# BİR VERİ KÜMESİ ÜZERİNDE DESTEK VEKTÖR MAKİNELERİ VE YAPAY NÖRON AĞI ALGORİTMALARI ANALİZİ

Aleyna Nil Uzunoğlu

Uzay Yazılımları Mühendisi - Türk Havacılık ve Uzay Sanayii

aleynanil.uzunoglu@tai.com.tr

**Özet**

Bu analiz çalışmasında, bir veri kümesi üzerinde destek vektör makineleri ve yapay nöron ağı algoritmaları çalıştırılarak analiz yapılmıştır. Yapay nöron ağında farklı öğrenme yaklaşımları denenmiş, farklı katman sayılarında farklı öğrenme stilleri karşılaştırılmıştır. Destek vektör makinelerinde farklı kernel(çekirdek) fonksiyonları ile denemeler yapılmıştır. Tüm bu denemelerde en yüksek doğruluk elde eden modeller için kayıp değerlerinin grafikleri ve karmaşıklık matrisi (confusion matrix) verilmiş, bu sonuçlar üzerine yorumlar paylaşılmıştır.

**Anahtar Kelimeler**

Destek Vektör Makineleri, Yapay Sinir Ağları, Algoritma, Makine Öğrenmesi

**Giriş**

Farklı çözüm yöntemleriyle bir problemi ele almak, makine öğrenmesi alanında sıkça tercih edilen bir yaklaşımdır. Bu yöntem, çeşitli öğrenme algoritmalarının belirli veri türleri ve problemler üzerinde nasıl performans gösterdiğini değerlendirmek ve genellemek açısından önemli çıktılar sunar. Bu çalışmada, program akışı içerisinde sklearn kütüphanesi yardımı ile oluşturulan ve lineer olarak ayrılamayan bir veri kümesi üzerinde yapay sinir ağları(ANN) ve destek vektör makineleri (SVM) kullanılarak performans değerlendirmesi, karşılaştırması ve yorumlaması yapılacaktır.

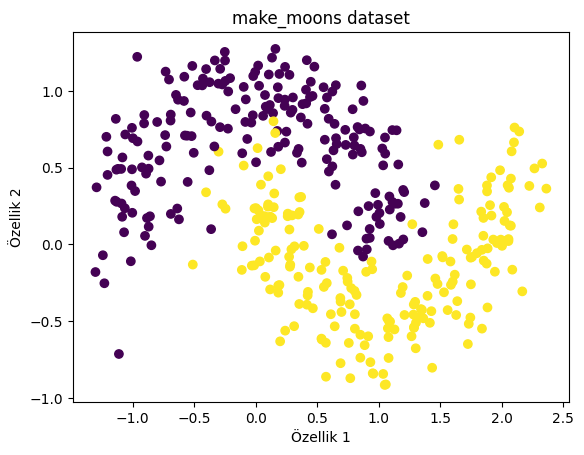
Çalışma kapsamında, veri kümesinin özellikleri detaylı şekilde analiz edilecek ve farklı parametrelerle her iki algoritmanın performansı değerlendirilecektir. Elde edilen sonuçlar doğrultusunda, doğruluk, kesinlik (precision), duyarlılık (recall) ve F1 skoru gibi çeşitli performans ölçütleri karşılaştırılarak, bu yöntemlerin avantaj ve dezavantajları ortaya konacaktır. Bu tür karşılaştırmalı yaklaşımlar, farklı problem türlerine uygun algoritmaların seçiminde rehber niteliği taşımaktadır.

**4) Metodoloji**

**4.1) Veri Kümesi**

Veri kümesi, program içerisinde scikit learn kütüphanesinin make\_moons fonksiyonu ile oluşturulmuştur. Bu fonksiyon, ismini oluşturduğu veri kümesinin iç içe geçmiş iki yarım ay şeklinde olmasından alır. Veri kümesinin lineer olarak ayrıştırılamaması nedeniyle, bu tür veriler üzerinde daha karmaşık sınıflandırıcıların kullanılması gerekliliğini ortaya koyar. Örneğin, bu tür veri kümeleri üzerinde lojistik regresyon gibi lineer sınıflandırıcılar yetersiz kalabilirken, destek vektör makineleri (SVM) gibi çekirdek yöntemleri ya da yapay sinir ağları gibi daha karmaşık modeller daha başarılı sonuçlar verebilir.

“make\_moons()” fonksiyonunun ürettiği veri kümesi, genellikle makine öğrenimi algoritmalarının doğruluğunu ve genelleme yeteneğini test etmek amacıyla kullanılır.



Veri kümesinde 400 adet veri bulunmaktadır. Veri oluşturulurken 0.2 oranla noise eklenmiştir. Bu ekleme verinin sınıflandırılmasını zorlaştırmak amacıyla yapılmıştır. Veri kümesi, istenildiği gibi, %60 eğitim, %20 doğrulama, %20 test oranında olmak üzere, scikit learn kütüphanesi yardımıyla rastgele bölünmüştür. Sonuç olarak, 240 adet eğitim verisi, 80 adet doğrulama verisi ve 80 adet test verisi bulunmaktadır.

**4.2) Yapay Nöron Ağları**

Yapay Nöron Ağları, insan beyninin nöral yapısından esinlenerek geliştirilmiş bir derin öğrenme teknolojisidir. Girdi katmanından, sayısı kullanıcı tarafından belirlenebilen gizli katmanlardan ve bir çıktı katmanından oluşur. Bu katmanlarda bulunan nöronlara girdi olarak bir veri birimi verilir. Her bir nöronun bir weight değeri bulunur.

Algoritmanın çalışmasında 2 yönlü bir yaklaşım izlenir. İleri yönlü iletim ve geri yönlü iletim.

İleri yönlü iletimde, nöronların ağırlık değerleri girdi değeri ile çarpılır ve belirli bir eşik değerinin üstünde ise o nöron ateşlenir. Giriş katmanından çıkış katmanına kadar bu şekilde ilerleyen işlemlerde bir kayıp (loss) hesaplanır. Geri yönlü iletimde, bu kayıp değerleri ve aktivasyon fonksiyonunun türevi kullanılarak ağırlıklar güncellenir. Eğitim bu ağırlıklarla yeniden gerçekleştirilir.

Son zamanlarda lineer yöntemlerle çözülemeyen problemleri oldukça efektif bir şekilde çözebilen bu algoritmanın kullanım alanı ve popülerliği zaman geçtikçe yaygınlaşmıştır.

Analizde yapay sinir ağı ile ilgili çalışmalar yapılmış, farklı katmanlarda farklı optimizasyon yaklaşımlarıyla denemeler gerçekleştirilmiştir. Bu yaklaşımın detayları ilerleyen bölümlerde anlatılacaktır.

**4.3) Destek Vektör Makineleri**

Destek Vektör Makineleri (SVM), hem sınıflandırma hem de regresyon problemlerinde kullanılan güçlü ve esnek bir makine öğrenmesi algoritmasıdır. Ancak genellikle sınıflandırma problemlerinde tercih edilir. Destek vektör makinelerinin temel amacı, farklı sınıfları en iyi şekilde ayıran bir **hiperdüzlem** bulmaktır.

Analizde destek vektör makineleri ile ilgili çalışmalar yapılmış, farklı kernel fonksiyonları ile denemeler gerçekleştirilmiştir.

#### **Kernel Fonksiyonları:**

Kernel fonksiyonları, SVM'nin doğrusal olmayan problemlerde verimli bir şekilde çalışmasını sağlar. En yaygın kullanılan kernel türleri:

* **Doğrusal Kernel (Linear Kernel):** Veri doğrusal olarak ayrılabiliyorsa kullanılır.
* **Polinomsal Kernel (Polynomial Kernel):** Veriyi polinomsal bir şekilde dönüştürür.
* **Gaussian RBF (Radial Basis Function) Kernel:** Verileri çok boyutlu bir Gaussian dağılımına göre dönüştürür ve en popüler kernel türüdür.

**5)Deneysel Sonuçlar ve Tartışma**

**5.1)Yapay Nöron Ağları**

İlk denemeler yapay nöron ağları ile yapılmıştır. Belirtilen eğitim verileriyle, Stochastic Gradient Descent yöntemiyle (batch size=1),

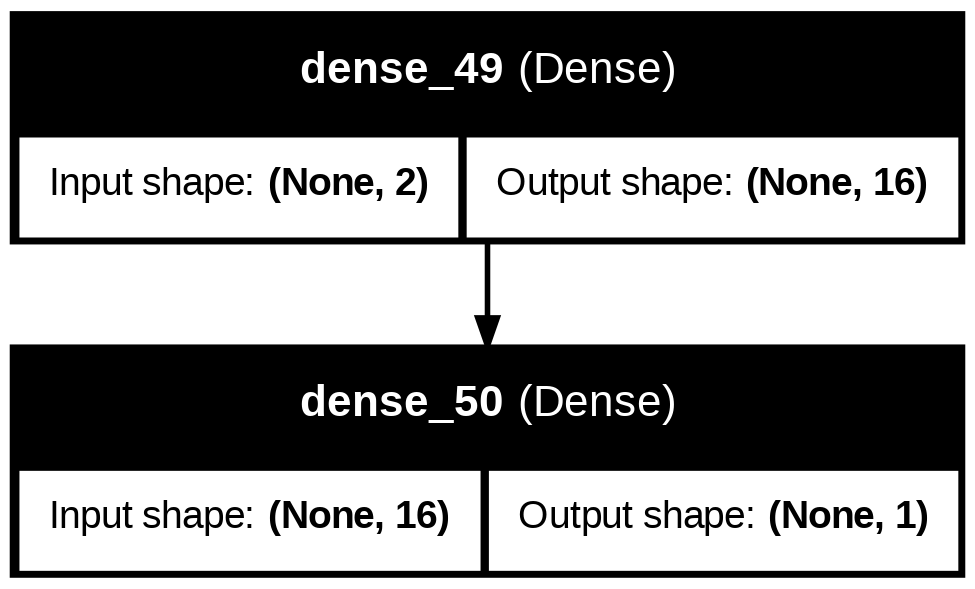
Mini-Batch Gradient Descent yöntemiyle(batch\_size=12),

Batch Gradient Descent(batch\_size=len(training\_data)) yöntemiyle eğitimler yapılmıştır. Bu yöntemler ayrı ayrı 1 katmanlı, 2 katmanlı ve 3 katmanlı ağlarda gerçekleştirilmiştir.

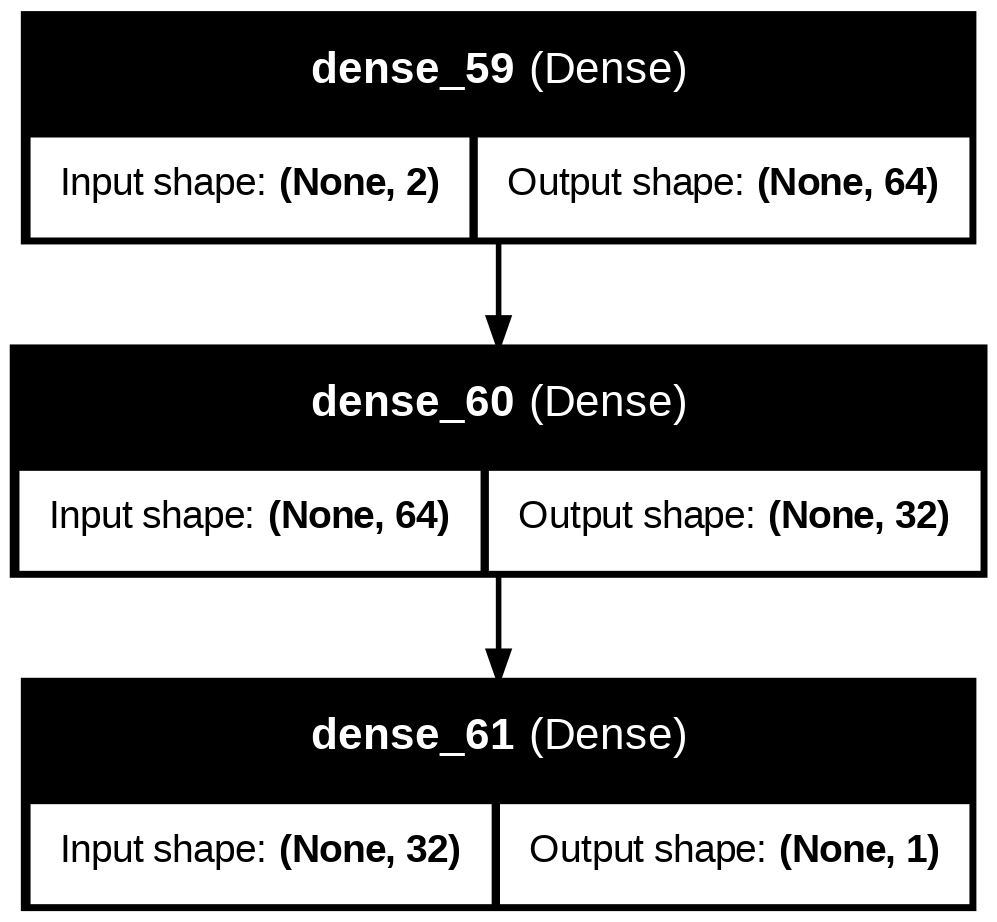
Farklı katmanlarda farklı optimizasyon yaklaşımlarıyla denemeler gerçekleştirilmiştir. Ödevde istenildiği gibi, loss fonksiyonu olarak Binary Cross Entropy kullanılmıştır. Kurulan ağ, ara katmanlarda aktivasyon fonksiyonu olarak “ReLU” fonksiyonu, çıktı katmanında aktivasyon fonksiyonu olarak “sigmoid” kullanır.

Tek katmanlı ağ için 16 nöronlu bir yapı, 2 katmanlı bir ağ için 1.katmanda 64 nöron 2.katmanda 32 nöron olacak şekilde bir yapı, 3 katmanlı bir ağ için 1.katmanda 32, 2.katmanda 64, 3.katmanda 32 olacak şekilde bir yapı kurulmuştur.

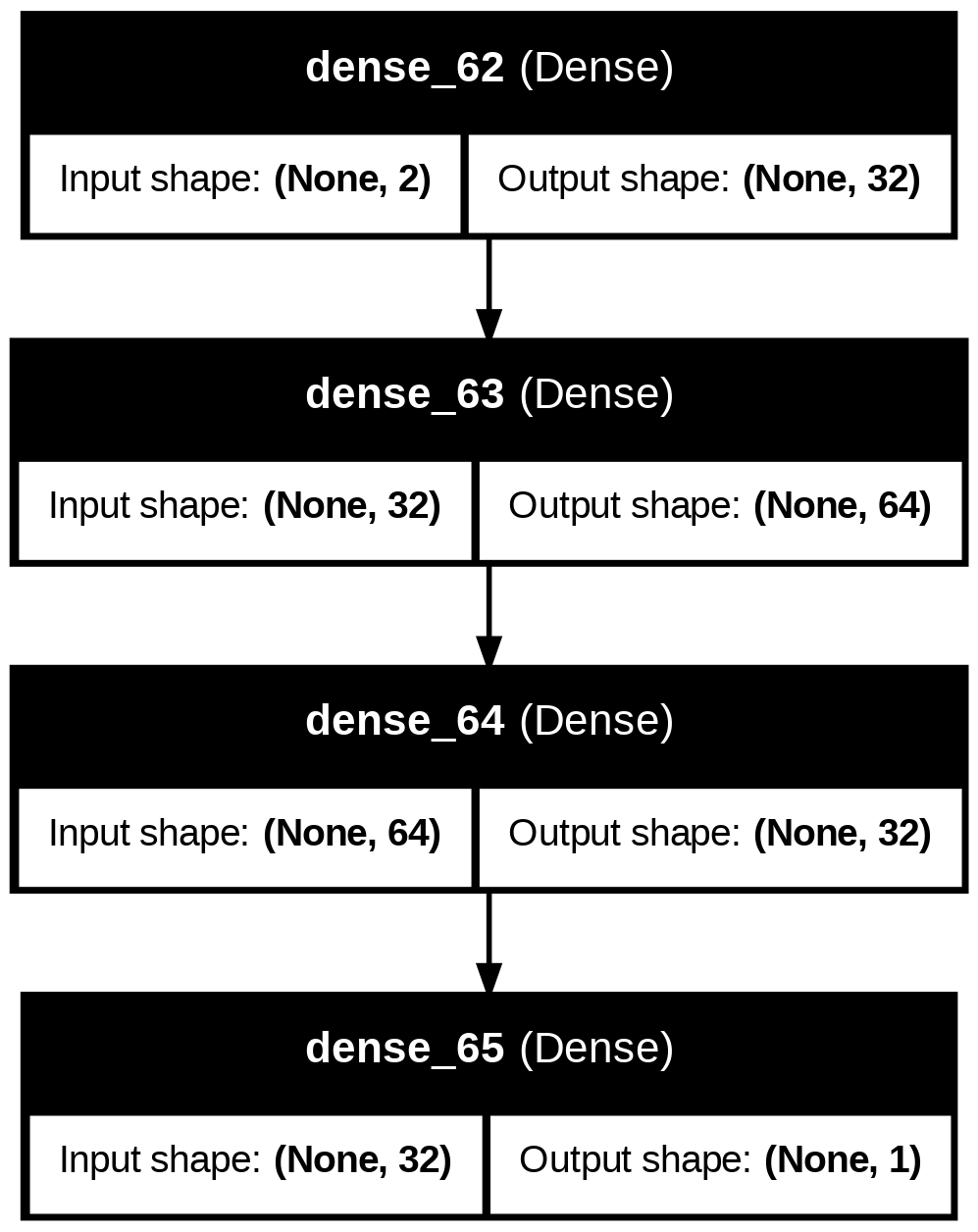
Bu ağların görselleştirilmesi aşağıdaki gibidir:



1 katmanlı yapay sinir ağı görselleştirmesi



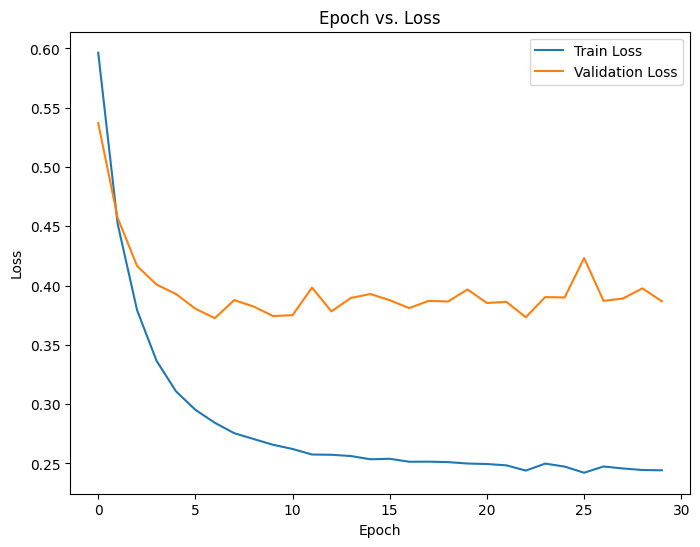
2 katmanlı yapay sinir ağı görselleştirmesi



3 Katmanlı Yapay Sinir Ağı Görselleştirmesi

**5.1.1) 1 Katmanlı Ağ**

**5.1.1.1) Stochastic Gradient Descent**



Grafik : SGD yöntemiyle eğitilen tek katmanlı ağın eğitim ve doğrulama verileri için epoch/loss grafiği

Accuracy: 0.8000

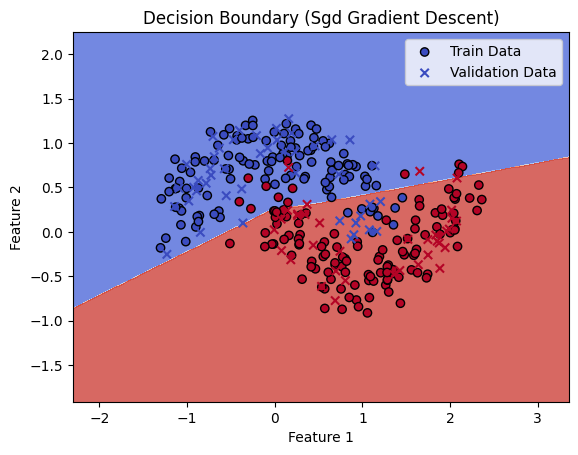
Loss: 0.4141

Precision: 0.7599

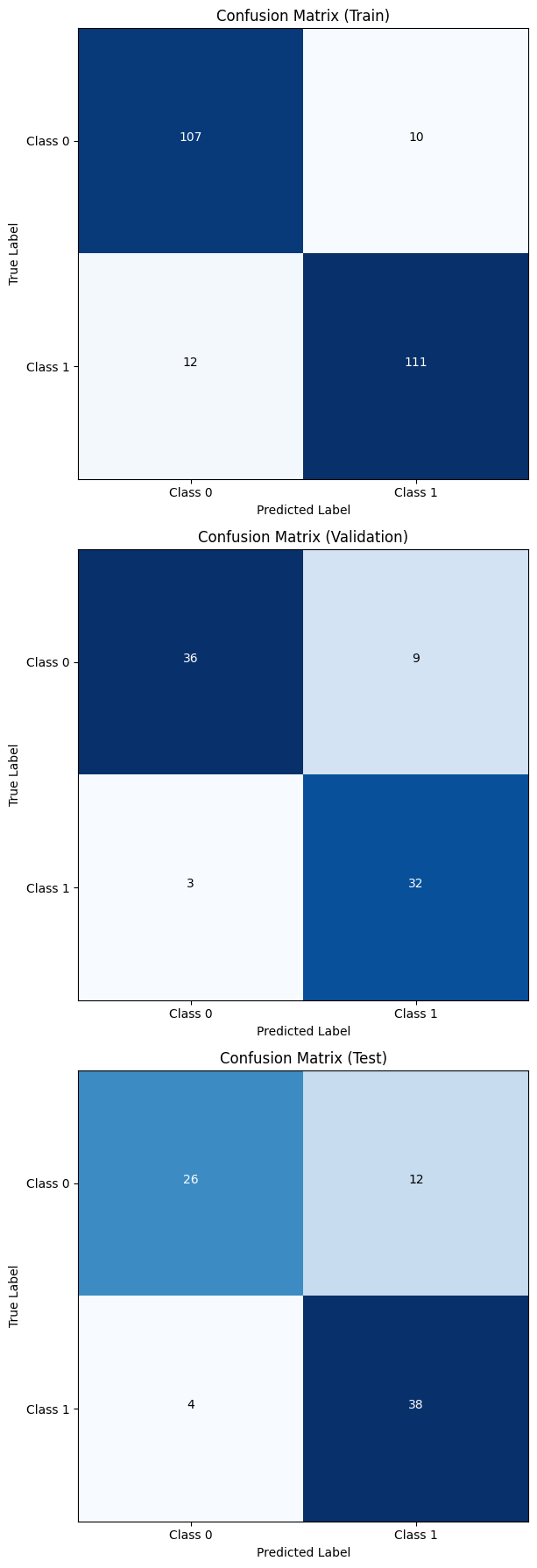
Recall: 0.9047

değerleri bu şekildedir.

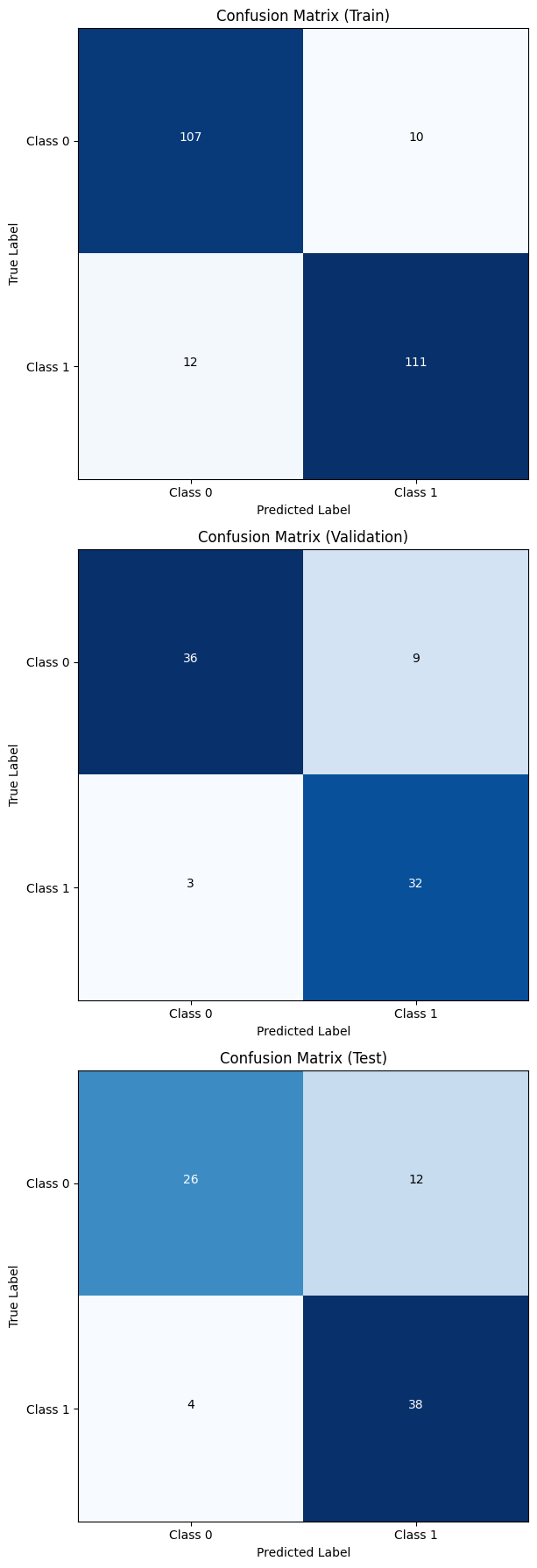
Karar sınırı grafiği aşağıda verilmiştir.



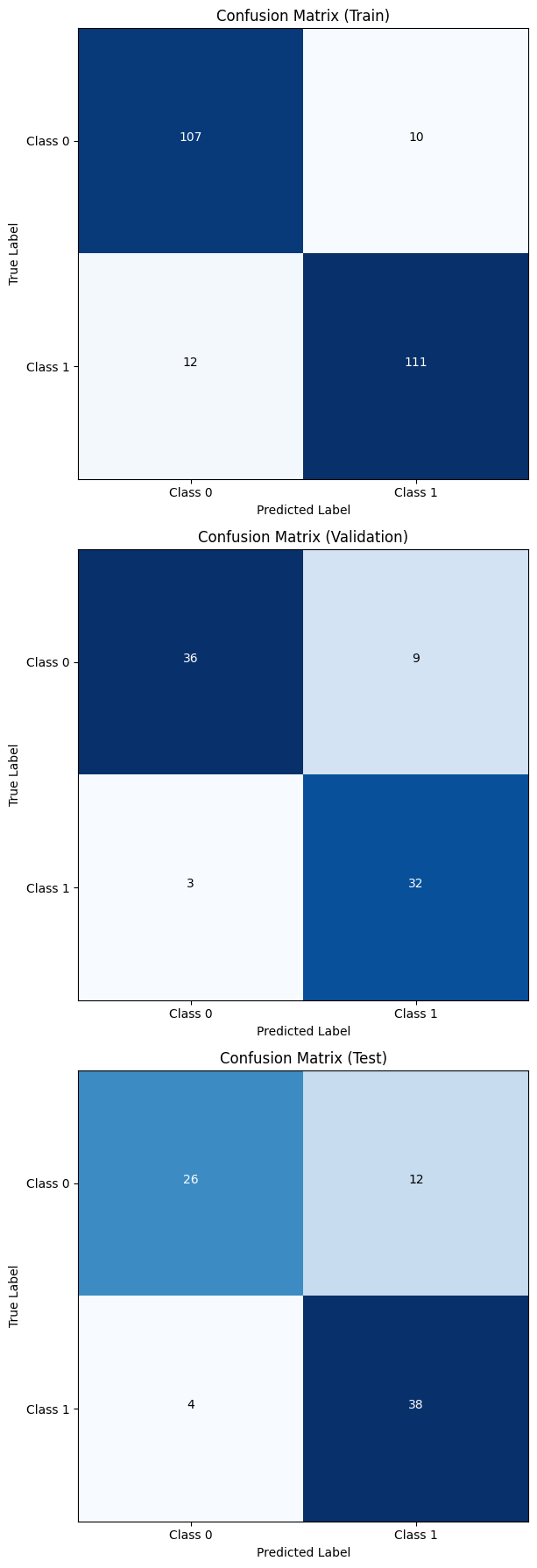
Grafik: SGD yöntemiyle eğitilen tek katmanlı ağın karar sınırı



Grafik: Eğitim verisinin karmaşıklık matrisi



Grafik: Doğrulama verisinin karmaşıklık matrisi



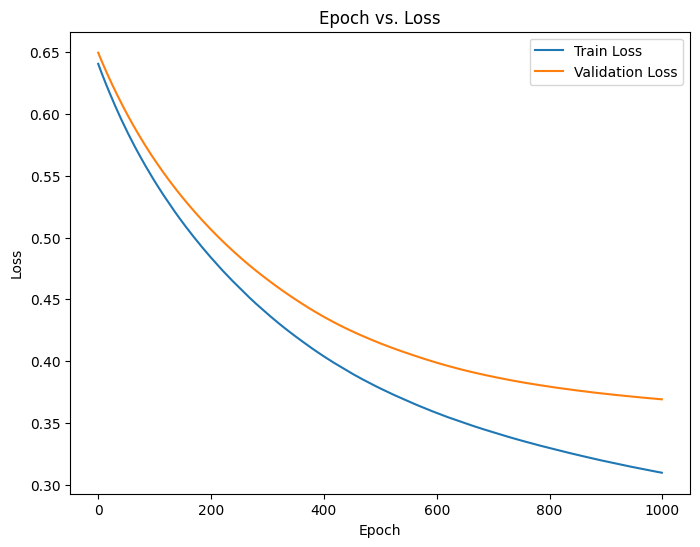
Grafik: Test verisinin karmaşıklık matrisi

SGD yaklaşımında, eğitilen her bir veride ağırlık güncellemesi gerçekleştiği için, öğrenmenin durdurulacağı noktaya daha erken ulaşıldığı görülmüştür.

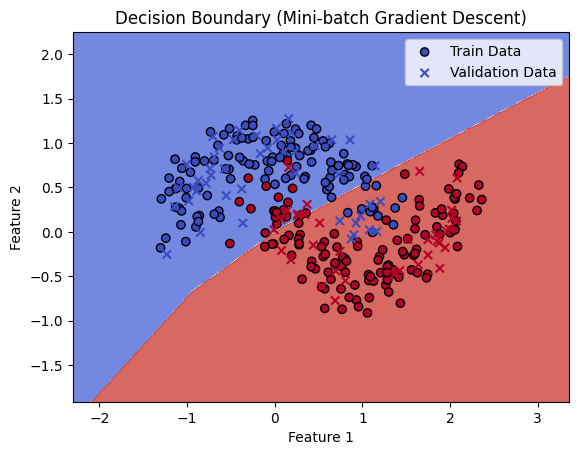
5.1.1.2) Mini Batch Gradient Descent

Mini-Batch yaklaşımı, SGD ve Batch GD arasında bir yöntemdir. Batch boyutunu kullanıcı belirler. Bu projede batch boyutu 128,64,32, 24,12 değerleri denenmiş ve en iyi performans gösteren batch boyutu 24 olduğu görülmüştür. Bundan dolayı bu batch boyutu ile ilerlenmiştir.

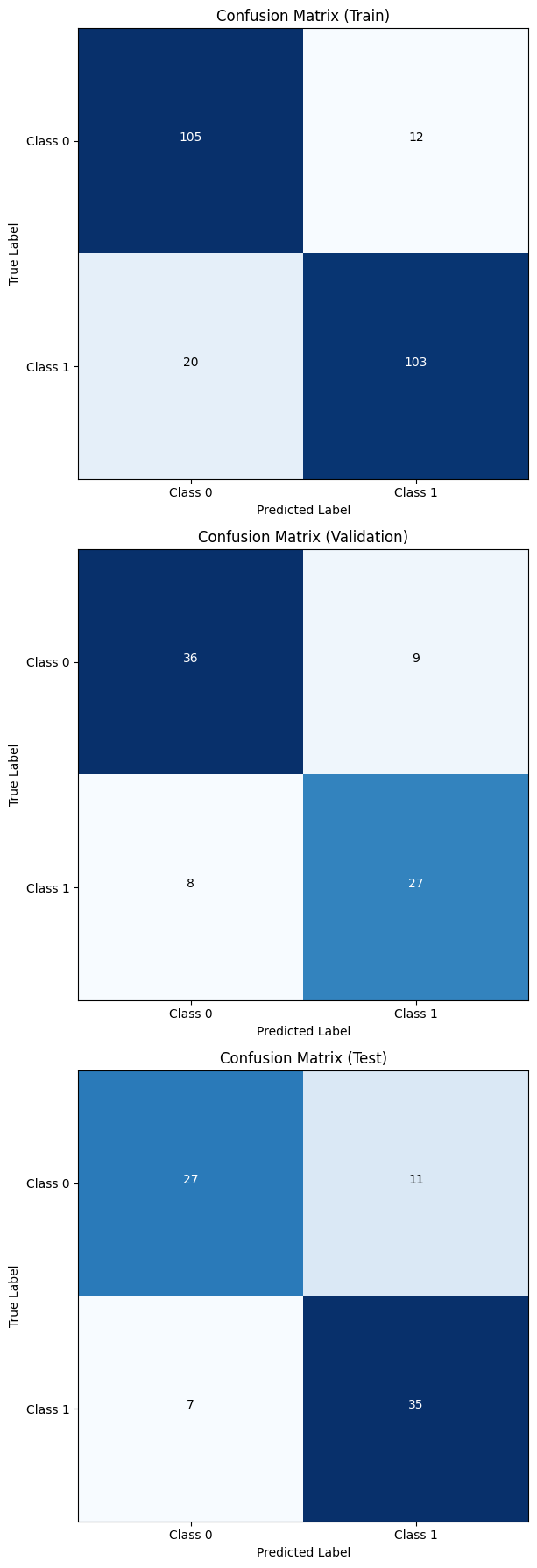
Accuracy değeri 0.7749, precision değeri 0.7608, recall değeri 0.8333 olarak bulunmuştur. Loss değeri 0.4196dır.

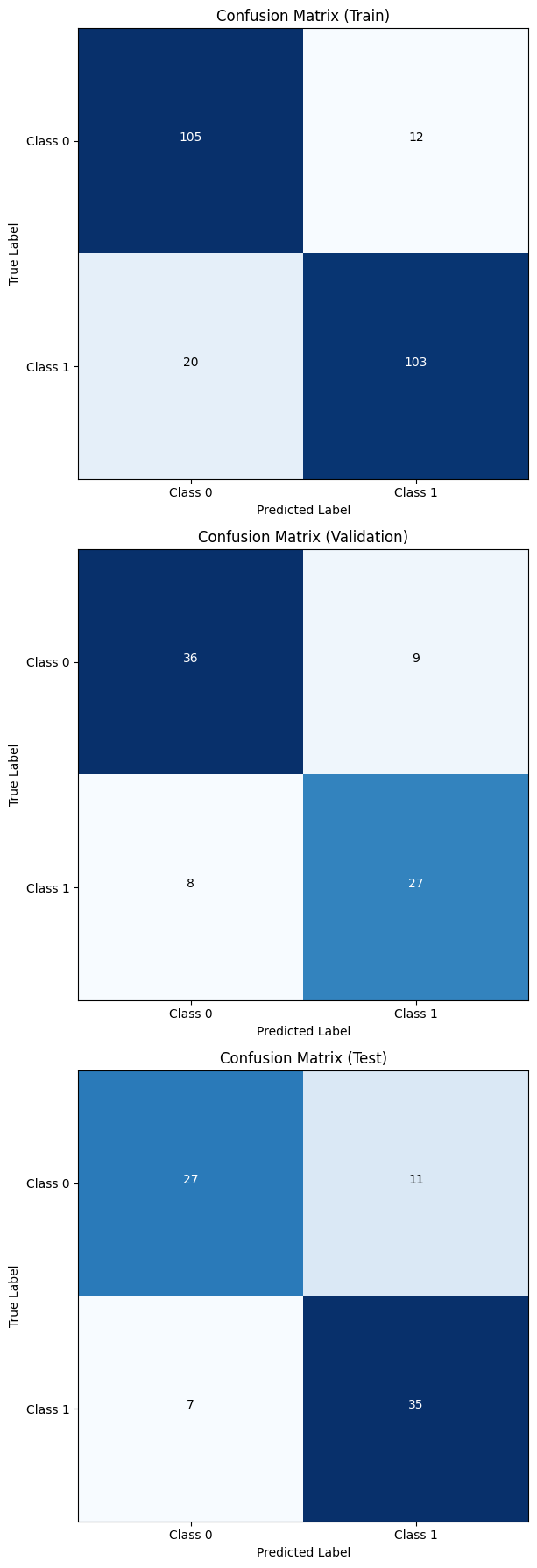


Grafik: 24 Batch Size ile Mini Batch yöntemiyle eğitilen tek katmanlı ağın eğitim ve doğrulama verileri için Epoch/loss grafiği

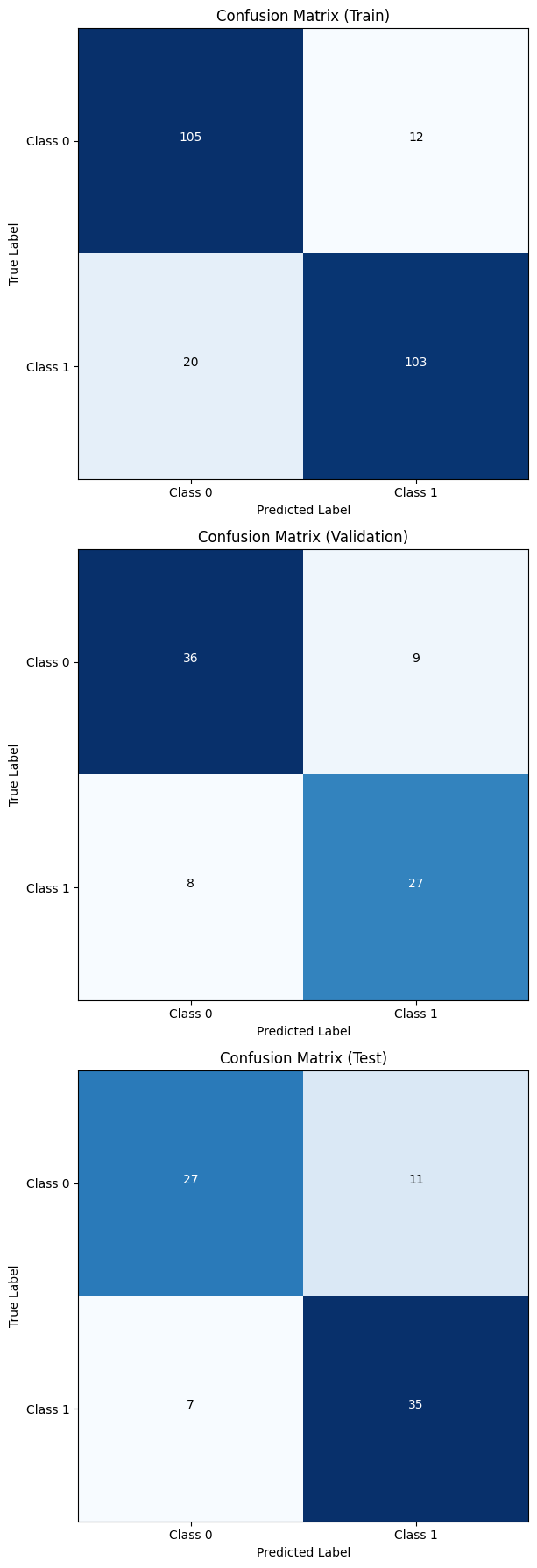


Grafik : Karar Sınırı

Grafik: Eğitim verisinin karmaşıklık matrisi



Grafik: Doğrulama verisinin karmaşıklık matrisi

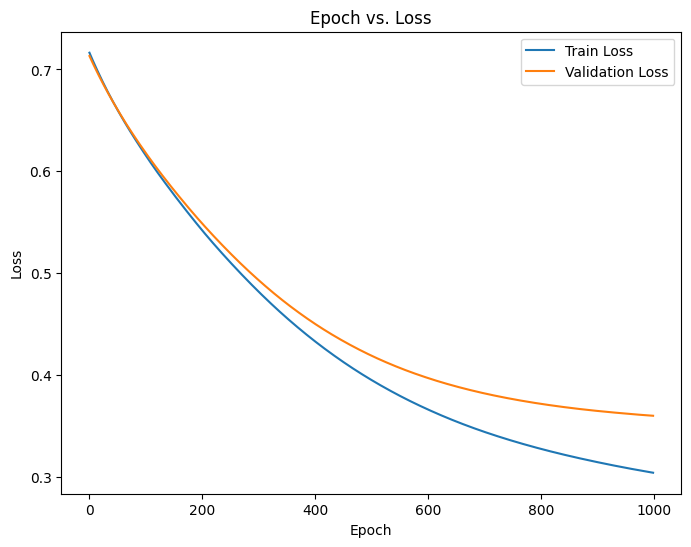


Grafik: Test verisinin karmaşıklık matrisi

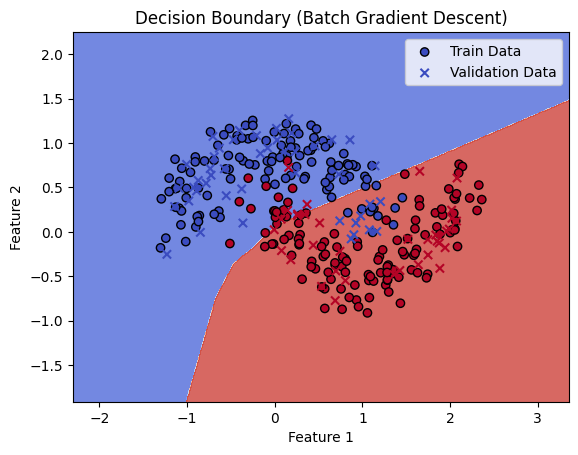
5.1.1.3) Batch Gradient Descent

Batch Gradient Descent (BGD), tüm eğitim veri setini bir kerede işleyerek ağırlıkların güncellenmesini sağlayan bir öğrenme yöntemidir. Bu yöntem, her iterasyonda tüm veri seti üzerinde hesaplama yaptığı için daha kararlı bir şekilde ağırlık güncellemesi yapar ancak büyük veri setlerinde yavaş çalışabilir.

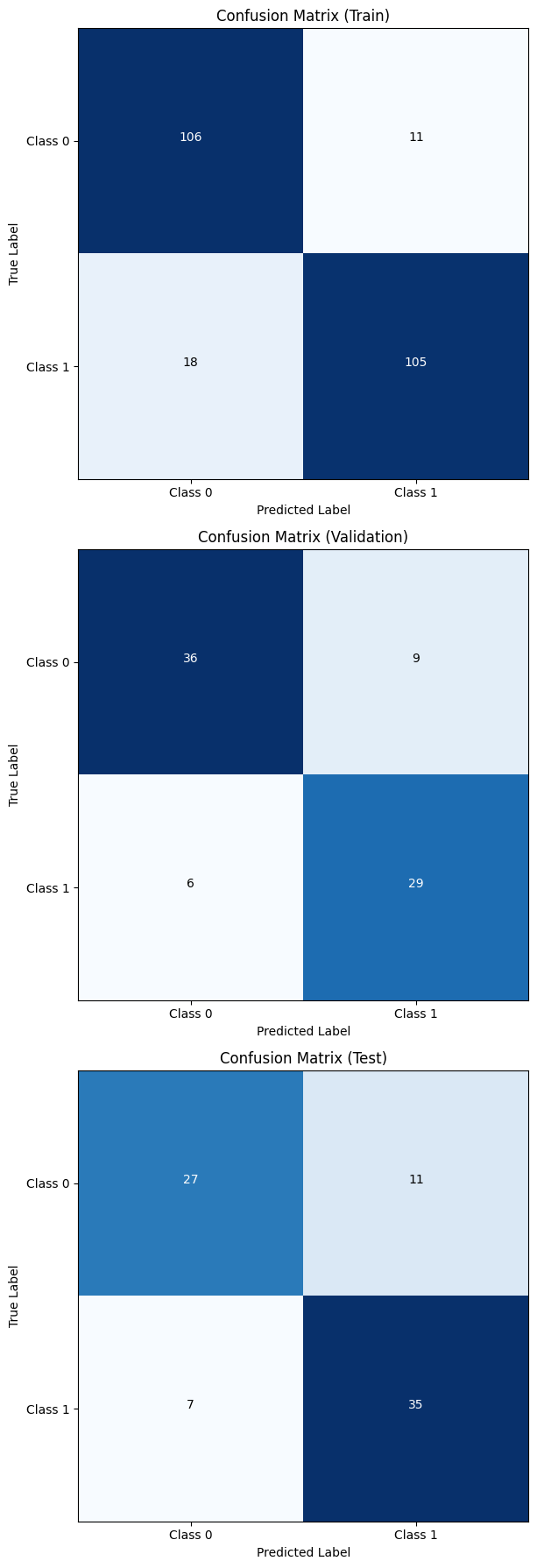
Bu yöntemde loss değeri 0.4093, accuracy 0.7749, precision değeri 0.7608, recall değeri 0.8333 bulunmuştur.



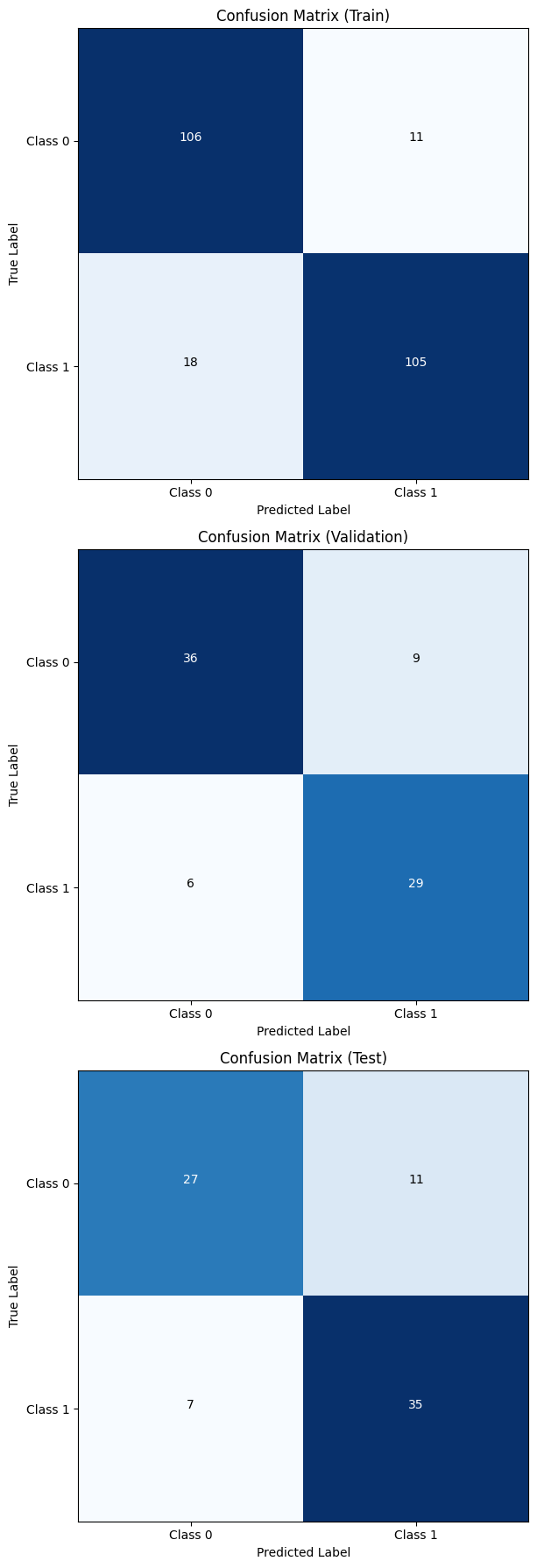
Grafik: Batch Gradient Descent yöntemiyle eğitilen tek katmanlı ağın eğitim ve doğrulama verileri için epoch/loss grafiği



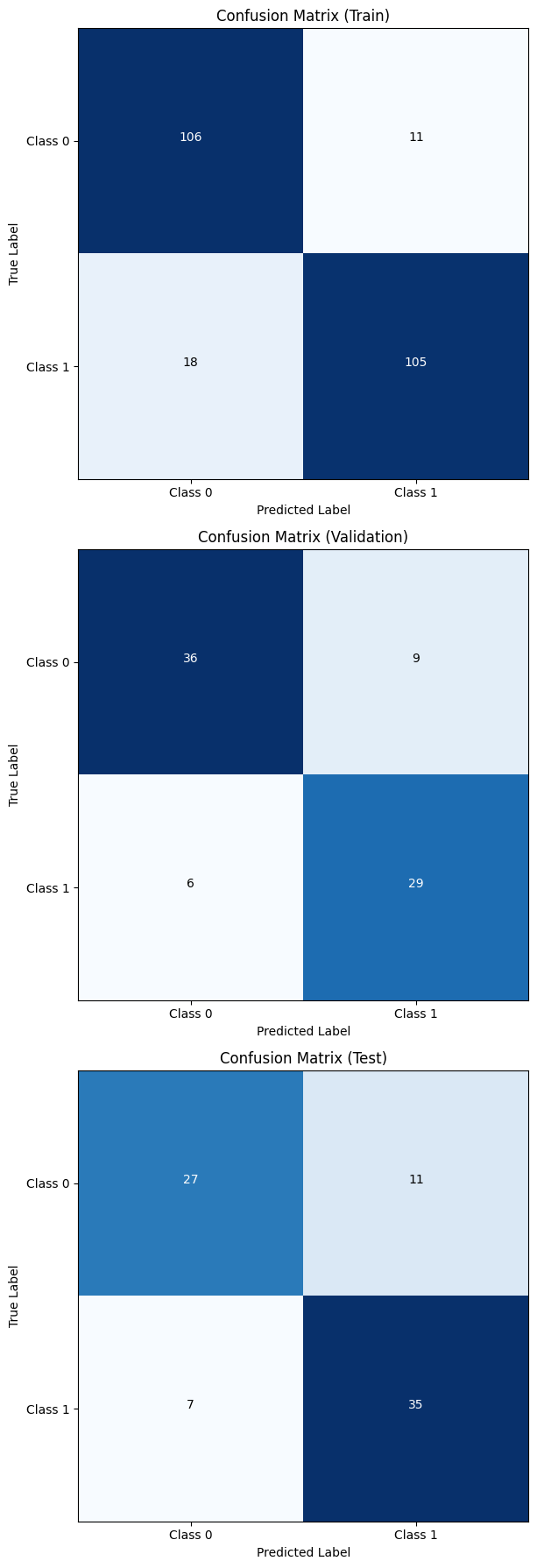
Grafik: Karar sınırı



Grafik: Eğitim verisinin karmaşıklık matrisi



Grafik: Doğrulama verisinin karmaşıklık matrisi



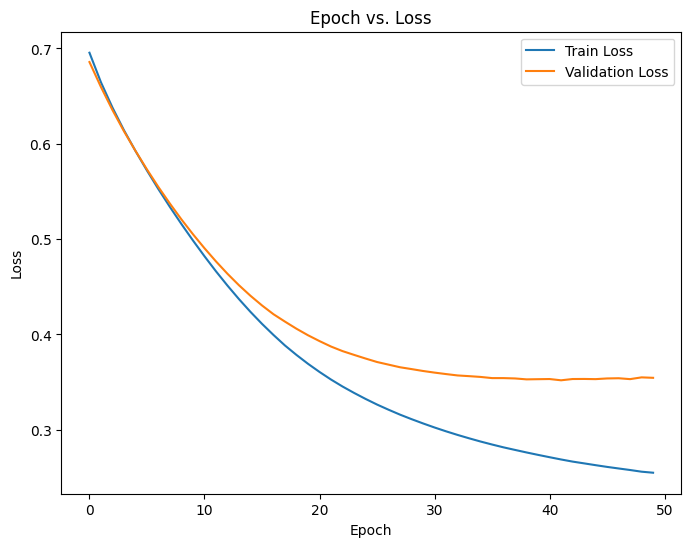
Grafik: Test verisinin karmaşıklık matrisi

5.1.2) 2 Katmanlı Ağ

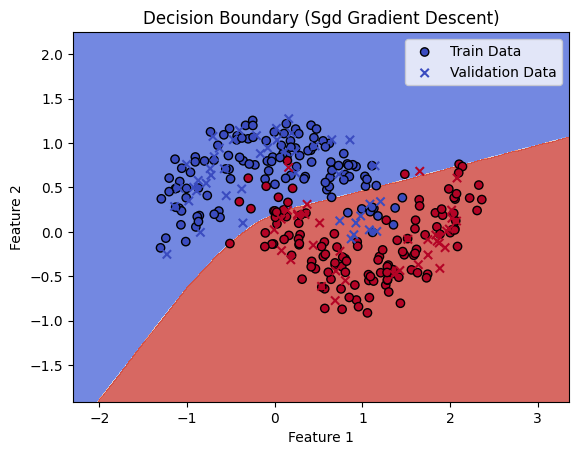
Bir katmanlı ağda yapılan tüm işlemler 2 katmanlı ağda da gerçekleştirilmiştir. Burada 64 nöronlu bir gizli ağ katmanı ile 32 nöronlu 2. gizli ağ katmanı kullanılmıştır.

5.1.2.1) Stochastic Gradient Descent

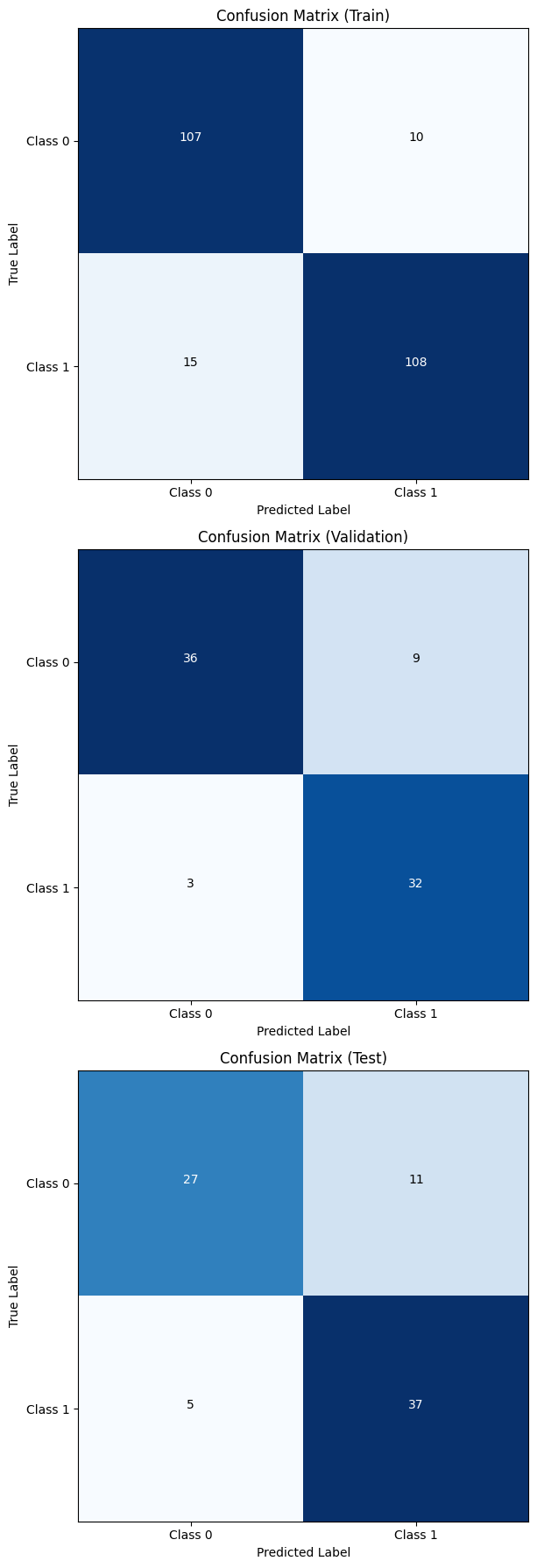
Loss değeri 0.3935, Accuracy değeri 0.8000, precision değeri 0.7708, recall değeri 0.8809dur.



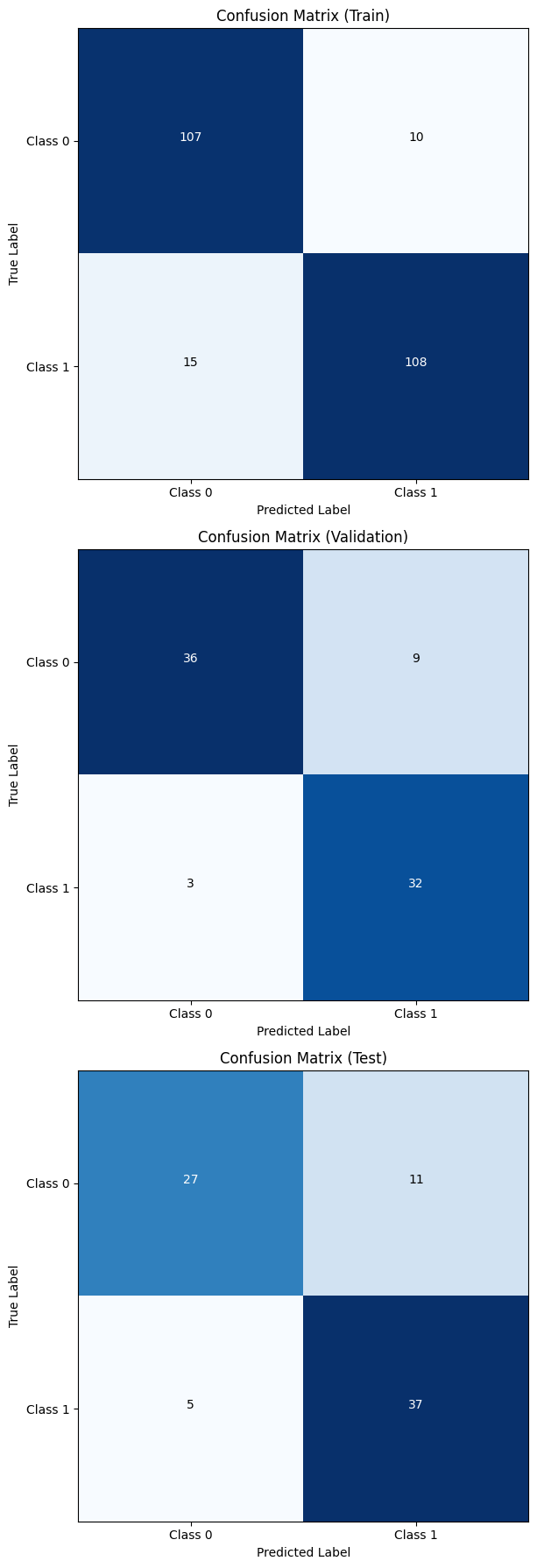
Grafik: Stochastic Gradient Descent yöntemiyle eğitilen iki katmanlı ağın eğitim ve doğrulama verileri için epoch/loss grafiği



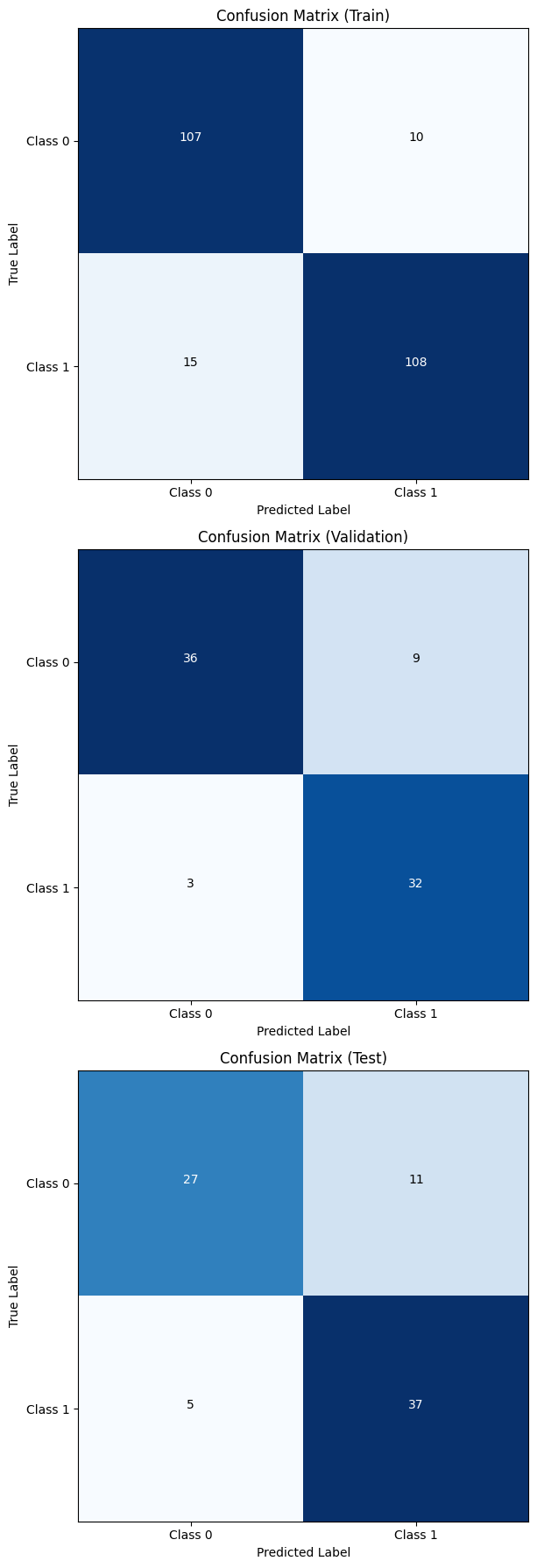
Grafik: Karar sınırı



Grafik: Eğitim verisinin karmaşıklık matrisi



Grafik: Doğrulama verisinin karmaşıklık matrisi

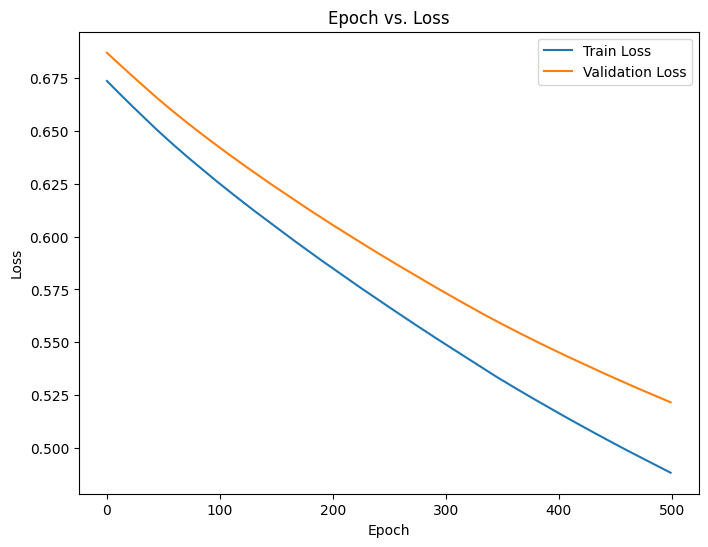


Grafik: Test verisinin karmaşıklık matrisi

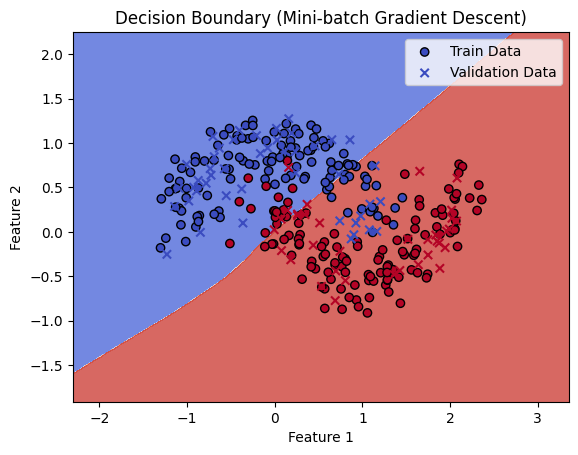
5.1.2.2) Mini Batch Gradient Descent

Mini batch yaklaşımında batch değeri 24 olarak kullanılmıştır.

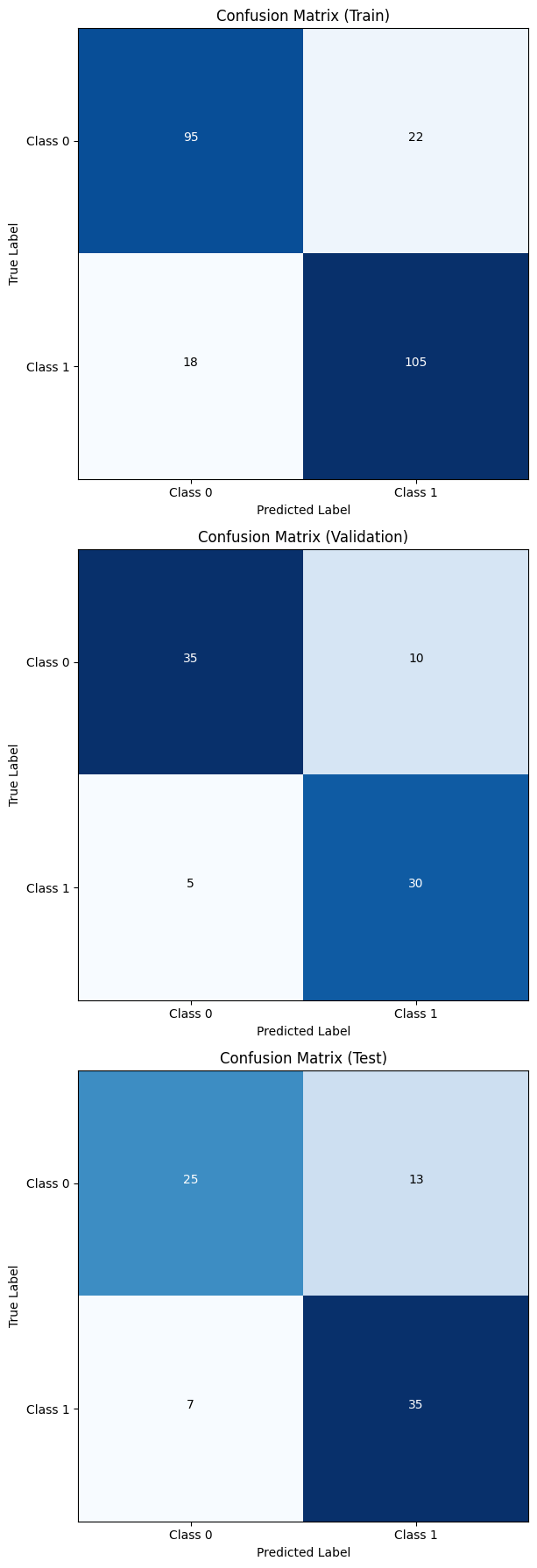
Loss değeri 0.5390, Accuracy 0.75, precision 0.72, recall 0.83 olarak bulunmuştur. Grafikler aşağıdaki gibidir:



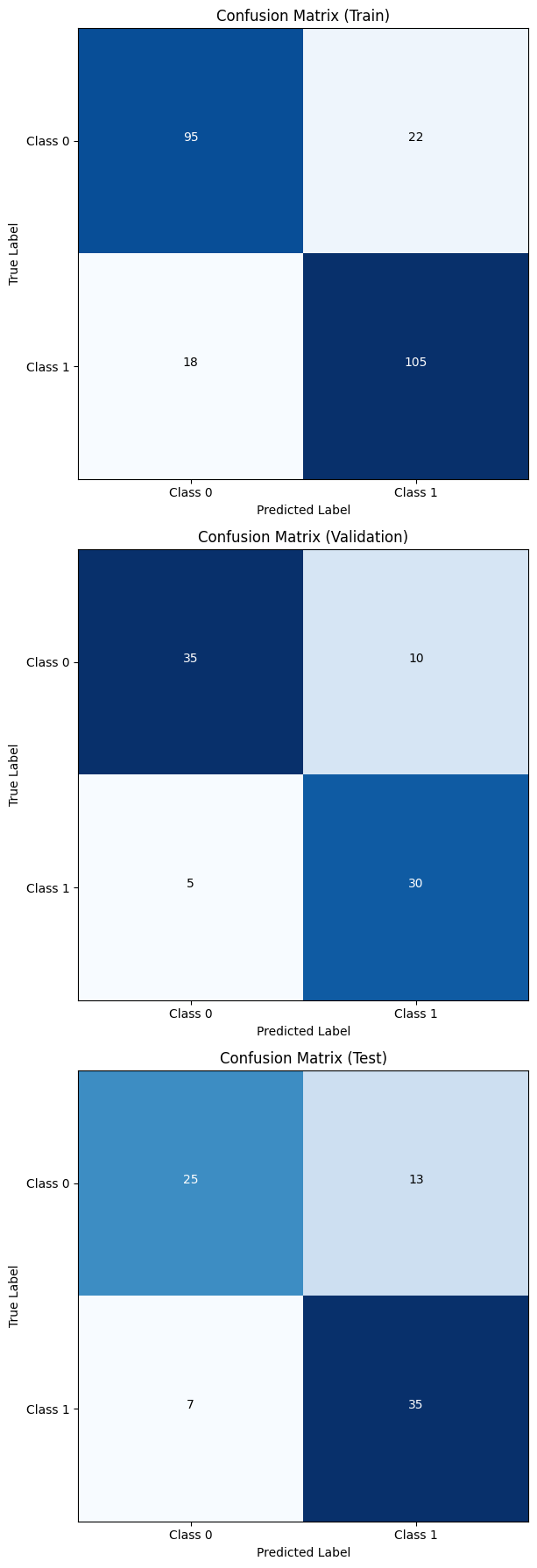
Grafik: Mini Batch Gradient Descent yöntemiyle eğitilen iki katmanlı ağın eğitim ve doğrulama verileri için epoch/loss grafiği



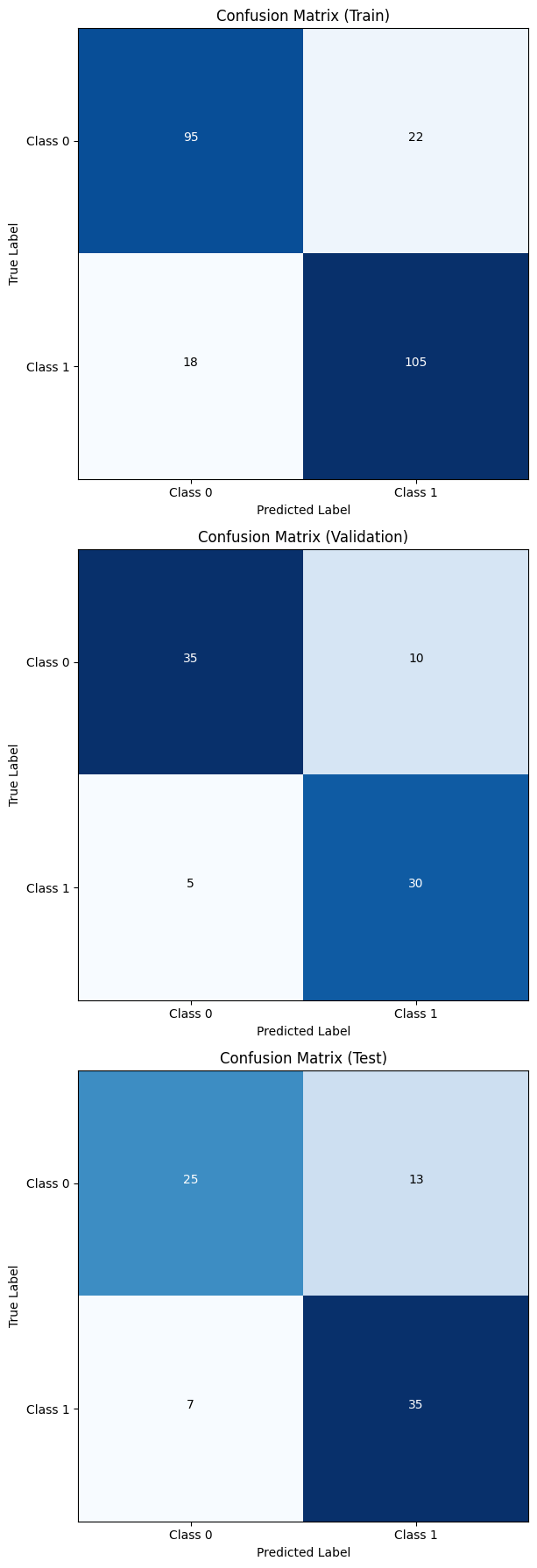
Grafik: Karar Sınırı



Grafik: Eğitim verisinin karmaşıklık matrisi



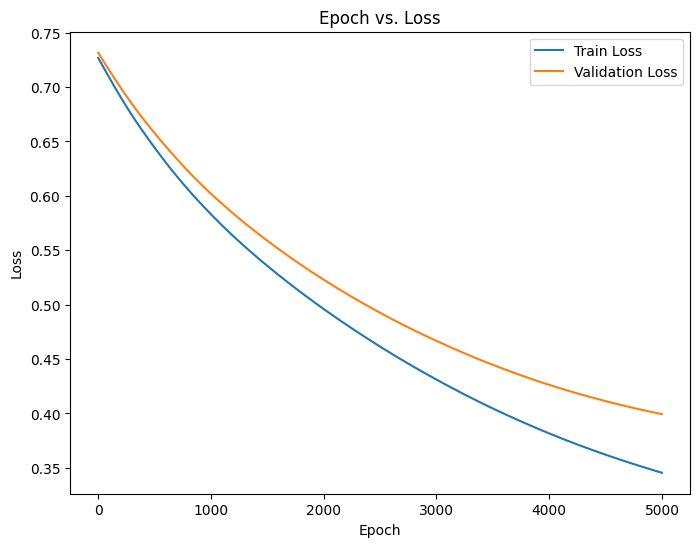
Grafik: Doğrulama verisinin karmaşıklık matrisi



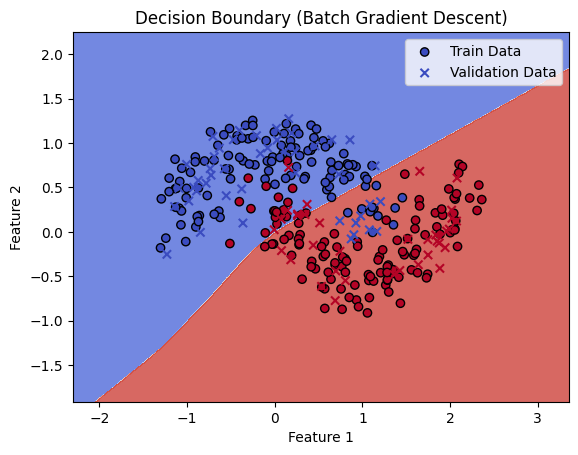
Grafik: Test verisinin karmaşıklık matrisi

5.1.2.3) Batch Gradient Descent

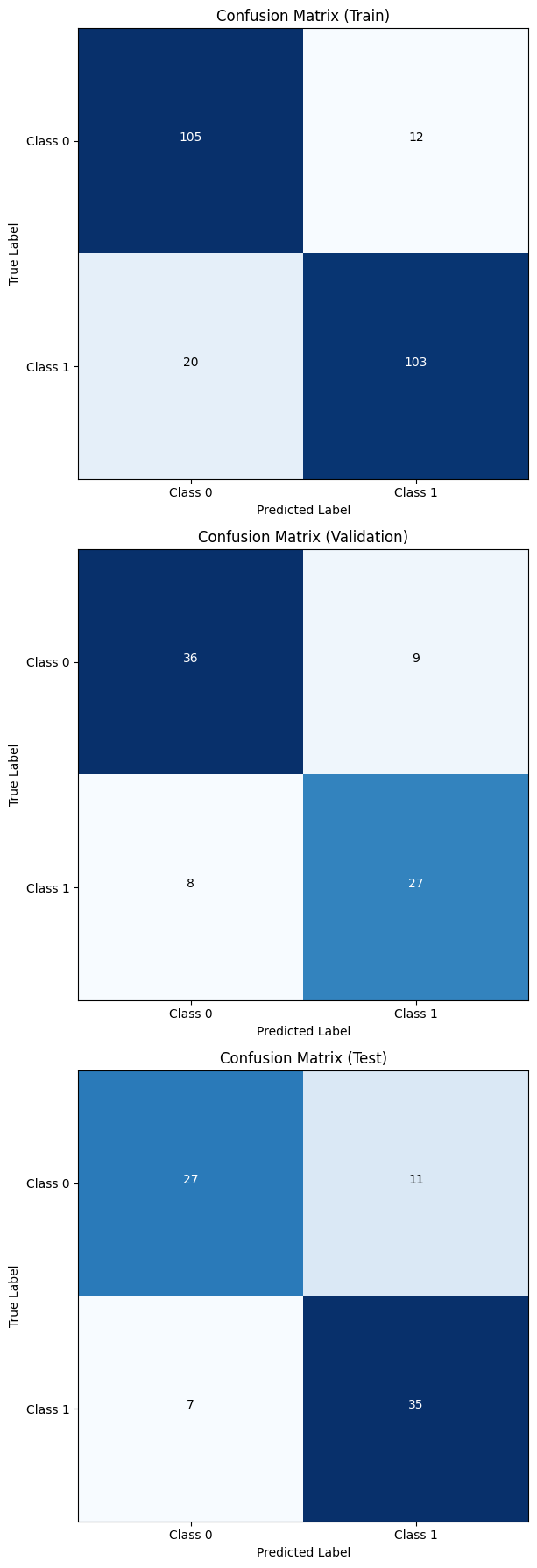
Bu yaklaşımda loss değeri 0.4372, accuracy 0.7703, precision değeri 0.7495, recall değeri 0.8101 bulunmuştur.



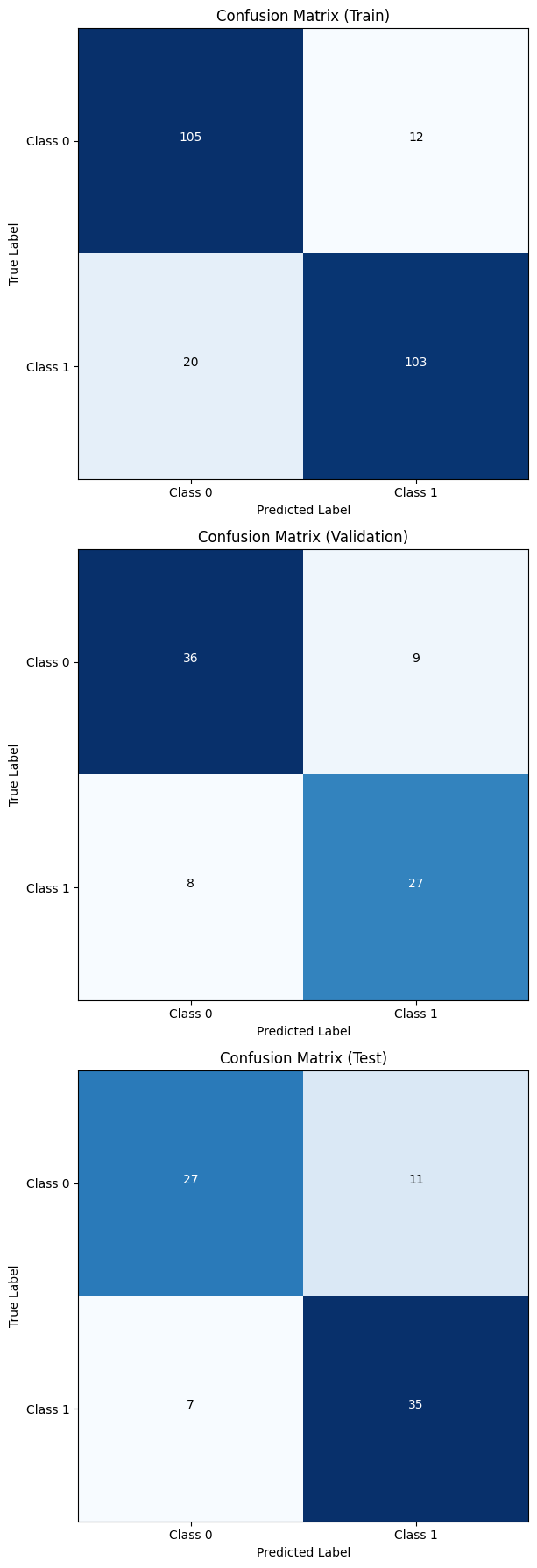
Grafik: Batch Gradient Descent yöntemiyle eğitilen iki katmanlı ağın eğitim ve doğrulama verileri için epoch/loss grafiği



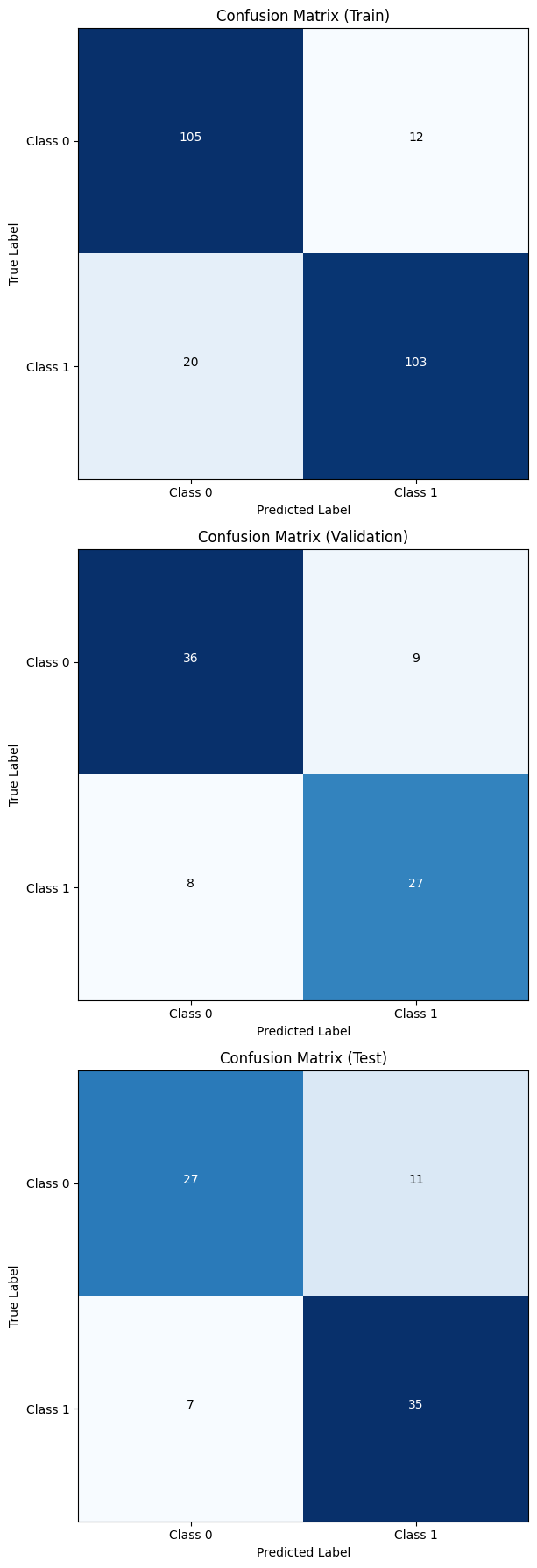
Grafik: Karar Sınırı



Grafik: Eğitim verisi için karmaşıklık matrisi



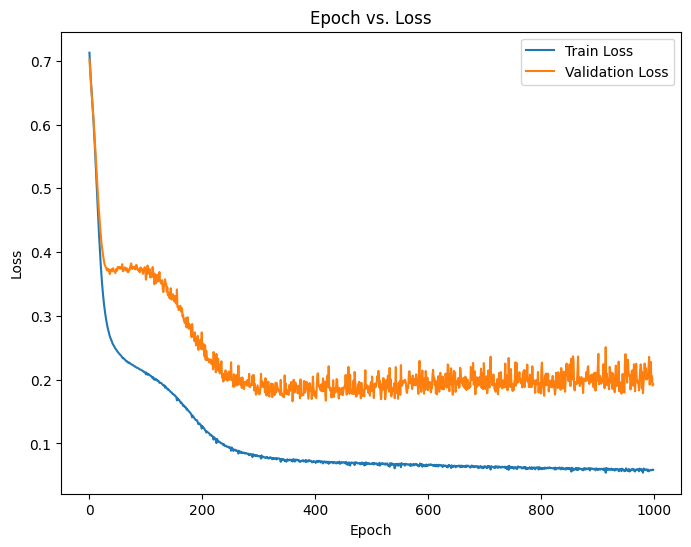
Grafik: Doğrulama verisi için karmaşıklık matrisi



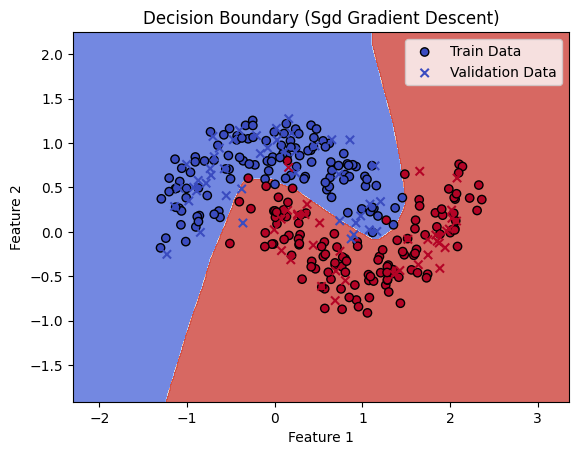
Grafik: Test verisi için karmaşıklık matrisi

5.1.3) 3 Katmanlı Ağ

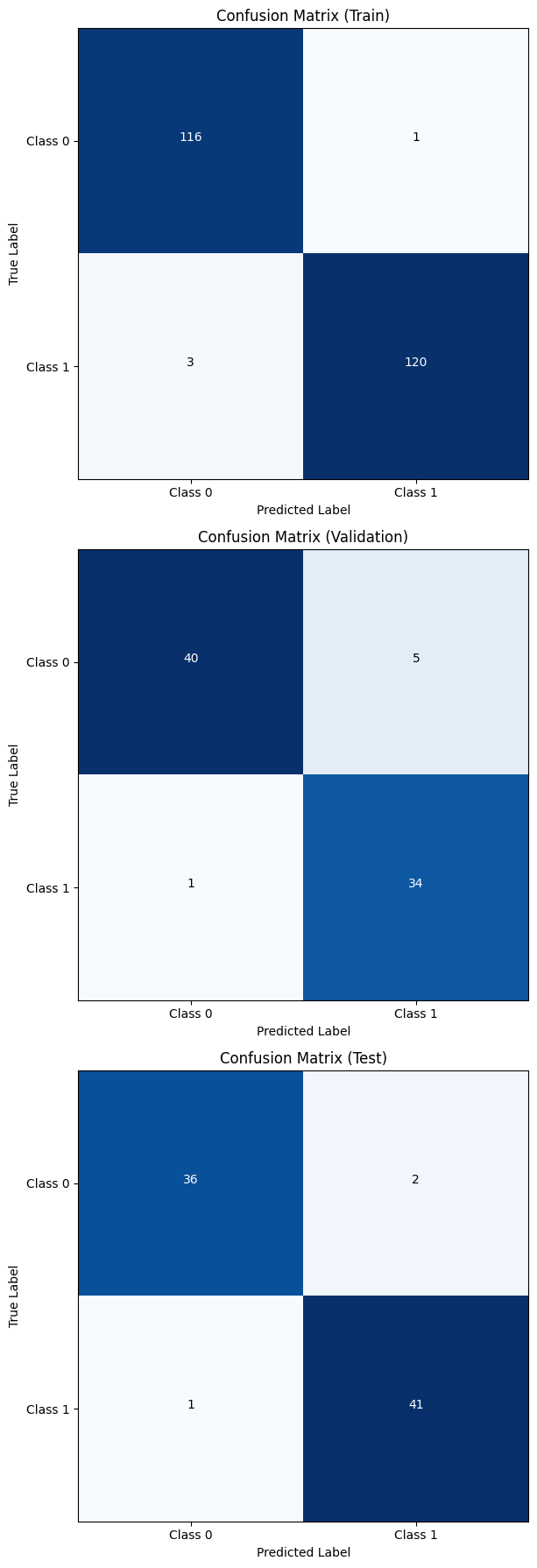
5.1.3.1) Stochastic Gradient Descent



Grafik: Stochastic Gradient Descent yöntemiyle eğitilen üç katmanlı ağın eğitim ve doğrulama verileri için epoch/loss grafiği



Grafik: Karar Sınırı

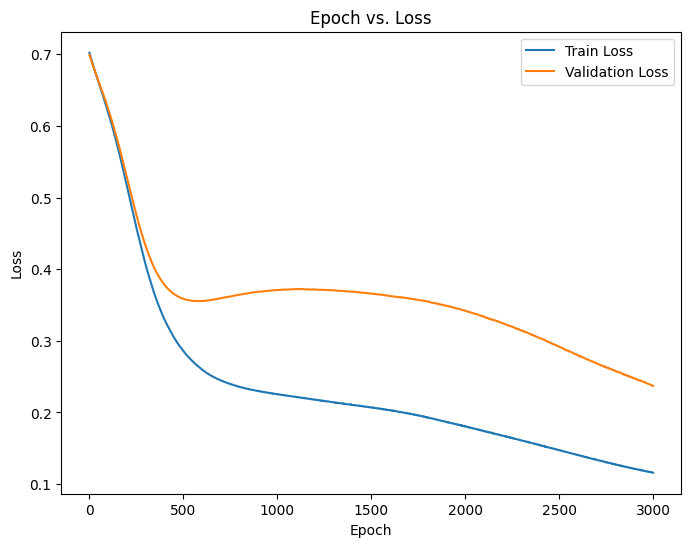


Grafikler: Eğitim, doğrulama ve test verileri için karmaşıklık matrisleri

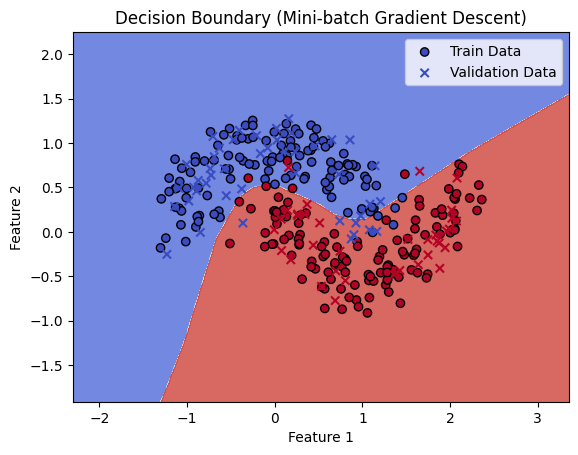
Loss: 0.111, Accuracy: 0.962, Precision: 0.9534, Recall değeri 0.9761 olarak bulunmuştur.

5.1.3.2) Mini Batch Gradient Descent

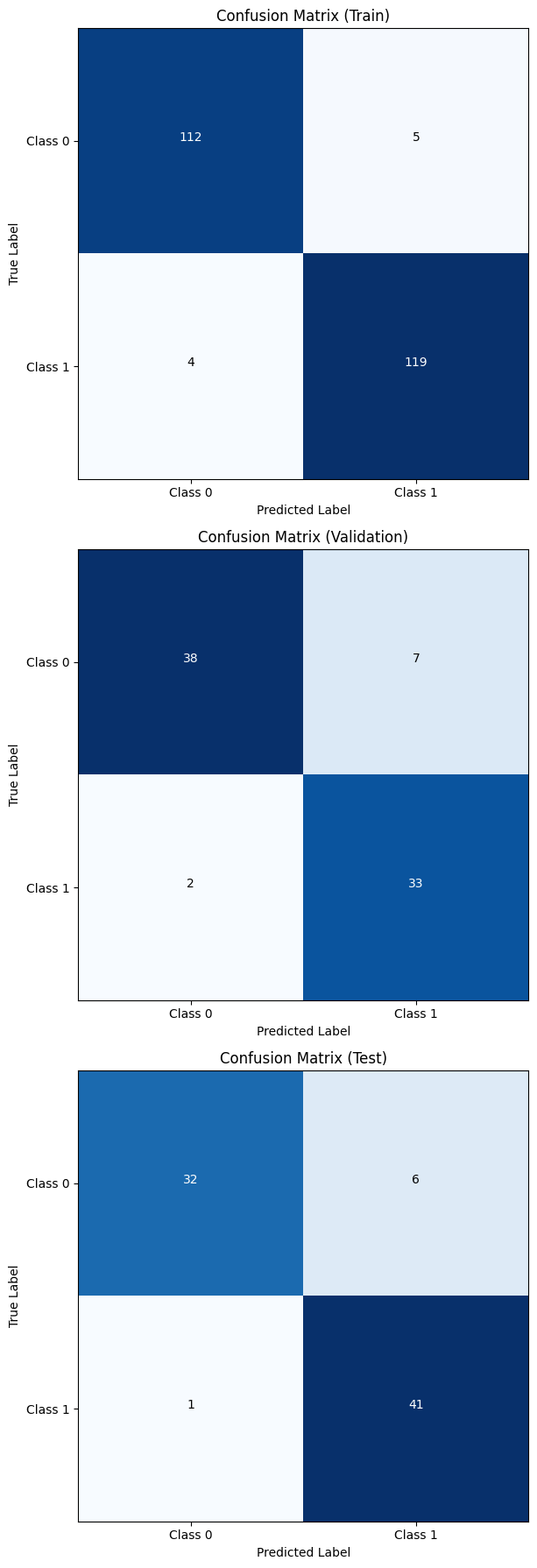
Bu yaklaşımda loss 0.183, accuracy 0.912, precision 0.872, recall ise 0.976 bulunmuştur.



