->tensorflow) (2.1.5)
```

[ ]:

```python
# STACKING
import numpy as np
import pandas as pd
from sklearn.ensemble import RandomForestClassifier
from sklearn.model_selection import train_test_split
from sklearn.metrics import accuracy_score, f1_score
from tensorflow.keras.models import Sequential
from tensorflow.keras.layers import Dense, Dropout, Conv1D, LSTM, Flatten, Input
import torch
import torch.nn.functional as F
from torch_geometric.nn import GCNConv, global_mean_pool
from torch_geometric.data import Data, DataLoader
from sklearn.base import BaseEstimator, ClassifierMixin
from scikeras.wrappers import KerasClassifier

# Prepare the base models (CNN, LSTM, GCN)
def build_cnn_model():
    model = Sequential()
    model.add(Input(shape=(X_train.shape[1], 1)))  # Specify input shape as the␣
 ↪first layer
    model.add(Conv1D(filters=64, kernel_size=3, activation='relu'))
    model.add(Flatten())
    model.add(Dense(128, activation='relu'))
    model.add(Dense(y_train.shape[1], activation='sigmoid'))
    model.compile(optimizer='adam', loss='binary_crossentropy',␣
 ↪metrics=['binary_accuracy'])
    return model

def build_lstm_model():
    model = Sequential()
    model.add(Input(shape=(X_train.shape[1], 1)))  # Specify input shape as the␣
 ↪first layer
    model.add(LSTM(128))
    model.add(Flatten())
    model.add(Dense(128, activation='relu'))
    model.add(Dense(y_train.shape[1], activation='sigmoid'))
    model.compile(optimizer='adam', loss='binary_crossentropy',␣
 ↪metrics=['binary_accuracy'])
    return model

# Define GCN model
class GCN(torch.nn.Module):
    def __init__(self, num_node_features, num_classes):
```

```python
        super(GCN, self).__init__()
        self.conv1 = GCNConv(num_node_features, 128)
        self.conv2 = GCNConv(128, 64)
        self.fc = torch.nn.Linear(64, num_classes)

    def forward(self, data):
        x, edge_index = data.x, data.edge_index
        x = self.conv1(x, edge_index)
        x = F.relu(x)
        x = self.conv2(x, edge_index)
        x = F.relu(x)
        x = global_mean_pool(x, data.batch)
        return torch.sigmoid(self.fc(x))

# Convert NumPy arrays to torch_geometric Data objects
def numpy_to_geometric(X, y):
    data_list = []
    for i in range(X.shape[0]):
        x_tensor = torch.tensor(X[i], dtype=torch.float).view(-1, 1)  # Shape: (1024, 1)
        edge_index = torch.tensor([[0], [0]], dtype=torch.long)  # Dummy edge_index, modify if you have real edges
        data = Data(x=x_tensor, edge_index=edge_index, y=torch.tensor(y[i], dtype=torch.float).view(1, -1))
        data_list.append(data)
    return data_list

# Train the GCN model
def train_gnn_model(model, train_loader, optimizer, criterion, num_epochs=50):
    model.train()
    for epoch in range(num_epochs):
        total_loss = 0
        for data in train_loader:
            optimizer.zero_grad()
            output = model(data)
            loss = criterion(output, data.y)
            loss.backward()
            optimizer.step()
            total_loss += loss.item()
        print(f'Epoch {epoch + 1}/{num_epochs}, Loss: {total_loss / len(train_loader)}')

# Wrapping GCN in scikit-learn estimator
class GCNClassifier(BaseEstimator, ClassifierMixin):
    def __init__(self, epochs=50, lr=0.001):
        self.model = GCN(num_node_features=1, num_classes=y_train.shape[1])
        self.epochs = epochs
```

```
        self.lr = lr

    def fit(self, X, y):
        optimizer = torch.optim.Adam(self.model.parameters(), lr=self.lr)
        criterion = torch.nn.BCELoss()
        # Convert your data into torch_geometric DataLoader for GCN
        train_data_list = numpy_to_geometric(X, y)
        train_loader = DataLoader(train_data_list, batch_size=32, shuffle=True)
        train_gnn_model(self.model, train_loader, optimizer, criterion, self.
 ↪epochs)
        return self

    def predict(self, X):
        test_data_list = numpy_to_geometric(X, np.zeros((X.shape[0], y_train.
 ↪shape[1])))  # Dummy labels
        test_loader = DataLoader(test_data_list, batch_size=32, shuffle=False)
        self.model.eval()
        preds = []
        with torch.no_grad():
            for data in test_loader:
                output = self.model(data)
                preds.append((output > 0.5).float().cpu().numpy())
        return np.vstack(preds)

# Train CNN, LSTM, and GCN separately
cnn_clf = KerasClassifier(model=build_cnn_model, epochs=10, batch_size=64,␣
 ↪verbose=0)
lstm_clf = KerasClassifier(model=build_lstm_model, epochs=10, batch_size=64,␣
 ↪verbose=0)
gcn_clf = GCNClassifier()

# Convert data for CNN and LSTM models
X_train_cnn = X_train.reshape((X_train.shape[0], X_train.shape[1], 1))
X_test_cnn = X_test.reshape((X_test.shape[0], X_test.shape[1], 1))

# Train the models separately
cnn_clf.fit(X_train_cnn, y_train)
lstm_clf.fit(X_train_cnn, y_train)
gcn_clf.fit(X_train, y_train)

# Make predictions
y_pred_cnn = cnn_clf.predict(X_test_cnn)
y_pred_lstm = lstm_clf.predict(X_test_cnn)
y_pred_gcn = gcn_clf.predict(X_test)

# Combine predictions (e.g., majority voting or averaging)
y_pred_combined = (y_pred_cnn + y_pred_lstm + y_pred_gcn) / 3
```

```python
y_pred_combined = (y_pred_combined > 0.5).astype(int)

# Evaluate the combined predictions
binary_acc = np.mean(np.equal(y_test, y_pred_combined).astype(int))
normal_acc = accuracy_score(y_test, y_pred_combined)
f1 = f1_score(y_test, y_pred_combined, average="micro")

print(f'Combined Binary Accuracy: {binary_acc:.4f}')
print(f'Combined Normal Accuracy: {normal_acc:.4f}')
print(f'Combined F1 Score: {f1:.4f}')
```

```
/usr/local/lib/python3.10/dist-packages/torch_geometric/deprecation.py:26:
UserWarning: 'data.DataLoader' is deprecated, use 'loader.DataLoader' instead
  warnings.warn(out)
```

```
Epoch 1/50, Loss: 0.6739373960915733
Epoch 2/50, Loss: 0.516200849238564
Epoch 3/50, Loss: 0.29819218332276626
Epoch 4/50, Loss: 0.26257388600531745
Epoch 5/50, Loss: 0.2589092342292561
Epoch 6/50, Loss: 0.2570877881611095
Epoch 7/50, Loss: 0.25859983002438264
Epoch 8/50, Loss: 0.2591512071735719
Epoch 9/50, Loss: 0.2583932727575302
Epoch 10/50, Loss: 0.257055439054966
Epoch 11/50, Loss: 0.25774850082748074
Epoch 12/50, Loss: 0.26023025153314366
Epoch 13/50, Loss: 0.2565988126923056
Epoch 14/50, Loss: 0.25922684853567796
Epoch 15/50, Loss: 0.25824036624501734
Epoch 16/50, Loss: 0.25621501268709407
Epoch 17/50, Loss: 0.2580552031012142
Epoch 18/50, Loss: 0.2586713170304018
Epoch 19/50, Loss: 0.2573778484674061
Epoch 20/50, Loss: 0.2599196052726577
Epoch 21/50, Loss: 0.25765814193907904
Epoch 22/50, Loss: 0.25725574002546425
Epoch 23/50, Loss: 0.2587461142855532
Epoch 24/50, Loss: 0.25984319474767237
Epoch 25/50, Loss: 0.2600095399162349
Epoch 26/50, Loss: 0.25921396122259255
Epoch 27/50, Loss: 0.25611692228738
Epoch 28/50, Loss: 0.25813260586822734
Epoch 29/50, Loss: 0.2632749510162017
Epoch 30/50, Loss: 0.25847487546065273
Epoch 31/50, Loss: 0.25935643692226973
Epoch 32/50, Loss: 0.2573310049141155
Epoch 33/50, Loss: 0.2561477887279847
```

```
Epoch 34/50, Loss: 0.25697218232295094
Epoch 35/50, Loss: 0.25843360844780416
Epoch 36/50, Loss: 0.2573666099239798
Epoch 37/50, Loss: 0.2588011407676865
Epoch 38/50, Loss: 0.2588825935826582
Epoch 39/50, Loss: 0.2586155446136699
Epoch 40/50, Loss: 0.25716781484730106
Epoch 41/50, Loss: 0.26015467152876015
Epoch 42/50, Loss: 0.25855714783949013
Epoch 43/50, Loss: 0.25645596884629307
Epoch 44/50, Loss: 0.25892195105552673
Epoch 45/50, Loss: 0.25508502169566993
Epoch 46/50, Loss: 0.2577857042060179
Epoch 47/50, Loss: 0.256553759031436
Epoch 48/50, Loss: 0.25614406124633904
Epoch 49/50, Loss: 0.25632842542494044
```

```
WARNING:tensorflow:5 out of the last 11 calls to <function
TensorFlowTrainer.make_predict_function.<locals>.one_step_on_data_distributed at
0x7f3b4bea3880> triggered tf.function retracing. Tracing is expensive and the
excessive number of tracings could be due to (1) creating @tf.function
repeatedly in a loop, (2) passing tensors with different shapes, (3) passing
Python objects instead of tensors. For (1), please define your @tf.function
outside of the loop. For (2), @tf.function has reduce_retracing=True option that
can avoid unnecessary retracing. For (3), please refer to
https://www.tensorflow.org/guide/function#controlling_retracing and
https://www.tensorflow.org/api_docs/python/tf/function for  more details.
```

```
Epoch 50/50, Loss: 0.2565710426253431
```

```
WARNING:tensorflow:5 out of the last 11 calls to <function
TensorFlowTrainer.make_predict_function.<locals>.one_step_on_data_distributed at
0x7f3b4bea3880> triggered tf.function retracing. Tracing is expensive and the
excessive number of tracings could be due to (1) creating @tf.function
repeatedly in a loop, (2) passing tensors with different shapes, (3) passing
Python objects instead of tensors. For (1), please define your @tf.function
outside of the loop. For (2), @tf.function has reduce_retracing=True option that
can avoid unnecessary retracing. For (3), please refer to
https://www.tensorflow.org/guide/function#controlling_retracing and
https://www.tensorflow.org/api_docs/python/tf/function for  more details.
/usr/local/lib/python3.10/dist-packages/torch_geometric/deprecation.py:26:
UserWarning: 'data.DataLoader' is deprecated, use 'loader.DataLoader' instead
  warnings.warn(out)
```

```
Combined Binary Accuracy: 0.9114
Combined Normal Accuracy: 0.2884
Combined F1 Score: 0.9536
```

```
[ ]: #STACKING - CNN,GCN,XGBoost

import numpy as np
import pandas as pd
from sklearn.ensemble import StackingClassifier, RandomForestClassifier
from sklearn.metrics import accuracy_score, f1_score
from sklearn.model_selection import train_test_split
from xgboost import XGBClassifier
from tensorflow.keras.models import Sequential
from tensorflow.keras.layers import Dense, Dropout, Conv1D, Flatten, Input
import torch
import torch.nn.functional as F
from torch_geometric.nn import GCNConv, global_mean_pool
from torch_geometric.data import DataLoader
from sklearn.base import BaseEstimator, ClassifierMixin
from scikeras.wrappers import KerasClassifier

# Prepare the base models (CNN, GCN, XGBoost)
def build_cnn_model():
    model = Sequential()
    model.add(Input(shape=(X_train.shape[1], 1)))  # Specify input shape as the␣
↪first layer
    model.add(Conv1D(filters=64, kernel_size=3, activation='relu'))
    model.add(Flatten())
    model.add(Dense(128, activation='relu'))
    model.add(Dense(y_train.shape[1], activation='sigmoid'))
    model.compile(optimizer='adam', loss='binary_crossentropy',␣
↪metrics=['binary_accuracy'])
    return model

# Define GCN model
class GCN(torch.nn.Module):
    def __init__(self, num_node_features, num_classes):
        super(GCN, self).__init__()
        self.conv1 = GCNConv(num_node_features, 128)
        self.conv2 = GCNConv(128, 64)
        self.fc = torch.nn.Linear(64, num_classes)

    def forward(self, data):
        x, edge_index = data.x, data.edge_index
        x = self.conv1(x, edge_index)
        x = F.relu(x)
        x = self.conv2(x, edge_index)
        x = F.relu(x)
        x = global_mean_pool(x, data.batch)
        return torch.sigmoid(self.fc(x))
```

```python
# Convert NumPy arrays to torch_geometric Data objects
def numpy_to_geometric(X, y):
    data_list = []
    for i in range(X.shape[0]):
        x_tensor = torch.tensor(X[i], dtype=torch.float).view(-1, 1)  # Shape:␣
↪(1024, 1)
        edge_index = torch.tensor([[0], [0]], dtype=torch.long)  # Dummy␣
↪edge_index, modify if you have real edges
        data = Data(x=x_tensor, edge_index=edge_index, y=torch.tensor(y[i],␣
↪dtype=torch.float).view(1, -1))
        data_list.append(data)
    return data_list

# Train the GCN model
def train_gnn_model(model, train_loader, optimizer, criterion, num_epochs=50):
    model.train()
    for epoch in range(num_epochs):
        total_loss = 0
        for data in train_loader:
            optimizer.zero_grad()
            output = model(data)
            loss = criterion(output, data.y)
            loss.backward()
            optimizer.step()
            total_loss += loss.item()
        print(f'Epoch {epoch + 1}/{num_epochs}, Loss: {total_loss /␣
↪len(train_loader)}')

# Wrapping GCN in scikit-learn estimator
class GCNClassifier(BaseEstimator, ClassifierMixin):
    def __init__(self, epochs=50, lr=0.001):
        self.model = GCN(num_node_features=1, num_classes=y_train.shape[1])
        self.epochs = epochs
        self.lr = lr

    def fit(self, X, y):
        optimizer = torch.optim.Adam(self.model.parameters(), lr=self.lr)
        criterion = torch.nn.BCELoss()
        # Convert your data into torch_geometric DataLoader for GCN
        train_data_list = numpy_to_geometric(X, y)
        train_loader = DataLoader(train_data_list, batch_size=32, shuffle=True)
        train_gnn_model(self.model, train_loader, optimizer, criterion, self.
↪epochs)
        return self

    def predict(self, X):
```

```
        test_data_list = numpy_to_geometric(X, np.zeros((X.shape[0], y_train.
↪shape[1])))  # Dummy labels
        test_loader = DataLoader(test_data_list, batch_size=32, shuffle=False)
        self.model.eval()
        preds = []
        with torch.no_grad():
            for data in test_loader:
                output = self.model(data)
                preds.append((output > 0.5).float().cpu().numpy())
        return np.vstack(preds)

# Train CNN, GCN, and XGBoost separately
cnn_clf = KerasClassifier(model=build_cnn_model, epochs=10, batch_size=64,␣
↪verbose=0)
gcn_clf = GCNClassifier()
xgb_clf = XGBClassifier(use_label_encoder=False, eval_metric='mlogloss')

# Convert data for CNN model
X_train_cnn = X_train.reshape((X_train.shape[0], X_train.shape[1], 1))
X_test_cnn = X_test.reshape((X_test.shape[0], X_test.shape[1], 1))

# Train the models separately
cnn_clf.fit(X_train_cnn, y_train)
gcn_clf.fit(X_train, y_train)
xgb_clf.fit(X_train, y_train)

# Make predictions
y_pred_cnn = cnn_clf.predict(X_test_cnn)
y_pred_gcn = gcn_clf.predict(X_test)
y_pred_xgb = xgb_clf.predict(X_test)

# Combine predictions (e.g., majority voting or averaging)
y_pred_combined = (y_pred_cnn + y_pred_gcn + y_pred_xgb) / 3
y_pred_combined = (y_pred_combined > 0.5).astype(int)

# Evaluate the combined predictions
binary_acc = np.mean(np.equal(y_test, y_pred_combined).astype(int))
normal_acc = accuracy_score(y_test, y_pred_combined)
f1 = f1_score(y_test, y_pred_combined, average="micro")

print(f'Combined Binary Accuracy: {binary_acc:.4f}')
print(f'Combined Normal Accuracy: {normal_acc:.4f}')
print(f'Combined F1 Score: {f1:.4f}')
```

```
/usr/local/lib/python3.10/dist-packages/torch_geometric/deprecation.py:26:
UserWarning: 'data.DataLoader' is deprecated, use 'loader.DataLoader' instead
  warnings.warn(out)
```

```
Epoch 1/50, Loss: 0.6835679709911346
Epoch 2/50, Loss: 0.5414968860499999
Epoch 3/50, Loss: 0.3169667238698286
Epoch 4/50, Loss: 0.2672891862252179
Epoch 5/50, Loss: 0.26069994402282376
Epoch 6/50, Loss: 0.2581184133887291
Epoch 7/50, Loss: 0.25941094142549176
Epoch 8/50, Loss: 0.2586912907221738
Epoch 9/50, Loss: 0.257185157169314
Epoch 10/50, Loss: 0.2569063967641662
Epoch 11/50, Loss: 0.25725077487090053
Epoch 12/50, Loss: 0.25880808076437783
Epoch 13/50, Loss: 0.26063894874909344
Epoch 14/50, Loss: 0.2568010337212506
Epoch 15/50, Loss: 0.25858409746604805
Epoch 16/50, Loss: 0.2599362923818476
Epoch 17/50, Loss: 0.2572337945594507
Epoch 18/50, Loss: 0.25850092970273075
Epoch 19/50, Loss: 0.25629755065721627
Epoch 20/50, Loss: 0.2569630557999891
Epoch 21/50, Loss: 0.2587113831849659
Epoch 22/50, Loss: 0.2570759049233268
Epoch 23/50, Loss: 0.25866379895631003
Epoch 24/50, Loss: 0.2569151095607701
Epoch 25/50, Loss: 0.25623830407857895
Epoch 26/50, Loss: 0.2574890895801432
Epoch 27/50, Loss: 0.2557968339499305
Epoch 28/50, Loss: 0.2583071396631353
Epoch 29/50, Loss: 0.25793038703062954
Epoch 30/50, Loss: 0.2565421133356936
Epoch 31/50, Loss: 0.25644948815598206
Epoch 32/50, Loss: 0.25924227649674697
Epoch 33/50, Loss: 0.25696951063240275
Epoch 34/50, Loss: 0.26029423098353777
Epoch 35/50, Loss: 0.25965481339132085
Epoch 36/50, Loss: 0.2574901870068382
Epoch 37/50, Loss: 0.2580518538461012
Epoch 38/50, Loss: 0.2573310483027907
Epoch 39/50, Loss: 0.256336992716088
Epoch 40/50, Loss: 0.25705330030006524
Epoch 41/50, Loss: 0.257974773645401
Epoch 42/50, Loss: 0.25994909028796587
Epoch 43/50, Loss: 0.2576182449565214
Epoch 44/50, Loss: 0.25712910645148335
Epoch 45/50, Loss: 0.2566972208373687
Epoch 46/50, Loss: 0.2556195929646492
Epoch 47/50, Loss: 0.2557796380099128
Epoch 48/50, Loss: 0.25889087556039586
```

```
Epoch 49/50, Loss: 0.25743040880736184
Epoch 50/50, Loss: 0.2568163065349354
```

```
/usr/local/lib/python3.10/dist-packages/xgboost/core.py:158: UserWarning:
[17:38:42] WARNING: /workspace/src/learner.cc:740:
Parameters: { "use_label_encoder" } are not used.

  warnings.warn(smsg, UserWarning)
/usr/local/lib/python3.10/dist-packages/torch_geometric/deprecation.py:26:
UserWarning: 'data.DataLoader' is deprecated, use 'loader.DataLoader' instead
  warnings.warn(out)
```

```
Combined Binary Accuracy: 0.9036
Combined Normal Accuracy: 0.2097
Combined F1 Score: 0.9486
```

[ ]:
```python
# weighted voting stack

import numpy as np
import pandas as pd
from sklearn.metrics import accuracy_score, f1_score
from sklearn.model_selection import train_test_split
from xgboost import XGBClassifier
from tensorflow.keras.models import Sequential
from tensorflow.keras.layers import Dense, Dropout, Conv1D, Flatten, Input
import torch
import torch.nn.functional as F
from torch_geometric.nn import GCNConv, global_mean_pool
from torch_geometric.data import DataLoader
from scikeras.wrappers import KerasClassifier

# Prepare the base models (CNN, GCN, XGBoost)
def build_cnn_model():
    model = Sequential()
    model.add(Input(shape=(X_train.shape[1], 1)))  # Specify input shape as the␣
↪first layer
    model.add(Conv1D(filters=64, kernel_size=3, activation='relu'))
    model.add(Flatten())
    model.add(Dense(128, activation='relu'))
    model.add(Dense(y_train.shape[1], activation='sigmoid'))
    model.compile(optimizer='adam', loss='binary_crossentropy',␣
↪metrics=['binary_accuracy'])
    return model

# Define GCN model
class GCN(torch.nn.Module):
    def __init__(self, num_node_features, num_classes):
        super(GCN, self).__init__()
```

```
        self.conv1 = GCNConv(num_node_features, 128)
        self.conv2 = GCNConv(128, 64)
        self.fc = torch.nn.Linear(64, num_classes)

    def forward(self, data):
        x, edge_index = data.x, data.edge_index
        x = self.conv1(x, edge_index)
        x = F.relu(x)
        x = self.conv2(x, edge_index)
        x = F.relu(x)
        x = global_mean_pool(x, data.batch)
        return torch.sigmoid(self.fc(x))

# Convert NumPy arrays to torch_geometric Data objects
def numpy_to_geometric(X, y):
    data_list = []
    for i in range(X.shape[0]):
        x_tensor = torch.tensor(X[i], dtype=torch.float).view(-1, 1)  # Shape: (1024, 1)
        edge_index = torch.tensor([[0], [0]], dtype=torch.long)  # Dummy edge_index
        data = Data(x=x_tensor, edge_index=edge_index, y=torch.tensor(y[i], dtype=torch.float).view(1, -1))
        data_list.append(data)
    return data_list

# Train the GCN model
def train_gnn_model(model, train_loader, optimizer, criterion, num_epochs=50):
    model.train()
    for epoch in range(num_epochs):
        total_loss = 0
        for data in train_loader:
            optimizer.zero_grad()
            output = model(data)
            loss = criterion(output, data.y)
            loss.backward()
            optimizer.step()
            total_loss += loss.item()
        print(f'Epoch {epoch + 1}/{num_epochs}, Loss: {total_loss / len(train_loader)}')

# Wrapping GCN in scikit-learn estimator
class GCNClassifier(BaseEstimator, ClassifierMixin):
    def __init__(self, epochs=50, lr=0.001):
        self.model = GCN(num_node_features=1, num_classes=y_train.shape[1])
        self.epochs = epochs
        self.lr = lr
```

```
    def fit(self, X, y):
        optimizer = torch.optim.Adam(self.model.parameters(), lr=self.lr)
        criterion = torch.nn.BCELoss()
        train_data_list = numpy_to_geometric(X, y)
        train_loader = DataLoader(train_data_list, batch_size=32, shuffle=True)
        train_gnn_model(self.model, train_loader, optimizer, criterion, self.
↪epochs)
        return self

    def predict(self, X):
        test_data_list = numpy_to_geometric(X, np.zeros((X.shape[0], y_train.
↪shape[1])))  # Dummy labels
        test_loader = DataLoader(test_data_list, batch_size=32, shuffle=False)
        self.model.eval()
        preds = []
        with torch.no_grad():
            for data in test_loader:
                output = self.model(data)
                preds.append((output > 0.5).float().cpu().numpy())
        return np.vstack(preds)

# CNN and GCN classifiers
cnn_clf = KerasClassifier(model=build_cnn_model, epochs=10, batch_size=64,␣
↪verbose=0)
gcn_clf = GCNClassifier()

# XGBoost Classifier
xgb_clf = XGBClassifier(use_label_encoder=False, eval_metric='mlogloss')

# Reshape data for CNN
X_train_cnn = X_train.reshape((X_train.shape[0], X_train.shape[1], 1))
X_test_cnn = X_test.reshape((X_test.shape[0], X_test.shape[1], 1))

# Train the models
cnn_clf.fit(X_train_cnn, y_train)
gcn_clf.fit(X_train, y_train)
xgb_clf.fit(X_train, y_train)

# Get predictions
y_pred_cnn = cnn_clf.predict(X_test_cnn)
y_pred_gcn = gcn_clf.predict(X_test)
y_pred_xgb = xgb_clf.predict(X_test)

# Apply weighted voting (assigning different weights)
weights = [0.3, 0.3, 0.4]  # Adjust based on individual model performance
```

```python
y_pred_combined = (weights[0] * y_pred_cnn + weights[1] * y_pred_gcn +␣
↪weights[2] * y_pred_xgb) / sum(weights)
y_pred_combined = (y_pred_combined > 0.5).astype(int)

# Evaluate the combined predictions
binary_acc = np.mean(np.equal(y_test, y_pred_combined).astype(int))
normal_acc = accuracy_score(y_test, y_pred_combined)
f1 = f1_score(y_test, y_pred_combined, average="micro")

print(f'Weighted Combined Binary Accuracy: {binary_acc:.4f}')
print(f'Weighted Combined Normal Accuracy: {normal_acc:.4f}')
print(f'Weighted Combined F1 Score: {f1:.4f}')
```

```
/usr/local/lib/python3.10/dist-packages/torch_geometric/deprecation.py:26:
UserWarning: 'data.DataLoader' is deprecated, use 'loader.DataLoader' instead
  warnings.warn(out)
```

```
Epoch 1/50, Loss: 0.6648543371873743
Epoch 2/50, Loss: 0.5041698673192192
Epoch 3/50, Loss: 0.30221597149091606
Epoch 4/50, Loss: 0.26395757846972523
Epoch 5/50, Loss: 0.26018740412066965
Epoch 6/50, Loss: 0.2587503137833932
Epoch 7/50, Loss: 0.2594177451203851
Epoch 8/50, Loss: 0.25928236281170564
Epoch 9/50, Loss: 0.2580554130322793
Epoch 10/50, Loss: 0.2577374196227859
Epoch 11/50, Loss: 0.25821089218644533
Epoch 12/50, Loss: 0.25897260974435243
Epoch 13/50, Loss: 0.2611478650394608
Epoch 14/50, Loss: 0.2567102650509161
Epoch 15/50, Loss: 0.25586119425647397
Epoch 16/50, Loss: 0.2565890769748127
Epoch 17/50, Loss: 0.25657804266494866
Epoch 18/50, Loss: 0.2560475630795254
Epoch 19/50, Loss: 0.2575608395478305
Epoch 20/50, Loss: 0.2590412501903141
Epoch 21/50, Loss: 0.2556345554835656
Epoch 22/50, Loss: 0.25624973239267573
Epoch 23/50, Loss: 0.25925029770416375
Epoch 24/50, Loss: 0.25813396055908766
Epoch 25/50, Loss: 0.25686664528706493
Epoch 26/50, Loss: 0.25663844408357844
Epoch 27/50, Loss: 0.2568141940762015
Epoch 28/50, Loss: 0.26106853870784535
Epoch 29/50, Loss: 0.2593110234421842
Epoch 30/50, Loss: 0.25686338237103296
Epoch 31/50, Loss: 0.25578832056592493
```

```
Epoch 32/50, Loss: 0.25763848774573384
Epoch 33/50, Loss: 0.2571669207776294
Epoch 34/50, Loss: 0.25816017739913044
Epoch 35/50, Loss: 0.25845351727569804
Epoch 36/50, Loss: 0.2565111706362051
Epoch 37/50, Loss: 0.2584740409956259
Epoch 38/50, Loss: 0.2559247889062938
Epoch 39/50, Loss: 0.2566489603589563
Epoch 40/50, Loss: 0.25794688086299333
Epoch 41/50, Loss: 0.2594768812551218
Epoch 42/50, Loss: 0.2584151336375405
Epoch 43/50, Loss: 0.25652579437283907
Epoch 44/50, Loss: 0.257324598291341
Epoch 45/50, Loss: 0.25777848941438336
Epoch 46/50, Loss: 0.25773319339050965
Epoch 47/50, Loss: 0.25667060122770424
Epoch 48/50, Loss: 0.257029487806208
Epoch 49/50, Loss: 0.25525499222909703
Epoch 50/50, Loss: 0.25613880069816813
```

```
/usr/local/lib/python3.10/dist-packages/xgboost/core.py:158: UserWarning:
[18:04:10] WARNING: /workspace/src/learner.cc:740:
Parameters: { "use_label_encoder" } are not used.

  warnings.warn(smsg, UserWarning)
/usr/local/lib/python3.10/dist-packages/torch_geometric/deprecation.py:26:
UserWarning: 'data.DataLoader' is deprecated, use 'loader.DataLoader' instead
  warnings.warn(out)
```

```
Weighted Combined Binary Accuracy: 0.9032
Weighted Combined Normal Accuracy: 0.2172
Weighted Combined F1 Score: 0.9485
```

[ ]:
```
!pip install transformers rdkit-pypi torch torchvision
```

```
Requirement already satisfied: transformers in /usr/local/lib/python3.10/dist-
packages (4.44.2)
Requirement already satisfied: rdkit-pypi in /usr/local/lib/python3.10/dist-
packages (2022.9.5)
Requirement already satisfied: torch in /usr/local/lib/python3.10/dist-packages
(2.4.0+cu121)
Requirement already satisfied: torchvision in /usr/local/lib/python3.10/dist-
packages (0.19.0+cu121)
Requirement already satisfied: filelock in /usr/local/lib/python3.10/dist-
packages (from transformers) (3.16.0)
Requirement already satisfied: huggingface-hub<1.0,>=0.23.2 in
/usr/local/lib/python3.10/dist-packages (from transformers) (0.24.6)
Requirement already satisfied: numpy>=1.17 in /usr/local/lib/python3.10/dist-
packages (from transformers) (1.26.4)
```

```
Requirement already satisfied: packaging>=20.0 in
/usr/local/lib/python3.10/dist-packages (from transformers) (24.1)
Requirement already satisfied: pyyaml>=5.1 in /usr/local/lib/python3.10/dist-
packages (from transformers) (6.0.2)
Requirement already satisfied: regex!=2019.12.17 in
/usr/local/lib/python3.10/dist-packages (from transformers) (2024.5.15)
Requirement already satisfied: requests in /usr/local/lib/python3.10/dist-
packages (from transformers) (2.32.3)
Requirement already satisfied: safetensors>=0.4.1 in
/usr/local/lib/python3.10/dist-packages (from transformers) (0.4.5)
Requirement already satisfied: tokenizers<0.20,>=0.19 in
/usr/local/lib/python3.10/dist-packages (from transformers) (0.19.1)
Requirement already satisfied: tqdm>=4.27 in /usr/local/lib/python3.10/dist-
packages (from transformers) (4.66.5)
Requirement already satisfied: Pillow in /usr/local/lib/python3.10/dist-packages
(from rdkit-pypi) (9.4.0)
Requirement already satisfied: typing-extensions>=4.8.0 in
/usr/local/lib/python3.10/dist-packages (from torch) (4.12.2)
Requirement already satisfied: sympy in /usr/local/lib/python3.10/dist-packages
(from torch) (1.13.2)
Requirement already satisfied: networkx in /usr/local/lib/python3.10/dist-
packages (from torch) (3.3)
Requirement already satisfied: jinja2 in /usr/local/lib/python3.10/dist-packages
(from torch) (3.1.4)
Requirement already satisfied: fsspec in /usr/local/lib/python3.10/dist-packages
(from torch) (2024.6.1)
Requirement already satisfied: MarkupSafe>=2.0 in
/usr/local/lib/python3.10/dist-packages (from jinja2->torch) (2.1.5)
Requirement already satisfied: charset-normalizer<4,>=2 in
/usr/local/lib/python3.10/dist-packages (from requests->transformers) (3.3.2)
Requirement already satisfied: idna<4,>=2.5 in /usr/local/lib/python3.10/dist-
packages (from requests->transformers) (3.8)
Requirement already satisfied: urllib3<3,>=1.21.1 in
/usr/local/lib/python3.10/dist-packages (from requests->transformers) (2.0.7)
Requirement already satisfied: certifi>=2017.4.17 in
/usr/local/lib/python3.10/dist-packages (from requests->transformers)
(2024.8.30)
Requirement already satisfied: mpmath<1.4,>=1.1.0 in
/usr/local/lib/python3.10/dist-packages (from sympy->torch) (1.3.0)
```

[ ]:

```python
# molbert

import torch
from transformers import RobertaTokenizer, RobertaModel
from rdkit import Chem
import numpy as np
import pandas as pd
```

```
from sklearn.model_selection import train_test_split
from sklearn.metrics import accuracy_score, f1_score
from sklearn.ensemble import RandomForestClassifier

# Load pre-trained ChemBERTa (RobertaModel)
tokenizer = RobertaTokenizer.from_pretrained("seyonec/ChemBERTa-zinc-base-v1")
model = RobertaModel.from_pretrained("seyonec/ChemBERTa-zinc-base-v1")

# Function to convert SMILES into ChemBERTa embeddings
def smiles_to_chemberta(smiles_list):
    embeddings = []
    for smile in smiles_list:
        inputs = tokenizer(smile, return_tensors="pt", truncation=True,␣
↪padding=True)
        with torch.no_grad():
            outputs = model(**inputs)
        # Take the mean of the last hidden state for each token to get the␣
↪molecular embedding
        mol_embedding = torch.mean(outputs.last_hidden_state, dim=1).squeeze().
↪cpu().numpy()
        embeddings.append(mol_embedding)
    return np.array(embeddings)

# Load your dataset (assuming you have a 'Chemical Compound' column with SMILES␣
↪strings)
adr_df = pd.read_csv('binary adr.csv')

# Extract SMILES
smiles_list = adr_df['Chemical Compound']

# Convert SMILES to ChemBERTa embeddings
X = smiles_to_chemberta(smiles_list)

# Target (Adverse Reaction Labels)
reaction_columns = ['Hepatobiliary disorders', 'Metabolism and nutrition␣
↪disorders', 'Eye disorders',
                    'Musculoskeletal and connective tissue disorders',␣
↪'Gastrointestinal disorders',
                    'Immune system disorders', 'Reproductive system and breast␣
↪disorders',
                    'Neoplasms benign, malignant and unspecified (incl cysts␣
↪and polyps)',
                    'General disorders and administration site conditions',␣
↪'Endocrine disorders',
                    'Surgical and medical procedures', 'Vascular disorders',␣
↪'Blood and lymphatic system disorders',
```

```python
                    'Skin and subcutaneous tissue disorders', 'Congenital,␣
↪familial and genetic disorders',
                    'Infections and infestations', 'Respiratory, thoracic and␣
↪mediastinal disorders',
                    'Psychiatric disorders', 'Renal and urinary disorders',␣
↪'Pregnancy, puerperium and perinatal conditions',
                    'Ear and labyrinth disorders', 'Cardiac disorders',␣
↪'Nervous system disorders',
                    'Injury, poisoning and procedural complications']

y = adr_df[reaction_columns].values

# Split into training and testing data
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2,␣
↪random_state=42)

# Train a classifier using the ChemBERTa embeddings
rf_clf = RandomForestClassifier()
rf_clf.fit(X_train, y_train)

# Predict and evaluate
y_pred = rf_clf.predict(X_test)

# Evaluate
binary_acc = np.mean(np.equal(y_test, y_pred).astype(int))
f1 = f1_score(y_test, y_pred, average="micro")

print(f'ChemBERTa Binary Accuracy: {binary_acc:.4f}')
print(f'ChemBERTa F1 Score: {f1:.4f}')
```

```
/usr/local/lib/python3.10/dist-packages/huggingface_hub/utils/_token.py:89:
UserWarning:
The secret `HF_TOKEN` does not exist in your Colab secrets.
To authenticate with the Hugging Face Hub, create a token in your settings tab
(https://huggingface.co/settings/tokens), set it as secret in your Google Colab
and restart your session.
You will be able to reuse this secret in all of your notebooks.
Please note that authentication is recommended but still optional to access
public models or datasets.
  warnings.warn(

tokenizer_config.json:   0%|          | 0.00/166 [00:00<?, ?B/s]

vocab.json:   0%|          | 0.00/9.43k [00:00<?, ?B/s]

merges.txt:   0%|          | 0.00/3.21k [00:00<?, ?B/s]

special_tokens_map.json:   0%|          | 0.00/150 [00:00<?, ?B/s]
```

```
config.json:   0%|          | 0.00/501 [00:00<?, ?B/s]
```

```
/usr/local/lib/python3.10/dist-
packages/transformers/tokenization_utils_base.py:1601: FutureWarning:
`clean_up_tokenization_spaces` was not set. It will be set to `True` by default.
This behavior will be depracted in transformers v4.45, and will be then set to
`False` by default. For more details check this issue:
https://github.com/huggingface/transformers/issues/31884
  warnings.warn(
```

```
pytorch_model.bin:   0%|          | 0.00/179M [00:00<?, ?B/s]
```

```
ChemBERTa Binary Accuracy: 0.9090
ChemBERTa F1 Score: 0.9519
```

[ ]:
```
# Combine ChemBERTa embeddings with Morgan Fingerprints
from rdkit.Chem import AllChem

# Generate Morgan Fingerprints (1024-bit)
morgan_fingerprints = [AllChem.GetMorganFingerprintAsBitVect(Chem.
 ↪MolFromSmiles(smile), 2, nBits=1024) for smile in smiles_list]

# Convert to NumPy arrays
fingerprint_array = []
for fp in morgan_fingerprints:
    arr = np.zeros((1,))
    DataStructs.ConvertToNumpyArray(fp, arr)
    fingerprint_array.append(arr)
fingerprint_array = np.array(fingerprint_array)

# Combine both feature sets
X_combined = np.hstack((X, fingerprint_array))

# Use the combined features for training
X_train, X_test, y_train, y_test = train_test_split(X_combined, y, test_size=0.
 ↪2, random_state=42)

# Train and evaluate (e.g., with RandomForestClassifier)
rf_clf.fit(X_train, y_train)
y_pred = rf_clf.predict(X_test)
binary_acc = np.mean(np.equal(y_test, y_pred).astype(int))
f1 = f1_score(y_test, y_pred, average="micro")

print(f'Hybrid Binary Accuracy: {binary_acc:.4f}')
print(f'Hybrid F1 Score: {f1:.4f}')
```

```
Hybrid Binary Accuracy: 0.9096
Hybrid F1 Score: 0.9523
```

[ ]:
```
!pip install --upgrade scikit-learn
```

```
Requirement already satisfied: scikit-learn in /usr/local/lib/python3.10/dist-
packages (1.5.2)
Requirement already satisfied: numpy>=1.19.5 in /usr/local/lib/python3.10/dist-
packages (from scikit-learn) (1.26.4)
Requirement already satisfied: scipy>=1.6.0 in /usr/local/lib/python3.10/dist-
packages (from scikit-learn) (1.13.1)
Requirement already satisfied: joblib>=1.2.0 in /usr/local/lib/python3.10/dist-
packages (from scikit-learn) (1.4.2)
Requirement already satisfied: threadpoolctl>=3.1.0 in
/usr/local/lib/python3.10/dist-packages (from scikit-learn) (3.5.0)
```

[ ]:
```
# NEW START
```

[ ]:
```
!pip install rdkit-pypi
!pip install torch torch-geometric transformers scikit-learn tensorflow
```

```
Requirement already satisfied: rdkit-pypi in /usr/local/lib/python3.10/dist-
packages (2022.9.5)
Requirement already satisfied: numpy in /usr/local/lib/python3.10/dist-packages
(from rdkit-pypi) (1.26.4)
Requirement already satisfied: Pillow in /usr/local/lib/python3.10/dist-packages
(from rdkit-pypi) (10.4.0)
Requirement already satisfied: torch in /usr/local/lib/python3.10/dist-packages
(2.4.1+cu121)
Collecting torch-geometric
  Downloading torch_geometric-2.6.0-py3-none-any.whl.metadata (63 kB)
                           63.1/63.1 kB
5.5 MB/s eta 0:00:00
Requirement already satisfied: transformers in
/usr/local/lib/python3.10/dist-packages (4.44.2)
Requirement already satisfied: scikit-learn in /usr/local/lib/python3.10/dist-
packages (1.3.2)
Requirement already satisfied: tensorflow in /usr/local/lib/python3.10/dist-
packages (2.17.0)
Requirement already satisfied: filelock in /usr/local/lib/python3.10/dist-
packages (from torch) (3.16.0)
Requirement already satisfied: typing-extensions>=4.8.0 in
/usr/local/lib/python3.10/dist-packages (from torch) (4.12.2)
Requirement already satisfied: sympy in /usr/local/lib/python3.10/dist-packages
(from torch) (1.13.2)
Requirement already satisfied: networkx in /usr/local/lib/python3.10/dist-
packages (from torch) (3.3)
Requirement already satisfied: jinja2 in /usr/local/lib/python3.10/dist-packages
(from torch) (3.1.4)
Requirement already satisfied: fsspec in /usr/local/lib/python3.10/dist-packages
(from torch) (2024.6.1)
```

```
Requirement already satisfied: aiohttp in /usr/local/lib/python3.10/dist-
packages (from torch-geometric) (3.10.5)
Requirement already satisfied: numpy in /usr/local/lib/python3.10/dist-packages
(from torch-geometric) (1.26.4)
Requirement already satisfied: psutil>=5.8.0 in /usr/local/lib/python3.10/dist-
packages (from torch-geometric) (5.9.5)
Requirement already satisfied: pyparsing in /usr/local/lib/python3.10/dist-
packages (from torch-geometric) (3.1.4)
Requirement already satisfied: requests in /usr/local/lib/python3.10/dist-
packages (from torch-geometric) (2.32.3)
Requirement already satisfied: tqdm in /usr/local/lib/python3.10/dist-packages
(from torch-geometric) (4.66.5)
Requirement already satisfied: huggingface-hub<1.0,>=0.23.2 in
/usr/local/lib/python3.10/dist-packages (from transformers) (0.24.7)
Requirement already satisfied: packaging>=20.0 in
/usr/local/lib/python3.10/dist-packages (from transformers) (24.1)
Requirement already satisfied: pyyaml>=5.1 in /usr/local/lib/python3.10/dist-
packages (from transformers) (6.0.2)
Requirement already satisfied: regex!=2019.12.17 in
/usr/local/lib/python3.10/dist-packages (from transformers) (2024.9.11)
Requirement already satisfied: safetensors>=0.4.1 in
/usr/local/lib/python3.10/dist-packages (from transformers) (0.4.5)
Requirement already satisfied: tokenizers<0.20,>=0.19 in
/usr/local/lib/python3.10/dist-packages (from transformers) (0.19.1)
Requirement already satisfied: scipy>=1.5.0 in /usr/local/lib/python3.10/dist-
packages (from scikit-learn) (1.13.1)
Requirement already satisfied: joblib>=1.1.1 in /usr/local/lib/python3.10/dist-
packages (from scikit-learn) (1.4.2)
Requirement already satisfied: threadpoolctl>=2.0.0 in
/usr/local/lib/python3.10/dist-packages (from scikit-learn) (3.5.0)
Requirement already satisfied: absl-py>=1.0.0 in /usr/local/lib/python3.10/dist-
packages (from tensorflow) (1.4.0)
Requirement already satisfied: astunparse>=1.6.0 in
/usr/local/lib/python3.10/dist-packages (from tensorflow) (1.6.3)
Requirement already satisfied: flatbuffers>=24.3.25 in
/usr/local/lib/python3.10/dist-packages (from tensorflow) (24.3.25)
Requirement already satisfied: gast!=0.5.0,!=0.5.1,!=0.5.2,>=0.2.1 in
/usr/local/lib/python3.10/dist-packages (from tensorflow) (0.6.0)
Requirement already satisfied: google-pasta>=0.1.1 in
/usr/local/lib/python3.10/dist-packages (from tensorflow) (0.2.0)
Requirement already satisfied: h5py>=3.10.0 in /usr/local/lib/python3.10/dist-
packages (from tensorflow) (3.11.0)
Requirement already satisfied: libclang>=13.0.0 in
/usr/local/lib/python3.10/dist-packages (from tensorflow) (18.1.1)
Requirement already satisfied: ml-dtypes<0.5.0,>=0.3.1 in
/usr/local/lib/python3.10/dist-packages (from tensorflow) (0.4.1)
Requirement already satisfied: opt-einsum>=2.3.2 in
/usr/local/lib/python3.10/dist-packages (from tensorflow) (3.3.0)
```

```
Requirement already satisfied:
protobuf!=4.21.0,!=4.21.1,!=4.21.2,!=4.21.3,!=4.21.4,!=4.21.5,<5.0.0dev,>=3.20.3
in /usr/local/lib/python3.10/dist-packages (from tensorflow) (3.20.3)
Requirement already satisfied: setuptools in /usr/local/lib/python3.10/dist-
packages (from tensorflow) (71.0.4)
Requirement already satisfied: six>=1.12.0 in /usr/local/lib/python3.10/dist-
packages (from tensorflow) (1.16.0)
Requirement already satisfied: termcolor>=1.1.0 in
/usr/local/lib/python3.10/dist-packages (from tensorflow) (2.4.0)
Requirement already satisfied: wrapt>=1.11.0 in /usr/local/lib/python3.10/dist-
packages (from tensorflow) (1.16.0)
Requirement already satisfied: grpcio<2.0,>=1.24.3 in
/usr/local/lib/python3.10/dist-packages (from tensorflow) (1.64.1)
Requirement already satisfied: tensorboard<2.18,>=2.17 in
/usr/local/lib/python3.10/dist-packages (from tensorflow) (2.17.0)
Requirement already satisfied: keras>=3.2.0 in /usr/local/lib/python3.10/dist-
packages (from tensorflow) (3.4.1)
Requirement already satisfied: tensorflow-io-gcs-filesystem>=0.23.1 in
/usr/local/lib/python3.10/dist-packages (from tensorflow) (0.37.1)
Requirement already satisfied: wheel<1.0,>=0.23.0 in
/usr/local/lib/python3.10/dist-packages (from astunparse>=1.6.0->tensorflow)
(0.44.0)
Requirement already satisfied: rich in /usr/local/lib/python3.10/dist-packages
(from keras>=3.2.0->tensorflow) (13.8.1)
Requirement already satisfied: namex in /usr/local/lib/python3.10/dist-packages
(from keras>=3.2.0->tensorflow) (0.0.8)
Requirement already satisfied: optree in /usr/local/lib/python3.10/dist-packages
(from keras>=3.2.0->tensorflow) (0.12.1)
Requirement already satisfied: charset-normalizer<4,>=2 in
/usr/local/lib/python3.10/dist-packages (from requests->torch-geometric) (3.3.2)
Requirement already satisfied: idna<4,>=2.5 in /usr/local/lib/python3.10/dist-
packages (from requests->torch-geometric) (3.10)
Requirement already satisfied: urllib3<3,>=1.21.1 in
/usr/local/lib/python3.10/dist-packages (from requests->torch-geometric) (2.0.7)
Requirement already satisfied: certifi>=2017.4.17 in
/usr/local/lib/python3.10/dist-packages (from requests->torch-geometric)
(2024.8.30)
Requirement already satisfied: markdown>=2.6.8 in
/usr/local/lib/python3.10/dist-packages (from
tensorboard<2.18,>=2.17->tensorflow) (3.7)
Requirement already satisfied: tensorboard-data-server<0.8.0,>=0.7.0 in
/usr/local/lib/python3.10/dist-packages (from
tensorboard<2.18,>=2.17->tensorflow) (0.7.2)
Requirement already satisfied: werkzeug>=1.0.1 in
/usr/local/lib/python3.10/dist-packages (from
tensorboard<2.18,>=2.17->tensorflow) (3.0.4)
Requirement already satisfied: aiohappyeyeballs>=2.3.0 in
/usr/local/lib/python3.10/dist-packages (from aiohttp->torch-geometric) (2.4.0)
```

```
Requirement already satisfied: aiosignal>=1.1.2 in
/usr/local/lib/python3.10/dist-packages (from aiohttp->torch-geometric) (1.3.1)
Requirement already satisfied: attrs>=17.3.0 in /usr/local/lib/python3.10/dist-
packages (from aiohttp->torch-geometric) (24.2.0)
Requirement already satisfied: frozenlist>=1.1.1 in
/usr/local/lib/python3.10/dist-packages (from aiohttp->torch-geometric) (1.4.1)
Requirement already satisfied: multidict<7.0,>=4.5 in
/usr/local/lib/python3.10/dist-packages (from aiohttp->torch-geometric) (6.1.0)
Requirement already satisfied: yarl<2.0,>=1.0 in /usr/local/lib/python3.10/dist-
packages (from aiohttp->torch-geometric) (1.11.1)
Requirement already satisfied: async-timeout<5.0,>=4.0 in
/usr/local/lib/python3.10/dist-packages (from aiohttp->torch-geometric) (4.0.3)
Requirement already satisfied: MarkupSafe>=2.0 in
/usr/local/lib/python3.10/dist-packages (from jinja2->torch) (2.1.5)
Requirement already satisfied: mpmath<1.4,>=1.1.0 in
/usr/local/lib/python3.10/dist-packages (from sympy->torch) (1.3.0)
Requirement already satisfied: markdown-it-py>=2.2.0 in
/usr/local/lib/python3.10/dist-packages (from rich->keras>=3.2.0->tensorflow)
(3.0.0)
Requirement already satisfied: pygments<3.0.0,>=2.13.0 in
/usr/local/lib/python3.10/dist-packages (from rich->keras>=3.2.0->tensorflow)
(2.18.0)
Requirement already satisfied: mdurl~=0.1 in /usr/local/lib/python3.10/dist-
packages (from markdown-it-py>=2.2.0->rich->keras>=3.2.0->tensorflow) (0.1.2)
Downloading torch_geometric-2.6.0-py3-none-any.whl (1.1 MB)
                          1.1/1.1 MB
51.4 MB/s eta 0:00:00
Installing collected packages: torch-geometric
Successfully installed torch-geometric-2.6.0
```

[ ]:
```python
# Hybrid Deep Learning Model (GNN + CNN/MLP + LSTM)

import numpy as np
import torch
import torch.nn.functional as F
from torch_geometric.nn import GCNConv, global_mean_pool
from tensorflow.keras.models import Sequential
from tensorflow.keras.layers import Dense, Flatten, Conv1D, LSTM, Dropout
from sklearn.metrics import accuracy_score, f1_score
from sklearn.model_selection import train_test_split

# Assuming X and y are your feature and label datasets
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2,␣
↪random_state=42)

# Convert to tensors for GCN model training
X_train_tensor = torch.tensor(X_train, dtype=torch.float)
```

```
X_test_tensor = torch.tensor(X_test, dtype=torch.float)
y_train_tensor = torch.tensor(y_train, dtype=torch.float)
y_test_tensor = torch.tensor(y_test, dtype=torch.float)

# GCN Model for GNN
class GCN(torch.nn.Module):
    def __init__(self, num_node_features, num_classes):
        super(GCN, self).__init__()
        self.conv1 = GCNConv(num_node_features, 128)
        self.conv2 = GCNConv(128, 64)
        self.fc = torch.nn.Linear(64, num_classes)

    def forward(self, data):
        x, edge_index = data.x, data.edge_index
        x = self.conv1(x, edge_index)
        x = F.relu(x)
        x = self.conv2(x, edge_index)
        x = F.relu(x)
        x = global_mean_pool(x, data.batch)
        return torch.sigmoid(self.fc(x))

# CNN model
def build_cnn_model():
    model = Sequential()
    model.add(Conv1D(filters=64, kernel_size=3, activation='relu',␣
 ↪input_shape=(X_train.shape[1], 1)))
    model.add(Dropout(0.5))
    model.add(Flatten())
    model.add(Dense(128, activation='relu'))
    model.add(Dense(y_train.shape[1], activation='sigmoid'))
    model.compile(optimizer='adam', loss='binary_crossentropy',␣
 ↪metrics=['accuracy'])
    return model

# LSTM model
def build_lstm_model():
    model = Sequential()
    model.add(LSTM(128, input_shape=(X_train.shape[1], 1)))
    model.add(Dropout(0.5))
    model.add(Flatten())
    model.add(Dense(128, activation='relu'))
    model.add(Dense(y_train.shape[1], activation='sigmoid'))
    model.compile(optimizer='adam', loss='binary_crossentropy',␣
 ↪metrics=['accuracy'])
    return model

# Custom binary accuracy calculation
```

```python
def calculate_binary_accuracy(y_true, y_pred):
    return np.mean(np.equal(y_true, y_pred).astype(int))

# Train CNN and LSTM
cnn_model = build_cnn_model()
cnn_model.fit(X_train, y_train, epochs=20, batch_size=64, validation_split=0.2)

lstm_model = build_lstm_model()
lstm_model.fit(X_train, y_train, epochs=20, batch_size=64, validation_split=0.2)

# Assuming GCN is wrapped and trained as before (you will need a proper GCN␣
↪data loader)
# Train and evaluate GCN similarly if required

# Evaluate CNN and LSTM Models
y_pred_cnn = cnn_model.predict(X_test)
y_pred_lstm = lstm_model.predict(X_test)

# Convert predictions to binary labels (0 or 1)
y_pred_cnn_bin = (y_pred_cnn > 0.5).astype(int)
y_pred_lstm_bin = (y_pred_lstm > 0.5).astype(int)

# Binary accuracy calculation
cnn_bin_acc = calculate_binary_accuracy(y_test, y_pred_cnn_bin)
lstm_bin_acc = calculate_binary_accuracy(y_test, y_pred_lstm_bin)

# Normal accuracy and F1 score
cnn_normal_acc = accuracy_score(y_test, y_pred_cnn_bin)
lstm_normal_acc = accuracy_score(y_test, y_pred_lstm_bin)

cnn_f1_score = f1_score(y_test, y_pred_cnn_bin, average='micro')
lstm_f1_score = f1_score(y_test, y_pred_lstm_bin, average='micro')

# Print metrics for CNN
print(f"Binary Accuracy (CNN): {cnn_bin_acc:.4f}")
print(f"Normal Accuracy (CNN): {cnn_normal_acc:.4f}")
print(f"F1 Score (CNN): {cnn_f1_score:.4f}")

# Print metrics for LSTM
print(f"Binary Accuracy (LSTM): {lstm_bin_acc:.4f}")
print(f"Normal Accuracy (LSTM): {lstm_normal_acc:.4f}")
print(f"F1 Score (LSTM): {lstm_f1_score:.4f}")
```

```
Epoch 1/20
```

```
/usr/local/lib/python3.10/dist-
packages/keras/src/layers/convolutional/base_conv.py:107: UserWarning: Do not
pass an `input_shape`/`input_dim` argument to a layer. When using Sequential
```

```
models, prefer using an `Input(shape)` object as the first layer in the model
instead.
  super().__init__(activity_regularizer=activity_regularizer, **kwargs)
```

```
14/14             6s 229ms/step -
accuracy: 0.0048 - loss: 0.4659 - val_accuracy: 0.0000e+00 - val_loss: 0.3061
Epoch 2/20
14/14             0s 8ms/step -
accuracy: 0.0000e+00 - loss: 0.2665 - val_accuracy: 0.0000e+00 - val_loss:
0.2782
Epoch 3/20
14/14             0s 7ms/step -
accuracy: 0.0016 - loss: 0.2326 - val_accuracy: 0.0000e+00 - val_loss: 0.2759
Epoch 4/20
14/14             0s 7ms/step -
accuracy: 6.3803e-04 - loss: 0.2067 - val_accuracy: 0.0000e+00 - val_loss:
0.2723
Epoch 5/20
14/14             0s 7ms/step -
accuracy: 4.1813e-04 - loss: 0.1923 - val_accuracy: 0.0000e+00 - val_loss:
0.2792
Epoch 6/20
14/14             0s 6ms/step -
accuracy: 7.6824e-04 - loss: 0.1660 - val_accuracy: 0.0000e+00 - val_loss:
0.2897
Epoch 7/20
14/14             0s 7ms/step -
accuracy: 4.1813e-04 - loss: 0.1585 - val_accuracy: 0.0000e+00 - val_loss:
0.3066
Epoch 8/20
14/14             0s 7ms/step -
accuracy: 0.0019 - loss: 0.1363 - val_accuracy: 0.0000e+00 - val_loss: 0.3236
Epoch 9/20
14/14             0s 7ms/step -
accuracy: 0.0035 - loss: 0.1222 - val_accuracy: 0.0000e+00 - val_loss: 0.3415
Epoch 10/20
14/14             0s 6ms/step -
accuracy: 0.0013 - loss: 0.1110 - val_accuracy: 0.0000e+00 - val_loss: 0.3620
Epoch 11/20
14/14             0s 7ms/step -
accuracy: 0.0013 - loss: 0.0932 - val_accuracy: 0.0000e+00 - val_loss: 0.3791
Epoch 12/20
14/14             0s 8ms/step -
accuracy: 0.0035 - loss: 0.0842 - val_accuracy: 0.0000e+00 - val_loss: 0.4081
Epoch 13/20
14/14             0s 7ms/step -
accuracy: 0.0019 - loss: 0.0728 - val_accuracy: 0.0000e+00 - val_loss: 0.4304
Epoch 14/20
```

```
14/14            0s 7ms/step -
accuracy: 0.0013 - loss: 0.0635 - val_accuracy: 0.0000e+00 - val_loss: 0.4556
Epoch 15/20
14/14            0s 8ms/step -
accuracy: 2.3662e-04 - loss: 0.0571 - val_accuracy: 0.0141 - val_loss: 0.4844
Epoch 16/20
14/14            0s 9ms/step -
accuracy: 6.3803e-04 - loss: 0.0534 - val_accuracy: 0.0141 - val_loss: 0.5031
Epoch 17/20
14/14            0s 8ms/step -
accuracy: 0.0016 - loss: 0.0505 - val_accuracy: 0.0188 - val_loss: 0.5279
Epoch 18/20
14/14            0s 7ms/step -
accuracy: 0.0036 - loss: 0.0433 - val_accuracy: 0.0188 - val_loss: 0.5477
Epoch 19/20
14/14            0s 7ms/step -
accuracy: 0.0108 - loss: 0.0408 - val_accuracy: 0.0235 - val_loss: 0.5674
Epoch 20/20
14/14            0s 7ms/step -
accuracy: 0.0092 - loss: 0.0397 - val_accuracy: 0.0235 - val_loss: 0.5909
Epoch 1/20
```

```
/usr/local/lib/python3.10/dist-packages/keras/src/layers/rnn/rnn.py:204:
UserWarning: Do not pass an `input_shape`/`input_dim` argument to a layer. When
using Sequential models, prefer using an `Input(shape)` object as the first
layer in the model instead.
  super().__init__(**kwargs)
```

```
14/14            3s 81ms/step -
accuracy: 0.0035 - loss: 0.6689 - val_accuracy: 0.0000e+00 - val_loss: 0.3455
Epoch 2/20
14/14            1s 51ms/step -
accuracy: 0.0000e+00 - loss: 0.3113 - val_accuracy: 0.0000e+00 - val_loss:
0.2627
Epoch 3/20
14/14            1s 52ms/step -
accuracy: 0.0000e+00 - loss: 0.2662 - val_accuracy: 0.0000e+00 - val_loss:
0.2582
Epoch 4/20
14/14            1s 44ms/step -
accuracy: 0.0000e+00 - loss: 0.2607 - val_accuracy: 0.0000e+00 - val_loss:
0.2581
Epoch 5/20
14/14            1s 44ms/step -
accuracy: 0.0000e+00 - loss: 0.2715 - val_accuracy: 0.0000e+00 - val_loss:
0.2566
Epoch 6/20
14/14            1s 45ms/step -
accuracy: 0.0035 - loss: 0.2721 - val_accuracy: 0.0000e+00 - val_loss: 0.2558
```

```
Epoch 7/20
14/14            1s 44ms/step -
accuracy: 5.2229e-04 - loss: 0.2684 - val_accuracy: 0.0000e+00 - val_loss:
0.2563
Epoch 8/20
14/14            1s 44ms/step -
accuracy: 0.0016 - loss: 0.2598 - val_accuracy: 0.0000e+00 - val_loss: 0.2572
Epoch 9/20
14/14            1s 44ms/step -
accuracy: 0.0000e+00 - loss: 0.2654 - val_accuracy: 0.0000e+00 - val_loss:
0.2568
Epoch 10/20
14/14            1s 44ms/step -
accuracy: 4.1813e-04 - loss: 0.2581 - val_accuracy: 0.0000e+00 - val_loss:
0.2570
Epoch 11/20
14/14            1s 44ms/step -
accuracy: 7.6824e-04 - loss: 0.2610 - val_accuracy: 0.0000e+00 - val_loss:
0.2567
Epoch 12/20
14/14            1s 44ms/step -
accuracy: 9.1705e-04 - loss: 0.2674 - val_accuracy: 0.0000e+00 - val_loss:
0.2568
Epoch 13/20
14/14            1s 43ms/step -
accuracy: 0.0000e+00 - loss: 0.2641 - val_accuracy: 0.0000e+00 - val_loss:
0.2573
Epoch 14/20
14/14            1s 43ms/step -
accuracy: 0.0019 - loss: 0.2607 - val_accuracy: 0.0000e+00 - val_loss: 0.2564
Epoch 15/20
14/14            1s 50ms/step -
accuracy: 7.6824e-04 - loss: 0.2536 - val_accuracy: 0.0000e+00 - val_loss:
0.2564
Epoch 16/20
14/14            1s 49ms/step -
accuracy: 0.0035 - loss: 0.2716 - val_accuracy: 0.0000e+00 - val_loss: 0.2566
Epoch 17/20
14/14            1s 52ms/step -
accuracy: 0.0013 - loss: 0.2645 - val_accuracy: 0.0000e+00 - val_loss: 0.2568
Epoch 18/20
14/14            1s 54ms/step -
accuracy: 3.2343e-04 - loss: 0.2488 - val_accuracy: 0.0000e+00 - val_loss:
0.2570
Epoch 19/20
14/14            1s 44ms/step -
accuracy: 6.3803e-04 - loss: 0.2605 - val_accuracy: 0.0000e+00 - val_loss:
0.2569
```

```
Epoch 20/20
14/14            1s 43ms/step -
accuracy: 0.0035 - loss: 0.2657 - val_accuracy: 0.0000e+00 - val_loss: 0.2567
9/9              1s 38ms/step
9/9              0s 22ms/step
Binary Accuracy (CNN): 0.8841
Normal Accuracy (CNN): 0.1199
F1 Score (CNN): 0.9369
Binary Accuracy (LSTM): 0.9114
Normal Accuracy (LSTM): 0.2884
F1 Score (LSTM): 0.9536
```

[ ]:

```
import torch
import torch.nn.functional as F
from tensorflow.keras.models import Sequential
from tensorflow.keras.layers import Dense, Flatten, Conv1D, LSTM
from sklearn.metrics import accuracy_score, f1_score
import numpy as np

# Assuming X_train, X_test, y_train, y_test are defined as your feature and␣
↪label sets.

# Binary Accuracy Calculation
def calculate_binary_accuracy(y_true, y_pred):
    return np.mean(np.equal(y_true, y_pred).astype(int))

# CNN model
def build_cnn_model():
    model = Sequential()
    model.add(Conv1D(filters=64, kernel_size=3, activation='relu',␣
↪input_shape=(X_train.shape[1], 1)))
    model.add(Flatten())
    model.add(Dense(128, activation='relu'))
    model.add(Dense(y_train.shape[1], activation='sigmoid'))
    model.compile(optimizer='adam', loss='binary_crossentropy',␣
↪metrics=['accuracy'])
    return model

# LSTM model
def build_lstm_model():
    model = Sequential()
    model.add(LSTM(128, input_shape=(X_train.shape[1], 1)))
    model.add(Flatten())
    model.add(Dense(128, activation='relu'))
    model.add(Dense(y_train.shape[1], activation='sigmoid'))
    model.compile(optimizer='adam', loss='binary_crossentropy',␣
↪metrics=['accuracy'])
```

```
    return model

# Train CNN model and calculate binary accuracy on both training and test sets
cnn_model = build_cnn_model()

# Train CNN for 10 epochs and calculate binary accuracy on the training set at␣
↪each epoch
for epoch in range(15):
    cnn_model.fit(X_train, y_train, epochs=1, batch_size=64, verbose=1)

    # Predict on training set and calculate binary accuracy
    y_pred_train_cnn = cnn_model.predict(X_train)
    y_pred_train_cnn_bin = (y_pred_train_cnn > 0.5).astype(int)
    train_binary_acc_cnn = calculate_binary_accuracy(y_train,␣
↪y_pred_train_cnn_bin)
    print(f"Training Binary Accuracy (CNN, Epoch {epoch+1}):␣
↪{train_binary_acc_cnn:.4f}")

# Calculate binary accuracy on the test set
y_pred_test_cnn = cnn_model.predict(X_test)
y_pred_test_cnn_bin = (y_pred_test_cnn > 0.5).astype(int)
test_binary_acc_cnn = calculate_binary_accuracy(y_test, y_pred_test_cnn_bin)
print(f"Test Set Binary Accuracy (CNN): {test_binary_acc_cnn:.4f}")

# Train LSTM model and calculate binary accuracy on both training and test sets
lstm_model = build_lstm_model()

# Train LSTM for 10 epochs and calculate binary accuracy on the training set at␣
↪each epoch
for epoch in range(15):
    lstm_model.fit(X_train, y_train, epochs=1, batch_size=64, verbose=1)

    # Predict on training set and calculate binary accuracy
    y_pred_train_lstm = lstm_model.predict(X_train)
    y_pred_train_lstm_bin = (y_pred_train_lstm > 0.5).astype(int)
    train_binary_acc_lstm = calculate_binary_accuracy(y_train,␣
↪y_pred_train_lstm_bin)
    print(f"Training Binary Accuracy (LSTM, Epoch {epoch+1}):␣
↪{train_binary_acc_lstm:.4f}")

# Calculate binary accuracy on the test set
y_pred_test_lstm = lstm_model.predict(X_test)
y_pred_test_lstm_bin = (y_pred_test_lstm > 0.5).astype(int)
test_binary_acc_lstm = calculate_binary_accuracy(y_test, y_pred_test_lstm_bin)
print(f"Test Set Binary Accuracy (LSTM): {test_binary_acc_lstm:.4f}")
```

```python
# Calculate F1 score for CNN and LSTM models
cnn_f1 = f1_score(y_test, y_pred_test_cnn_bin, average='micro')
lstm_f1 = f1_score(y_test, y_pred_test_lstm_bin, average='micro')

# Print F1 scores
print(f"F1 Score (CNN): {cnn_f1:.4f}")
print(f"F1 Score (LSTM): {lstm_f1:.4f}")
```

```
/usr/local/lib/python3.10/dist-
packages/keras/src/layers/convolutional/base_conv.py:107: UserWarning: Do not
pass an `input_shape`/`input_dim` argument to a layer. When using Sequential
models, prefer using an `Input(shape)` object as the first layer in the model
instead.
  super().__init__(activity_regularizer=activity_regularizer, **kwargs)
```

```
17/17                3s 82ms/step -
accuracy: 0.1590 - loss: 0.4489
34/34                1s 14ms/step
Training Binary Accuracy (CNN, Epoch 1): 0.9017
17/17                0s 5ms/step -
accuracy: 0.1472 - loss: 0.2459
34/34                0s 2ms/step
Training Binary Accuracy (CNN, Epoch 2): 0.9172
17/17                0s 5ms/step -
accuracy: 0.0034 - loss: 0.2244
34/34                0s 1ms/step
Training Binary Accuracy (CNN, Epoch 3): 0.9251
17/17                0s 5ms/step -
accuracy: 0.0011 - loss: 0.1959
34/34                0s 1ms/step
Training Binary Accuracy (CNN, Epoch 4): 0.9297
17/17                0s 5ms/step -
accuracy: 0.0011 - loss: 0.1679
34/34                0s 1ms/step
Training Binary Accuracy (CNN, Epoch 5): 0.9353
17/17                0s 4ms/step -
accuracy: 0.0025 - loss: 0.1545
34/34                0s 1ms/step
Training Binary Accuracy (CNN, Epoch 6): 0.9492
17/17                0s 5ms/step -
accuracy: 5.0381e-04 - loss: 0.1336
34/34                0s 1ms/step
Training Binary Accuracy (CNN, Epoch 7): 0.9585
17/17                0s 5ms/step -
accuracy: 0.0031 - loss: 0.1227
34/34                0s 1ms/step
Training Binary Accuracy (CNN, Epoch 8): 0.9623
17/17                0s 5ms/step -
```

```
accuracy: 0.0041 - loss: 0.1033
34/34             0s 1ms/step
Training Binary Accuracy (CNN, Epoch 9): 0.9712
17/17             0s 5ms/step -
accuracy: 0.0018 - loss: 0.0901
34/34             0s 2ms/step
Training Binary Accuracy (CNN, Epoch 10): 0.9740
17/17             0s 5ms/step -
accuracy: 0.0012 - loss: 0.0777
34/34             0s 1ms/step
Training Binary Accuracy (CNN, Epoch 11): 0.9800
17/17             0s 5ms/step -
accuracy: 0.0018 - loss: 0.0667
34/34             0s 2ms/step
Training Binary Accuracy (CNN, Epoch 12): 0.9843
17/17             0s 5ms/step -
accuracy: 0.0034 - loss: 0.0628
34/34             0s 1ms/step
Training Binary Accuracy (CNN, Epoch 13): 0.9865
17/17             0s 5ms/step -
accuracy: 0.0068 - loss: 0.0535
34/34             0s 1ms/step
Training Binary Accuracy (CNN, Epoch 14): 0.9890
17/17             0s 5ms/step -
accuracy: 0.0065 - loss: 0.0480
34/34             0s 1ms/step
Training Binary Accuracy (CNN, Epoch 15): 0.9909
9/9             0s 9ms/step
Test Set Binary Accuracy (CNN): 0.8803

/usr/local/lib/python3.10/dist-packages/keras/src/layers/rnn/rnn.py:204:
UserWarning: Do not pass an `input_shape`/`input_dim` argument to a layer. When
using Sequential models, prefer using an `Input(shape)` object as the first
layer in the model instead.
  super().__init__(**kwargs)

17/17             2s 44ms/step -
accuracy: 0.0122 - loss: 0.6623
34/34             1s 15ms/step
Training Binary Accuracy (LSTM, Epoch 1): 0.8999
17/17             1s 37ms/step -
accuracy: 0.0483 - loss: 0.2942
34/34             0s 11ms/step
Training Binary Accuracy (LSTM, Epoch 2): 0.8999
17/17             1s 37ms/step -
accuracy: 9.1225e-04 - loss: 0.2640
34/34             1s 16ms/step
Training Binary Accuracy (LSTM, Epoch 3): 0.8999
17/17             1s 42ms/step -
```

```
accuracy: 6.7974e-04 - loss: 0.2604
34/34             1s 18ms/step
Training Binary Accuracy (LSTM, Epoch 4): 0.8999
17/17             1s 46ms/step -
accuracy: 1.5858e-04 - loss: 0.2580
34/34             0s 11ms/step
Training Binary Accuracy (LSTM, Epoch 5): 0.8999
17/17             1s 37ms/step -
accuracy: 0.0012 - loss: 0.2677
34/34             0s 11ms/step
Training Binary Accuracy (LSTM, Epoch 6): 0.8999
17/17             1s 37ms/step -
accuracy: 0.0012 - loss: 0.2542
34/34             0s 12ms/step
Training Binary Accuracy (LSTM, Epoch 7): 0.8999
17/17             1s 36ms/step -
accuracy: 6.7974e-04 - loss: 0.2572
34/34             0s 12ms/step
Training Binary Accuracy (LSTM, Epoch 8): 0.8999
17/17             1s 36ms/step -
accuracy: 1.5858e-04 - loss: 0.2516
34/34             0s 11ms/step
Training Binary Accuracy (LSTM, Epoch 9): 0.8999
17/17             1s 37ms/step -
accuracy: 0.0022 - loss: 0.2578
34/34             0s 11ms/step
Training Binary Accuracy (LSTM, Epoch 10): 0.8999
17/17             1s 37ms/step -
accuracy: 7.8825e-04 - loss: 0.2547
34/34             0s 12ms/step
Training Binary Accuracy (LSTM, Epoch 11): 0.8999
17/17             1s 42ms/step -
accuracy: 3.4523e-04 - loss: 0.2572
34/34             1s 16ms/step
Training Binary Accuracy (LSTM, Epoch 12): 0.8999
17/17             1s 46ms/step -
accuracy: 0.0017 - loss: 0.2510
34/34             0s 12ms/step
Training Binary Accuracy (LSTM, Epoch 13): 0.8999
17/17             1s 38ms/step -
accuracy: 0.0011 - loss: 0.2648
34/34             0s 11ms/step
Training Binary Accuracy (LSTM, Epoch 14): 0.8999
17/17             1s 37ms/step -
accuracy: 7.8825e-04 - loss: 0.2549
34/34             0s 11ms/step
Training Binary Accuracy (LSTM, Epoch 15): 0.8999
9/9             0s 11ms/step
```

```
Test Set Binary Accuracy (LSTM): 0.9114
F1 Score (CNN): 0.9348
F1 Score (LSTM): 0.9536
```

[ ]:
```python
# Hybrid Deep Learning Model (GNN + CNN/MLP + LSTM) -  to prevent overfitting␣
↪using dropout for CNN and LTSM

import torch
import torch.nn.functional as F
from tensorflow.keras.models import Sequential
from tensorflow.keras.layers import Dense, Flatten, Conv1D, LSTM, Dropout
from sklearn.metrics import accuracy_score, f1_score
import numpy as np

# Assuming X_train, X_test, y_train, y_test are defined as your feature and␣
↪label sets.

# Binary Accuracy Calculation
def calculate_binary_accuracy(y_true, y_pred):
    return np.mean(np.equal(y_true, y_pred).astype(int))

# CNN model with Dropout
def build_cnn_model_with_dropout():
    model = Sequential()
    model.add(Conv1D(filters=64, kernel_size=3, activation='relu',␣
↪input_shape=(X_train.shape[1], 1)))
    model.add(Dropout(0.5))  # Dropout layer added here
    model.add(Flatten())
    model.add(Dense(128, activation='relu'))
    model.add(Dropout(0.5))  # Dropout layer added here
    model.add(Dense(y_train.shape[1], activation='sigmoid'))
    model.compile(optimizer='adam', loss='binary_crossentropy',␣
↪metrics=['accuracy'])
    return model

# LSTM model with Dropout
def build_lstm_model_with_dropout():
    model = Sequential()
    model.add(LSTM(128, input_shape=(X_train.shape[1], 1)))
    model.add(Dropout(0.5))  # Dropout layer added here
    model.add(Flatten())
    model.add(Dense(128, activation='relu'))
    model.add(Dropout(0.5))  # Dropout layer added here
    model.add(Dense(y_train.shape[1], activation='sigmoid'))
    model.compile(optimizer='adam', loss='binary_crossentropy',␣
↪metrics=['accuracy'])
    return model
```

```
# Train CNN model and calculate binary accuracy on both training and test sets
cnn_model = build_cnn_model_with_dropout()

# Train CNN for 15 epochs and calculate binary accuracy on the training set at␣
↪each epoch
for epoch in range(20):
    cnn_model.fit(X_train, y_train, epochs=1, batch_size=64, verbose=1)

    # Predict on training set and calculate binary accuracy
    y_pred_train_cnn = cnn_model.predict(X_train)
    y_pred_train_cnn_bin = (y_pred_train_cnn > 0.5).astype(int)
    train_binary_acc_cnn = calculate_binary_accuracy(y_train,␣
↪y_pred_train_cnn_bin)
    print(f"Training Binary Accuracy (CNN, Epoch {epoch+1}):␣
↪{train_binary_acc_cnn:.4f}")

# Calculate binary accuracy on the test set
y_pred_test_cnn = cnn_model.predict(X_test)
y_pred_test_cnn_bin = (y_pred_test_cnn > 0.5).astype(int)
test_binary_acc_cnn = calculate_binary_accuracy(y_test, y_pred_test_cnn_bin)
print(f"Test Set Binary Accuracy (CNN): {test_binary_acc_cnn:.4f}")

# Train LSTM model and calculate binary accuracy on both training and test sets
lstm_model = build_lstm_model_with_dropout()

# Train LSTM for 15 epochs and calculate binary accuracy on the training set at␣
↪each epoch
for epoch in range(15):
    lstm_model.fit(X_train, y_train, epochs=1, batch_size=64, verbose=1)

    # Predict on training set and calculate binary accuracy
    y_pred_train_lstm = lstm_model.predict(X_train)
    y_pred_train_lstm_bin = (y_pred_train_lstm > 0.5).astype(int)
    train_binary_acc_lstm = calculate_binary_accuracy(y_train,␣
↪y_pred_train_lstm_bin)
    print(f"Training Binary Accuracy (LSTM, Epoch {epoch+1}):␣
↪{train_binary_acc_lstm:.4f}")

# Calculate binary accuracy on the test set
y_pred_test_lstm = lstm_model.predict(X_test)
y_pred_test_lstm_bin = (y_pred_test_lstm > 0.5).astype(int)
test_binary_acc_lstm = calculate_binary_accuracy(y_test, y_pred_test_lstm_bin)
print(f"Test Set Binary Accuracy (LSTM): {test_binary_acc_lstm:.4f}")

# Calculate F1 score for CNN and LSTM models
```

```python
cnn_f1 = f1_score(y_test, y_pred_test_cnn_bin, average='micro')
lstm_f1 = f1_score(y_test, y_pred_test_lstm_bin, average='micro')

# Print F1 scores
print(f"F1 Score (CNN): {cnn_f1:.4f}")
print(f"F1 Score (LSTM): {lstm_f1:.4f}")
```

```
/usr/local/lib/python3.10/dist-
packages/keras/src/layers/convolutional/base_conv.py:107: UserWarning: Do not
pass an `input_shape`/`input_dim` argument to a layer. When using Sequential
models, prefer using an `Input(shape)` object as the first layer in the model
instead.
  super().__init__(activity_regularizer=activity_regularizer, **kwargs)
```

```
17/17             14s 216ms/step -
accuracy: 0.0972 - loss: 0.5176
34/34             1s 17ms/step
Training Binary Accuracy (CNN, Epoch 1): 0.9001
17/17             0s 6ms/step -
accuracy: 0.0184 - loss: 0.3431
34/34             0s 2ms/step
Training Binary Accuracy (CNN, Epoch 2): 0.9009
17/17             0s 6ms/step -
accuracy: 0.0103 - loss: 0.3008
34/34             0s 2ms/step
Training Binary Accuracy (CNN, Epoch 3): 0.9052
17/17             0s 7ms/step -
accuracy: 0.0025 - loss: 0.2624
34/34             0s 3ms/step
Training Binary Accuracy (CNN, Epoch 4): 0.9088
17/17             0s 12ms/step -
accuracy: 0.0034 - loss: 0.2605
34/34             0s 4ms/step
Training Binary Accuracy (CNN, Epoch 5): 0.9118
17/17             0s 9ms/step -
accuracy: 0.0080 - loss: 0.2326
34/34             0s 2ms/step
Training Binary Accuracy (CNN, Epoch 6): 0.9179
17/17             0s 13ms/step -
accuracy: 0.0025 - loss: 0.2242
34/34             0s 2ms/step
Training Binary Accuracy (CNN, Epoch 7): 0.9228
17/17             0s 6ms/step -
accuracy: 0.0071 - loss: 0.2091
34/34             0s 2ms/step
Training Binary Accuracy (CNN, Epoch 8): 0.9248
17/17             0s 7ms/step -
accuracy: 0.0037 - loss: 0.2075
```

```
34/34              0s 2ms/step
Training Binary Accuracy (CNN, Epoch 9): 0.9268
17/17              0s 7ms/step -
accuracy: 7.8825e-04 - loss: 0.1818
34/34              0s 2ms/step
Training Binary Accuracy (CNN, Epoch 10): 0.9289
17/17              0s 8ms/step -
accuracy: 8.3514e-04 - loss: 0.1842
34/34              0s 2ms/step
Training Binary Accuracy (CNN, Epoch 11): 0.9286
17/17              0s 7ms/step -
accuracy: 0.0012 - loss: 0.1698
34/34              0s 2ms/step
Training Binary Accuracy (CNN, Epoch 12): 0.9392
17/17              0s 6ms/step -
accuracy: 4.9648e-04 - loss: 0.1638
34/34              0s 1ms/step
Training Binary Accuracy (CNN, Epoch 13): 0.9459
17/17              0s 6ms/step -
accuracy: 0.0036 - loss: 0.1625
34/34              0s 1ms/step
Training Binary Accuracy (CNN, Epoch 14): 0.9482
17/17              0s 6ms/step -
accuracy: 0.0013 - loss: 0.1499
34/34              0s 1ms/step
Training Binary Accuracy (CNN, Epoch 15): 0.9502
17/17              0s 6ms/step -
accuracy: 0.0083 - loss: 0.1509
34/34              0s 1ms/step
Training Binary Accuracy (CNN, Epoch 16): 0.9571
17/17              0s 6ms/step -
accuracy: 5.8329e-04 - loss: 0.1451
34/34              0s 1ms/step
Training Binary Accuracy (CNN, Epoch 17): 0.9600
17/17              0s 6ms/step -
accuracy: 0.0020 - loss: 0.1356
34/34              0s 1ms/step
Training Binary Accuracy (CNN, Epoch 18): 0.9603
17/17              0s 6ms/step -
accuracy: 0.0032 - loss: 0.1372
34/34              0s 2ms/step
Training Binary Accuracy (CNN, Epoch 19): 0.9606
17/17              0s 6ms/step -
accuracy: 0.0017 - loss: 0.1257
34/34              0s 1ms/step
Training Binary Accuracy (CNN, Epoch 20): 0.9664
9/9              0s 53ms/step
Test Set Binary Accuracy (CNN): 0.8968
```

```
/usr/local/lib/python3.10/dist-packages/keras/src/layers/rnn/rnn.py:204:
UserWarning: Do not pass an `input_shape`/`input_dim` argument to a layer. When
using Sequential models, prefer using an `Input(shape)` object as the first
layer in the model instead.
  super().__init__(**kwargs)
```

```
17/17            5s 47ms/step -
accuracy: 0.0339 - loss: 0.6619
34/34            1s 23ms/step
Training Binary Accuracy (LSTM, Epoch 1): 0.8854
17/17            1s 50ms/step -
accuracy: 0.0367 - loss: 0.3534
34/34            1s 16ms/step
Training Binary Accuracy (LSTM, Epoch 2): 0.8999
17/17            1s 38ms/step -
accuracy: 0.0138 - loss: 0.3091
34/34            0s 12ms/step
Training Binary Accuracy (LSTM, Epoch 3): 0.8999
17/17            1s 38ms/step -
accuracy: 0.0119 - loss: 0.2847
34/34            0s 12ms/step
Training Binary Accuracy (LSTM, Epoch 4): 0.8999
17/17            1s 38ms/step -
accuracy: 0.0102 - loss: 0.2899
34/34            0s 12ms/step
Training Binary Accuracy (LSTM, Epoch 5): 0.8999
17/17            1s 38ms/step -
accuracy: 0.0053 - loss: 0.2894
34/34            0s 12ms/step
Training Binary Accuracy (LSTM, Epoch 6): 0.8999
17/17            1s 38ms/step -
accuracy: 0.0014 - loss: 0.2827
34/34            0s 12ms/step
Training Binary Accuracy (LSTM, Epoch 7): 0.8999
17/17            1s 38ms/step -
accuracy: 0.0030 - loss: 0.2947
34/34            0s 12ms/step
Training Binary Accuracy (LSTM, Epoch 8): 0.8999
17/17            1s 38ms/step -
accuracy: 0.0035 - loss: 0.2646
34/34            1s 16ms/step
Training Binary Accuracy (LSTM, Epoch 9): 0.8999
17/17            1s 46ms/step -
accuracy: 4.9648e-04 - loss: 0.2792
34/34            1s 16ms/step
Training Binary Accuracy (LSTM, Epoch 10): 0.8999
17/17            1s 44ms/step -
accuracy: 0.0000e+00 - loss: 0.2744
```

```
34/34              0s 12ms/step
Training Binary Accuracy (LSTM, Epoch 11): 0.8999
17/17              1s 39ms/step -
accuracy: 0.0032 - loss: 0.2758
34/34              0s 12ms/step
Training Binary Accuracy (LSTM, Epoch 12): 0.8999
17/17              1s 38ms/step -
accuracy: 0.0017 - loss: 0.2672
34/34              0s 12ms/step
Training Binary Accuracy (LSTM, Epoch 13): 0.8999
17/17              1s 38ms/step -
accuracy: 0.0000e+00 - loss: 0.2678
34/34              0s 12ms/step
Training Binary Accuracy (LSTM, Epoch 14): 0.8999
17/17              1s 38ms/step -
accuracy: 0.0000e+00 - loss: 0.2720
34/34              0s 12ms/step
Training Binary Accuracy (LSTM, Epoch 15): 0.8999
9/9            0s 13ms/step
Test Set Binary Accuracy (LSTM): 0.9114
F1 Score (CNN): 0.9445
F1 Score (LSTM): 0.9536
```