**MLP ve RBF Modellerinin Performans Analizi ve Değerlendirme Raporu**

**1. Gerekli Kütüphanelerin Yüklenmesi**

Bu çalışma 2 notebook kod dosyası üzerinde yapılmıştır. Veri seti Her iki notebook’ta da yapay sinir ağlarının eğitimi, veri görselleştirme ve ön işleme için kapsamlı Python kütüphaneleri kullanılmıştır. Kütüphanelerin detaylı işlevleri aşağıda açıklanmıştır:

* **Numpy ve Pandas:**
  + numpy: Matematiksel hesaplamalar ve dizi işlemleri.
  + pandas: Veri manipülasyonu ve çerçeve yapıları ile çalışmak için.
* **Matplotlib ve Seaborn:**
  + Eğitim süreci ve sonuçlarını görselleştirmek için grafikler çizildi.
  + Eğitim ve doğrulama doğruluğu/zaman kayıplarının gösterimi.
* **Scikit-learn:**
  + Eğitim verilerinin bölünmesi ve metrik hesaplamaları (ör. doğruluk, precision-recall gibi).
* **TensorFlow ve Keras:**
  + Modellerin oluşturulması, eğitimi ve değerlendirilmesi.
  + RBF modelinin özel katman implementasyonu.

Kod Örneği (MLP'den):

*import numpy as np*

*import pandas as pd*

*import matplotlib.pyplot as plt*

*import tensorflow as tf*

*from tensorflow.keras.models import Sequential*

*from tensorflow.keras.layers import Dense, Flatten*

*from sklearn.model\_selection import train\_test\_split*

*from sklearn.metrics import classification\_report*

**2. Veri Seti Yükleme ve Ön İşleme**

Veri seti olarak **Fashion MNIST** kullanılmıştır. Bu veri seti, 10 farklı sınıfa ait moda ürünlerinden oluşan 28x28 piksel boyutunda gri tonlamalı görüntüler içerir.

**Ön İşleme Adımları:**

1. **Veri Seti Yükleme:**
   * TensorFlow'un dahili veri seti modülü kullanılarak veri seti indirildi.
   * Eğitim (60.000 örnek) ve test (10.000 örnek) olarak ikiye ayrıldı.
2. **Veri Normalizasyonu:**
   * Görüntü piksel değerleri 0 ile 255 arasında olduğundan standard scaler kullanılarak veriyi ortalaması 0, standart sapması 1 olacak şekilde ölçeklendirerek, özelliklerin aynı ölçektolması sağlandı.
   * Kod:

*scaler = StandardScaler()*

*x\_train\_scaled = scaler.fit\_transform(x\_train\_flat)*

*x\_test\_scaled = scaler.transform(x\_test\_flat)*

1. **Verilerin Yeniden Şekillendirilmesi:**
   * MLP modeli için girişler düzleştirildi (1D vektör haline getirildi).
   * RBF modeli için özel olarak çekirdek gereksinimlerine uygun bir format sağlandı.
2. **Etiket İşleme:**
   * Kategorik etiketler one-hot encoding yöntemi ile dönüştürüldü.

**3. Model Eğitimi ve Hiperparametre Denemeleri MLP Modeli**

**Model Yapısı:**

1. **Fonksiyon Tanımı**

* **create\_mlp\_model**: Çok katmanlı algılayıcı (MLP) modelini dinamik olarak oluşturan bir fonksiyon.
* layer\_sizes: Modeldeki her katmanın nöron sayısını belirten bir liste (örneğin, [64, 128, 256]).
* learning\_rate: Optimizasyon algoritmasının öğrenme hızı.
* activation: Katmanlarda kullanılacak aktivasyon fonksiyonu (örneğin, relu, sigmoid).
* solver: Modelin optimizasyon algoritmasını belirlemek için (örneğin, adam, sgd).
* alpha: L2 regularizasyonu için ceza terimini belirler (aşırı öğrenmeyi önlemek için kullanılır).

1. **Model Başlatma**

* *model = Sequential(),* TensorFlow'un Keras API'si kullanılarak sıralı bir model (Sequential) başlatılır.
* Modelde katmanlar sıralı bir yapı içinde eklenir.

1. **İlk Katman Eklenemesi**

* İlk Dense yani tam bağlantılı katman eklenir.

*model.add(Dense(*

*layer\_sizes[0], input\_dim=x\_train.shape[1],activation=activation, kernel\_regularizer='l2')*

*)*

1. Ek Katmanların Eklenmesi- Döngü

* Amaç birden fazla gizli katman eklemek

*for size in layer\_sizes[1:]:*

*model.add(Dense(size, activation=activation, kernel\_regularizer='l2'))*

* layer\_sizes listesindeki ikinci ve sonraki her öğe için yeni bir katman oluşturur.
* Her katmanda, belirtilen nöron sayısı (size) ve aktivasyon fonksiyonu kullanılır.

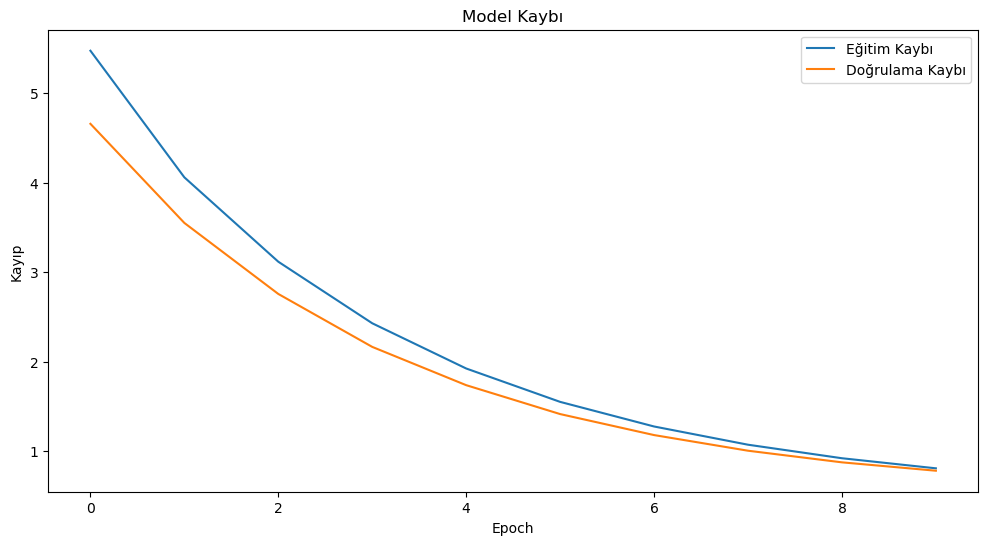
1. **Çıkış Katmanının Eklenmesi**
   * Çıkış katmanı, modelin tahmin ettiği sınıf sayısı kadar nöron içerir.
   * *y\_train.shape[1]:* Çıkış sınıfı sayısını belirler (örneğin, 10 sınıflı bir sınıflandırma problemi için 10).
   * *activation='softmax'*: Çok sınıflı bir problem için her sınıfın olasılığını hesaplar.

**Hiperparametre Denemeleri:**

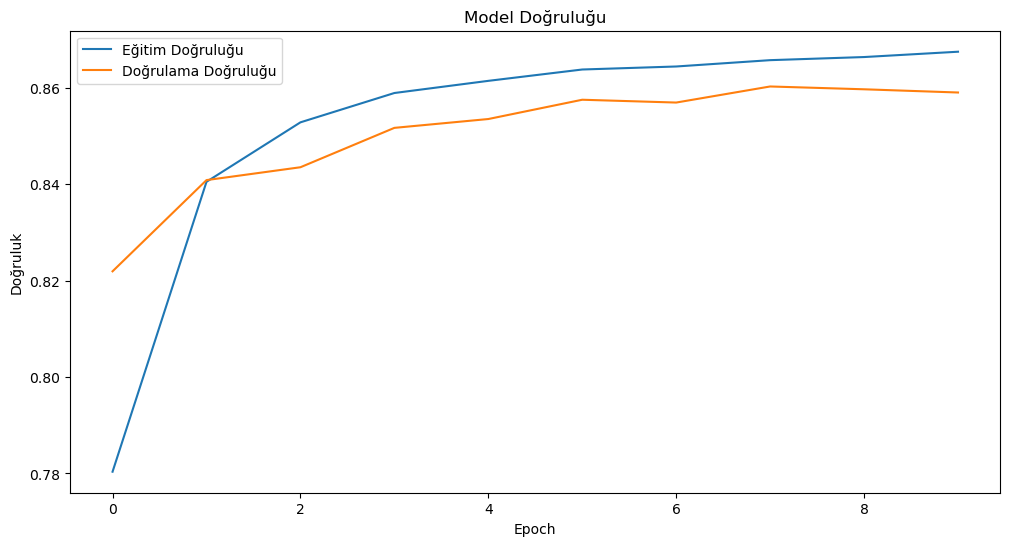
* + **Katman Sayısı ve Nöron Miktarı:** [[64], [128], [128, 64], [256, 128]] nöronlu farklı modeller test edildi.
  + **Öğrenme Oranı:** [0.001, 0.01] denenmiştir
  + **Aktivasyon fonksiyonu:** ['relu', 'tanh'] denenmiştir
  + **Optimizasyon algoritması:** ['adam', 'sgd'] denenmiştir
  + **Batch Size:** 64 üzerinden yapılmıştır.
  + **Alpha:** L2 regulürizasyon parametrei eklendi. Aşı öğrenmeyi önler.

**Sonuçlar:**

* Toplam 4x2x2x2x2 = 64 farlı değer denenmiştir. %72 ile %87 aralığında sonuçlar alındı.
* Eğitim doğruluğu: %87.
* Test doğruluğu: %86.
* MLP için en iyi parametreler: **([128], 0.01, 'relu', 'sgd', 0.001)** -> Acc: 0.8740



Model kaybı grafiği, eğitim ve doğrulama kayıplarının (loss) epoklara göre azalarak birbirine yakın seyrettiği durumlarda modelin iyi genelleştiğini gösterir.



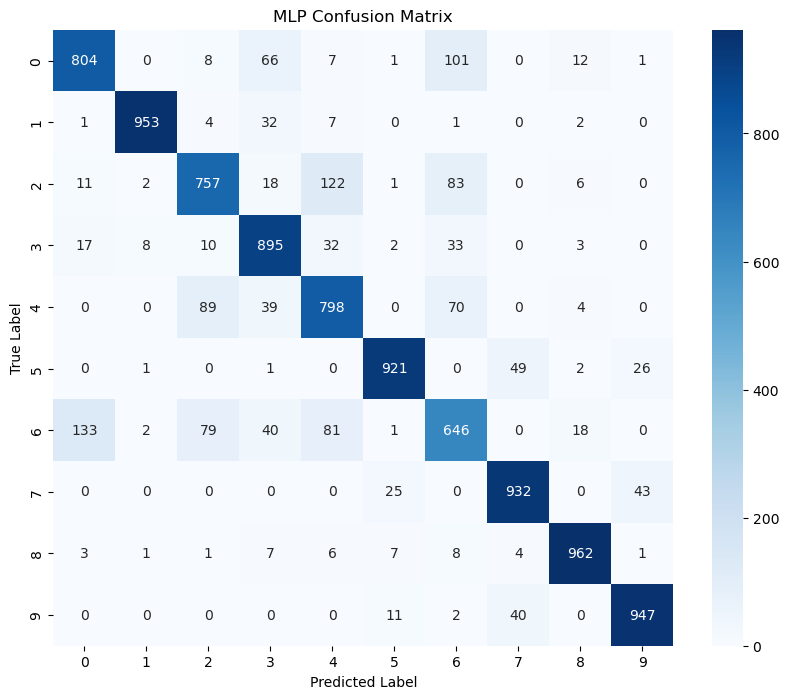
Model doğruluğu grafiği, eğitim ve doğrulama doğruluğunun epoklara göre artarak birbirine yakın seyrettiği durumlarda modelin iyi performans gösterdiğini ve aşırı öğrenme yapmadığını gösterir.

**MLP için Karmaşıklık Matrix yorumlanması**

* Confusion matrix, bir modelin tahminlerinin gerçek değerlerle karşılaştırmasını tablo halinde sunarak doğru ve yanlış sınıflandırmaların detayını gösterir.

A screenshot of a computer screen

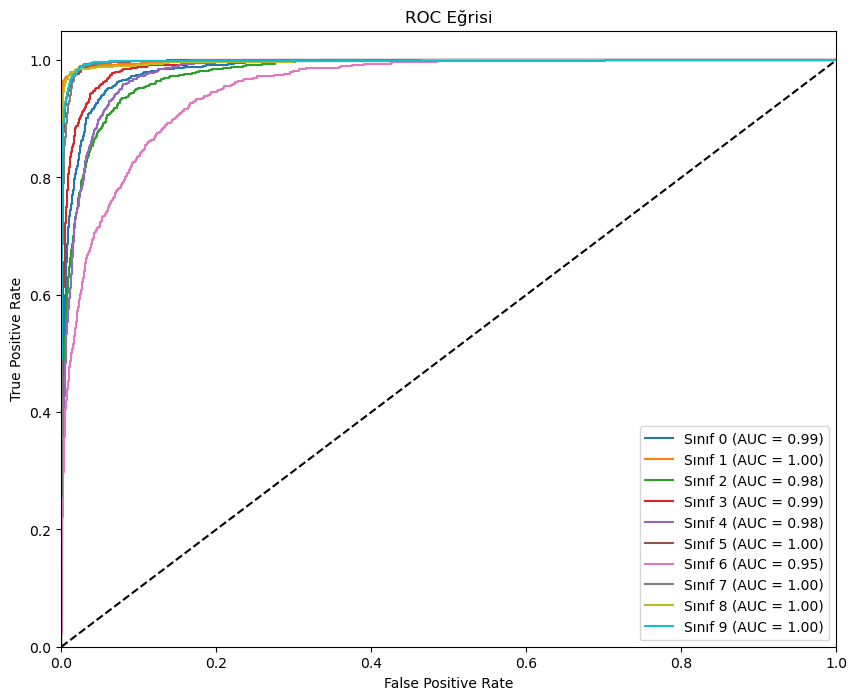
Description automatically generated



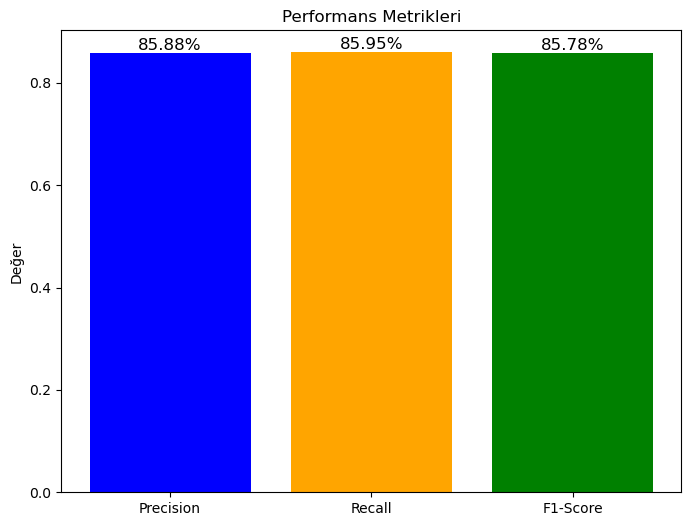
Modelimiz genel itibari ile güzel tahmin ederken özellikle 6. Sınıfın tahminini diğer sınıflara göre daha kötü tahminleme eğilimindedir. (modelimizin 6.sınıf olarak tahmin ettiği ama aslında 0.sınıfa ait olduğunu grafikten kolaylıkla yorumlayabiliriz. 133 değeri yanlış tahmin etmiş)

**MLP modeli ROC Eğrisi**

ROC eğrisi, bir modelin sınıflandırma yeteneğini her eşik değerinde değerlendirerek doğruluk ve yanılgı arasındaki dengeyi görselleştirmesi açısından kritik bir performans ölçütüdür.

****

**MLP modeli F1 skor, Precision ve Recall Performans Görselleştirme**

****

F1 skoru, precision (doğru tahmin edilen pozitiflerin oranı) ve recall (gerçek pozitiflerin doğru tahmin edilme oranı) arasındaki dengeyi ölçen, bu iki metriğin harmonik ortalamasıdır. Değerlendirme metriklerimizin skorları birbirlerine çok yakın çıktığı yorumunu yapabiliriz.

**RBF Modeli**

**Model Yapısı:**

* Özel RBF katmanı:
  + Gaussian çekirdek fonksiyonları.
  + Çekirdek merkezi sayısı (ör. 50, 100).
* Tam bağlantılı katmanlar:
  + Çıkışta sınıflandırma.

Kod:

class RBFLayer(Layer):

def \_\_init\_\_(self, num\_centers, sigma=1.0):

super(RBFLayer, self).\_\_init\_\_()

self.num\_centers = num\_centers

self.sigma = sigma

# RBF özel fonksiyonlar burada tanımlandı.

model = Sequential([

RBFLayer(100, sigma=2.0),

Dense(10, activation='softmax')

])

model.compile(optimizer='adam', loss='categorical\_crossentropy', metrics=['accuracy'])

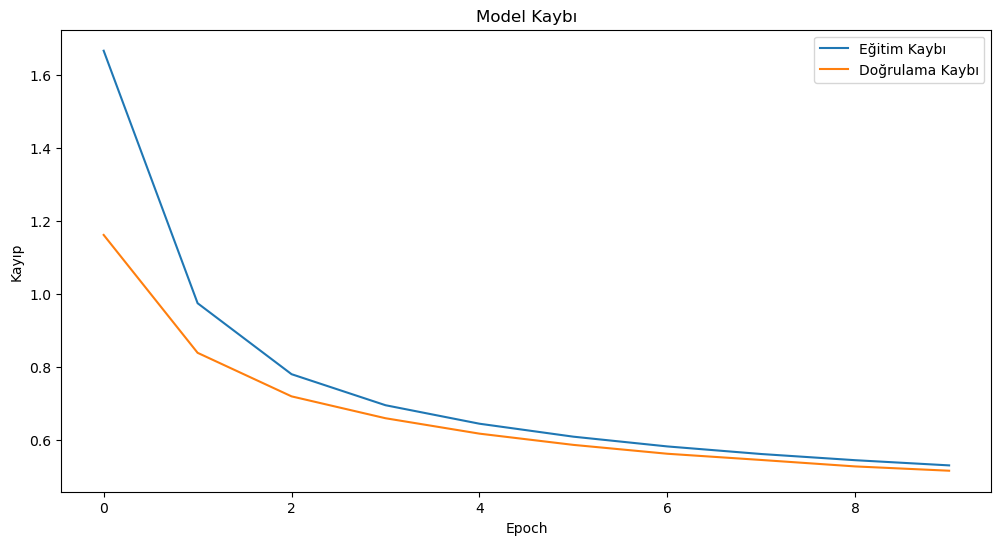
history = model.fit(x\_train, y\_train, validation\_split=0.2, epochs=20, batch\_size=64)

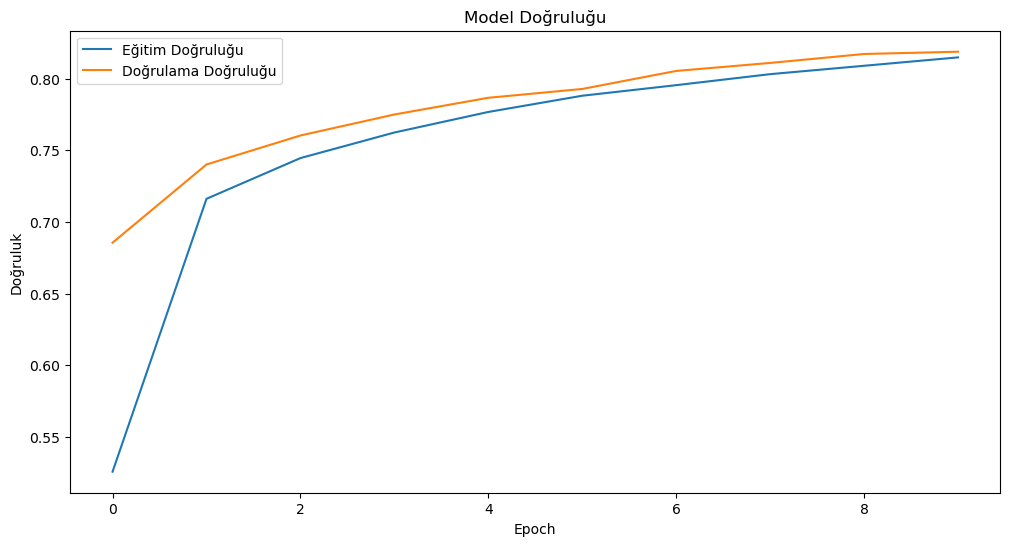
**Hiperparametre Denemeleri:**

* **Merkez Sayısı:** [50, 100, 200] değerleriyle deneme.
* **Sigma:** [0.01, 0.001] değerleriyle deneme.
* **Regularization Parametresi:** Çeşitli l2 değerleri test edildi.

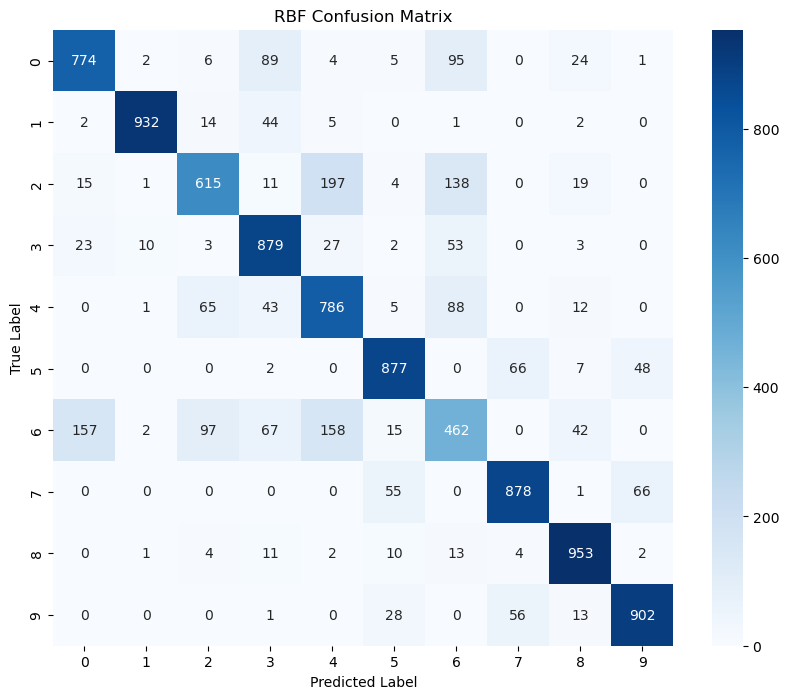
**Sonuçlar:**

* Toplam 3x2=6 farklı değer kombinasyonu denenmiştir. %61 ile %81 doğruluk oranı alındı.
* Eğitim doğruluğu %81
* Test doğruluğu: %79.
* RBF için en iyi parametreler: (200, 0.001) -> Acc: 0.8188
* Çekirdek sayısının artışı performansı artırmış ancak eğitim süresini uzatmıştır.





**RBF modeli** **Karmaşıklık Matrix yorumlanması**



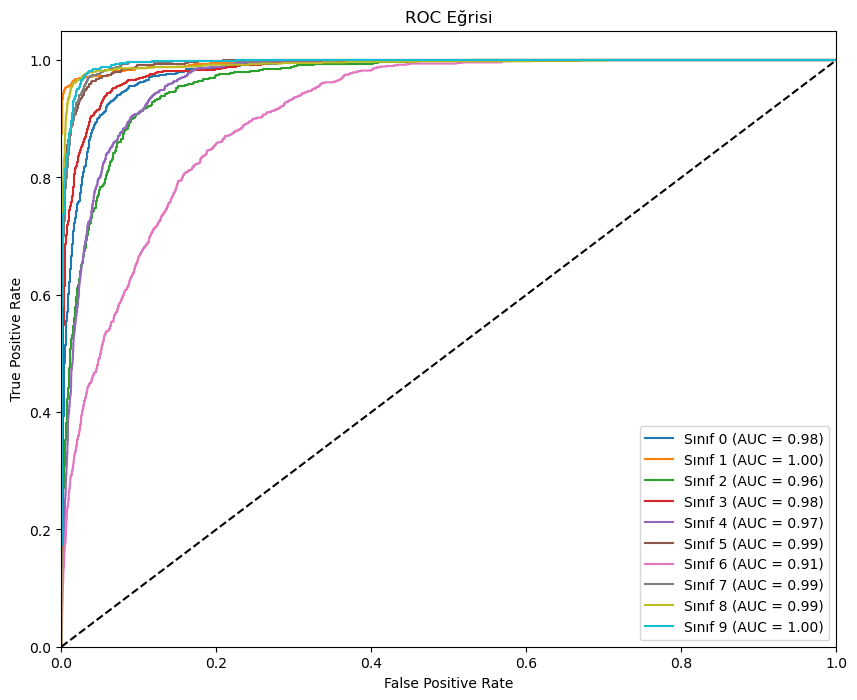
A screenshot of a computer screen

Description automatically generated

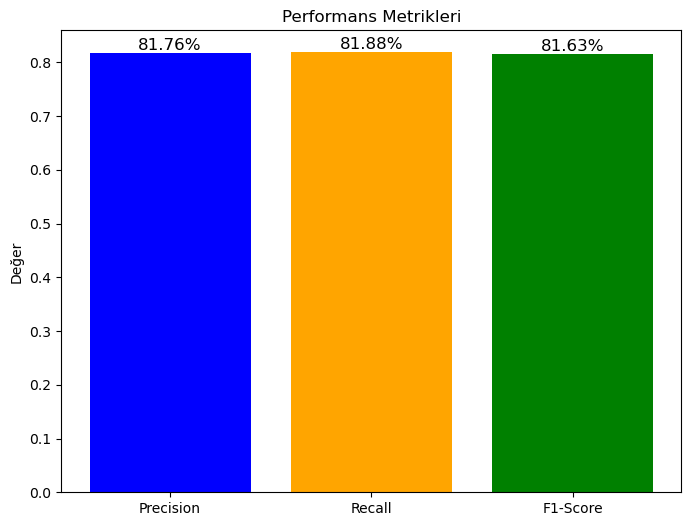
Modelimiz MLP modeli ile kıyaslandığında daha kötü tahminleme yaptığı gözlenmekte ama genel itibari ile güzel tahmin ederken özellikle 6. Sınıfın tahminini diğer sınıflara göre daha kötü tahminleme eğilimindedir. Bu durmu MLP modeli içinde gözlenmektedir (modelimizin 6.sınıf olarak tahmin ettiği ama aslında 0.sınıfa ait olduğunu grafikten kolaylıkla yorumlayabiliriz. 157 değeri yanlış tahmin etmiş MLP modelinde bu sayı 133’tü.)

**RBF modeli için ROC Curve**

ROC eğrisi, bir sınıflandırma modelinin tüm eşik değerlerinde doğru pozitif oranı (TPR) ile yanlış pozitif oranı (FPR) arasındaki ilişkiyi göstererek modelin ayırma yeteneğini görselleştirir. Ek olarak ROC eğrisi, ideal bir modelde sol üst köşeye (TPR=1, FPR=0) yakın seyreder; eğri ne kadar bu köşeye yakınsa modelin ayırma gücü o kadar yüksektir.



**RBF modeli için** **F1 skor, Precision ve Recall Performans Görselleştirme**



**4. Sonuç**

**MLP Modeli:**

* Daha hızlı eğitim süresi ve yüksek doğruluk oranı sağladı.
* Hiperparametre optimizasyonu ile **%87** eğitim doğruluğuna ulaşıldı.

**RBF Modeli:s**

* Verideki uzaysal ilişkileri modellemede güçlüdür.
* Performans açısından MLP’ye göre biraz geride kalmıştır **(%81** doğruluk).
* Özellikle çekirdek sayısının optimizasyonu önemlidir.

**Karşılaştırma:**

* MLP modeli genel doğrulukta daha iyi performans gösterirken, RBF modeli küçük veri kümelerinde esneklik sağlamıştır.

**Genel Çıkarımlar ve Öneriler**

1. **MLP Modelleri:**
   * Geniş çaplı veri kümelerinde daha iyi performans sağlar.
   * Basit yapı, hızlı ve verimlidir.
2. **RBF Modelleri:**
   * Karmaşık ve özelliklere dayalı ayrım gerektiren görevlerde avantajlıdır.
   * Çekirdek parametreleri doğru seçildiğinde güçlü bir alternatif olabilir.
3. **Gelecek Çalışmalar:**
   * RBF modeli için farklı çekirdek fonksiyonları (ör. polynomial, sigmoid) test edilebilir.
   * MLP modeline dropout gibi düzenleme yöntemleri eklenerek aşırı öğrenme önlenebilir.

**BEŞİR ARSLAN**

**234312026**