T.C.
DOKUZ EYLÜL ÜNİVERSİTESİ
FEN FAKÜLTESİ
İSTATİSTİK BÖLÜMÜ

MAKİNE ÖĞRENMESİ YAKLAŞIMLARININ KARPAL TÜNEL SENDROMU CİDDİYET SINIFLAMASINDA KULLANILMASI

Bitirme Projesi Raporu

Alper ENGİN
Atadeniz SAYAR
Cem GÖRENER

Mayıs 2022

Rapor Değerlendirme

"MAKİNE ÖĞRENMESİ YAKLAŞIMLARININ KARPAL TÜNEL SENDROMU CİDDİYET SINIFLAMASINDA KULLANILMASI" başlıklı bitirme projesi raporu tarafımdan okunmuş, kapsamı ve niteliği açısından bir Bitirme Projesi raporu olarak kabul edilmiştir.

Dr. Engin YILDIZTEPE

Teşekkür

Tüm çalışma süresince yönlendiriciliği, katkıları ve yardımları ile yanımızda olan danışmanımız Dr. Engin YILDIZTEPE 'ye ve böyle bir çalışmayı yapmamız için bize fırsat tanıyan Dokuz Eylül Üniversitesi Fen Fakültesi İstatistik Bölümüne teşekkür ederiz.

Alper ENGİN
Atadeniz SAYAR
Cem GÖRENER

Özet

Özet, çalışmanın önemini ve faydasını anlatan bir bölüm degildir. Çalısmayı ana hatlarıyla anlatacak ve 300 kelimeyi aşmayacak şekilde hazırlanmalıdır. En az üç en çok beş anahtar kelime ilgili yere yazılmalıdır.

ikinci paragraf buradan başlar

Anahtar Kelimeler: anahtar kelime 1, anahtar kelime 2, anahtar kelime 3

Abstract

The preface pretty much says it all.

Second paragraph of abstract starts here.

Keywords: keyword1, keyword2, keyword3

İçerik

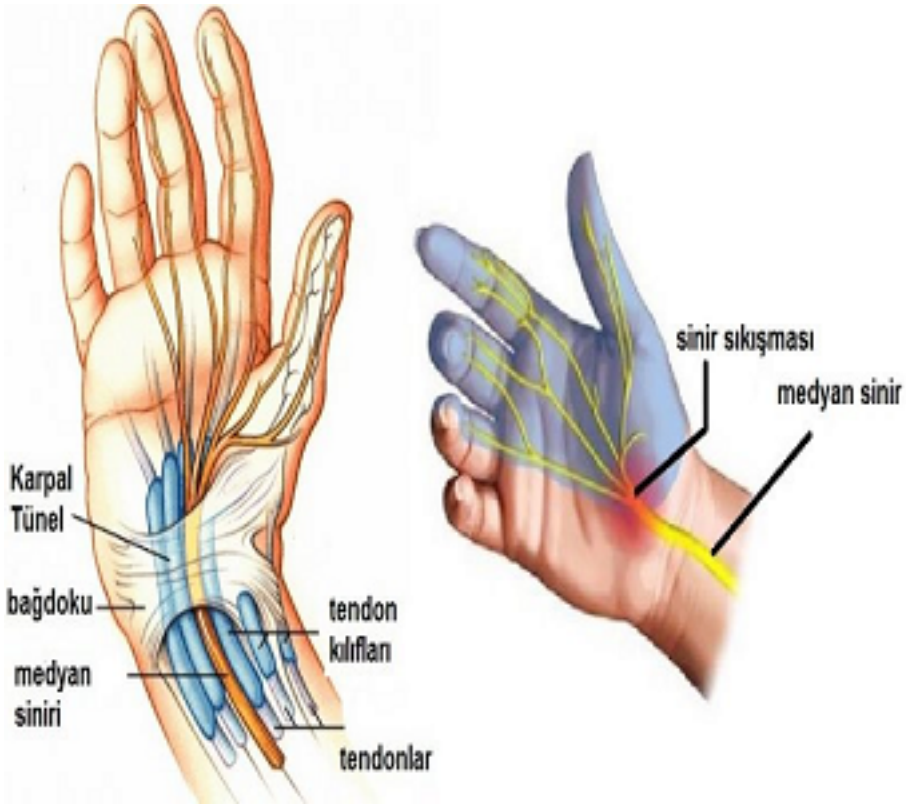
Tablo Listesi

Şekil Listesi

Bölüm 1

Karpal Tünel Sendromu

Karpal tünel sendromu, medyan sinirin karpal tüneli içerisinde baskıya uğraması sonucu ortaya çıkan semptomların genel adıdır (Werner ve Andary, 2002).
Tarihte ilk kez Pajet tarafından, 1854 yılında medyan sinir hasarının bulguları gözlenirken tanımlanmıştır (Pfeffer, Gelberman, Boyes ve Rydevik, 1988).
Karpal tünel sendromu (KTS), tanımlanması ve terimleştirilmesi ilk olarak 1947 yılında Brain, Wright ve Wilkinson tarafından yapılmıştır (Love, 1955).



Şekil 1.1: Karpal Tünel Anatomisi ve Medyan Sinirin Sıkışması

1.1 Epidemiyoloji

KTS prevalansı kadınlarda %3 ila %3.4 arasında, erkeklerde ise %0.6 ila % 2.7 arasında olarak belirlenmiştir. İnsidans ise kadınlarda 100.000'de 140, erkeklerde 100.000'de 52 olarak saptanmıştır. Kadınlarda genellikle menopoz dönemimde sıklıkla görülmüş olsa da hem erkek hem de kadınlarda gözlenme sıklığı yaş ile doğru orantılıdır. Özellikle 20 ila 50 yaş arasında daha sıktır. KTS'nin %40 ila %60 oranında her iki elde de başlayabileceği çeşitli yayınlarda bildirilmiş olup, iki elde de görüldüğü olgularda baskın elin genellikle semptomları daha önce ve daha şiddetli gösterdiği söylenebilir. KTS tek elde görüldüğü durumlarda ise genellikle semptomlar baskın elde görülür (Bagatur, 2006).

1.2 Etiyoloji

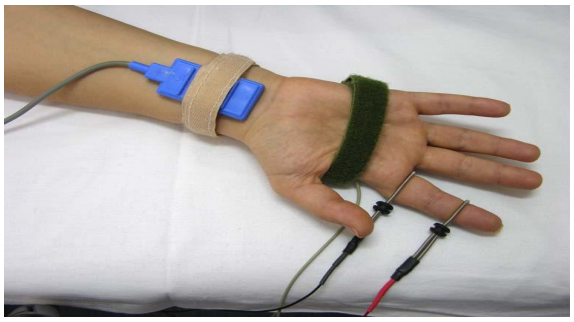
Karpal tünel sendromunun en sık nedeni; herhangi bir etiyolojik etkenin saptanamadığı idiopatik KTS'dir. İdiopatik KTS'de ailesel yatkınlık, obezite, VKİ fazla olması, kara şeklinde bilek yapısı gibi kişisel faktörlerin etken olduğu düşünülmektedir. Günlük yaşamda ki mekanik etkenler de idiopatik KTS üzerinde etkin rol oynamaktadır. Montaj işinde çalışan işçiler, fabrika çalışanları, klavye ve bilgisayar kullananlarda olduğu gibi el bilek fleksiyonun aktif olarak yapıldığı belli hareketlerin çok sık tekrarlanması da KTS ile ilişkili bulunmuştur (Robbins, 1963).

1.3 Semptomlar

Hastalığın şiddetine bağlı olarak semptomlar değişkendir. Erken evrelerde medyan sinirin duyusal liflerinin tutulumuna bağlı şikayetler görülür. En yaygın semptom el bileğinin merkezinden uzak dokularda sızlama ve uyuşuklukla beraber yanıcı tarzda ağrıdır. Başparmak tarafından itibaren ilk üç parmak ve dördüncü parmağın yanal yarısı etkilenir. Daha ileri dönemlerde el ayasında kas güçsüzlüğü ve körelme meydana gelir . Bu hastalarda elde, özellikle aktivite ile artan beceriksizlik ve objeleri kavramada kuvvetsizlik görülür (Aroori ve Spence, 2008).

1.4 Tanı ve Ciddiyet Değerlendirmesi

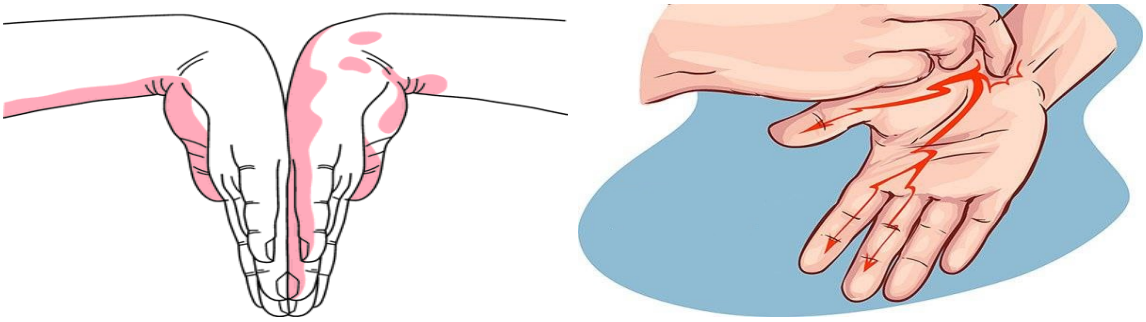
Karpal tünel sendromunda tanı koymak için hastanın hikayesi, klinik semptomlar, fizik muayene bulguları ve bu bulguları destekleyen çeşitli testler kullanılmaktadır(Ghasemi-Rad ve diğerleri, 2014). Bu testler elektronörofizyolojik, provokatif testler ve tıbbi görüntülemeye dayanan testlerdir. Elektronörofizyolojik testler karpal Tünel'e bağlanan elektrotlar ile elektrik sinyallerinin incelenmesi ve sonuçların bilgisayar ile yorumlanmasına dayananan testlerdir.



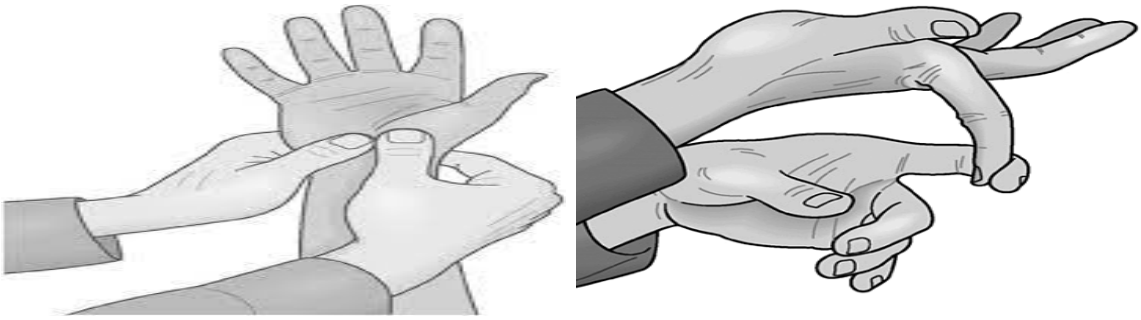
Şekil 1.2: Elektronörofizyolojik Test (Kumaş, 2005)

Provokatif testler hastanın bilek ve parmak eklemlerine fiziksel baskı uygulayacak şekilde bir takım testler uygulanması ve alınan sonuçların değerlendirmesine dayanan deneysel test yöntemleridir. Tanısal testler genellikle karpal tüneli görüntülemeye dayanan testlerdir.

- Phalen Testi
  - 60 saniye boyunca parmaklar ayak ucuna bakacak şekilde el dış yüzleri birleştirilir. Meydan sinir bölgesinde karıncalanma oluşur veya artarsa test pozitiftir.
- Ters Phalen testi
  - 60 saniye boyunca parmaklar yukarı bakacak şekilde el dış yüzleri birleştirilir. Meydan sinir bölgesinde karıncalanma oluşur veya artarsa test pozitiftir.
- Tinel testi
  - Uygulayıcı tarafından karpal tünelin üstüne perküsyon yapılır. Medyan sinir bölgesinde karıncalanma ve elektrik şoku hissi oluşursa test pozitiftir (Kurt, 2020).
- Karpal kompresyon testi
  - El bileği düz tutulurken medyan sinirin yakınına başparmak ile bastırılır. Medyan sinir bölgesinde karıncalanma oluşur veya artarsa test pozitiftir.
- Gerilmiş median sinir stres (GMSS) testi
  - Medyan sinir hareketliliğinin azaldığı durumlarda medyan sinirin gerilerek lokal iskeminin arttırılması mantığına dayanır.



Şekil 1.3: Phalen ve Tinel Testi

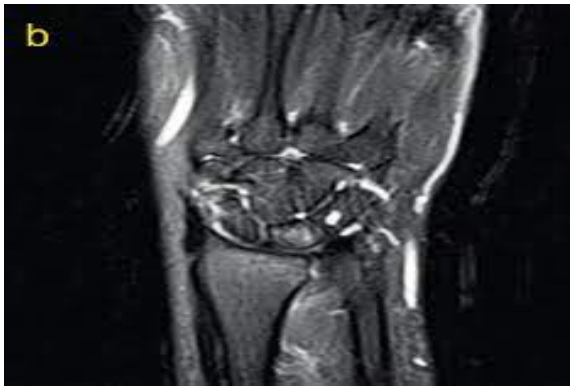


Şekil 1.4: Karpal kompresyon ve GMSS Testi

Görüntülemeye dayalı testlerde el bileği ve parmakların hareketi sırasında karpal tünel içerisindeki değişiklikleri ve medyan sinirin hareketlerini yorumlayarak hastaya tanı koymayı kolaylaştırır fakat hastalığın şiddeti hakkında bilgi vermez.

- Ultrasonografi
- Düz radyografi
- Bilgisayarlı tomografi
- Manyetik rezonans görüntüleme

Şekil 1.5: Ultrasonografi ve Düz radyografi



Şekil 1.6: Bilgisayarlı tomografi

İdiopatik karpal tünel sendromunda hastalığın tanımlanmasında Boston Karpal Tünel Sendromu Anketi(BKTSA) kullanılmaktadır (Levine ve diğerleri, 1993). Bu ankete farklı dillere çevrilmiş ve ülkelere göre uyarlanmıştır. Anketin amacı hastanın yanıtlarına göre bir ciddiyet sınıflandırması yapmaktır. Anketin Türkçe versiyonu Sezgin ve ark. (Sezgin ve diğerleri, 2006) tarafından yayımlanmıştır , ancak BKTSA sadece hastaların verdiği yanıtlara dayanarak bir semptom şiddeti belirlemeyi amaçlar.

Teknolojinin hızla gelişmesi ile birlikte hastalara uygulanan testlerin sonuçlarının toplanmasının kolaylaşmasının yanı sıra testlerin sonuçlarına bağlı olarak hastaya tanı koymak ve tanının şiddetini ve derecesini tespit etmek oldukça kolaylaşmıştır.
Makine öğrenmesi ve Yapay zeka uygulamalarının yaygınlaşması ile birlikte bu yöntemlerin tıp alanında da kullanımı artmıştır.

Makine öğrenmesi yöntemlerinin KTS tanısında kullanılmaına örnek olarak. Ardakani ve ark. (Ardakani ve diğerleri, 2020) tarafından hasta olduğu bilinen kişilerden elde edilen bilgisayarlı tomografi görüntüleri, derin öğrenme metodları kullanılarak başka kişilerin hasta olup olmadığını tespit etmek için kullanılmıştır.
Bir diğer çalışma ise 2021 yılında Koyama ve ark. (Koyama ve diğerleri, 2021) tarafından geliştirilen bir mobil uygulama sayesinde hastaların ekranın farklı yerlerinde çıkan cisimlere ulaşma sürelerini baz alarak hastalığın evresini tahminlemeyi amaçlamıştır. Bu uygulama hastanın kendi kendine ev ortamında hastalığına ön tanı koyabilmesi açısından yararlı olabilir.
Bunların yanı sıra KTS ciddiyet skoru belirlemek için makine öğrenmesi yöntemlerini kullanan çalışmalar da yapılmaktadır. Güncel bir çalışmada Park ve ark. (Park ve diğerleri, 2021) 1037 hastadan elde edilen verileri farklı makine öğrenmesi yöntemlerinde kullanarak KTS ciddiyet sınıflandırmasını tahmin etmeyi amaçlamışlardır.

Bölüm 2

Yöntem

Bu bölümde uygulama kısmında kullanılan sınıflama algoritmalarına değinilmiştir.

2.1 K - En Yakın Komşuluk Algoritaması (K-NN)

K-NN algoritması, Cover ve Hart tarafından önerilen, örnek veri noktasının bulunduğu sınıfın ve en yakın komşunun, k değerine göre belirlendiği bir sınıflandırma yöntemidir (Cover ve Hart, 1967).
K-NN algoritması, en temel örnek tabanlı öğrenme algoritmaları arasındadır. Örnek tabanlı öğrenme algoritmalarında, öğrenme işlemi eğitim setinde tutulan verilere dayalı olarak gerçekleştirilmektedir. Yeni karşılaşılan bir örnek, eğitim setinde yer alan örnekler ile arasındaki benzerliğe göre sınıflandırılmaktadır (Mitchell ve Learning, 1997). K-NN algoritmasında, eğitim setinde yer alan örnekler n boyutlu sayısal nitelikler ile belirtilir. Her örnek n boyutlu uzayda bir noktayı temsil edecek biçimde tüm eğitim örnekleri n boyutlu bir örnek uzayında tutulur. Bilinmeyen bir örnek ile karşılaşıldığında, eğitim setinden ilgili örneğe en yakın k tane örnek belirlenerek yeni örneğin sınıf etiketi, k en yakın komşusunun sınıf etiketlerinin çoğunluk oylamasına göre atanır (Mining, 2006).

2.1.1 K-NN Parametleri

K-NN algoritmasında performansı etkileyen 3 adet hiper parametre mevcuttur. Bunlar; Uzaklık ölçütü, komşu sayısı(k) ve ağırlıklandırma yöntemidir.

Uzaklık Ölçütü

En bilinen ve yaygın olarak kullanılan 3 uzaklık;

- Minkowski Uzaklığı
- Öklid Uzaklığı
- Manhattan Uzaklığı

Komşu Sayısı (k)

En yakın komşuluk algoritmasında komşu sayısına (k) göre sınıflama yapıldığından algoritma için en önemli parametresi olduğu söylenebilir. k = 5 olarak belirlendiğinde yeni gözlem kendisine en yakın 5 değer baz alınarak sınıflandırılır.

Ağırlıklandırma

Komşular için ağırlık değerleri atanması ile sınıflandırılmakta olan örneğe daha yakın olan komşu örneklerin, çoğunluk oylamasına daha fazla katkı koyması amaçlanır. En çok kullanılan ağırlık değeri atama yöntemleri, her bir komşunun ağırlığının, d, komşular arası uzaklık olmak üzere, 1/d ya da $1/d^2$ şeklinde alınmasıdır (Doad ve Bartere, 2013).

2.2 Rassal Ormanlar

Rassal ormanlar sınıflandırıcısı, her biri farklı girdi vektörlerinden oluşan ve her ağacın yalnızca bir sınıfa oy verdiği karar ağaçlarının kombinasyonudur (Leo Breiman, 1999). Rassal ormanlar sınıflandırması ağaç derinliğini büyütmek için her dalda rastgele değişkenleri içerir.

Karar ağaçlarını modellemek, bir budama metodu ve nitelik seçme ölçütü seçmeyi gerektirir. Karar ağacının tüme varımı için kullanınlan bir sürü nitelik seçme yaklaşımı vardır ve çoğu yaklaşımlar niteliğe direkt olarak kalite ölçümü belirler. Karar ağacının tümü varmında ki en sık kullanılan nitelik seçme ölçümleri Information gain ratio kriteri ve Gini indexidir (L. Breiman, Friedman, Olshen ve Stone, t.y.).

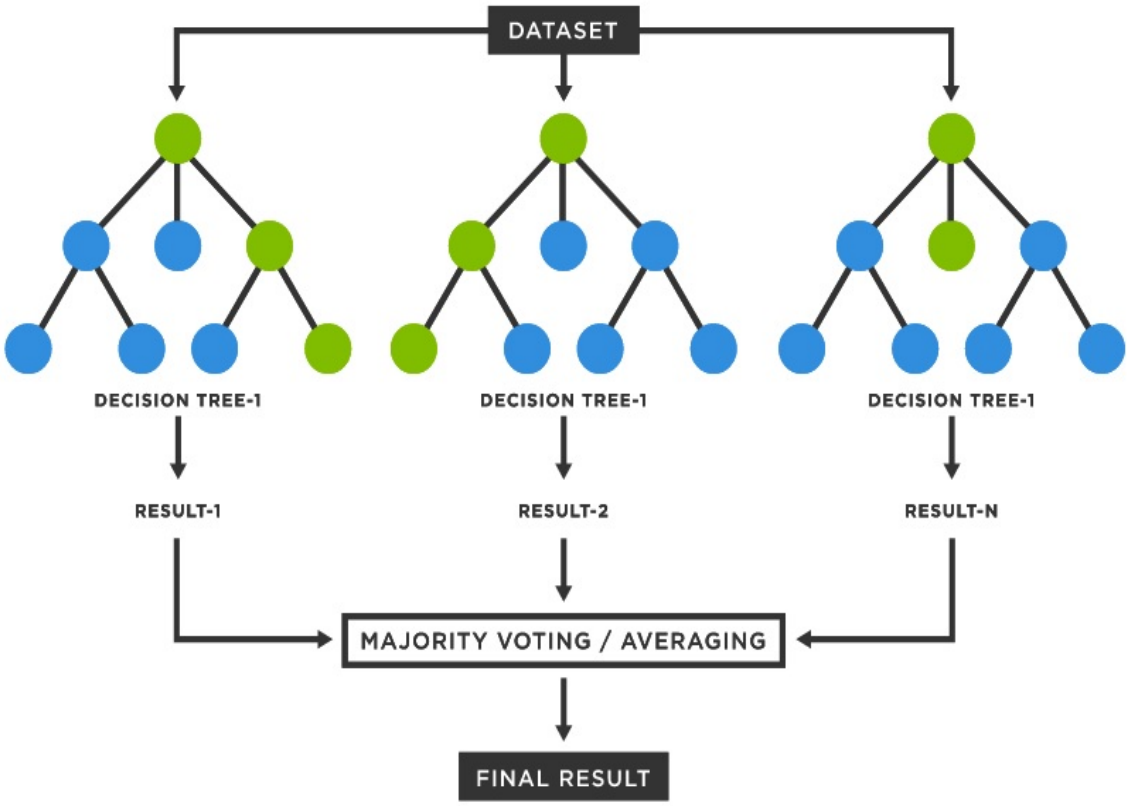
Verilen herhangi bir eğitim verisi için gini index o verinin hangi sınıfa ait olduğu olasılığını hesaplar.

$$\sum \sum_{j \neq i} (f(C_i, T)/|T|)(f(C_j, T)/|T|)$$

Bu denklemde $f(C_i, T)/|T|)$ seçilen gözlemin C_i sınıfa ait olma olasılığıdır.

