SSB GCN Modeli - 1

May 10, 2025

[14]:
```
import pandas as pd
import torch
import numpy as np
from sklearn.preprocessing import StandardScaler, LabelEncoder
from sklearn.neighbors import kneighbors_graph
from torch_geometric.data import Data
from torch_geometric.utils import from_scipy_sparse_matrix
from torch_geometric.nn import GCNConv
import torch.nn.functional as F
from sklearn.metrics import classification_report, confusion_matrix
from sklearn.utils.class_weight import compute_class_weight
```

[18]:
```
# CSV dosyasını oku
df = pd.read_csv("C:\\Users\\Lenovo\\Desktop\\OneDrive_2025-05-07\\TON_IoT␣
 ↪datasets\\Processed_datasets\\Processed_IoT_dataset\\IoT_Fridge.csv")

# Sayısal olmayan sütunları çıkar
features = df.drop(['date', 'time', 'type', 'label', 'temp_condition'], axis=1)

# Etiketleri al ve sayısal forma dönüştür
labels = df['label']
label_encoder = LabelEncoder()
y = torch.tensor(label_encoder.fit_transform(labels), dtype=torch.long)

# Veriyi normalize et
scaler = StandardScaler()
X_scaled = scaler.fit_transform(features)
x = torch.tensor(X_scaled, dtype=torch.float)

# Numpy array'e çevir
y_np = data.y.numpy()

# Ağırlıkları hesapla (0 = Normal, 1 = Anomali)
class_weights = compute_class_weight(class_weight='balanced', classes=np.
 ↪unique(y_np), y=y_np)
class_weights = torch.tensor(class_weights, dtype=torch.float)
```

```
print(f"Sınıf ağırlıkları: {class_weights}")
```

```
Sınıf ağırlıkları: tensor([0.5861, 3.4034])
```

[19]:
```
# KNN tabanlı adjacency matrisi oluştur
knn_graph = kneighbors_graph(X_scaled, n_neighbors=10, mode='connectivity',␣
 ↪include_self=False)
edge_index, _ = from_scipy_sparse_matrix(knn_graph)

# PyG formatında veri objesi oluştur
data = Data(x=x, edge_index=edge_index, y=y)
```

[20]:
```
class GCN(torch.nn.Module):
    def __init__(self, in_channels, hidden_channels, out_channels):
        super(GCN, self).__init__()
        self.conv1 = GCNConv(in_channels, hidden_channels)
        self.conv2 = GCNConv(hidden_channels, out_channels)

    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)
        return x
```

[21]:
```
model = GCN(in_channels=data.num_node_features, hidden_channels=64,␣
 ↪out_channels=2)
optimizer = torch.optim.Adam(model.parameters(), lr=0.01, weight_decay=5e-4)
criterion = torch.nn.CrossEntropyLoss(weight=class_weights)

model.train()
for epoch in range(1, 201):
    optimizer.zero_grad()
    out = model(data)
    loss = criterion(out, data.y)
    loss.backward()
    optimizer.step()
    if epoch % 20 == 0:
        print(f"Epoch {epoch}, Loss: {loss.item():.4f}")
```

```
Epoch 20, Loss: 0.6923
Epoch 40, Loss: 0.6922
Epoch 60, Loss: 0.6921
Epoch 80, Loss: 0.6921
Epoch 100, Loss: 0.6921
Epoch 120, Loss: 0.6921
Epoch 140, Loss: 0.6921
Epoch 160, Loss: 0.6921
```

```
Epoch 180, Loss: 0.6921
Epoch 200, Loss: 0.6921
```

[22]:
```
model.eval()

# Tahminler
pred = model(data).argmax(dim=1)
true_labels = data.y.cpu().numpy()
predictions = pred.cpu().numpy()

# Confusion matrix ve classification raporu
print("Confusion Matrix:")
print(confusion_matrix(true_labels, predictions))

print("\nClassification Report:")
print(classification_report(true_labels, predictions, target_names=["Normal",␣
 ↪"Anomali"], zero_division=0))
```

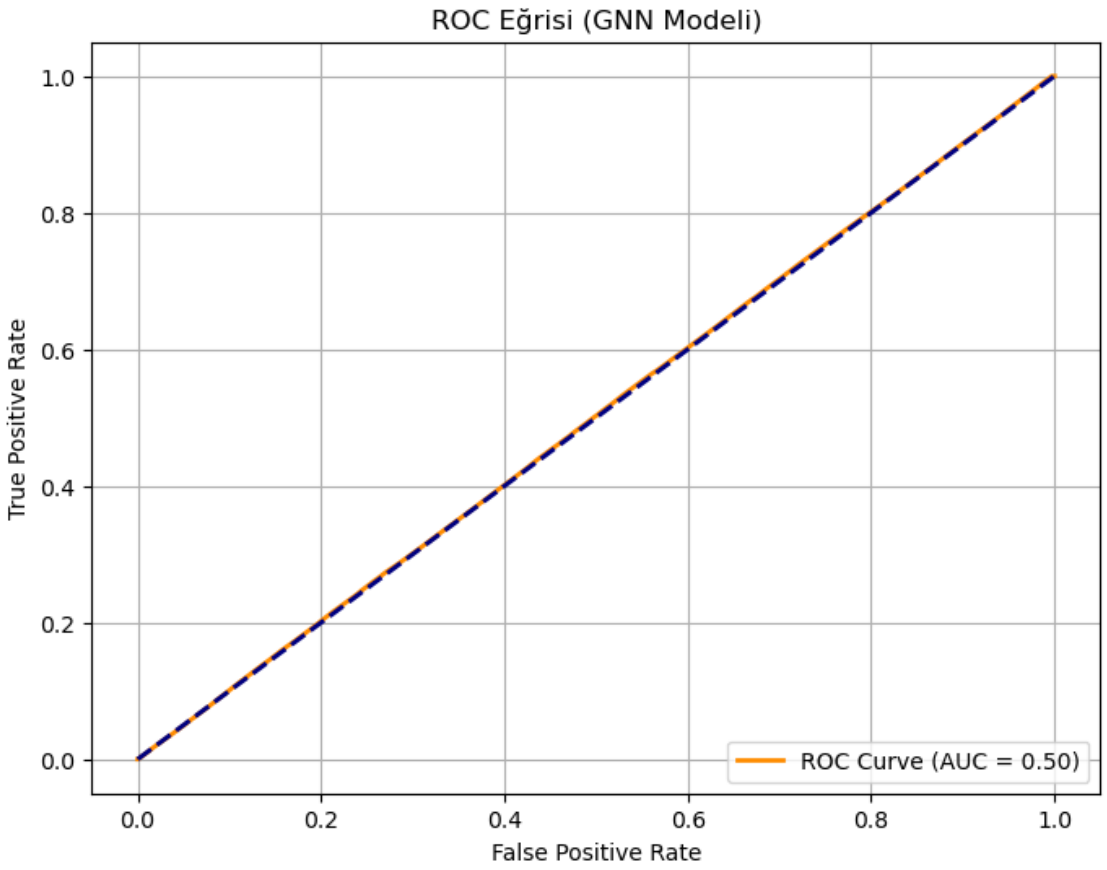
```
Confusion Matrix:
[[103169 397658]
 [ 17641  68608]]

Classification Report:
              precision    recall  f1-score   support

      Normal       0.85      0.21      0.33    500827
     Anomali       0.15      0.80      0.25     86249

    accuracy                           0.29    587076
   macro avg       0.50      0.50      0.29    587076
weighted avg       0.75      0.29      0.32    587076
```

- Anomali tespiti yapılmaya başlandı! Önceden model hiç anomaliyi yakalayamıyordu.
- Şimdi:
- GCN modeline class weight (sınıf ağırlığı) ekleyerek "anomalileri" daha görünür hale getirmeyi denedim.
- 68,608 anomaliden doğru tahmin etmiş,
- Yanlışlıkla 17,641 tanesini normal sanmış.

- Anomali recall = 0.80 → Bu çok iyi. Yani gerçek anomalilerin %80'ini bulabiliyorsun.
- Normal precision = 0.85 → Model normal verilerde hâlâ sağlam.
- F1-score biraz düşük çünkü dengesiz sınıflarda zor toparlanıyor.

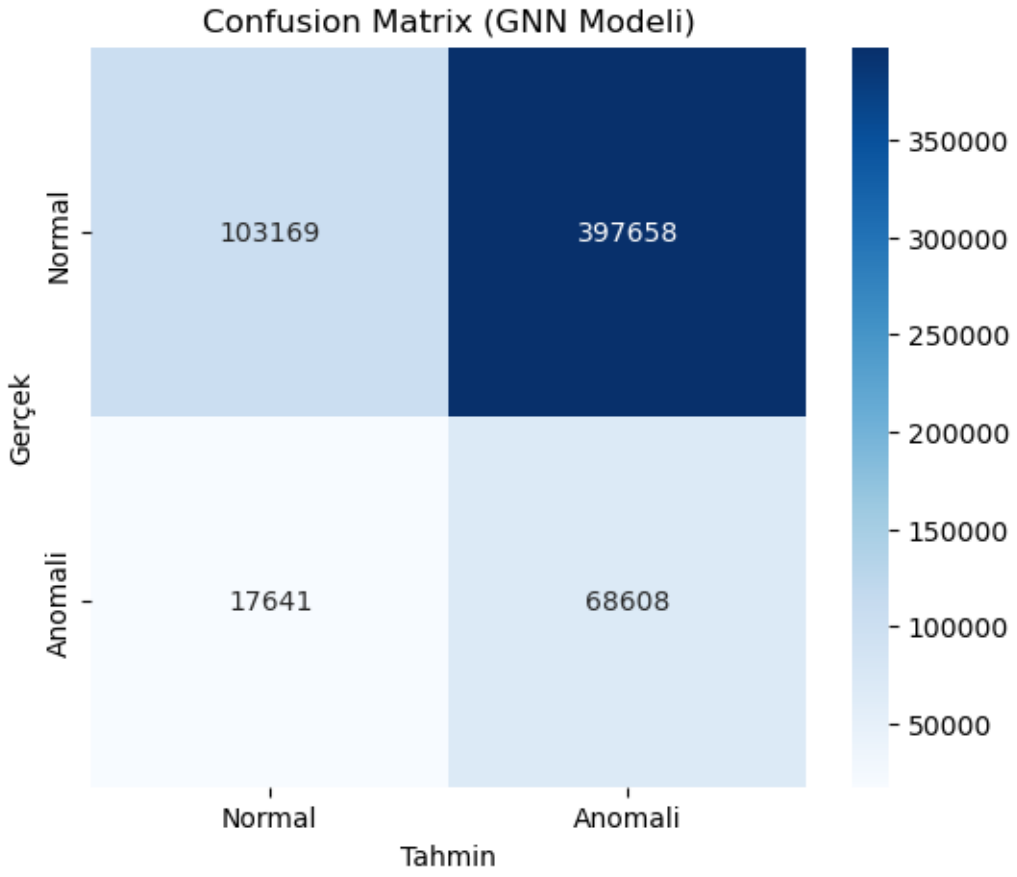
[23]:
```
import matplotlib.pyplot as plt
from sklearn.metrics import roc_curve, auc, confusion_matrix,␣
 ↪ConfusionMatrixDisplay
import seaborn as sns
```

```python
# Tahmin olasılıkları (softmax değilse sigmoid kullanılabilir)
y_proba = model(data).detach().cpu().numpy()[:, 1]  # sadece sınıf 1 (anomali) için olasılık

# Gerçek etiketleri al
y_true = data.y.cpu().numpy()

# ROC Curve ve AUC
fpr, tpr, thresholds = roc_curve(y_true, y_proba)
roc_auc = auc(fpr, tpr)

# Çiz
plt.figure(figsize=(8, 6))
plt.plot(fpr, tpr, color='darkorange', lw=2, label=f'ROC Curve (AUC = {roc_auc:.2f})')
plt.plot([0, 1], [0, 1], color='navy', lw=2, linestyle='--')
plt.xlabel('False Positive Rate')
plt.ylabel('True Positive Rate')
plt.title('ROC Eğrisi (GNN Modeli)')
plt.legend(loc='lower right')
plt.grid(True)
plt.show()
```



- AUC = 0.50, modelin pozitif (anomali) ve negatif (normal) sınıfları ayırt edemediğini gösterir.
- ROC eğrisi tam olarak köşegen üzerindeyse (False Positive Rate = True Positive Rate), bu modelin şans eseri tahmin yaptığı anlamına gelir.

[24]:
```
# Confusion Matrix
cm = confusion_matrix(y_true, predictions)

# Görsel çizim
plt.figure(figsize=(6, 5))
sns.heatmap(cm, annot=True, fmt='d', cmap='Blues',
            xticklabels=['Normal', 'Anomali'],
            yticklabels=['Normal', 'Anomali'])
plt.title('Confusion Matrix (GNN Modeli)')
plt.xlabel('Tahmin')
plt.ylabel('Gerçek')
plt.show()
```



Doğru Tahminler - True Normal (TN): 103,169 - True Anomaly (TP): 68,608

Hatalı Tahminler - False Negative (FN): 17,641 → Anomaliyken "Normal" denmiş. - False Positive (FP): 397,658 → Normalken "Anomali" denmiş.

- Anomalilerde başarı (Recall) oldukça yüksek: 68,608 / (68,608 + 17,641) 0.79
- Normal sınıf için Precision çok düşük: Çünkü çok fazla normal örnek "anomali" olarak etiketlenmiş (yüksek FP).
- Bu da Precision değerini düşürür, çünkü model gereğinden fazla alarm veriyor.

[ ]: