SSB LSTM Modeli - 1

May 9, 2025

[1]:
```
import pandas as pd
import numpy as np
from sklearn.preprocessing import StandardScaler
from sklearn.model_selection import train_test_split
from sklearn.metrics import confusion_matrix, classification_report

import tensorflow as tf
from tensorflow.keras.models import Sequential
from tensorflow.keras.layers import LSTM, Dense, Dropout
```

[2]:
```
# Veri oku
df = pd.read_csv("C:\\Users\\Lenovo\\Desktop\\OneDrive_2025-05-07\\TON_IoT␣
 ↪datasets\\Processed_datasets\\Processed_IoT_dataset\\IoT_Fridge.csv")
df.drop(['date', 'time', 'type'], axis=1, inplace=True)
df = pd.get_dummies(df, columns=['temp_condition'], drop_first=True)

# Giriş ve hedef
X = df.drop('label', axis=1)
y = df['label']

# Ölçekle
scaler = StandardScaler()
X_scaled = scaler.fit_transform(X)

# LSTM için input [örnek_sayısı, zaman_adımı, özellik_sayısı]
# Burada her veri noktasını 1 zaman adımı kabul ediyoruz
X_lstm = X_scaled.reshape((X_scaled.shape[0], 1, X_scaled.shape[1]))

# Eğitim/test böl
X_train, X_test, y_train, y_test = train_test_split(X_lstm, y, test_size=0.2,␣
 ↪random_state=42)
```

[3]:
```
model = Sequential()
model.add(LSTM(64, input_shape=(X_train.shape[1], X_train.shape[2]),␣
 ↪return_sequences=False))
model.add(Dropout(0.3))
model.add(Dense(1, activation='sigmoid'))  # Binary output
```

```
model.compile(loss='binary_crossentropy', optimizer='adam',␣
 ↪metrics=['accuracy'])

# Eğit
history = model.fit(X_train, y_train, epochs=30, batch_size=64,␣
 ↪validation_split=0.2,
                    class_weight={0: 1, 1: 4})  # Anomalilere ağırlık veriyoruz
```

```
Epoch 1/30
5871/5871 [==============================] - 19s 3ms/step - loss: 0.4610 -
accuracy: 0.7991 - val_loss: 0.3109 - val_accuracy: 0.8007
Epoch 2/30
5871/5871 [==============================] - 17s 3ms/step - loss: 0.4484 -
accuracy: 0.7995 - val_loss: 0.3003 - val_accuracy: 0.8007
Epoch 3/30
5871/5871 [==============================] - 17s 3ms/step - loss: 0.4481 -
accuracy: 0.7995 - val_loss: 0.3138 - val_accuracy: 0.8007
Epoch 4/30
5871/5871 [==============================] - 17s 3ms/step - loss: 0.4477 -
accuracy: 0.7995 - val_loss: 0.3058 - val_accuracy: 0.8007
Epoch 5/30
5871/5871 [==============================] - 16s 3ms/step - loss: 0.4476 -
accuracy: 0.7995 - val_loss: 0.3117 - val_accuracy: 0.8007
Epoch 6/30
5871/5871 [==============================] - 16s 3ms/step - loss: 0.4475 -
accuracy: 0.7995 - val_loss: 0.3050 - val_accuracy: 0.8007
Epoch 7/30
5871/5871 [==============================] - 17s 3ms/step - loss: 0.4473 -
accuracy: 0.7995 - val_loss: 0.3096 - val_accuracy: 0.8007
Epoch 8/30
5871/5871 [==============================] - 17s 3ms/step - loss: 0.4472 -
accuracy: 0.7995 - val_loss: 0.3123 - val_accuracy: 0.8007
Epoch 9/30
5871/5871 [==============================] - 17s 3ms/step - loss: 0.4472 -
accuracy: 0.7995 - val_loss: 0.3218 - val_accuracy: 0.8007
Epoch 10/30
5871/5871 [==============================] - 17s 3ms/step - loss: 0.4470 -
accuracy: 0.7995 - val_loss: 0.3188 - val_accuracy: 0.8007
Epoch 11/30
5871/5871 [==============================] - 17s 3ms/step - loss: 0.4470 -
accuracy: 0.7995 - val_loss: 0.3195 - val_accuracy: 0.8007
Epoch 12/30
5871/5871 [==============================] - 17s 3ms/step - loss: 0.4469 -
accuracy: 0.7995 - val_loss: 0.3143 - val_accuracy: 0.8007
Epoch 13/30
5871/5871 [==============================] - 17s 3ms/step - loss: 0.4469 -
```

```
accuracy: 0.7995 - val_loss: 0.3166 - val_accuracy: 0.8007
Epoch 14/30
5871/5871 [==============================] - 17s 3ms/step - loss: 0.4468 -
accuracy: 0.7995 - val_loss: 0.3185 - val_accuracy: 0.8007
Epoch 15/30
5871/5871 [==============================] - 18s 3ms/step - loss: 0.4468 -
accuracy: 0.7995 - val_loss: 0.3093 - val_accuracy: 0.8007
Epoch 16/30
5871/5871 [==============================] - 18s 3ms/step - loss: 0.4467 -
accuracy: 0.7995 - val_loss: 0.3133 - val_accuracy: 0.8007
Epoch 17/30
5871/5871 [==============================] - 17s 3ms/step - loss: 0.4468 -
accuracy: 0.7995 - val_loss: 0.3181 - val_accuracy: 0.8007
Epoch 18/30
5871/5871 [==============================] - 17s 3ms/step - loss: 0.4467 -
accuracy: 0.7995 - val_loss: 0.3169 - val_accuracy: 0.8007
Epoch 19/30
5871/5871 [==============================] - 17s 3ms/step - loss: 0.4467 -
accuracy: 0.7995 - val_loss: 0.3143 - val_accuracy: 0.8007
Epoch 20/30
5871/5871 [==============================] - 17s 3ms/step - loss: 0.4467 -
accuracy: 0.7995 - val_loss: 0.3159 - val_accuracy: 0.8007
Epoch 21/30
5871/5871 [==============================] - 17s 3ms/step - loss: 0.4467 -
accuracy: 0.7995 - val_loss: 0.3159 - val_accuracy: 0.8007
Epoch 22/30
5871/5871 [==============================] - 17s 3ms/step - loss: 0.4467 -
accuracy: 0.7995 - val_loss: 0.3165 - val_accuracy: 0.8007
Epoch 23/30
5871/5871 [==============================] - 17s 3ms/step - loss: 0.4467 -
accuracy: 0.7995 - val_loss: 0.3164 - val_accuracy: 0.8007
Epoch 24/30
5871/5871 [==============================] - 17s 3ms/step - loss: 0.4467 -
accuracy: 0.7995 - val_loss: 0.3125 - val_accuracy: 0.8007
Epoch 25/30
5871/5871 [==============================] - 17s 3ms/step - loss: 0.4467 -
accuracy: 0.7995 - val_loss: 0.3181 - val_accuracy: 0.8007
Epoch 26/30
5871/5871 [==============================] - 17s 3ms/step - loss: 0.4467 -
accuracy: 0.7995 - val_loss: 0.3169 - val_accuracy: 0.8007
Epoch 27/30
5871/5871 [==============================] - 17s 3ms/step - loss: 0.4467 -
accuracy: 0.7995 - val_loss: 0.3143 - val_accuracy: 0.8007
Epoch 28/30
5871/5871 [==============================] - 18s 3ms/step - loss: 0.4467 -
accuracy: 0.7995 - val_loss: 0.3147 - val_accuracy: 0.8007
Epoch 29/30
5871/5871 [==============================] - 17s 3ms/step - loss: 0.4467 -
```

```
accuracy: 0.7995 - val_loss: 0.3164 - val_accuracy: 0.8007
Epoch 30/30
5871/5871 [==============================] - 17s 3ms/step - loss: 0.4467 -
accuracy: 0.7995 - val_loss: 0.3164 - val_accuracy: 0.8007
```

[4]:
```
y_pred = model.predict(X_test)
y_pred = [1 if i > 0.5 else 0 for i in y_pred]

print("Confusion Matrix:")
print(confusion_matrix(y_test, y_pred))

print("\nClassification Report:")
print(classification_report(y_test, y_pred))
```

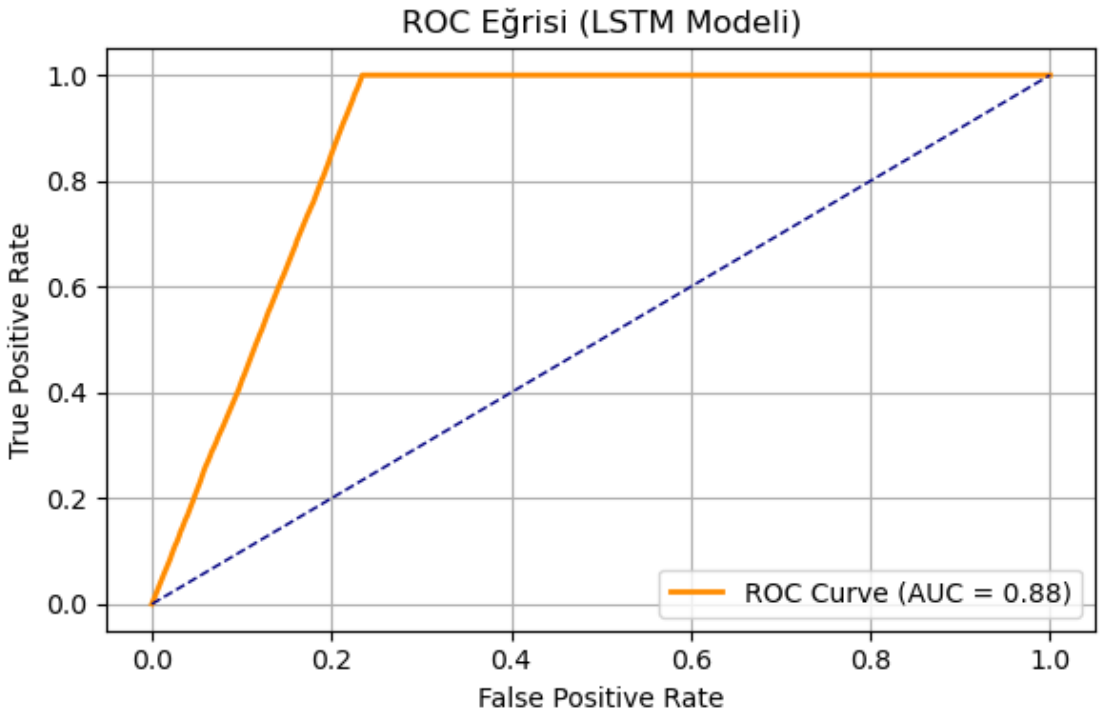
```
3670/3670 [==============================] - 5s 1ms/step
Confusion Matrix:
[[76619 23408]
 [    0 17389]]

Classification Report:
              precision    recall  f1-score   support

           0       1.00      0.77      0.87    100027
           1       0.43      1.00      0.60     17389

    accuracy                           0.80    117416
   macro avg       0.71      0.88      0.73    117416
weighted avg       0.92      0.80      0.83    117416
```

-Recall (Anomali) = 1.00 Anomalilerin hepsi yakalanmış! Hiçbir saldırıyı kaçırmamış. -Precision (Anomali) = 0.43 Tahmin ettiği "anomalilerin" sadece %43'ü gerçekten anomali. Yani biraz false alarm (yanlış pozitif) üretiyor. Ancak bu tür sistemlerde "alarm vereyim de yanlış çıksın, yeter ki gerçek saldırıyı kaçırmayayım" yaklaşımı geçerlidir -Accuracy = 0.80 Verinin %80'i doğru tahmin edilmiş. -F1-Score (Anomali) = 0.60 Precision ve Recall arasında bir denge. LSTM modeli anomaliyi yakalıyor ama biraz alarm üretiyor.

[5]:
```
from sklearn.metrics import roc_curve, auc
import matplotlib.pyplot as plt

# LSTM modeli .predict() sonucu: 0 ile 1 arasında olasılıklar verir
y_proba = model.predict(X_test)

# ROC eğrisi hesapla
fpr, tpr, thresholds = roc_curve(y_test, y_proba)
roc_auc = auc(fpr, tpr)

# ROC eğrisi çiz
```

```
plt.figure(figsize=(6, 4))
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=1, linestyle='--')
plt.xlabel('False Positive Rate')
plt.ylabel('True Positive Rate')
plt.title('ROC Eğrisi (LSTM Modeli)')
plt.legend(loc="lower right")
plt.grid(True)
plt.tight_layout()
plt.show()
```

```
3670/3670 [==============================] - 5s 1ms/step
```

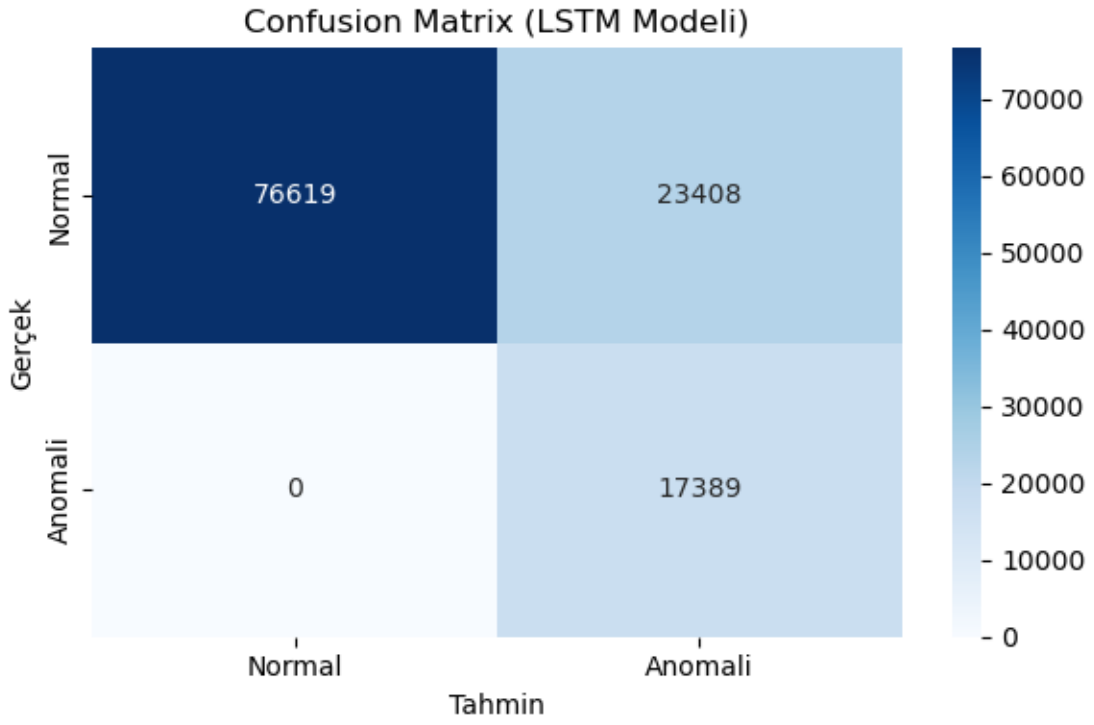


ROC Eğrisi (Receiver Operating Characteristic) X Ekseni (False Positive Rate - Yanlış alarm oranı): Ne kadar "yanlış yere alarm verdi" Y Ekseni (True Positive Rate - Doğru tespit oranı): Ne kadar "doğru alarm verdi" Eğri, sol üst köşeye ne kadar yakınsa model o kadar iyi

AUC (Area Under Curve) = 0.88 Bu ne demek? Modelin normal ile anomalileri ayırabilme yeteneği %88 oranında başarılı. AUC: 0.5: Rastgele seçim 0.7-0.8: İyi 0.8-0.9: Çok iyi (Benim Sonucum) 0.9+: Mükemmel (veya overfit riski olabilir)

Grafik Gözlemleri: Eğri hızlıca yukarı fırlamış → model saldırıları çabuk fark ediyor. 0.2'nin altında FPR varken TPR 1'e ulaşmış → erken ve doğru tespit yapıyor. ROC eğrisindeki keskin dönüşler → net bir karar sınırı var.

[6]:
```
from sklearn.metrics import confusion_matrix
import seaborn as sns

# Confusion matrix hesapla
cm = confusion_matrix(y_test, y_pred)

# Görselleştir
plt.figure(figsize=(6, 4))
sns.heatmap(cm, annot=True, fmt='d', cmap='Blues',
            xticklabels=['Normal', 'Anomali'],
            yticklabels=['Normal', 'Anomali'])

plt.xlabel('Tahmin')
plt.ylabel('Gerçek')
plt.title('Confusion Matrix (LSTM Modeli)')
plt.tight_layout()
plt.show()
```



1. Sıfır Kaçırma (FN = 0) Gerçek anomalilerin hiçbiri kaçırılmamış. Bu, saldırı ya da anormal davranışları yakalama açısından güzel.
2. False Positive Var (23.408 adet) Normal bazı davranışlar "anomali" zannedilmiş. Bu yanlış alarm demek ama anomali tespit sistemlerinde bu genellikle kabul edilebilir (özellikle güvenlikte: yanlış alarm » gözden kaçan saldırı).

3. Dengeli Dağılım ve Sağlam Öğrenme True Positive ve True Negative değerlerin yüksekliği LSTM'in hem öğrendiğini hem de genelleme yapabildiğini gösteriyor.

Sonuç Olarak: Anomali kaçırmıyor. Alarm veriyor ama "önlem almak için" yeterince güvenli. AUC = 0.88 ve accuracy = 0.80 ile güzel bir sıralı veri modeli.

[ ]: