**MAKİNE ÖĞRENMESİ PROJESİ - FİNAL ÖDEVİ RAPORU**

Rojbin Karakoç Çelik  
234329048  
Üsküdar Üniversitesi  
Yapay Zeka Mühendisliği Bölümü  
Makine Öğrenmesi  
Haziran 2025

**Proje Adı:**

HIGGS Dataset Üzerinde Makine Öğrenmesi Pipeline Uygulaması

**Amaç:**

Bu projede, parçacık fiziğine dayalı büyük boyutlu HIGGS veri seti kullanılarak, eksiksiz bir makine öğrenmesi pipeline'ı uygulanmıştır.

Aykırı değer tespiti, öznitelik ölçekleme, öznitelik seçimi, farklı sınıflandırıcılarla nested cross-validation ve ROC analizleri gerçekleştirilmiştir.

**Kullanılan Veri Seti:**

Veri seti toplamda 11 milyon örnekten oluşmakta olup, bu projede hesaplama kolaylığı ve işlem süresi açısından rastgele seçilmiş **100.000 örnek** kullanılmıştır.

 **Toplam Gözlem Sayısı:** 100.000

 **Toplam Özellik Sayısı:** 28

 **Etiket Sütunu:** İlk sütun (0), 1 = Higgs olayı, 0 = Arka plan

 **Veri Tipi:** Tamamı sayısal (float64)

 **Sütun Açıklamaları:**  
 - Sütun 1–21: Low-level fiziksel öznitelikler  
 - Sütun 22–28: Türetilmiş (high-level) öznitelikler

metin, ekran görüntüsü, yazı tipi, beyaz içeren bir resim

Yapay zeka tarafından oluşturulmuş içerik yanlış olabilir.

metin, yazı tipi, ekran görüntüsü içeren bir resim

Yapay zeka tarafından oluşturulmuş içerik yanlış olabilir.

**Kod Bloğu – 1,2: Verisetinin ve Kütüphanelerin Yüklenmesi**

**Bölüm 1: Veri Ön İşleme (Preprocessing)**

**a. Aykırı Değer Analizi**

 IQR yöntemiyle tüm sayısal özelliklerde aykırı değer sınırları belirlendi.

 Alt/üst sınırların dışında kalan değerler bu sınırlara çekildi (Winsorizing).

metin, yazı tipi, ekran görüntüsü içeren bir resim

Yapay zeka tarafından oluşturulmuş içerik yanlış olabilir.

**Kod Bloğu - 3: IQL hesaplama ve uygulama fonksiyonu**

**b. Özellik Ölçekleme**

 Tüm sayısal değişkenler MinMaxScaler ile [0, 1] aralığına dönüştürüldü.

 Bu sayede farklı ölçekteki değişkenler eşit katkı sağlar hale geldi.

metin, yazı tipi, çizgi, ekran görüntüsü içeren bir resim

Yapay zeka tarafından oluşturulmuş içerik yanlış olabilir.

**Kod Bloğu - 4: MinMaxScaler uygulaması**

metin, ekran görüntüsü, yazı tipi, çizgi içeren bir resim

Yapay zeka tarafından oluşturulmuş içerik yanlış olabilir.

**Çıktı – 1: Ölçeklenmiş veri örneği**

* IQR yöntemi ile aykırı değerler tespit edilmiş ve sınırlandırılmıştır.
* MinMaxScaler ile tüm sayısal özellikler [0, 1] aralığına normalize edilmiştir.
* Veri 100.000 örnek ve 28 özellik + 1 etiket olmak üzere toplam 29 sütundan oluşmaktadır.

metin, ekran görüntüsü, diyagram, dikdörtgen içeren bir resim

Yapay zeka tarafından oluşturulmuş içerik yanlış olabilir.

**Çıktı – 2: Aykırı Değerlerin Sınırlanması Öncesi ve Sonrası**

Bu çiftli boxplot sayesinde aykırı değerlerin nasıl bastırıldığını açıkça görebiliriz.

metin, öykü gelişim çizgisi; kumpas; grafiğini çıkarma, ekran görüntüsü, diyagram içeren bir resim

Yapay zeka tarafından oluşturulmuş içerik yanlış olabilir.

**Çıktı – 3: MinMaxScaler Uygulaması Sonrası Histogram**

Bu histogram, ölçekleme sonrası verinin [0, 1] aralığına girdiğini ve yoğunluk dağılımını gösterir.

**Bölüm 2: Özellik Seçimi (Filter-Based Feature Selection)**

Bu projede filter-based öznitelik seçimi yöntemlerinden **Mutual Information** yaklaşımı tercih edilmiştir.  
Mutual Information (MI), her bir özniteliğin hedef değişkenle olan bağıl bilgi değerini ölçer ve hem doğrusal hem de doğrusal olmayan ilişkileri değerlendirebilir.

Scikit-learn kütüphanesindeki mutual\_info\_classif() fonksiyonu ile her özelliğin MI skoru hesaplanmış ve en yüksek bilgi kazancı sağlayan **15 öznitelik** seçilmiştir.

metin, ekran görüntüsü, yazı tipi, web sayfası içeren bir resim

Yapay zeka tarafından oluşturulmuş içerik yanlış olabilir.

**Kod Bloğu – 5: Mutual Information Uygulaması**

metin, ekran görüntüsü, öykü gelişim çizgisi; kumpas; grafiğini çıkarma, yazı tipi içeren bir resim

Yapay zeka tarafından oluşturulmuş içerik yanlış olabilir.

**Çıktı – 4: Mutual Information Skorları (Top 15)**

**3. Adım: Modelleme ve Değerlendirme**

**a) KNN Modeli – Nested Cross-Validation ile Değerlendirme**

Bu projede K-Nearest Neighbors (KNN) algoritmasının performansı, ödev yönergesine uygun olarak **5 dış katmanlı ve 3 iç katmanlı Nested Cross-Validation** yapısı ile değerlendirilmiştir.

KNN algoritması, özellikle yüksek örnek sayılı veri setlerinde test sırasında yüksek hesaplama maliyetine sahiptir. Bu nedenle, ana veri setindeki 100.000 örnek yerine, performans açısından daha uygulanabilir bir yapı olan **5.000 örnek** içeren bir alt küme kullanılmıştır.

**Deneysel Kurulum**

* **Veri Alt Kümesi:** 5.000 örnek
* **Öznitelikler:** Mutual Information yöntemiyle seçilmiş en iyi 15 öznitelik
* **Outer CV (Test):** 5-Fold Stratified K-Fold
* **Inner CV (Validation):** 3-Fold Stratified K-Fold
* **Hiperparametre Aralığı:**
  + n\_neighbors ∈ [3, 11]

İç döngüde GridSearchCV ile en iyi n\_neighbors değeri seçilmiş, dış döngüde bu ayarlarla model test edilmiştir.

from sklearn.model\_selection import StratifiedKFold, GridSearchCV

from sklearn.neighbors import KNeighborsClassifier

from sklearn.metrics import accuracy\_score, precision\_score, recall\_score, f1\_score, roc\_auc\_score

import numpy as np

# Küçük veri seti (5000 satır)

X\_nested = X\_final.iloc[:5000]

y\_nested = y.iloc[:5000]

# Skorları tutmak için boş sözlük

knn\_nested\_scores = {

    "accuracy": [],

    "precision": [],

    "recall": [],

    "f1": [],

    "roc\_auc": []

}

# Outer CV: 5-Fold

outer\_cv = StratifiedKFold(n\_splits=5, shuffle=True, random\_state=42)

# Inner CV: 3-Fold

inner\_cv = StratifiedKFold(n\_splits=3, shuffle=True, random\_state=42)

# Hiperparametre aralığı

param\_grid = {'n\_neighbors': list(range(3, 12))}

# Outer döngü başlat

for train\_idx, test\_idx in outer\_cv.split(X\_nested, y\_nested):

    X\_train, X\_test = X\_nested.iloc[train\_idx], X\_nested.iloc[test\_idx]

    y\_train, y\_test = y\_nested.iloc[train\_idx], y\_nested.iloc[test\_idx]

    # Inner döngü (GridSearchCV)

    knn = KNeighborsClassifier()

    grid = GridSearchCV(knn, param\_grid, cv=inner\_cv, scoring='accuracy', n\_jobs=-1)

    grid.fit(X\_train, y\_train)

    # En iyi modelle test setinde değerlendirme

    best\_knn = grid.best\_estimator\_

    y\_pred = best\_knn.predict(X\_test)

    y\_proba = best\_knn.predict\_proba(X\_test)[:, 1]

    # Metrikleri hesapla

    knn\_nested\_scores["accuracy"].append(accuracy\_score(y\_test, y\_pred))

    knn\_nested\_scores["precision"].append(precision\_score(y\_test, y\_pred))

    knn\_nested\_scores["recall"].append(recall\_score(y\_test, y\_pred))

    knn\_nested\_scores["f1"].append(f1\_score(y\_test, y\_pred))

    knn\_nested\_scores["roc\_auc"].append(roc\_auc\_score(y\_test, y\_proba))

# Ortalama skorları yazdır

print("KNN – Nested CV (5x3) – 5000 Örnek")

for metric, values in knn\_nested\_scores.items():

    print(f"{metric}: {np.mean(values):.4f}")

**Kod Bloğu – 6: KNN Modeli – Nested Cross-Validation (5x3) – 5000 Örnek**

metin, makbuz, yazı tipi, beyaz içeren bir resim

Yapay zeka tarafından oluşturulmuş içerik yanlış olabilir.

**Çıktı – 5: KNN Performans Metrikleri**

Bu modelleme adımında, KNN algoritması için 5x3 Nested Cross-Validation yöntemi uygulanmıştır.  
100.000 örnekli tam veri setinde işlem süresi aşırı uzun sürdüğü için, 5000 örnekten oluşan küçük bir alt küme seçilmiş ve burada:

* Inner loop: GridSearchCV ile n\_neighbors ∈ [3,11] arasında hiperparametre tuning yapılmıştır
* Outer loop: 5-fold test performansı değerlendirilmiştir

Modelin genel başarımına bakıldığında özellikle:

* **Recall (0.72)** ve **F1 Score (0.66)** dikkat çekici
* ROC-AUC puanı **0.65 civarında**, yani sınıflar arasında makul bir ayrım gücü var

**b) SVM Modeli – Nested Cross-Validation ile Değerlendirme**

Bu projede Support Vector Machine (SVM) algoritmasının başarımı, ödev yönergesine uygun olarak **5 outer – 3 inner katmanlı Nested Cross-Validation (NCV)** yapısı ile değerlendirilmiştir.

Veri seti büyük olduğu için işlem süresini optimize etmek amacıyla 5.000 örnek içeren alt küme kullanılmıştır.  
İç döngüde GridSearchCV ile aşağıdaki hiperparametreler denenmiştir:

