SSB K-Means Modeli - 1

May 8, 2025

[8]:
```
# Gerekli kütüphaneler
import pandas as pd                        # Veri işleme
import numpy as np                         # Sayısal işlemler
from sklearn.preprocessing import StandardScaler  # Verileri ölçeklemek için
from sklearn.cluster import KMeans         # K-Means algoritması
from sklearn.metrics import confusion_matrix, classification_report  #␣
 ↪Değerlendirme metrikleri
```

[9]:
```
# CSV dosyasını okuma
df = pd.read_csv("C:\\Users\\Lenovo\\Desktop\\OneDrive_2025-05-07\\TON_IoT␣
 ↪datasets\\Processed_datasets\\Processed_IoT_dataset\\IoT_Fridge.csv")

# Kullanılmayacak sütunları çıkarma (tarih, zaman gibi anlamlı olmayanlar)
df.drop(['date', 'time', 'type'], axis=1, inplace=True)

# Kategorik veriyi sayısala çevirme (örneğin: temp_condition → low/high)
df = pd.get_dummies(df, columns=['temp_condition'], drop_first=True)

# Giriş (X) ve hedef (y) verilerini ayırma
X = df.drop('label', axis=1)  # Giriş özellikleri
y = df['label']               # 0: normal, 1: anomali
```

[10]:
```
# K-Means uzaklık temelli çalıştığı için ölçekleme
scaler = StandardScaler()
X_scaled = scaler.fit_transform(X)
```

[11]:
```
# K-Means modelini oluşturma (2 küme: normal ve anomali)
kmeans = KMeans(n_clusters=2, random_state=42)
kmeans.fit(X_scaled)  # Modeli veriye uygula
```

```
C:\Users\Lenovo\anaconda3\Lib\site-packages\sklearn\cluster\_kmeans.py:870:
FutureWarning: The default value of `n_init` will change from 10 to 'auto' in
1.4. Set the value of `n_init` explicitly to suppress the warning
  warnings.warn(
```

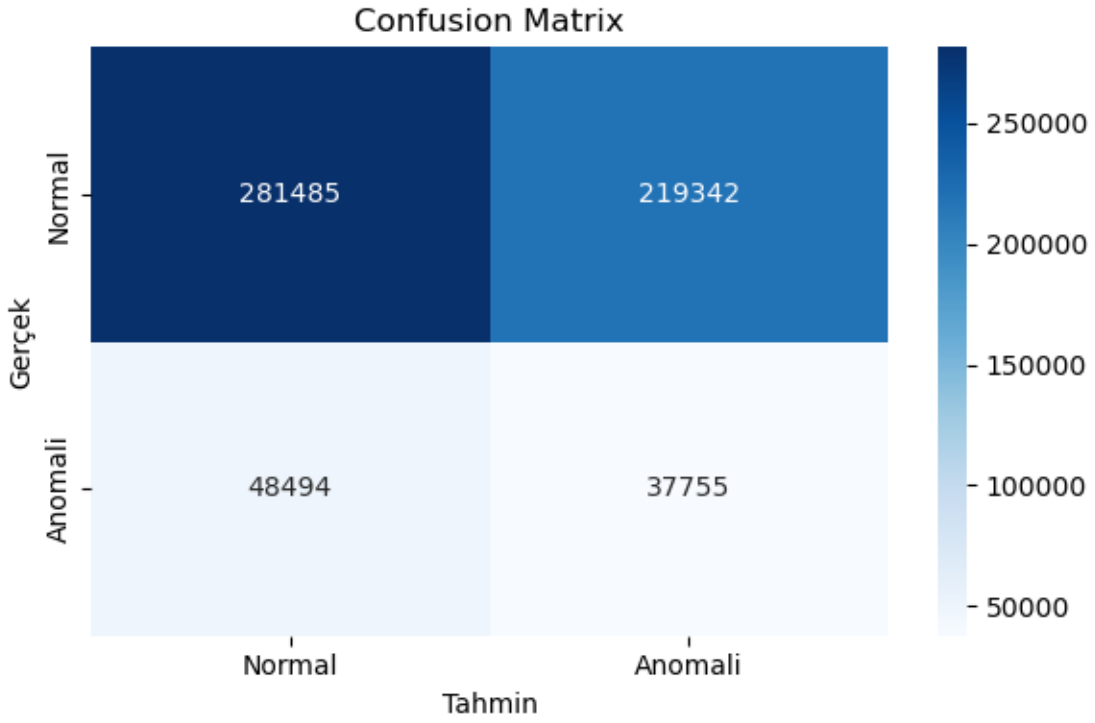
[11]:
```
KMeans(n_clusters=2, random_state=42)
```

[12]:
```
from sklearn.metrics import confusion_matrix, classification_report

# Tahminleri al
y_pred = kmeans.labels_

# Küme etiketleri doğru eşleşmeyebilir, tersleyerek karşılaştır
cm1 = confusion_matrix(y, y_pred)
cm2 = confusion_matrix(y, 1 - y_pred)
if cm2[0,0] + cm2[1,1] > cm1[0,0] + cm1[1,1]:
    y_pred = 1 - y_pred

# Sonuçları yazdır
print("Confusion Matrix:")
print(confusion_matrix(y, y_pred))

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

```
Confusion Matrix:
[[281485 219342]
 [ 48494  37755]]

