ECG Heartbeat Categorization Model report

This report explores the performance of different models on ECG classification tasks, including CNN and XGBoost. The goal is to identify the most effective approach for accurately detecting arrhythmia patterns in ECG data.

1. XGBoost

Since this is a medical classification task, recall is prioritized over accuracy, especially for classes representing pathological conditions. In a clinical context, failing to detect a disease (false negative) is generally more critical than raising a false alarm (false positive).

In the test data, the category distribution is still various as following:

- N 18117
- S 556
- V 1448
- F 162
- Q 1608

In the result we are going to use macro avg recall as our benchmark, for two propers:

1. Missing a diagnosis (false negative) can be fatal, false positives can be checked by further test.
2. The imbalance data set is common. We need a benchmark not masked by the dominant class.

The **macro average recall** is chosen as the **benchmark**.

The RandomizedSearchCV has been used to ensure to find the good result without consuming too much resources.

The gamma and reg_alpha are fixed to prevent overfitting.

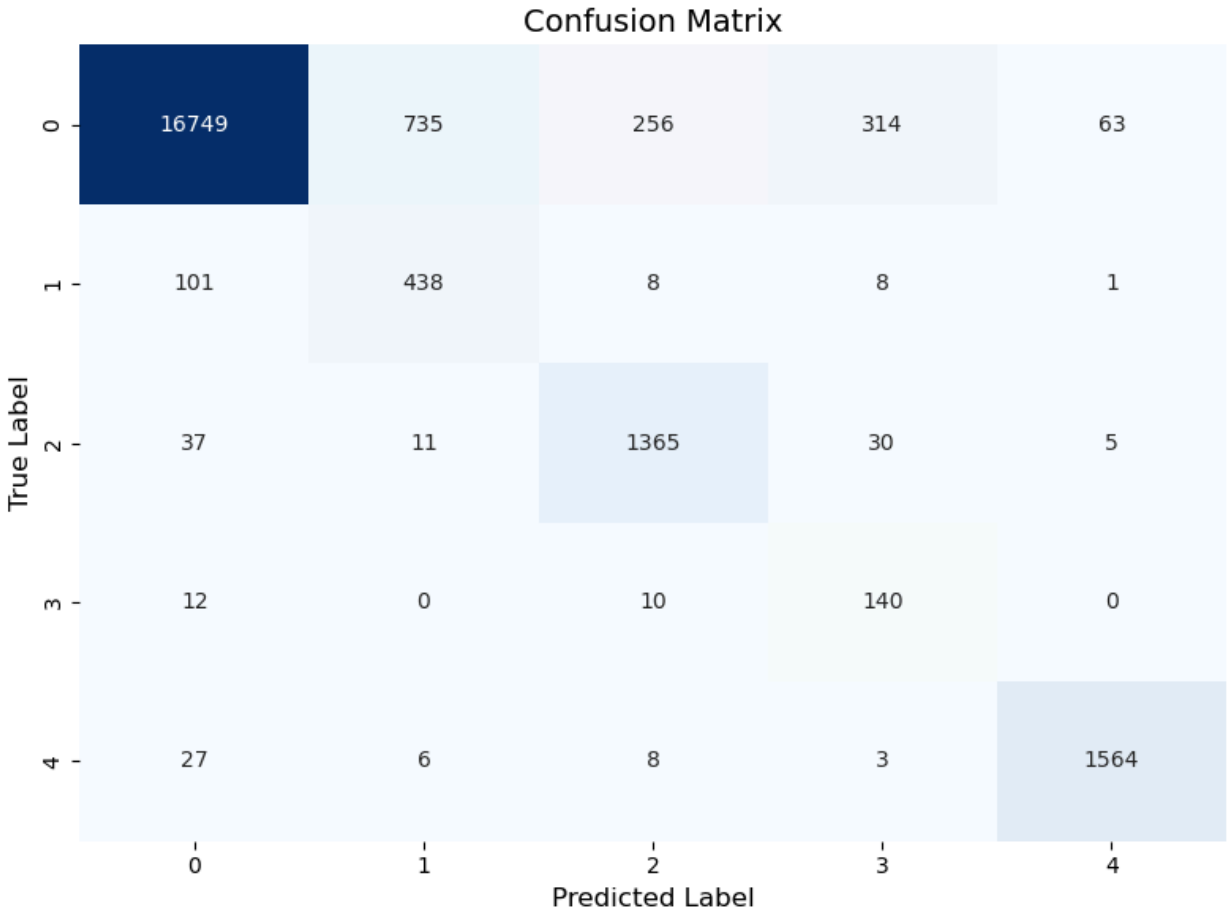
The results:

```
search.best_params_
```
✓ 0.0s

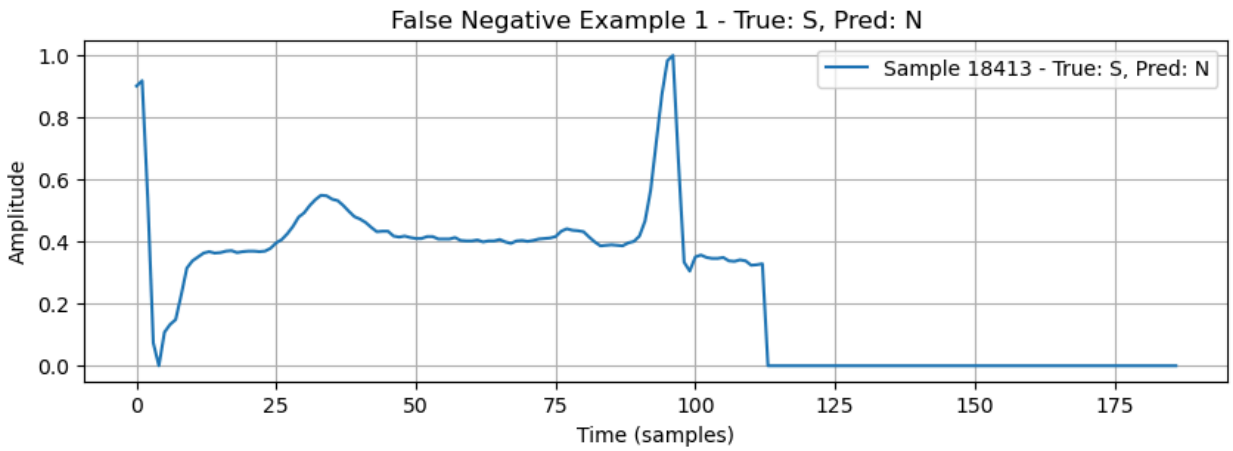
```
{'subsample': 0.6,
 'reg_alpha': 0.5,
 'n_estimators': 300,
 'min_child_weight': 5,
 'max_depth': 5,
 'learning_rate': 0.1,
 'gamma': 0.8,
 'colsample_bytree': 0.8}
```

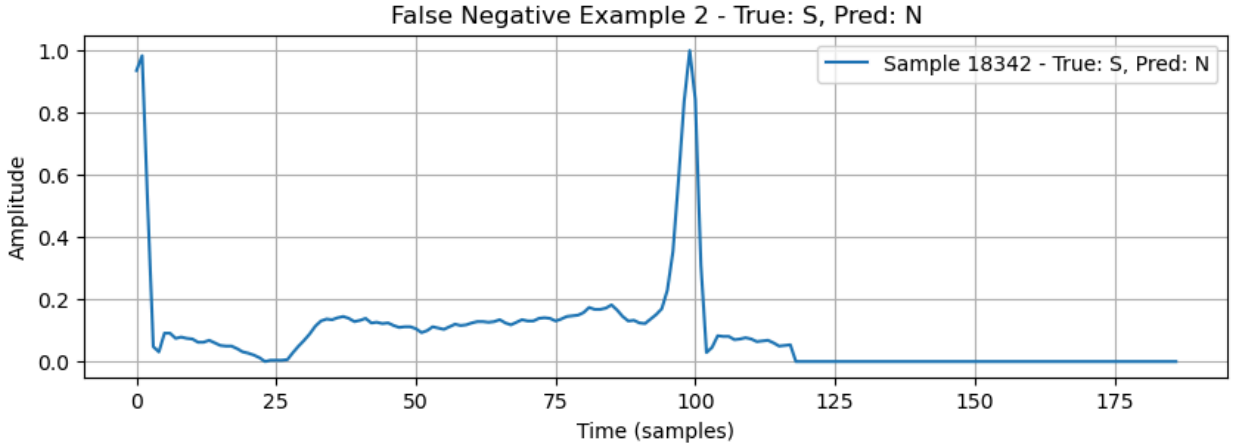
	precision	recall	f1-score	support
0.0	0.990	0.924	0.956	18117
1.0	0.368	0.788	0.502	556
2.0	0.829	0.943	0.882	1448
3.0	0.283	0.864	0.426	162
4.0	0.958	0.973	0.965	1608
accuracy			0.925	21891
macro avg	0.685	0.898	0.746	21891
weighted avg	0.956	0.925	0.936	21891

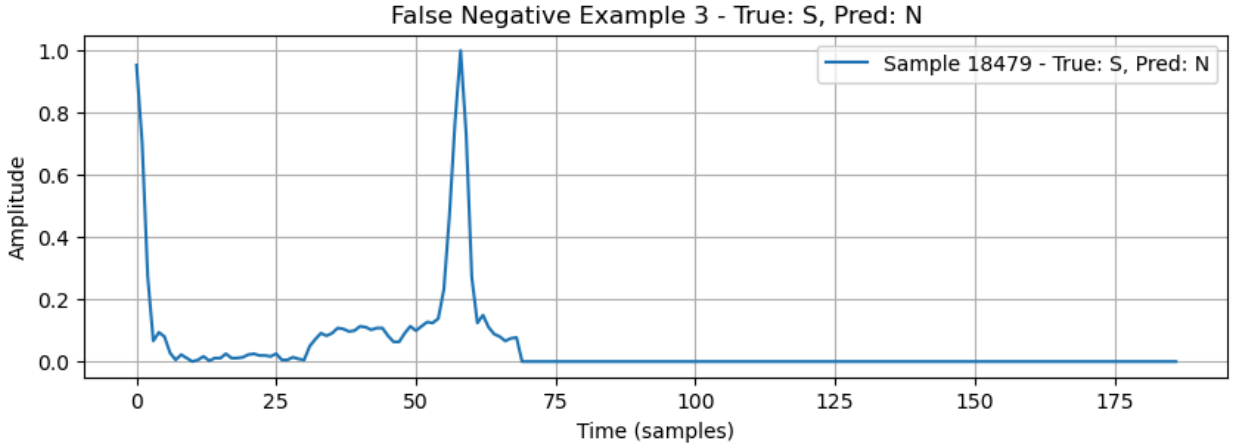
And the confusion matrix:

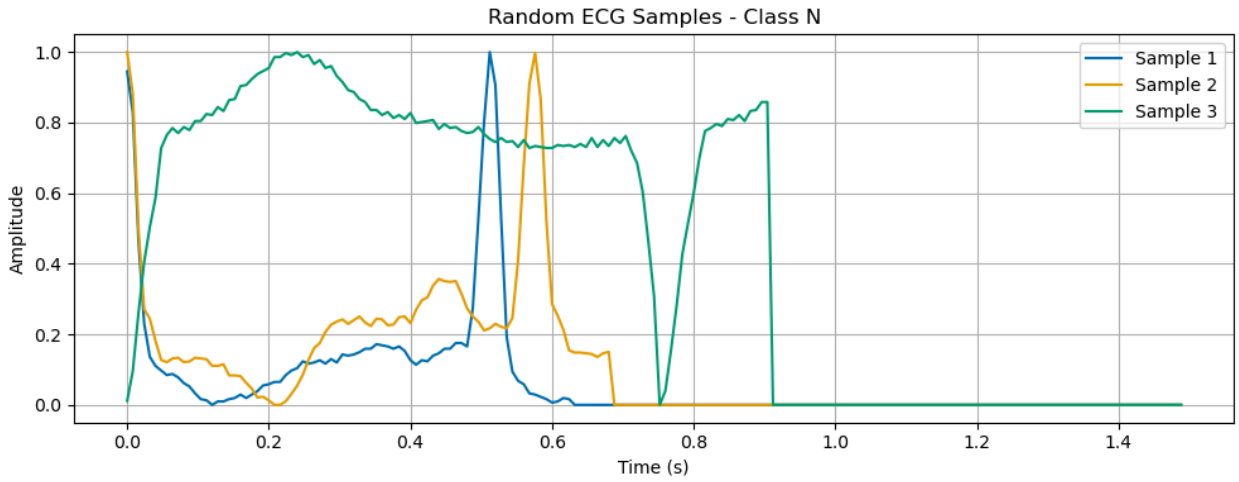


According to the benchmark, the weakest point of the model is recall of class 1(S) is too low. There are 102 entries of class 1(S) that are falsely categorized to class 0(N).


False Negative Example 1 - True: S, Pred: N
Sample 18413 - True: S, Pred: N
Amplitude
Time (samples)

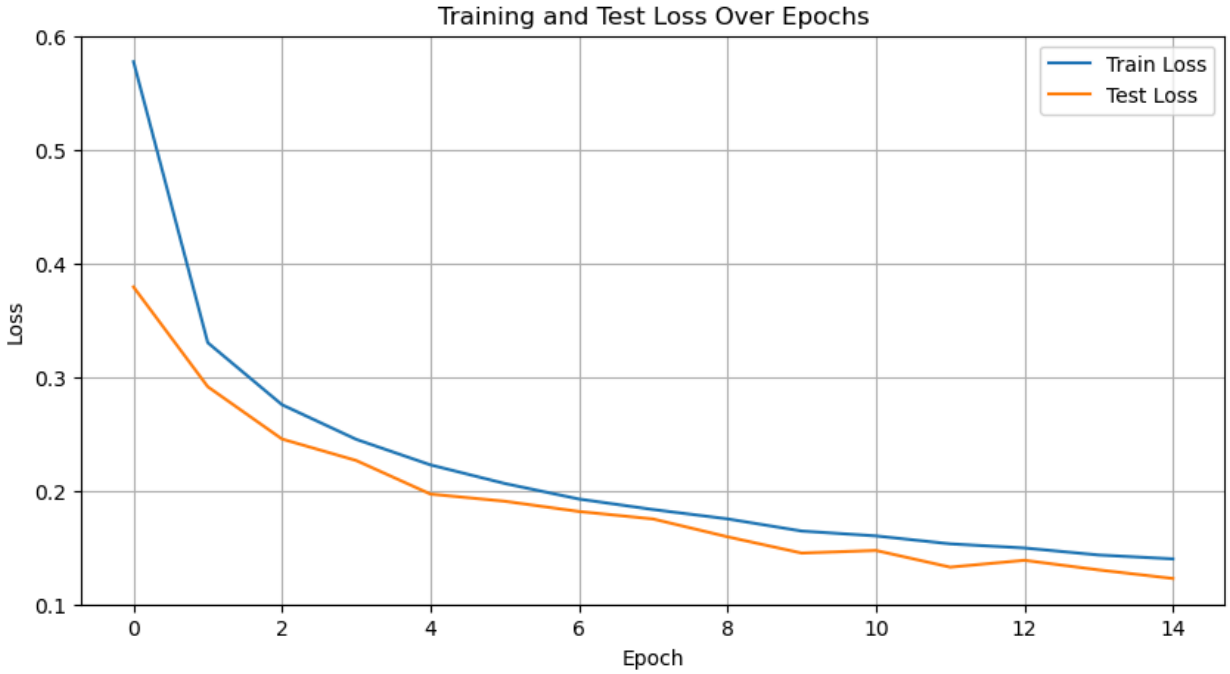

False Negative Example 2 - True: S, Pred: N
Sample 18342 - True: S, Pred: N
Amplitude
Time (samples)


False Negative Example 3 - True: S, Pred: N
Sample 18479 - True: S, Pred: N
Amplitude
Time (samples)

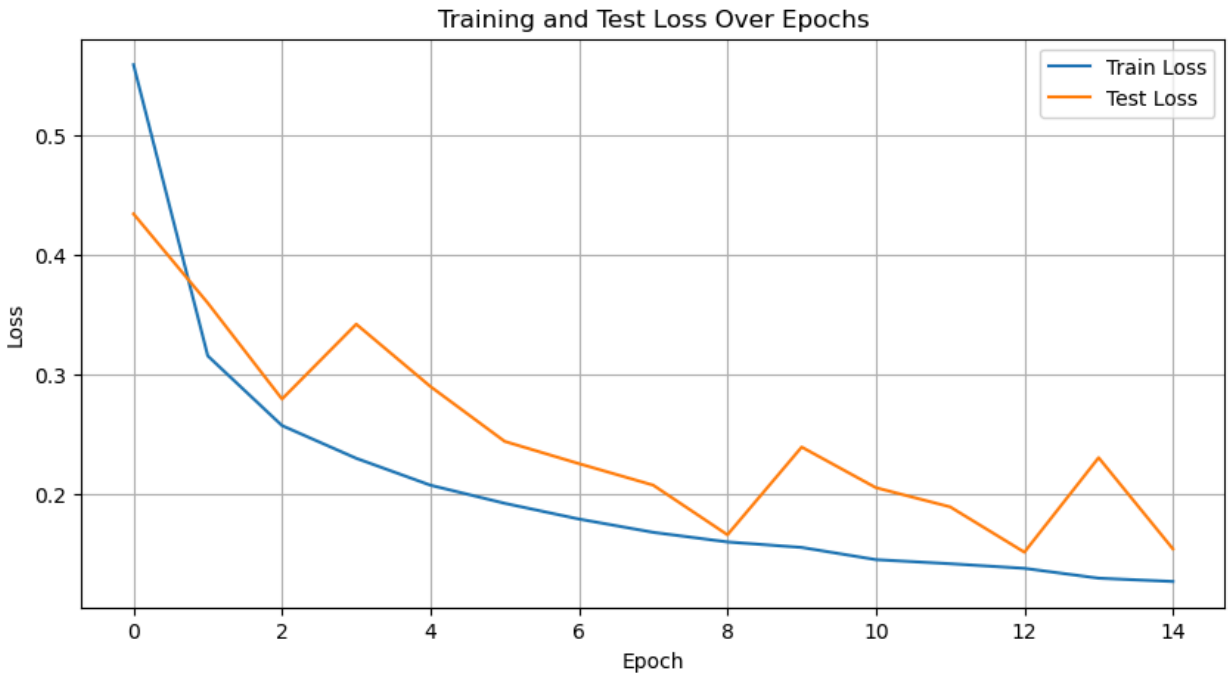


2. CNN

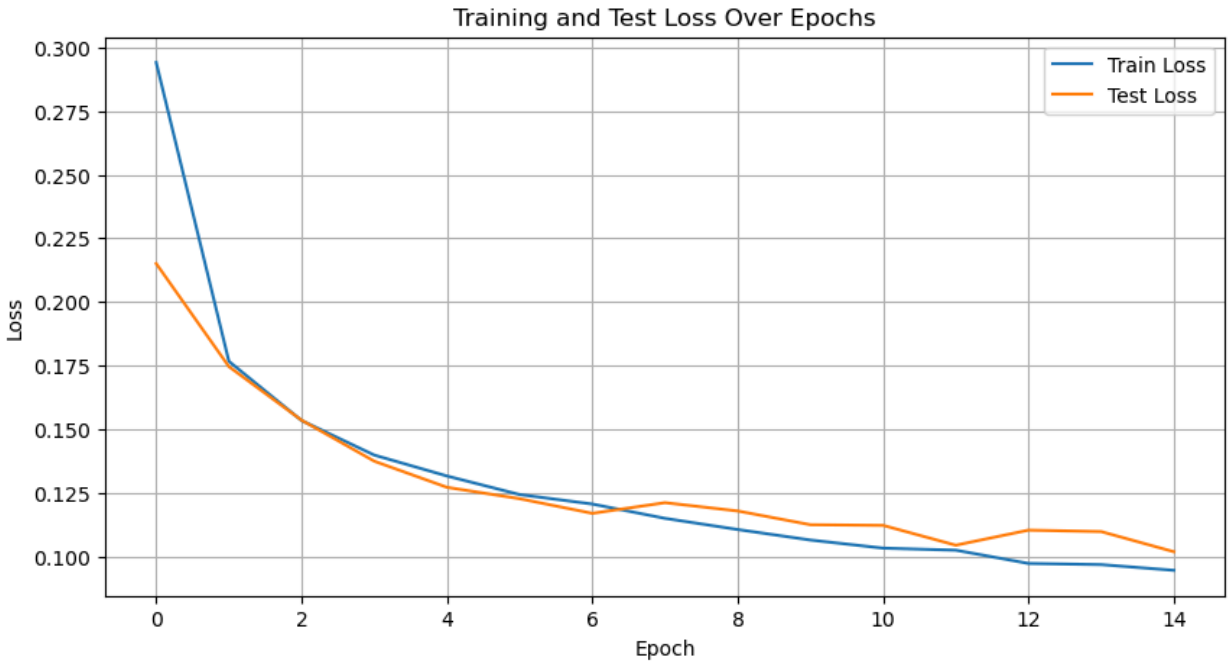
Raw-Data -> SMOTE-Tomek -> Train val split



Raw-Data -> Train val split -> SMOTE-Tomek



Raw-Data -> Train val split -> GAN



CNNv1

```
CNN_v1(
  (conv1): Conv1d(1, 32, kernel_size=(7,), stride=(1,), padding=(3,))
  (pool1): MaxPool1d(kernel_size=2, stride=2, padding=0, dilation=1,
ceil_mode=False)
  (conv2): Conv1d(32, 64, kernel_size=(5,), stride=(1,), padding=(2,))
  (pool2): MaxPool1d(kernel_size=2, stride=2, padding=0, dilation=1,
ceil_mode=False)
  (conv3): Conv1d(64, 128, kernel_size=(3,), stride=(1,), padding=(1,))
```

```
  (pool3): MaxPool1d(kernel_size=2, stride=2, padding=0, dilation=1,
ceil_mode=False)
  (global_pool): AdaptiveMaxPool1d(output_size=1)
  (fc1): Linear(in_features=128, out_features=64, bias=True)
  (relu): ReLU()
  (fc2): Linear(in_features=64, out_features=5, bias=True)
)
```

	precision	recall	f1-score	support
0	0.984	0.992	0.988	18118
1	0.869	0.635	0.734	556
2	0.925	0.946	0.935	1447
3	0.922	0.588	0.718	160
4	0.963	0.980	0.972	1608
accuracy			0.976	21889
macro avg	0.933	0.828	0.869	21889
weighted avg	0.975	0.976	0.975	21889

CNNv2

```
CNN_v2(
  (conv1): Conv1d(1, 32, kernel_size=(7,), stride=(1,), padding=(3,))
  (bn1): BatchNorm1d(32, eps=1e-05, momentum=0.1, affine=True,
track_running_stats=True)
  (pool1): MaxPool1d(kernel_size=2, stride=2, padding=0, dilation=1,
ceil_mode=False)
  (conv2): Conv1d(32, 64, kernel_size=(5,), stride=(1,), padding=(2,))
  (bn2): BatchNorm1d(64, eps=1e-05, momentum=0.1, affine=True,
track_running_stats=True)
  (pool2): MaxPool1d(kernel_size=2, stride=2, padding=0, dilation=1,
ceil_mode=False)
  (conv3): Conv1d(64, 128, kernel_size=(3,), stride=(1,), padding=(1,))
  (bn3): BatchNorm1d(128, eps=1e-05, momentum=0.1, affine=True,
track_running_stats=True)
  (pool3): MaxPool1d(kernel_size=2, stride=2, padding=0, dilation=1,
ceil_mode=False)
  (dropout): Dropout(p=0.5, inplace=False)
  (global_pool): AdaptiveMaxPool1d(output_size=1)
  (fc1): Linear(in_features=128, out_features=64, bias=True)
  (relu): ReLU()
  (fc2): Linear(in_features=64, out_features=5, bias=True)
)
```

	precision	recall	f1-score	support
0	0.984	0.996	0.990	18118
1	0.893	0.658	0.758	556
2	0.973	0.933	0.953	1447
3	0.889	0.750	0.814	160
4	0.983	0.991	0.987	1608
accuracy			0.981	21889
macro avg	0.944	0.866	0.900	21889
weighted avg	0.980	0.981	0.980	21889

ResNet

```
ResNet1D(
  (initial): Sequential(
    (0): Conv1d(1, 64, kernel_size=(7,), stride=(2,), padding=(3,), bias=False)
    (1): BatchNorm1d(64, eps=1e-05, momentum=0.1, affine=True,
track_running_stats=True)
    (2): ReLU(inplace=True)
    (3): MaxPool1d(kernel_size=3, stride=2, padding=1, dilation=1,
ceil_mode=False)
  )
  (layer1): Sequential(
    (0): ResidualBlock1D(
      (conv1): Conv1d(64, 64, kernel_size=(3,), stride=(1,), padding=(1,),
bias=False)
      (bn1): BatchNorm1d(64, eps=1e-05, momentum=0.1, affine=True,
track_running_stats=True)
      (relu): ReLU(inplace=True)
      (conv2): Conv1d(64, 64, kernel_size=(3,), stride=(1,), padding=(1,),
bias=False)
      (bn2): BatchNorm1d(64, eps=1e-05, momentum=0.1, affine=True,
track_running_stats=True)
    )
    (1): ResidualBlock1D(
      (conv1): Conv1d(64, 64, kernel_size=(3,), stride=(1,), padding=(1,),
bias=False)
      (bn1): BatchNorm1d(64, eps=1e-05, momentum=0.1, affine=True,
track_running_stats=True)
      (relu): ReLU(inplace=True)
      (conv2): Conv1d(64, 64, kernel_size=(3,), stride=(1,), padding=(1,),
bias=False)
      (bn2): BatchNorm1d(64, eps=1e-05, momentum=0.1, affine=True,
track_running_stats=True)
    )
  )
  (layer2): Sequential(
    (0): ResidualBlock1D(
      (conv1): Conv1d(64, 128, kernel_size=(3,), stride=(2,), padding=(1,),
bias=False)
      (bn1): BatchNorm1d(128, eps=1e-05, momentum=0.1, affine=True,
track_running_stats=True)
      (relu): ReLU(inplace=True)
      (conv2): Conv1d(128, 128, kernel_size=(3,), stride=(1,), padding=(1,),
bias=False)
      (bn2): BatchNorm1d(128, eps=1e-05, momentum=0.1, affine=True,
track_running_stats=True)
      (downsample): Sequential(
        (0): Conv1d(64, 128, kernel_size=(1,), stride=(2,), bias=False)
        (1): BatchNorm1d(128, eps=1e-05, momentum=0.1, affine=True,
track_running_stats=True)
      )
    )
    (1): ResidualBlock1D(
      (conv1): Conv1d(128, 128, kernel_size=(3,), stride=(1,), padding=(1,),
bias=False)
      (bn1): BatchNorm1d(128, eps=1e-05, momentum=0.1, affine=True,
track_running_stats=True)
```

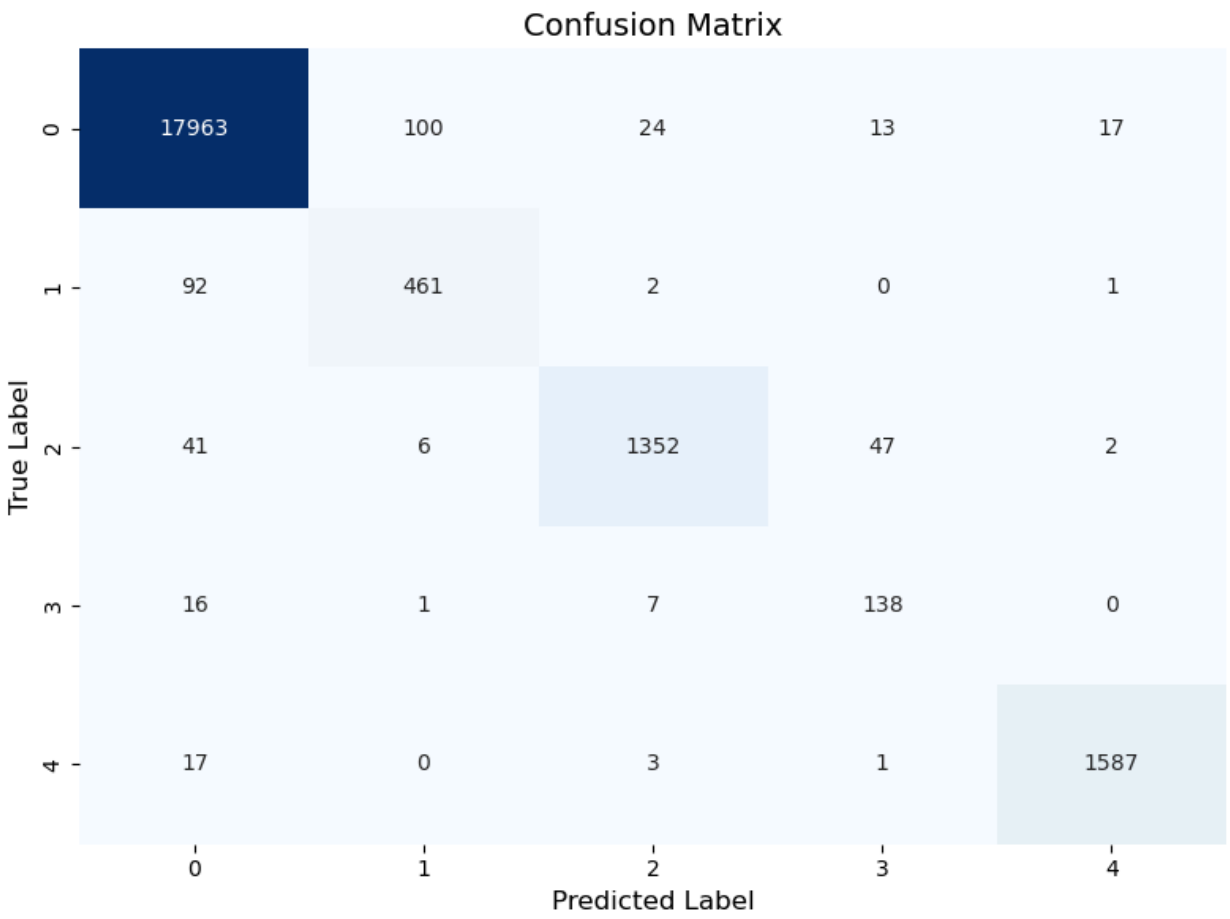
```
      (relu): ReLU(inplace=True)
      (conv2): Conv1d(128, 128, kernel_size=(3,), stride=(1,), padding=(1,),
bias=False)
      (bn2): BatchNorm1d(128, eps=1e-05, momentum=0.1, affine=True,
track_running_stats=True)
    )
  )
  (layer3): Sequential(
    (0): ResidualBlock1D(
      (conv1): Conv1d(128, 256, kernel_size=(3,), stride=(2,), padding=(1,),
bias=False)
      (bn1): BatchNorm1d(256, eps=1e-05, momentum=0.1, affine=True,
track_running_stats=True)
      (relu): ReLU(inplace=True)
      (conv2): Conv1d(256, 256, kernel_size=(3,), stride=(1,), padding=(1,),
bias=False)
      (bn2): BatchNorm1d(256, eps=1e-05, momentum=0.1, affine=True,
track_running_stats=True)
      (downsample): Sequential(
        (0): Conv1d(128, 256, kernel_size=(1,), stride=(2,), bias=False)
        (1): BatchNorm1d(256, eps=1e-05, momentum=0.1, affine=True,
track_running_stats=True)
      )
    )
    (1): ResidualBlock1D(
      (conv1): Conv1d(256, 256, kernel_size=(3,), stride=(1,), padding=(1,),
bias=False)
      (bn1): BatchNorm1d(256, eps=1e-05, momentum=0.1, affine=True,
track_running_stats=True)
      (relu): ReLU(inplace=True)
      (conv2): Conv1d(256, 256, kernel_size=(3,), stride=(1,), padding=(1,),
bias=False)
      (bn2): BatchNorm1d(256, eps=1e-05, momentum=0.1, affine=True,
track_running_stats=True)
    )
  )
  (layer4): Sequential(
    (0): ResidualBlock1D(
      (conv1): Conv1d(256, 512, kernel_size=(3,), stride=(2,), padding=(1,),
bias=False)
      (bn1): BatchNorm1d(512, eps=1e-05, momentum=0.1, affine=True,
track_running_stats=True)
      (relu): ReLU(inplace=True)
      (conv2): Conv1d(512, 512, kernel_size=(3,), stride=(1,), padding=(1,),
bias=False)
      (bn2): BatchNorm1d(512, eps=1e-05, momentum=0.1, affine=True,
track_running_stats=True)
      (downsample): Sequential(
        (0): Conv1d(256, 512, kernel_size=(1,), stride=(2,), bias=False)
        (1): BatchNorm1d(512, eps=1e-05, momentum=0.1, affine=True,
track_running_stats=True)
      )
    )
    (1): ResidualBlock1D(
```

```
      (conv1): Conv1d(512, 512, kernel_size=(3,), stride=(1,), padding=(1,),
bias=False)
      (bn1): BatchNorm1d(512, eps=1e-05, momentum=0.1, affine=True,
track_running_stats=True)
      (relu): ReLU(inplace=True)
      (conv2): Conv1d(512, 512, kernel_size=(3,), stride=(1,), padding=(1,),
bias=False)
      (bn2): BatchNorm1d(512, eps=1e-05, momentum=0.1, affine=True,
track_running_stats=True)
    )
  )
  (global_pool): AdaptiveAvgPool1d(output_size=1)
  (dropout): Dropout(p=0.5, inplace=False)
  (fc): Linear(in_features=512, out_features=5, bias=True)
)
```

	precision	recall	f1-score	support
0	0.993	0.991	0.992	18118
1	0.854	0.876	0.865	556
2	0.972	0.952	0.962	1447
3	0.738	0.863	0.795	160
4	0.979	0.990	0.985	1608
accuracy			0.985	21889
macro avg	0.907	0.934	0.920	21889
weighted avg	0.985	0.985	0.985	21889

3. Evaluation and Comparison

	precision	recall	f1-score	support
0	0.991	0.991	0.991	18117
1	0.812	0.829	0.820	556
2	0.974	0.934	0.953	1448
3	0.693	0.852	0.765	162
4	0.988	0.987	0.987	1608
accuracy			0.982	21891
macro avg	0.892	0.919	0.903	21891
weighted avg	0.983	0.982	0.982	21891



4. Conclusion and future work

The test set perform the marco avg recall on 0.919 which is much better than ensemble trees methods

LSTM could be implemented to improve the model.