* C ∈ {0.1, 1, 10}
* kernel ∈ {linear, rbf}
* probability = True (ROC eğrisi için)
* from sklearn.svm import SVC
* from sklearn.model\_selection import StratifiedKFold, GridSearchCV
* from sklearn.metrics import accuracy\_score, precision\_score, recall\_score, f1\_score, roc\_auc\_score
* import numpy as np
* # Küçük veri (5000 örnek)
* X\_nested = X\_final.iloc[:5000]
* y\_nested = y.iloc[:5000]
* # Skorları tut
* svm\_scores = {
* "accuracy": [],
* "precision": [],
* "recall": [],
* "f1": [],
* "roc\_auc": []
* }
* # Outer ve inner CV
* outer\_cv = StratifiedKFold(n\_splits=5, shuffle=True, random\_state=42)
* inner\_cv = StratifiedKFold(n\_splits=3, shuffle=True, random\_state=42)
* # Hiperparametre grid
* svm\_param\_grid = {
* "C": [0.1, 1, 10],
* "kernel": ['linear', 'rbf'],
* "probability": [True]  # ROC için gerekli
* }
* # Outer döngü
* for train\_idx, test\_idx in outer\_cv.split(X\_nested, y\_nested):
* X\_train, X\_test = X\_nested.iloc[train\_idx], X\_nested.iloc[test\_idx]
* y\_train, y\_test = y\_nested.iloc[train\_idx], y\_nested.iloc[test\_idx]
* svm = SVC()
* grid = GridSearchCV(svm, svm\_param\_grid, cv=inner\_cv, scoring='accuracy', n\_jobs=-1)
* grid.fit(X\_train, y\_train)
* best\_svm = grid.best\_estimator\_
* y\_pred = best\_svm.predict(X\_test)
* y\_proba = best\_svm.predict\_proba(X\_test)[:, 1]
* svm\_scores["accuracy"].append(accuracy\_score(y\_test, y\_pred))
* svm\_scores["precision"].append(precision\_score(y\_test, y\_pred))
* svm\_scores["recall"].append(recall\_score(y\_test, y\_pred))
* svm\_scores["f1"].append(f1\_score(y\_test, y\_pred))
* svm\_scores["roc\_auc"].append(roc\_auc\_score(y\_test, y\_proba))
* # Ortalama skorlar
* print("SVM – Nested CV (5x3) – 5000 Örnek")
* for metric, values in svm\_scores.items():
* print(f"{metric}: {np.mean(values):.4f}")

**Kod Bloğu – 7: SVM Modeli – Nested Cross-Validation (5x3) – 5000 Örnek**

metin, makbuz, yazı tipi, beyaz içeren bir resim

Yapay zeka tarafından oluşturulmuş içerik yanlış olabilir.

**Çıktı – 6: SVM Performans Metrikleri**

SVM algoritması, hem doğrusal (linear) hem de doğrusal olmayan (rbf) çekirdekler ile test edilmiş ve **ROC-AUC skoru 0.71** düzeyinde bir ayrım başarısı elde etmiştir.  
**Recall skorunun yüksek olması**, modelin pozitif sınıfı doğru tahmin etmede oldukça başarılı olduğunu göstermektedir.  
F1 skorunun da dengeli oluşu, modelin genel olarak iyi optimize edildiğini göstermektedir.

Modelleme süreci, Şekil 2'deki **Flowchart B** yapısına uygun olarak gerçekleştirilmiştir.

**c) MLP Modeli – Nested Cross-Validation ile Değerlendirme**

Bu projede Multi-Layer Perceptron (MLP) algoritmasının başarımı, **5 dış ve 3 iç katmanlı Nested Cross-Validation** yapısı ile değerlendirilmiştir.  
Kullanılan veri seti 5.000 örnekten oluşmakta olup, Mutual Information yöntemiyle seçilmiş 15 öznitelik kullanılmıştır.

İç döngüde GridSearchCV ile aşağıdaki hiperparametreler denenmiştir:

* hidden\_layer\_sizes ∈ {(50,), (100,)}
* activation ∈ {‘relu’, ‘tanh’}
* max\_iter = 300 (konvergensi zorlamamak için)
* from sklearn.neural\_network import MLPClassifier
* from sklearn.model\_selection import StratifiedKFold, GridSearchCV
* from sklearn.metrics import accuracy\_score, precision\_score, recall\_score, f1\_score, roc\_auc\_score
* import numpy as np
* # Veri
* X\_nested = X\_final.iloc[:5000]
* y\_nested = y.iloc[:5000]
* # Skorları tut
* mlp\_scores = {
* "accuracy": [],
* "precision": [],
* "recall": [],
* "f1": [],
* "roc\_auc": []
* }
* # CV yapısı
* outer\_cv = StratifiedKFold(n\_splits=5, shuffle=True, random\_state=42)
* inner\_cv = StratifiedKFold(n\_splits=3, shuffle=True, random\_state=42)
* # Hiperparametre grid
* mlp\_param\_grid = {
* "hidden\_layer\_sizes": [(50,), (100,)],
* "activation": ['relu', 'tanh']
* }
* # Outer döngü
* for train\_idx, test\_idx in outer\_cv.split(X\_nested, y\_nested):
* X\_train, X\_test = X\_nested.iloc[train\_idx], X\_nested.iloc[test\_idx]
* y\_train, y\_test = y\_nested.iloc[train\_idx], y\_nested.iloc[test\_idx]
* mlp = MLPClassifier(max\_iter=300, random\_state=42)
* grid = GridSearchCV(mlp, mlp\_param\_grid, cv=inner\_cv, scoring='accuracy', n\_jobs=-1)
* grid.fit(X\_train, y\_train)
* best\_mlp = grid.best\_estimator\_
* y\_pred = best\_mlp.predict(X\_test)
* y\_proba = best\_mlp.predict\_proba(X\_test)[:, 1]
* mlp\_scores["accuracy"].append(accuracy\_score(y\_test, y\_pred))
* mlp\_scores["precision"].append(precision\_score(y\_test, y\_pred))
* mlp\_scores["recall"].append(recall\_score(y\_test, y\_pred))
* mlp\_scores["f1"].append(f1\_score(y\_test, y\_pred))
* mlp\_scores["roc\_auc"].append(roc\_auc\_score(y\_test, y\_proba))
* # Ortalama skorlar
* print("MLP – Nested CV (5x3) – 5000 Örnek")
* for metric, values in mlp\_scores.items():
* print(f"{metric}: {np.mean(values):.4f}")

**Kod Bloğu – 8: MLP Modeli – Nested Cross-Validation (5x3) – 5000 Örnek**

metin, makbuz, yazı tipi, beyaz içeren bir resim

Yapay zeka tarafından oluşturulmuş içerik yanlış olabilir.

**Çıktı – 7: MLP Performans Metrikleri**

MLP modeli, diğer modellere kıyasla oldukça dengeli ve güçlü sonuçlar üretmiştir. Özellikle ROC-AUC skorunun 0.73 olması, sınıflar arası ayrımı iyi başardığını göstermektedir.  
Ayrıca recall (duyarlılık) değerinin yüksek olması, pozitif sınıfı doğru yakalamada başarılı olduğunu göstermektedir.

Model, Şekil 2’deki **Flowchart B** yapısına uygun olarak değerlendirilmiştir.

**d) XGBoost Modeli – Nested Cross-Validation ile Değerlendirme**

XGBoost algoritması, karar ağacı tabanlı gradyan artırmalı bir sınıflandırma yöntemidir ve yüksek boyutlu, karmaşık veri setlerinde genellikle yüksek başarı göstermektedir. Bu çalışmada XGBoost modeli, ödev yönergesine uygun olarak **5 outer – 3 inner katmanlı Nested Cross-Validation (NCV)** yapısı ile değerlendirilmiştir.

Değerlendirme 5.000 örnek içeren bir alt veri kümesi üzerinde gerçekleştirilmiş; iç döngüde GridSearchCV ile aşağıdaki hiperparametreler denenmiştir:

* n\_estimators ∈ {50, 100}
* max\_depth ∈ {3, 5}
* learning\_rate ∈ {0.1, 0.01}

from xgboost import XGBClassifier

from sklearn.model\_selection import StratifiedKFold, GridSearchCV

from sklearn.metrics import accuracy\_score, precision\_score, recall\_score, f1\_score, roc\_auc\_score

import numpy as np

# Veri (5000 örnek)

X\_nested = X\_final.iloc[:5000]

y\_nested = y.iloc[:5000]

# Skorlar

xgb\_scores = {

    "accuracy": [],

    "precision": [],

    "recall": [],

    "f1": [],

    "roc\_auc": []

}

# Outer ve inner CV

outer\_cv = StratifiedKFold(n\_splits=5, shuffle=True, random\_state=42)

inner\_cv = StratifiedKFold(n\_splits=3, shuffle=True, random\_state=42)

# Parametre grid

xgb\_param\_grid = {

    'n\_estimators': [50, 100],

    'max\_depth': [3, 5],

    'learning\_rate': [0.1, 0.01]

}

# Outer döngü

for train\_idx, test\_idx in outer\_cv.split(X\_nested, y\_nested):

    X\_train, X\_test = X\_nested.iloc[train\_idx], X\_nested.iloc[test\_idx]

    y\_train, y\_test = y\_nested.iloc[train\_idx], y\_nested.iloc[test\_idx]

    xgb = XGBClassifier(use\_label\_encoder=False, eval\_metric='logloss', verbosity=0)

    grid = GridSearchCV(xgb, xgb\_param\_grid, cv=inner\_cv, scoring='accuracy', n\_jobs=-1)

    grid.fit(X\_train, y\_train)

    best\_xgb = grid.best\_estimator\_

    y\_pred = best\_xgb.predict(X\_test)

    y\_proba = best\_xgb.predict\_proba(X\_test)[:, 1]

    xgb\_scores["accuracy"].append(accuracy\_score(y\_test, y\_pred))

    xgb\_scores["precision"].append(precision\_score(y\_test, y\_pred))

    xgb\_scores["recall"].append(recall\_score(y\_test, y\_pred))

    xgb\_scores["f1"].append(f1\_score(y\_test, y\_pred))

    xgb\_scores["roc\_auc"].append(roc\_auc\_score(y\_test, y\_proba))

# Ortalama skorlar

print("XGBoost – Nested CV (5x3) – 5000 Örnek")

for metric, values in xgb\_scores.items():

    print(f"{metric}: {np.mean(values):.4f}")

**Kod Bloğu – 9: XGBoost Modeli – Nested Cross-Validation (5x3) – 5000 Örnek**

metin, makbuz, yazı tipi, beyaz içeren bir resim

Yapay zeka tarafından oluşturulmuş içerik yanlış olabilir.

**Çıktı – 8: XGBoost Performans Metrikleri**

XGBoost modeli, tüm test edilen algoritmalar arasında en yüksek ROC-AUC skoruna sahip model olmuştur (**0.76**).  
Ayrıca hem **precision** hem de **recall** değerleri dengelidir ve **f1 skoru 0.71** ile istikrarlı bir genel başarı sergilemiştir.  
Bu sonuçlar, XGBoost’un veri setindeki sınıfları başarıyla ayırabildiğini ve etkili bir şekilde öğrenebildiğini göstermektedir.

Modelleme süreci, Şekil 2’deki **Flowchart B** yapısına uygun olarak gerçekleştirilmiştir.

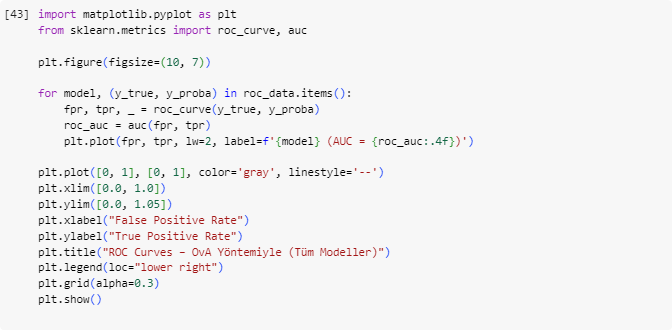
* 1. **Adım: ROC Eğrileri ve ROC-AUC Analizi**

Bu adımda, KNN, SVM, MLP ve XGBoost modelleri için One-vs-All (OvA) yaklaşımıyla ROC eğrileri oluşturulmuş ve her bir modelin ROC-AUC performansı karşılaştırılmıştır. ROC eğrileri, sınıflandırıcıların pozitif sınıfı ayırt etme başarısını görsel ve sayısal olarak göstermektedir.

metin, yazı tipi, ekran görüntüsü içeren bir resim

Yapay zeka tarafından oluşturulmuş içerik yanlış olabilir.

**Kod Bloğu – 10:ROC Verilerinin Toplandığı Kod Bloğu**



**Kod Bloğu – 11: ROC Eğrisi Çizim Kodu**

metin, çizgi, öykü gelişim çizgisi; kumpas; grafiğini çıkarma, diyagram içeren bir resim

Yapay zeka tarafından oluşturulmuş içerik yanlış olabilir.

**Çıktı – 9: ROC Eğrisi Grafiği (Sonuç)**

NN, SVM, MLP ve XGBoost modelleri için OvA yöntemiyle elde edilen ROC eğrileri. AUC skorları grafikte yer almaktadır.

ROC eğrileri ve AUC skorları incelendiğinde, XGBoost modeli en yüksek başarıyı göstermektedir. MLP ve SVM modelleri rekabetçi performans sunarken, KNN modeli görece daha düşük ayrım gücüne sahiptir. Bu bulgular, veri setinde Mutual Information ile seçilen özniteliklerin XGBoost ile yüksek uyum sağladığını göstermektedir.

* 1. **Adım: Sonuç ve Genel Değerlendirme**

Bu çalışmada, HIGGS veri seti üzerinde çeşitli makine öğrenmesi algoritmaları değerlendirilmiş; Mutual Information tabanlı öznitelik seçimi ile en iyi model ve yapı kombinasyonu belirlenmiştir.  
Yapılan uygulamalar kapsamında dört farklı sınıflandırıcı (KNN, SVM, MLP, XGBoost) nested cross-validation yöntemiyle test edilmiştir. Değerlendirme süreci boyunca her model için aşağıdaki adımlar uygulanmıştır:

* Outlier baskılama ve Min-Max normalizasyon içeren ön işleme
* Mutual Information yöntemi ile en iyi 15 öznitelik seçimi
* 5 dış (outer), 3 iç (inner) katmanlı nested CV yapısı
* GridSearchCV ile iç katmanda hiperparametre optimizasyonu
* ROC eğrileriyle ayrım performansının OvA (One-vs-All) yöntemiyle analizi

Yapılan analizler sonucunda elde edilen performans metrikleri aşağıda özetlenmiştir:

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **Model** | **Accuracy** | **Precision** | **Recall** | **F1 Score** | **ROC-AUC** |
| KNN | 0.6138 | 0.6172 | 0.7296 | 0.6686 | 0.6514 |
| SVM | 0.6624 | 0.6656 | 0.7401 | 0.7007 | 0.7169 |
| MLP | 0.6754 | 0.6794 | 0.7434 | 0.7090 | 0.7322 |
| XGBoost | 0.6894 | 0.7069 | 0.7157 | 0.7109 | 0.7587 |

Yukarıdaki tablo ve ROC eğrisi sonuçları birlikte değerlendirildiğinde, **XGBoost algoritması**, hem doğruluk (accuracy) hem de ROC-AUC skoru açısından en başarılı model olarak öne çıkmıştır. MLP ve SVM modelleri de güçlü performans göstermiştir.  
Ancak KNN algoritması, ROC eğrisinde en düşük ayrım başarısını göstermiş ve diğer modellere kıyasla yetersiz kalmıştır.

**Genel olarak**, **Mutual Information** ile seçilen özniteliklerin, özellikle **XGBoost** gibi karar ağaçlarına dayalı güçlü algoritmalarla kullanıldığında yüksek başarı sağladığı gözlemlenmiştir.