

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

Nisan 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 değildir. Çalışmayı 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

<b>Introduction</b>	<b>1</b>
<b>Bölüm 1: Karpal Tünel Sendromu</b>	<b>3</b>
1.1 Epidemiyoloji	4
1.2 Etiyoloji	4
1.3 Semptomlar	4
1.4 Tanı ve Ciddiyet Değerlendirmesi	4
<b>Bölüm 2: Yöntem</b>	<b>9</b>
2.1 K - En Yakın Komşuluk Algoritması (K-NN)	9
2.1.1 K-NN Parametleri	9
2.2 Rassal Ormanlar	10
2.3 XGBoost	10
2.4 Yapay Sinir Ağları	10
2.5 DENEME	10
<b>Bölüm 3: Veri Seti</b>	<b>11</b>
3.1 Çok Sınıflı(Multiclass) Sınıflama Problemi	12
3.1.1 K-En Yakın Komşuluk Modeli	12
3.1.2 Rassal Ormanlar Modeli	15
3.1.3 eXtreme Gradient Boosting (XGBoost)	18
3.1.4 Neural Networks (Yapay Sinir Ağları)	21
3.1.5 Çok Sınıflı Sınıflama Probleminin Modellerinin Değerlendirdirilmesi	24
3.2 İki Sınıflı Sınıflama	25
3.2.1 K-En Yakın Komşuluk Modeli	25
3.2.2 Rassal Ormanlar Modeli	28
3.2.3 XGBoost	31
3.2.4 Neural Network (Yapay Sinir Ağları) Modeli	34
3.2.5 İki Sınıflı Sınıflama Probleminin Modellerinin Değerlendirdirilmesi	37
<b>Bölüm 4: Bölüm 4 Başlık</b>	<b>39</b>
4.1 Bu bir alt başlık	39
4.1.1 Bu ikinci seviye bir alt başlık	39

<b>Sonuç</b> . . . . .	<b>41</b>
<b>Kaynaklar</b> . . . . .	<b>43</b>
<b>Ek A: Gerekli Paketlerin Yüklenmesi ve Verilerin İşlenmesi</b> . . . . .	<b>45</b>
<b>Ek B: 3 Sınıflı</b> . . . . .	<b>49</b>
B.1 Roc Curve ve OneVsRest . . . . .	49
B.2 KNN . . . . .	50
B.3 Rassal Ormanlar . . . . .	51
B.4 XGBoost . . . . .	52
B.5 Neural Network . . . . .	53
<b>Ek C: İKİ SINIFLI</b> . . . . .	<b>55</b>
C.1 Preprocess . . . . .	55
C.2 KNN . . . . .	56
C.3 Rassal Ormanlar . . . . .	57
C.4 XGBoost . . . . .	57
C.5 Neural Networks . . . . .	58

# Tablo Listesi

3.1	Sayısal Değişkenlerin Tanımlayıcı İstatistikleri . . . . .	11
3.2	Değişkenlerin Bağımlı Değişkene Göre Tanımlayıcı İstatistikleri (P-Value Değerleri Tek Yönlü Varyans Analiz Testi ile Elde Edilmiştir.) . . . . .	11
3.3	Katagorik Değişkenlerin Bağımlı Değişkence Frekans Dağılımı (P-Value Değerleri Ki-Kare Bağımsızlık Testi ile Elde Edilmiştir.) . . . . .	12



# Şekil Listesi

1.1	Karpal Tünel Anatomisi ve Medyan Sinirin Sıkışması . . . . .	3
1.2	Elektronörofizyolojik Test (Kumaş, 2005) . . . . .	5
1.3	Phalen ve Tinel Testi . . . . .	6
1.4	Karpal kompresyon ve GMSS Testi . . . . .	6
1.5	Ultrasonografi ve Düz radyografi . . . . .	6
1.6	Bilgisayarlı tomografi . . . . .	7
3.1	Model Train Scores . . . . .	13
3.2	K-En Yakın Komşuluk Modeli Karmaşıklık Matrisi . . . . .	14
3.3	K-En Yakın Komşuluk Modeli ROC Eğrisi ve AUC Değeri . . . . .	15
3.4	Model Train Scores . . . . .	16
3.5	Rassal Ormanlar Modeli Karmaşıklık Matrisi . . . . .	17
3.6	Rassal Ormanlar Modeli ROC Eğrisi ve AUC Değeri . . . . .	18
3.7	Model Train Scores . . . . .	19
3.8	XGBoost Modeli Karmaşıklık Matrisi . . . . .	20
3.9	XGBoost Modeli ROC Eğrisi ve AUC Değeri . . . . .	21
3.10	Model Train Scores . . . . .	22
3.11	Sinir Ağları Modeli Karmaşıklık Matrisi . . . . .	23
3.12	Sinir Ağları Modeli ROC Eğrisi ve AUC Değeri . . . . .	24
3.13	Model Train Scores . . . . .	25
3.14	K-En Yakın Komşuluk Modeli Karmaşıklık Matrisi . . . . .	27
3.15	K-En Yakın Komşuluk Modeli ROC Eğrisi ve AUC Değeri . . . . .	28
3.16	Model Train Scores . . . . .	29
3.17	Rassal Ormanlar Modeli Karmaşıklık Matrisi . . . . .	30
3.18	Rassal Ormanlar Modeli ROC Eğrisi ve AUC Değeri . . . . .	31
3.19	Model Train Scores . . . . .	32
3.20	XGBoost Modeli Karmaşıklık Matrisi . . . . .	33
3.21	XGBoost Modeli ROC Eğrisi ve AUC Değeri . . . . .	34
3.22	Model Train Scores . . . . .	35
3.23	Yapar Sinir Ağları Modeli Karmaşıklık Matrisi . . . . .	36
3.24	Yapar Sinir Ağları Modeli ROC Eğrisi ve AUC Değeri . . . . .	37





# Introduction

Welcome to the *R Markdown* thesis template. This template is based on (and in many places copied directly from) the Reed College LaTeX template, but hopefully it will provide a nicer interface for those that have never used TeX or LaTeX before. Using *R Markdown* will also allow you to easily keep track of your analyses in **R** chunks of code, with the resulting plots and output included as well. The hope is this *R Markdown* template gets you in the habit of doing reproducible research, which benefits you long-term as a researcher, but also will greatly help anyone that is trying to reproduce or build onto your results down the road.

Hopefully, you won't have much of a learning period to go through and you will reap the benefits of a nicely formatted thesis. The use of LaTeX in combination with *Markdown* is more consistent than the output of a word processor, much less prone to corruption or churlashing, and the resulting file is smaller than a Word file. While you may have never had problems using Word in the past, your thesis is likely going to be about twice as large and complex as anything you've written before, taxing Word's capabilities. After working with *Markdown* and **R** together for a few weeks, we are confident this will be your reporting style of choice going forward.

## **Why use it?**

*R Markdown* creates a simple and straightforward way to interface with the beauty of LaTeX. Packages have been written in **R** to work directly with LaTeX to produce nicely formatting tables and paragraphs. In addition to creating a user friendly interface to LaTeX, *R Markdown* also allows you to read in your data, to analyze it and to visualize it using **R** functions, and also to provide the documentation and commentary on the results of your project. Further, it allows for **R** results to be passed inline to the commentary of your results. You'll see more on this later.

## **Who should use it?**

Anyone who needs to use data analysis, math, tables, a lot of figures, complex cross-references, or who just cares about the final appearance of their document should use *R Markdown*. Of particular use should be anyone in the sciences, but the user-friendly nature of *Markdown* and its ability to keep track of and easily include figures, automatically generate a table of contents, index, references, table of figures, etc. should make it of great benefit to nearly anyone writing a thesis project.



# 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 duyuşal 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

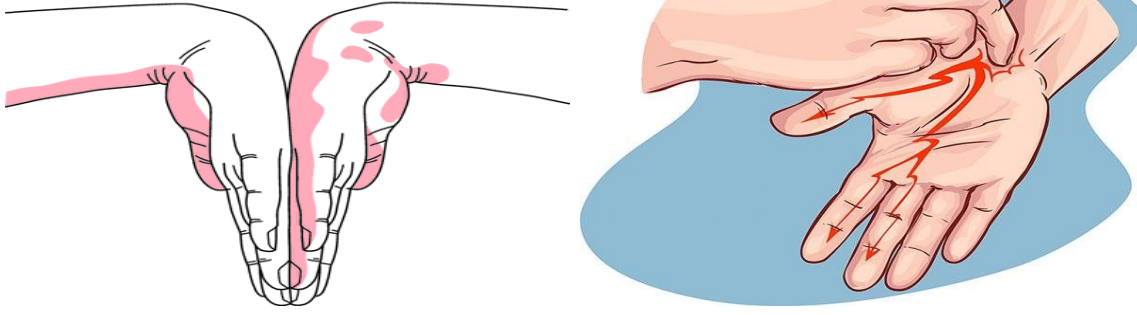
Karpal tünel sendromunda tanı koymak için hastanın hikayesi, klinik semptomlar, fizik muayene bulguları ve bu bulguları destekleyen çeşitli testler kullanılmaktadır(Ghasemi-Rad ve diğerleri, 2014). Bu testler elektronörofizyolojik, provokatif testler ve tıbbi görüntülemeye dayanan testlerdir. Elektronörofizyolojik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



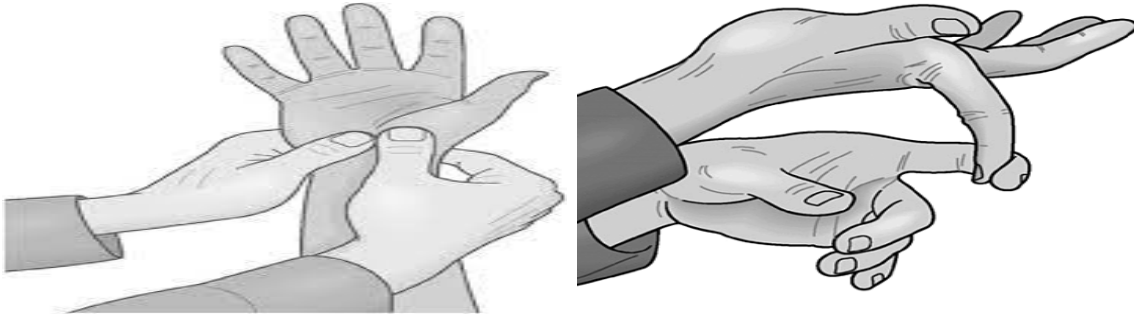
Şekil 1.2: Elektronörofizyolojik Test (Kumaş, 2005)

Provokatif testler hastanın bilek ve parmak eklemlerine fiziksel baskı uygulayacak şekilde bir takım testler uygulanması ve alınan sonuçların değerlendirilmesine 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 pozitifdir.
- Ters Phalen testi
  - 60 saniye boyunca parmaklar yukarı bakacak şekilde el dış yüzleri birleştirilir. Meydan sinir bölgesinde karıncalanma oluşur veya artarsa test pozitifdir.
- Tinel testi
  - Uygulayıcı tarafından karpal tünelin üstüne perküsyon yapılır. Medyan sinir bölgesinde karıncalanma ve elektrik şoku hissi oluşursa test pozitifdir (Kurt, 2020).
- Karpal kompresyon testi
  - El bileği düz tutulurken medyan sinirin yakınına başparmak ile bastırılır. Medyan sinir bölgesinde karıncalanma oluşur veya artarsa test pozitifdir.
- 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 kullanılmasına 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 kullanmış olduğumuz sınıflama algoritmalarına değineceğiz.

### 2.1 K - En Yakın Komşuluk Algoritması (K-NN)

K-NN algoritması, T. M. Cover ve P. E. 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

Torbalama yöntemi ile oluşturulmuş  $n$  adet karar ağacının sonuca göre çalışır.

## 2.3 XGBoost

NO COMMENT

## 2.4 Yapay Sinir Ağları

NO COMMENT

## 2.5 DENEME

- Liste1
- Liste2
  - liste3

## Bölüm 3

### Veri Seti

Veri setindeki düşük varyanslı değişken sayısı : 0

Tablo 3.1: Sayısal Değişkenlerin Tanımlayıcı İstatistikleri

	Overall
Age,years (mean $\pm$ SD)	58 $\pm$ 10.8
BMI, kg/m <sup>2</sup> (mean $\pm$ SD)	24.8 $\pm$ 3.4
Duration, months (mean $\pm$ SD)	8.3 $\pm$ 9.6
NRS (mean $\pm$ SD)	4.4 $\pm$ 1.8
CSA, mm <sup>2</sup> (mean $\pm$ SD)	15.2 $\pm$ 4.3
PB, mm (mean $\pm$ SD)	2.5 $\pm$ 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 $\pm$ SD)	57.3 $\pm$ 10.6	59.2 $\pm$ 10.8	57.8 $\pm$ 11.2	0.069
BMI, kg/m <sup>2</sup> (mean $\pm$ SD)	24.2 $\pm$ 3.4	24.7 $\pm$ 3	25.8 $\pm$ 3.7	0
Duration, months (mean $\pm$ SD)	4.3 $\pm$ 5	8.5 $\pm$ 8.2	15.9 $\pm$ 12.8	0
NRS (mean $\pm$ SD)	3.3 $\pm$ 1.3	4.9 $\pm$ 1.5	6.1 $\pm$ 1.5	0
CSA, mm <sup>2</sup> (mean $\pm$ SD)	13.2 $\pm$ 3	15.4 $\pm$ 3.2	18.9 $\pm$ 5	0
PB, mm (mean $\pm$ SD)	2.1 $\pm$ 0.8	2.6 $\pm$ 2.4	3.1 $\pm$ 2.3	0

Tablo 3.3: Katagorik Değişkenlerin Bağımlı Değişken 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 El Kasılması, 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

Bu çalışmada, KTS ciddiyet sınıflandırması için K-En Yakın Komşuluk, Rassal Ormanlar, 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

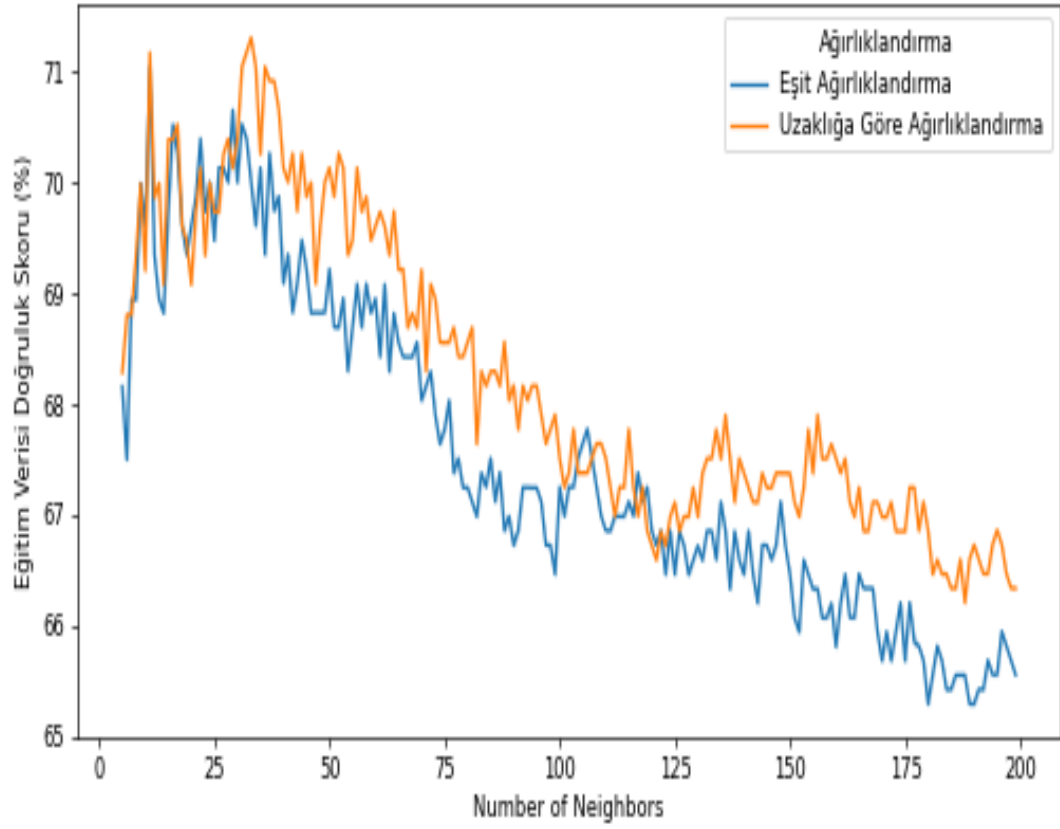
Bu bölümde KTS ciddiyet sınıflandırması için hedef değişkenin 3 farklı ciddiyet düzeyine sahip olduğu durumda farklı sınıflama algoritmaları ile ciddiyet düzeyinin tahminlenmesi amaçlanmıştır.

### 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: Model Train Scores

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

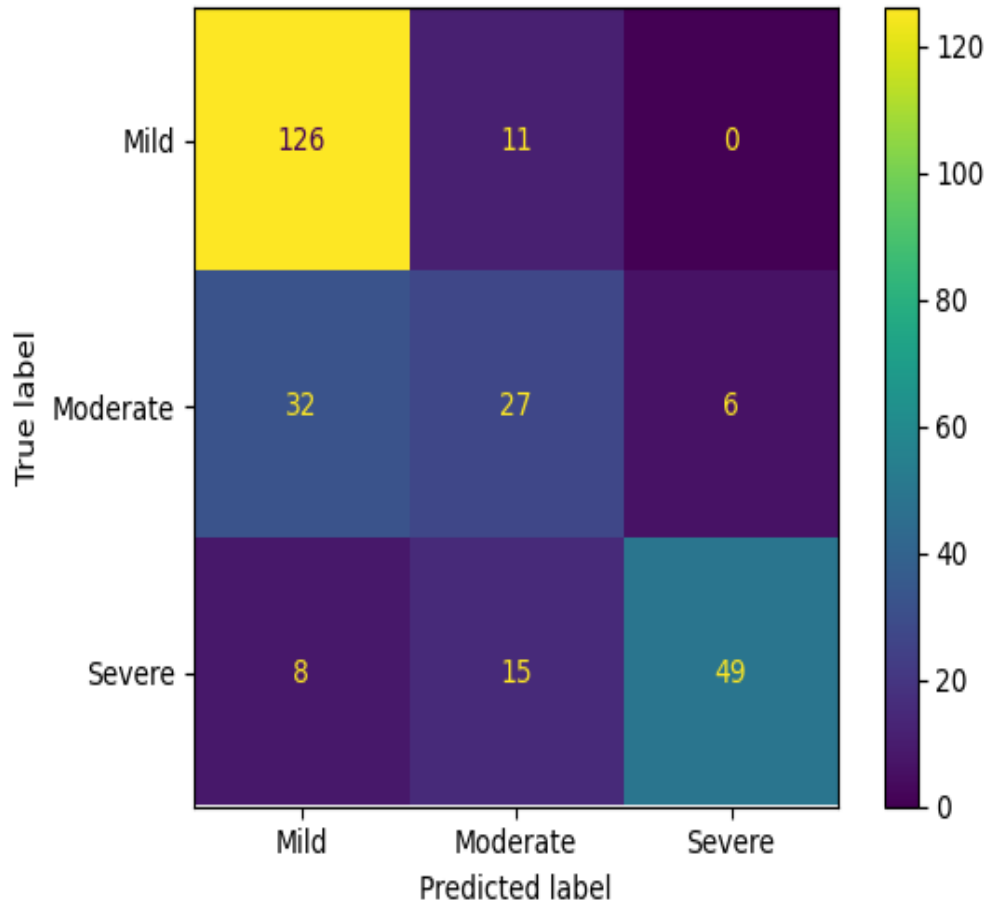
- 'algorithm': 'auto'
- 'n\_neighbors': 33
- 'weights': 'distance'

### En İyi Parametrelili Model

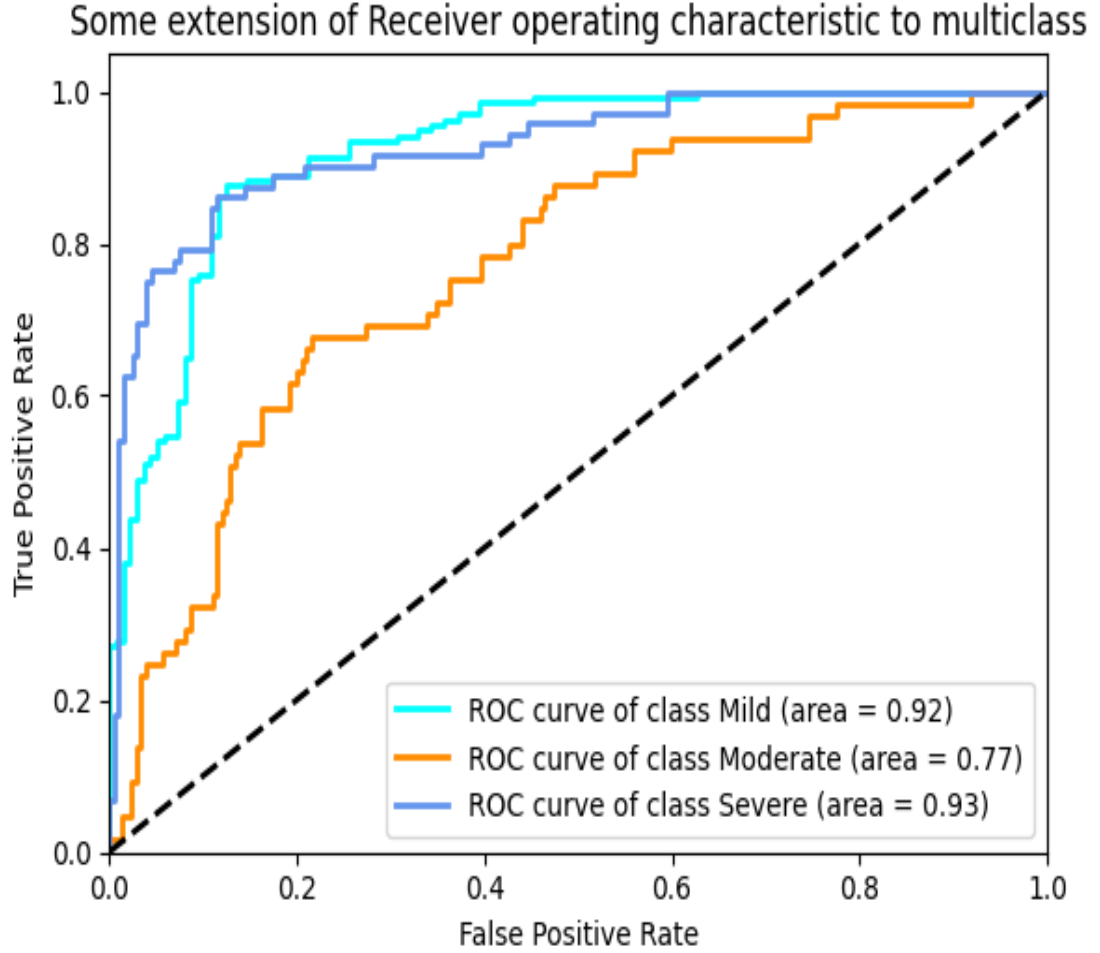
Bulunan parametrelerle kurulan modelin sınıflandırma metrikleri aşağıdaki gibidir.

	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: K-En Yakın Komşuluk Modeli Karmaşıklık Matrisi



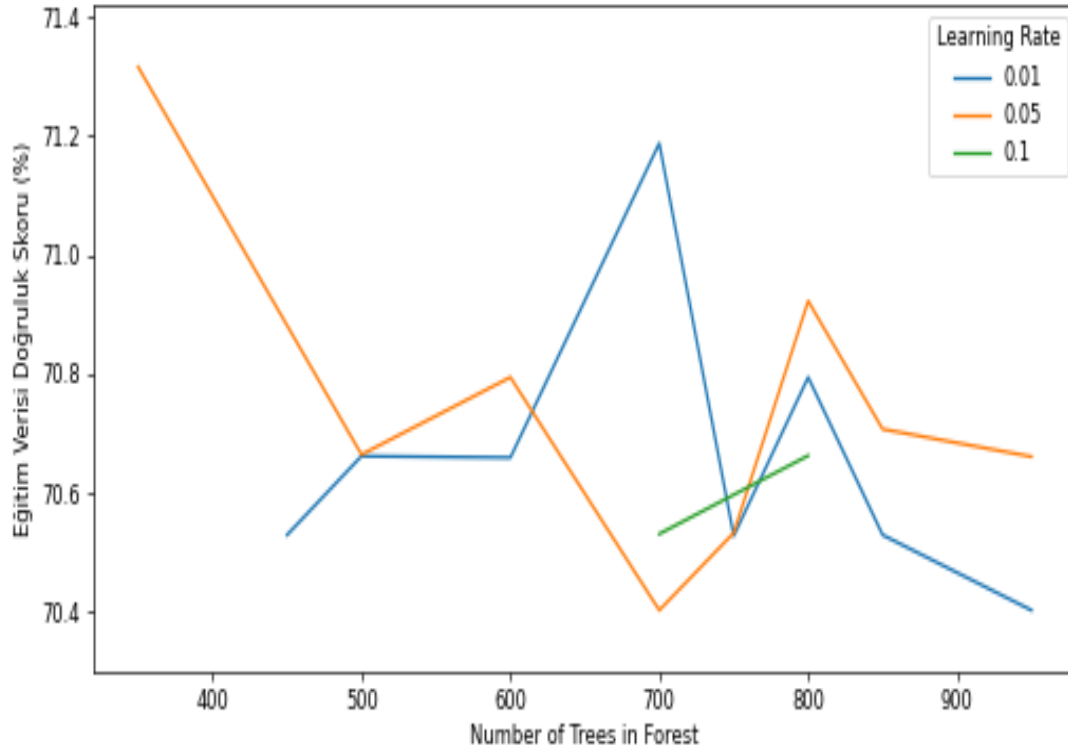
Şekil 3.3: K-En Yakın Komşuluk Modeli ROC Eğrisi ve AUC Değeri

### 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: Model Train Scores

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

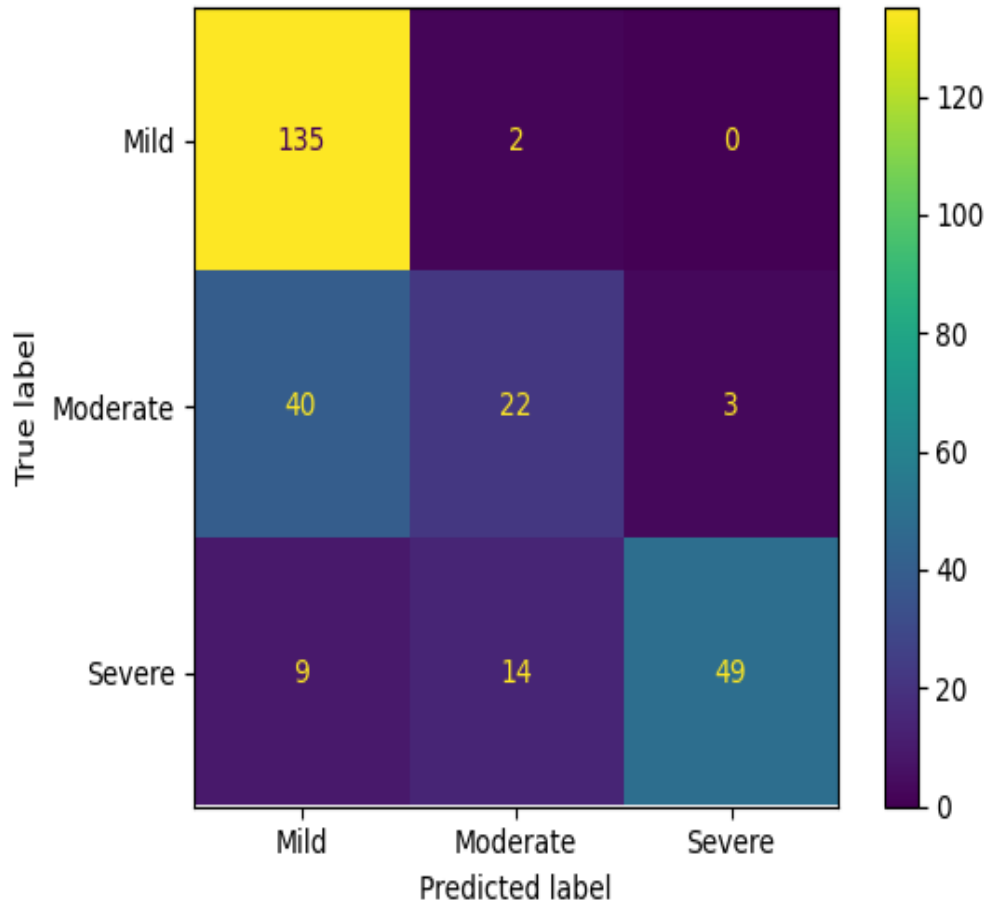


### En İyi Parametrelili Model

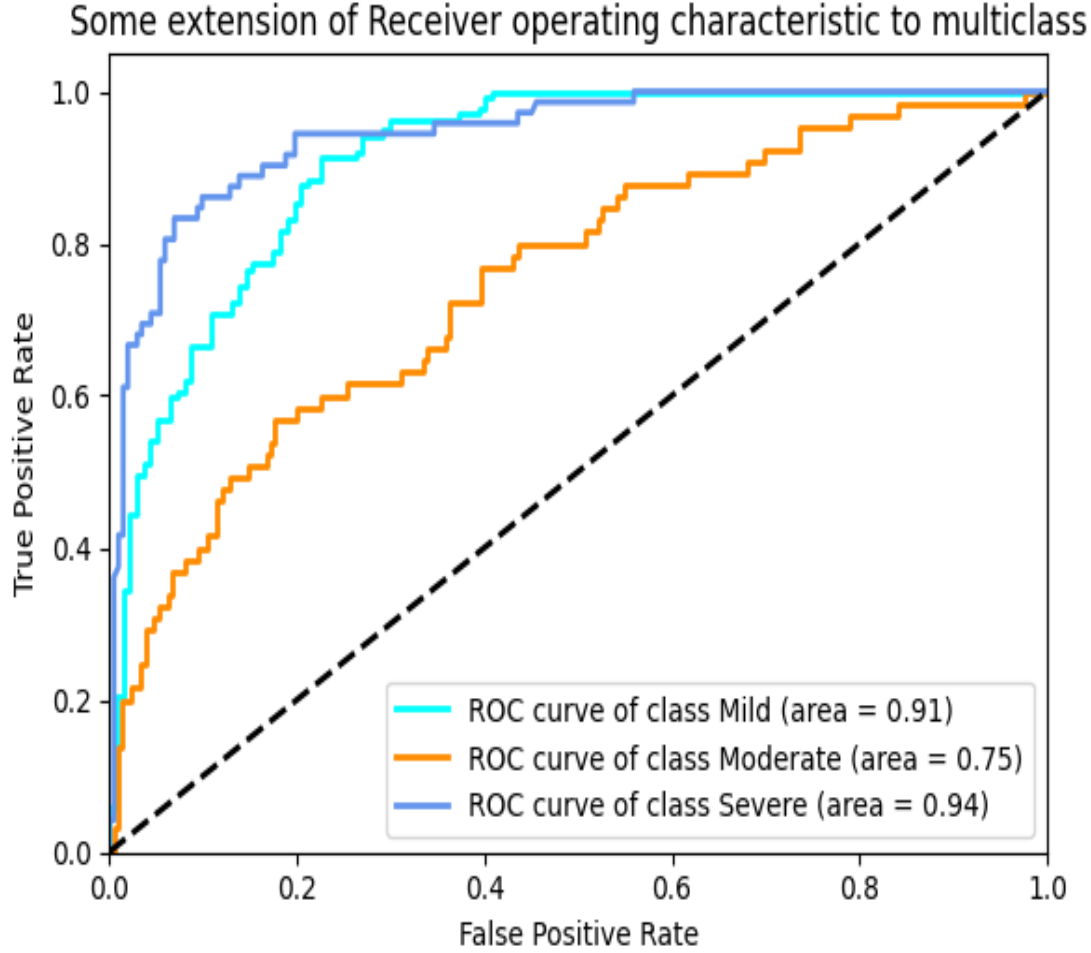
Bulunan parametrelerle kurulan modelin sınıflandırma metrikleri aşağıdaki gibidir.

	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: Rassal Ormanlar Modeli Karmaşıklık Matrisi



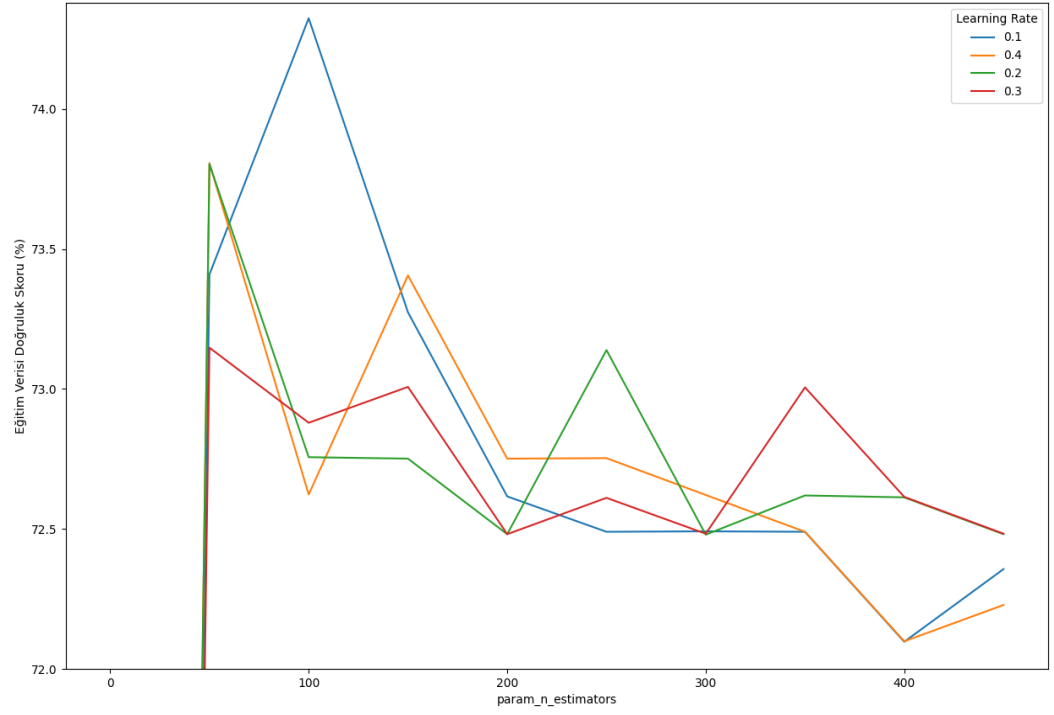
Şekil 3.6: Rassal Ormanlar Modeli ROC Eğrisi ve AUC Değeri

### 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: Model Train Scores

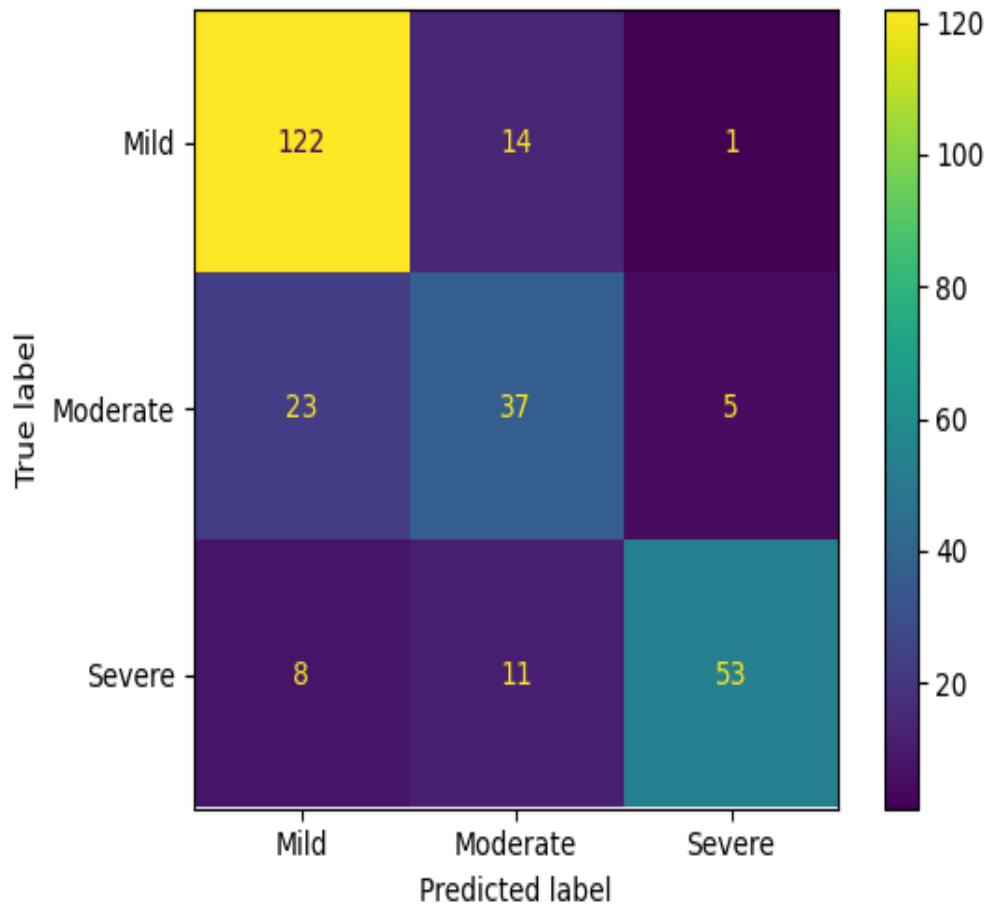
XGBoost modeli için en yüksek doğruluk oranı aşağıdaki parametreler ile bulunmuştur;

- 'eta':0.1
- 'max\_depth':3
- 'min\_child\_weight':10
- 'n\_estimators':100
- 'objective':'multi:softprob'
- 'sumsample':0.5

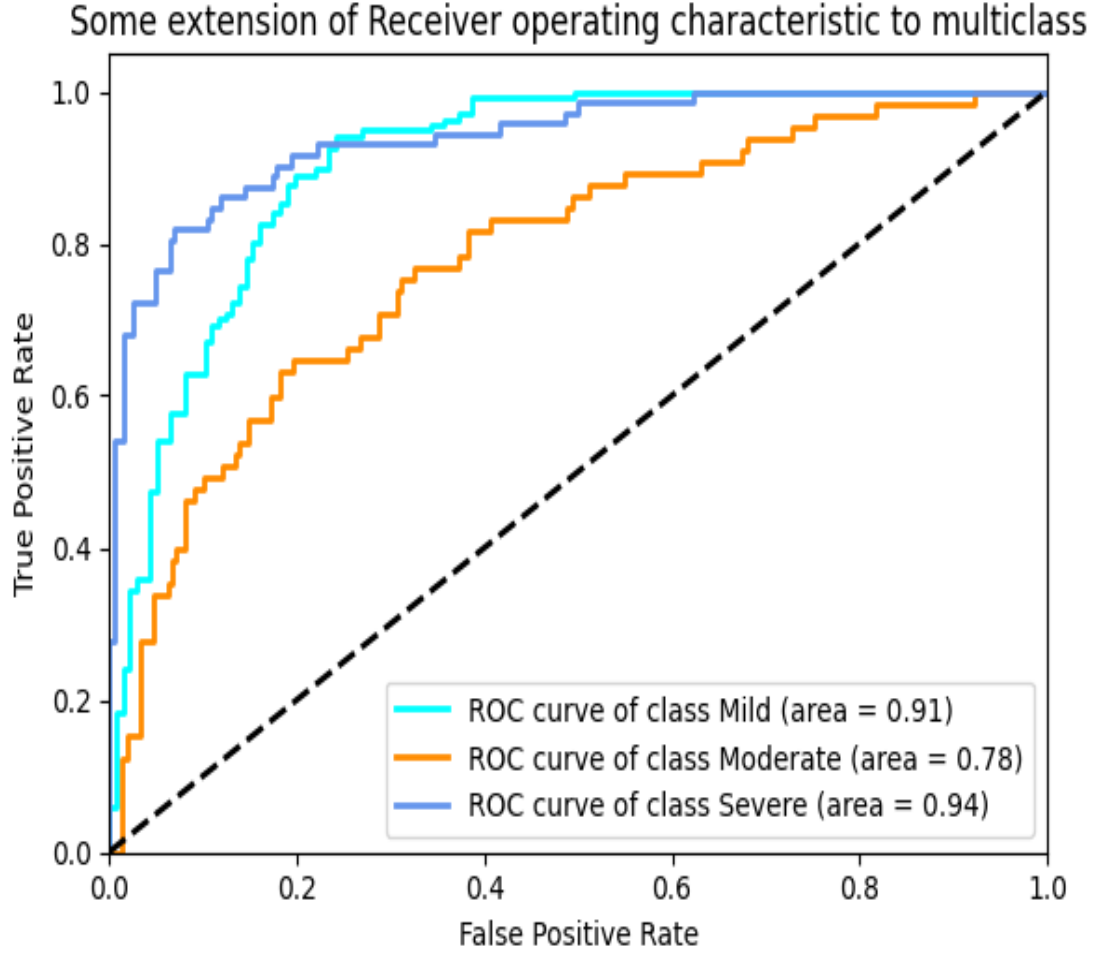
Bulunan parametrelerle kurulan modelin sınıflandırma metrikleri aşağıdaki gibidir.

	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: XGBoost Modeli Karmaşıklık Matrisi



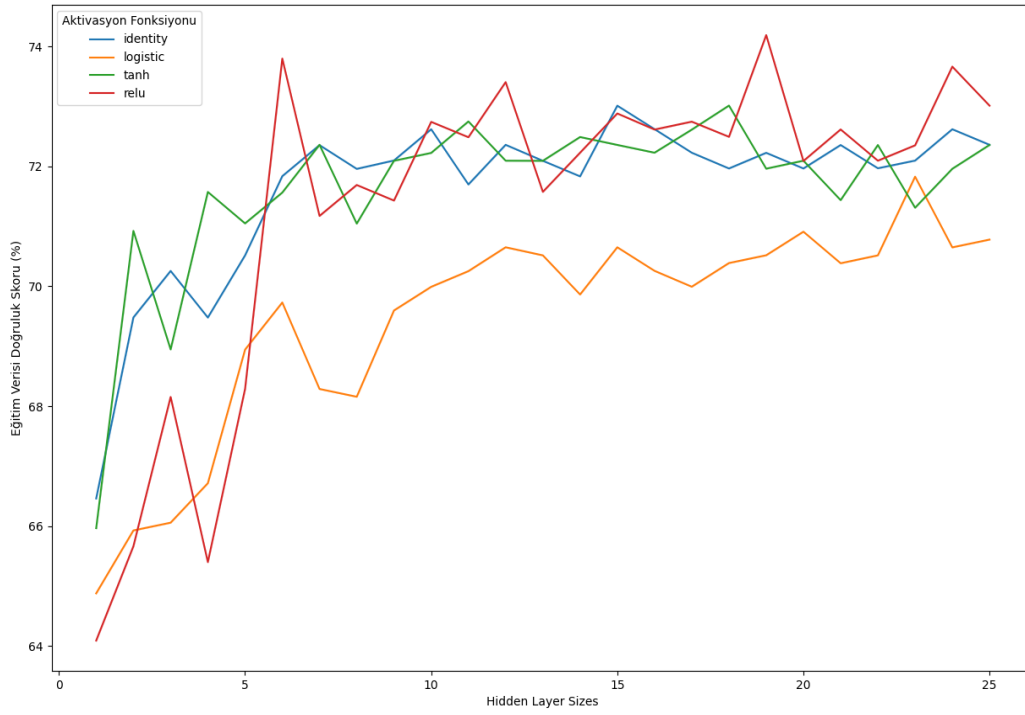
Şekil 3.9: XGBoost Modeli ROC Eğrisi ve AUC Değeri

### 3.1.4 Neural Networks (Yapay Sinir Ağları)

Bu bölümde veri seti üzerinde 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: Model Train Scores

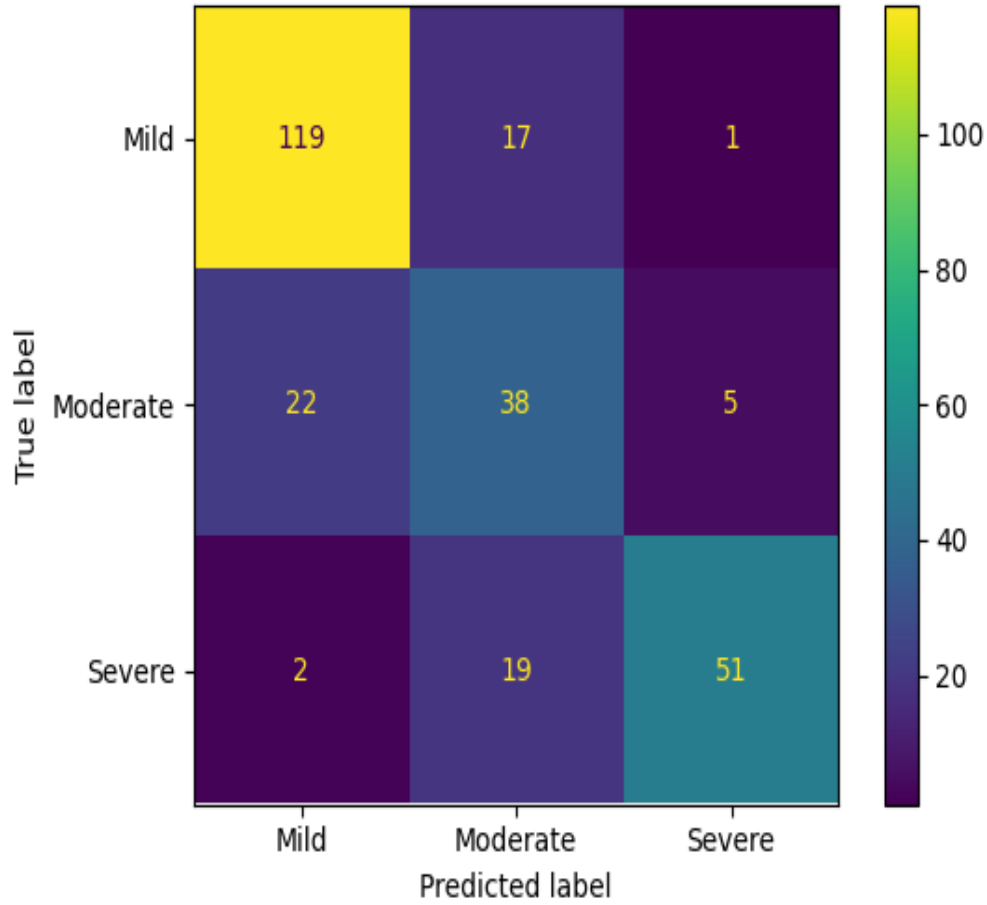
Yapar sinir ağıları modeli için en yüksek doğruluk oranı aşağıdaki parametreler ile bulunmuştur;

- 'activation': 'relu'
- 'hidden\_layer\_sizes': 19
- 'learning\_rate': 'adaptive'

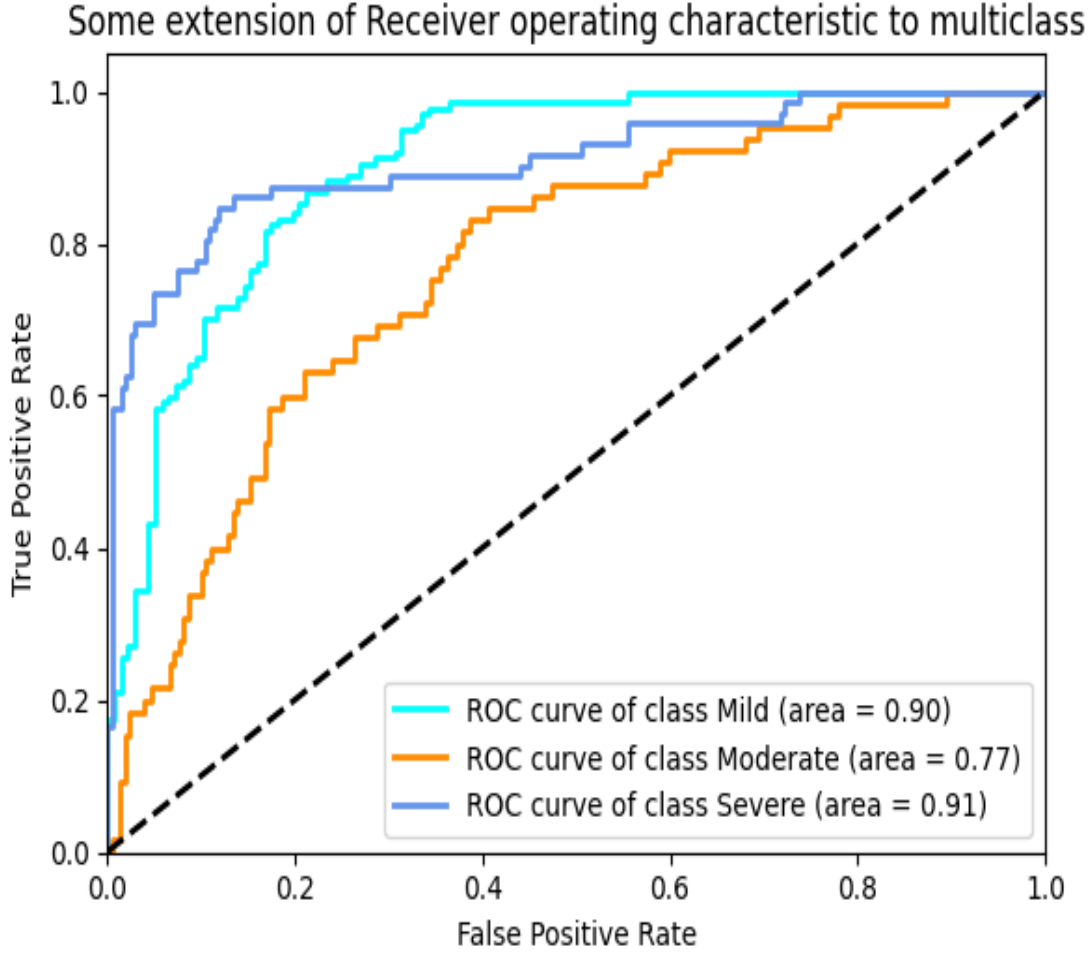
Bulunan parametrelerle kurulan modelin sınıflandırma metrikleri aşağıdaki gibidir.

	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: Sinir Ağları Modeli Karmaşıklık Matrisi



Şekil 3.12: Sinir Ağları Modeli ROC Eğrisi ve AUC Değeri

### 3.1.5 Çok Sınıflı Sınıflama Probleminin Modellerinin Değerlendirilmesi

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

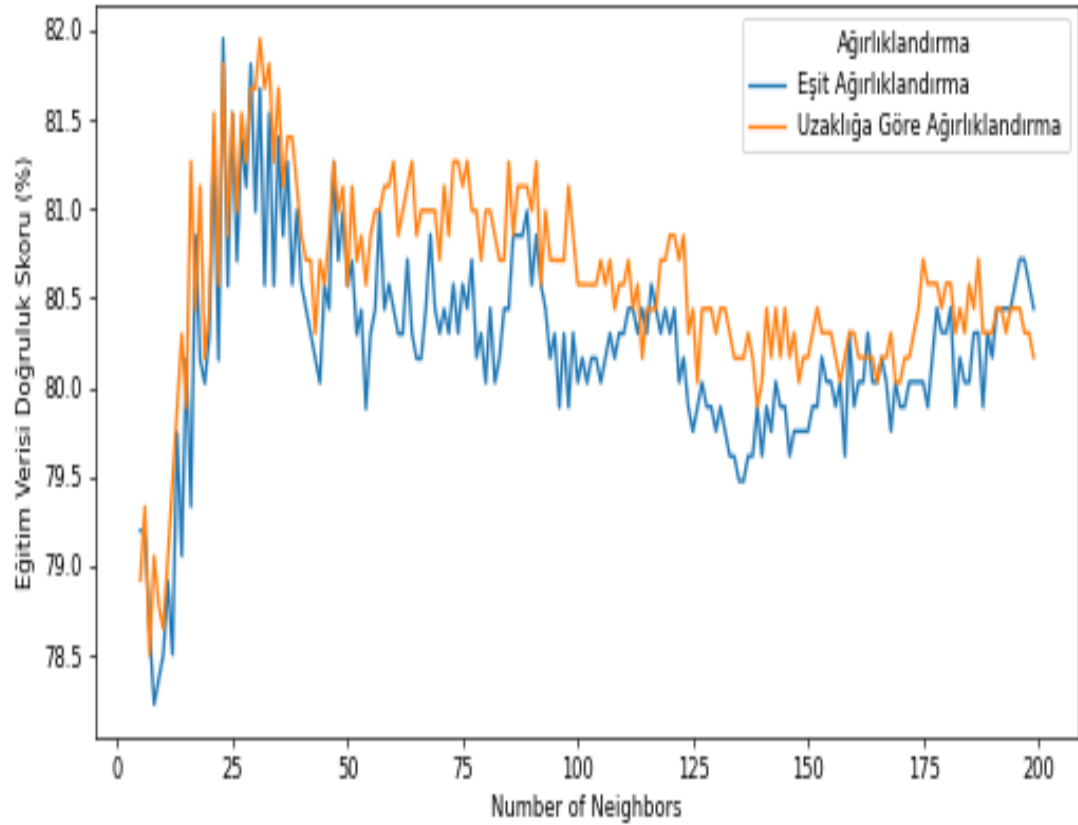
Bu bölümde KTS ciddiye sınıflandırması için hedef değışkenin 3 farklı ciddiye düzeyine sahip olduđu veri seti ‘Moderate’ ve ‘Severe’ ciddiye düzeyleri birleřtirilerek problemin 2 farklı ciddiye düzeyine indirgenip farklı sınıflama algoritmaları ile ciddiye düzeyinin tahminlenmesi amaçlanmıřtır.

### 3.2.1 K-En Yakın Komřuluk Modeli

Bu bölümde veri seti üzerinde K - En yakın komřuluk modeli kullanılmıř ve çıktıları değerdendirilmiřtir.

#### Hiper Parametre Seęimi

Daha önce belirlenen parametre uzayını ve Scikit-Learn kütüphanesinde bulunan GridSearchCV algoritması ile en yüksek doęruluk oranı yakalanana kadar ęalıřması saęlanmıřtır.



řekil 3.13: Model Train Scores

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

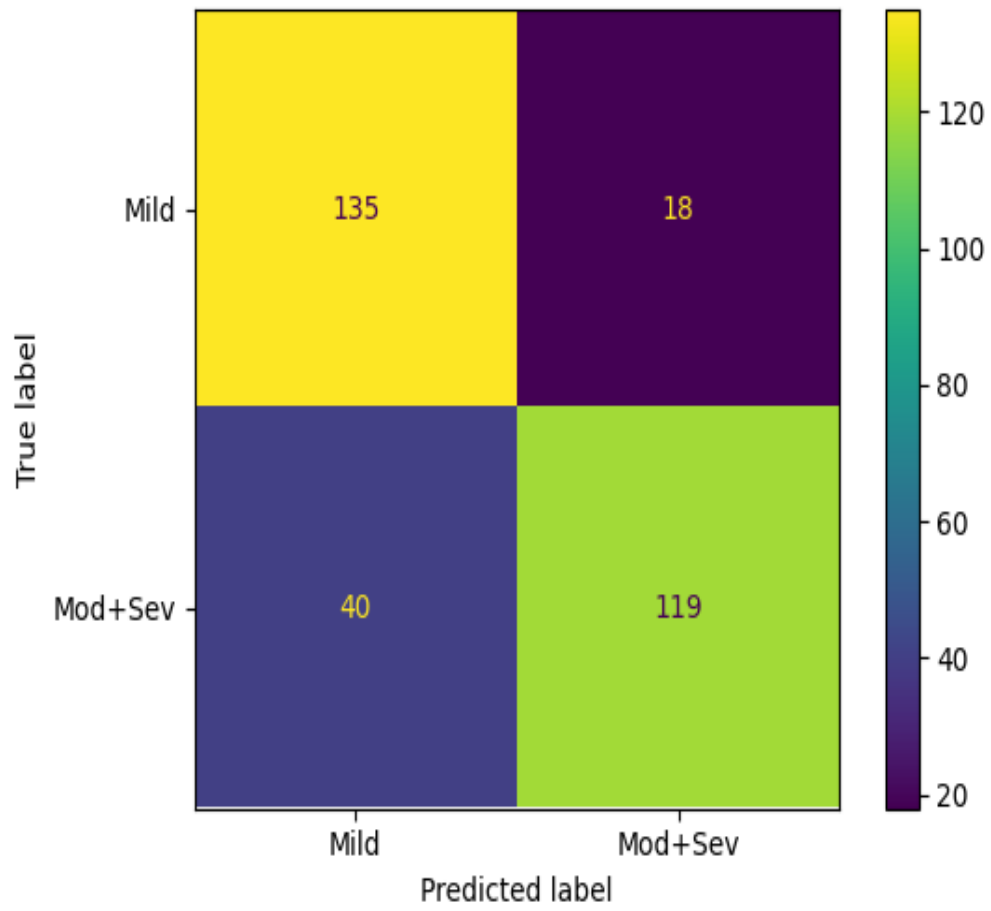
- ‘algorithm’:‘auto’
- ‘n\_neighbors’:23
- ‘weights’:‘uniform’

### En İyi Parametrelili Model

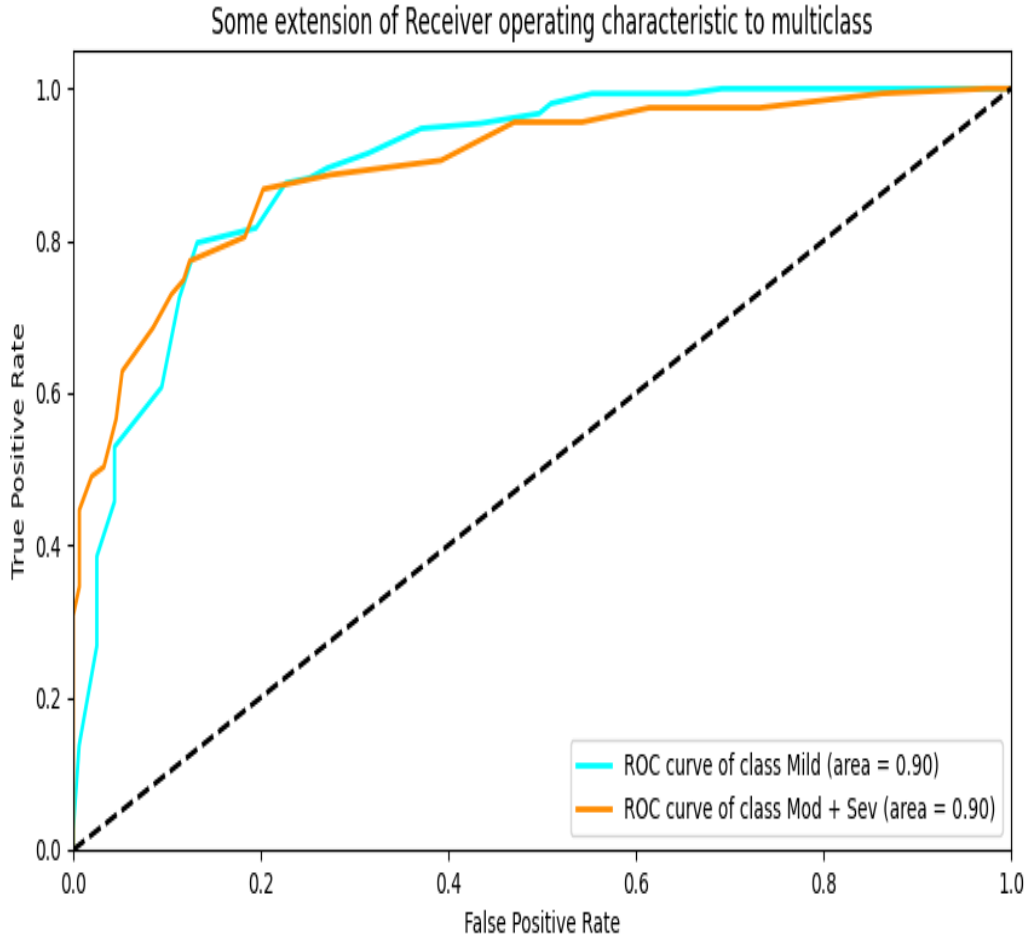
Bulunan parametrelerle kurulan modelin sınıflandırma metrikleri aşağıdaki gibidir.

	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: K-En Yakın Komşuluk Modeli Karmaşıklık Matrisi



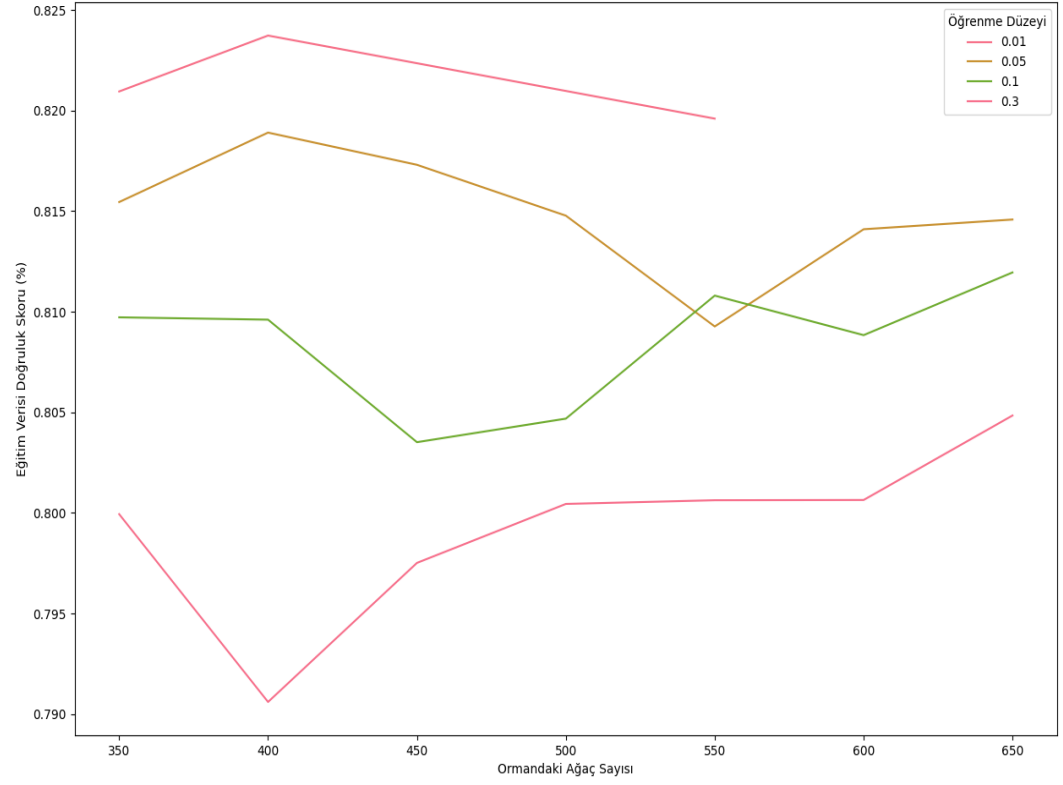
Şekil 3.15: K-En Yakın Komşuluk Modeli ROC Eğrisi ve AUC Değeri

### 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: Model Train Scores

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

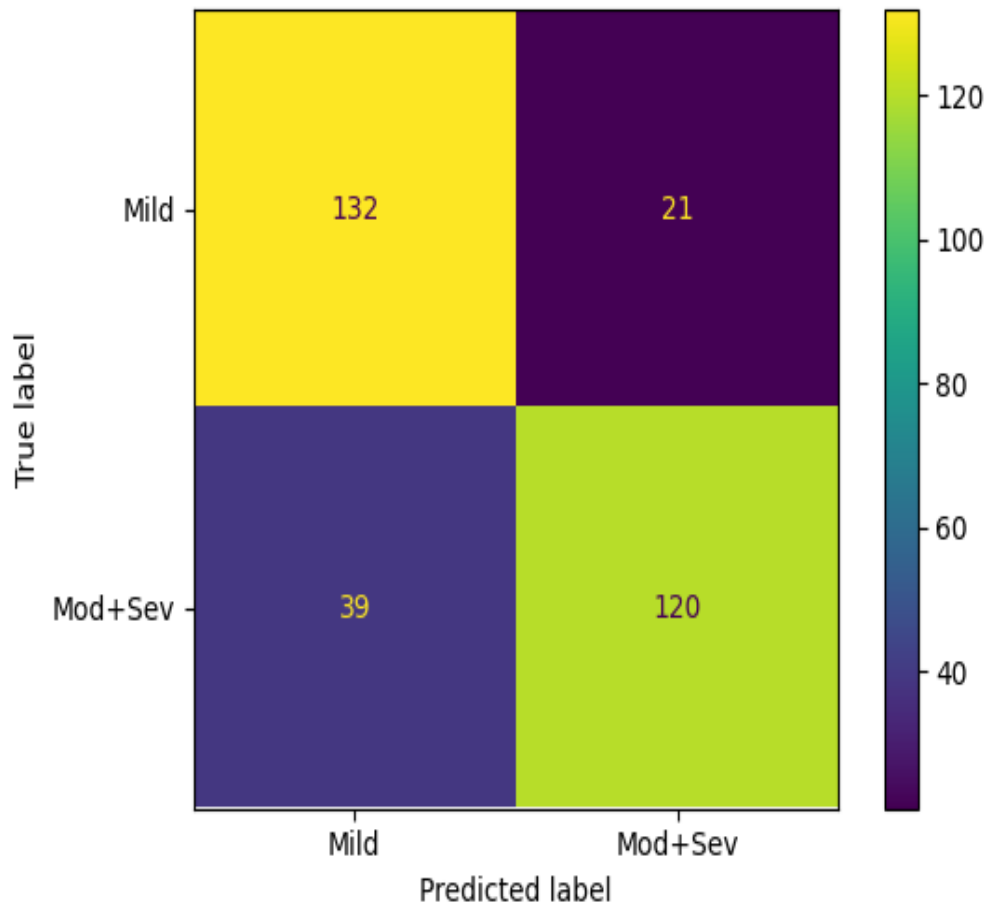
- ‘ccp\_alpha’: 0.01
- ‘max\_features’: ‘auto’
- ‘max\_samples’: 10
- ‘n\_estimators’: 400

### En İyi Parametrelili Model

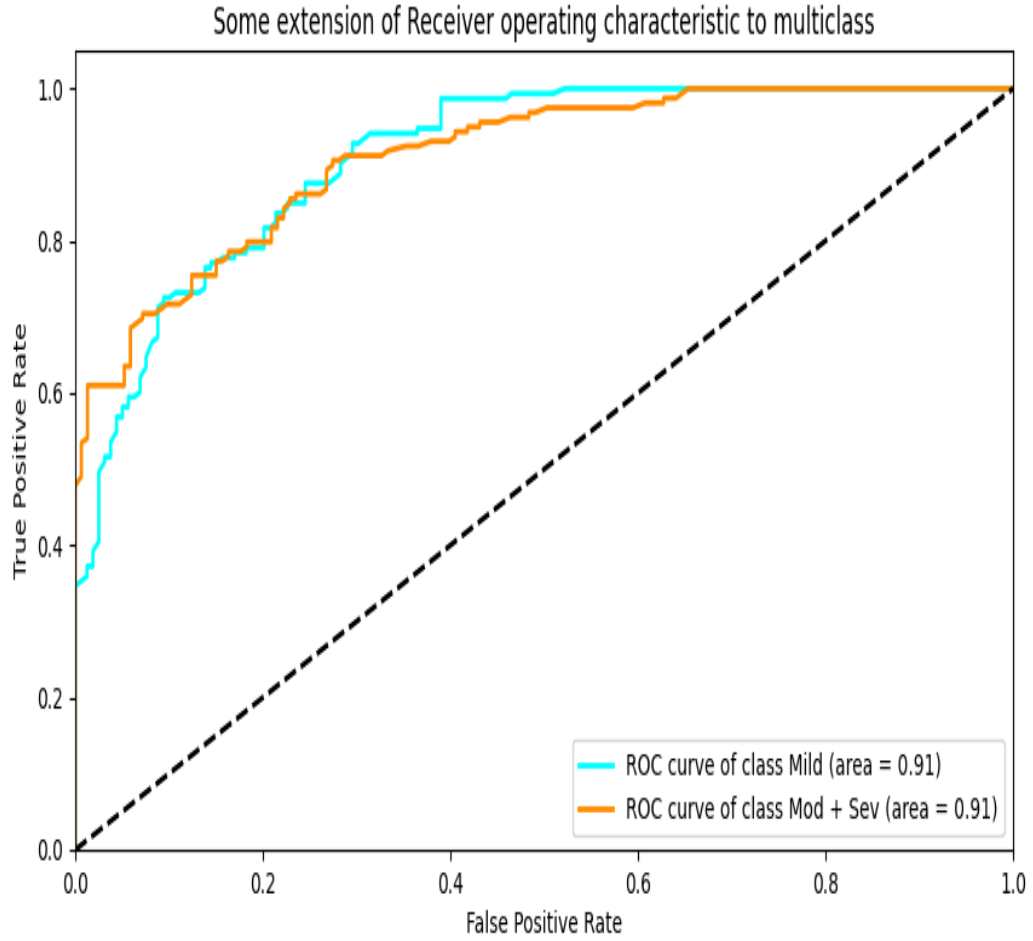
Bulunan parametrelerle kurulan modelin sınıflandırma metrikleri aşağıdaki gibidir.

	precision	recall	f1-score	support
Mild	0.77	0.86	0.81	153
Mod+Sev	0.85	0.75	0.80	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 : 0.8087310395856456



Şekil 3.17: Rassal Ormanlar Modeli Karmaşıklık Matrisi



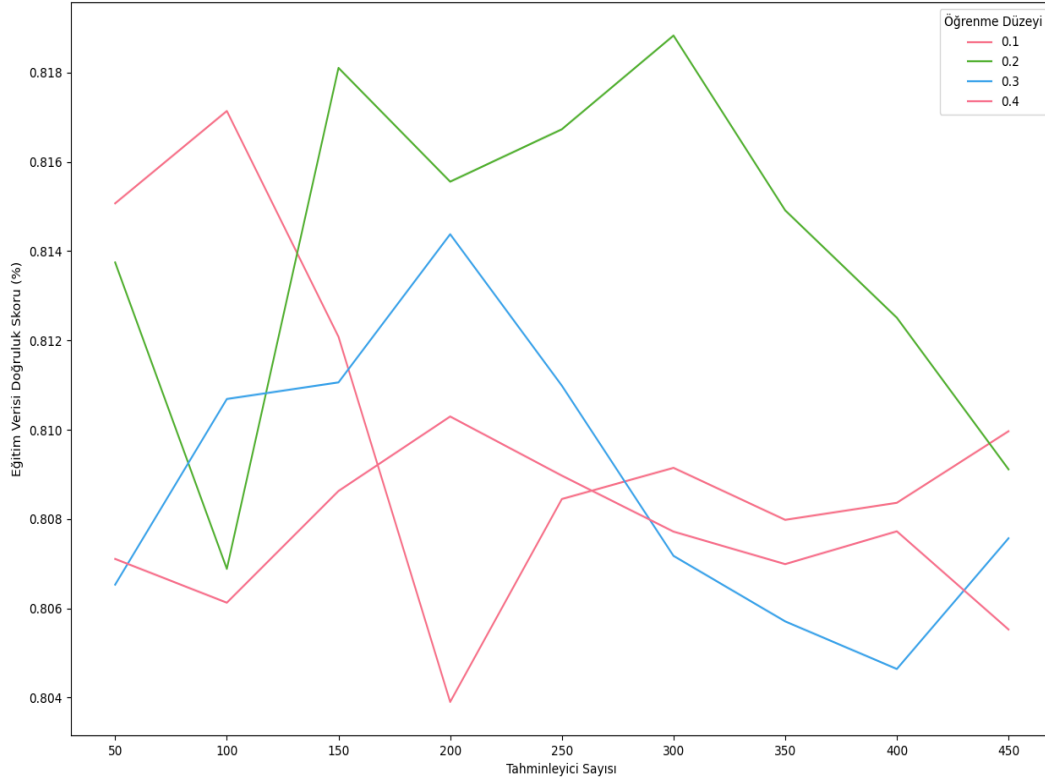
Şekil 3.18: Rassal Ormanlar Modeli ROC Eğrisi ve AUC Değeri

### 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: Model Train Scores

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

- ‘eta’: 0.2
- ‘max\_depth’: 10
- ‘min\_child\_weight’: 10
- ‘n\_estimators’: 400

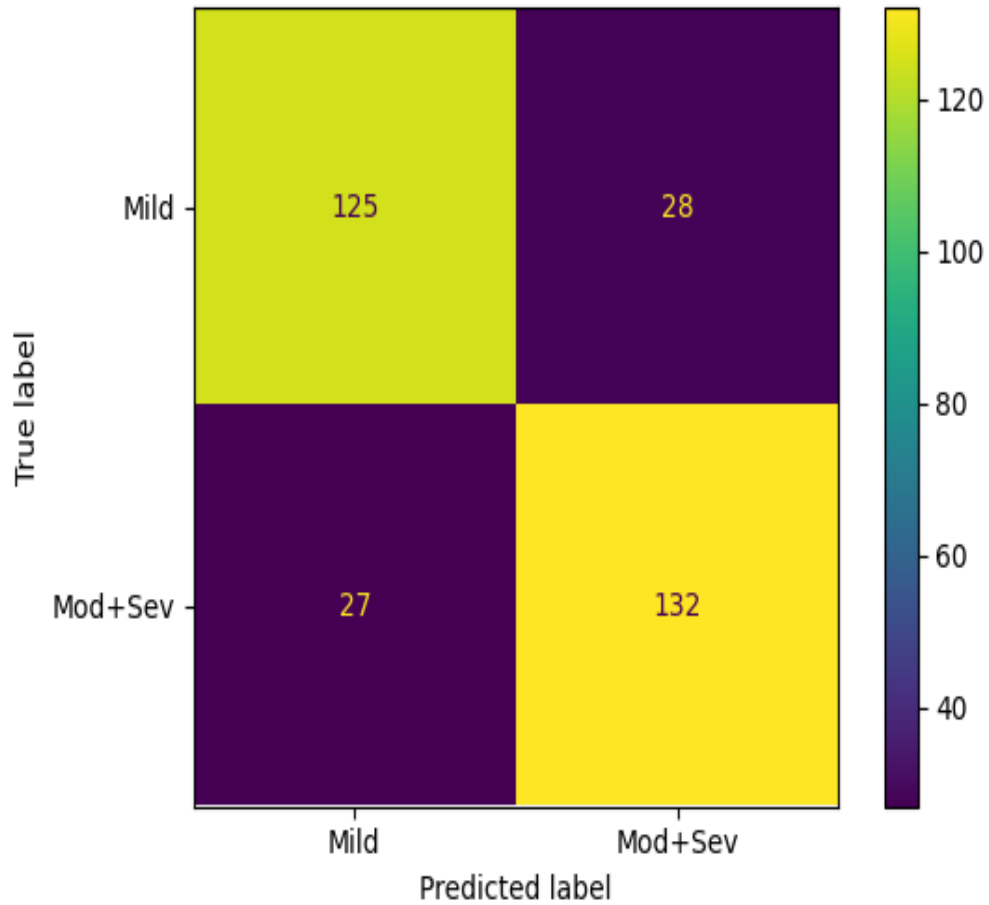


### En İyi Parametrelili Model

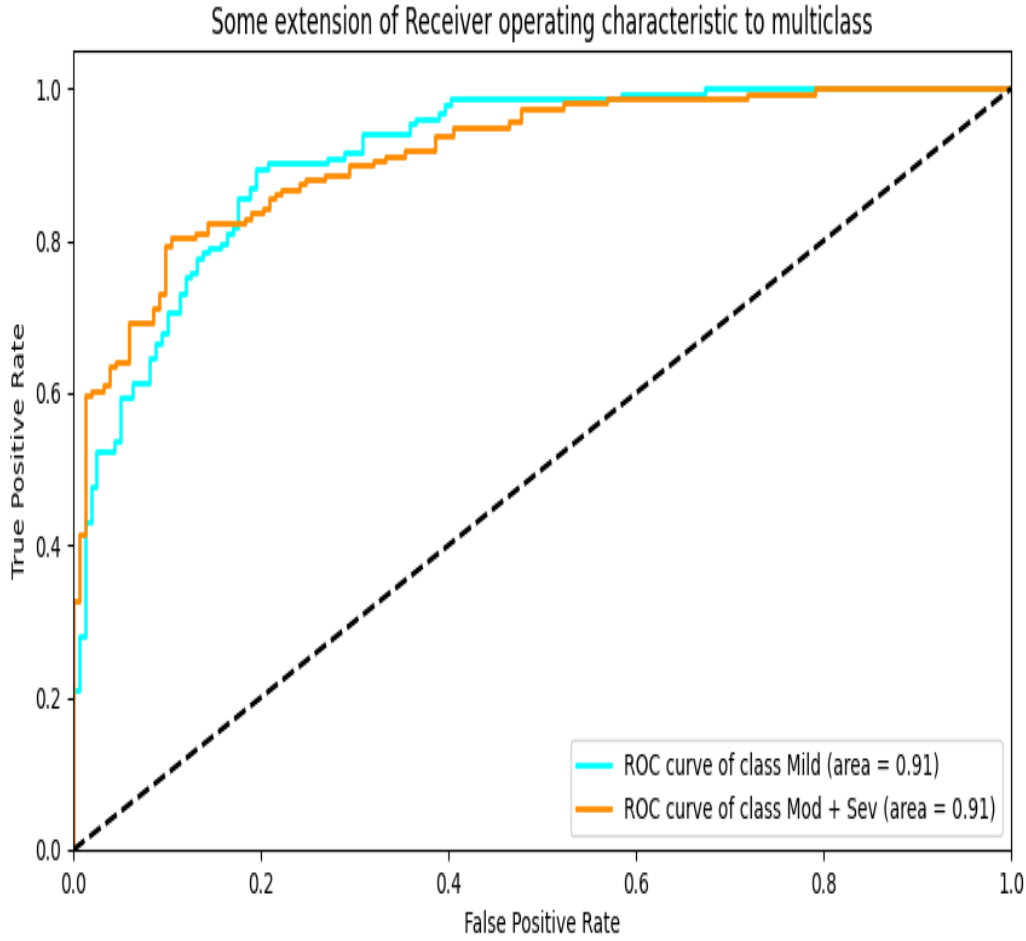
Bulunan parametrelerle kurulan modelin sınıflandırma metrikleri aşağıdaki gibidir.

	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: XGBoost Modeli Karmaşıklık Matrisi



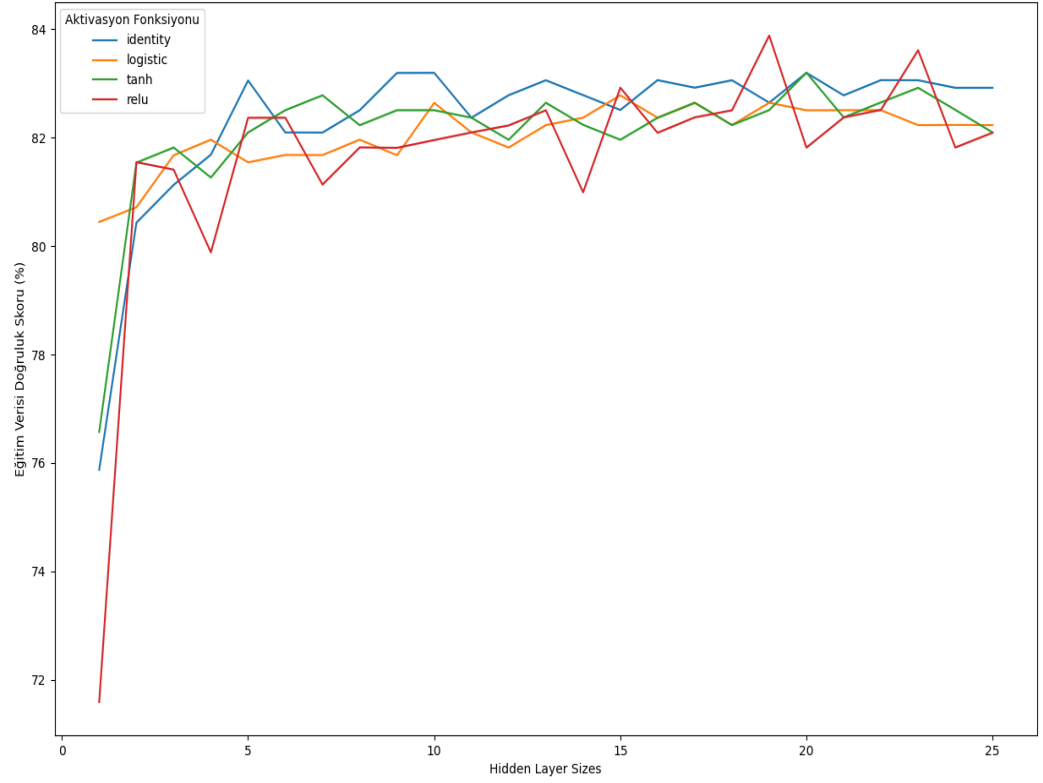
Şekil 3.21: XGBoost Modeli ROC Eğrisi ve AUC Değeri

### 3.2.4 Neural Network (Yapay Sinir Ağları) Modeli

Bu bölümde veri seti üzerinde 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: Model Train Scores

Sinir ağıları modeli için en yüksek doğruluk oranı aşağıdaki parametreler ile bulunmuştur;

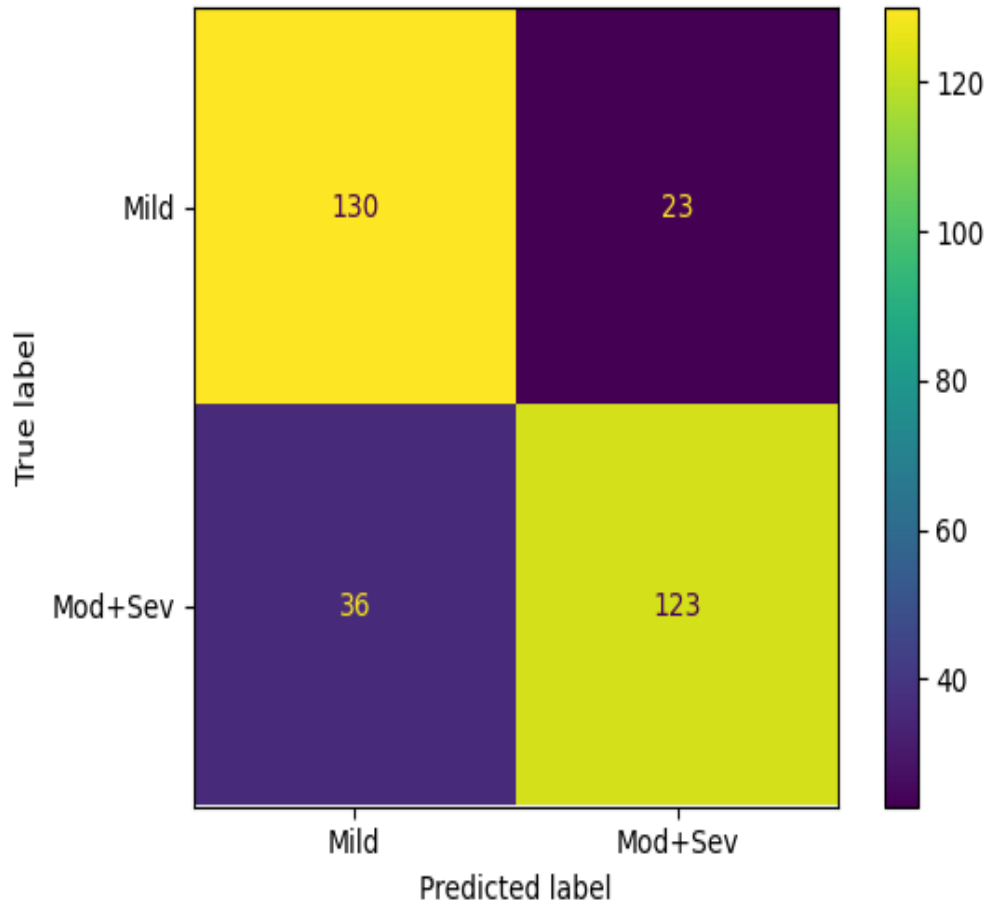
- ‘activation’: ‘relu’
- ‘hidden\_layer\_sizes’: 19
- ‘learning\_rate’: ‘adaptive’

### En İyi Parametrelili Model

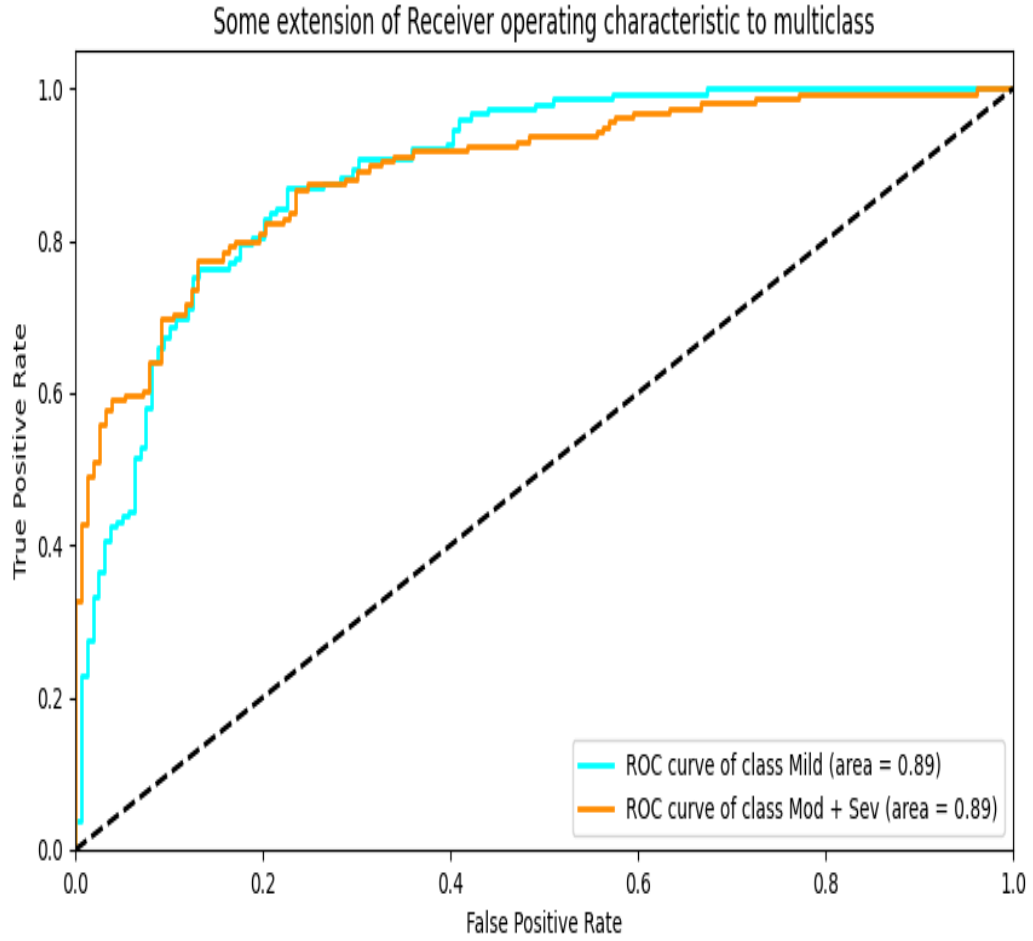
Bulunan parametrelerle kurulan modelin sınıflandırma metrikleri aşağıdaki gibidir.

	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: Yapar Sinir Ağları Modeli Karmaşıklık Matrisi



Şekil 3.24: Yapar Sinir Ağları Modeli ROC Eğrisi ve AUC Değeri

### 3.2.5 İki Sınıflı Sınıflama Probleminin Modellerinin Değerlendirilmesi

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 indirildiği durumda XGBoost modeli diğer modellerden daha iyi sonuç vermektedir.



## Bölüm 4

### Bölüm 4 Başlık

#### 4.1 Bu bir alt başlık

Bu bölümde şu konular yer almaktadır...

##### 4.1.1 Bu ikinci seviye bir alt başlık





# 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.

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

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

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

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.

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). İdiyopatik 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.

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

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

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 tunnel syndrome. *The Journal of Hand Surgery: British & European Volume*, 13(1), 28-34.

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.

Sezgin, M., Incel, N. A., Sevim, S., Yen, H. Çamdevi, As, I. 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.

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

## Ek A

# Gerekli Paketlerin Yüklenmesi ve Verilerin İşlenmesi

```
# 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 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())
#####
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 Ön işleme*

```

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),traindata_P["Severity"],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"]]))
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"]]))
y_test = y_test.to_numpy().ravel()

```





# Ek B

## 3 Sınıflı

### 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
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
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","Uza
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"])
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
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_max_samples);
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"])
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
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")
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"]]
```

```

#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").ma
        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=
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","Mod
plt.savefig("figure/xgb_conf.png");

roc(XGBClassifier(booster="gbtree",eta="0.1",max_depth=3,min_child_weight=10,n_est

```

## 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
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,verbo
pd.DataFrame(gridd.cv_results_).to_csv("NN_GridSearch_Results.csv",sep=";",index=F
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"]
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"  
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"  
plt.savefig("figure/nn_conf.png")
```

```
roc(MLPClassifier(activation="relu",hidden_layer_sizes=19,learning_rate="adaptive",random
```

# Ek C

## İKİ SINIFLI

### 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_s
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=1)
        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_
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
plt.ylabel("Eğitim Verisi Doğruluk Skoru (%)");
plt.xlabel("Number of Neighbors");
plt.legend(title="Ağırlıklandırma",loc="upper right",labels=["Eşit Ağırlıklandırma","Uza
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"])
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,
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=rf_bin)
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"])
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",
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_s
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
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
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,
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",
plt.savefig("figure/nn_bin_conf.png")

```

```

roc_bin(NN_modelbin)

```