Grafik: Mini Batch Gradient Descent yöntemiyle eğitilen üç katmanlı ağın eğitim ve doğrulama verileri için epoch/loss grafiği



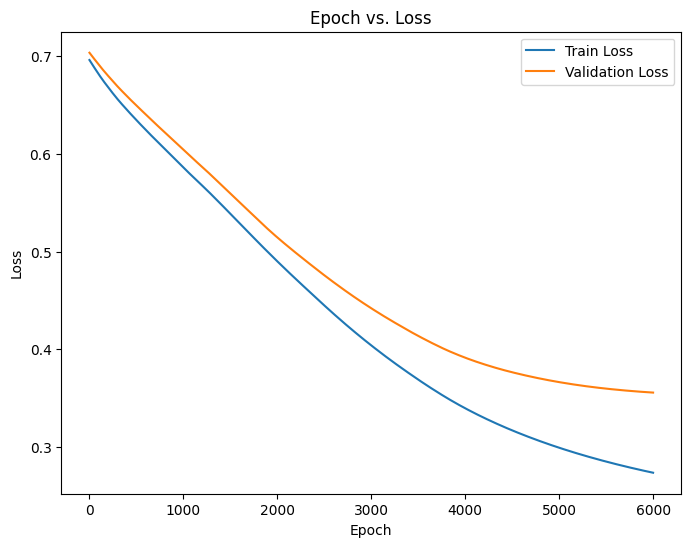
Grafik: Karar Sınırı



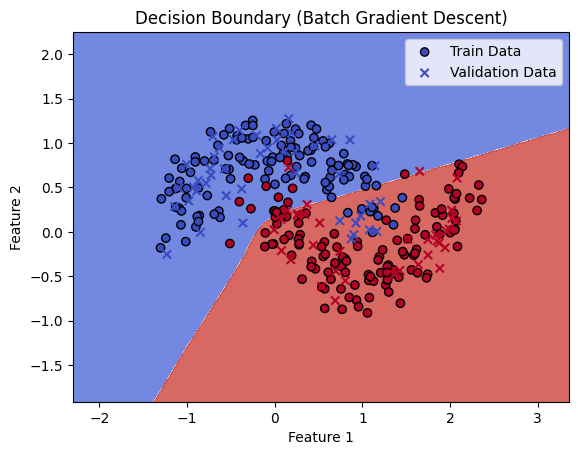
Grafikler: Eğitim, doğrulama ve test verileri için karmaşıklık matrisleri

5.1.3.3) Batch Gradient Descent

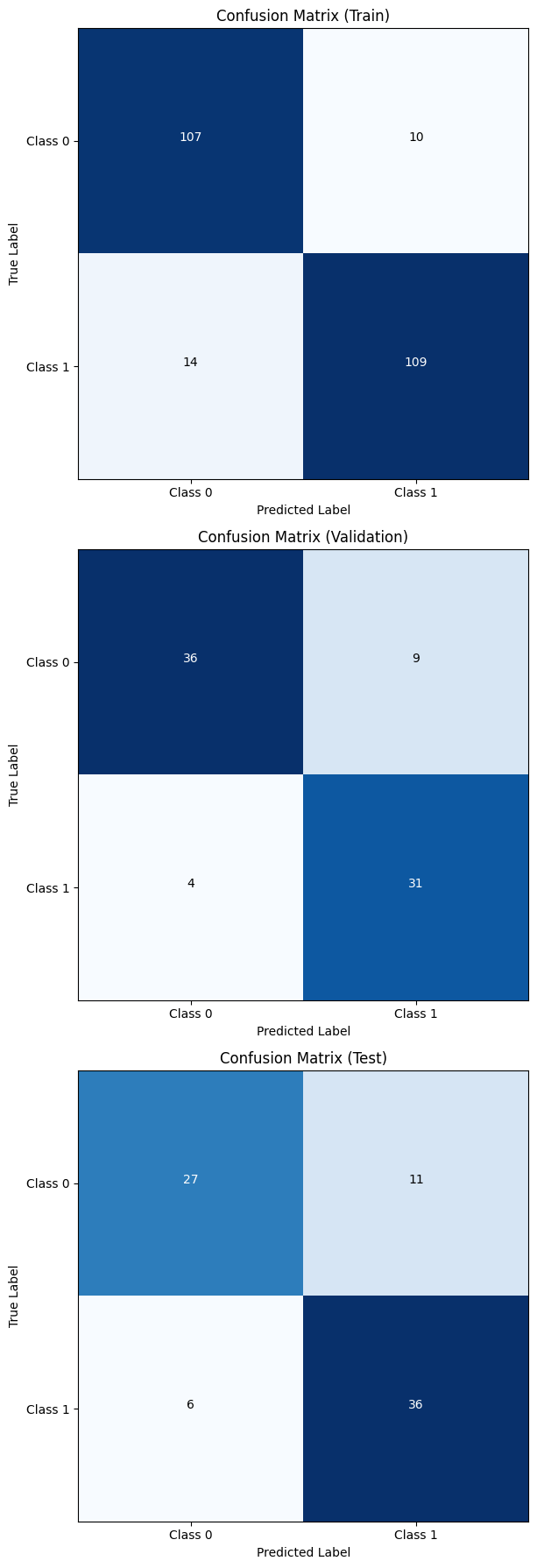
Bu yaklaşımda loss değeri 0.3904, accuracy 0.787, precision değeri 0.765, recall değeri 0.857 bulunmuştur.



Grafik: Batch Gradient Descent yöntemiyle eğitilen üç katmanlı ağın eğitim ve doğrulama verileri için epoch/loss grafiği



Grafik: Karar Sınırı



Grafikler: Eğitim, doğrulama ve test verileri için karmaşıklık matrisleri

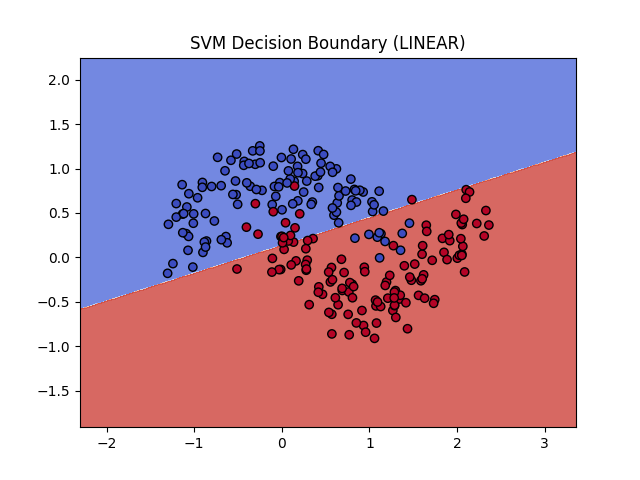
5.2) Destek Vektör Makineleri

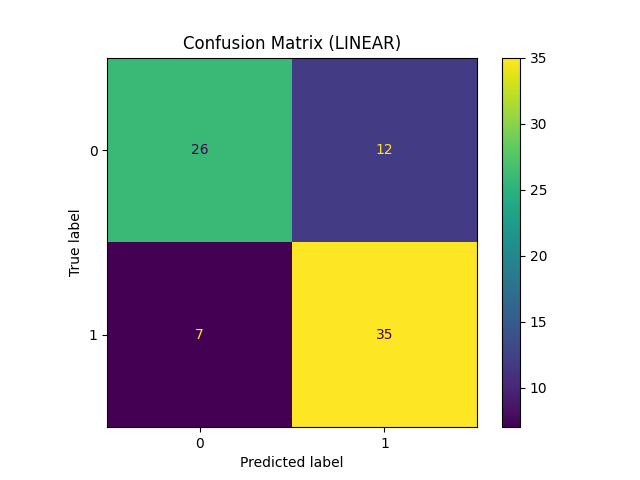
Önceki bölümlerde detaylıca anlatılan destek vektör makineleri, 3 farklı kernel fonksiyonu ile analiz edilmiştir.

5.2.1) Linear Fonksiyon

En iyi parametreler: {'C': 1} olarak bulunmuştur.

Doğrulama seti doğruluğu: 0.8958dir.



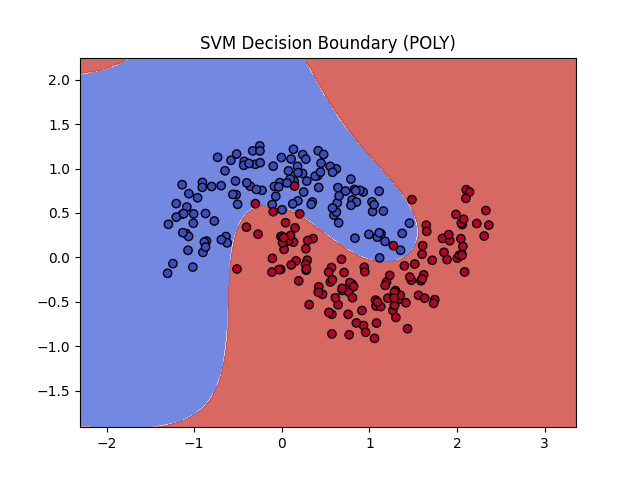
Linear fonksiyon için oluşan karar sınırı

Linear fonksiyon için oluşan karmaşıklık matrisi

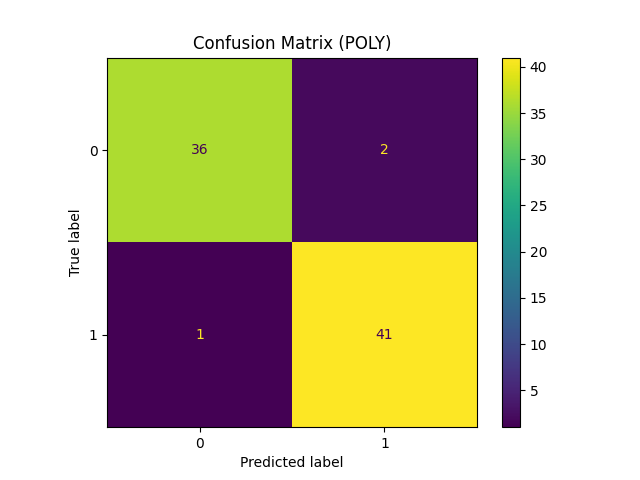
5.2.2.) Polynomial Function

En iyi parametreler: {'C': 10, 'coef0': 1, 'degree': 4} olarak bulunmuştur.

Doğrulama seti doğruluğu: 0.9750dir.



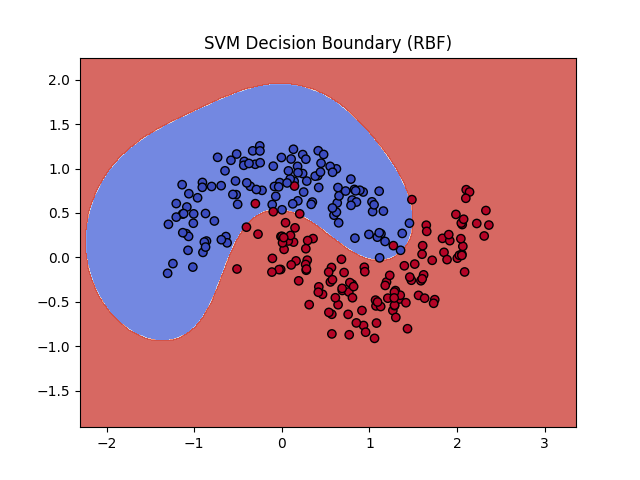
Polynomial fonksiyon için oluşan karar sınırı



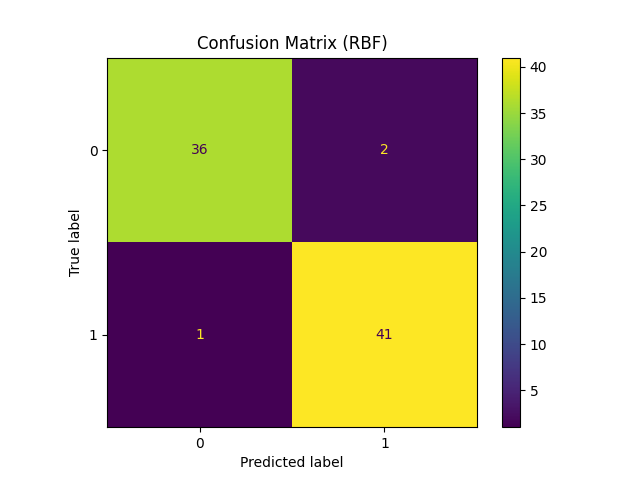
Polynomial fonksiyon için oluşan karmaşıklık matrisi

5.2.3) RBF Fonksiyonu

En iyi parametreler: {'C': 10, 'gamma': 'scale'} olarak bulunmuştur. Doğrulama seti doğruluğu: 0.9708dir.



RBF Fonksiyonu için oluşan karar sınırı



RBF fonksiyonu için oluşan karmaşıklık matrisi

6) Sonuç

Deney sonuçları, yapay sinir ağlarında katman sayısının ve kullanılan optimizasyon yöntemlerinin doğruluğa doğrudan etkisi olduğunu göstermiştir. Özellikle, üç katmanlı ağların Stochastic Gradient Descent (SGD) yöntemiyle eğitilmesi sonucunda 0.962 doğruluk değeri elde edilmiştir. Destek vektör makineleri ise doğrusal olmayan veri kümeleri için uygun kernel fonksiyonları ile etkili bir performans sergilemiş, en yüksek doğruluk değeri %97.5 ile Polynomial kernel kullanılarak sağlanmıştır. Elde edilen sonuçlar, problemin özelliklerine bağlı olarak her iki yöntemin de güçlü yanlarının olduğunu ortaya koymaktadır. Yapay sinir ağları, katmanlı yapısı sayesinde farklı veri yapılarına uyum sağlarken, destek vektör makineleri özellikle belirli veri türlerinde yüksek sınıflandırma başarısı sunmuştur.