Her seferinde bir ağaç, değişkenlerin bir kombinasyonunu kullanarak yeni eğitim verisi üzerinde en büyük derinliğe kadar büyür. Son derinliğine ulaşmış bu ağaçlar budanmamıştır. Bu durum, (Quinlan, 2014) veya diğer karar ağacı methodlarına göre rassal ormanlanların en büyük avantajıdır. Bazı durumlarda maliyet ve karmaşıklığı minimum yapabilmek için rassal ormanlar içerisindeki karar ağaçlarına budama yapılır. Budama parametresi olan 'ccp_alpha' değerinden daha küçük olan en büyük maliyet ve karmaşıklık değerini bulunana kadar karar ağacı budanır.

Sonuç olarak, rastgele ağaçlar methodu, kullanıcının herhangi bir değer belirleyebildiği, N kadar büyüyecek karar ağaçları içerir. Yeni veri setini sınıflandırmak için, seçilen maksimum değişken sayısına göre veri setinde ki rastgele olarak paylaştırılmış her gözlem, N kadar karar ağacının her biri tarafından sınıflandırılır. Bu durum için ormanlar en fazla oya sahip sınıfı seçer.



Şekil 2.1: Rassal Ormanlar Model Örneği (TIBCO, t.y.)

2.3 XGBoost

Bu bölümde XGboost algoritmasına ve XGBoost algoritmasının daha iyi anlaşabilmesi için XGBoost'un altyapısını oluşturan gradyan arttırma algoritmasına değinilmiştir.

2.3.1 Gradyan Arttırımı

Gradyan artırma fikri, Leo Breiman'ın , artırmanın uygun bir maliyet fonksiyonu üzerinde bir optimizasyon algoritması olarak yorumlanabileceği gözleminden kaynaklanmıştır (Leo Breiman, 1997). Açık regresyon gradyan artırma algoritmaları daha sonra Jerome H. Friedman (Friedman, 2001), tarafından Llew Mason, Jonathan Baxter, Peter Bartlett ve Marcus Frean'ın daha genel fonksiyonel gradyan artırma perspektifiyle eş zamanlı olarak geliştirildi (Mason, Baxter, Bartlett ve Frean, 1999).

Gradyan artırma, genellikle karar ağaçları gibi zayıf tahmin modelleri topluluğu şeklinde bir tahmin modeli üreten, regresyon, sınıflandırma ve diğer görevler için bir makine öğrenimi tekniğidir. Bir karar ağacı zayıf öğrenen ise, ortaya çıkan algoritmaya genellikle rastgele ormandan daha iyi performans gösteren gradyan destekli ağaçlar denir (Hastie, Tibshirani ve Friedman, 2009). Modeli, diğer artırma yöntemlerinin yaptığı gibi aşamalı bir şekilde oluşturur ve keyfi bir türevlenebilir kayıp fonksiyonun optimizasyonuna izin vererek bunları genelleştirir.

Gradyan Arttırımı Algoritması ;

- Adım 1 : $\{(x_i, y_i)\}_{i=1}^{n}$ şeklinde veri ve türevlenebilir bir kayıp fonksiyonu $L(y_i, F(x))$ tanımlanır.

- Adım 2 : $F_0(x) = \min_{\gamma} \sum_{i=1}^{n} L(y_i, \gamma)$ γ = Tahmin Değeri olacak şekilde başlangıç değeri $(F_0(x))$ minumum olacak şekilde türevlenip 0'a eşitlenir ve $F_0(x)$'in minumum değeri elde edilir.

- Adım 3 :

  - Adım 3.1 : m=1 $\rightarrow$ M'e kadar bir önceki tahmin değerine göre hatalar hesaplanır. (M burada sınıflandırma veya regresyon ağacı sayısıdır)

$$r_{im} = -\left[\frac{\partial L(y_i, F(x_i)}{\partial F(x_i)}\right]_{F(x)=F_{m-1}(x)} \quad i = 1 \rightarrow n$$

  - Adım 3.2 : r_{im} değerlerine göre regresyon veya sınıflandırma ağacı eğitilir ve R_{jm} terminal bölgeleri oluşturulur.

  - Adım 3.3 : Her bir yaprak için çıktı değeri hesaplanır.

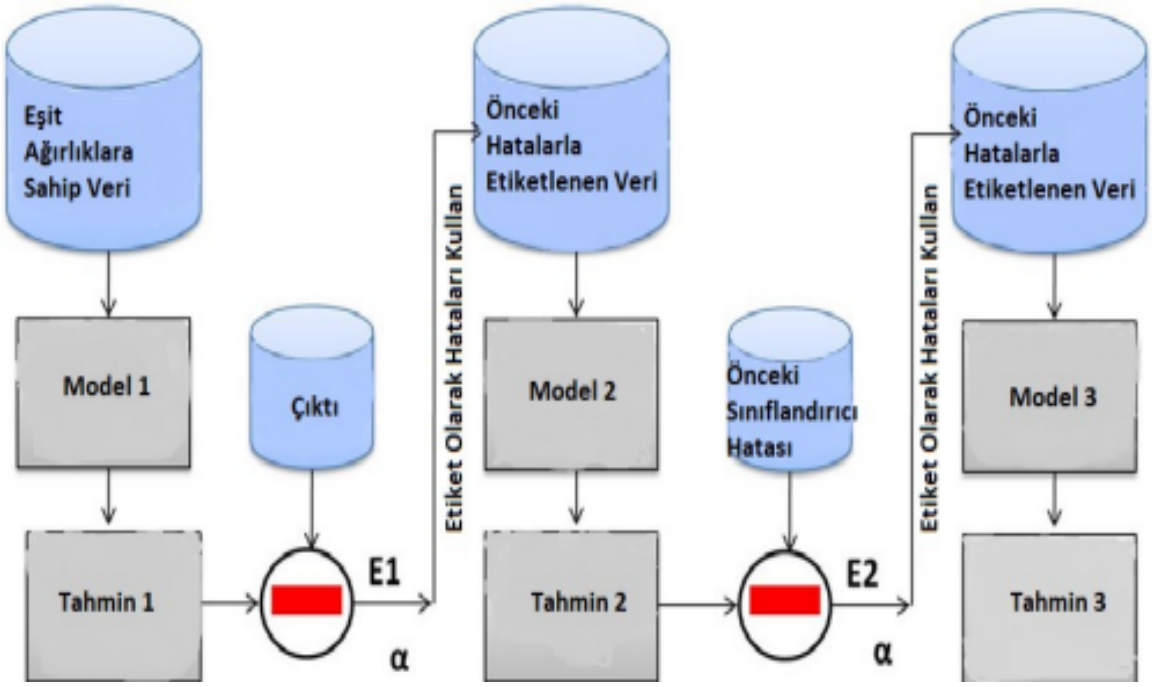
$$\gamma_{jm} = \min_{\gamma} \sum_{x_i \in R_{ij}} L(y_i, F_{m-1}(x_i) + \gamma)$$

    Yine aynı şekilde kayıp fonksiyonunun türevi alınır ve değerler toplanıp sınıfa eşitlenir. Çıkan sayı yaprağın değeridir.

  - Adım 3.4 : Her gözlem için tahmin oluşturulur.

$$F_m(x) = F_{m-1}(x) + \nu \sum_{j=1}^{J_m} \gamma_{jm} I(x \in R_{jm})$$

    Formül incelendiğinde yeni ağacın tahmin değerinin önceki ağacın tahmin değeri + (öğrenme düzeyi $\times$ yeni ağacın değeri) olduğu görülmektedir.



Şekil 2.2: Gradyan Arttırma Çalışma Mantığı (Dikker, 2017)

2.3.2 XGBoost (Aşırı Gradyan Arttırımı)

XGBoost, temeli gradyan arttırımı ve karar ağacı algoritmalarına dayanan bir makine gözetimli öğrenme tekniğidir. XGBoost algoritmasının orijinal hali Friedman tarafından 2002 yılında geliştirilmiştir (Dikker, 2017). Sonrasında Washington Üniversitesi'nde iki araştırmacı olan TianqiChen ve Carlos Guestrin tarafından SIGKDD (Bilgi İşlem Makinalarının Bilgi Keşfi ve Veri Madenciliği Özel İlgi Grubu Derneği) 2016 konferansında makale olarak sunulmuştur ve makine öğrenme dünyasında çok popüler olmuştur (Chen ve Guestrin, 2016).

XGBoost, yüksek tahmin etme gücüne sahip olup, diğer algoritmalardan 10 kat daha hızlıdır. Ayrıca XGBoost, genel performansı iyileştiren ve aşırı uyum ya da aşırı öğrenmeyi azaltan bir dizi düzetlme işlemleri içermektedir (Yangın, 2019).

Gradyan arttırımı, güçlü bir sınıflandırıcı oluşturmak için, bir dizi zayıf sınıflandırıcıyı arttırma ile birleştiren topluluk yöntemidir. Güçlü öğrenici, temel bir öğrenici ile başlayarak yinelemeli olarak eğitilmektedir. Hem gradyan arttırımı hem de XGBoost aynı prensibi izlemektedir. Aralarındaki temel farklar uygulama detaylarında yatmaktadır. XGBoost, farklı düzeltme teknikleri kullanarak, ağaçların karmaşıklığını kontrol ederek daha iyi bir performans elde etmeyi başarmaktadır (Salam Patrous, 2018).

- Adım 1 : $\{(x_i, y_i)\}_{i=1}^{n}$ şeklinde veri ve türevlenebilir bir kayıp fonksiyonu $L(y_i, F(x))$ tanımlanır. XGBoost algoritması kayıp fonksiyonu olarak

$$L(y_i, P_i) = -[y_i log(P_i) + (1 - y_i) log(1 - P_i)]$$

fonksiyonunu kullanır burada P_i tahmin değerine eşittir.

- Adım 2 : $F_0(x) = \min_{\gamma} \sum_{i=1}^{n} L(y_i, \gamma)$, γ = Tahmin Değeri olacak şekilde başlangıç değeri ($F_0(x)$) minumum olacak şekilde türevlenip 0'a eşitlenir ve $F_0(x)$'in minumum değeri elde edilir.

- Adım 3 : Her bir yaprak için optimal çıktı değerini elde etmek amacı ile aşağıdaki denklemden yararlanılır.

$$L(y, P_i + 0_{value}) \approx L(y, P_i) + gO_{value} + \frac{1}{2} hO_{value}^2 \tag{2.1}$$

$$L(y, P_i) = L(y_i, log(Odds)_i) = -y_i log(Odds) + log\left(1 + e^{log(Odds)}\right)$$

$$g = \left[\frac{d}{dlog(Odds)} L(y_i, log(Odds)_i)\right] = -y_i + \frac{e^{log(Odds)_i}}{1 + e^{log(Odds)_i}} = -(y_i - P_i)$$

$$h = \left[\frac{d^2}{dlog(Odds)^2} L(y_i, log(Odds)_i)\right] = \frac{e^{log(Odds)_i}}{(1 + e^{log(Odds)_i})^2} = P_i \times (1 - P_i)$$

  - Adım 3.1 : Denlem 2.1'den yararlanıldığı takdirde aşağıdaki denklem elde edilmektedir.

$$(g_1 + g_2 + \ldots + g_n)O_{value} + \frac{1}{2}(h_1 + h_2 + \ldots + h_n + \lambda)O_{value}^2 \tag{2.2}$$

  - Adım 3.2 : Çıktı değerini elde edebilmek için Denklem 2.2'de elde edilen denklemin O_{value}'ya göre türevi alınarak 0'a eşitlenir.

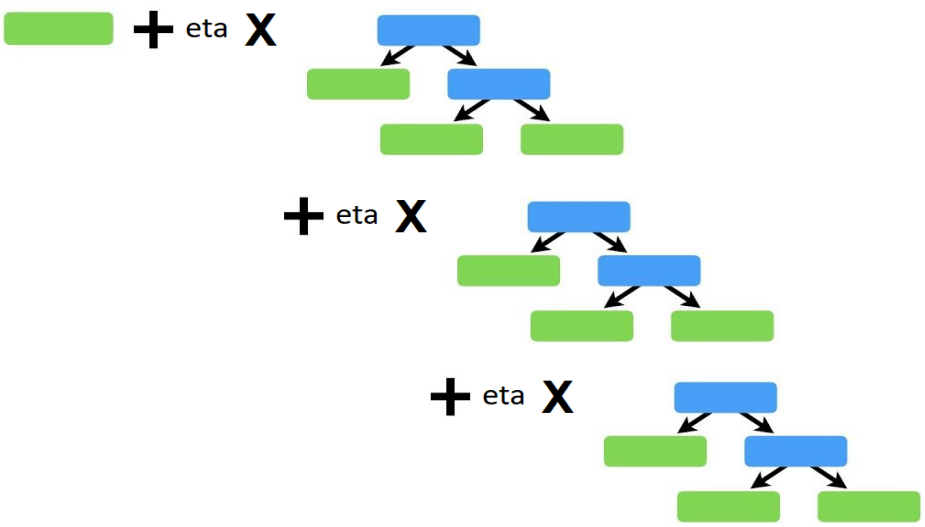
$$\frac{d}{dO_{value}}(g_1 + g_2 + \ldots + g_n)O_{value} + \frac{1}{2}(h_1 + h_2 + \ldots + h_n + \lambda)O_{value}^2 = 0$$

$$O_{value} = \frac{-(g_1 + g_2 + \ldots + g_n)}{(h_1 + h_2 + \ldots + h_n + \lambda)} \tag{2.3}$$

  - Adım 3.3 : Elde edilen Denklem 2.3 yardımı ile benzerlik skoru hesaplanabilir. Bu nokta Benzerlik skoru yardımı ile karar noktasına karar verilir. Benzerlik skoru ne kadar düşükse o karar noktasının veri setini daha keskin bir biçimde ayrıştırdığı düşünülür.

$$\text{Benzerlik Skoru} = \frac{(g_1 + g_2 + \ldots + g_n)^2}{(h_1 + h_2 + \ldots + h_n + \lambda)} \tag{2.4}$$

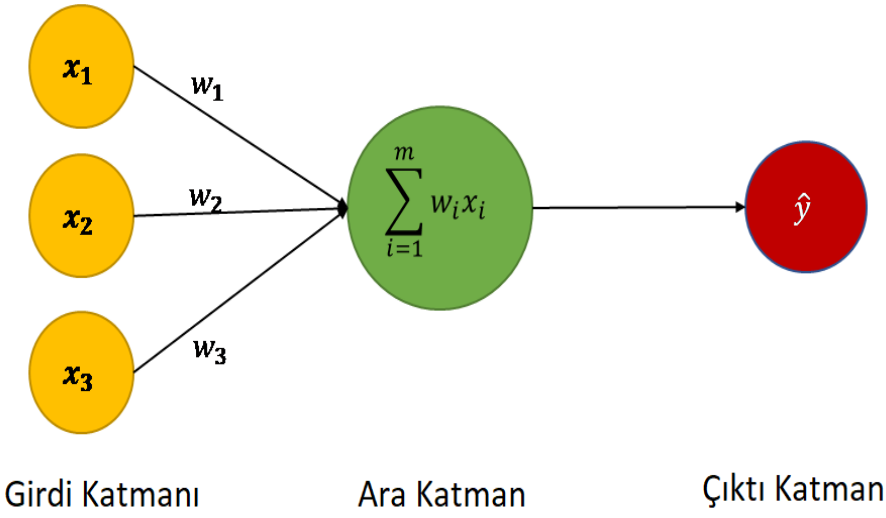
- Adım 4 : Sırası ile Denklem 2.4 ve 2.3, M adet ağaç için hesaplandıktan sonra her ağacın çıktı sonucu eta (öğrenme düzeyi) ile çarpılıp kümülatif olarak toplanır. Son olarak elde edilen ağaçların kümülatif toplamı başta hesaplanan ilk değer ile toplanıp gözlem için tahmin verisi oluşturulur.



Şekil 2.3: XGBoost Çalışma Mantığı

2.4 Yapay Sinir Ağları

Yapay Sinir ağları insan beyninin en temel özelliği olan öğrenme fonksiyonunu gerçekleştiren bilgisayar sistemleridir. Öğrenme işlemini örnekler yardımı ile gerçekleştirirler. Bu ağlar birbirine bağlı yapay sinir hücrelerinden oluşur. Yapay sinir ağları biyolojik sinir sisteminden etkilenerek geliştirilmiştir. Biyolojik sinir hücreleri birbirleri ile sinapsisler vasıtası ile iletişim kurarlar. Bir sinir hücresi işlediği bilgileri Axon'ları yolu ile diğer hücrelere gönderirler (Öztemel, 2003).



Şekil 2.4: Basit Bir Yapay Nöron Örneği (Sahu, 2021)

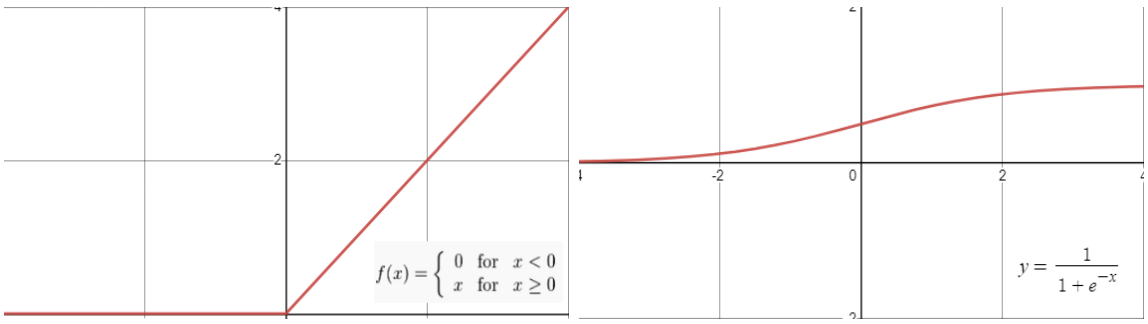
Benzer şekilde yapay sinir hücreleri dışarıdan gelen bilgileri bir toplama fonksiyonu ile toplar ve aktivasyon fonksiyonundan geçirerek çıktıyı üretip ağın bağlantılarının

üzerinden diğer hücrelere gönderir. Temel bir ağ 3 katmandan meydana gelir;

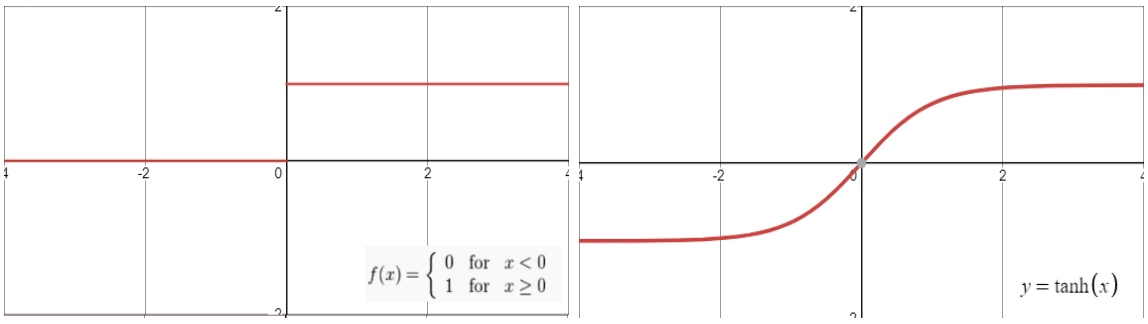
- Girdi Katmanı
- Ara Katmanlar
- Çıktı Katmanı

Her ara katmanın sonunda olabileceği gibi yalnızca çıktı katmanının sonunda da aktivasyon fonksiyonu olabilir. Literatürde en yaygın kullanılan 4 adet aktivasyon fonksiyonu mevcuttur;

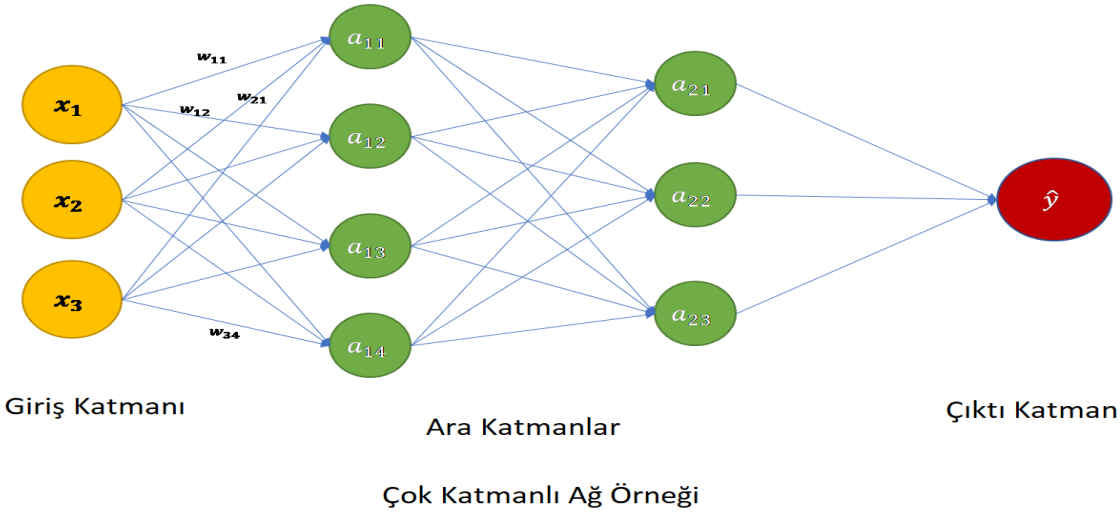
- ReLU
- Sigmoid (Lojistik)
- Step
- tanh



Şekil 2.5: ReLU ve Sigmoid Fonksiyonu

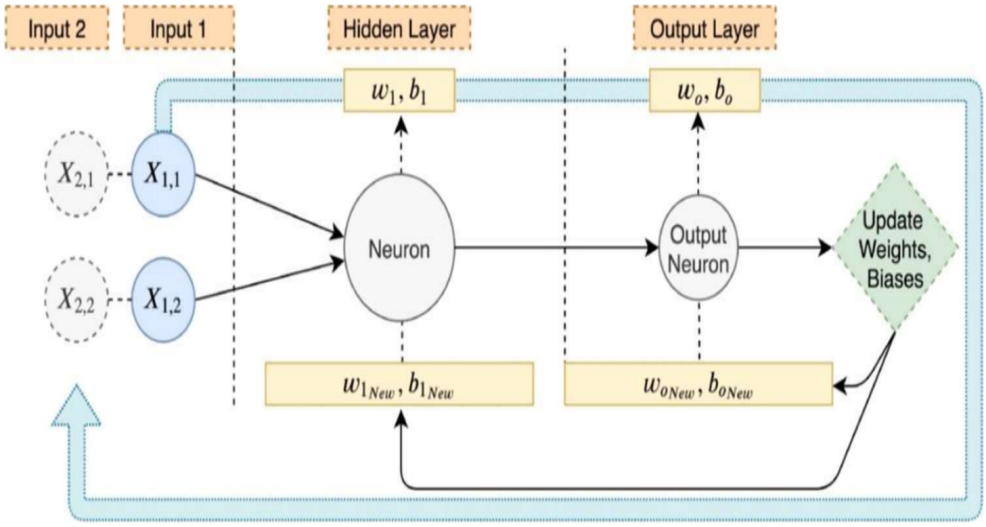


Şekil 2.6: Step ve Tanh Fonksiyonu



Şekil 2.7: Çok Katmanlı Yapay Sinir Ağı Örneği (Sahu, 2021)

Başlangıçta girdi katmanından alınan verilerin ağırlıkları (weights) rastgele olarak atanır ve ağ çalıştırılır. Çıktı katmanından sonra elde edilen sonuç orijinal verinin bağımlı değişkeni ile en yakın değeri üretene kadar sinir ağı geri yayılım yöntemi ile ağırlıkları optimize etmeye çalışır. Maliyet fonksiyonun türevleri alınarak ağırlıkları optimize etmeyi amaçlayan geri yayılım algoritması orijinal verinin bağımlı değişkeni ile tahmin değeri arasında en düşük hata değerini bulduğu zaman ağın yeni ağırlıklarını son bulduğu ağırlıklar olarak tayin eder.



Şekil 2.8: Geri Yaylım Örneği

Ağın ağırlıkları belirlendikten sonra her bir ağırlığın ne anlama geldiği bilinmemektedir. Bu nedenle yapay sinir ağlarına "kara kutu" yakıştırması yapılmaktadır. Ağın performansını etkileyen bağlıca faktörler kullanılan aktivasyon fonksiyonu, öğrenme stratejisi ve öğrenme kuralıdır (Öztemel, 2003).

Bölüm 3

Uygulama

Bu bölümde yapılan uygulama sunulmuştur.

Uygulamada kullanılan veriler 2021 yılında Güney Kore'de yapılan bir araştırmadan alınmıştır (Park ve diğerleri, 2021). Uzmanlar tarafından yapılan tetkikler ile her bir hasta için (her bir el için) ciddiyet sınıflandırılması mild, moderate, severe olarak belirlenmiştir. Çalışmada 1037 elden alınan verilerin 405 (39.05%) adedi erkek hastalardan, 632 (60.95%) adedi kadın hastalardan elde edilmiştir. Uzmanlar tarafından 1037 adet elin, 507 (48.9%) adedi mild, 276 (26.6%) adedi moderate ve 254 (24.5%) adedi severe olarak sınıflandırılmıştır.

Araştırmada uzman hekimler tarafından hastalara yöneltilen sorular ile şikayetleri olan ellerine ilişkin aşağıdaki değişkenler için veri toplanmıştır;

- Hands (Eller)
- Age (Yaş)
- Sex (Cinsiyet)
- BMI (Body-mass index, Vücut kitle indeksi)
- Side (Right side involvement, Sağ Elde Bulgu)
- Diabetes (Diyabet Hastalığı Durumu)
- Duration (Duration in months, Dayanma süresi (Aylık))
- NRS (Numeric rating scale of pain, Hissedilen acının numerik karşılığı)
- NP (Noctural Pain, Gece Ağrıları)
- Weakness (Tenar weakness and/or atrophy, Avuç İçi Zayıflık ve/veya Körelme)
- CSA (Cyclosporine dosage in mm^2, Siklosporin dozu (mm^2))
- PB (fexor retinaculum in mm, Fleksör retinakulum (mm))

Tablo 3.1: Sayısal Değişkenlerin Tanımlayıcı İstatistikleri

	Overall
Age,years (mean ± SD)	58 ± 10.8
BMI, kg/m^2 (mean ± SD)	24.8 ± 3.4
Duration, months (mean ± SD)	8.3 ± 9.6
NRS (mean ± SD)	4.4 ± 1.8
CSA, mm^2 (mean ± SD)	15.2 ± 4.3
PB, mm (mean ± SD)	2.5 ± 1.8

Tablo 3.2: Değişkenlerin Bağımlı Değişkene Göre Tanımlayıcı İstatistikleri (P-Value Değerleri Tek Yönlü Varyans Analiz Testi ile Elde Edilmiştir.)

	Mild	Moderate	Severe	P Value
Age,years (mean ± SD)	57.3 ± 10.6	59.2 ± 10.8	57.8 ± 11.2	0.069
BMI, kg/m^2 (mean ± SD)	24.2 ± 3.4	24.7 ± 3	25.8 ± 3.7	0
Duration, months (mean ± SD)	4.3 ± 5	8.5 ± 8.2	15.9 ± 12.8	0
NRS (mean ± SD)	3.3 ± 1.3	4.9 ± 1.5	6.1 ± 1.5	0
CSA, mm^2 (mean ± SD)	13.2 ± 3	15.4 ± 3.2	18.9 ± 5	0
PB, mm (mean ± SD)	2.1 ± 0.8	2.6 ± 2.4	3.1 ± 2.3	0

Tablo 3.3: Katagorik Değişkenlerin Bağımlı Değişkence Frekans Dağılımı (P-Value Değerleri Ki-Kare Bağımsızlık Testi ile Elde Edilmiştir.)

	Mild	Moderate	Severe	P value
Eller, n (%)	507 (48.9)	276 (26.6)	254 (24.5)	-
Cinsiyet (Kadın), n (%)	308 (60.7)	153 (55.4)	171 (67.3)	0.02
Sağ Elde Bulgu, n (%)	243 (47.9)	149 (54)	119 (46.9)	0.181
Diyabet, n (%)	47 (9.3)	45 (16.3)	54 (21.3)	0
Gece Ağrıları, n (%)	102 (20.1)	142 (51.4)	212 (83.5)	0
Avuç İçi Zayıflık ve/veya Körelme, n (%)	1 (0.2)	24 (8.7)	169 (66.5)	0

Kullanılan verilere ait tanımlayıcı istatistikler Tablo 3.1, Tablo 3.2 ve Tablo 3.3 de verilmiştir. Varsayım kontrollerinin ardından sayısal değişkenlerin ciddiyet sınıflarına göre farklılık gösterip göstermediği tek yönlü varyans analizi ile araştırılmıştır ($\alpha = 0.05$). Tablo 3.2'de yer alan P-Value değerleri incelendiğinde ciddiyet sınıflamasının yaşa göre anlamlı bir değişim göstermediği diğer tüm değişkenler için anlamlı bir fark olduğu görülmüştür.

Tablo 3.3'de yer alan P-Value değerleri incelendiğinde ciddiyet sınıflamasının KTS'nin sağ veya sol elde görülmesine göre anlamlı bir değişim göstermediği diğer tüm değişkenler için anlamlı bir fark olduğu görülmüştür.

Bu çalışmada, KTS ciddiyet sınıflandırması için K-En Yakın Komşuluk, Rassal Ormanlar, Yapay Sinir Ağları ve XGBoost yöntemleri kullanılmıştır.
İlk olarak orijinal verilerdeki 3 sınıf (Mild, Moderate, Severe) için sınıflandırma hedeflenmiştir.
Bölüm 3.1'de bu problem için farklı modeller ile elde edilen sonuçlara yer verilmiştir.
Uygulamanın ikinci bölümünde hedef değişken iki sınıfa indirgenmiş ve bu probleme ait sonuçlar bölüm 3.2'de paylaşılmıştır.
Uygulamada Python programlama dili ve Scikit-Learn, Pandas, Numpy, Matplotlib,XGBoost kütüphanelerinden yararlanılmıştır.
Bu bölümde modellerin performanslarını değerlendirmek üzere kesinlik, duyarlılık, F1-skoru, doğruluk oranı, dengelenmiş doğruluk oranı hesaplanmıştır.

3.1 Çok Sınıflı(Multiclass) Sınıflama Problemi

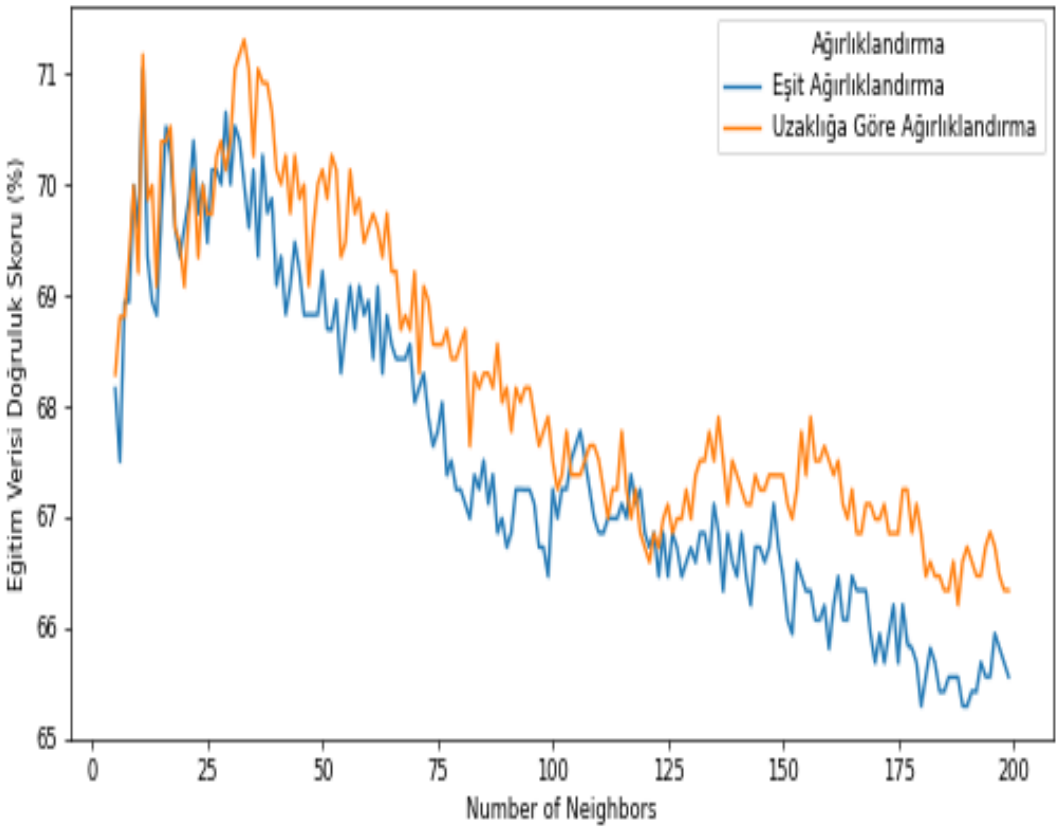
Bu bölümde KTS ciddiyet sınıflandırması için hedef değişkenin 3 farklı ciddiyet düzeyine sahip olduğu durumda farklı sınıflama algoritmaları ile ciddiyet düzeyinin tahminlenmesi amaçlanmıştır.

3.1.1 K-En Yakın Komşuluk Modeli

Bu bölümde veri seti üzerinde K - En yakın komşuluk modeli kullanılmış ve çıktıları değerlendirilmiştir.

Hiper Parametre Seçimi

Daha önce belirlenen parametre uzayını ve Scikit-Learn kütüphanesinde bulunan GridSearchCV algoritması ile en yüksek doğruluk oranı yakalanana kadar çalışması sağlanmıştır.



Şekil 3.1: Üç Sınıflı K-NN Eğitim Verisi Doğruluk Skorları

K-En yakın komşuluk modeli için en yüksek doğruluk oranı aşağıdaki parametreler ile bulunmuştur;

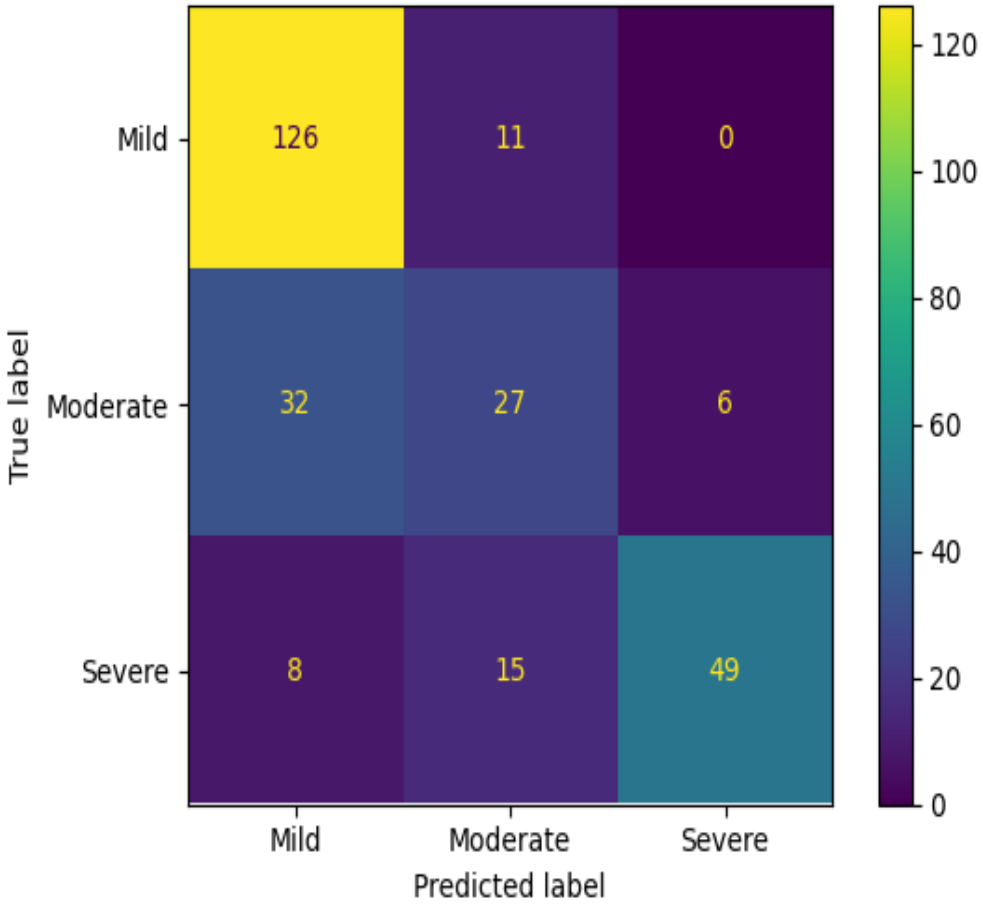
- 'algorithm':'auto'
- 'n_neighbors':33
- 'weights':'distance'

En İyi Parametreli Model

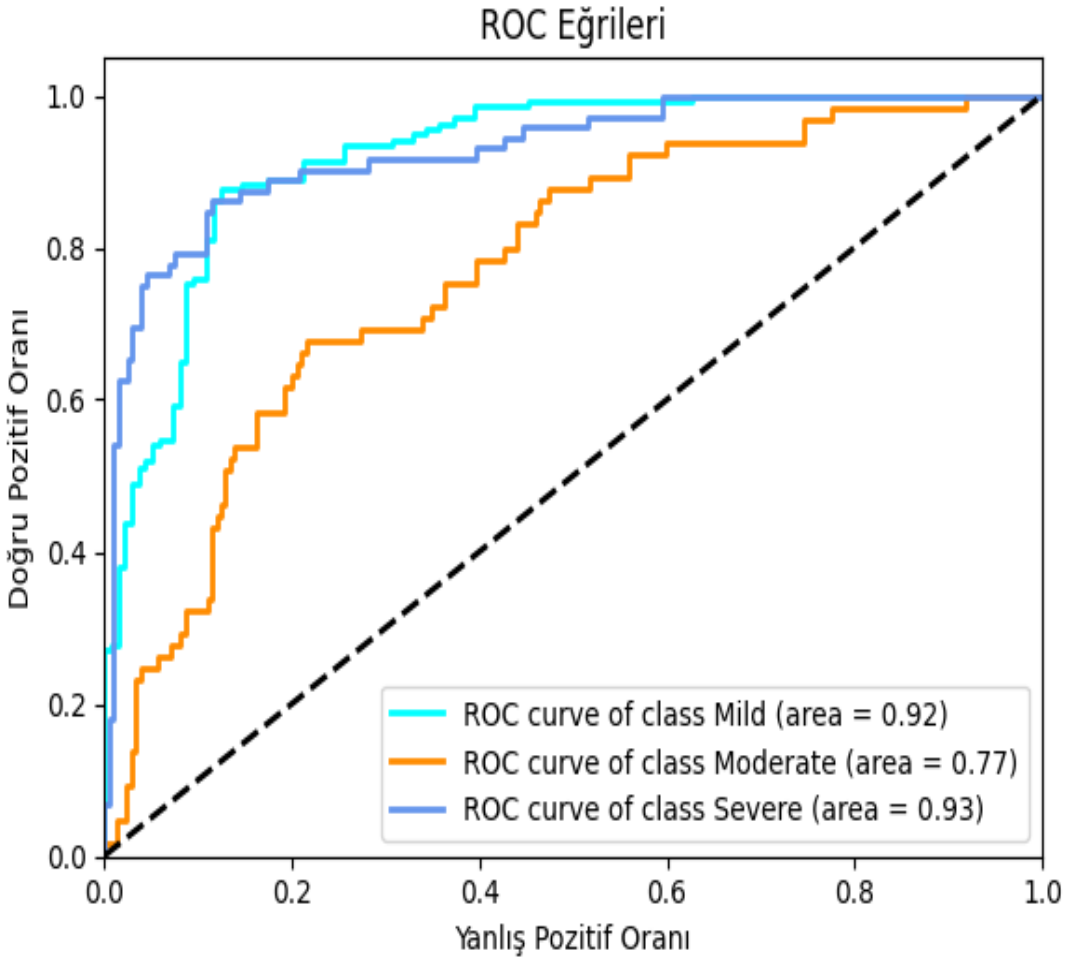
Bulunan parametrelerle kurulan modelin sınıflandırma metrikleri aşağıdaki gibidir.

	precision	recall	f1-score	support
Mild	0.76	0.92	0.83	137
Moderate	0.51	0.42	0.46	65
Severe	0.89	0.68	0.77	72
accuracy			0.74	274
macro avg	0.72	0.67	0.69	274
weighted avg	0.73	0.74	0.73	274

Balanced Accuracy Score : 0.6718827333790838



Şekil 3.2: Üç Sınıflı K-NN Modeli Karmaşıklık Matrisi



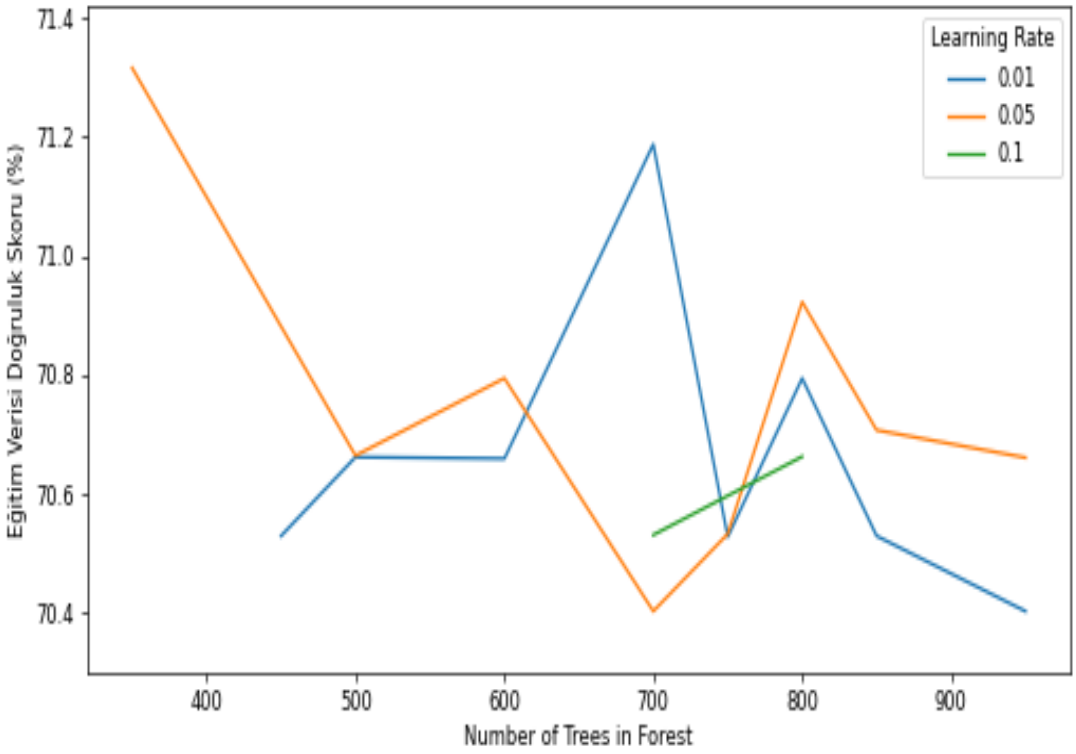
Şekil 3.3: Üç Sınıflı K-NN Modeli ROC Eğrisi ve AUC Değerleri

3.1.2 Rassal Ormanlar Modeli

Bu bölümde veri seti üzerinde rassal ormanlar modeli kullanılmış ve çıktıları değerlendirilmiştir.

Hiper Parametre Seçimi

Daha önce belirlenen parametre uzayını ve Scikit-Learn kütüphanesinde bulunan GridSearchCV algoritması ile en yüksek doğruluk oranı yakalanana kadar çalışması sağlanmıştır.



Şekil 3.4: Üç Sınıflı Rassal Ormanlar Modeli Eğitim Verisi Doğruluk Skorları

Rassal ormanlar modeli için en yüksek doğruluk oranı aşağıdaki parametreler ile bulunmuştur;

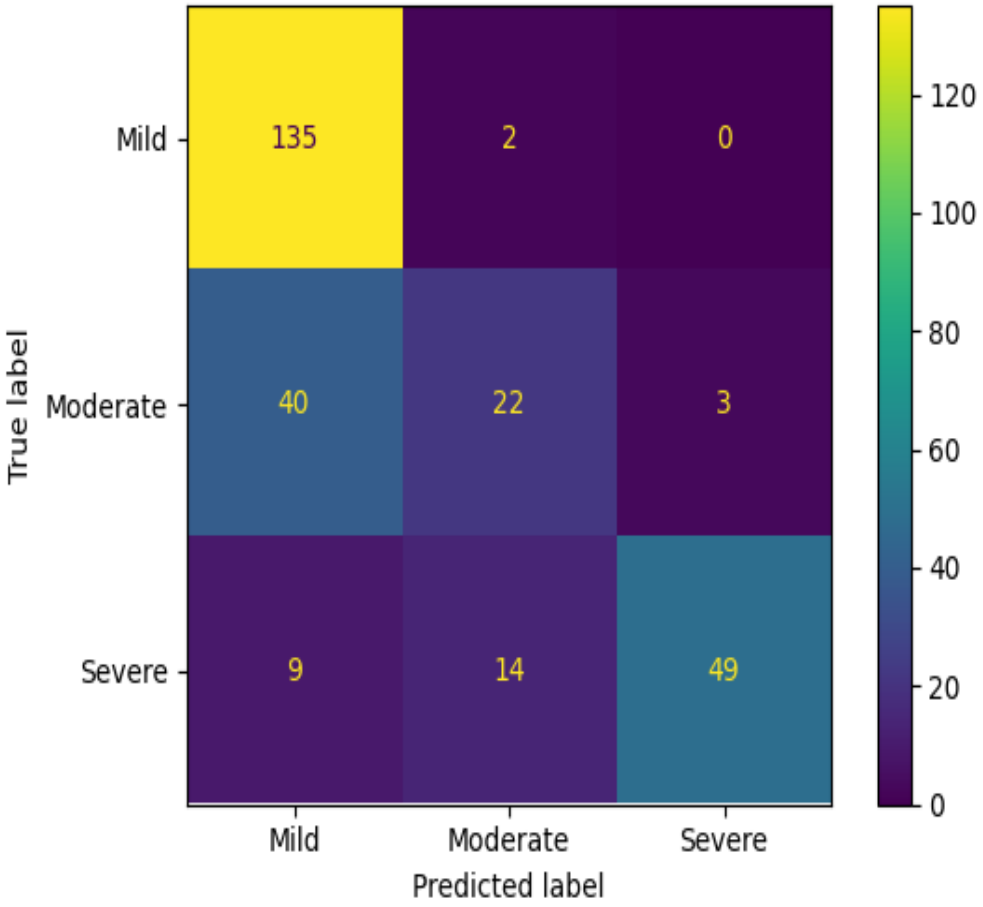
- 'ccp_alpha':0.05
- 'criterion':'gini'
- 'weights':'distance'
- 'max_features':'auto'
- 'max_samples':10
- 'n_estimators':350

En İyi Parametreli Model

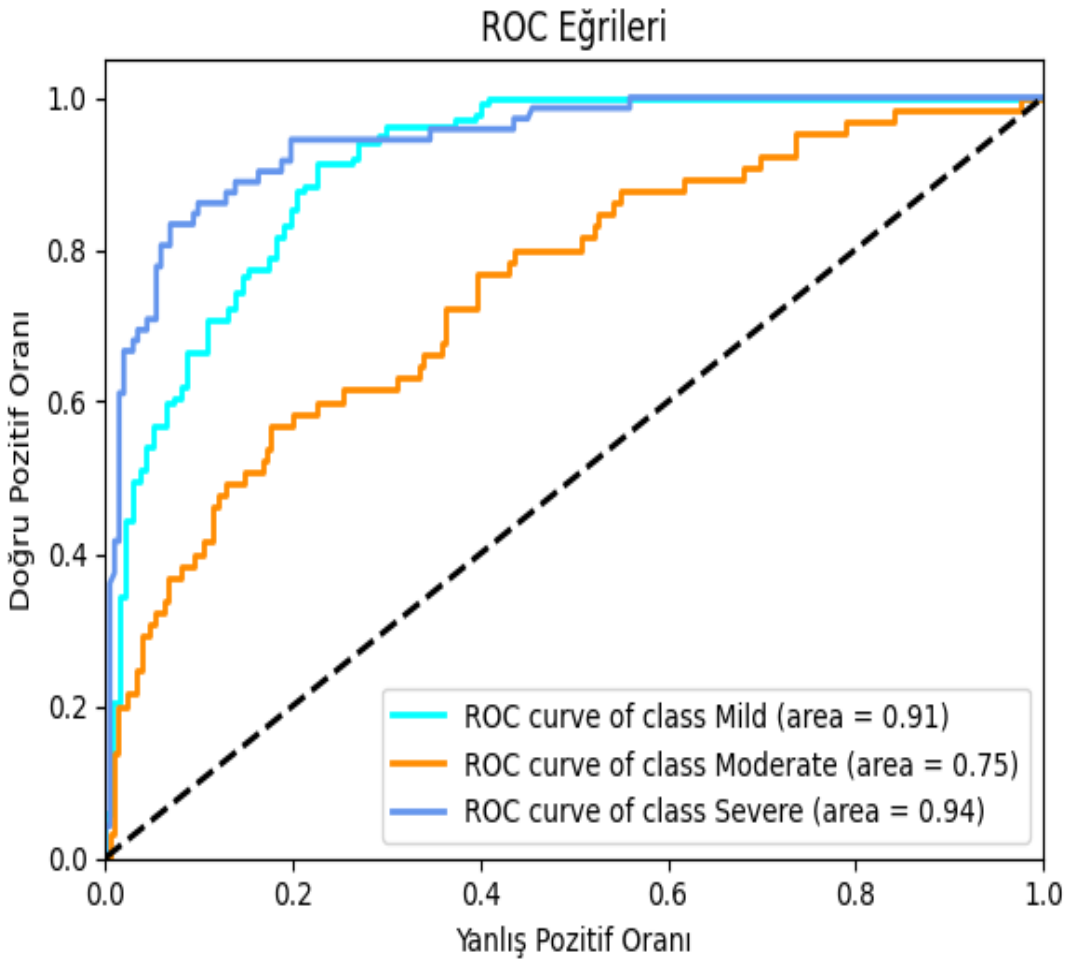
Bulunan parametrelerle kurulan modelin sınıflandırma metrikleri aşağıdaki gibidir.

	precision	recall	f1-score	support
Mild	0.73	0.99	0.84	137
Moderate	0.58	0.34	0.43	65
Severe	0.94	0.68	0.79	72
accuracy			0.75	274
macro avg	0.75	0.67	0.69	274
weighted avg	0.75	0.75	0.73	274

```
Balanced Accuracy Score : 0.6681395179570361
```



Şekil 3.5: Üç Sınıflı Rassal Ormanlar Modeli Karmaşıklık Matrisi



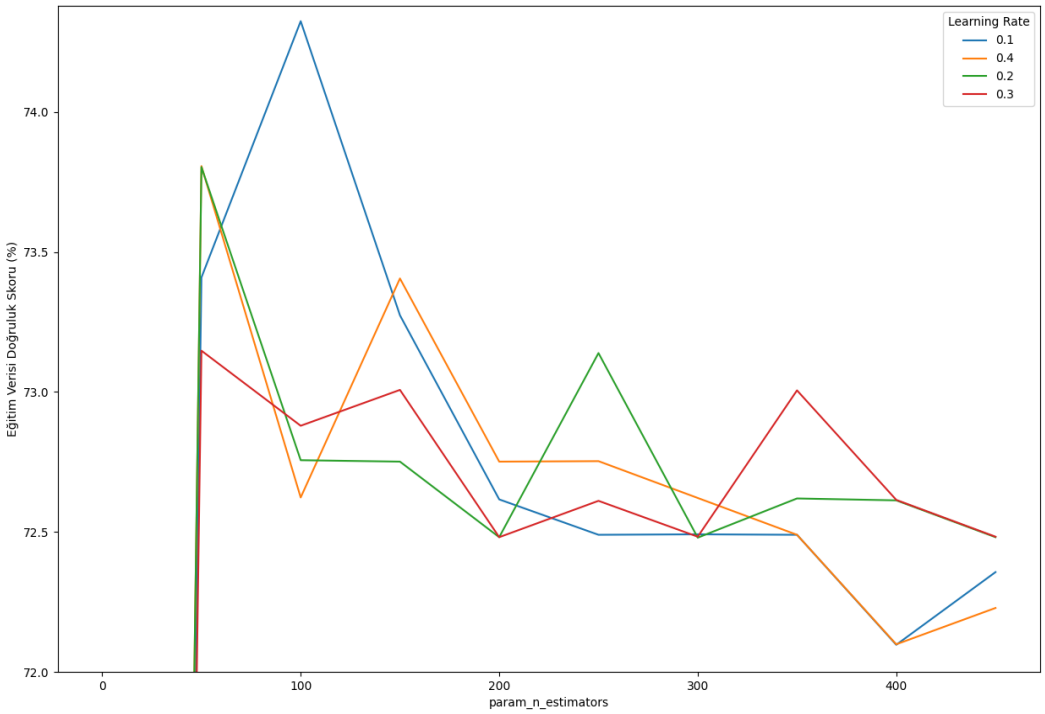
Şekil 3.6: Üç Sınıflı Rassal Ormanlar Modeli ROC Eğrisi ve AUC Değerleri

3.1.3 eXtreme Gradient Boosting (XGBoost)

Bu bölümde veri seti üzerinde XGBoost modeli kullanılmış ve çıktıları değerlendirilmiştir.

Hiper Parametre Seçimi

Daha önce belirlenen parametre uzayını ve Scikit-Learn kütüphanesinde bulunan GridSearchCV algoritması ile en yüksek doğruluk oranı yakalanana kadar çalışması sağlanmıştır.



Şekil 3.7: Üç Sınıflı XGBoost Modeli Eğitim Verisi Doğruluk Skorları

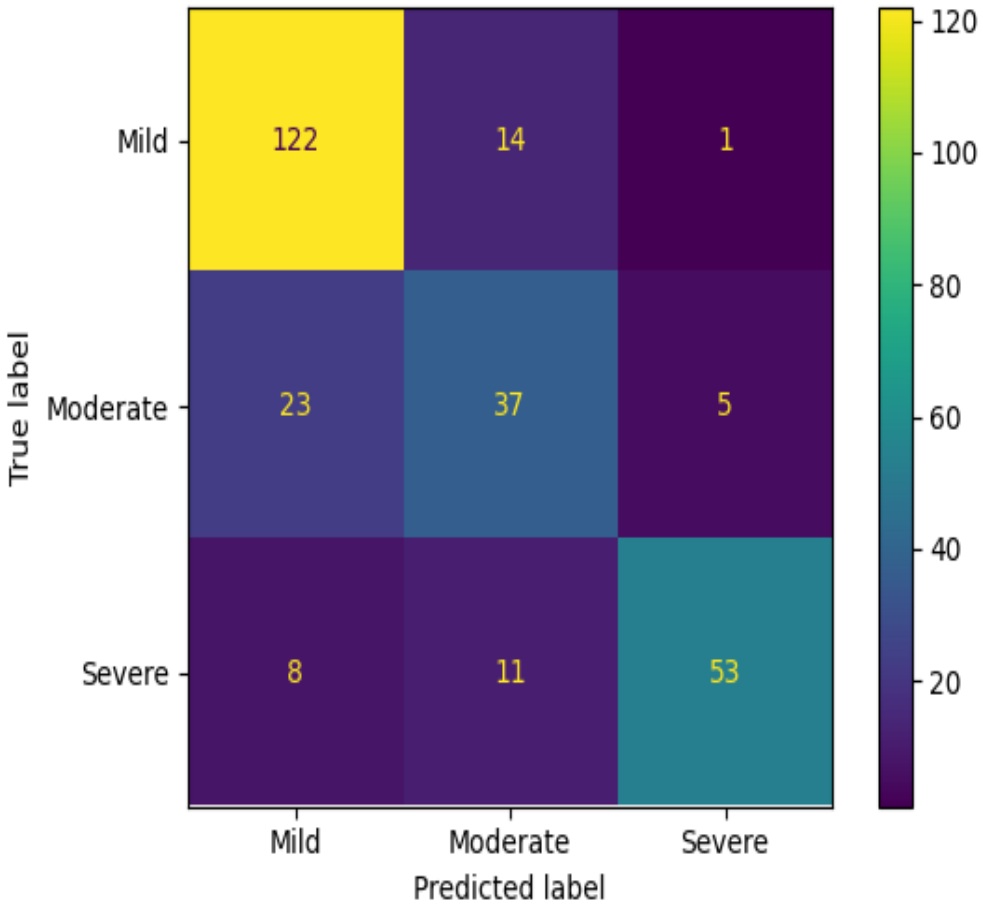
XGBoost modeli için en yüksek doğruluk oranı aşağıdaki parametreler ile bulunmuştur;

- 'eta':0.1
- 'max_depth':3
- 'min_child_weight':10
- 'n_estimators':100
- 'objective':'multi:softprob'
- 'sumsample':0.5

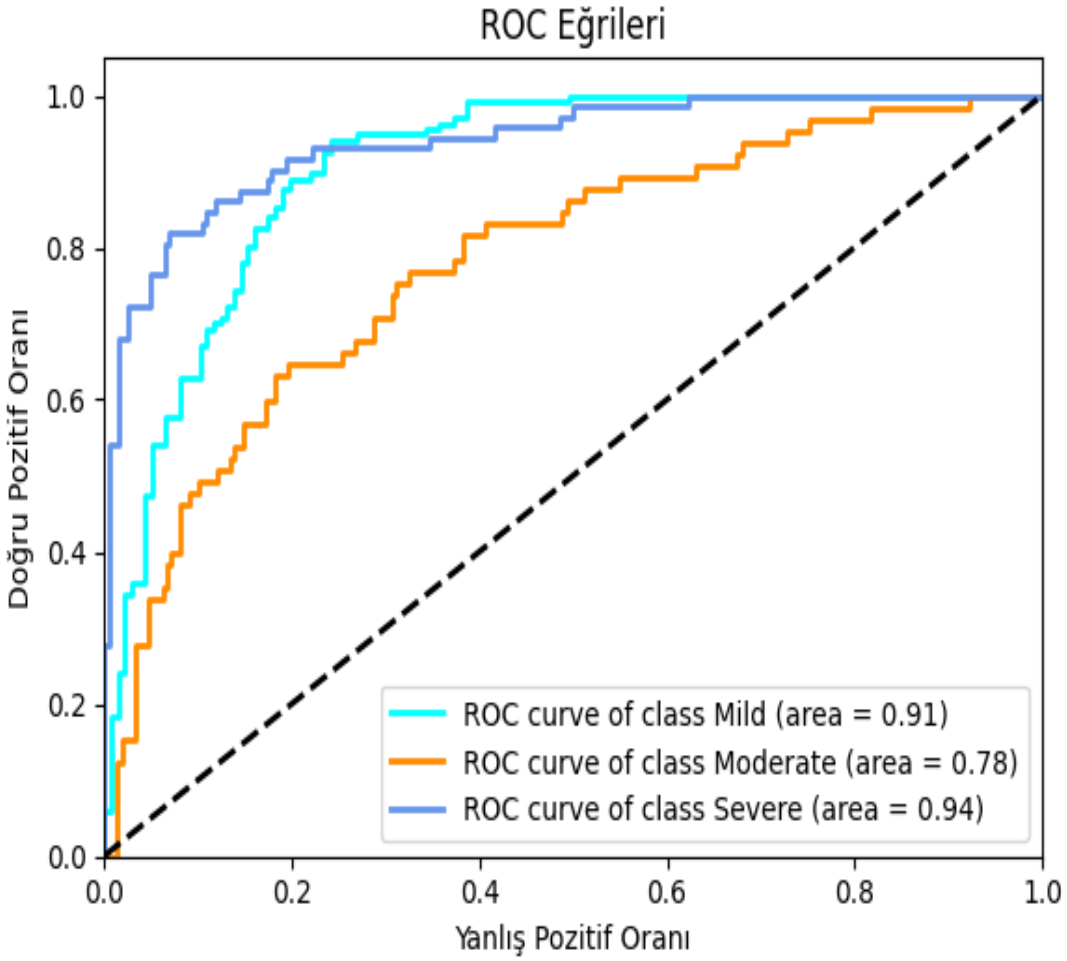
Bulunan parametrelerle kurulan modelin sınıflandırma metrikleri aşağıdaki gibidir.

	precision	recall	f1-score	support
Mild	0.80	0.89	0.84	137
Moderate	0.60	0.57	0.58	65
Severe	0.90	0.74	0.81	72
accuracy			0.77	274
macro avg	0.76	0.73	0.74	274
weighted avg	0.78	0.77	0.77	274

```
Balanced Accuracy Score : 0.7319509430823299
```



Şekil 3.8: Üç Sınıflı XGBoost Modeli Karmaşıklık Matrisi



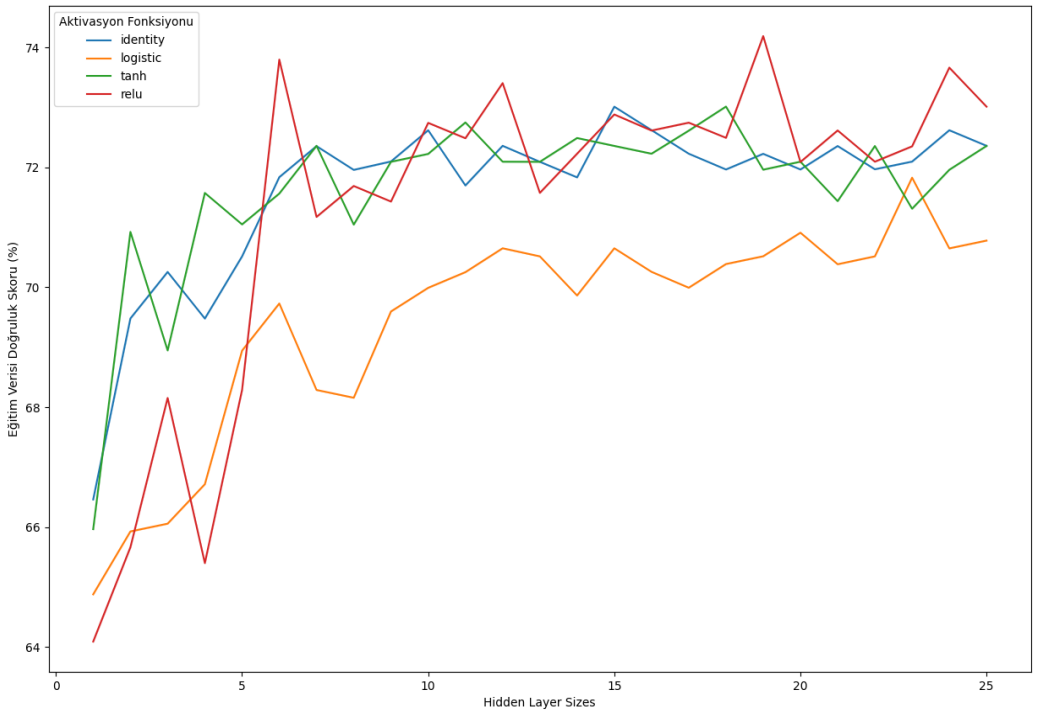
Şekil 3.9: Üç Sınıflı XGBoost Modeli ROC Eğrisi ve AUC Değerleri

3.1.4 Yapay Sinir Ağları (Neural Networks)

Bu bölümde veri seti üzerinde yapay sinir ağları modeli kullanılmış ve çıktıları değerlendirilmiştir.

Hiper Parametre Seçimi

Daha önce belirlenen parametre uzayını ve Scikit-Learn kütüphanesinde bulunan GridSearchCV algoritması ile en yüksek doğruluk oranı yakalanana kadar çalışması sağlanmıştır.



Şekil 3.10: Üç Sınıflı Yapay Sinir Ağları Modeli Eğitim Verisi Doğruluk Skorları

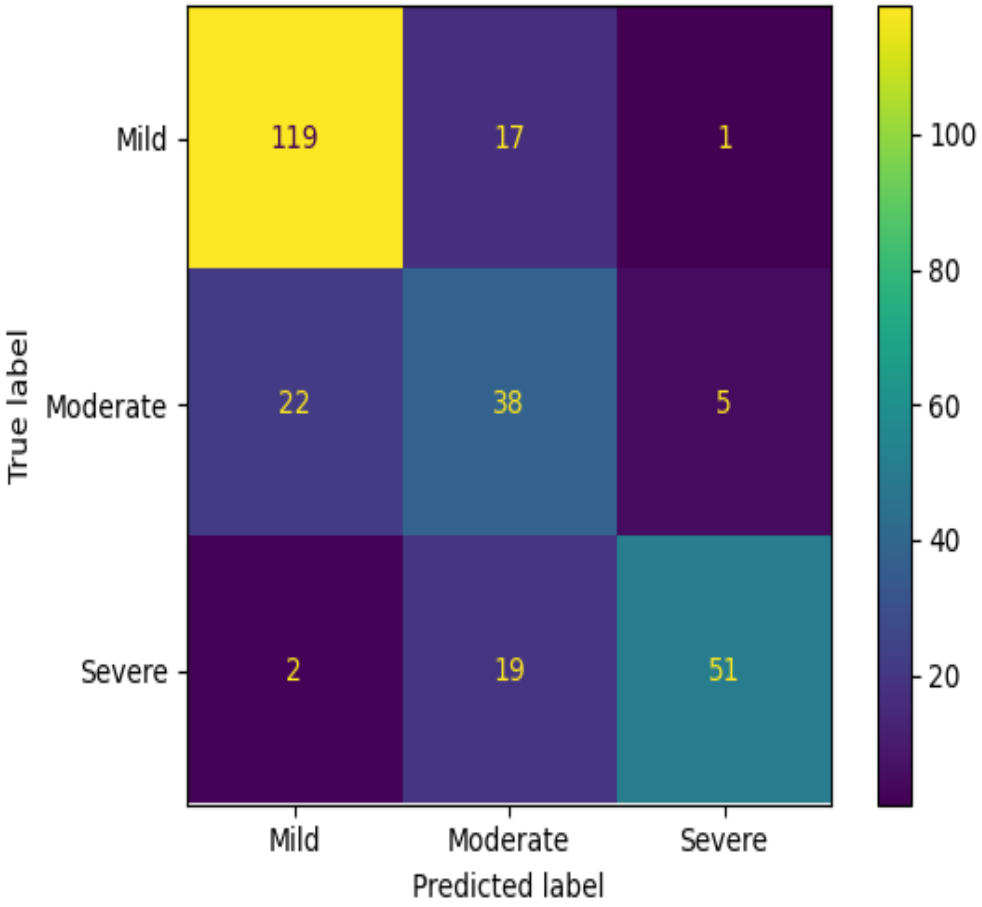
Yapay sinir ağları modeli için en yüksek doğruluk oranı aşağıdaki parametreler ile bulunmuştur;

- 'activation':'relu'
- 'hidden_layer_sizes:19
- 'learning_rate':'adaptive'

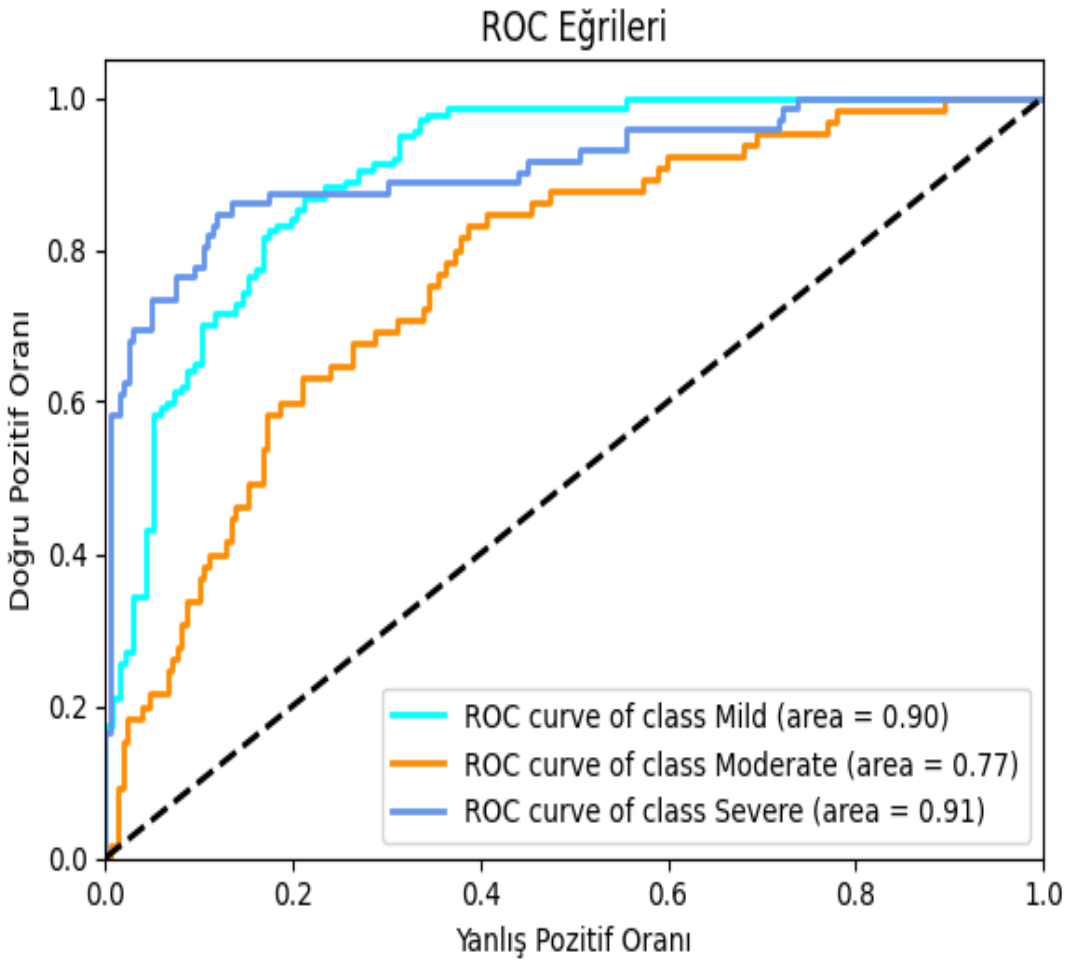
Bulunan parametrelerle kurulan modelin sınıflandırma metrikleri aşağıdaki gibidir.

	precision	recall	f1-score	support
Mild	0.83	0.87	0.85	137
Moderate	0.51	0.58	0.55	65
Severe	0.89	0.71	0.79	72
accuracy			0.76	274
macro avg	0.75	0.72	0.73	274
weighted avg	0.77	0.76	0.76	274

```
Balanced Accuracy Score : 0.7205206188782832
```



Şekil 3.11: Üç Sınıflı Yapay Sinir Ağları Modeli Karmaşıklık Matrisi



Şekil 3.12: Üç Sınıflı Yapay Sinir Ağları Modeli ROC Eğrisi ve AUC Değerleri

3.1.5 Çok Sınıflı Sınıflama Probleminin Modellerinin Değerlendirdirilmesi

Bölüm 3.1.1, 3.1.2, 3.1.3 ve 3.1.4'den elde edilen sonuçlar incelenmiş olup, üç sınıflı problem için %77 doğru sınıflama oranı ile en iyi model XGBoost olarak bulunmuştur. Bölüm 3.1.3'de bulunan performans metrikleri yakından incelendiğinde, duyarlılık metriği 'Moderate' ve 'Severe' sınıfları için 'Mild' sınıfına kıyasla daha düşük kalmıştır. Duyarlılık metriğindeki düşüklüğün sebep olabileceği yanlış sınıflandırmaların önüne geçebilmek amacı ile bölüm 3.2'de problem iki sınıflı probleme indirgenecek ve modeller tekrar çalıştırılacaktır.

3.2 İki Sınıflı Sınıflama

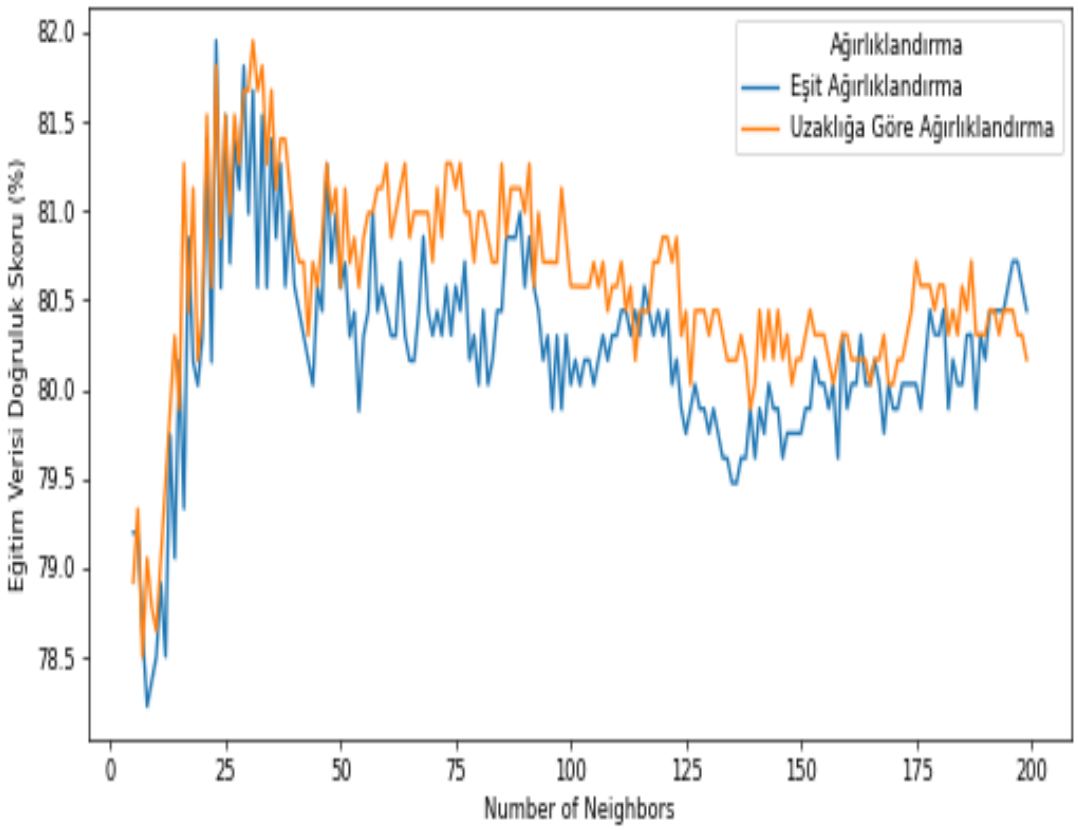
Bu bölümde KTS ciddiyet sınıflandırması için hedef değişkenin 3 farklı ciddiyet düzeyine sahip olduğu veri seti 'Moderate' ve 'Severe' ciddiyet düzeyleri birleştirilerek problemin 2 farklı ciddiyet düzeyine indirgenip farklı sınıflama algoritmaları ile ciddiyet düzeyinin tahminlenmesi amaçlanmıştır.

3.2.1 K-En Yakın Komşuluk Modeli

Bu bölümde veri seti üzerinde K - En yakın komşuluk modeli kullanılmış ve çıktıları değerlendirilmiştir.

Hiper Parametre Seçimi

Daha önce belirlenen parametre uzayını ve Scikit-Learn kütüphanesinde bulunan GridSearchCV algoritması ile en yüksek doğruluk oranı yakalanana kadar çalışması sağlanmıştır.



Şekil 3.13: İki Sınıflı K-NN Modeli Eğitim Verisi Doğruluk Skorları

K-En yakın komşuluk modeli için en yüksek doğruluk oranı aşağıdaki parametreler ile bulunmuştur;

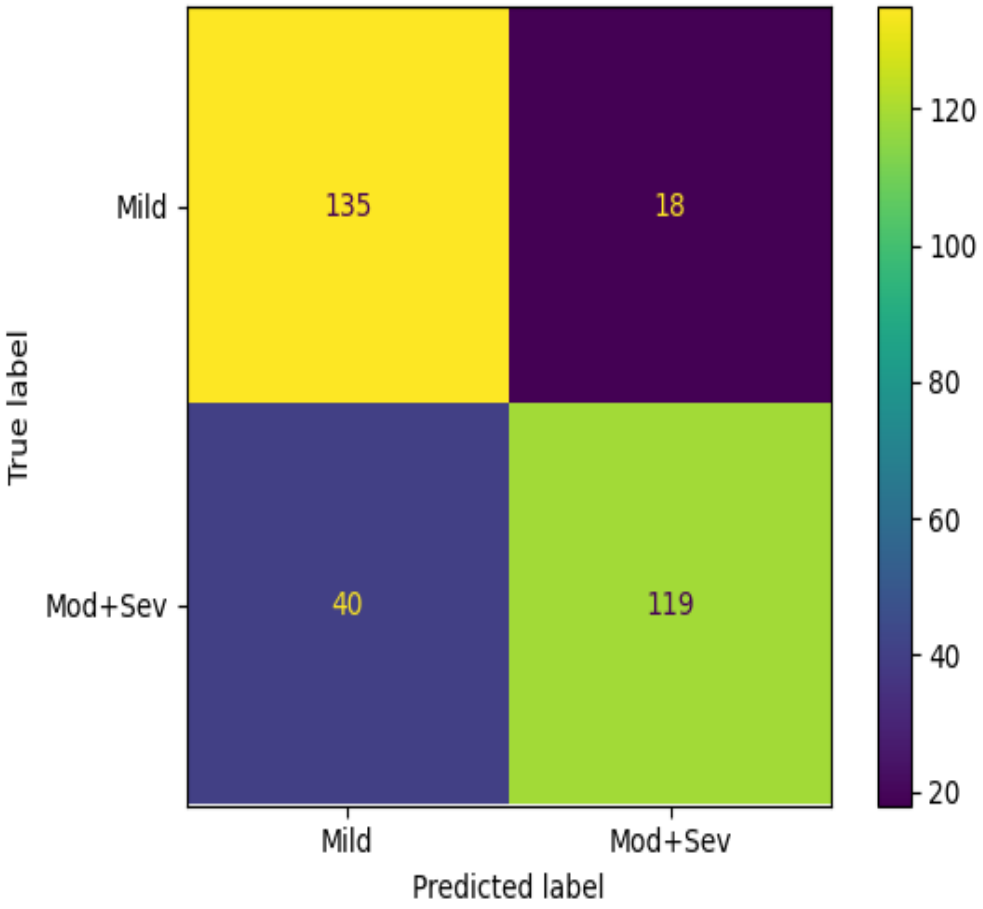
- 'algorithm':'auto'
- 'n_neighbors':23
- 'weights':'uniform'

En İyi Parametreli Model

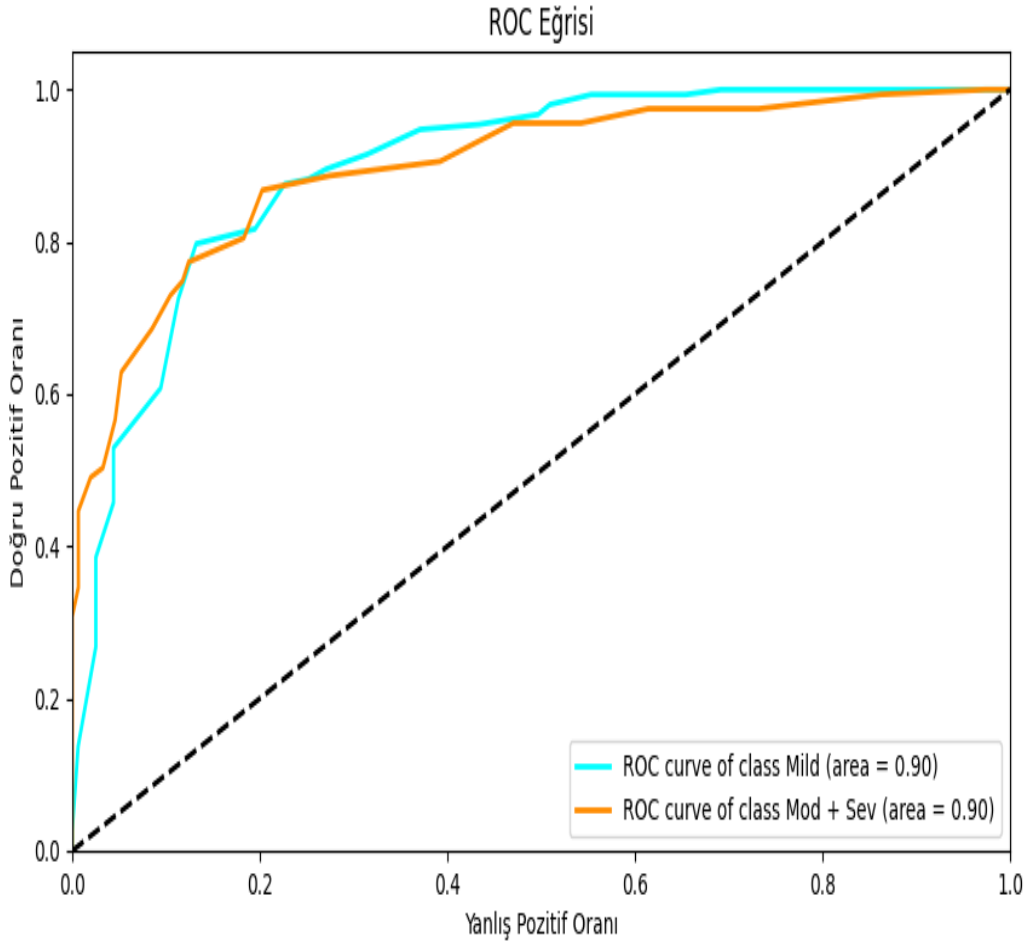
Bulunan parametrelerle kurulan modelin sınıflandırma metrikleri aşağıdaki gibidir.

	precision	recall	f1-score	support
Mild	0.77	0.88	0.82	153
Mod+Sev	0.87	0.75	0.80	159
accuracy			0.81	312
macro avg	0.82	0.82	0.81	312
weighted avg	0.82	0.81	0.81	312

```
Balanced Accuracy Score : 0.8153903070662227
```



Şekil 3.14: İki Sınıflı K-NN Modeli Karmaşıklık Matrisi



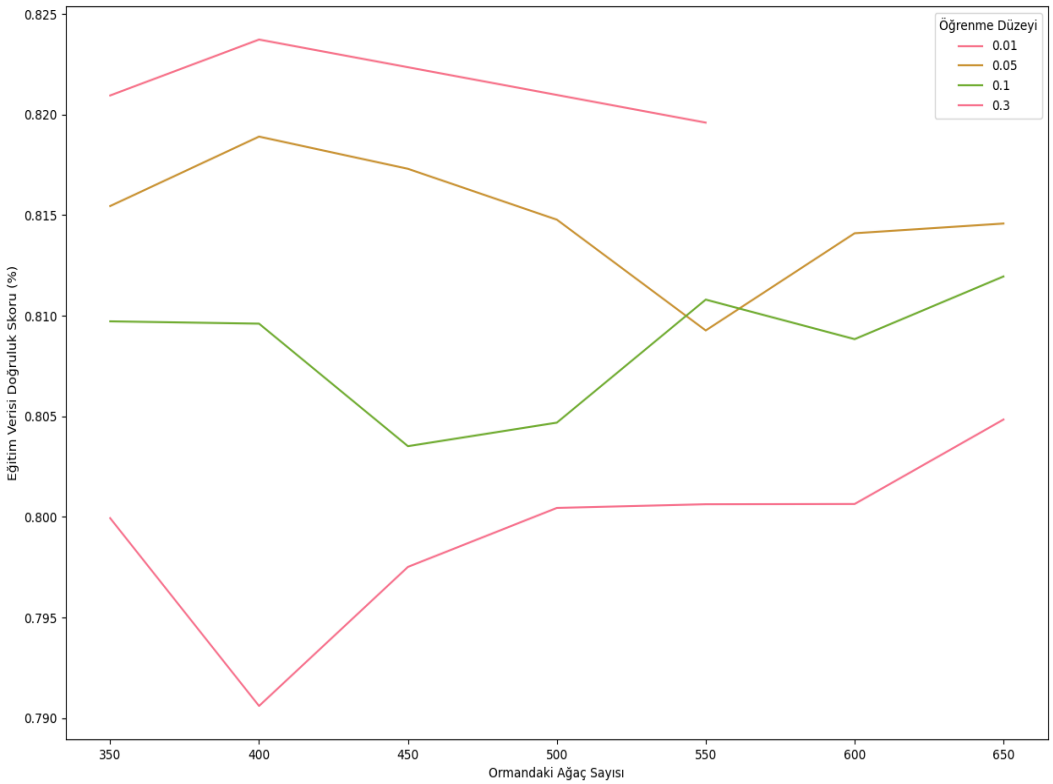
Şekil 3.15: İki Sınıflı K-NN Modeli ROC Eğrisi ve AUC Değerleri

3.2.2 Rassal Ormanlar Modeli

Bu bölümde veri seti üzerinde rassal ormanlar modeli kullanılmış ve çıktıları değerlendirilmiştir.

Hiper Parametre Seçimi

Daha önce belirlenen parametre uzayını ve Scikit-Learn kütüphanesinde bulunan GridSearchCV algoritması ile en yüksek doğruluk oranı yakalanana kadar çalışması sağlanmıştır.



Şekil 3.16: İki Sınıflı Rassal Ormanlar Modeli Eğitim Verisi Doğruluk Skorları

Rassal ormanlar modeli için en yüksek doğruluk oranı aşağıdaki parametreler ile bulunmuştur;

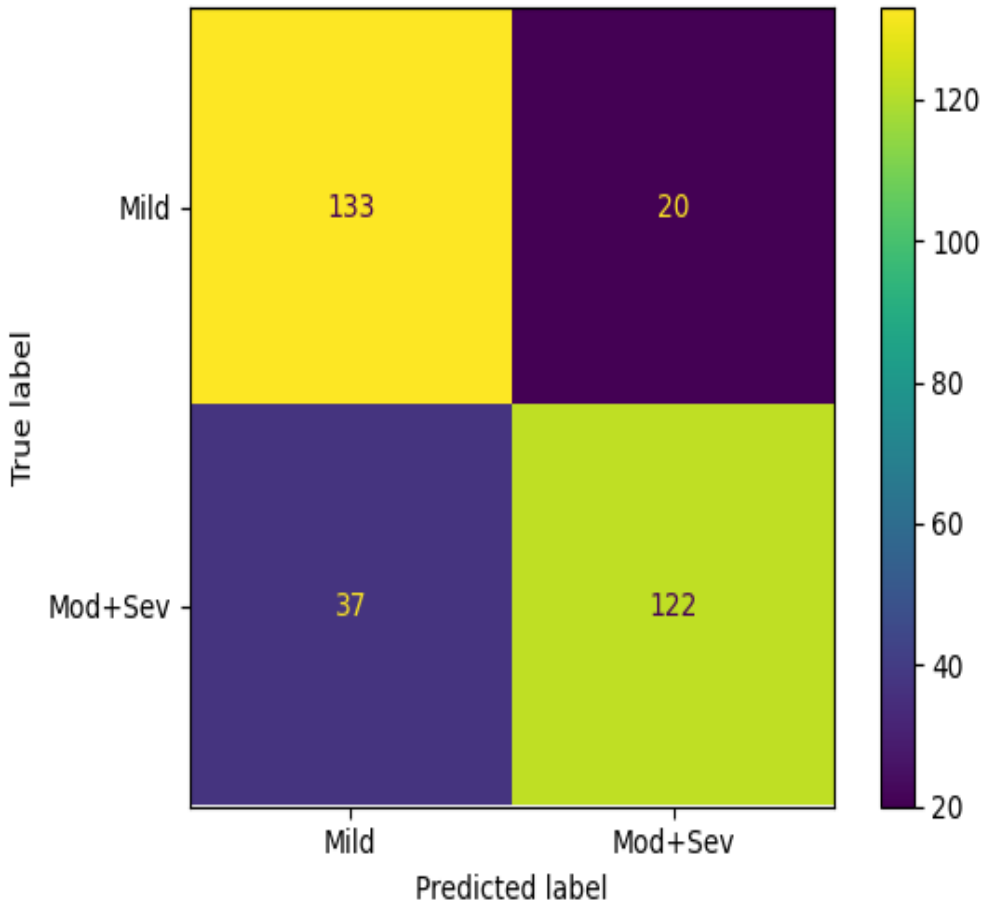
- 'ccp_alpha': 0.01
- 'max_features': 'auto'
- 'max_samples': 10
- 'n_estimators': 400

En İyi Parametreli Model

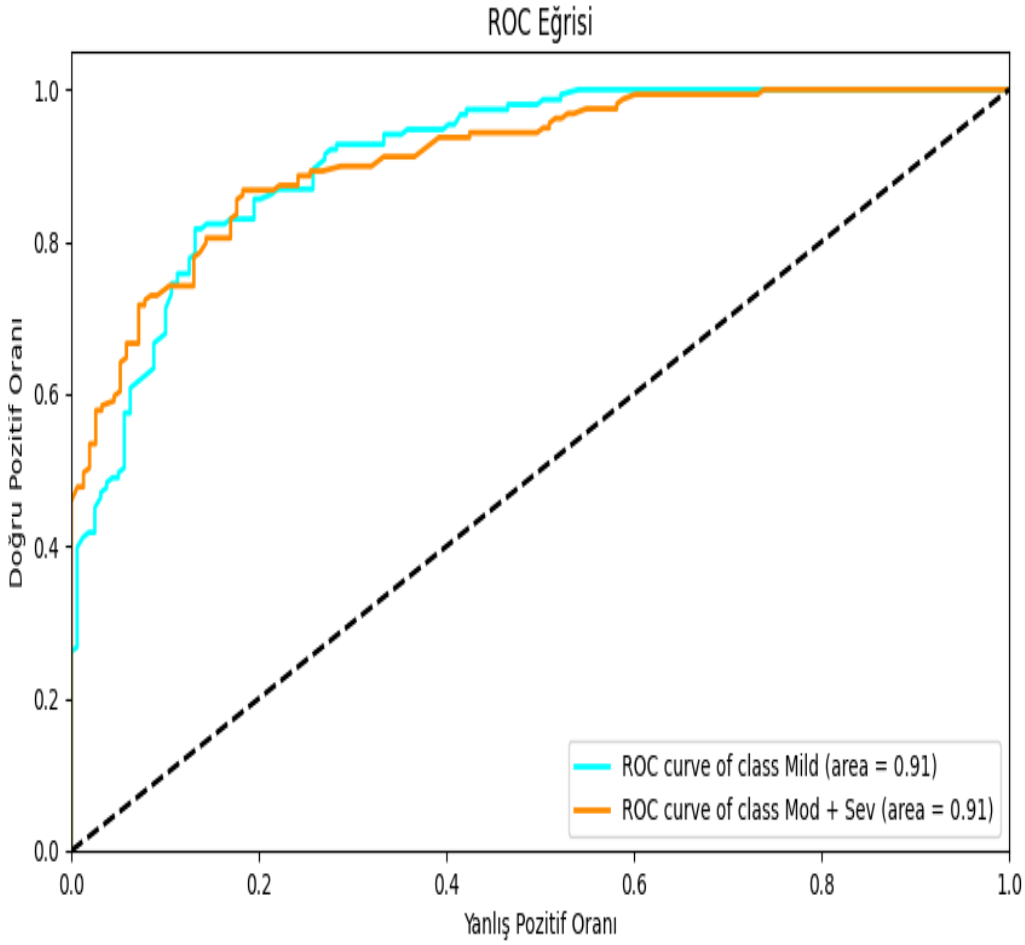
Bulunan parametrelerle kurulan modelin sınıflandırma metrikleri aşağıdaki gibidir.

	precision	recall	f1-score	support
Mild	0.78	0.87	0.82	153
Mod+Sev	0.86	0.77	0.81	159
accuracy			0.82	312
macro avg	0.82	0.82	0.82	312
weighted avg	0.82	0.82	0.82	312

```
Balanced Accuracy Score : 0.8182883216179554
```



Şekil 3.17: İki Sınıflı Rassal Ormanlar Modeli Karmaşıklık Matrisi



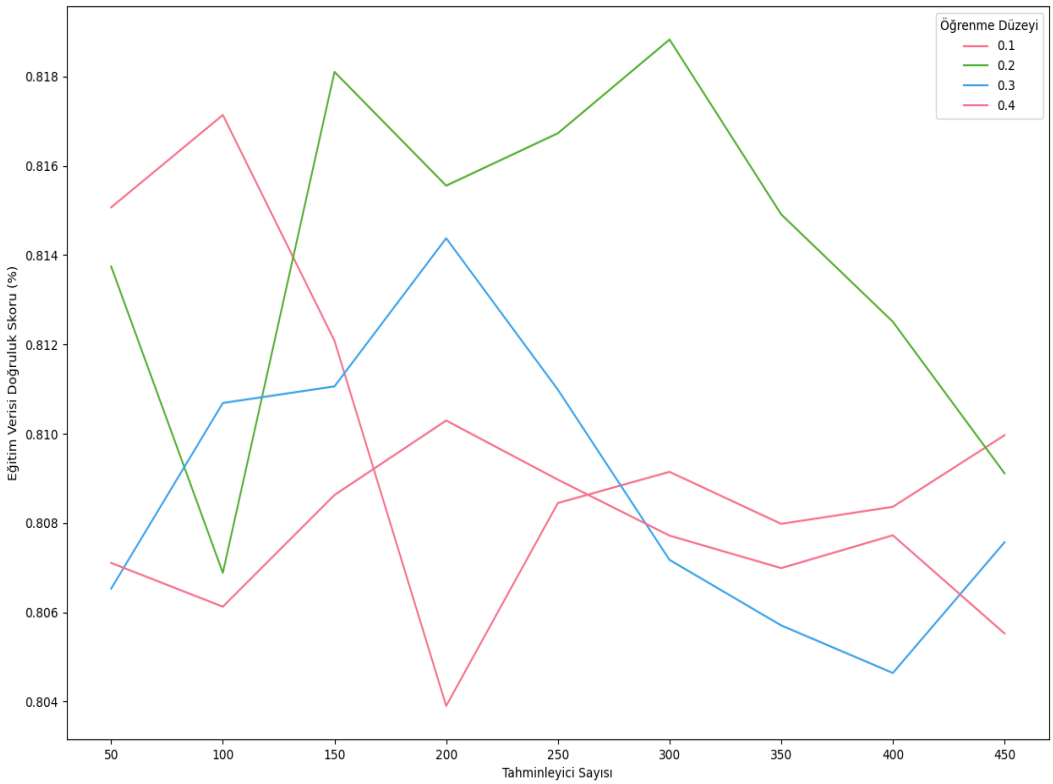
Şekil 3.18: İki Sınıflı Rassal Ormanlar Modeli ROC Eğrisi ve AUC Değerleri

3.2.3 XGBoost

Bu bölümde veri seti üzerinde xgboost modeli kullanılmış ve çıktıları değerlendirilmiştir.

Hiper Parametre Seçimi

Daha önce belirlenen parametre uzayını ve Scikit-Learn kütüphanesinde bulunan GridSearchCV algoritması ile en yüksek doğruluk oranı yakalanana kadar çalışması sağlanmıştır.



Şekil 3.19: İki Sınıflı XGBoost Modeli Eğitim Verisi Doğruluk Skorları

XGBoost modeli için en yüksek doğruluk oranı aşağıdaki parametreler ile bulunmuştur;

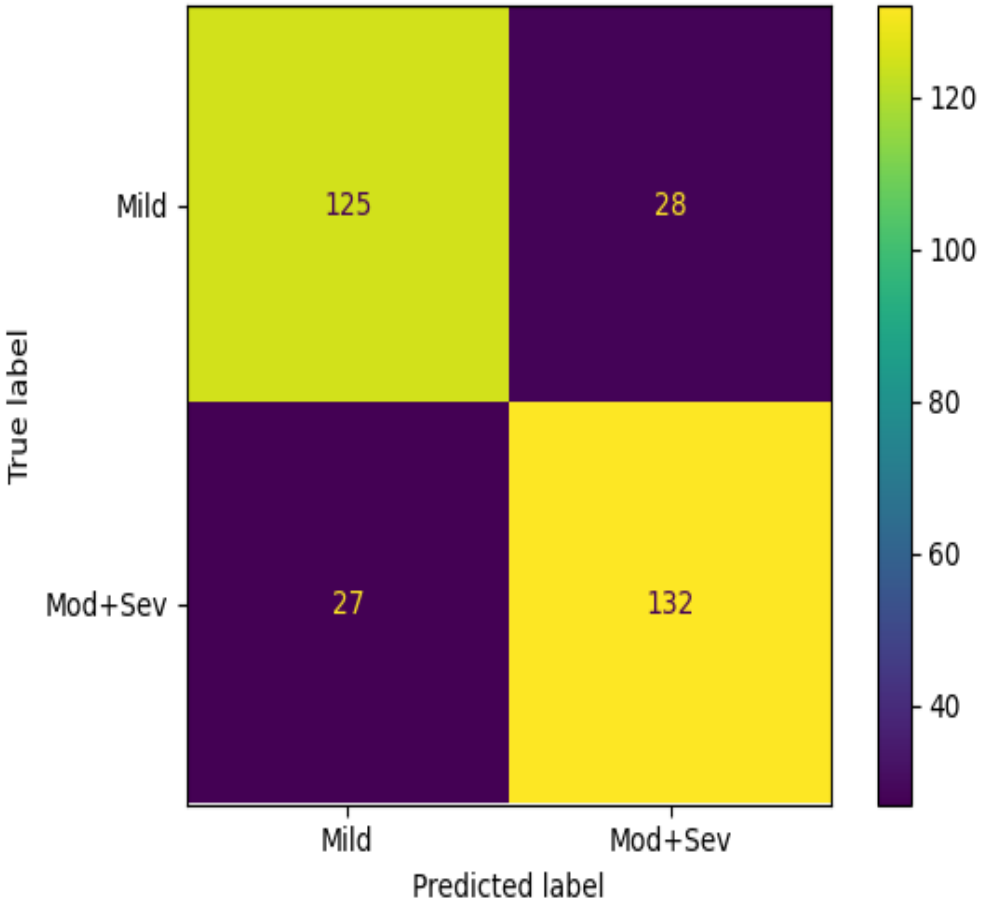
- 'eta': 0.2
- 'max_depth': 10
- 'min_child_weight': 10
- 'n_estimators': 400

En İyi Parametreli Model

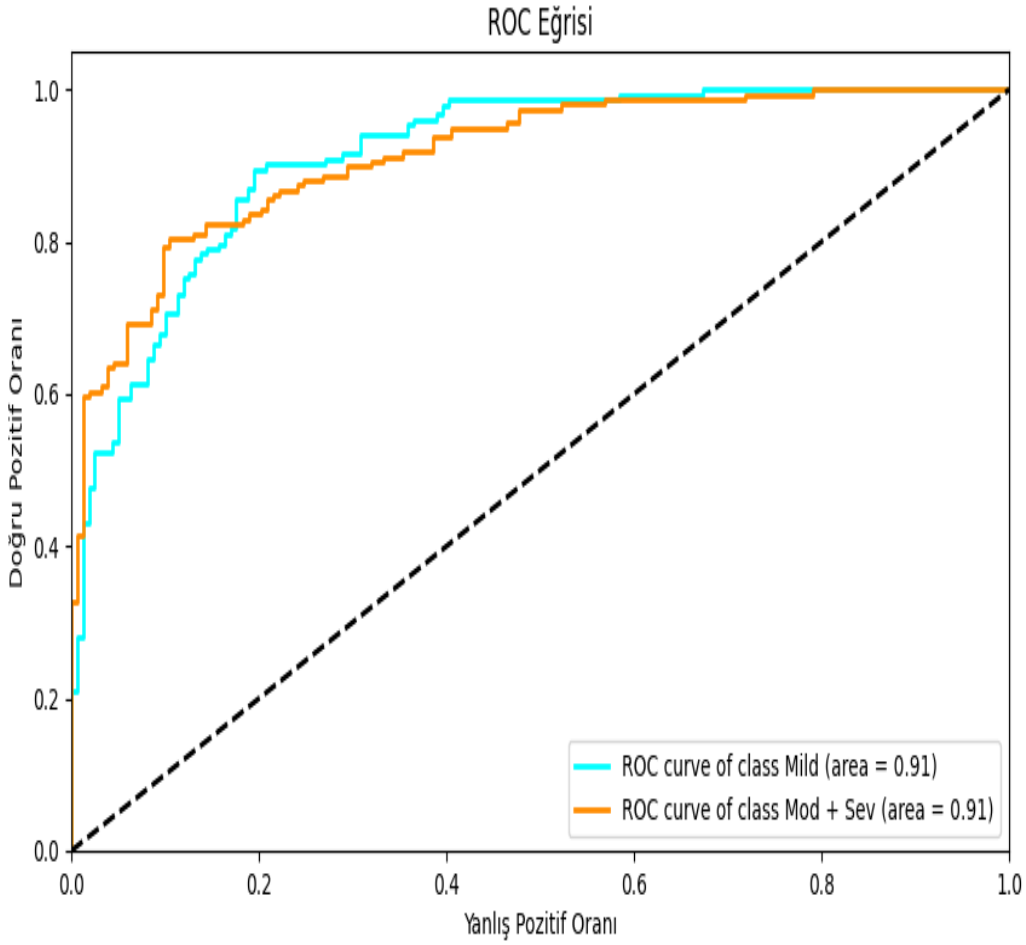
Bulunan parametrelerle kurulan modelin sınıflandırma metrikleri aşağıdaki gibidir.

	precision	recall	f1-score	support
Mild	0.82	0.82	0.82	153
Mod+Sev	0.82	0.83	0.83	159
accuracy			0.82	312
macro avg	0.82	0.82	0.82	312
weighted avg	0.82	0.82	0.82	312

```
Balanced Accuracy Score : 0.8235910716487853
```



Şekil 3.20: İki Sınıflı XGBoost Modeli Karmaşıklık Matrisi



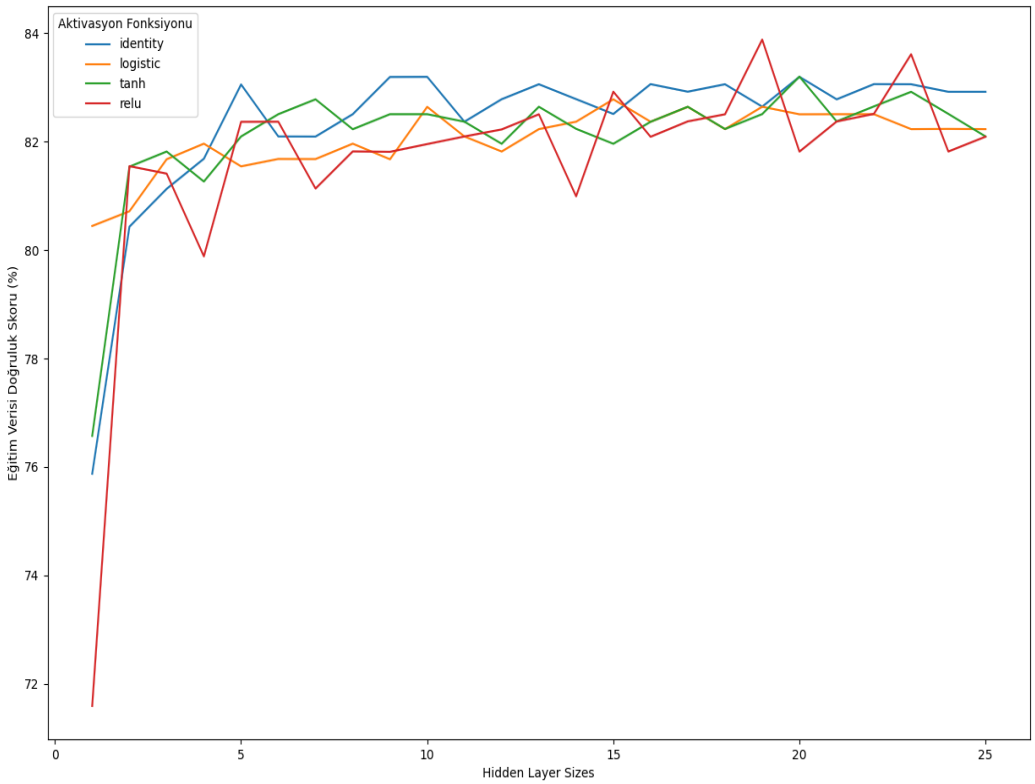
Şekil 3.21: İki Sınıflı XGBoost Modeli ROC Eğrisi ve AUC Değerleri

3.2.4 Neural Network (Yapay Sinir Ağları) Modeli

Bu bölümde veri seti üzerinde yapay sinir ağları modeli kullanılmış ve çıktıları değerlendirilmiştir.

Hiper Parametre Seçimi

Daha önce belirlenen parametre uzayını ve Scikit-Learn kütüphanesinde bulunan GridSearchCV algoritması ile en yüksek doğruluk oranı yakalanana kadar çalışması sağlanmıştır.



Şekil 3.22: İki Sınıflı Yapay Sinir Ağları Modeli Eğitim Verisi Doğruluk Skorları

Yapay sinir ağları modeli için en yüksek doğruluk oranı aşağıdaki parametreler ile bulunmuştur;

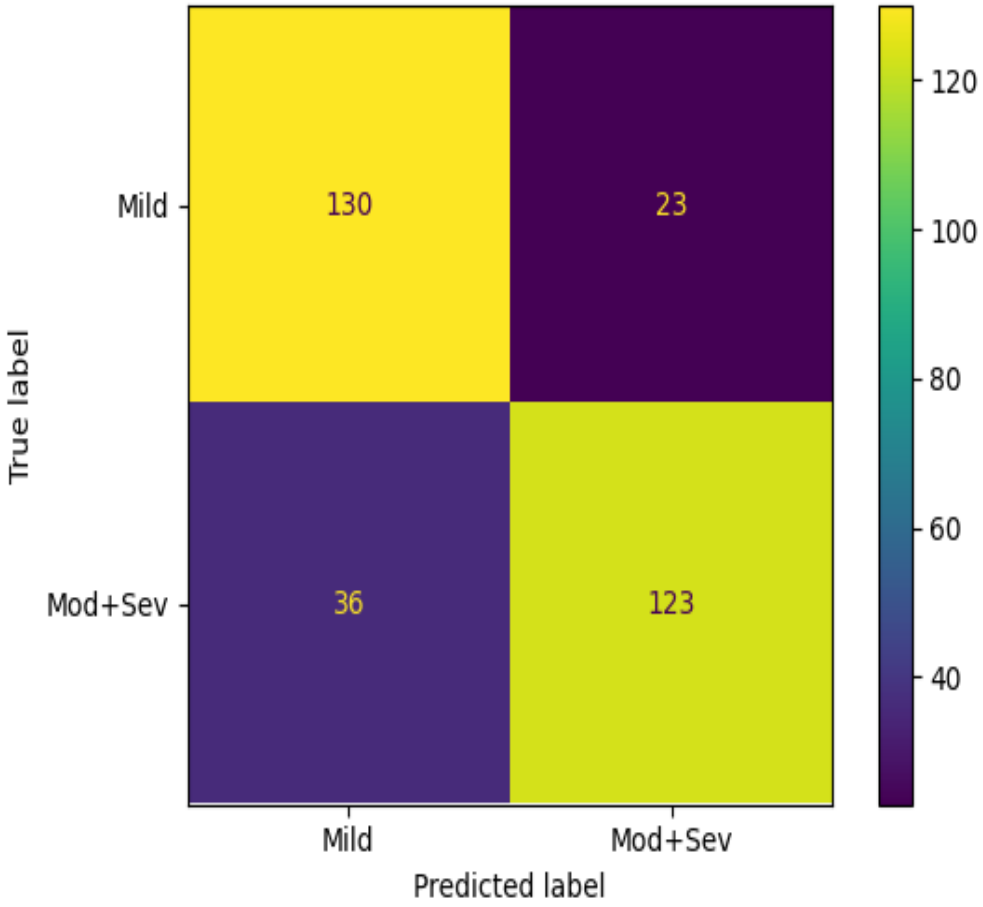
- 'activation': 'relu'
- 'hidden_layer_sizes': 19
- 'learning_rate': 'adaptive'

En İyi Parametreli Model

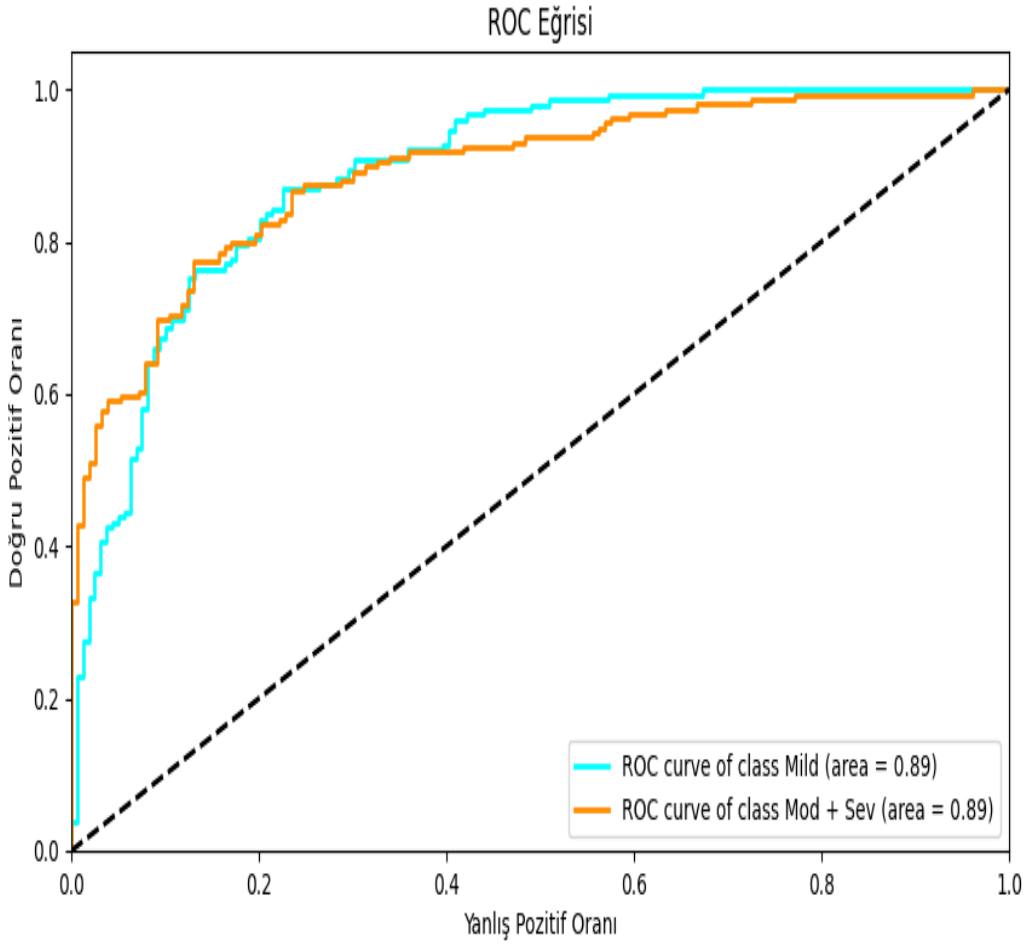
Bulunan parametrelerle kurulan modelin sınıflandırma metrikleri aşağıdaki gibidir.

	precision	recall	f1-score	support
Mild	0.78	0.85	0.82	153
Mod+Sev	0.84	0.77	0.81	159
accuracy			0.81	312
macro avg	0.81	0.81	0.81	312
weighted avg	0.81	0.81	0.81	312

```
Balanced Accuracy Score : 0.8116290541373783
```



Şekil 3.23: İki Sınıflı Yapay Sinir Ağları Modeli Karmaşıklık Matrisi



Şekil 3.24: İki Sınıflı Yapay Sinir Ağları Modeli ROC Eğrisi ve AUC Değerleri

3.2.5 İki Sınıflı Sınıflama Probleminin Modellerinin Değerlendirdirilmesi

Bölüm 3.2.1, 3.2.2, 3.2.3 ve 3.2.4'den elde edilen sonuçlar incelenmiş olup, iki sınıflı problem için %82 doğru sınıflama oranları ile XGBoost ve Rassal Ormanlar modelleri en iyi modeller olarak bulunmuştur.
Bölüm 3.2.2 ve 3.2.3'de bulunan performans metrikleri yakından incelendiğinde, XGBoost modeli kesinlik, duyarlılık ve f1-skor metrikleri bakımından Rassal Ormanlar modelinden daha iyi sonuç vermiştir.
Ciddiyet sınıflandırma probleminin 'Mild' ve 'Moderate + Severe' olacak şekilde 2 sınıfa indirgendiği durumda XGBoost modeli diğer modellerden daha iyi sonuç vermektedir.

Sonuç

If we don't want Conclusion to have a chapter number next to it, we can add the `{-}` attribute.

More info

And here's some other random info: the first paragraph after a chapter title or section head *shouldn't be* indented, because indents are to tell the reader that you're starting a new paragraph. Since that's obvious after a chapter or section title, proper typesetting doesn't add an indent there.

Kaynaklar

10 Ardakani, A. A., Afshar, A., Bhatt, S., Bureau, N. J., Tahmasebi, A., Acharya, U. R. ve Mohammadi, A. (2020). Diagnosis of carpal tunnel syndrome: A comparative study of shear wave elastography, morphometry and artificial intelligence techniques. *Pattern Recognition Letters*, *133*, 77-85.

Aroori, S. ve Spence, R. (2008). Carpal Tunnel Syndrome. *The Ulster Medical Society*, *77*, 1-17.

Arora, A. (2022, Mart). 8 unique machine learning interview questions on backpropagation. *Analytics Arora*. `https://analyticsarora.com/8-unique-machine-learning-interview-questions-on-backpropagation/` adresinden erişildi.

Bagatur, A. E. (2006). Karpal Tünel Sendromu. *Türkiye Klinikleri J Surg Med Sci.*, *2*(17), 52-63.

Breiman, Leo. (1997). *Arcing the edge*. Technical Report 486, Statistics Department, University of California.

Breiman, Leo. (1999). 1 RANDOM FORESTS–RANDOM FEATURES.

Breiman, L., Friedman, J., Olshen, R. ve Stone, C. (t.y.). *Classification and Regression Trees. 1984 Monterey, CA Wadsworth & Brooks. Cole Advanced Books & Software Google Scholar*.

Chen, T. ve Guestrin, C. (2016). Xgboost: A scalable tree boosting system. *Proceedings of the 22nd acm sigkdd international conference on knowledge discovery and data mining* içinde (ss. 785-794).

Cover, T. ve Hart, P. (1967). Nearest neighbor pattern classification. *IEEE Transactions on Information Theory*, *13*(1), 21-27. doi:10.1109/TIT.1967.1053964

Dikker, J. (2017). Master thesis Boosted tree learning for balanced item recommendation in online retail.

Doad, P. ve Bartere, M. (2013). A Review: Study of Various Clustering Techniques. *International Journal of Engineering Research & Technology*, *2*(11), 3141-3145.

Friedman, J. H. (2001). Greedy function approximation: A gradient boosting machine. *The Annals of Statistics*, *29*(5).

Ghasemi-Rad, M., Nosair, E., Vegh, A., Mohammadi, A., Akkad, A., Lesha, E., ... others. (2014). A handy review of carpal tunnel syndrome: From anatomy to diagnosis and treatment. *World journal of radiology*, *6*(6), 284.

Hastie, T., Tibshirani, R. ve Friedman, J. (2009). *The elements of Statistical Learning, second edition: Data Mining, Inference, and prediction*. Springer.

Koyama, T., Sato, S., Toriumi, M., Watanabe, T., Nimura, A., Okawa, A., ... Fujita, K. (2021). A Screening Method Using Anomaly Detection on a Smartphone

for Patients With Carpal Tunnel Syndrome: Diagnostic Case-Control Study. *JMIR mHealth and uHealth*, *9*(3), e26320.

Kumaş, F. F. (2005).Idiyopatik karpal t"unel sendromu tedavisinde terap"otik ultrason, steroid enjeksiyonu ve splint kullanımının etkinliğinin randimize kontroll"u araştırılması.

Kurt, A. (2020). *Karpal t"unel sendrom hastalarında bilateral ince motor beceri, skapular diskinezi, hareket korkusu ve fonksiyonun sağlıklılarla karşılaştırılması*. (Yayımlanmamış mathesis). Sağlık Bilimleri Enstit"us"u.

Levine, D. W., Simmons, B. P., Koris, M. J., Daltroy, L. H., Hohl, G. G., Fossel, A. H. ve Katz, J. N. (1993). A self-administered questionnaire for the assessment of severity of symptoms and functional status in carpal tunnel syndrome. *The Journal of bone and joint surgery. American volume*, *75*(11), 1585-1592.

Love, J. (1955). Median neuritis or carpal tunnel syndrome; diagnosis and treatment. *North Carolina medical journal*, *16*(10), 463-469.

Mason, L., Baxter, J., Bartlett, P. ve Frean, M. (1999). Boosting algorithms as gradient descent. *Advances in neural information processing systems*, *12*.

Mining, W. I. D. (2006). Data mining: Concepts and techniques. *Morgan Kaufinann*, *10*, 559-569.

Mitchell, T. M. ve Learning, M. (1997). McGraw-Hill. *New York*, 154-200.

Öztemel, E. (2003). Yapay sinir ağlari. *PapatyaYayincilik, Istanbul*.

Park, D., Kim, B. H., Lee, S.-E., Kim, D. Y., Kim, M., Kwon, H. D., ... Lee, J. W. (2021). Machine learning-based approach for disease severity classification of carpal tunnel syndrome. *Scientific Reports*, *11*(1), 1-10.

Pfeffer, G., Gelberman, R., Boyes, J. ve Rydevik, B. (1988). The history of carpal tunnnel syndrome. *The Journal of Hand Surgery: British & European Volume*, *13*(1), 28-34.

Quinlan, J. R. (2014). *C4. 5: programs for machine learning*. Elsevier.

Robbins, H. (1963). Anatomical study of the median nerve in the carpal tunnel and etiologies of the carpal-tunnel syndrome. *JBJS*, *45*(5), 953-966.

Sahu, V. (2021, Haziran). Power of a single neuron. *Medium*. Towards Data Science. `https://towardsdatascience.com/power-of-a-single-neuron-perceptron-c418ba445095` adresinden erişildi.

Salam Patrous, Z. (2018). Evaluating XGBoost for user classification by using behavioral features extracted from smartphone sensors.

Sezgin, M., İncel, N. A., Sevim, S., Çamdeviren, H., As, İ. ve Erdoğan, C. (2006). Assessment of symptom severity and functional status in patients with carpal tunnel syndrome: reliability and validity of the Turkish version of the Boston Questionnaire. *Disability and rehabilitation*, *28*(20), 1281-1286.

TIBCO. (t.y.). What is a random forest? *TIBCO Software*. `https://www.tibco.com/reference-center/what-is-a-random-forest` adresinden erişildi.

Werner, R. A. ve Andary, M. (2002). Carpal tunnel syndrome: pathophysiology and clinical neurophysiology. *Clinical Neurophysiology*, *113*(9), 1373-1381.

Yangın, G. (2019). *Xgboost ve karar ağaçları tabanlı algoritmaların diyabet veri setleri üzerine uygulaması*. (Yayımlanmamış doktora tezi). Yüksek Lisans Tezi, Mimar Sinan Güzel Sanatlar Üniversitesi Fen Bilimleri.

Ek A

Gerekli Paketlerin Yüklenmesi ve Verilerin Okunması ve Veri Önişleme

```r
# R için gerekli paketlerin kurulması
if(!require(reticulate)) install.packages("reticulate", repos = "http://cran.rstudio.com")
if(!require(tidyverse)) install.packages("tidyverse", repos = "http://cran.rstudio.com")
if(!require(caret)) install.packages("caret",repos = "http://cran.rstudio.com")
if(!require(caretEnsemble))  install.packages("caretEnsemble",repos = "http://cran.rstudio.com")
if(!require(doParallel))  install.packages("doParallel",repos = "http://cran.rstudio.com")
if(!require(data.table))  install.packages("data.table",repos = "http://cran.rstudio.com")
if(!require(dplyr))  install.packages("dplyr",repos = "http://cran.rstudio.com")
if(!require(e1071))  install.packages("e1071",repos = "http://cran.rstudio.com")
#if(!require(gbm))  install.packages("gbm",repos = "http://cran.rstudio.com")
if(!require(kernlab))  install.packages("kernlab",repos = "http://cran.rstudio.com")
#if(!require(randomForest))  install.packages("randomForest",repos = "http://cran.rstudio.com")
if(!require(tidyverse))  install.packages("tidyverse",repos = "http://cran.rstudio.com")
#if(!require(xgboost))  install.packages("xgboost",repos = "http://cran.rstudio.com")
if(!require(smotefamily))  install.packages("smotefamily",repos = "http://cran.rstudio.com")
```

```python
# Python için gerekli paketlerin ve veri setinin yüklenmesi
import warnings
warnings.filterwarnings("ignore", category=FutureWarning)
from warnings import simplefilter
simplefilter(action='ignore', category=FutureWarning)
import numpy as np, pandas as pd, matplotlib.pyplot as plt
import seaborn as sns
from sklearn.feature_selection import VarianceThreshold
from scipy.stats import f_oneway, chi2_contingency
CTS = pd.read_csv("data/CTS.csv",sep=",")
dataGroup = CTS[["Severity","Age","BMI","CSA","PB","Duration","NRS"]]
dataOverall = CTS[["Age","BMI","CSA","PB","Duration","NRS"]]
meanoval, stdoval = round(dataOverall.mean(),1), round(dataOverall.std(ddof=1),1)
means = round(dataGroup.groupby("Severity").mean(),1)
stds = round(dataGroup.groupby("Severity").std(ddof=1),1)
##
mild = CTS[CTS.Severity == "mild"]
moderate = CTS[CTS.Severity == "moderate"]
severe = CTS[CTS.Severity == "severe"]
numVar = ["Age","BMI","CSA","PB","Duration","NRS"]
catVar = ["Sex","Side","Diabetes","NP","Weakness"]
p_values = []
```

```
p_vals2 = []
for i in numVar:
    _,p_val = f_oneway(mild[i],moderate[i],severe[i])
    p_values.append(round(p_val,3))
for i in catVar:
    var_0 = np.array([sum(mild[i] == 0),sum(moderate[i] == 0),sum(severe[i] == 0)])
    var_1 = np.array([sum(mild[i] == 1),sum(moderate[i] == 1),sum(severe[i] == 1)])
    p_vals2.append(round(chi2_contingency(np.array([var_1,var_0]),correction=False)[1],3))
CTS_kor = CTS.drop(["Severity","Mild","Mod","Sev"],axis=1)
zeroVar = CTS_kor.shape[1]-((VarianceThreshold(threshold=0).fit(CTS_kor)).get_support()).sum()
######
catDF = CTS.groupby("Severity").sum()[["Sex","Side","Diabetes","NP","Weakness"]]
sex, rside, diab, np, weak = catDF["Sex"],catDF["Side"],catDF["Diabetes"],catDF["NP"],catDF["Weakness"]
hands = CTS.groupby("Severity").count()["NP"]
handsx = hands*100
```

```
# R için veri setinin tanıtılması ve Rastgele olarak ayrılması
  means <- as.data.frame(py$means)
  stds <- as.data.frame(py$stds)
  meanOval <- as.data.frame(t(py$meanoval))
  stdOval <- as.data.frame(t(py$stdoval))
  CTS <- as.data.frame(read_csv("data/CTS.csv"))
  seed<-0923
  set.seed(seed)
  ind<-sample(2,nrow(CTS),replace = T,prob = c(0.7,0.3))
  traindata_top <- CTS[ind==1,]
  testdata_top <- CTS[ind==2,]
# BURADAN SORNASI R VERİ ÖNİŞLEME
  CTS$Severity<-as.factor(CTS$Severity)
  CTS$Mild<-as.factor(CTS$Mild)
  CTS$Mod<-as.factor(CTS$Mod)
  CTS$Sev<-as.factor(CTS$Sev)
  CTS$Sex <-as.factor(CTS$Sex)
  CTS$Side <-as.factor(CTS$Side)
  CTS$Diabetes <-as.factor(CTS$Diabetes)
  CTS$NP <- as.factor(CTS$NP)
  CTS$Weakness <- as.factor(CTS$Weakness)
  predata<-CTS
  st_model<-preProcess(predata[,5:10], method=c("center","scale"))
  data<-predict(st_model, predata)
  data=as.data.frame(data)
  ohe_feats = c('Sex','Side','Diabetes','NP','Weakness')
  dummies = dummyVars(~ Sex+Side+Diabetes+NP+Weakness, data = data)
  df_ohe <- as.data.frame(predict(dummies, newdata = data))
  df_combined <- cbind(data[,-c(which(colnames(data) %in% ohe_feats))],df_ohe)
  dat = as.data.table(df_combined)
  traindata<-dat[ind==1,]
  testdata<-dat[ind==2,]
  trainmc<-traindata
  testmc<-testdata
  trainmc$Mild=NULL
  trainmc$Mod=NULL
  trainmc$Sev=NULL
  testmc$Mild=NULL
  testmc$Mod=NULL
  testmc$Sev=NULL
  hco <- nrow(CTS)
  hco <- hco * 100
```

```
# Python için Veri Önişleme
from sklearn.preprocessing import LabelEncoder
from sklearn.preprocessing import StandardScaler
LE = LabelEncoder().fit(["mild","moderate","severe"])
traindata_P = pd.DataFrame(r.traindata_top)
traindata_P.drop(["Mild","Mod","Sev"],axis=1,inplace=True)
testdata_P = pd.DataFrame(r.testdata_top)
```

```
testdata_P.drop(["Mild","Mod","Sev"],axis=1,inplace=True)
X_train, X_test, y_train, y_test = traindata_P.drop(["Severity"],axis=1),testdata_P.drop(["Severity"],
axis=1),pd.DataFrame(LE.transform(traindata_P.Severity)),
pd.DataFrame(LE.transform(testdata_P.Severity))
Stand = StandardScaler().fit(r.CTS[["Age","BMI","CSA","PB","Duration","NRS"]])
X_train[["Age","BMI","CSA","PB","Duration","NRS"]]=pd.DataFrame(Stand.transform(X_train[["Age","BMI",
"CSA","PB","Duration","NRS"]]),columns=["Age","BMI","CSA","PB","Duration","NRS"])
y_train = y_train.to_numpy().ravel()
X_test[["Age","BMI","CSA","PB","Duration","NRS"]] = pd.DataFrame(Stand.transform(X_test[["Age","BMI",
"CSA","PB","Duration","NRS"]]),columns=["Age","BMI","CSA","PB","Duration","NRS"])
y_test = y_test.to_numpy().ravel()
```

Ek B

Sınıflandırma (Üç Sınıf)

B.1 Roc Curve ve OneVsRest

```
# OneVsRest Modellerin ve ROC curvelerin oluşturulması
from sklearn.metrics import roc_auc_score, roc_curve, classification_report, confusion_matrix
from sklearn.metrics import ConfusionMatrixDisplay, auc
import pandas as pd
from sklearn.multiclass import OneVsRestClassifier
import matplotlib.pyplot as plt
from itertools import cycle
from sklearn.preprocessing import label_binarize
def roc(model):
    """ Unfitted model"""
    model_name = str(model.__class__).split(".")[-1][:-2]
    model = OneVsRestClassifier(model).fit(X_train,y_train)
    plt.figure()
    y_pred = model.predict_proba(X_test)
    fpr = dict()
    tpr = dict()
    thresh = dict()
    thresh_df = pd.DataFrame(columns=["Class","Threshold"])
    roc_auc = dict()
    for i in range(3):
        fpr[i], tpr[i], thresh[i] = roc_curve(y_test, y_pred[:, i],pos_label=i)
        roc_auc[i] = auc(fpr[i], tpr[i])
    for i,j in zip(thresh.keys(),["mildVsAll","modVsAll","sevVsAll"]):
        thresh[j] = thresh.pop(i)
        if j == "sevVsAll" : break
    for i,j in zip(thresh.keys(),thresh.values()):
        for x in j:
            thresh_df = thresh_df.append({"Class":i,"Threshold":x},ignore_index=True)
    thresh_df.to_csv(f'data/thresholds_{model_name}.csv',index=False)
    colors = cycle(["aqua", "darkorange", "cornflowerblue"])
    for i, color, j in zip(range(3), colors,["Mild","Moderate","Severe"]):
        plt.plot(
            fpr[i],
            tpr[i],
            color=color,
            lw=2,
            label="ROC curve of class {0} (area = {1:0.2f})".format(j, roc_auc[i])
        )
    plt.plot([0, 1], [0, 1], "k--", lw=2)
    plt.xlim([0.0, 1.0])
    plt.ylim([0.0, 1.05])
```

```
    plt.xlabel("False Positive Rate")
    plt.ylabel("True Positive Rate")
    plt.title("Some extension of Receiver operating characteristic to multiclass")
    plt.legend(loc="lower right")
    plt.savefig(f'figure/roc_curve_{model_name}.png')
```

B.2 KNN

```
# 3 sınıf için KNN Modeli kurma ve GridSearchCV algoritmasını hazırlama
import numpy as np
from sklearn.neighbors import KNeighborsClassifier
from sklearn.model_selection import GridSearchCV
from sklearn.metrics import ConfusionMatrixDisplay,confusion_matrix,classification_report
from sklearn.metrics import balanced_accuracy_score
KNN_model = KNeighborsClassifier()
params = {"n_neighbors":np.arange(5,200),
          "weights":["uniform", "distance"],
          "algorithm":["auto","ball_tree","kd_tree","brute"]}
GSC = GridSearchCV(KNN_model,param_grid=params,
                   cv=10,verbose=1,scoring="accuracy").fit(X_train,y_train)
pd.DataFrame(GSC.cv_results_).to_csv("data/KNN_GridSearch_Results.csv",index=False)
knn = pd.read_csv("data/KNN_GridSearch_Results.csv")
plt.figure(figsize=(10,5),dpi=60);
sns.lineplot(x=knn.param_n_neighbors,y=knn.mean_test_score*100,hue=knn.param_weights);
plt.ylabel("Eğitim Verisi Doğruluk Skoru (%)");
plt.xlabel("Number of Neighbors");
plt.legend(title="Ağırlıklandırma",loc="upper right",labels=["Eşit Ağırlıklandırma",
"Uzaklığa Göre Ağırlıklandırma"]);
plt.savefig("figure/KNN_Grid_Graph.png");
```

```
print(f'En İyi Parametreler : {GSC.best_params_}')
```

```
# En iyi parametreler ile modelin tekrar kurulması
KNN_model = KNeighborsClassifier(n_neighbors = 33,
                                 weights ='distance').fit(X_train,y_train)
y_pred = KNN_model.predict(X_test)
print(classification_report(y_test,y_pred,target_names=["Mild","Moderate","Severe"]))
print(f'Balanced Accuracy Score : {balanced_accuracy_score(y_test,y_pred)}')
```

```
# 3 sınıflı KNN için ConfusionMatrix
ConfusionMatrixDisplay(confusion_matrix(y_test,y_pred),
display_labels=["Mild","Moderate","Severe"]).plot();
plt.savefig("figure/knn_conf.png");
```

```
# 3 sınıflı KNN için ROC curve
roc(KNeighborsClassifier(n_neighbors = 33,
                         weights ='distance'))
```

B.3 Rassal Ormanlar

```
# 3 sınıf için rassal ormanlar Modeli kurma ve GridSearchCV algoritmasını hazırlama
from sklearn.ensemble import RandomForestClassifier
from sklearn.model_selection import GridSearchCV
```

```
from sklearn.metrics import ConfusionMatrixDisplay,confusion_matrix,classification_report
from sklearn.metrics import balanced_accuracy_score
RFC_model = RandomForestClassifier()
param_grid = {"n_estimators":np.arange(350,1000,50),
              "criterion":["gini","index"],
              "max_features":["auto","sqrt","log2"],
              "ccp_alpha":[0.01,0.05,.1,0.3,.5,.7,.9,1],
              "max_samples":np.arange(1,X_train.shape[1],1)}
GSC = GridSearchCV(RFC_model,param_grid=params,
                   cv=10,verbose=1,scoring="accuracy",random_state=13).fit(X_train,y_train)
results = pd.read_csv("data/RF_GridSearch_Results.csv")
ginis = results[results["param_criterion"] == "gini"]
ginis = ginis[ginis["param_ccp_alpha"] <= 0.1]
ginis = ginis[ginis["mean_test_score"]>=0.704]
#Grafik Çizimi
plt.figure(figsize=(10,5),dpi=60);
plt.ylim(0.703*100,round(ginis.mean_test_score.unique().max()*100,2)+0.1);
sns.lineplot(y=ginis.mean_test_score*100,x=ginis.param_n_estimators,hue=ginis.param_ccp_alpha,
palette=sns.color_palette(n_colors=3),err_style=None);
plt.ylabel("Eğitim Verisi Doğruluk Skoru (%)");
plt.xlabel("Number of Trees in Forest");
plt.legend(title="Learning Rate",loc="upper right",labels=[0.01,0.05,0.1]);
plt.savefig("figure/RF_Grid_Graph.png");
```

```
print(f'En İyi Parametreler : {GSC.best_params_}')
```

```
RFC_model = RandomForestClassifier(ccp_alpha=0.05,criterion="gini",max_features="auto",
                                   max_samples=10,n_estimators=350,random_state=13).fit(X_train,y_train)
y_pred = RFC_model.predict(X_test)
print(classification_report(y_test,y_pred,target_names=["Mild","Moderate","Severe"]))
print(f'Balanced Accuracy Score : {balanced_accuracy_score(y_test,y_pred)}')
```

```
ConfusionMatrixDisplay(confusion_matrix(y_test,y_pred),
display_labels=["Mild","Moderate","Severe"]).plot();
plt.savefig("figure/rfc_conf.png")
```

```
roc(RandomForestClassifier(ccp_alpha=0.01,criterion="gini",max_features="sqrt",
                                     max_samples=10,n_estimators=900,random_state=13))
```

B.4 XGBoost

```
# 3 sınıf için XGB Modeli
from xgboost.sklearn import XGBClassifier
from sklearn.model_selection import GridSearchCV
from sklearn.metrics import ConfusionMatrixDisplay,confusion_matrix,classification_report
from sklearn.metrics import balanced_accuracy_score
XGB_model = XGBClassifier()
param_grid = {"booster":["gbtree","gblinear"],
              "eta":np.arange(0,1,0.1),
              "min_child_weight":np.arange(0,5,1),
              "max_depth":np.arange(3,10,1),
              "gamma":np.arange(0,1,0.1),
              "sumsample":np.arange(0,1,0.1),
              "colsample_bytree":np.arange(0,1,.1),
              "n_estimators":np.arange(0,1000,50),
              "objective":["multi:softmax","multi:softprob"],
              "eval_metric":["auc"],
              "use_label_encoder":[False]}
```

```
GSC = GridSearchCV(XGB_model,param_grid=param_grid,cv=10,verbose=1,n_jobs=-1,scoring="accuracy")
GSC.fit(X_train,y_train)
resultsxgb = pd.read_csv("data/XGBoost_GridSearch_Results.csv",sep=";")
resultsxgb = resultsxgb.sort_values("rank_test_score")
sorted_xgb = resultsxgb[["param_booster","param_eta","param_max_depth","param_min_child_weight",
"param_n_estimators","param_objective","mean_test_score","rank_test_score"]]
#Grafik Çizimi
plt.figure(figsize=(15,10),dpi=100);
plt.ylim(0.72*100,round(sorted_xgb.mean_test_score.unique().max()*100,2)+0.05);
for i in sorted_xgb.param_eta.unique():
    if i <=0.4:
        x = sorted_xgb[sorted_xgb.param_eta == i].groupby("param_n_estimators").max()
        sns.lineplot(x=x.index,y=x.mean_test_score*100);
plt.ylabel("Eğitim Verisi Doğruluk Skoru (%)");
plt.legend([.1,.4,.2,.3],loc=1,title="Learning Rate");
plt.savefig("figure/XGB_Grid_Graph.png");
```

```
print(f'En İyi Parametreler : {GSC.best_params_}')
```

```
XGB_model = XGBClassifier(booster="gbtree",eta="0.1",max_depth=3,min_child_weight=10,n_estimators=100,
objective="multi:softprob",eval_metric="auc",use_label_encoder=False,num_class=2).fit(X_train,y_train)
y_pred = XGB_model.predict(X_test)
print(classification_report(y_test,y_pred,target_names=["Mild","Moderate","Severe"]))
print(f'Balanced Accuracy Score : {balanced_accuracy_score(y_test,y_pred)}')
```

```
ConfusionMatrixDisplay(confusion_matrix(y_test,y_pred),
                       display_labels=["Mild","Moderate","Severe"]).plot();
plt.savefig("figure/xgb_conf.png");
```

```
roc(XGBClassifier(booster="gbtree",eta="0.1",max_depth=3,min_child_weight=10,n_estimators=100,
objective="multi:softprob",eval_metric="auc",use_label_encoder=False,num_class=2))
```

B.5 Neural Network

```
# 3 sınıflı sinir ağları modeli
from sklearn.neural_network import MLPClassifier
from sklearn.model_selection import GridSearchCV
from sklearn.metrics import ConfusionMatrixDisplay,confusion_matrix,classification_report
from sklearn.metrics import balanced_accuracy_score
params = {"hidden_layer_sizes":np.arange(1,26,1),
          "learning_rate":["adaptive","constant","invscaling"],
          "activation":["identity","logistic","tanh","relu"]}
gridd = GridSearchCV(MLPClassifier(random_state=13),param_grid=params,cv=10,
                                   verbose=1,n_jobs=-1,scoring="accuracy")
gridd.fit(X_train,y_train)
pd.DataFrame(gridd.cv_results_).to_csv("NN_GridSearch_Results.csv",sep=";",index=False)
result_nn = pd.read_csv("data/NN_GridSearch_Results.csv",sep=";")
#Grafik Çizimi
plt.figure(figsize=(15,10),dpi=100);
sns.lineplot(x=result_nn["param_hidden_layer_sizes"],y=result_nn["mean_test_score"]*100,
             hue=result_nn["param_activation"]);
plt.xlabel("Hidden Layer Sizes");
plt.ylabel("Eğitim Verisi Doğruluk Skoru (%)");
plt.legend(title="Aktivasyon Fonksiyonu");
plt.savefig("figure/NN_Grid_Graph.png");
```

```
print(f'En İyi Parametreler : {gridd.best_params_}')
```

```
NN_model = MLPClassifier(activation="relu",hidden_layer_sizes=19,learning_rate="adaptive",
                         random_state=13,max_iter=3000).fit(X_train,y_train)
y_pred = NN_model.predict(X_test)
print(classification_report(y_test,y_pred,target_names=["Mild","Moderate","Severe"]))
print(f'Balanced Accuracy Score : {balanced_accuracy_score(y_test,y_pred)}')
```

```
ConfusionMatrixDisplay(confusion_matrix(y_test,y_pred),
                       display_labels=["Mild","Moderate","Severe"]).plot();
plt.savefig("figure/nn_conf.png")
```

```
roc(MLPClassifier(activation="relu",hidden_layer_sizes=19,learning_rate="adaptive",
                  random_state=13,max_iter=3000))
```

Ek C

Sınıflandırma (İki Sınıf)

C.1 Preprocess

```
# 3 sınıfı 2 sınıfa indirgeme
from sklearn.model_selection import train_test_split
bin_data = pd.read_csv("data/binary_data.csv",sep=";")
X_bin = bin_data.drop(["Severity","New_Sev"],axis=1)
y_bin = bin_data["New_Sev"]
Xbin_train, Xbin_test, ybin_train, ybin_test = train_test_split(X_bin,y_bin,test_size=0.3,
                                                                stratify=y_bin,random_state=13)
def roc_bin(model):
    model_name = str(model.__class__).split(".")[-1][:-2]
    ybin_pred2 = model.predict_proba(Xbin_test)
    plt.figure(figsize=(10,5),dpi=100)
    fpr = dict()
    tpr = dict()
    thresh = dict()
    roc_auc = dict()
    for i in range(2):
        fpr[i], tpr[i], thresh[i] = roc_curve(ybin_test, ybin_pred2[:, i],pos_label=i)
        roc_auc[i] = auc(fpr[i], tpr[i])
    colors = cycle(["aqua", "darkorange", "cornflowerblue"])
    for i, color, j in zip(range(2), colors,["Mild","Mod + Sev"]):
        plt.plot(
        fpr[i],
        tpr[i],
        color=color,
        lw=2,
        label="ROC curve of class {0} (area = {1:0.2f})".format(j, roc_auc[i])
        )
    plt.plot([0, 1], [0, 1], "k--", lw=2)
    plt.xlim([0.0, 1.0])
    plt.ylim([0.0, 1.05])
    plt.xlabel("False Positive Rate")
    plt.ylabel("True Positive Rate")
    plt.title("Some extension of Receiver operating characteristic to multiclass")
    plt.legend(loc="lower right")
    plt.savefig(f'figure/{model_name}_binary_roc.png')
```

C.2 KNN

```
KNN_model_bin = KNeighborsClassifier()
params = {"n_neighbors":np.arange(5,200),
          "weights":["uniform", "distance"],
          "algorithm":["auto","ball_tree","kd_tree","brute"]}
GSC = GridSearchCV(KNN_model_bin,param_grid=params,cv=10,verbose=1,scoring="accuracy",n_jobs=-1)
GSC.fit(Xbin_train,ybin_train)
pd.DataFrame(GSC.cv_results_).to_csv("data/KNN_bin_GridSearch_Results.csv",index=False)
knn_bin = pd.read_csv("data/KNN_bin_GridSearch_Results.csv",sep=";")
plt.figure(figsize=(10,5),dpi=60);
sns.lineplot(x=knn_bin["param_n_neighbors"],y=knn_bin["mean_test_score"]*100,
             hue=knn_bin["param_weights"]);
plt.ylabel("Eğitim Verisi Doğruluk Skoru (%)");
plt.xlabel("Number of Neighbors");
plt.legend(title="Ağırlıklandırma",loc="upper right",
           labels=["Eşit Ağırlıklandırma","Uzaklığa Göre Ağırlıklandırma"]);
plt.savefig("figure/KNN_bin_Grid_Graph.png");
```

```
print(f'En İyi Parametreler : {GSC.best_params_}')
```

```
KNN_modelbin = KNeighborsClassifier(n_neighbors = 23,
                                    weights ='uniform').fit(Xbin_train,ybin_train)
ybin_pred = KNN_modelbin.predict(Xbin_test)
print(classification_report(ybin_test,ybin_pred,target_names=["Mild","Mod+Sev"]))
print(f'Balanced Accuracy Score : {balanced_accuracy_score(ybin_test,ybin_pred)}')
```

```
ConfusionMatrixDisplay(confusion_matrix(ybin_test,ybin_pred),
                       display_labels=["Mild","Mod+Sev"]).plot();
plt.savefig("figure/knn_bin_conf.png")
```

```
roc_bin(KNN_modelbin)
```

C.3 Rassal Ormanlar

```
param_grid = {"n_estimators":np.arange(350,700,50),
              "max_features":["auto","sqrt","log2"],
              "ccp_alpha":[0.01,0.05,.1,0.3,.5],
              "max_samples":np.arange(1,X_train.shape[1],1)}
GSC = GridSearchCV(RandomForestClassifier(random_state=13,criterion="gini"),
                   param_grid=param_grid,cv=10,verbose=1,n_jobs=-1,scoring="accuracy")
GSC.fit(Xbin_train,ybin_train)
pd.DataFrame(GSC.cv_results_).to_csv("data/RF_bin_Grid_Res.csv",index=False)
rf_bin = pd.read_csv("data/RF_bin_Grid_Res.csv",sep=";")
maxed = rf_bin.groupby("rank_test_score").max()
maxed = maxed[maxed["mean_test_score"] > 0.75]
plt.figure(figsize=(15,10),dpi=100);
sns.lineplot(x="param_n_estimators",y="mean_test_score",hue="param_ccp_alpha",data=maxed,
             err_style=None,palette="husl");
plt.xlabel("Ormandaki Ağaç Sayısı");
plt.ylabel("Eğitim Verisi Doğruluk Skoru (%)");
plt.legend(title="Öğrenme Düzeyi");
plt.savefig("figure/RF_bin_Grid_Graph.png");
```

```
print(f'En İyi Parametreler : {GSC.best_params_}')
```

```
RF_modelbin = RandomForestClassifier(ccp_alpha = 0.01,
                                     max_features = "auto",
                                     max_samples = 10,
                                     n_estimators = 400).fit(Xbin_train,ybin_train)
ybin_pred = RF_modelbin.predict(Xbin_test)
print(classification_report(ybin_test,ybin_pred,target_names=["Mild","Mod+Sev"]))
print(f'Balanced Accuracy Score : {balanced_accuracy_score(ybin_test,ybin_pred)}')
```

```
ConfusionMatrixDisplay(confusion_matrix(ybin_test,ybin_pred),
                       display_labels=["Mild","Mod+Sev"]).plot();
plt.savefig("figure/rf_bin_conf.png");
```

```
roc_bin(RF_modelbin)
```

C.4 XGBoost

```
param_grid = {"eta":np.arange(0,.5,0.1),
              "min_child_weight":np.arange(1,11,1),
              "max_depth":np.arange(3,11,1),
              "n_estimators":np.arange(0,500,50)}
GSC = GridSearchCV(XGBClassifier(eval_metric="auc",use_label_encoder=False,booster="gbtree"),
                                 param_grid=param_grid,cv=10,verbose=1,n_jobs=-1,scoring="accuracy")
GSC.fit(Xbin_train,ybin_train)
pd.DataFrame(GSC.cv_results_).to_csv("data/XGB_bin_Grid_Res.csv",index=False)
XGB_bin = pd.read_csv("data/XGB_bin_Grid_Res.csv",sep=";")
maxed = XGB_bin.groupby("rank_test_score").max()
maxed = maxed[maxed["mean_test_score"] > 0.75]
plt.figure(figsize=(15,10),dpi=100);
sns.lineplot(x="param_n_estimators",y="mean_test_score",hue="param_eta",data=maxed,
             err_style=None,palette="husl");
plt.xlabel("Tahminleyici Sayısı");
plt.ylabel("Eğitim Verisi Doğruluk Skoru (%)");
plt.legend(title="Öğrenme Düzeyi");
plt.savefig("figure/XGB_bin_Grid_Graph.png");
```

```
print(f'En İyi Parametreler : {GSC.best_params_}')
```

```
XGB_modelbin = XGBClassifier(eval_metric="auc",
                             use_label_encoder=False,
                             booster="gbtree",
                             eta=0.2,
                             max_depth=10,
                             min_child_weight=10,
                             n_estimators=400).fit(Xbin_train,ybin_train)
ybin_pred = XGB_modelbin.predict(Xbin_test)
print(classification_report(ybin_test,ybin_pred,target_names=["Mild","Mod+Sev"]))
print(f'Balanced Accuracy Score : {balanced_accuracy_score(ybin_test,ybin_pred)}')
```

```
ConfusionMatrixDisplay(confusion_matrix(ybin_test,ybin_pred),
                       display_labels=["Mild","Mod+Sev"]).plot();
plt.savefig("figure/XGB_bin_conf.png")
plt.close('all')
```

```
roc_bin(XGB_modelbin)
plt.close('all')
```

C.5 Neural Networks

```
params = {"hidden_layer_sizes":np.arange(1,26,1),
          "learning_rate":["adaptive","constant","invscaling"],
          "activation":["identity","logistic","tanh","relu"]}
grid2 = GridSearchCV(MLPClassifier(random_state=13),param_grid=params,cv=10,verbose=1,
                     n_jobs=-1,scoring="accuracy").fit(Xbin_train,ybin_train)
pd.DataFrame(grid2.cv_results_).to_csv("data/NN_bin_Grid_Res.csv",index=False)
nn_bin = pd.read_csv("data/NN_bin_Grid_Res.csv",sep=";")
plt.figure(figsize=(15,10),dpi=100);
sns.lineplot(x=nn_bin["param_hidden_layer_sizes"],y=nn_bin["mean_test_score"]*100,
             hue=nn_bin["param_activation"]);
plt.xlabel("Hidden Layer Sizes");
plt.ylabel("Eğitim Verisi Doğruluk Skoru (%)");
plt.legend(title="Aktivasyon Fonksiyonu");
plt.savefig("figure/NN_bin_Grid_Graph.png");
```

```
print(f'En İyi Parametreler : {grid2.best_params_}')
```

```
NN_modelbin = MLPClassifier(hidden_layer_sizes=19,
                            learning_rate='adaptive',
                            random_state=13,
                            max_iter=3000).fit(Xbin_train,ybin_train)
ybin_pred = NN_modelbin.predict(Xbin_test)
print(classification_report(ybin_test,ybin_pred,target_names=["Mild","Mod+Sev"]))
print(f'Balanced Accuracy Score : {balanced_accuracy_score(ybin_test,ybin_pred)}')
```

```
ConfusionMatrixDisplay(confusion_matrix(ybin_test,ybin_pred),
                       display_labels=["Mild","Mod+Sev"]).plot();
plt.savefig("figure/nn_bin_conf.png")
```

```
roc_bin(NN_modelbin)
```