Classification Report:
              precision    recall  f1-score   support

           0       0.85      0.56      0.68    500827
           1       0.15      0.44      0.22     86249

    accuracy                           0.54    587076
   macro avg       0.50      0.50      0.45    587076
weighted avg       0.75      0.54      0.61    587076
```

[13]:
```
import matplotlib.pyplot as plt
import seaborn as sns
from sklearn.metrics import confusion_matrix

# Confusion matrix hesapla
cm = confusion_matrix(y, 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')
```

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



1. Confusion Matrix Yorumu Tahmin: Normal Tahmin: Anomali Gerçek: Normal 281.485 219.342 Gerçek: Anomali 48.494 37.755

Yorum: True Negative (281k): Normal verileri doğru tanımış → güzel!

True Positive (37k): Anomaliyi doğru tespit etmiş.

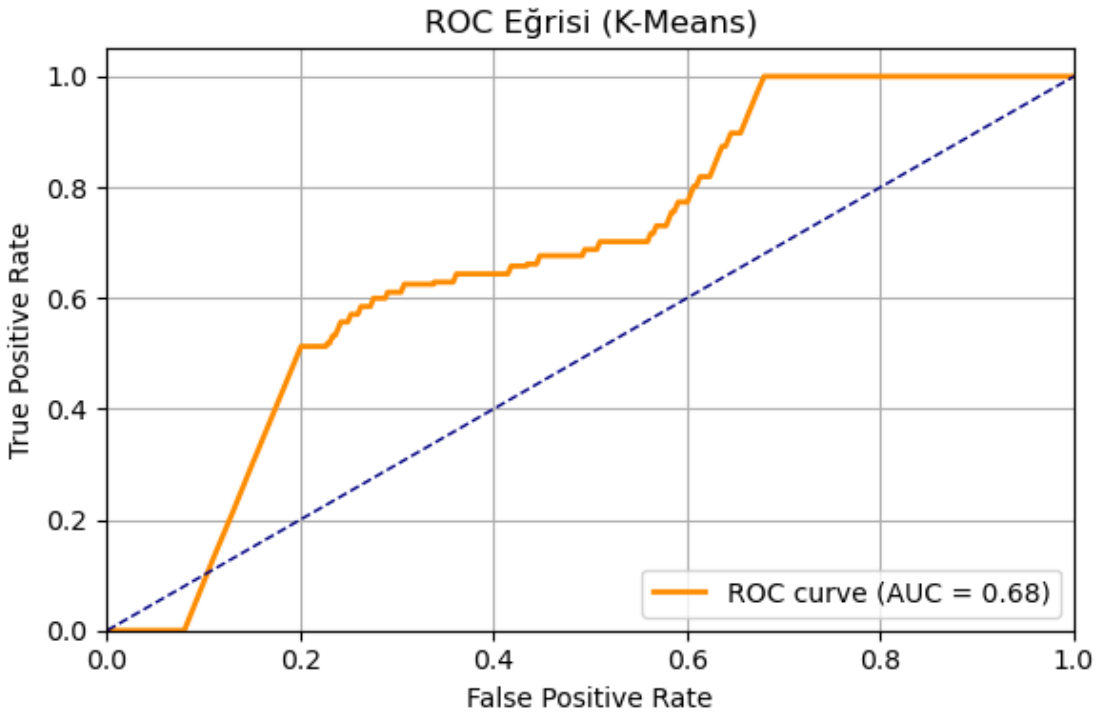
False Positive (219k): Normal verileri anomali sanmış → bu biraz fazla

False Negative (48k): Anomalileri gözden kaçırmış.

Genel: Model anomalileri tanıma konusunda idare eder ama normal verilerde çok fazla hata yapıyor. F1-score ve precision değerleri düşük olabilir ama bu durum K-Means gibi denetimsiz (unsupervised) modeller için normaldir.

[14]:
```
from sklearn.metrics import roc_curve, auc

# ROC eğrisi için: y_pred label değil, puan olmalı.
# KMeans'te puan yerine "cluster center uzaklığı" kullanabiliriz
# Anomali olasılığı: merkezlere uzaklık (anormal veriler daha uzakta olur)

# Her örnek için en yakın merkeze olan mesafeyi al
distances = kmeans.transform(X_scaled).min(axis=1)
```

```python
# ROC eğrisi
fpr, tpr, thresholds = roc_curve(y, distances)
roc_auc = auc(fpr, tpr)

# Çizim
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.xlim([0.0, 1.0])
plt.ylim([0.0, 1.05])
plt.xlabel('False Positive Rate')
plt.ylabel('True Positive Rate')
plt.title('ROC Eğrisi (K-Means)')
plt.legend(loc="lower right")
plt.grid(True)
plt.tight_layout()
plt.show()
```



2. ROC Eğrisi Yorumu AUC (Area Under Curve) = 0.68 Bu değer, modelin normal ve anormal verileri ayırt etme becerisini gösterir.

Ne Anlama Geliyor? 0.5 = şans başarısı,

0.68 = ortalamanın üstü, ayırt etme gücü var ama sınırlı,

0.80 olsaydı çok iyi olurdu.

Modelin karar verirken mesafe temelli ayrımı fena değil ama daha güçlü bir modelle (örneğin Isolation Forest ya da Autoencoder) daha iyi sonuç alınabilir.

Sonuç ve Öneri Güçlü Yanı İyileştirme Gereken Yanı Basit, etiket gerektirmiyor Anomaliyi net ayırt edemiyor ROC eğrisi 0.68 → ortalamanın üstü Normal verilerde çok fazla hata yapıyor

[ ]: