T.C.
İSTANBUL ÜNİVERSİTESİ-CERRAHPAŞA
LİSANSÜSTÜ EĞİTİM ENSTİTÜSÜ

YÜKSEK LİSANS TEZİ

Derin Öğrenme ile Optik Koherens Tomografi Görüntülerinden Retinal Bozukluk Tanısı

Gülsüm ARI

DANIŞMAN
Doç. Dr. Abdurrahim AKGÜNDOĞDU

Elektrik-Elektronik Mühendisliği Anabilim Dalı

Elektrik-Elektronik Mühendisliği Programı

Ocak, 2023

TEZ KABUL VE ONAYI

Gülsüm ARI tarafından, **Doç. Dr. Abdurrahim AKGÜNDOĞDU** danışmanlığında hazırlanan **"Derin Öğrenme ile Optik Koherens Tomografi Görüntülerinden Retinal Bozukluk Tanısı"** başlıklı bu çalışma, jürimiz tarafından **05/01/2023** tarihinde yapılan sınav sonucunda **oy birliği** ile başarılı bulunarak **Yüksek Lisans Tezi** olarak kabul edilmiştir.

Tez Jürisi

		İmza	**Sonuç**
DANIŞMAN	Doç. Dr. Abdurrahim AKGÜNDOĞDU İstanbul Üniversitesi-Cerrahpaşa Üniversitesi Elektrik-Elektronik Mühendisliği Anabilim Dalı		☒ Kabul ☐ Ret
ÜYE	Doç. Dr. Yasin ÖZÇELEP İstanbul Üniversitesi-Cerrahpaşa Üniversitesi Elektrik-Elektronik Mühendisliği Anabilim Dalı		☒ Kabul ☐ Ret
ÜYE	Doç. Dr. Atilla ÖZMEN Kadir Has Üniversitesi Elektrik-Elektronik Mühendisliği		☒ Kabul ☐ Ret

BEYAN

Bu tez çalışmasının kendi çalışmam olduğunu, tezin planlanmasından yazımına kadar bütün aşamalarda etik dışı davranışımın olmadığını, bu tezdeki bütün bilgileri akademik ve bilimsel etik kuralları içinde elde ettiğimi, bu tez çalışmasıyla elde edilmeyen bütün bilgi ve yorumlara kaynak gösterdiğimi ve bu kaynakları da kaynaklar listesine aldığımı, yine bu tezin çalışılması ve yazımı sırasında patent ve telif haklarını ihlal edici bir davranışımın olmadığını ve her türlü hukuki sorumluluğu aldığımı kabul ederim.

Gülsüm ARI

(İmza)

Sevgili aileme ve yakın zamanda kaybettiğim Etime teyzeme ithaf ediyorum…

BÜTÇE DESTEKLERİ

DERİN ÖĞRENME İLE OPTİK KOHERENS TOMOGRAFİ GÖRÜNTÜLERİNDEN RETİNAL BOZUKLUK TANISI

Bu tez çalışması için herhangi bir kurumdan bütçe desteği alınmamıştır.

TEŞEKKÜR

Yüksek Lisans tez çalışmalarım sürecinde akademik desteklerini benden esirgemeyen danışman hocam Sayın Doç. Dr Abdurrahim AKGÜNDOĞDU'ya en içten teşekkürlerimi sunarım.

Ayrıca Yüksek Lisans tez çalışmamda manevi desteklerini esirgemeyen beni her daim cesaretlendiren Sayın Prof. Dr. Ali Muhittin ALBORA'ya da teşekkürlerimi sunarım.

Bu süreçte her zaman yanımda olan aileme ve arkadaşlarıma teşekkürü bir borç bilirim.

Ocak 2023 Gülsüm ARI

İÇİNDEKİLER

Sayfa No

ŞEKİL LİSTESİ

Sayfa No

TABLO LİSTESİ

Sayfa No

SEMBOL VE KISALTMA LİSTESİ

Semboller	**Açıklama**
i1,i2,….	: YSA girişleri
w1,w2,….	: YSA ağırlıları
b1,b2,….	: Eşik değerleri
β	: Momentum
Kısaltmalar	**Açıklama**
ESA	: Evrişimli Sinir Ağları
YSA	: Yapay Sinir Ağları
DL	: Deep Learning
ANN	: Artifical Neural Network
ML	: Machine Learning
MLP	: Multi Layer Perceptron
DVM	:Destek Vektör Makineleri
SVM	: Support Vector Machine
CNV	: Choroidal Neovascularization
DME	: Diabetic Macular Edema
CNN	: Convolutional Neural Network
RL	: Reinforcement learning

ÖZET

YÜKSEK LİSANS TEZİ

DERİN ÖĞRENME İLE OPTİK KOHERENS TOMOGRAFİ GÖRÜNTÜLERİNDEN RETİNAL BOZUKLUK TANISI

Gülsüm ARI

İstanbul Üniversitesi-Cerrahpaşa

Lisansüstü Eğitim Enstitüsü

Elektrik-Elektronik Mühendisliği Anabilim Dalı

Elektrik-Elektronik Mühendisliği Programı

Danışman: Doç. Dr. Abdurrahim AKGÜNDOĞDU

Sürekli artan nüfus ve bu nüfusun yapacağı iş yükü insanlarda, onlara yardımcı olabilecek makineler fikrini ortaya çıkarmıştır. İlk başlarda bu makineler şu an akıllı sayılamayacak düzeyde iken günümüzde uzman eksikliğinin ya da zamandan tasarrufun önemli olduğu tüm alanlara yayılmıştır. Öğrenen makine fikrinin ortaya atılmasından beri bu alanda çok fazla çalışma yapılmış olup ve bu öğrenen sistemlerin başarımı kanıtlanmıştır. Şu an bu sistemler tüm veri formatlarına, veri büyüklüklerine uyum sağlayabilmektedir. Tezin yapılma amacı olan retinal bozukluk tanısı da bu öğrenen sistemlerden biri olan bir derin öğrenme modelinin oluşturulduğu ve gerçekleştirildiği bir sistemdir. Derin öğrenme yönteminin kullanıldığı bu çalışmada başta evrişimli sinir ağları olmak üzere, temel makine öğrenmesi algoritmaları anlatılmış olup, çalışmaya adını veren optik koherens görüntü verilerinden oluşan CNV, DME, DRUSEN retinal bozuklukları ve bir normal durumu için eğitilen modelin sınıflandırma ve tahmin gerçekleştirmede başarısını göstermektedir. Modelin gerçekleştirilme aşamasında başta derin öğrenme olmak üzere yapay zekada en çok kullanılan programlama dillerinden biri olan Python dili kullanılmış olup Anaconda Spyder IDE'si üzerinde yazılmıştır. On epok üzerine eğitilen modelde %93,6 eğitim başarımı %92,9 doğruluk başarımı elde edilmiş olup en son

değerlendirilen modelde %82 test başarımı elde edilmiştir. Sınıflandırmadan sonra tahmin aşamasına geçildiğinde model kendi verilen verilerde doğru tahminlerde bulunmuştur. Böylelikle tez amacını gerçekleştirmiş olmuştur. Yapılan tez çalışması, derin öğrenmenin, oftalmolojide kullanımını ve avantajlarını kanıtlamaktadır.

Ocak 2023 , 81 sayfa.

Anahtar kelimeler: Makine Öğrenmesi,Derin öğrenme, ESA, Oftalmoloji

ABSTRACT

M.Sc. THESIS

RETINAL DISORDER DIAGNOSIS FROM OPTICAL COHERENCE TOMOGRAPHY IMAGES with DEEP LEARNING

Gülsüm ARI

İstanbul University-Cerrahpaşa

Institute of Graduate Studies

Department of Electrical and Electronics Engineering

Electrical and Electronics Engineering Programme

Supervisor : Assoc. Prof. Dr. Abdurrahim AKGUNDOGDU

The ever-increasing population and the workload of this population have revealed the idea of machines that can help people. At first, these machines were at a level that could not be considered smart at the moment, but today they have spread to all areas where lack of specialists or time saving is important. A lot of work has been done in this field since the idea of learning machine was introduced and the success of these learning systems has been proven. Currently, these systems can adapt to all data formats and data sizes. The diagnosis of retinal disorder, which is the purpose of the thesis, is a system in which a deep learning model, which is one of these learning systems, is created and implemented. In this study, in which deep learning method is used, basic machine learning algorithms, especially convolutional neural networks, are explained, and the model trained for CNV, DME, DRUSEN retinal disorders and a normal state, which consists of optical coherence image data, which gives its name to the study, shows the success of classification and prediction. Python language, one of the most used programming languages in artificial intelligence, especially deep learning, was used in the realization phase of the model and it was written on the Anaconda Spyder IDE. In the model

trained on ten epochs, 93,6% training accuracy and 92,9% validation accuracy were obtained, and 82% test accuracy was obtained in the last evaluated model. When the estimation phase was passed after the classification, the model made correct predictions on the given data. Thus, the thesis has achieved its purpose. The thesis study proves the use and advantages of deep learning in ophthalmology.

January 2023, 81 pages.

Keywords: Machine Learning,Deep Learning, CNN, Ophtolmology

1. GİRİŞ

Yaşlanma ilerleyen yaş ile vücudun tüm fiziksel ve ruhsal birimlerinde meydana gelen değişikler bütünüdür. “Yaşlanma sürecinden en erken etkilenen fiziksel birim gözdür. Görme duyusunu etkileyen yaşa bağlı doğal bazı değişiklikler olacağı gibi yaşla birlikte daha sık ortaya çıkan ciddi hastalıklar da gözü etkileyebilmektedir” [1]. Bu noktada göz hastalıklarında erken ve doğru teşhis önem taşımaktadır. Yaşa bağlı meydana gelen göz rahatsızlıklarına örnek olarak bu çalışmada kullanılan choroidal neovascularization (CNV), diabetic macular edema (DME) ve drusen verilebilir.

Optik koherens tomografi (OKT), retinal hastalıkların tanısında kullanılan bir tıbbi görüntüleme yöntemidir [2]. OKT, yansıyan ışığın görüntülenmesine dayanmaktadır. Fakat bir kamera gibi yalnızca iki boyutlu görüntü değil, derinlik boyutunu da elde etmektedir.

Bu çalışmanın amacı Makine öğrenmesinin bir alt kolu olan Derin öğrenme ile elde bulunan OKT görüntülerinde herhangi bir uzman yardımı almadan yaşa bağlı göz hastalıklarının tanısını koymaktır. Böylece sorumlu Oftalmoloji uzmanları, hızlı konulan tanı sayesinde erken tedaviye başlayabilecektir.

Makine öğrenmesi, bilgisayarların bir sorun için açık açık programlanmadan öğrenme becerisi kazanabilmesidir. Makine öğrenmesi makinelerin kendilerine gösterilen girdiler ve bu girdilerin sonuçlarına bakarak, yeni girdiler için ilişikli kurallar çıkarmasını, tahminler yapabilmesini sağlamaktır. Tom Mithell makine öğrenmesini daha teknik olarak “Belli bir T görevi ve P performans ölçütü için eğer P performans ölçütü, T görevinde E tecrübeleriyle artıyorsa bu E tecrübelerinden öğrendiği söylenen bilgisayar programıdır” şeklinde tanımlamıştır [3].

Bu tez çalışmasında kullanılmış olan derin öğrenme yöntemi makine öğrenmesinin bir alt yöntemidir. Derin öğrenmeyi makine öğrenmesinden ayıran en önemli özellik makine öğrenmesinde olan kullanıcının kendisinin yaptığı öznitelik çıkarımının derin öğrenmeden katmanlar aracılığıyla yapılmasıdır. Derin öğrenmeye derin adını veren üst üste gelen bu katmanlı ağ mimarisidir. Bu katmanlı yapı sayesinde derin öğrenme modelleri hem öznitelik çıkarımını kendi gerçekleştirmekte hem de üst üste yığılan katmanlar sayesinde hiyerarşik bir

öğrenme işlemi gerçekleştirmektedir. Böylelikle model ilk katmanda öğrendiği bilgiyi bir sonraki katmanda gördüğünde makine öğrenmesi modellerinin aksine tekrar öğrenmek zorunda kalmaz ve önceki katmandan aldığı bilginin üstüne koyarak devam etmektedir. Bu yüzden derin öğrenme modelleri diğer modellere göre daha hızlı çalışmaktadır. Retinal bozukluklar üzerinde çalışan bu modelde derin öğrenmenin kullanılmasının sebeplerinden en önemlisi budur.

Bu çalışmada kullanılan derin öğrenme yöntemi evrişimli sinir ağlarıdır. Bölüm 4.2.'de detaylı olarak anlatılmış olup kısaca görüntü işlemede en çok kullanılan, veriler üzerinde çıkardığı pencereleri filtrelerle çarpan bir algoritmasıdır. Bu filtreyle çarpım olayı evrişimin kendisidir.

Yukarıda belirtilen tüm başlıklar, bu tez çalışmasının ileriki bölümlerinde açıklanmış olup, teze adını veren amaç Python programlamam dili, ilgili derin öğrenme ve yardımcı kütüphaneler kullanılarak modellenmiştir.

2. GENEL KISIMLAR

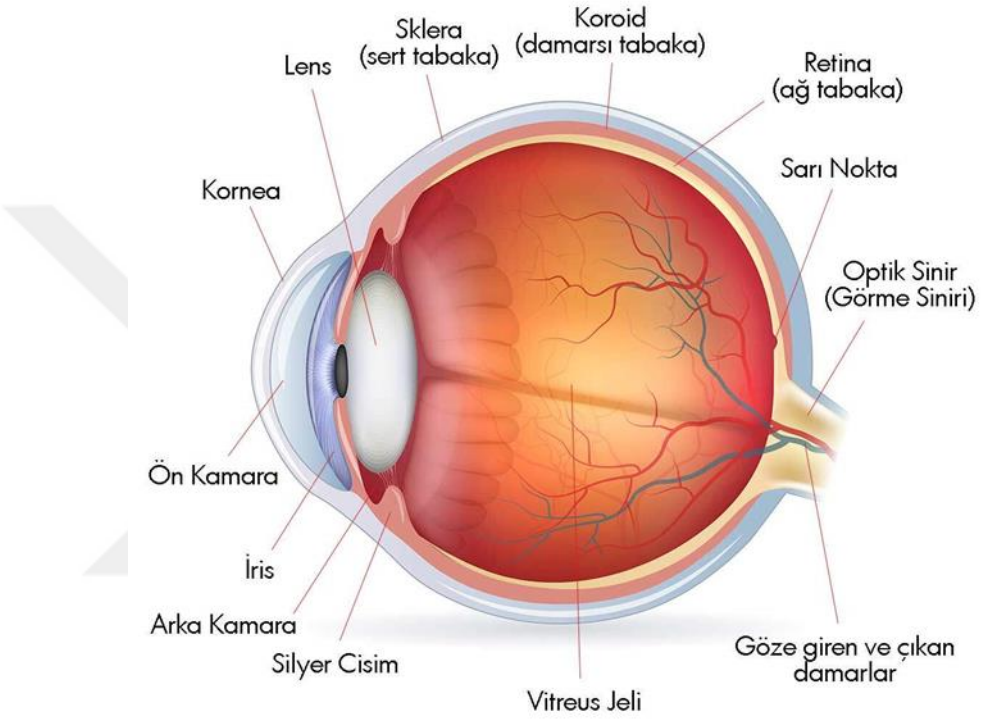
2.1. OFTALMOLOJİ

Oftalmoloji bilimi tıbbın alt dallarından biri olup göz ve görüş ile ilgili olan tüm rahatsızlıkları konusu edinmektedir. Yunanca kökenli bir kelime olan Oftalmoji'nin uzmanlarına Oftalmolog denmekte olup aynı zamanda aldıkları medikal eğitim ve cerrahi operasyon yapabilmeleri sebebi ile cerrah olarak kabul edilmektedirler.

Oftalmoloji biliminin konusu olan göz hastalıkları, genetik ve travmatik gerçekleşen olayların haricinde, yaşam döngüsünün engellenemez bir parçası olan yaşlanma sürecinden en fazla etkilenen ve buna tepki gösteren fiziksel birimlerin başında gelmektedir.

Görme organı olan göz, Orbita adı verilen çukurumsu yapısı sayesinde gözde meydana gelebilecek tümör gibi kitlesel hastalıkların, gözden içeri girmesini bir süre olsa da engelleyebilecek bir kemik yapının içinde bulunmaktadır. Normal olarak kabul edilen bir bireyde güresi 24.2 mm'dir. Gözün en önde bulunan saydam kısmına Kornea denilmektedir. Kornea ile birbirlerini tamamlayacak şekilde bulunan beyaz kısma Sklera adı verilmektedir. Sklera sert yapısı ve içinde barındırdığı lifler sayesinde gözün koordineli bir şekilde hareket etmesini sağlamaktadır. Sklera'nın iki göz için koordineli hareket edememesi durumunda Şaşılık denilen göz kusuru ortaya çıkmaktadır. Sklera'dan sonra Uvea adı verilen damar tabaka olarak da bilinen yapı bulunmaktadır. Görevi gözü beslemektir. Kornea tarafından olacak şekilde sırasıyla iris, silyer cisim ve koroid denilen üç bölümden oluşmaktadır. Uvea'nın arkasında Lens denilen kısım bulunmaktadır. Lens, renksiz ve esnek bir yapıya sahiptir. Gözde odaklanma işlemini gerçekleştirmektedir. Bireylerin yaşlanması ile lensin esnekliğini kaybedip sertleşmesi ve ağırlığının artması sonucunda yakını görememe sorununa yol açmaktadır. Retina en arka kısmında yer alan ve gözün görme işlemini gerçekleştiren yapıdır. Ağ tabaka olarak da bilinmektedir. Işığı elektrik enerjisine dönüştürerek beyine anlamlı görüntüler gönderilmesini sağlamaktadır. Retina 'da ters olarak oluşturulan görüntü beyinde düzeltilerek anlamlandırılır bir hale gelmektedir. Retina içten dışa birçok yapıdan meydana gelmektedir. Makula ya da sarı nokta adı verilen bölge retinanın merkezi olmakta ve hedefe yönelik görme işlemini gerçekleştirmektedir. Makula bölgesi asıl görülmek istenen nesneye bakıldığında makula görev yapmaktadır. Hedefin arka planında yer alan nesneler ise ayrıntı olarak silik bir

şekilde görülmektedir. Bu nokta da retinanın diğer bir bölgesi olan periferik bölge görev almaktadır. Yaşa bağlı hastalıklılar daha çok retinanın makula bölgesinde meydana gelmektedir. Vitreus lens ile retina arasında kalan kısımdır. Retinaya yapışık bir şekilde bulunmaktadır. Kolajenli yapısı sayesinde gözü beslemekte ve şeffaf bir formda bulunmaktadır. Gözün anatomik yapısı aşağıdaki görselde gösterilmiştir.



Şekil 2.1: Gözün yapısı **[4]**.

Yaşlanmanın, gözde en çok etkilendiği kısım retinanın merkezi olan makula bölgesidir. Makulada gerçekleşen rahatsızlıklar, bireylerde odak problemleri yaşamalarına sebep olmaktadır. Yaşa bağlı makula dejenerasyonları (YBMD), genellikle kadınlarda erkeklerden daha çok rastlanılan ve 65 yaş üstü bireylerde sıkça görülmektedir. Bu çalışmada kullanılan verilerin ait olduğu Drusen, koroidal neovaskülarizasyon (KN) ve diyabetik makula ödemi gibi hastalıklar, yaşa bağlı makulada ya da diyabetik retinopati sırasında ortaya çıkan hastalıklardan bazılarıdır.

"Drusen, retina pigment epiteli bazal membranı ile Bruch membranı iç kolajen zonu arası ekstraselüler materyal birikimidir. Klinik olarak retinanın derinliklerinde sarı beyaz birikintiler şeklinde görülür" [5] .Drusen genellikle genç yaşlarda oluşan ama semptom göstermediği için

ileri yaşlarda ancak tanı konulabilen bir göz hastalığıdır. Koroidal neovaskülarizasyon (KN), "Bruch membranına girip Bruch membranının kalınlaşmış iç yüzeyi ile diğer tabakalı arasında ve/veya retina altı mesafede proliferasyonu neticesinde ortaya çıkar" [5].KN de Drusen gibi semptom göstermeyen bir göz hastalığıdır.

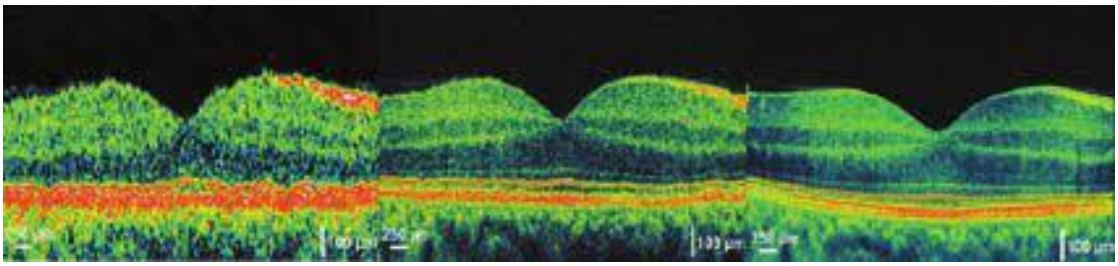
Diyabetik makula ödemi (DME), "diyabet hastalarında kan şekerinin yüksek olmasından kaynaklanan her tür retina (görmeye yarayan sinir hücrelerinin bulunduğu tabaka) hasarını kapsayan, diyabetik retinopatinin seyri sırasında herhangi bir zamanda, zayıflayan damarlardan sızan kan ve sıvının makula adı verilen görme merkezinde birikmesidir" [6].

2.1.1. Optik Koherens Tomografi

Optik koherens tomografi (OKT), başta oftalmoloji olmak üzere biyolojik alanda mikro yapıdaki dokuların incelenmesinde kullanılan bir yöntemdir. "OKT cihazıyla retina ve ön segmentin gerçek zamanlı, non-kontakt kesit görüntülerinin elde edilmesinden dolayı, bu teknoloji oftalmolojide etkin olarak kullanılmaktadır. OKT, retinada optik disk ve makula gibi anatomik yerlerin görüntülenmesinin yanında; retina sinir lifi, fotoreseptörler ve retina pigment epiteli gibi intaretinal yapıların incelenmesini de sağlar. Ayrıca OKT görüntülemesi ile retinanın morfometrik veya kantitatif ölçümleri elde edildiğinden, hastalıkların tanı ve takibinde önemli bir tanı yöntemidir" [7] .

OKT gönderilen ışığın yansıma zamanına göre ölçüm yapmaktadır. Daha uzakta olan yapılar geç yansırken daha yakında olan yapılar erken yansımaktadır. OKT, işte bu gönderdiği ışığın gecikme zamanına göre çalışmaktadır. Elde edilen OKT verileri üç boyutlu olacak şekildedir.

Temel çalışma prensibi incelendiğinde, OKT'nin radar sistemleriyle benzerlik gösterdiği görülmektedir.



Şekil 2.2: Örnek OKT görüntüsü **[8]**.

Yukarıdaki görselde, normal bir bireyin farklı çözünürlüklerdeki Stratus OKT görüntüsü verilmektedir.

2.1.2. Oftalmolojide Yapay Zekâ ve Derin Öğrenme Üzerine Yapılmış Çalışmalar

Tıbbın diğer tüm birimlerinde olduğu gibi oftalmolojide erken teşhis büyük önem taşımaktadır. İlerleyen yazılım ve donanımdaki imkanlar ve ihtiyaçlar neticesinde, yapay zekanın tıp alanında kullanımı artmıştır. Yapay zekanın dolayısıyla derin öğrenmenin, birçok tıbbi alanda geniş bir uygulama alanı olsa da oftalmolojide korneal ekstazi, glokom, yaşa bağlı makula dejenerasyonu (YBMD) ve diyabetik retinopati gibi görüntü üzerinden tanı ve teşhisin esas olduğu göz hastalıklarında büyük bir etkisi vardır.

Oftalmolojide görevli oftalmologlar hastalığın teşhisini gözün ve gözü çevreleyen yapıların doğrudan ya da dolaylı olarak yapılan görselleştirmeler aracılığıyla ortaya çıkan nesne örüntülerine dayanarak yaparlar. Parampal S. Grewal ve arkadaşları yaptıkları makalede görüntülemeye dayalı olan bağımlılığın, oftalmolojinin derin öğrenme algoritmalara için ne kadar uygun olduğunu ve oftalmologların çalışma yöntemleri ve performansları için büyük bir potansiyel değişim belirttiğini söylemiştir [9].

2015 yılında Xiangyu Chen ve arkadaşlarının yaptığı evrişimli sinir ağları (ESA) yardımı ile glokomun tespitini konu alan çalışmada, dört evrişim ve iki tam bağlı katmanla oluşturdukları ağı, veri çeşitlendirme ve gürültü ekleyerek performansını arttırmış ve sonuçta iki ayrı veri seti (ORIGA ve SCES) üzerinde çalıştırılarak sırasıyla 0,831ve 0,887 başarım elde etmiştir [10].

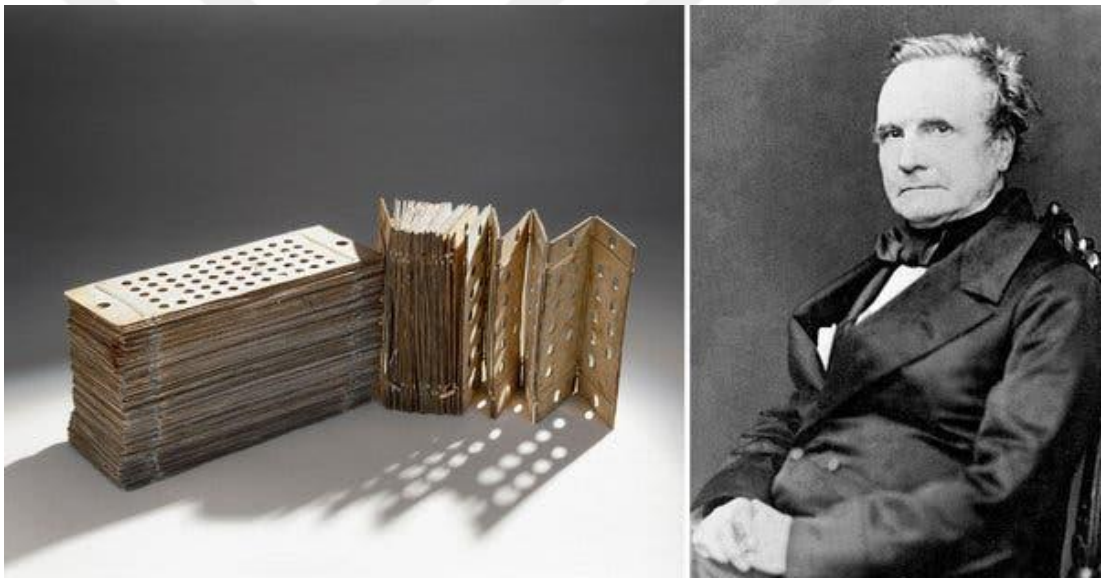
Cecilia S. Lee ve arkadaşlarının yaptığı 1289 OKT görüntüsü kullanarak yaptıkları derin öğrenme tabanlı makula ödemi segmentasyonu çalışmasında, kullanılan ESA modeli ve görevli dört uzman kullanılarak yapılan çalışmada, modelin çapraz doğrulama dice katsayısı 0,911 bulunmuştur. Ayrıca uzmanların birbiri arasında ve uzman-ESA karşılaştırılmalarında dice katsayıları birbirine çok benzer çıkmıştır. Bu çalışma, ESA modelinin hastalık tanısında nerdeyse görevli bir uzmanla aynı performansı gösterdiğini kanıtlamıştır [11].

2.2. YAPAY ZEKA

Düşünebilen makineler yapmak bilimin erken tarihinden beri bilim insanlarının hayalleri arasında yer almaktadır. 1800'lü yıllarda İngiliz matematikçi ve makine mühendisi Charles BOBBGE, o dönemde kullanılan logaritma cetvellerinde bazı hatalar buldu. Bu hatalı cetvellerin, mühendislik hesaplamalarında ve o dönemde büyük önem taşıyan askeri ve sivil alanda gemi rotaları hazırlarken sorun çıkaracağını belirtti. Bu hesaplamaların otomatik bir makineyle daha hızlı ve daha gelişmiş bir şekilde yapılabileceği fikrini ilk 1812'de ortaya attı

ve 1823'te projeye başladı. "Fark makinesi adını verdiği makine, polinom fonksiyonlarının değerlerini verilen ilk değere göre hesaplayacaktı. Proje o zamanın şartlarının hassas parçalar üretecek düzeyde olmaması ve yüksek bir bütçeye sahip olmasından dolayı yarıda kaldı. Daha sonraları Babbage, Fark Makinesi-2 adını verdiği bir makine daha tasarlasa da o da aynı sebeplerden dolayı yarıda kaldı.

Babbage Fark Makinesini tasarlarken "Analitik Makine" adını verdiği başka bir makine daha tasarlamıştı. Analitik Makineyi Fark Makinesinden ayıran en önemli özellikler daha geniş amaçlı olması ve delikli kartlarla programlanabilir oluşuydu. Makine ve kullanıcı arasındaki bilgi alışverişi delikli kartlarla sağlanacak şekildeydi. Yapısı itibariyle ilk genel maksatlı bilgisayar olarak tanımlanan Analitik Makine dönemin maddi ve teknolojik imkansızlıkları nedeniyle tamamlanamamıştır.



Şekil 2.3: Charles BABBAGE ve onun tasarladığı bilgisayarın programlanacağı delikli kartlar **[12]**.

İlk genel maksatlı bilgisayar olarak bilinen "Analitik Makine" üzerine yaptığı çalışmalarla bilinen İngiliz matematikçi ve tarihin ilk programcısı olarak bilinen Ada LOVELACE 1843'te yaptığı bir konuşmada bu makinenin bir şey yaratmak için yapılmadığını halihazırda yapılabilen şeyleri yapmasa insanlara yardım etmek için geliştirildiğini söylemiştir. Lovelace'a göre bu tür bir makine uygun şekilde programlanırsa karmaşık müzik eserleri bestelemek, grafik üretmek ve karmaşık matematiksel problemleri çözmek vb. için kullanılabilirdi. Lovelace, Babbage'a gönderdiği mektuplarda söz konusu makinenin belli ve sonlu sayıda adımdan oluşan bir plan kullanarak ne şekilde Bernoulli sayılarını hesaplayabileceğini anlattı.

Bu anlatım bilgisayar tarihinde somut bir makineye uygulanabilecek olan ilk “bilgisayar programı” olarak bilinmektedir [13].

Şekil 2.4: Tarihteki ilk bilgisayar programcısı Ada LOVELACE **[14]**.

Yapay zekâ ortaya çıktığı ilk zamanlar kısa vadede o zamanki görevlerini yerine getirmiştir. Bunlar insan beyninin ve dönemin hesaplama makinelerinin yavaş kaldığı karmaşık matematiksel hesaplamalardır. Ama gelişen teknoloji ve dönemin uçan arabalar veya konuşan hizmetçi robotlar gibi ütopik hedefler ile yükselişe geçen Yapay zekada, asıl sorunun matematiksel hesaplamalar değil, sağlıklı bir insan için çok kolay olan konuşma tanıma veya nesne bulma gibi daha sezgisel konular olduğu ortaya çıkmıştır. Zaman ile bu sorunu üzerine gidilmiş hesap makinesinden insan beynine yakınlaşma çabası başlamıştır. İlk olarak belli başlı kuralları olan ve belirli bir düzende olan problemler çözülmeye başlanmıştır. Seksenli yıllarda Carnegie Mellon Üniversitesinde geliştirilen Deep Tought adlı satranç makinesi yasal bir turnuvada satranç ustasını yenerek bunu yapan ilk makine olmuştur. Daha sonra Deep Taought ekibi IBM Araştırma merkezine geçerek Deep Taught 2’yi diğer adıyla Deep Blue’yu geliştirmişlerdir. Deep Blue 1997 yılında satranç dünya şampiyonu Gary Gasparov’u altı maçlık bir oyunda 3,5 ‘e 2,5 skor ile yenmiştir [3].

1800’lü yıllarda başlayan düşünen makineler fikri ancak yakın bir zamanda sezgisel problemleri gerçekleştirecek düzeye gelmiştir. Bunun sebebi gündelik hayatta karşılaşılan rutinlerin sonsuz sayıda versiyonlarının olması ve programcının bu gösterimleri makineye girdi olarak programlanması gerekmesiydi. Böylelikle teorik olarak makine mantıksal çıkarım kuralları kullanarak akıl yürütebilirdi. Özellikle 80’li yıllarda kullanılan Knowladge base olarak

bilinen (expert system olarak da bilinir) bu yaklaşım yetersiz kalmış ve beklenen başarıyı elde edememiştir. Bu yaklaşımın kullanıldığı projelere örnek olarak Lenat ve Guha'nın 1989 yılında Cyc Cyc veri bankasıyla oluşturduğu CyCL dilini kullanarak yazdığı çıkarım motoru örnek verilebilir [15].

1933 yılında "Hesaplanabilir Sayılar" adını verdiği makalesinde İngiliz matematikçi Alan Turing sadece matematiksel bir sistemden değil aynı zamanda gerekli kurallar ortaya koyulduğunda insanlar yerine düşünebilecek ve gerekli hesapları yapabilecek sanal makinelerden bahsetti. Makalede adı geçen Hesaplayıcı makinenin Türkçe 'de karşılığı "bilgi sayar" olup bir bant üzerine yazılmış sembolleri okuyup daha önce girilmiş sembollerle karşılaştırıp buna göre bir sonuç verip aynı şekilde bandın üzerine yazacak şekilde tasarlanmıştır. Mantıksal dönüşümler yapması insan zekasına benzerliğini göstermektedir. 1950 yılında "Bilgisayar Mekanizması ve Zekâ" adındaki makalesini yayınlayan Alan Turing, Turing testi olarak kabul edilen bir makinenin zeki olup olmadığını belirleyecek olan bir deney fikri ortaya atmıştır. Buna göre, makine, insan olan soru soran kişiyi, bu kişi ile yapmış olduğu konuşmadan sonra insan olduğuna inandırabiliyorsa, bu makinenin (bilgisayarın) insan kadar zeki olduğu kabul edilecekti.

1956 yılında New Hampshire eyaletinde bulunan Dartmouth kolejinde gerçekleşen 8 haftalık çalıştayda YZ araştırmaları tartışılmış bilgisayarlar, doğal dil işleme, sinir ağları, hesaplama teorisi, soyutlama ve yaratıcılık gibi konular işlenmiştir. Yapay zekâ terimi ilk kez burada ortaya atılmıştır. Konferans yapılması için çıkan önergeden bir kesit şöyledir: "1956 yazında Hanover, New Hampshire'daki Dartmouth Koleji'nde 2 aylık, 10 kişilik bir yapay zekâ çalışmasının yapılmasını öneriyoruz. Çalışma, öğrenmenin her yönünün veya zekanın diğer herhangi bir özelliğinin, prensipte makinenin onu simüle edebileceği kadar kesin bir şekilde tanımlanabileceği varsayımı temelinde ilerleyecektir. Makinelerin dili nasıl kullanacaklarını, soyutlamaları ve kavramları nasıl oluşturacaklarını, insanlar için ayrılmış problem türlerini nasıl çözeceklerini ve kendilerini nasıl geliştireceklerini bulmaya çalışılacaktır. Bir yaz boyunca dikkatle seçilmiş bir grup bilim insanı üzerinde birlikte çalışırsa bu sorunlardan bir veya daha fazlasında önemli bir ilerleme sağlanabileceğini düşünüyoruz." [16].

Stuart Russel ve Peter Norvig tarafından kaleme alınan "Yapay Zeka: Modern Bir Yaklaşım" adlı eserlerinde YZ için sekiz yaklaşım iki boyutta ele alınmıştır [17]. Bu yaklaşımlar şöyledir

- İnsanca davranış: Örnek olarak alan Turing tarafından ortaya atılan Turing testini verilebilmektedir. Bu yaklaşıma uygun olabilmek için makineden doğal dil işleme (NLP) , bilgi gösterimi (knowledge representetion) , makine öğrenmesi (machine learning) , otomatik akıl yürütme (automated reasoning) bilgisayar görüsü (computer vision) ve robotik konularını bilmesi beklenmektedir. Belli olduğu üzere bu yaklaşım modern yapay zekanın ana disiplinlerini içermektedir. Makinelerin insanlar gibi davranmasına dayalı bir yaklaşımdır.
- İnsanca düşünme: Bilişsel modelleme yaklaşımı olarak da bilinir. Makinelerin insan gibi düşünebilme ve karar verme problem çözme yeteneğine sahip olmasına dayanmaktadır. Makineye verilen girişlere göre alınan çıkışların, aynı girişlerin insana verildiğinde elde edilen çıkışlarla korelasyonunun incelenmesine dayalıdır. Bu yaklaşıma örnek olarak Newell ve Simon'nun gerçekleştirdiği GPS (Genel Problem Solver)'i gösterilebilir. Yapay Zekanın insan bilişsel sistemiyle olan benzerliği ve bu yaklaşım üzerine kurulu olması iki konunun da birbiri ile sürekli iç içe olduğunu ortaya koymaktadır.
- Rasyonelce düşünme: Bilgisayar tabanlı hesaplama modellerin kullanılması ile insanlara ait zihinsel özelliklerin makinelere aktarılması yaklaşımıdır. Mantık üzerine dayalıdır. Mantıksal tabana dayalı olan tüm sorunların yine mantıksal örüntüler ile çözülebileceğini eğer mantıksal bir çözümü yoksa programın sonsuz bir döngüde devam edeceğini söylemektedir. Düşünce yasaları yaklaşımı olarak da bilinir.
- Rasyonelce hareket: Rasyonelce davranmak, kişinin inançları veya dünya hakkındaki anlayışı göz önüne alındığında, kişinin hedeflerine ulaşmak için hareket etmesi anlamına gelir. Ajan, bir ortamı algılayan ve o ortam içinde hareket eden bir sistemdir. Akıllı bir ajan, amaçlarına göre rasyonel olarak mantıksal hareket eden kişidir. Örneğin, oyun oynamak için tasarlanmış bir ajan, oyunu kazanma şansını artıran hamleler yapmalıdır. Akıllı bir ajan yani sistem oluştururken, ajan teorik olarak mümkün olan en iyi karar verme modeli tasarlamaktan çok, ajanın hareket ettiği şartlar içinde mümkün olan en iyi kararı modeli tasarlamaya doğru kayar. Ancak teorik olarak mümkün olan en iyi olan kararı verip "mükemmel rasyonalite" denilen şeyi elde etmek, gerçek bir ortamda genellikle mümkün değildir.

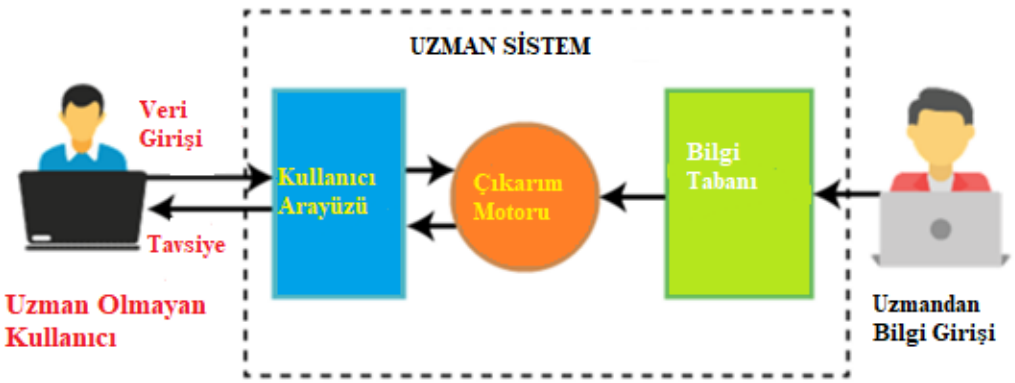
Yukarıda açıklanan tarihsel gelişmeler ve yaklaşımlar doğrultusunda yapay zekayı insan beyninin kısmen yavaş kaldığı matematiksel işlemler dışında insanlara özgü olan, olayları yorumlama, deneyimlerden yararlanma, genelleme yapabilme ve öğrenme gibi akılcı davranışların makinelere kazandırılması olarak açıklanabilir. Yapay zekanın amacı insan zekasının gerektiği durumlar için kendi zekâları olan ve daha hızlı sonuca varabilen makineler

üretmektir. Yani yapay zekâ insan beyninin bir modellemesidir. Yapay zekâ modellenirken kullanıldığı tekniklere göre ayrılır. Bunların bazıları aşağıda gösterilmiştir.

2.2.1. Uzman Sistemler

Uzman sistemler, konularında uzman olarak nitelendirilebilecek kişilerin bilgilerine sahip makinelerdir. Bilgi tabanlı olan bu sistemler kullanıcılarına karar verme ve sonuca ulaşmada konunun uzmanı olarak insan desteği olmadan yardımda bulunurlar. Uzman sistemler üç bileşenden oluşmaktadır. Bunlar kullanıcı ara yüzü, çıkarım motoru ve bilgi tabanıdır. Kullanıcı ara yüzü, uzman olmayan kullanıcı ile sistem arasındaki ilişkiyi kuran kullanıcıdan aldığı soruları uygun formatta çıkarım motoruna iletir. Daha sonra çıkarım motorundan aldığı sonuçları da kullanıcıya çıkış olarak iletmektedir. Çıkarım motoru sistemin beyni olarak çalışır ileri ve geri zincirleme yöntemleri ile çalışır. Bilgi tabanı depo görevi görmektedir. Aynı konuda başka uzmanlardan aldığı bilgileri burada tutar. Uzman sistemler, özellikle çeşitli alt uzmanlık dalları olan tıpta ve eğitimde kullanılmaktadır.

Abdülkerim Öncü'nün yaptığı "Uzman Sistem Yaklaşımı ile Web Tabanlı Öğretim Değerlendirme Sisteminin Geliştirilmesi" adlı doktora çalışmasında ÖDM (Öğrenim Değerlendirme Sistemi) adında 7 modülden oluşan bir sistem tasarlanmıştır. Bu modüller sırasıyla öğretim modülü, sınav, psikoloji, çoklu kriter, veri toplama, analiz ve raporlama modülleridir. Analiz modülünde Uzman Sistemler kullanılmıştır. Çalışma, öğretimin hem öğrenci hem de kullanıcı olarak geçen öğretmenler tarafından hiçbir yazılım bilgisini ihtiyaç duymadan kullanabilir olması ile eğitimde verimliliği arttırılmasını hedeflemiştir [18] .



Şekil 2.5: Uzman sistemlerin çalışma döngüsü **[19]**.

2.2.2. Bilgisayar Görüsü

Bilgisayar görüsü, fotoğraf, resim ve video gibi dijital verilerden anlamlı bilgiler çıkarmaya yarayan yapay zekâ konusudur. Nesne tanıma, nesnelerin arasındaki mesafeyi anlama ya da nesneleri birbirinden ayırma gibi, insan görüsünün yapabildiği her şeyi kendisine konu eden bilgisayar görüsü ne kadar çok veriyle eğitilirse o kadar iyi çalışmaktadır. Bunun için çok fazla veriye ihtiyaç duyan bilgisayar görü, yeterli miktarda veriyle eğitildiğinde insan kapasitesinin çok üstünde bir farkındalık ve hızda performans göstermektedir. Örneğin Python'da Keras kütüphanesinde bulunan kedi ve köpeklerden oluşan veri seti 3 milyondan fazla görüntü verisinden oluşmaktadır. Derin öğrenme, katmanlar arası hiyerarşik çıkarımlar yapabilme ve çok fazla veriyle tekrar tekrar eğitilebilme imkanına sahip olduğu için bilgisayarlı görünün en çok kullanıldığı yöntemidir. Bu konuda en çok kullanılan derin öğrenme algoritması evrişimli sinir ağları (ESA)'dır. Evrişim ve biriktirme çiftlerinden oluşan ESA'lar görüntüyü piksel piksel işlemektedir. İnternet sayesinde neredeyse her gün ortaya çıkan milyarlarca dijital görüntü ve bu görüntülerin işlenebileceği donanımsal gelişmelerin ilerlemesinden dolayı bilgisayarlı görü yapay zekanın en popüler alanlarından biri haline gelmiş ve oluşturulan modellerin başarı yüzdeleri 100'e yaklaşmıştır.

2.2.3. Konuşma Tanıma

Konuşma tanıma bir makinenin, insan sesini yazılı bir formata dönüştürme yeteneğidir. Fiziksel (el ile) bir müdahale gerektirmeden çalışma özelliği itibari ile fiziksel engelli bireylerin yaşamlarında önemli rol oynamaktadır. Ses tanımanın aksine sadece bir kişinin sesine odaklanmaz tüm kullanıcılardan aldığı verileri işlemektedir. Başlangıçta ses dalgası şeklinde olan konuşmaları donanım itibari ile kendi işleyebileceği formata elektriksel sinyale dönüştürür. Daha sonra kullanıcıdan aldığı komutlara göre çıktı vermektedir. Yapay zeka konusu itibari ile burada devreye girmektedir. Zamanla kullanıcıdan aldığı komutları öğrenip konuşmanın kime ait olduğunu etiketleyebilir, istenilen verileri genelleyerek benzer sonuçlara çıktı verebilir, yanlış veya eksik verilen giriş verisini tamamlayabilir ya da zararlı gördüğü sonuçları filtreleyebilmektedir. Günümüzde araçlarda sesle komut alabilen sistemlerde, kullanıcının konuşmasını yazılı formata dönüştürüp zamandan tasarruf etmesini sağlayan sanal not defterlerinde ya da satın alınan ürün ile sorun yaşayan müşteri ile konuşup müşteri hizmetlerine gerek kalmadan problemi çözen sistemlerde konuşma tanıma tabanlı yapay zeka sistemleri kullanılmaktadır. Amazon'un geliştirdiği 2014 yılında piyasaya sürdüğü Alexa ve Apple'ın 2011 yılında kullanıcılarına sunduğu Siri konuşma tanıma kullanan en popüler yapay

zekâ sistemleridir. En çok kullanılan konuşma tanıma algoritmaları Doğal Dil İşleme, Dinamik Zaman Bükülmesi ve Nöron Ağlar'dır.

Dinamik zaman bükülmesi, zamana bağlı olan veriler ile önceden bildiği kelimeler arasındaki benzerlikleri kıyaslayıp belirli kurallar içinde en iyi sonucu bulan yöntemdir. İnsan beynindeki nöron ağ yapısına atıfta bulunarak geliştirilen nöron ağlar özellikle derin öğrenmekte kullanılıp olup, makinelerin ellerindeki bilgiden yeni bilgiler türetmesi gibi davranışlar göstermesinde rol oynar. Doğal dil işleme çok kullanılan bir yapay zekâ yöntemidir. İnsanlar tarafından kullanılan doğal dili köklerinden eklerine göre anlayıp, ayırıp sonuçlar türetebilen bunun sonucunda öğrenme işlemini gerçekleştirmektedir.

2.3. MAKİNE ÖĞRENMESİ

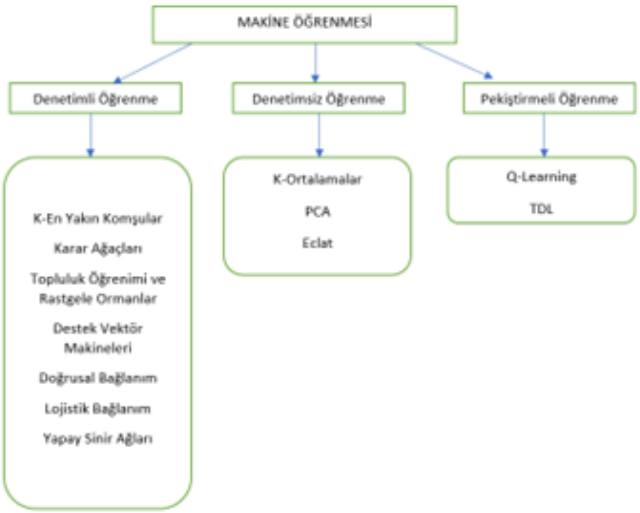
Makine öğrenmesi, kullanıcılar tarafından yazılmış programları çözmekte yetersiz kalacağı görevleri yerine getirmeyi olanak sağlar. Yani bunu kullanıcıdan almayı beklemeden ya da programda açık açık kodlanmadan, kendi kendine öğrenme ile yapabilmesine makine öğrenmesi denir. Öğrenme ile görev aynı şey değildir. Öğrenme, görevi başarıyla gerçekleştirmeyi sağlayacak kadar beceriyi sistemin kazanmasıdır.

Makine öğrenimi programlamada yeni bir çığır açmıştır. Klasik programlama mantığında sisteme kurallar önceden tanıtılıp veriler girdi olarak verildiğinde bu girdilere uygun sonuçlar üretirken, makine öğrenmesinde girdiler ve bu girdilere ait beklenen çıktıları sisteme tanıtıp farklı girdiler için uygun sonuçları üretmesine sağlayan kurallar öğrenebilmesi gerekir. Bu kuralları da istatiksel tabanlı olarak bulmaktadır. Yıllar içinde gelişen donanım ve artan veri sayıları ile klasik makine öğrenmesi algoritmaları yetersiz kalmıştır. Bunun sonucunda da daha az matematik ve istatistik tabanlı olup daha çok programlamaya dayalı olan derin öğrenmeye talep artmıştır.

Şekil 2.6: Klasik programlama ile makine öğrenmesi kıyaslaması.

2.3.1. Makine Öğrenmesinin Türleri

Derin Öğrenmeye geçmeden önce makine öğrenmesini detaylıca anlatmakta fayda vardır. Çok sayıda makine öğrenmesi sistemi bulunmaktadır. Bunları girdi olarak verilen bilginin etiketli veya etiketsiz olup olmadığına göre ya da ne kadarının etiketli olduğuna göre, giren verinin tamamının mı yoksa aşamalı olarak kısım kısım mı sisteme tanıtılmasına göre ya da örnek tabanlı ve model tabanlı olarak ayrılabilir. Bu kriterler birbirinden tamamen ayrı olarak değil hibrit bir şekilde de kullanılabilir.

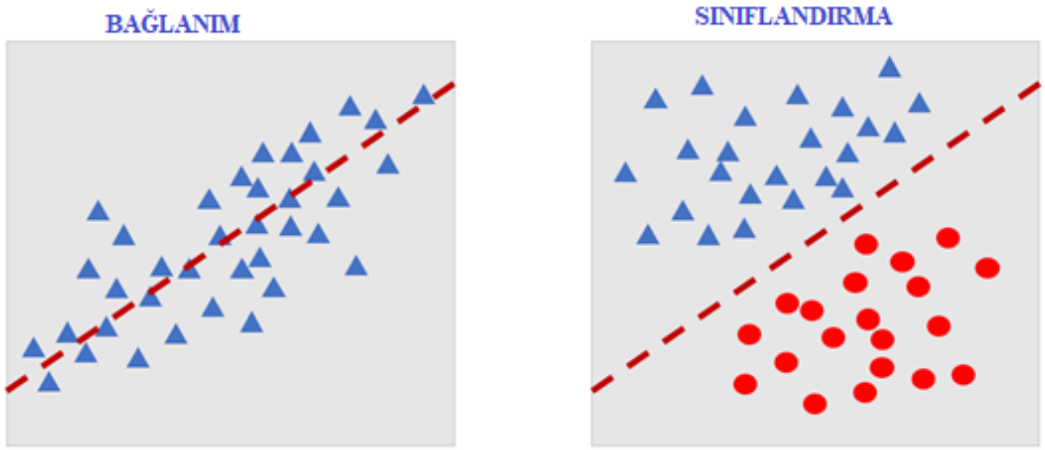


Şekil 2.7: Tez çalışmasında açıklanan algoritmalar.

2.3.1.1. Denetimli Öğrenme

Modeli besleyen verilerin etiketli olmasına yani beklenen sonuçları içermesi durumuna denetimli öğrenme denir. En basit denetimli öğrenme görevleri sınıflandırma ve bağlanımdır. Sınıflandırma görevi eğitilen modele yeni verilen girildiğinde hangi sınıfa ait olduğunu

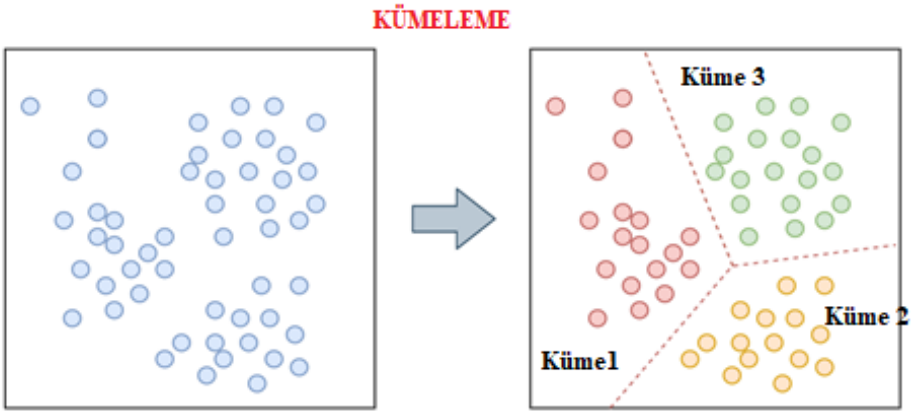
belirtmektedir. En basit örneği mail kutusunun postaları gerekli ve gereksiz olarak ayırabilmesidir. Diğer görev bağlanım ise sayısal bir çıktı vermektedir. Verilen girdilere ve etiketlere bakarak, modelden istenilen veri için sayısal bir tahminde bulunmaktadır. Örneğin evlerin özniteliklerinin çeşitli olduğu (vergi miktarı, bulunduğu konumun suç oranı, konutun yaşı vb.) ve etiket değerinin kirayı belirlediği bir veri setiyle eğitilen bir modelin, belirli özniteliklerdeki bir evin kirasını tahmin edebilmesi bir bağlanım görevidir. Bağlanım problemi sınıflandırma için de kullanılabilmektedir. İstenilen verinin hangi sınıfa ait olduğunun yüzde değerini verebilmesine lojistik bağlanım denir. En bilinen denetimli öğrenme algoritmaları; K-En yakın komşular, Doğrusal bağlanım, Lojistik bağlanım, Destek vektör makineleri (SVM), Karar ağaçları ve Rastgele ormanlar, YSA ve derin öğrenme konuları bölüm 2.3.2' de detaylı olarak anlatılmıştır.



Şekil 2.8: Bağlanım ve sınıflandırma yönteminin çalışma şekilleri **[20]**.

2.3.1.2. Denetimsiz Öğrenme

Denetimsiz öğrenmede modelse eğitilen veriler etiketsizdir. Model kendi kendine öğrenmeye, veriler arası benzerlikler bulmaya çalışır. Örneklerle açıklamak gerekirse, Linkedn profilinize bakan ziyaretçilerden oluşan bir veri seti olduğu varsayılsın. Model, benzer ziyaretçileri gruplayarak varsayımlar üretebilir. Bu varsayımlar, ziyaretçilerin %50'sinin 5-10 yıl arası çalışmış kadın yazılımcıların olduğu ve sabahları profilinize baktığı ya da %30'unun çalışan ihtiyacı olan şirketler olduğu ve çalışma saatleri içinde profilinize baktığı olabilir. Bunu denetimsiz öğrenmede kümeleme denir. Elde edilen kümeleriz kendi içlerinde daha küçük kümelere ayrılmasına da hiyerarşik kümeleme algoritması denir.



Şekil 2.9: Kümeleme algoritması çalışma prensibi **[21]**.

Bir diğer denetimsiz öğrenme görevi ise boyut azalmadır. Boyut azaltmanın görevi veri kaybı yaşamadan, eldeki bilgiyi en basit şekilde gösterebilmektir. Bunu birbiriyle ilişkisi olan birkaç niteliği bir araya getirerek yapılabilir. Örnek olarak, bir çalışan memurun yaşı ile memurluk kıdemi arasındaki benzerlik yüksek olabilir. Boyut azaltma algoritması bu iki özelliği tek bir özellikmiş gibi kabul edip öyle davranabilmektedir. Buna makine öğrenmesinde öznitelik çıkarımı denir.

Denetimsiz öğrenme algoritmalarının temel çalışma prensibi veriyi kullanışlı bir halde temsil etmeye dayalıdır. Bu yüzden görselleştirme çok önemlidir. Çok sayıda heterojen verinin oluşturduğu veri setleri ile beslenen modellerde görselleştirme algoritmaları her bir alt kümeyi belirtebilmek için çok boyutlu gösterimler çizerler. Örneğin iki boyutta üst üste gelen iki farklı kümenin gösteriminde üçüncü boyut kümelerin birbirlerinden ayrı gösterimini sağlamaktadır.

Bir diğer denetimsiz öğrenme algoritması anomali tespitidir. Normal durumlarca eğitilmiş modelin başka bir örnek gördüğünde bunun normal durumlara benzeyip benzememesine göre anomali tespiti yapabilmektedir. Örnek olarak kredi kartından her zamankinden fazla yapılan alışverişte sistemin bunu algılayıp olağandışı olarak kabul edip kart sahibini uyarması gösterilebilir.

Literatürlerde en çok kullanılan denetimsiz öğrenme algoritmalarında K ortalamalar, PCA ve diğerleri aşağıda gösterilmiştir.

K-Ortalamalar (K-Means) yöntemi bir kümeleme algoritması olan denetimsiz bir makine öğrenmesi türüdür. Etiketsiz yani çıktısını bilmediği verileri, kendi aralarında benzetimler yaparak çıktı verebilen bir kümeleme algoritmasıdır. K belirlenen küme sayısını temsil

etmektedir. Veri kümesi içerisinde rastgele K kadar merkez noktası belirlenerek verilerin bu K merkez noktalarına uzaklıkları ölçülür. En kısa mesafe hangi kümeyle ise veri o kümeye ait olmaktadır. Merkez noktaları rastgele seçildiği için bu işlem birkaç kez tekrarlanmalıdır ki optimal bir değere ulaşılabilsin. Optimal kümeleri bulmaktaki amaç her kümenin elemanlarının birbirine maksimum oranda benzerlik göstermesi ve kümelerin kendi aralarında maksimum uzaklık göstermesidir. Bu uzaklık, verilerin etiketleri bilinmediği için çıktı verirken başarımı arttırmayı ve sınıflar arası benzerliği azaltmayı temsil etmektedir. K- ortalamalar yönteminde belirlenen K sayıdaki kümelerin elemanları kendi için minimum varyansa sahip olmalıdırlar.

K-Ortalamalar yönteminde veri K kadar alt kümeye ayrılırken ve optimal K değerini bulmaya çalışılması aşağıda gösterilen denklemde belirtilmiştir.

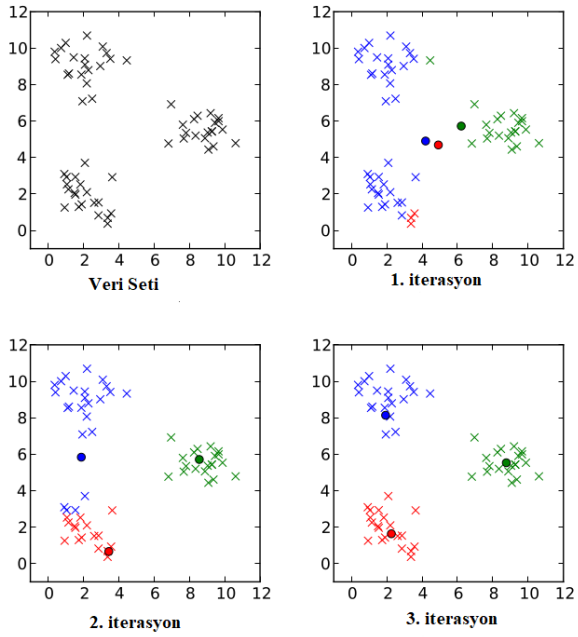
$$J = \sum_{i=1}^{m} \sum_{k=1}^{K} w_{ik} \, ||x^i - n_k||^2 \qquad (2.1)$$

$$w_{ik} = \begin{cases} 1, & x^i, \; k.\text{kümeye aitse} \\ 0, & \text{değilse} \end{cases} \qquad (2.2)$$

n_k = x^i'nin kümesinin merkezi

Kümeye ait olan bir elemanın kümenin merkezine olan uzaklığı hesaplanırken aşağıda gösterilen Öklid formülü kullanılabilmektedir.

$$d(p,q) = \sqrt{(p_1 - q_1)^2 + (p_2 - q_2)^2 + \cdots + (p_n - q_n)^2} = \sqrt{\sum_{i=1}^{n} (p_i - q_i)^2} \qquad (2.3)$$



Şekil 2.10: K-Ortalamalar algoritması çalışma prensibi **[22]**.

K- ortalamalar yönteminde uygun K küme değerini bulmak için iki farklı yöntem bulunmaktadır. Bunlar Dirsek Metodu (Elbow Method) ve Siluet Analizidir (Silhouette Analysis).

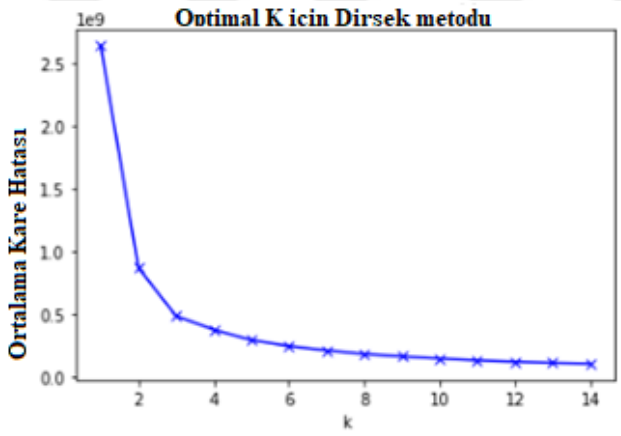
Dirsek metodunda etiketsiz verilerin her K küme sayısı için küme merkezine uzaklıklarının karesi toplamı hesaplanmaktadır. Hesaplanan uzaklıklara ve K küme sayısına göre grafik çizilir ve grafikte ani bir azalmanın görüldüğü değer optimal K değerini vermektedir

Siluet analiz yönteminde her K küme sayısına göre aşağıdaki denklem uygulanmaktadır. Bu denkleme göre kümelerdeki her elaman için oluşan Siluet katsayısı -1 ile 1 arasında bir değer almaktadır. 1 değeri o verinin hesaplandığı kümeye ait olduğunu göstermektedir. –1 değeri ise verinin yanlış bir küme ile gruplandığını göstermektedir.

$$Siluet\ Katsayısı\ (i) = {b(i) - a(i)}\Big/{max\{a(i), b(i)\}} \qquad (2.4)$$

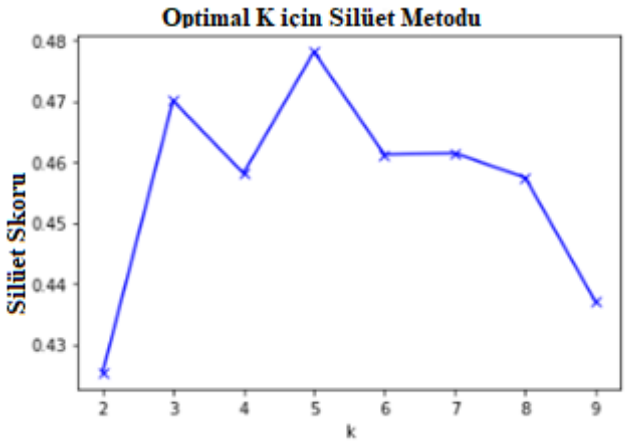
a(i)= aynı kümedeki diğer verilere olan ortalama uzaklık

b(i)=komşu kümelerdeki verilere olan ortalama uzaklık



Şekil 2.111: Dirsek yöntemi **[23]**.

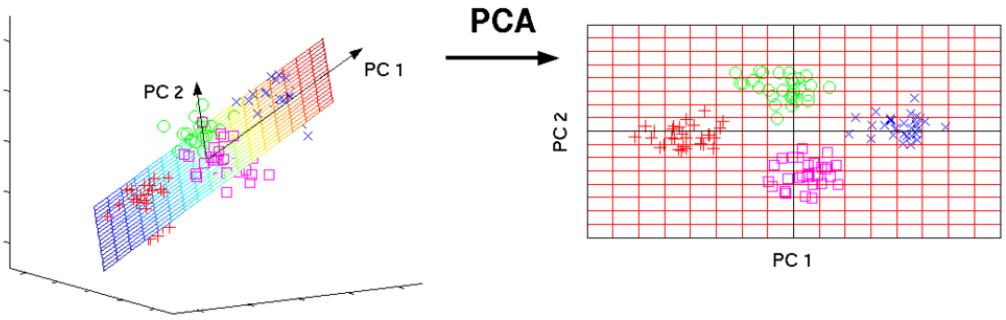
Yukarıdaki grafikte görüldüğü üzere dirsek metoduna göre bu veri setine en uygun kümeleme yöntemi için K değeri 3 seçilmelidir. Böylelikle optimal K değeri bulunmuş olur.



Şekil 2.12: Siluet Yöntemi **[24]**.

Yukarıdaki grafikte görüldüğü üzere Siluet metoduna göre bu veri setine en uygun kümeleme yöntemi için K değeri Siluet skorunun maksimum olduğu 5 seçilmelidir. Böylelikle optimal K değeri bu veri kümesi için bulunmuş olur.

Temel bileşenler analizi (Principal Component Analysis, PCA) yöntemi genelde çok boyuta sahip olan veri kümelerinde gürültüyü azaltma, boyut küçültme gibi nedenlerden dolayı kullanılan bir denetimsiz öğrenme algoritmasıdır. Temel bileşenler analizinde amaç çok boyutlu olan veri kümesini daha detay olarak adlandırılabilecek niteliklerden arındırmaktır. Başka bir şekilde ifade etmek istenirse amaç verinin boyutu daraltıldığında ana veriden bilgi kaçağı olmamasını sağlamaktır. Boyutu yani niteliği azaltılan veriden kayıp olmaması imkânsız olacağı için, bunu veride yer alan daha az önemli niteliklere uygulayabilmek esastır. Bir grafik üzerinden düşünüldüğünde ise çoklu özelliklere sahip karmaşık bir veriye farklı bir açıdan bakabilmek ve veriler arası daha anlamlandırılabilir ayrımlar çıkarabilmek PCA olarak adlandırılmaktadır. Aşağıdaki görselde de görüldüğü gibi çok sayıdaki karmaşık bir veri kümesine farklı noktalardan bakmak boyut azaltılmasına neden olmaktadır.



Şekil 2.23: PCA yönteminin gösterimi **[25]**.

Temel bileşenler analizi i boyuttaki ham veriden, en anlamlı özellikleri olan özünü kaybettirmeden i<k olacak şekilde k boyutta ifade edilmesini sağlamaktır. k boyuta indirilen verideki değişkenleri temel bileşen denir . İlk dönüştürülen temel bileşen varyansı en büyük olan bileşendir ve diğer dönüşen bileşenler varyansı azalan şekilde devam etmektedir. Varyans bir rastgele dağılımda, değişken ile dağılımın ortalamasının arasındaki farkın karesidir. Varyansı yüksek olmasının istenilmesindeki sebep etiketsiz olan verileri birbirinden ayırabilecek şekilde gösterilmesidir.

Eclat, birliktelik kural çıkarımı tabanlı bir denetimsiz makine öğrenmesi algoritmasıdır. Birliktelik kural çıkarımı, veriler arasındaki benzerlikleri çözerek beraber gerçekleşme olasılıklarını bulan bir yöntemdir. Bu yönteme verilecek en popüler örnek internet üzerinden alışveriş yapılmasını sağlayan online alışveriş sitelerinde, kullanıcıların daha önce aldıkları ya da inceledikleri ürünlere göre önerilerde bulunan sistemler verilebilmektedir. Diğer bir ifadeyle birliktelik kural çıkarımı yöntemi kullanıcıların davranışlarını inceleyip öğrenerek, bir sonraki davranışları hakkında kullanıcılara ya da yukarıdaki örnekte gösterildiği üzere satıcılara hangi ürünlerin beraber satıldığı hakkında önerilerde bulunmaktadır. Birliktelik kural çıkarımı yönteminde iki önemli nokta bulunmaktadır. Bunlar kullanıcılar tarafından başlangıçta belirlene Destek ve Güven faktörleridir.

Tablo 2.1: Apriori için verilerin düzenlenmesi

TID	**Öğeler**
1	Ekmek Tereyağı, Reçel
2	Tereyağı, Soda
3	Tereyağı, Süt
4	Ekmek, Tereyağı, Soda
5	Ekmek, Süt
6	Tereyağı, Süt
7	Ekmek, Süt
8	Ekmek, Tereyağı, Süt, Reçel
9	Ekmek, Tereyağı, Süt

Tablo 2.2: Eclat için verilerin düzenlenmesi

TID	**Öğeler**
Ekmek	1,4,5,7,8,9
Tereyağı	1,2,3,4,6,8,9
Süt	3,5,6,7,8,9
Soda	2,4
Reçel	1,8

Eclat algoritması hangi ürünlerin beraber kullanıldığı ya da hangi verilerin birbirine yakın olduğu hakkında kullanıcılarına bilgi veren, diğer birliktelik kural çıkarımı algoritması olan Apriori ‘in aksine yatay formatta değil dikey formattaki verilen çalışan bir yöntemdir. Eclat’taki ana veri setindeki sık kullanılan ögeleri bulmaktadır. Buna derin öncelikli arama denilmektedir.

Üstteki tabloda verilen örneği Eclat ile çözmek istediğimizde şu adımların yapılması gerekmektedir: Öncelikle yatay formatta verilen verileri dikey formata çevrilmelidir. Sıklıklarına göre (örneğin ürünlerin sepette bulunma miktarı) numaralandırılan ögelerden,

kullanıcının seçtiği minimum destek değerine (minsup) göre elemanlar atılmalıdır. Aşağıda adımları gösterilen örnekte bu değer 2'dir. Tek sayıda bir işlem numarası listesi olmadığı için atılan bir öğe olmayacaktır. Saha sonra ikinci adım olarak sepetteki öğelerin ikili kombinasyonları şeklinde sıklıkları bulunacaktır. Bu listeden de tek kalan işlem sıklığının olduğu satır yok edilecektir. Daha sonra öğelerin üçlü kombinasyonları yapılacak ve bu işlem TID sütunundaki tüm sayı değerleri minimum destek değerinden küçük oluncaya kadar devam edecektir. Örneğin minimum destek değeri 2 olarak seçildiğinde, işlem numarası listesindeki değerlerin hepsi 1 olduğunda durmak ve bir önceki adımda elde edilen tablodaki önerilen kombinasyonlar kullanılmaya başlanmaktadır.

Tablo 2.3: Eclat uygulaması için birinci iterasyon

Öğeler	**TID**
Ekmek, Tereyağı	1,4 ,8,9
Ekmek, Süt	5,7,8,9
Ekmek, Soda	4
Ekmek, Reçel	1,8
Tereyağı, Süt	3,6,8,9
Tereyağı, Soda	2,4
Tereyağı, Reçel	1,8
Süt, Reçel	8

İkili kombinasyonları yapılan öğelerden (alışveriş sepetindeki ürünler) tek sayıda olan işlem numarasına sahip olanlar atılmaktadır. Bunlar "Ekmek, Soda" ve "Süt, Reçel" satırlarıdır. Daha sonra tekrardan baştan başlanarak üçlü ürün kombinasyonları yapılmaya başlanmaktadır.

Tablo 2.4: Eclat uygulaması ikinci iterasyon.

Öğeler	**TID**
Ekmek, Tereyağı, Süt	8,9
Ekmek, Tereyağı, Soda	4
Ekmek, Tereyağı, Reçel	1,8
Tereyağı, Süt, Soda	---
Tereyağı, Süt, Reçel	8

Yukarıda yapılan üçlü kombinasyon önerilerinde "Ekmek, Tereyağı, Soda" ve "Tereyağı, Süt, Reçel" seçenekleri bir adet işlem numarası içerdiği için önerilerden silinmektedir. "Tereyağı, Süt, Soda" ile ise hiç işlem yapılmadığı için (örneğe göre bu ürünler hiç beraber alınmamış) tablodan kaldırılmıştır. Geriye aşağıda verilen tablo elde edilmiştir.

Tablo 2.5: Eclat uygulaması üçüncü iterasyon.

Öğeler	**TID**
Ekmek, Tereyağı, Süt	8,9
Ekmek, Tereyağı, Reçel	1,8

Tekrardan aynı işlemler dörtlü kombinasyonlar için uygulandığında aşağıdaki tablo elde edilmektedir.

Tablo 2.6: Eclat uygulaması dördüncü iterasyon.

Öğeler	**TID**
Ekmek, Tereyağı, Süt, Reçel	8
Ekmek, Tereyağı, Süt, Soda	---
Tereyağı, Süt, Soda,Reçel	---

Yukarıda görüldüğü üzere elde edilen dörtlü kombinasyondaki en küçük işlem numarası sayısı, minimum destek sayısı olan 2'den küçük olduğu için bir önceki iterasyonda elde edilen üçlü kombinasyon tablosunu dönülmektedir. Bu topla kullanıcılara birliktelik kural çıkarımı yöntemini kullanarak, beraber gerçekleştirilebilecek önerilerde bulunmaktadır. Örnek

üzerinden anlatılmak istenirse Ekmek, Tereyağı, Süt ve Ekmek, Tereyağı, Reçel ürünlerini market sahiplerine ya da internet satışı yapan kullanıcılara müşterilerine, bu gruptaki ürünlerden herhangi birini aldığında diğerlerini öneri olarak sunmalarına veya satış politikalarını belirlerken kullanmalarına olanak vermektedir.

Apriori (Önsel) algoritması birliktelik kural çıkarımı algoritmalarından birisidir. Eclat algoritmasının aksine verileri dikey değil yatay biçimde taramaktadır. Derin öncelikli arama değil, yayılım öncelikli arama (breadth first search) kullanılmaktadır. Eclta'a göre daha büyük veri setlerinde kullanılmakta bundan dolayı daha yavaş çalışmaktadır. Algoritma destek, güven ve artış adındaki parametreleri ile çalışmaktadır.

Destek parametresi, Eclat algoritmasında olduğu gibi bir market sepeti düşünüldüğünde yapılan N sayıdaki işlemin kaçında A ürününün bulunduğunu gösteren parametredir.

$$Destek = {A \text{ ürününü } içeren\ alışveriş\ sayısı}/{Tüm\ alışveriş\ sayısı} \quad (2.5)$$

Güven parametresi, sadece A ürünü içeren işlemlerin kaçında B elemanının da olduğu gösteren parametredir.

$$Güven = {A\ ve\ B \text{ ürün}lerin\ içeren\ işlem\ sayısı}/{A \text{ ürününü } içeren\ işlemlerin\ sayısı} \quad (2.6)$$

Artış parametresi, A'nın satın alındığı durumlarda B'nin de alınırsa meydana gelecek olan güven artışını ifade etmektedir.

$$Artış = {Güven(A \rightarrow B)}/{Destek(B)} \quad (2.7)$$

Aşağıda gösterilen tablodaki öğelere ve işlem sayılarını baz alarak Apriori algoritmasını kullanıp, birliktelik kural çıkarımı yöntemiyle kombinasyon önerileri yapılabilmektedir. Öncelikle Eclat 'da olduğu gibi kullanıcı tanımlı bir minimum destek değeri tanımlanmalıdır. Bu örnek için bu değer 3 olarak kabul edilsin.

Tablo 2.7: Apriori için örnek uygulama listesi.

İşlem Listesi	**Öğeler**
100	A, C, D
200	B, C, E
300	A, B, C, E
400	B, E

İlk olarak verilen tablodaki her bir değer için destek parametresi elde edilmelidir. Bu değerler hesaplandığında aşağıdaki tabloya geçiş yapılmaktadır.

Tablo 2.8: Apriori uygulaması birinci iterasyon.

Öğeler	**Destek**
A	2
B	3
C	3
D	1
E	3

Sonraki adımda, en başta belirlenen minimum destek değerinin altında olan değerler tablodan yok edilmelidir. Bu işlem yapıldığında aşağıdaki tablo edilmektedir. Elde edilen bu tablodaki öğeler, baştan başlayarak ikili kombinasyonlar şeklinde yazılacaktır. Yazılan ikili öğelerin en baştaki tabloda beraber bulunma değerleri hesaplanacak ve destek sütununa yazılacaktır.

Tablo 2.9: Apriori uygulaması ikinci iterasyon.

Öğeler	**Destek**
B	3
C	3
E	3

Tablo 2.10: Apriori uygulaması üçüncü iterasyon.

Öğeler	**Destek**
B, C	2
B, E	3
C, E	2

Elde edilen yukarıdaki tabloda, minimum destek değerinin altında kalan destek değerine sahip olan kombinasyonlar yok edilecektir. Böylece geriye aşağıdaki tablo elde edilecektir.

Tablo 2.11: Apriori uygulaması dördüncü iterasyon

Öğeler	**Destek**
B, E	3

Elde edilen iki kombinasyona göre, Apriori algoritması kullanıcılarına bu iki ürünün beraber iyi bir kombinasyon olduğunu verebilmektedir.

2.3.1.3. Yarı Denetimli Öğrenme

Etiketli ve etiketsiz verilerin bir arada olduğu öğrenim şeklidir. Bu öğrenme şeklinde hem denetimli hem de denetimsiz öğrenme algoritmaları hibrit şekilde çalışır. Genellikle çok sayıda etiketsiz ve az sayıda etiketli verinin bulunduğu karışık verilerle çalışırlar. Etiketli verilerden öğrendikleri hedefleri yani muhtemel sonuçları, etiketsiz verilerdeki kümelerle eşleştirebilirler. Google Photos ve Pinterest gibi uygulamalar yarı denetimli öğrenmeye birer örnektir.

Şekil 2.34: Yarı denetimli öğrenmedeki veri gösteriminin farkları **[26]**.

2.3.1.4. Pekiştirmeli Öğrenme

Pekiştirmeli Öğrenme (Reinforcement Learning), makine öğrenmesinin bir tekniği olup çalışma prensibi bakımından denetimli öğrenme ve denetimsiz öğrenmeden farklı olmaktadır. Pekiştirmeli öğrenme, diğer makine öğrenmesi sistemlerinden çok farklı olan ve üzerinde ayrı olarak sürekli çalışmaların yapıldığı bir alandır. Diğer öğrenme türlerinde olduğu gibi konu giriş verisinin etiketli ya da etiketsiz olmasına göre bunların sınıflandırılması veya benzerlik gösteren verilen en uygun şekilde kümelenmesi değil, çalışma sürecinin ödül ve ceza sistemine göre ardışık şekilde davranış göstermesi üzerine kuruludur. Pekiştirmeli Öğrenmede, ajan adı verilen sistem biriminin sürekli bir öğrenme eylemi içerisinde, doğru bildiği eylemlerden ödül yani pozitif, yanlış yaptığı eylemlerden de ceza yani negatif geribildirim uygulaması sayesinde modelin öğrenmesi pekiştirilmektedir. Bu yüzden Pekiştirmeli Öğrenme adını almaktadır. Bu teknikte ajanın yani sistemin belirlediği eylemler bütününe, yani seçtiği algoritmaya politika denmektedir. Politika, ajanın içinde bulunduğu görevi gerçekleştireceği çevrede karşılaştığı durumlara karşı verdiği cevaplardır. Ajan adı verilen birim nesnel bir varlık olarak değil, eylemleri gerçekleştirip öğrenmeyi sağlayan bir araç olarak kabul edilmektedir. Başlangıçta rastgele olarak belirlenen eylemler, model ödül ve cezalarla eğitildikçe, eğitimin gerçekleştiği çevreye uyum sağlayacak ve mantıklı kararlar vermeye başlayacaktır. Pekiştirmeli Öğrenmede amaç öğrenme işlemi sonrasında sistemin maksimum seviyede ödüle ya da sürekli negatif geribildirim alabileceği durumlar karşısında minimum ceza puanına ulaşmasını sağlamaktır. Ödül ifadesi pekiştirmeli öğrenme sistemlerinde haz, acı kavramları ile ilişkilendirilip olup geri bildirim sinyali olarak görev yapmaktadır. Ajan bu geri bildirimlere göre deneyimlerinden yararlanarak aksiyonlar arasından hangisinin daha olumlu yanıtlar alacığına göre seçim yapabilmektedir. Bazı durumlarda, ajanın karşılaştığı sorunlardan anlık olarak yüksek puan ya da düşük ceza puanına sahip eylemi seçmek yerine tam tersi bir eylem seçilebilmektedir. Bunun sebebi anlık olarak dezavantajlı görünen eyleme bağlı olan diğer eylemlerin daha yüksek pozitif

veya düşük ödül cezalarına sahip olmasından kaynaklanmasıdır. Bu pekiştirmeli öğrenmede durum değeri olarak adlandırılmaktadır. Anlık değil olası toplam ödül ya da ceza puanı olarak kabul edilmektedir. Amacı ödülleri eniyilemek olan pekiştirmeli öğrenme sistemlerinde politika seçimi performansı arttırmada çok önemli bir yer tutmaktadır. Görev boyunca olabilecek muhtemel tüm sonuçları karşılayabilecek şekilde dinamik bir politika seçimi ajanın elde edebileceği ödülü eniyilemekte büyük önem taşımaktadır. Pekiştirmeli öğrenme sistemleri model adı verilen çevreden, bağımsız olarak modelsiz olarak da kullanılabilme ve modelin aksine önceden eğitilmeden de performans gösterebilmektedirler [27]. Pekiştirmeli öğrenme deneyimin önem arz ettiği deneme yanılma yöntemiyle öğrenmesi açısından masa üstü ve bilgisayar oyunlarında, akıllı ev araç gereçlerinde sıklıkla kullanılmaktadır.

Denetimli ve denetimsiz öğrenmelerin aksine eğitim setinin kullanılmadığı durumlarda görev alan pekiştirmeli öğrenme, ajanların görevi birden çok kez yerine getirip öğrenmesi ile gerçekleştirilmektedir. Bunun sebebi çok sayıda bulunan belirsiz koşul ve alt koşullar sebebi ile ajanın görevi tanıyabilmesidir. Pekiştirmeli öğrenme eğitim veri setinin olmaması yani herhangi bir ön veriye ihtiyaç duymadan karar alması açısından canlıların özellikle hayvanların karar alma ve öğrenme sistemine benzetilmektedir. Karar verme sürecini ve aldığı kararların doğruluğunu arttırabilmek için geliştirilmiş olan bazı pekiştirmeli öğrenme yöntemleri bulunmaktadır. Bunlardan en çok kullanılanı Markov karar süreçleri ve Q-Öğrenme yöntemidir.

Richard BELLMAN tarafından tanımlanan Markov karar süreçleri, pekiştirmeli öğrenme sistemlerinde aksiyomları gerçekleştiren ajanların, bulundukları durumdan bir sonraki duruma geçmeleri için vermeleri gereken kararları matematiksel olarak formüle etmeye yaramaktadır. Markov karar süreçlerine göre t anında S_t durumunda bulunan bir ajanın, S_{t+1} durumuna geçmesi sadece S_t durumuna bağlıdır. Bu kural Markov karar süreçlerinin belleksizliği olarak adlandırılmaktadır çünkü karar verme süreci geçmişteki durumlara değil sadece o andaki (t) duruma bağlıdır. Yaptığı aksiyomlara göre ödül veya ceza alan ve bunu eniyilemeye çalışan ajan için bir sonraki duruma geçmesi, karar vermesi gereken bir veya birden fazla aksiyomun olasılıklarına bağlıdır.

Önceden bilinmeyen stokastik yani rastgele süreçlerde karar alma mekanizması olarak çalışan Markov süreçleri (Markov zincirleri) gelecekteki olası durumları hesaplayabilmek için güncel durumdaki olasılıkları kullanmaktadır. Bu özelliği taşıyan sistemleri diğer bir adı ise

Markovyen sistemlerdir. Markovyen sistemleri matematiksel olarak göstermek istersek, ($t_1<t_2<t_3\ldots<t_n$) olan bir zaman kümesi için aşağıdaki olasılık teoremi oraya çıkmaktadır.

$$P\left[X_{t_n} = x_{t_n} \middle| X_{t_{n-1}} = x_{t_{n-1}}, \ldots, X_{t_1} = x_1\right] = P[X_{t_{n-1}} = x_n | X_{t_{n-1}} = x_{n-1}] \quad (2.8)$$

Bu teoremi ajanın ve ödülün olduğu bir pekiştirmeli öğrenme sistemine uygulamak istersek aşağıdaki formülü elde edilmiş olur.

$$P[S_{t+1}|S_t] = P[S_{t+1}|S_1, S_2, S_3, \ldots, S_t] \quad (2.9)$$

Ödül değerinin R ile gösterildiği bir Markov karar süresinde (MKR) toplam ödül değeri sürekli şekilde artmaktadır. Toplam ödül değerini gösteren formül aşağıda gösterildiği gibidir. G toplam ödül değerini temsil etmektedir.

$$G = R_1 + R_2 + R_3 + \cdots + R_n \quad (2.10)$$

Bir pekiştirmeli öğrenme modelinde, öğrenen ajan her zaman periyodik sürelerle farklı durumlarla karşılaşmayabilir ya da karşılaşıp gerçekleştirdiği aksiyonların ödüllerini hemen almayabilir. Bu sistemde, G toplamını sonsuza doğru götürmektedir. Bu durumda görevleri gerçekleştiren ajanların, sistemi yavaşlatacak ödüllere sahip aksiyonları gerçekleştirmemeleri için indirim faktörü denilen (discount factor) bir çarpan belirlenmiştir. Sıfır ile bir arasında bir değer alan "γ" ile gösterilen indirim faktöründe, $\gamma = 0$ değeri için ajan uzun vadeye yayılan ödüllendirmelere sahip hareketlerden kaçınacak, $\gamma = 1$ olduğu durumda ise ajanın gelecekte alacağı ödüller daha önemli hale gelecek ve buna göre seçimlerini yapacaktır. İndirim faktörü değeri ödüllerin değerinden bağımsız olan tamamen ajanın gelecekte alacağı ödüller ile ne kadar ilgilenip ilgilenmediğini belirleyen bir değişkendir. İndirim faktörünün eklenmiş olduğu indirim faktörlü toplam ödül değeri aşağıda gösterildiği gibidir.

$$G_t = R_t + R_{t+1} + R_{t+2} + \cdots = \sum_{k=0}^{\infty} \gamma^k R_{t+k} \quad (2.11)$$

Kısacası bir Markov karar süreci ödül değerine, alınan aksiyona, durum değerine, bu durumları gerçekleştirecek olan aksiyonların gerçekleşme olasılığına ve indirim faktörü değişkenlerine bağlı bir karar alma sürecidir.

Richard BELLMAN'a göre, alabileceği ödül değerini eniyilemek isteyen bir ajanın aşağıda gösterilen Bellman Optimal Eşitlik Değeri denklemine uyması gerekmektedir. Bu denklem gelecekte yapacağı aksiyonların hangilerinin kendi için daha yüksek ödül ya da düşük ceza toplam puanına sahip olduğu kararını verirken kullanılmaktadır.

$$Q^{\pi}(s,a) = \sum_{s'} T(s,a,s')(r(s,a,s') + \gamma \sum_{a'} \pi(s',a')Q^{\pi}(s',a')) \quad (2.12)$$

$s_t = t\ anındaki\ durum$

$s'_t = s\ durumundan\ sonraki\ durum\ (\ s'_t = s_{t+1})$

$a_t =\ s_t\ durumundan\ s'_t\ durumuna\ geçilmesi\ için\ yapılması\ gerek\ aksiyon$

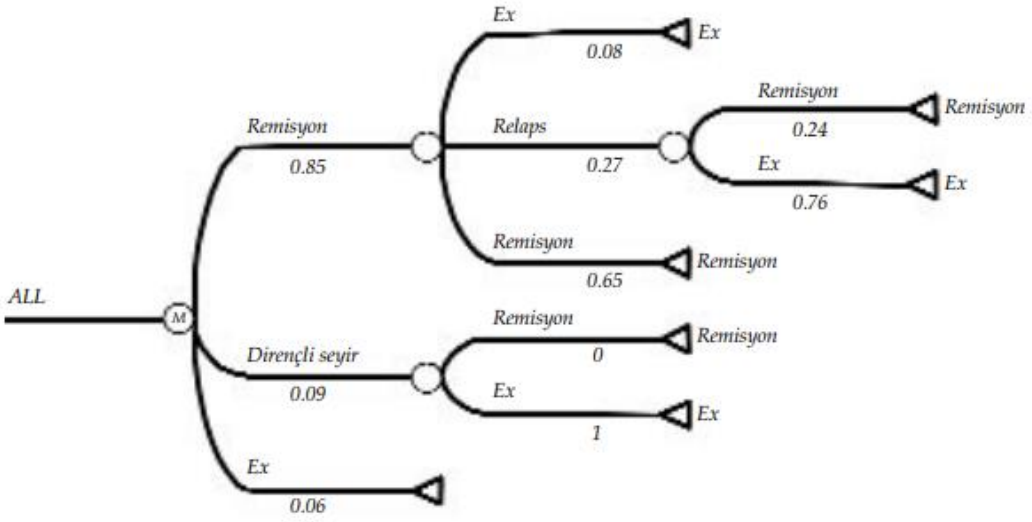
$r_t = r(s_t, a_t, s'_t)$

$= s_t\ durumundan\ s'_t\ durumuna\ geçileince\ kazanılan$ ödül $ya\ da\ ceza$

$T(s,a,s') =\ s_t durumundan\ s'_t\ durumuna\ geçme\ olasılığı$

$\pi(s,a) = ajanın\ uyguladığı\ karar\ verme\ politikası$

Aşağıdaki görselde yer alan Necdet SÜT ve arkadaşlarının gerçekleştirdiği "Sağlık Alanında Karar Vermede Döngüsel Süreçlerin Kullanımı: Bir Markov Model Uygulaması" adlı çalışmada yer alan akut lenfoblastik lösemi (ALL) hastalığında tedavi aşamalarına ilişkin oluşturulan bir Markov karar alma sürecine ait olan zincir gösterilmiştir. Belirtilen sayısal değerler ödül değerleri değil, zincirde yer alan durumlara geçiş olasılıklarıdır. Modelde yer alan geçiş olasılıklarının oluşturduğu geçiş matrisi ve izin verilen geçiş durumları aynı adlı çalışmada detaylı olarak yer almaktadır [28].



Şekil 2.15: Aynı adlı çalışmada kullanılan Markov modeli **[28]**.

Diğer bir pekiştirmeli öğrenme algoritması Q-Öğrenme algoritmasıdır. Q-Öğrenme algoritması, pekiştirmeli öğrenmenin model oluşturulmadan kullanılan bir algoritmasıdır. Q-Öğrenme algoritması yapı itibariyle Richard Sutton'un geliştirdiği Zamansal Fark Öğrenimi (Temporary Difference Learning, TDL) yöntemi ile benzerlik göstermektedir.

Zamansal fark öğrenimi yönteminde amaç, gelecekte yer alacak olan toplam ödül değerini tahmin etmektir. Zamansal fark öğrenimi yöntemi modele ihtiyaç duymadan geçmiş durumlardan öğrendiği eksik tecrübelerle ajanın son duruma varmasını beklemeden her durumda kendini güncelleyerek sonuç vermesidir. Belirtilen sonuçtan kasıt optimal indirimli durum değeridir. Bütün işlemler başlangıçta rastgele belirlenen bir politika altında gerçekleşmektedir.

Q-Öğrenme algoritması, politika dışı zamansal fark öğrenimi olarak kabul edilebilmektedir. Amacı $Q(s,a)$ değerini hesaplamak olan algoritma, bu her t anı için kendini güncellediği için bu değerin ortalaması hesaplanmaktadır. Bu değer $Q\hat{}(s,a)$ olarak kabul edilmektedir.

$$Q\hat{}_N(s,a) = \frac{1}{N}\sum_{n=1}^{N} Q_n(s,a) \quad (2.13)$$

$n = N-1$ *durumu için*:

$$Q\hat{}_N(s,a) = \frac{1}{N}\left(Q_N(s,a) + \sum_{n=1}^{N-1} Q_n(s,a)\right) \quad (2.14)$$

$$Q\hat{}_N(s,a) = \frac{1}{N}\left(Q_N(s,a) + NQ_{N-1}(s,a) - Q_{N-1}(s,a)\right) \quad (2.15)$$

$$Q\hat{}_N(s,a) = Q_{N-1}(s,a) + lr(Q_N(s,a) - Q_{N-1}(s,a)) \quad (2.16)$$

Formülde gösterilen lr değeri öğrenme oranı olarak kabul edilmekte ve ortalama değerin hesaplanacağı adım aralığı olmaktadır. Rastgele olmayan sistemler için Bellman Optimal Eşitlik Değeri denklemi uyarlanmak istenirse aşağıdaki denklem elde edilmektedir.

$$Q(s,a) = r(s,a,s') + \gamma max_{a'} Q(s',a') \quad (2.17)$$

(2.17) numaralı denklemde yerine koyulursa Q-öğrenme algoritmasında optimal olarak indirimli durum değeri fonksiyonu şu hale gelmektedir.

$$Q\hat{}_N(s,a) = Q_{N-1}(s,a) + lr(r(s,a,s') + \gamma max_{a'} Q(s',a') - Q_{N-1}(s,a)) \quad (2.18)$$

Bu denklem N sayıdaki her karar alma sürecinde tekrarlanacak ve güncellenecektir.

2.3.2. Denetimli Öğrenme Algoritmaları

2.3.2.1. K-En Yakın Komşular Yöntemi

K-En yakın komşu algoritması bir denetimli öğrenme algoritmasıdır. Hem sınıflandırma hem de regresyon problemlerinde kullanılmaktadır. Hangi sınıfa ait olduğu bilinmeyen verinin eğitim setindeki diğer verilere uzaklıklarını kıyaslayarak hangisine daha çok benzediğine göre sınıfını belirlemeye yarayan algoritmadır. KNN'deki mantık birbirine benzeyen verilerin

birbirine yakın olmaları üzerine dayalıdır. Normal sınıflandırma algoritmaları belirli sınıf üzerinden sınıf bilgilerini atarken KNN algoritması her veri noktası için, o noktaya en yakın komşu veriler üzerinden bir sınıflandırıcı oluşturarak değerleri sınıflandırır. Tekrarlanan sınıflayıcılar yüzünden KNN'in diğer adı Tembel Öğrenci Algoritmasıdır. KNN algoritması uygulanan işlemlerin uzun olmasına karşın doğruluk payı yüksek ve verideki gürültülere dirençli bir algoritmadır. Veri noktaları arasındaki uzaklıkları ölçerken kullanılan bazı uzaklık fonksiyonları şunlardır; Öklid uzaklık fonksiyonu, Manhattan uzaklık Fonksiyonu, Çebişev uzaklık fonksiyonu vb.

$$\sqrt{(p_1 - q_1)^2 + (p_2 - q_2)^2 + \ldots + (p_n - q_n)^2} = \sqrt{\sum_{i=1}^{n}(p_i - q_i)^2} \quad (2.19)$$

$$d_i(p, q) = \sum_{i=1}^{n}|p_i - q_i| \quad (2.20)$$

$$D_{Chebyshev} = max(|x_2 - x_1|, |y_2 - y_1|) \quad (2.21)$$

KNN algoritması şu şekilde çalışmaktadır:

- Veri yüklenir.
- k parametresi yani incelenecek komşu sayısı belirlenir.
- Belirlenen komşuların her biri için, belirlenen uzaklık fonksiyonu çalıştırılarak uzaklık hesaplanır. Hesaplanan uzaklıklar bir diziye eklenir.
- Elde edilen uzaklıklar dizisi küçükten büyüğe doğru sıralatılarak dizinin ilk elmanı alınır.
- Seçilen k komşunun etiketi yeni veriye seçilir.

KNN algoritmasında seçilen komşu sayısı önem taşımaktadır. Bunun için algoritma farklı k değerleri için birkaç kez çalıştırılarak, yeni veriler için en uygun tahmini yapabilecek model seçilmeye çalışılır.

KNN yöntemi uygulaması kolay olan bir makine öğrenmesi algoritmasıdır. K-En yakın komşular yöntemi, model eğitiminin kolay olması, matematiksel formüller üzerine dayalı hesaplaması kolay olması ve gürültünün olduğu karışık veri setlerine karşı duyarlı olması gibi avantajları ile sınıflandırma problemlerinde çokça tercih edilmektedir. KNN algoritmasının bu avantajlarına rağmen bazı dezavantajları da bulunmaktadır. Her nokta için hesaplanan uzaklıkların, işlem yükünü, zamanını ve maliyetini önemli ölçüde arttırmakta, seçilen k komşu sayısının rastgele olmasından dolayı optimal değerin bulunması için birkaç kez işlemin denenmesi gerekmesi bu dezavantajlardan bazılarıdır.

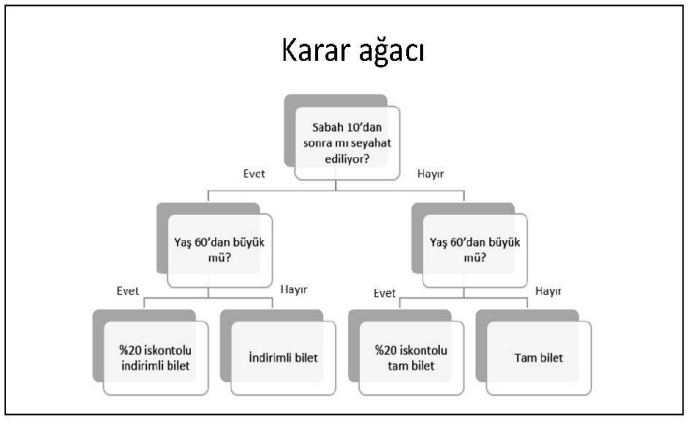
KNN yöntemi nesne ve örüntü tanımlamada sıkça kullanılan bir makine öğrenmesi türüdür. ÇINAR ve arkadaşlarının yaptığı KNN algoritması ile sınıflandırma çalışmasında, IONOLABTEC verilerinden eğitilen model ,99 Marmara depremi ve 99 güneş tutulmasının oluşturduğu hasarları tespit etmek gerçekleştirilmiştir [29] .

2.3.2.2. Karar Ağaçları

Karar ağaçları yöntemi makine öğrenmesinde sınıflandırma ve bağlanım problemlerinde sıkça kullanılan aynı zamanda Rastgele Ormanlar yönteminin de temelini oluşturan bir makine öğrenmesi türüdür. Veriyi ve modeli akış diyagramlarında açıkça temsil etmekte ve adından da anlaşılacağı üzere kökten dallara bir ağaca benzemektedir. Oluşturulan ağaç şemasında dallar kuralları, yapraklar ise çıkış değerlerini vermektedir. Kökten başlayan ağaç yapısı model çıktı verinceye kadar dallanma yapmaktadır. Yapraklar modelin çıktısını temsil etmektedir. Her dal yapısı ve kök, sınıflandırmada kullanılabilecek bir öz niteliği göstermektedir.

Karar ağaçları yöntemi kullanım kolaylığı ve anlaşılır olmalarından dolayı şeffaf kutu modeller olarak bilinmektedir. Modelin yaptığı sınıflandırmanın neye göre veya hangi kurallara göre yapıldığı gayet açıktır. Diğer veri ağaçları gibi üstten aşağıya doğru çalışmaktadırlar. İkili sınıflandırıcılar için basit "if else" döngü yapısıyla çalışmaktadır.

Karar ağaçları algoritması veri setini yinelemeli olarak gruplara ayırarak bu gruplardan öğrenme işlemini gerçekleştirmektedir. Gruplama işlemi her gruptaki düğüm değeri, hedef etiket değerine ulaşıncaya kadar devam etmektedir. Karar ağaçları sürekli ve kategorik verilerle çalışabilmektedir.



Şekil 2.16: Karar ağaçları modeli çalışma prensibi örneği **[30]**.

Ağaç akış şemasını oluştururken, kök değerine ve alt dallara atanacak özelliklerin belirlenmesi büyük önem taşımaktadır. Bunu belirlemek için geliştirilen iki önemli yöntem vardır. Bunlar bilgi kazancı ve Gini indeksidir.

Bilgi ağırlığı verileri kümeleme işlemi yaptıktan sonra elde edilen gruplardaki bilgilerin, sınıflar hakkında yeterince bilgi taşıyıp taşımadığının ölçüsüdür. Bilgi ağırlığının yüksek olması verideki belirsizliğinin azaltılmış olması anlamına gelmektedir. Alt kümelerdeki bilgi miktarı ve entropi yani düzensizlik değerine göre kök ve dal değerleri belirlenmektedir. Model bilgi kazancı miktarını maksimize etmeye çalışmaktadır. Aşağıdaki formül bilgi kazancını göstermektedir.

$$Bilgi\ Ağırlığı = Entropi_{TOPLAM} - (Ağırlıklı\ Ortalama * Entropi_{Her\ özellik\ için}) \quad (2.22)$$

$$Entropi = -k * \sum_{i=1}^{n} p_i * \log_2 pi \quad (2.23)$$

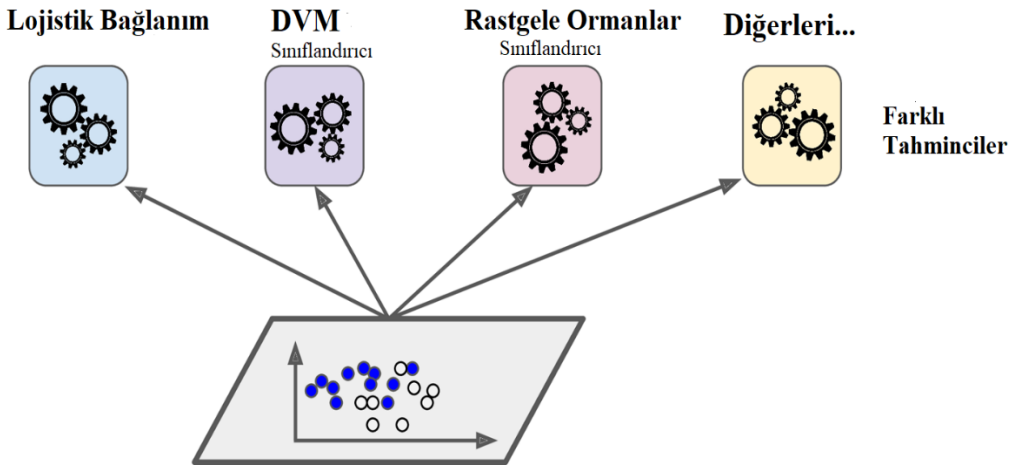
Gini indeksi ya da diğer adıyla Gini safsızlığının formülü de aşağı da gösterildiği gibidir.

$$Gini\ İndeksi = 1 - \sum(P(x = k))^2 \quad (2.24)$$

Karar ağaçları yöntemi performans olarak iyi ve karmaşık verilere karşı dayanıklı olsa da veri setlerinde meydana gelecek en küçük değişikliklere karşı hassastırlar ve aşırı uydurmaya karşı eğilimlidirler. Bu dezavantajlar doğrultusunda Rastgele Ormanlar yöntemi ortaya çıkmaktadır.

2.3.2.3. Topluluk Öğrenimi ve Rastgele Ormanlar

Topluluk öğrenimi (ensemble learning) kavramı makine öğrenmesi türleri içinde yapısı ve kullanımı itibari ile en farklı olan türüdür. Sınıflandırma veya bağlanım fark etmeksizin, oluşturulmak istenen modelin, tek bir modelden değil kendi içinde birden fazla modelin oluşturduğu sonuçların, ana model tarafından oylanarak bulunduğu bir makine öğrenmesi türüdür. Ana model, birden fazla modelin bilgisinden oluştuğu için başarımı tek bir modele göre daha yüksek ve güvenilirdir. Kullanılan alt modeller birbirinden farklı türlerden olabilmektedir. Farklı türlerden sınıflandırıcıların oluşturduğu modelde, farklı algoritmalar çalışacağı ve her tahmin edicinin yaptığı hatalar farklı olacağı için modelin genelleme yapabilme yeteneği yüksek olacaktır. Topluluk öğrenmesi ile oluşturulan ana model, diğer makine öğrenmesi türleri ile geliştirilen tekli modellere göre daha esnek yapılı ve daha kullanışlıdır. Bu yüzden de Topluluk öğrenme yöntemi makine öğrenmesinin en çok kullanılan ve en popüler olan dallarından biridir.



Şekil 2.17: Topluluk öğrenmesi yöntemi **[31]**.

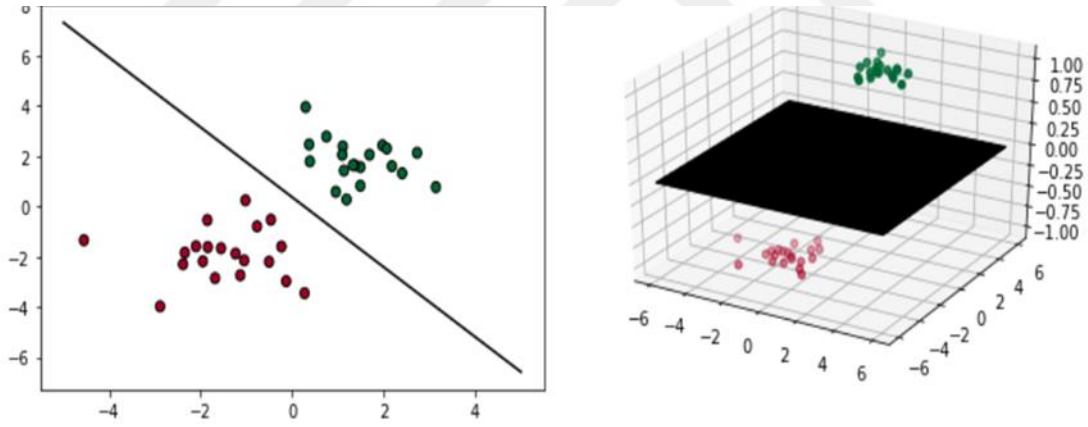
Topluluk öğrenme yöntemi, kullanılan modellerin aş zamanlı ya da sıralı olarak kullanılmasına göre iki ayrı metottan oluşmaktadır. Bunlar paketleme metodu (bagging) ve arttırma metodu (boosting) 'dur. Paketleme metodunda, ana modeli oluşturan her alt model eş zamanlı olarak çalışmakta ve veri setinin birer alt kümesiyle eğitilmektedir. Arttırma metodunda ise, alt modeller sıra ile çalışmakta ve kendilerinden önceki modellerin öğrendikleri bilgileri giriş olarak almaktadırlar.

Rastgele Ormanlar yöntemi paketleme ve arttırma metotlarının herhangi birisi ile çalışabilen bir topluluk öğrenme yöntemidir. Aynı zamanda denetimli bir makine öğrenmesi türüdür. Topluluk öğrenme yöntemindeki tüm tahmin edicileri karar ağaçlarından meydana gelmesi ile Rastgele Ormanlar adını almaktadır. Ana modeldeki karar ağacı sayısının artması ile paralel modelin başarımı da artmaktadır. Veri setinin alt kümleriyle eğitilen bir grup karar ağaçlarından oluşmaktadır. Sınıflandırma ve bağlanım problemleri için kullanılabilmektedir. Her karar ağacının çalıştığı veri seti, ana verisinin rastgele bir alt kümesi olacak şekildedir. Eş zamanlı olarak eğitilen her karar ağacının oluşturduğu tahminler ana model tarafından toplanıp oylanmaktadır. En çok oyu alan tahmin ana modelin tahmini olarak ortaya çıkmaktadır. Buna sert oylama sınıflandırıcı denmektedir. Örnek olarak yeni bir bilgisayar almak için kullanıcısına satın al ya da alma diye çıktı veren bir Rastgele Ormanlar modeli varsayılsın. Modelin 4 karar ağacından meydana geldiği ve RAM, ağırlık, fiyat ve marka niteliklerini işlediği kabul edilsin. Model eğitimin sonunda her karar ağacının çıktısına göre oylanıp ve çoğunluğun oyu ile tek bir karar ağacına göre daha yüksek bir başarı ile satın al ya da alma çıktısı verebilmektedir. Bağlanım problemlerinde ise, alt modellerin oluşturduğu birden fazla sürekli tahminin ortalamaları alınarak ana modelin çıktısı olarak verilmektedir.

Tek bir karar ağacı yerine çok sayıda karar ağacından oluştuğu için, rastgele ormanlar yöntemi Karar Ağaçları yöntemine göre daha başarılıdır Karar ağaçlarında görülen aşırı uydurma gibi sorunlar rastgele ormanlarda çözüme ulaşmıştır. Sağlık ve finans alanı başta olmak üzere çoğu alanda kullanılan yöntem son yıllarda e-ticaret alanında tüketicilerinin geçmiş davranışlarından yararlanarak müşterilerine satın almaları için yeni ürün önerilerinde bulunmaktadır.

2.3.2.4. Destek Vektör Makineleri

Destek vektör makineleri (Support Vector Machines), makine öğrenmesinin en çok kullanılan ve en popüler olan türlerinden biridir. Lineer olan veya olmayan sınıflandırma problemlerinde ve regresyonda sıklıkla kullanılmaktadır. Destek vektör makineleri lineer ayrılabilen sınıflandırma problemlerinde, karar sınırı ya da hiperdüzlem denilen sınıfları birbirinden ayırmaya yarayan sınırını çizerken, sınıfları mümkün olduğunca birbirlerinden ayırmaya çalışmaktadır. Bunun amacı modelin yeni verilerdeki başarı oranını arttırmaktır. Belirlenen karar sınırına yakın veri noktaları destek vektörü olarak adlandırılmaktadır. N boyutlu bir düzlemin (n-1) adet hiperdüzlemi bulunmaktadır. Destek vektör makineleri (DVM), ikili ya da çoklu sınıflandırmalarda sıklıkla kullanılmaktadır.



Şekil 2.18: Destek vektör makineleri algoritması kullanım örneği **[32]**.

$$b_0 + b_1 * x_1 + b_2 * x_2 + \cdots + b_n * x_n = 0 \qquad (2.25)$$

n boyutlu bir uzay için hiperdüzlem denklemi yukarıdaki formülde olduğu gibi gösterilir. Örneğin 2 boyutlu bir uzayda hiperdüzlemi denklemi aşağıdaki şekilde olur.

$$b_0 + b_1 * x_1 + b_2 * x_2 = 0 \qquad (2.26)$$

Hiper düzlemin üstündeki noktalar:

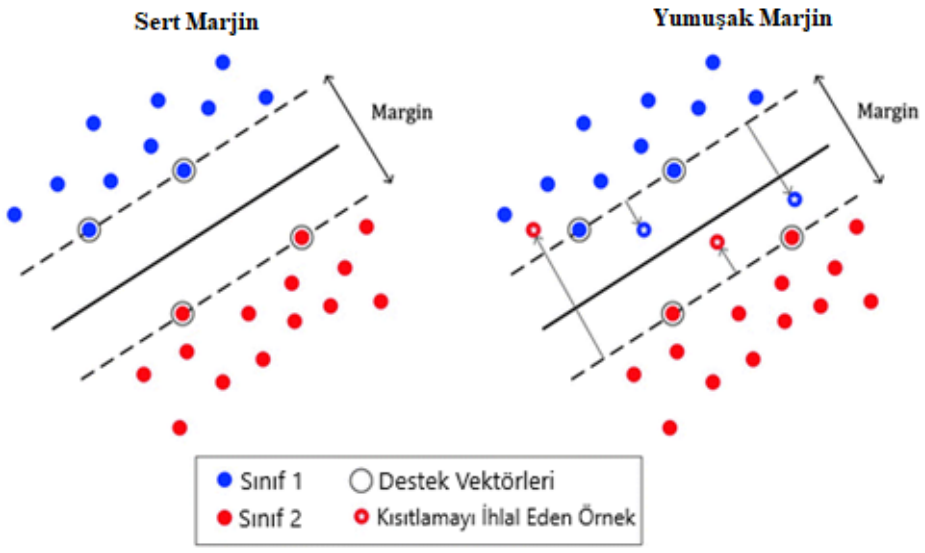
$$b_0 + b_1 * x_1 + b_2 * x_2 > 0 \quad (2.27)$$

Hiper düzlemin altındaki noktalar:

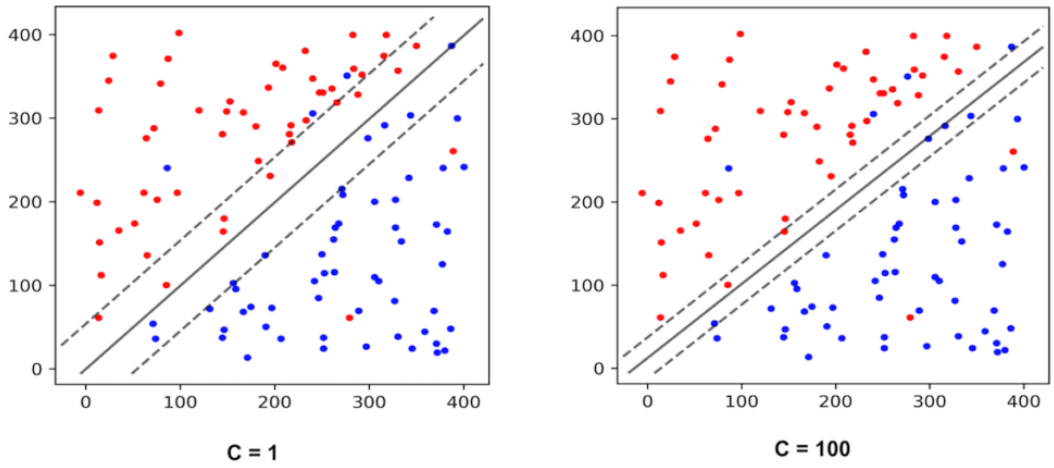
$$b_0 + b_1 * x_1 + b_2 * x_2 < 0 \quad (2.28)$$

Şeklinde ifade edilebilmektedir.

DVM, sınıflandırma yaparken marjin değerini yani veriler arası uzaklığı maksimum yapacak hiperdüzlemi bulmaya çalışmaktadır. Fakat veri noktaları her zaman belirlenilen sokağın yanında değil bazen üstünde hatta marjinin içinde bile olabilir. Eğer marjin değerini belirleme, bu içerideki verileri dikkate alıp yapılırsa buna sert sınırlama denir. Modelin sert sınırlamada genelleme yapabilme yeteneği gelişmemiştir. Böyle zamanlarda hiper düzlemi belirlemede daha esnek davranılabilmektedir. Sokağın üstünde veya içinde olan çok sayıda olmayan değerler görmezden gelinip karar sınırı çizilebilmektedir. Bu DVM'nin karışık ve gürültülü verilere karşı iyi bir performans gösterdiğini belirtmektedir. Buna yumuşak sınırlama denir. Aşağıda yer alan grafikte sert ve yumuşak sınırlama açıkça gösterilmektedir. DVM'lerde sınır esnekliğini belirlemede "C" hiper parametresi kullanılmaktadır. "C" hiper parametresi ne kadar büyük seçilirse sınırlardaki esneklik artacaktır. Modele genelleştirme yaptırabilmek için az sayıda ihlal gerçekleştirebilecek düzeyde "C" seçilmelidir.



Şekil 2.19: Sert ve yumuşak marjinin gösterimi **[33]**.



Şekil 2.20: C parametresinin değişiminin gösterimi **[34]**.

Gündelik hayatta hiçbir olay lineer olarak gelişmez. Bundan dolayı bu tür olayların ortaya çıkardığı sorunlar makine ortamına aktarıldığında çoğu zaman lineer bir düzlemde gösterilemezler. Lineer olmayan destek vektör makinesi sınıflandırmada, veri setini anlaşılır şekilde gösterebilmek için, öznitelik eklemesi yapılabilmektedir. Bu lineer bir veri dağılımı elde edilmesini sağlamaktadır. Lineer olarak gösterilemeyen verileri destek vektör makineleriyle sınıflandırırken kullanılabilecek bazı yöntemler vardır. Bunlardan en çok kullanılanı Kernel Hilesi denilen yöntemdir.

Lineer gösterilemeyen verilere eklenen öznitelik yöntemi küçük veri setlerinde soruna yol açmasa da karmaşık ve büyük veri setlerinde, yüksek dereceden çok fazla öznitelik olmasına yani boyut sayısının büyümesine sebep olmaktadır. Bu da sistemin performansını yavaşlatmakta ve işlem hızını düşürmektedir. Kernel Hilesi yöntemi bu sebepten geliştirilmiştir. Polinom kerneli adı verilen yöntem kernel hilesinin bir alt yöntemi olup, modele polinomsal nitelik eklemeden eklemiş gibi davranış göstermesini sağlamaktadır. Böylelikle boyutsal olarak büyümeyen ama öyleymiş gibi davranan model başarılı sınıflandırma yapabilir bir yandan da performansını yavaşlatmamış olmaktadır. Polinom kerneli kullanılan modelde, model aşırı uyduruyorsa polinom derecesi küçültülür, tam tersi az uyduruyorsa polinom derecesi büyütülmelidir.

2.3.2.5. Doğrusal Bağlanım

Tıptan matematiğe sosyal bilimlerden istatistiğe birçok alanda kullanılan regresyon yöntemi, sebep sonuç ilişkisi bulunan iki veya daha fazla değişken arasındaki ilişkiyi inceleyen ve makine öğrenmesi ile modelleyen bir analiz yöntemidir. Kullanılan değişkenlerin sınıfsal ya da

sayısal bir değeri olmasına göre doğrusal regresyon ve lojistik regresyon olmak üzere ikiye ayrılır.

Lineer regresyon (doğrusal bağlanım) bağımlı ve bağımsız değişkenler arasındaki ilişkinin lineer bir doğru ile ifade edilebilmesine denmektedir. Matematiksel olarak formüle etmek istersek aşağıdaki sonuçları elde etmiş oluruz.

$$y' = b + w * x + e \qquad (2.29)$$

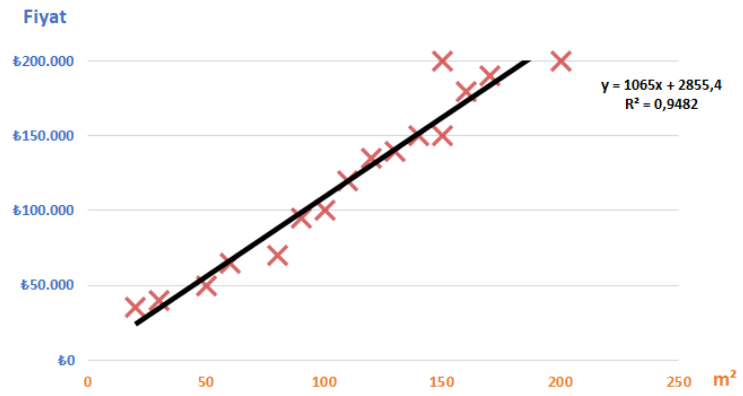
$y' = tahmin\ edilen\ etiket\ değeri$

$b = baz\ değeri$

$w = doğrunun\ eğimi$

$x = özellik$

$e = artık\ değer\ (y - y')$



Şekil 2.21: Doğrusal bağlanıma örnek gösterim **[35]**.

Yukarıda verilen grafik evlerin metrekaresine göre değişen satış fiyatlarını göstermektedir. X ekseninde metrekare sayısını, y ekseninde ise satış fiyatını yani modelin etiket değerleri bulunmaktadır. Her bir giriş değerine karşılık gelen etiketlere göre bir lineer doğru çizildiğinde elimizde lineer bir bağlanım modeli olmuş olmaktadır. Bu modele bakarak bir giriş değeri için ona karşılık gelen sürekli olan bir etiket değeri bulunabilmektedir.

Modelin başarımını ölçmek için tahmin edilen etiket değeri ile gerçek etiket değerinin arasındaki fark alınmalıdır. Bu formülde de geçen artık değerdir. Makine öğrenmesinde artık değer kayıp fonksiyonu olarak ifade edilmektedir. Lineer regresyon problemlerinde kayıp değeri bulmak için en çok kullanılan yöntem Ortalama Karesel Hata (MSE) yöntemidir.

Ortalama Karesel hata (MSE), regresyon grafiğinin eğitimde yer alan noktalara ne kadar uzaklıkta olduğuna bakmaktadır. Bunu regresyon çizgisinden elde edilen tahmini değer ile hedeflenen değer arasındaki farkın karesini alarak yapmaktadır. Karesini almasının sebebi negatif işareti ortadan kaldırmaktır. Birden fazla tahmini değer ve gerçek değer arasında bu işlemi yaptığı için ortalama adını almaktadır. Aradaki fark yani MSE sıfıra ne kadar yakın olura modelin hata payı az, başarımı o kadar yüksek olur. MSE ne kadar fazla çıkarsa modelin başarımı o kadar düşük olur. Ortalama Karesel Hata formülü aşağıda gösterildiği gibidir.

$$MSE = \frac{1}{n}\sum_{i=1}^{n}(y_i - y_i')^2 \quad (2.30)$$

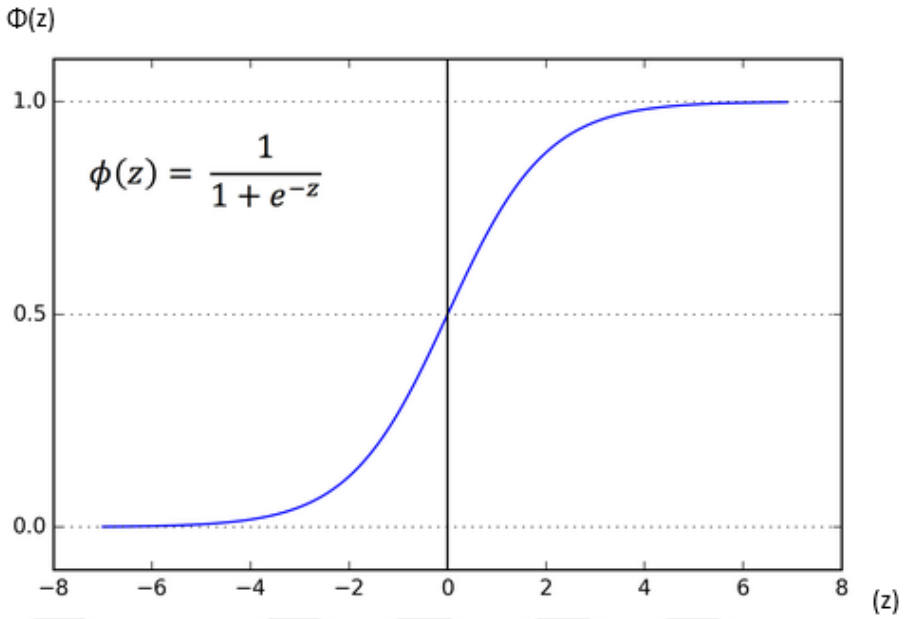
y'= tahmin edilen etiket değeri

y'= hedeflenen etiket değeri

2.3.2.6. Lojistik Bağlanım

Lojistik bağlanım sınıflandırma problemlerinde kullanılan denetimli bir makine öğrenmesi türüdür. Lineer regresyon nicel bir çıktı değil, verilen girdinin belirli bir sınıfa ait olma olasılığını vermektedir. Örnek olarak e-posta kutusuna gelen yeni bir mesajın yüzde on olasılıkla spam bir mesaj olma olasılığı gibi. Lojistik bağlanım da lineer bağlanımda olduğu gibi verilen giriş değerlerine dayanarak sonuç üretilmektedir. Lineer ve Lojistik regresyon arasında bazı farklar bulunmaktadır. Lineer regresyonda sürekli değişkenler kullanılırken, lojistik bağlanımda ayrık değişkenler kullanılmaktadır. Lojistik regresyon, sonuç değişkeninin rastgele bir olay olduğunu varsayar. Örneğin, vücut yağ oranına göre insanları analiz edersek sonuç obez ya da normal kilolu çıkabilir. Lojistik regresyon insanların obez olma ihtimalini düşünür. Eğer vücut yağ oranı yüzde otuzun üstündeyse kişi obez kabul edilir, değilse normal kilolu. Lojistik bağlanımın matematiksel formülü aşağıda gösterildiği gibidir.

$$\ln\left(\frac{y'}{(1-y')}\right) = b + w_1x_1 + w_2x_2 + \cdots + w_nx_n \quad (2.31)$$



Şekil 2.42: Lojistik bağlanım örneği **[36]**.

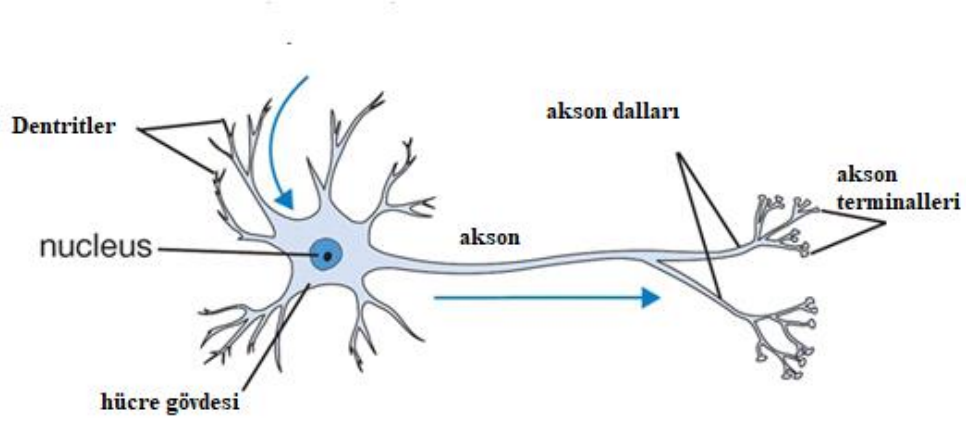
2.3.2.7. Yapay Sinir Ağları

Yapay sinir ağları (YSA) yöntemi insan beyni sinir sistemini örnek alarak modellenen öğrenme, tahmin yürütme, sonuca varma ve genelleme yapabilme gibi insana özgü yeteneklerin makinelere kazandırılmasını sağlayan bir araştırma konusudur. Sinir sisteminde bulunan nöronların çalışma prensibine atıf yaparak geliştirilen yapay sinir ağları veriyi işleyip bir sonraki nörona iletme ve bunları paralel şekilde yapabilme özellikleri ile oluşturdukları ağ yapısı ile insan beynini makine üzerinde gerçekleştirmeyi hedeflemektedir. YSA'nın biyolojik sinir sisteminde karşılık geldiği elemanları şöyle gösterebilir.

Tablo 2.12: Öğrenme işlemini gerçekleştiren biyolojik birimler ve YSA'daki karşılıkları

Biyolojik Sinir hücresi	**Yapay sinir hücresi**
Sinaps	Ağırlık
Dentrit	Toplam Fonksiyonu
Çekirdek	Aktivasyon Fonksiyonu
Akson	Çıktı

Sinaps, iki nöron arasındaki bağlantıyı kurup, aksondan aldığı bilgiyi diğer nöronun dendritine ulaşmasını sağlar. Sinapslar öğrenme işleminin gerçekleştiği kısımdır. Dentrit, nöronun uç kısmında bulunmaktadır. Diğer nöronlardan bilgi almayı sağlamaktadır. Çekirdek, dentritlerden aldığı bilgiyi aksona iletmektedir. Akson, çekirdekten iletilen veriyi çıktı olarak yeni bir nöron hücresine iletmektedir.



Şekil 2.23: Nöron hücre yapısı **[37]**.

Yapay sinir ağları beş temel elemandan meydana gelmektedir. Bunlar sırası ile girdi, ağırlık, toplamama fonksiyonu, aktivasyon fonksiyonu ve en son çıktıdır. Yakarıdaki şekilde gösterildiği gibi her bir girdi kendisine karşılık gelen ağırlık ile çarpılır. Daha sonra eşik değeri b eklenerek toplanır. Toplamı fonksiyonun kullanıcının seçtiği fonksiyona göre değişiklik gösterebilir (toplam, çarpım, maksimum, minimum vb. fonksiyonlar gibi). En son aktivasyon fonksiyonuna iletilen veri çıktı değerini alır. Yapay sinir ağları katmalarına göre tek ve çok katmanlı modeller olarak ikiye ayrılır.

Tek katmanlı Sinir ağları, sadece girdi ve çıktı katmanından oluşmaktadır. En basit ağ yapısıdır. Tüm girdiler çıktı katmanında birleşmektedir. 1960 yılında Perceptron modeli olarak geçen tek katmanlı ağda aktivasyon fonksiyonu toplam fonksiyonundan çıkan değerin kullanıcı tarafından belirlenen eşik değerinden büyük olması durumunda 1, küçük olması durumunda 0 sonuç vermektedir. Bunun sebebi aktivasyon fonksiyonu olarak basamak fonksiyonu kullanmasıdır. Diğer bir ifade ile Perceptron modeli ikili sınıflandırmada kesin evet ve hayır durumlarını temsil etmektedir. Bu model hata üzerine öğrenmeye kurulmuştur. Model hata ile karşılaştıkça öğrenip hata oranını azaltmayı gerçekleştirmiştir. Perceptron modelinde kullanılan öğrenme yöntemi Hebb yöntemidir. Bu algoritmaya göre çıkış değerlerinde elde edilen yanlış 0 ve 1 durumlarına göre ağırlık güncellemeleri aşağıdaki formüllere göre yapılmaktadır.

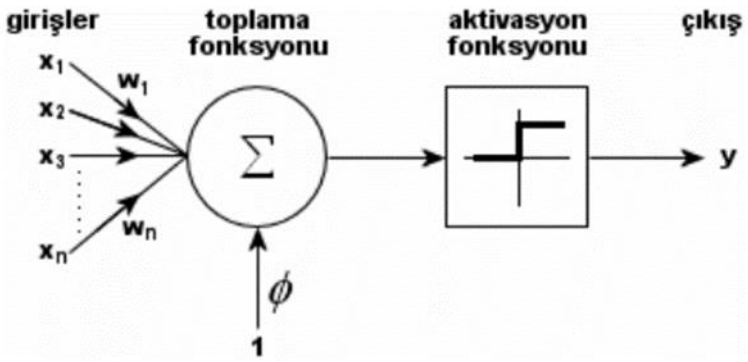
$$Yanlış\ çıkan\ 1\ durumu\ için: \quad w_{n+1} = w_n - \epsilon . x_i \qquad (2.32)$$

$$Yanlış\ çıkan\ 0\ durumu\ için: \quad w_{n+1} = w_n + \epsilon . x_i \qquad (2.33)$$

$$\emptyset_{n+1} = \emptyset_n \mp \epsilon \qquad (2.34)$$

ϵ : Öğ$renme\ katsayısı$

$\emptyset$: *Eşik değeri*



Şekil 2.24: Perceptron modeli **[38]**.

Diğer bir tek katmanlı algılayıcıya sahip olan model ise Adeline modelidir. Adaptive Linear Element kelimelerinin kısaltılması olan Adeline modelinde, klasik Perceptron modelinde kullanılan Hebb öğrenme yönteminin aksine Delta adı verilen farklı bir yöntem kullanılmaktadır. Gradyan azaltma yöntemi olarak da bilinen Delta öğrenme yönteminde, lineer aktivasyon fonksiyonu sonucu elde edilen çıkış değerinin başlangıçta elde edilmesi planlanan değerle olan farkının, belirlenen bir öğrenme katsayısıyla olan çarpımıyla beraber ağ ağırlıklarının güncellenmesi böylelikle öğrenme işleminin gerçekleştirilmesi anlamına gelmektedir. Delta öğrenme yöntemini matematiksel olarak açıklanmak istersek aşağıdaki sonuçlar elde edilmektedir.

$$Toplam\ Giriş\ değeri = NET = (\sum_{i=1}^{n} w_i . x_i) + \emptyset \quad (2.35)$$

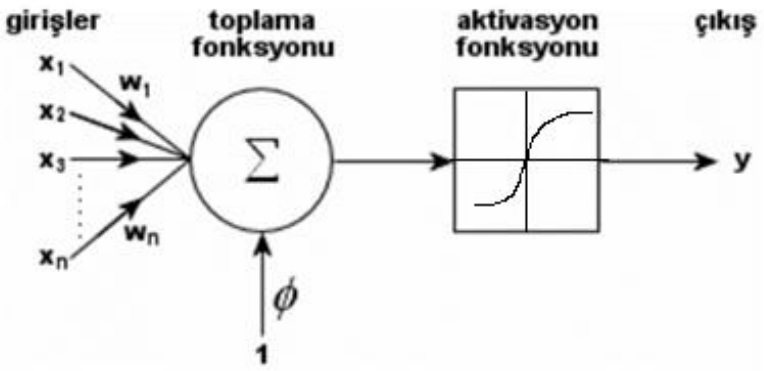
$$f(NET) = \frac{1}{1 + e^{-\beta . NET}}$$

$$Hedeflenen\ değer\ ile\ gerçek\ değer\ arasındaki\ fark = \gamma = y^* - y \quad (2.36)$$

$$\Delta w_i = \epsilon . \gamma . x_i \quad (2.37)$$

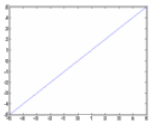
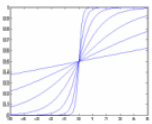
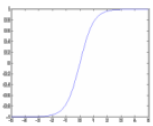
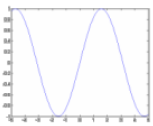
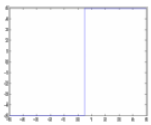
$$w_i(k+1) = w_i(k) + \epsilon . \gamma . x_i \quad (2.38)$$

Perceptron modelinde aktivasyon fonksiyonu olarak kullanılan basamak fonksiyonuna ek olarak Adeline modelinde sigmoid fonksiyonu kullanılmaktadır.



Şekil 2.25: Adeline modeli **[38]**.

Modelden istenilecek görevlere göre Adeline ve klasik Perceptron modelleri kullanılabilmektedir. Aktivasyon fonksiyonu olarak basamak fonksiyonu kullanan Perceptron modeli ikili sınıflandırma problemlerinde daha yüksek başarı gösterirken, sigmoidal aktivasyon fonksiyonu kullanan Adeline modeli 0 ve 1 arasında sürekli bir değer üretebildiği için sonucun hedef değerine lineer olarak ne kadar uzak veya yakın olduğunu gösterebilmektedir. Aşağıdaki görselde tek katmanlı algılayıcılarda kullanılan aktivasyon fonksiyonları verilmiştir.

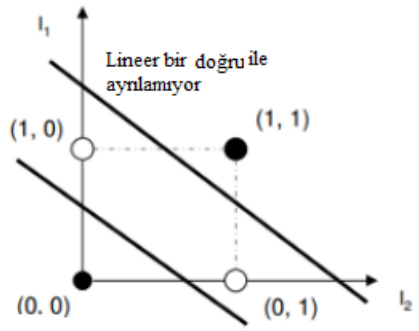
Fonksyonun Adı	Fonksyonun Şekli	Matematiksel İfadesi	Açıklama
Lineer Fonksyon		$F(NET) = NET$	Hesaplanan net giriş değeri, hücrenin çıkışı olarak kabul edilir
Sigmoid Fonksyonu		$F(NET) = \frac{1}{1+e^{-\beta NET}}$	β değeri keyfi şekilde değiştirilerek farklı eğimlere sahip eğriler elde edilebilir
Hiperbolik Tanjant Fonksyonu		$F(NET) = \frac{e^{NET}+e^{-NET}}{e^{NET}-e^{-NET}}$	Hesaplanan net giriş değeri, tanjant fonksyonuna uygulanır
Sinüs Fonksyonu		$F(NET) = Sin(NET)$	Öğrenilmesi istenen olayın, sinüs fonksyonuna uygun dağılım gösterdiği durumlarda kullanılır
Basamak Fonksyon		$F(NET) = \begin{cases} 1 & NET > k \\ 0 & NET \le k \end{cases}$	k, eşik değeri göstermektedir. Hesaplanan değerin, eşik değerden büyük yada küçük olmasına göre 1 yada 0 çıkışları üretilir

Şekil 2.265: YSA'da kullanılan aktivasyon fonksiyonları **[38]**.

Tek katmanlı algılayıcıya sahip modeller doğrusal olmayan problemlerin çözümünde yetersiz kaldığı ve sadece iki sınıflı sonuçlar verdiği için çok katmalı modeller konusu incelenme başlanmıştır.

Çok katmanlı sinir ağları, giriş ve çıkış katmanlarına ek olarak gizli katmanlar içermektedir. Giriş katmanlarınca alınan bilgi hiçbir işlem görmeden gizli katmana aktarılmaktadır. Gerçek hayatta gerçekleşen olayların doğrusal olmamasından yani bir grafik üzerinde gösterildiğinde sonuçlarının bir doğru ile ayırılmamasından dolayı tek katmanları ağlarca çözülemeyen problemler çok katmanlı sinir ağlarına konu olmuştur. Bu bağlamda konu olan ilk problem XOR problemidir. XOR mantıksal bir operatördür. Girdi olarak verilen değerler aynıysa 0, farklıysa 1 değeri üretmektedir. Oluşan 0 ve 1'lerin lineer bir doğru ile ayrılamamasından dolayı XOR çok katmanlı ağların başarılı ile gerçekleştirdiği ilk problemdir.

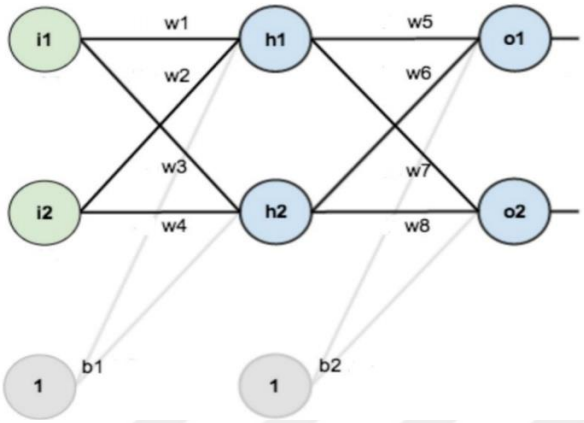
XOR		
I_1	I_2	Çıkış
0	0	0
0	1	1
1	0	1
1	1	0



Şekil 2.27: XOR mantıksal ifadesi gösterimi **[39]**.

Çok katmanlı ağlar güncel derin öğrenme algoritmalarının temelini oluşturmaktadır. Girdi katmanı, kullanıcının belirlediği sayıdaki ara katman ve çıktı katmanından oluşan bir ÇKA modeline, modeli eğitmek için çözümlenmesi istenen sonuç ile alakalı girdiler ve bu girdilerin üretmesi hedeflenen çıktılar sunulur. Her eğitim epoğu sonucu, ağ kendi ürettiği sonuç ile hedeflenen sonuç arasındaki farkı alır ve başlangıçta rastgele belirlenen katman ağırlıklarını güncellemeye başlar. Buna hatayı geriye yayma (backpropogation) denir. Diğer sistemlerde olduğu gibi amacı hatayı minimize etmektir. Çıkış katmanında yer alan her hücre için hata ayrı ayrı hesaplanır. Her çıkış noktası için hesaplanan hatalar toplanarak toplam hata bulunur. Buraya kadar olan kısın ileri besleme (ileri yayılım) olarak adlandırılır. İleri besleme geriye yayılımın bir parçasıdır. Geriye yayılım ileri besleme bitip çıkış katmanındaki her nodu için hata hesaplandıktan sonra başlar. Her epok sonucu elde edilen loss function yani elde edilen değer ile beklenen değer arasındaki farkı hesaplayıp ağırlıklarını güncelleyen model, ağın

başarı yüzdesini yükselterek sistemin öğrenmesini geliştirmesine ve genelleme yeteneğini kazabilmesini sağlamaktadır. İleri ve geri yayılmaya matematiksel olarak şu şekilde açıklanabilir



Şekil 2.28: Örnek YSA yapısı **[40].**

$$h_1 = w_1 * i_1 + w_2 * i_2 + b_1 * 1 \quad (2.39)$$

$$h_2 = w_3 * i_1 + w_4 * i_2 + b_1 * 1 \quad (2.40)$$

$$Out_{h1} = {}^{1}/_{(1+e^{-h1})} \quad (2.41)$$

$$Out_{h2} = {}^{1}/_{(1+e^{-h2})} \quad (2.42)$$

$$O_1 = w_5 * Out_{h1} + w_6 * Out_{h2} + b_2 * 1 \quad (2.43)$$

$$O_2 = w_7 * Out_{h1} + w_8 * Out_{h2} + b_2 * 1 \quad (2.44)$$

$$Out_{o1} = {}^{1}/_{(1+e^{-O1})} \quad (2.45)$$

$$Out_{o2} = {}^{1}/_{(1+e^{-O2})} \quad (2.46)$$

Çıktılar üsteki formülde görüldüğü üzere hesaplanmaktadır. Buraya kadar olan kısım ileri yayılım olarak adlandırılmaktadır. Bundan sonra ağ kayıp fonksiyonunu hesaplamalıdır.

Kayıp fonksiyonu aşağıda verilen formüllere göre hesaplanmaktadır. Formüllerde geçen $\hat{y}$ hedef değeri temsil etmektedir. E_T ileri yönlü beslemenin toplam kaybıdır.

$$E_{O1} = \frac{1}{2} * (\hat{y} - Out_{O1})^2 \quad (2.47)$$

$$E_{O2} = \frac{1}{2} * (\hat{y} - Out_{O2})^2 \quad (2.48)$$

$$E_T = E_{O1} + E_{O2} \quad (2.49)$$

Toplam kayıp fonksiyonu hesaplandıktan sonra ağın ağırlıklarının güncellenmesi için geri yönlü besleme yapılmalıdır. Bunun için ağdaki herhangi bir düğümü referans alarak (w_5 kabul edilmiştir) aşağıda gösterilen işlemler uygulanmaktadır. Ağın ağırlıklarının optimize edilmesi ile modelin doğruluk başarımı arttırılarak modele genelleme yapabilme yeteneği kazandırılmaktadır.

$$\frac{dE_T}{dw_5} = \frac{dE_T}{dOut_{01}} * \frac{dOut_{O1}}{dO_1} * \frac{dO_1}{dw_5} \quad (2.50)$$

$$w_5^+ = w_5 - n * \left(\frac{dE_T}{dw_5}\right) \quad (2.51)$$

w_5^+: w_5 düğümünün güncellenmiş ağırlığı

Öğrenme, tahmin etme ve eski bilgilerine dayanarak genelleme yapabilme özelliği sayesinde klasik programlama yöntemlerinde ayrılan YSA, yeni nesil derin öğrenme yöntemlerinin temelini oluşturuyor olup, gelişen bilgisayar donanımları ile teknolojiden, finansa, jeofizikten, sağlık bilimlerine, pazarlamaya çoğu alanda kullanılmaktadır. Doğrusal olmayan problemlerin çözümü için yapılan modellemelerin artması ile kullanımı artan yapay sinir ağları, lineer olan veya olmayan sistemlerin incelenmesi, görüntü işleme, ses ya da konuşma tanıma gibi mühendislik tabanlı konuların çözüme ulaşmasında önem belirtmektedir

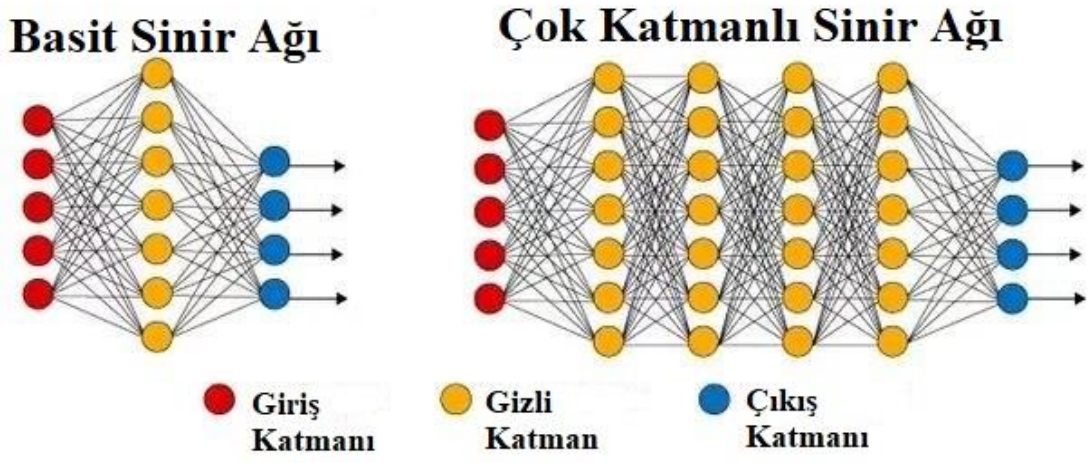
3. MATERYAL VE YÖNTEM

3.1. DERİN ÖĞRENME

Derin öğrenme, gelişen teknoloji ile büyüyen veri miktarı ve doğru orantılı olarak ilerleyen teknolojik donanımların ortaya çıkardığı sorunların, makine öğrenmesi algoritmalarının çözümlemede yavaş ve yetersiz kalmalarından dolayı ortaya çıkmıştır. İnternet üzerinden kullanılan sosyal ağlar ve diğer ortamlarda kullanılan, çoğunluğu görüntü olmak üzere oraya çıkan milyarlarca veri ile makine öğrenmesi yöntemlerinin yetersiz kalması derin öğrenmenin son yıllarda en çok kullanılan yapay zekâ alt dalı olmasının sebeplerindendir.

Makine öğrenmesi algoritmalarının daha çok matematik ve istatistik tabanlı olmalarına karşın derin öğrenmenin programlamaya daha uygun olması problemlerin çözümünde büyük bir avantaj sağlamaktadır.

Derin öğrenme, önceki bölümlerde açıklanan yapay sinir ağları mantığına dayanmaktadır. Çok sayıda gizli katmanda art arda işlenen ve bir gizli katman çıktısının kendinden bir sonraki gizli katmana girdi olarak iletilmesi ile çıkış katmanında sonuç elde edilen Çoklu Katman Algılayıcısı (Multi Layer Perceptron, MLP), derin öğrenmenin çalışma prensibidir.



Şekil 3.1: Basit sinir ağı ve çok katmanlı sinir ağı yapıları **[41]**.

Derin öğrenme yönteminde girdi olarak verilen verinin katmanlar arasında baştan sona doğru, daha temel sayılabilecek özelliklerden daha detay kabul edilebilecek özelliklere kadar öğrenilmektedir. Katmalar arası oluşan bu hiyerarşik öğrenmeden dolayı derin adını alan derin öğrenmede, derinlik sayısı oluşturulan katman sayısına bağlıdır.

Bir derin öğrenme modelinde yüksek bir başarım elde etmek için, tıpkı bir makine öğrenmesi modelinde olduğu gibi yapılması gereken bazı işlemler ve dikkat edilmesi gereken önemli noktalar bulunmaktadır. Bunlar modelin aşırı ya da yetersiz uydurmasının önüne geçmek, hiper parametrelerin en iyi şekilde ayarlanmasını sağlamak, uygun en iyileyici seçimi, verinin ihtiyaç duyulması durumunda veri seti çeşitlendirilmesi yapmak ve doğrulama yöntemini uygun şekilde seçmek olarak sıralandırılabilmektedir.

Hiper parametreler, tipik bir makine öğrenmesi algoritması için düşünüldüğünde yığın büyüklüğü (batch_size), kullanılan eniyileyici türü ve öğrenme oranı (learning rate) olarak sıralanabilir. Çoklu katman yapısına sahip Derin öğrenme için ise bunlara ek olarak kullanılacak gizli katman sayısı, bu katmanların sahip olduğu aktivasyon fonksiyonları da hiper parametre ayarında önem taşımaktadır. Hiper parametreler, amacı kayıp fonksiyonunu küçültmek olan, modelin daha iyi bir performans gösterebilmesini, daha önce görmediği verilerde modelin genelleme yapabilip iyi sonuçlar vermesini hedefleyen birimlerdir.

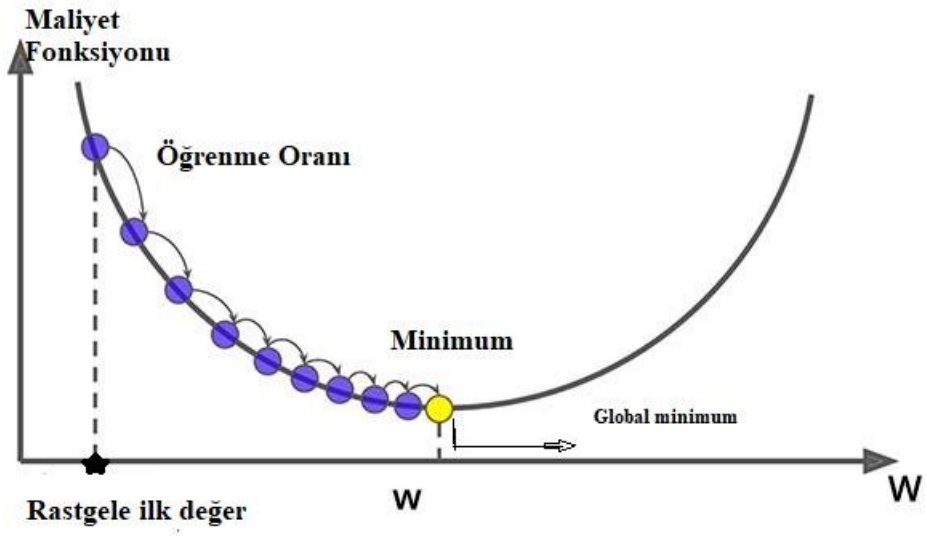
Hiper parametreleri ayarlamanın önemini gösterebilmek için örnek olarak Gradyan İniş (Gradient Descent) yöntemi gösterilebilmektedir. Amacı kayıp fonksiyonunu mümkün olan en küçük değerine indirmek olan eniyileyicilerden en çok kullanılan makine öğrenmesi tekniği olan Gradyan iniş yönteminde başlangıçta rastgele seçilen model parametreleri, her iterasyon ile birlikte güncellenmekte ve kayıp fonksiyonu (maliyet fonksiyonu) global minimuma yani mümkün olan en düşük değerine indirilmeye çalışılmaktadır. Bu yöntemde her iterasyonda kendinden bir sonraki iterasyondaki ağırlık değerini bulabilmek için o noktadaki türev (eğim) değerini öğretim oranı denilen hiper parametre ile çarpılıp güncel ağırlık değerinden çıkarılmaktadır. Gradyan iniş formülü aşağıdaki denklemde gösterildiği gibidir

$$\mathrm{W} = \mathrm{W} - \text{Learning rate} * \mathrm{dW} \qquad (3.1)$$

$W = Ağırlık\ değeri$

$dW = Ağırlığın\ türevi$

Gradyan iniş yöntemini bir sıradağ sisteminin en tepesinden bırakılan kayanın, dağın en alçak kısmına varması olarak düşünülebilmektedir. Burada en alçak kısım global minimum noktasını temsil etmektedir. Öğrenme oranı ise kayanın yaptığı salınımlar olarak düşünülebilir. Yüksekten bırakılan kayanın global minimumu bulmaya çalışırken yerel minimum denilen kısımda kalması ise istenmeyecek bir durumdur. Gradyan iniş yönteminin çalışma prensibi aşağıdaki görselde gösterildiği gibidir.



Şekil 3.2: Gradyan inişi yöntemi **[42]** .

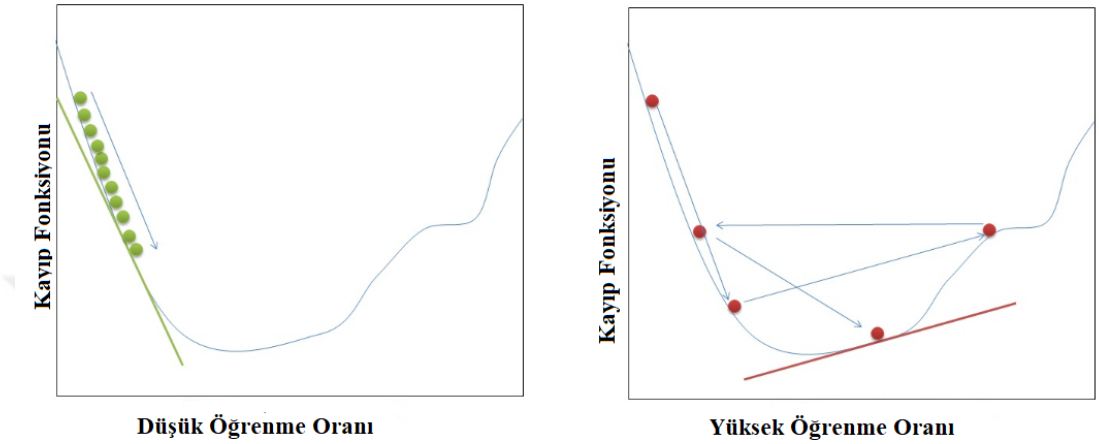
Gerçekleştirilen tez çalışmasında kullanılmış olan hiper parametreler, diğer seçenekler ile, makinalarda öğrenme mantığını daha iyi belirtebilmek ileriki bölümlerde açıklanacaktır.

3.1.1. Öğrenme Oranı

Öğrenme oranı hiper parametre ayarında kullanılan birimlerden biridir. Global minimuma ulaşmaya çalışan modelin adım aralığını belirlemektedir. Öğrenme oranı düşük seçilen modellerde kayıp fonksiyonunun minimuma değerine ulaşma süresi uzun olacak ve model daha çok iterasyon yapmak zorunda kalacaktır. Buna karşın katman ağırlıkları daha tutarlı şekilde değişmekte ve elde edilen model daha başarılı olacaktır.

Öğrenme oranı yüksek seçilen ağlarda değişen ağırlık değerleri arasındaki fark çok fazladır. Bu yüzden kayıp fonksiyonu global minimum noktasına ulaşamamaktadır lakin modelin eğitim süresi çok kısa olmaktadır. Kayıp fonksiyonu belirleme konusunda yüksek öğrenme oranı dezavantajdır.

Yukarıda geçen sebeplerden dolayı, öğrenme oranını belirlerken kayıp fonksiyonu minimum yapılmaya çalışılırken diğer bir yandan da modelin eğitim süresi de göz ardı edilmemelidir. Bu yüzden düşük ve yüksek öğrenme oranları arasında optimal bir öğrenme oranı seçmek model başarımı konusunda önem taşımaktadır. Öğrenme oranının kayıp fonksiyonunu bulmada etkisi aşağıdaki görsellerde belirtilmiştir.



Şekil 3.3: Öğrenme oranı seçiminin kayıp fonksiyonuna etkisi **[43]** .

3.1.2. En iyileyici Seçimi

Optimizer (yani En iyileyici) seçimi derin öğrenmede öğrenme işlemini gerçekleştiren kısımdır. Ağın her epok sonrası katmanların ağırlıklarının kayıp fonksiyonuna göre güncellenmesini ve modelin başarımının artmasını amaçlamaktadır. Eğitim oranı ve katman ağırlıkları ile beraber hiper parametre ayarını oluşturmaktadırlar. Model oluşturulurken hangi eniyileyicinin daha uygun olduğunu bulmak için tüm seçeneklerin denenmesi küçük veri setleri için uygun olsa da yüz binlerce verinin bulunduğu ve eğitim süresinin günler sürdüğü ağlarda bu yöntem kullanıcılarına dezavantaj sağlamamaktadır. En çok kullanılan eniyileyiciler Gradyan tabanlı eniyileyiciler (Stokastik Gradyan İnişi, Gradyan İniş, Momentum ile Stokastik Gradyan iniş), AdaGrad, RMSProp, AdaDelta ve bu çalışmada kullanılmış olan Adam eniyileyicileridir.

Önceki bölümlerde temel çalışma prensibi gösterilen Gradyan iniş yönteminin kendi içinde ayrıldığı türleri bulunmaktadır. Bunlardan ilki olan Stokastik Gradyan İnişi (Stochastik Gradient Descent, SGD) her iterasyon adımında sadece bir adet rastgele seçilen örnek üzerinden ağ parametrelerini yani katman ağırlıklarını güncellemektedir. Adında yer alan Stokastik yani rastgele kavramı kullandığı örnekleri rastgele seçmesinden kaynaklanmaktadır. Normal Gradyan iniş yönteminde tek seferde tüm veri ile çalışan eniyileme yöntemine oranla daha fazla gürültü ortaya çıkarmasına rağmen yüksek miktarda verilerin bulunduğu durumlarda diğer

gradyan yöntemlerine göre kullanımı daha avantajlı olmaktadır. Çünkü her iterasyonda tüm veri ile değil sadece seçilen rastgele veri üzerinden çalışmaktadır. Bu da modelin eğitim süresini çok büyük ölçüde kısaltmaktadır.

Momentum ile SGD, Stokastik Gradyan inişin minimum kayıp değerine ulaşmaya çalışırken ortaya çıkardığı gürültü değerini azaltan bir yöntemdir. Momentum eklenmiş Gradyan iniş algoritmasının minimum değeri daha hızlı yakınsamasını, gradyan salınımlarının daha büyük olmasını sağlamaktadır. Momentum eklenmiş Gradyan iniş formülü aşağıdaki denklemde verildiği gibidir.

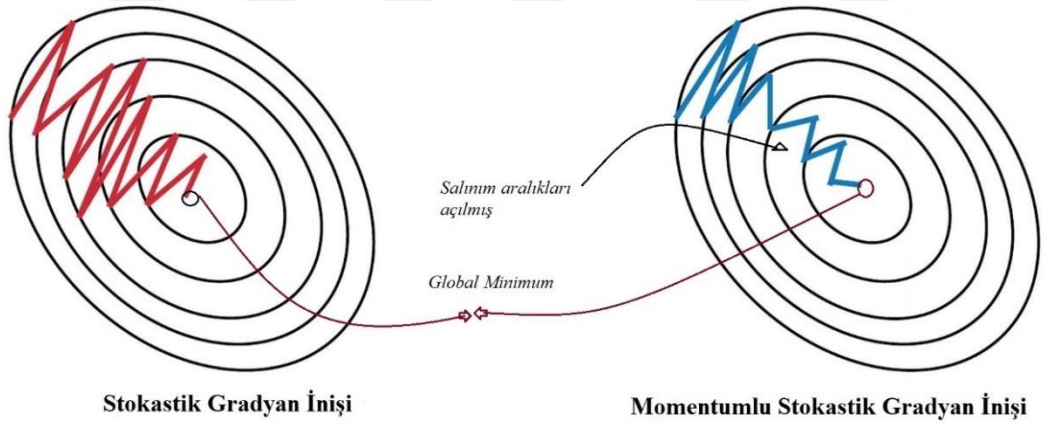
$$V_{dW} = \beta * V_{dW} + (1 - \beta) * V_{dW} \quad (3.2)$$

$$V_{db} = \beta * V_{db} + (1 - \beta) * V_{db} \quad (3.3)$$

$$W = W - Learning\ Rate * V_{dW} \quad (3.4)$$

$$b = b - Learning\ Rate * V_{db} \quad (3.5)$$

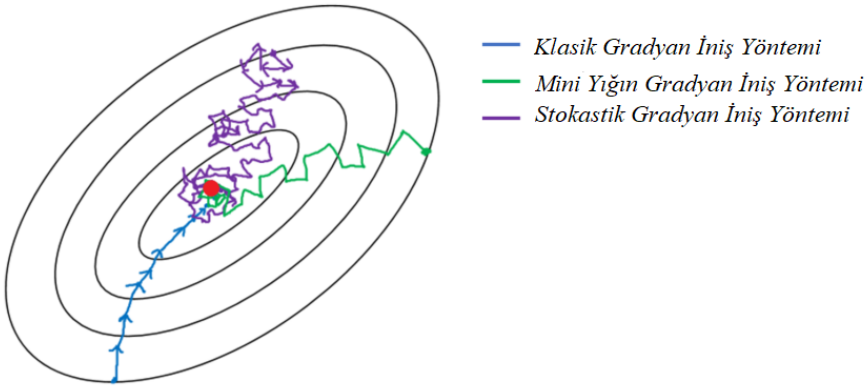
$$\beta = Momentum\ .0\ ile\ 1\ arasında\ değer\ alabilmektedir.$$



Şekil 3.4: SGD ve momentumlu SGD **[44]**.

Stokastik gradyan inişi ve momentum eklenmiş SGD arasındaki salınım farklarını gösteren görsel yukarıda gösterildiği gibidir. Diğer bir gradyan iniş yöntemi olan Mini Yığın Gradyan iniş yöntemi (Mini Batch Gradient Descent), klasik gradyan yöntemi ve Stokastik Gradyen yöntemi arasında yer alan iki yönteminde avantajlarını alıp optimal bir eniyileyici çıkarmaya çalışmaktadır. Ne birim zamanda bir örnek almakta ne de tüm veri setini aynı anda taramaktadır. Bunun yerine eğitim setini mini yığınlara bölerek her iterasyonda bu yığınlar üzerinde

çalışmaktadır. Kendi içinde "batch_size" denilen bir parametre bulundurmakta ve bu parametre 32,64,128… şeklinde olmaktadır. Mini yığın gradyan iniş yöntemi diğer makine öğrenmesi yöntemlerinin yanında, derin öğrenmede de kullanılmaktadır.



Şekil 3.5: Gradyan iniş yöntemlerinin kıyaslanması **[45]**.

AdaGrad diğer adıyla Uyarlanabilir Gradyan algoritması yöntemi, derin öğrenmede sıklıkla kullanılan eniyileyicilerden biridir. AdaGrad algoritması Stokastik gradyan iniş yönteminde ortaya çıkan gürültüyü ve rastgele seçilen örneklerden kaynaklanan geniş salınımları, öğrenim oranındaki değişimlerle azaltmaya çalışmaktadır. Her gradyan için sabit olan öğrenme oranını yani adım aralığı değerini değişken kabul etmektedir.

$$\text{SGD} \rightarrow \ W_t = W_{t-1} - \ \mu * \left(\frac{\partial L}{\partial W_{t-1}}\right) \quad (3.6)$$

$$\text{AdaGrad} \rightarrow \ W_t = W_{t-1} - \mu_t{}' * \left(\frac{\partial L}{\partial W_{t-1}}\right) \quad (3.7)$$

$$\mu_t' = \mu \Big/ \sqrt{a_t + \varepsilon} \quad (3.8)$$

$$\varepsilon = Sabit\ değer$$

$$a_t = \sum_1^t \left(\frac{\partial L}{\partial W_{t-1}}\right)^2 \quad (3.9)$$

$$a_t = Gemiş\ gradyanların\ karelerinin\ kümülatif\ toplamı$$

RMSProp algoritması AdaGrad yöntemiyle benzer şekilde çalışan bir algoritmadır. Değişmeyen öğrenim oranları dezavantajından kurtulmak için geçmiş momentumlu gradyenlerin karelerini almaktadır.

$$w_{t+1} = w_t - {a}/{\sqrt{S_t + \varepsilon}} * (\frac{\partial L}{\partial w_t})^2 \quad (3.10)$$

$$S_t = \beta * S_{t-1} + (1 - \beta) * (\frac{\partial L}{\partial w_t})^2 \quad (3.11)$$

Diğer bir SGD algoritması türü olan AdaDelta algoritması, aynı RMSProp algoritması gibi AdaGrad'in daha iyileştirilmiş versiyonudur. Diğer gradyan yöntemlerinde olduğu gibi ağırlıkların güncellemesini yaparken öğrenme oranından faydalanmamaktadır. Bunun yerine delta olarak belirlediği bir parametrenin karesinin momentum eklenmiş halini kullanmaktadır [46].

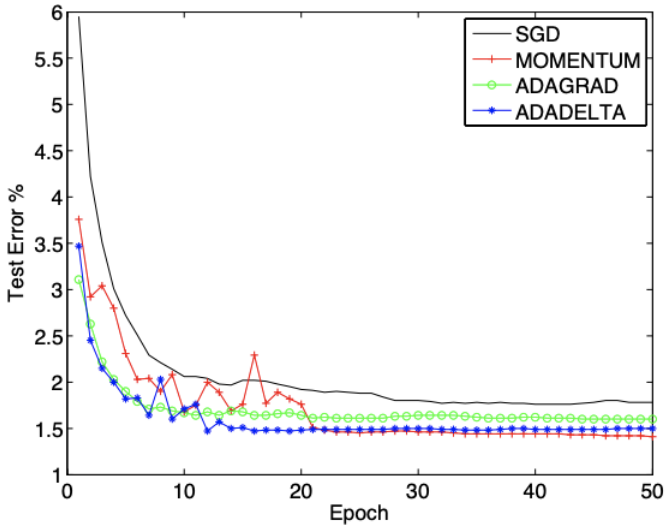
$$w_{t+1} = w_t - \frac{\sqrt{D_{t-1} - 1}}{\sqrt{S_t + \varepsilon}} * \frac{\partial L}{\partial w_t} \quad (3.12)$$

$$D_t = \beta * D_{t-1} + (1 - \beta) * (\Delta w_{t)}{}^2 \quad (3.13)$$

$$S_t = \beta * S_{t-1} + (1 - \beta) * (\frac{\partial L}{\partial w_t})^2 \quad (3.14)$$

$$\Delta w_t = w_t - w_{t-1} \quad (3.15)$$

SGD, Momentum, AdaGrad, AdaDelta en iyleyicilerinin performanslarını kıyaslamak için yapılan ve MNIST veri setinin kullanıldığı çalışmada ortaya çıkan sonuçlar alttaki görselde verilmiştir [47].



Şekil 3.6: Matthew Zeiler'in "Adadelta: an adaptive learning rate method" adlı çalışmasından bir en iyileyicilerin performansını gösteren bir grafik **[48]**.

Yukarıda gösterilen eniyileyicilerin dışında, en güncel ve çok kullanılan eniyileyici olan aynı zamanda bu tez çalışmasında da yer almış olan, Adam eniyileyicisidir. Adam algoritması RMSProp ve Momentum eniyileyicilerinin avantajlarının bir araya gelmesiyle ortaya çıkmıştır. Özellikle büyük veri setleriyle çalışıldığında hızdan ve hafızadan kazanç sağlanmasını sağlayan Adam algoritması, beta1 ve beta2 adlı iki parametre ile eniyileme işlemini gerçekleştirmektedir. Yapılan tez çalışmasında adam eniyileyicisinde öğrenme oranı lr=0,001 ve parametreler beta1=0,9, beta2=0,999 olarak belirlenmiştir.

$$w_{t+1} = w_t - \frac{a}{\sqrt{\widehat{S}_t + \varepsilon}} * \widehat{V}_t \quad (3.16)$$

$$\widehat{V}_t = \frac{V_t}{1-\beta_1^t} \quad (3.17)$$

$$\widehat{S}_t = \frac{S_t}{1-\beta_2^t} \quad (3.18)$$

$$V_t = \beta_1 * V_{t-1} + (1-\beta_1) * \frac{\partial L}{\partial w_t} \quad (3.19)$$

$$S_t = \beta_2 * S_{t-1} + (1-\beta_2) * (\frac{\partial L}{\partial w_t})^2 \quad (3.20)$$

3.1.3. Aşırı Uydurma ve Yetersiz Uydurma

Aşırı uydurma (overfitting) ve yetersiz uydurma (underfitting) problemleri, makine öğrenmesi modellerinde en istenmeyen problemlerdir. Doğrudan model başarımını etkilemektedirler. Eğitim veri setiyle eğitim gören modelin, yeterli veri göremediğinde öğrenme işlemini başarıyla

gerçekleştirememesi veya modelin eğitim esnasında ezber yapmasından kaynaklanmaktadır. İlkine yetersiz öğrenme ikincisine aşırı uydurma denilmektedir. Bu iki sorunda, test verilerinde iyi bir sorun alınamamasına neden olmaktadır.

Aşırı uydurma problemi, eğitim esnasında başarım metriğinin maksimum değere, kayıp metriğinin minimum değere ulaştığı epok değerinden sonra, modelin değerlendirme metriklerinin ters yöne ilerlemesi olarak görülmektedir. Aşırı uydurma problemi gelişen teknoloji ve büyüyen veri miktarları ile, makine öğrenmesi ve derin öğrenme de en çok görülen genelleştirme problemidir. Genelleştirme modelin ilk kez gördüğü veriler üzerinde de görevini başarı ile yerine getirebilmesidir.

Derin öğrenme problemlerinde, aşırı uydurma sorunun üstesinden gelmek için yapılan ilk önlem ağı küçültmektir. Ağdan kasıt ileriki bölümde görülecek olan konvolüsyon, biriktirme ve tamamen bağlı katmanlar bütünüdür. Küçültülen ağ ile her katman çıkışı oluşacak olan parametre sayısını düşürmek hedeflenmektedir. Diğer bir yöntem katmanlar arasına iletim sönümü yani gürültü eklemektir. Bu sayede her katman çıkışında öğrenilen bilginin bir kısmı yok olacak ve model ezber yapamayacaktır. Diğer bir yöntem ise katman ağırlıklarının düzenlileştirilmesidir. Bu yöntemdeki amaç her epok sonrası düzenlenen katman ağırlıklarının büyük değişimler almasının önüne geçmektir.

Yetersiz öğrenme ise eğitim esnasında modelin genelleştirme yapabilecek yeteneği kazanmayacak kadar veriyle eğitim görmesinden kaynaklanmaktadır. Verinin yeterli olmadığı böyle durumlarda veri çeşitlendirme (data agumentention) ile, elde bulunan verilerin üstünde oynanıp (boyutunun renginin değiştirilmesi, görüntünün kaydırılması vb.) yeni ve daha büyük bir veri seti elde edilmiş olur. Böylelikle model genelleştirme yapabilecek kadar verile eğitim görmüş olur.

3.2. EVRİŞİMLİ SİNİR AĞLARI

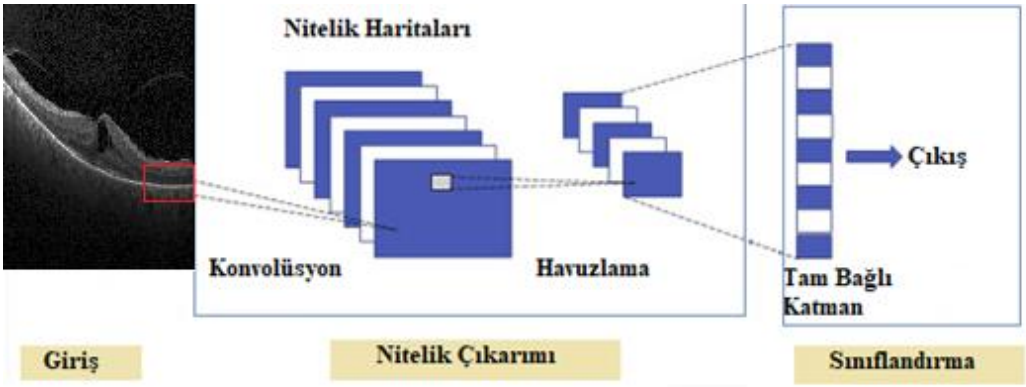
Evrişimli sinir ağları (Convolutional neural networks) yapay sinir ağları sisteminden geliştirilen, özel bir ileri beselemeli sinir ağları türüdür. Güncellenebilen nöron ağırlıkları ve eşik değerlerine sahip olmaları evrişimli sinir ağlarını, yapay sinir ağlarından türemelerine neden olmaktadır. İki ağ sistemini birbirinden ayıran fark ise, ESA mimarisinin kullanılmasına izin verdiği giriş veri tipinin görsel verileri olmasıdır.

Evrişimli sinir ağları (ESA) sayesinde, ilerleyen teknolojiyle paralel olarak büyüyen veri miktarları ile bu verileri işleyebilmenin önemi ve kullanımı artmıştır. Karşılaşılan veri her zaman vektör formunda değil imge formunda da olabileceği için çok katmanlı algılayıcılarla işlenememektedir. Yüksek performans gösterebilen GPU'lar, büyük boyutlardaki verilere denk olabilecek seviyeye ulaştığında evrişimli sinir ağları devreye girerek tam bağlı katmanların ya da çok katmalı algılayıcıların yetersiz kaldığı durumlarda kullanılmaya başlanmıştır. Evrişimli sinir ağları diğer çok katmalı algılayıcılar gibi önceki katmandan gelen çıktı değerini kendine girdi olarak atamaktadır.

Tam bağlı katmanların (fully connected layers) yetersiz kalmasının sebebi, verileri vektör formuna çevirip buradan öğrenme gerçekleştirirken, o verinin yanındaki ilişik veriden öğrenemeyip aradaki ilişkililiği kuramamasından kaynaklanmaktadır. Bunun aksine Evrişimli sinir ağlarında öğrenme hiyerarşik bir şekilde aşama aşama gerçekleşmektedir. Her evrişim katmanı ile önceden öğrenilen niteliklerin üzerine koyulup daha spesifik bilgiler çıkarılmakta böylelikle veriler arası (örüntüler arası) ilişki bulunmaktadır. Böylelikle ağ sistematik ve düzenli bir şekilde üst üste koyarak öğrenme işlemi gerçekleştirdiği için karmaşık verilerde daha iyi bir performans göstermektedir. Buna evrişimli sinir ağlarında öğrenme hiyerarşisi denilmektedir. Aynı zamanda evrişimli sinir ağlarında, katmanın bir görüntü verisinden öğrendiği bilgiyi (örüntüyü) başka bir görüntüde gördüğünde ya da aynı görüntüde farklı noktalarda tekrardan karşılaştığında yeniden öğrenmesine gerek kalmamaktadır. Bu da öğrenme işleminde kolaylık ve zamandan kazanıma neden olmaktadır. Tam bağlı katmanlarda ise ağ daha önce gördüğü bir örüntüyle tekrardan karşılaştığında o örüntüyü yeniden öğrenmek zorunda kalmaktadır. Bu tekrar tekrar öğrenme işlemi de ağa dolayısıyla bilgisayar işlemcisine yük olmaktadır.

Evrişimli sinir ağlarını, dolayısı ile derin öğrenmeyi diğer makine öğrenme yöntemlerinden ayıran özellik, öznitelik çıkarımının katmanlarda otomatik olarak kendi kendiliğinden gerçekleşmesidir. Evrişimli sinir ağları bu öznitelik çıkarımının gerçekleştiği katman evrişim katmanıdır. Buradan çıkan veriler tensör adını almaktadır. Daha sonra evrişim katmanlarından çıkan örüntüler sınıflandırma gibi modelin kuruluş amacı olan problemi gerçekleştirmek için tam bağlı katmana geçmelidir. Bunu gerçekleştirebilmek için, bu iki katman arasında geçişi sağlayan düzleştirme katmanı (Flatten) bulunmaktadır. Bir evrişim katmanı da kendi içinde art arda gelen evrişim ve havuzlama katmanlarından oluşmaktadır. Diğer bir ifade ile evrişimli

sinir ağları çok katmalı algılayıcılara birer örnektir Aşağıdaki görselde basit bir evrişimli sinir ağı modeli gösterilmiştir.



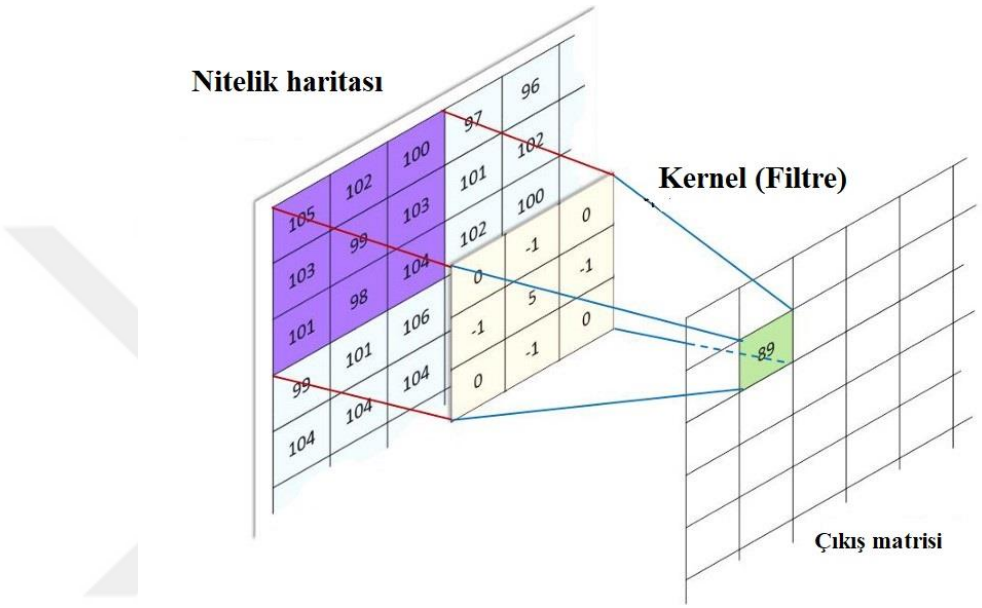
Şekil 3.7: Örnek CNN ağ yapısı **[8]**.

Görüntü verileri üzerinde işlem yapmak istediğimizde, bu görüntüler piksel değerlerinin oluşturduğu matrisler halinde gelmektedir. Derin öğrenme için konuşulduğunda bu matrisler üç boyutlu tensörler halindedir. Genişlikleri, yükseklikleri ve derinlikte denilebilen kanal sayıları vardır. Bu tez çalışmasında kullanılan retinal görüntü verileri 512x496 formunda bulunmaktadır. Adı geçen üç eksen birleşerek üç boyutlu tensör formunu almaktadır. Görseldeki kanal sayısı nitelik haritası dönüştürülen görüntü verinin renkli olup olmadığına göre değerler alabilmektedir. RGB formattaki bir veri için bu değer 3 olurken en çok kullanılan veri setlerinden biri olan ve Keras kütüphanesine gömülü şekilde gelen siyah beyaz yani gri seviyesindeki görsel verilerden oluşan MNIST veri seti için ise 1 olmaktadır. Görselden çıkarılacak üç boyutlu nitelik haritalarında 0 ile 255 değerleri arasında değerler yer almaktadır. Bunlar piksel yoğunluklarını göstermektedir.

3.2.1. Evrişim İşlemi

Evrişimli katmanlara adını veren evrişim işlemi aslında bir matematiksel işlemdir. Verilen iki fonksiyonun birbiri ile ne kadar örtüştüğünü belirlemektedir. Derin öğrenmede de ayı amaçla kullanılan evrişim işlemi girdi olarak verilen nitelik haritalarından filtreleri kullanarak çıktı nitelik haritaları oluşturmaktadır. Diğer adları Kernel olan filtreler evrişimli sinir ağlarında nitelik tanımlayıcılar olarak görev almaktadırlar. Her filtre matris formundaki nitelik haritasında belli bir özelliği aramaktadır. Bu özellikler, verilen görselde kırmızı bir rengin, bir hayvanın ya da bir yüz bilgisinin olup olmadığı gibi spesifik özellikler olabilmektedir. Filtrelerin kendi de matris formunda bulunmaktadır. Nitelik haritası boyunca kaydırılarak

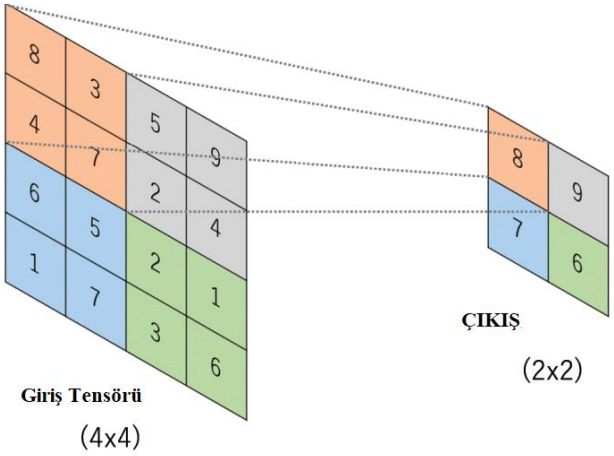
parçalar çıkarıp tek tek skaler değer üretmektedirler. Tüm nitelik haritası tarandığında ise, üretilen skaler değerler birleştirilerek tekrardan bir matris oluşmaktadır. Bu matris o giriş nitelik matrisinin çıktı nitelik matrisidir. Nitelik matrislerinden çıktı matrisi oluştururken, filtre ile karşılaştırılan matris parçasında elemanlar kendilerine karşılık gelen değer ile çarpılıp en sonunda bütün değerler toplanmaktır. Böylelikle her kaydırma adımında tek bir değer elde edilmektedir. Çıktı haritası filtrede yer alan özelliğin, farklı bölgelerdeki varlığını aramaktadır.



Şekil 3.8: Verilerden çıkarılan nitelik haritalarının filtre ile çarpılması **[49]**.

3.2.2. Havuzlama Katmanı

Havuzlama katmanı evrişim katmanlarından sonra yer almaktadır. Bunun sebebi evrişim katmanından çıkan verilerden seçici geçirgenlik yardımıyla boyut azaltmaktadır. Havuzlama katmanı iki farklı şekilde çalışmaktadır. Bunlardan ilki ortalama biriktirme, diğeri en iyileri biriktirme. Ortalama biriktirme, evrişimden çıkan nitelik haritasında pencereyi ilerleterek, bulunduğu andaki pencerenin elemanlarının ortalamasını alarak katman çıkışına vermektedir. En iyileri biriktirme yöntemi ise nitelik haritası üzerinde gezebilen pencerenin, o an gördüğü elemanlardan en büyüğünü çıktı olarak vermesidir. Havuzlama katmanı sayesinde verilerin boyutunda azalma olmakta fakat bu modelin daha küçük boyutlarla çalışacağı için modele eğitim zamanından kazanç sağlamaktadır.



Şekil 3.9: En iyileri biriktirme yöntemi **[50]**.

3.2.3. Adım Aralığı ve Doldurma

Filtrelerin evrişim işlemini gerçekleştirebilmeleri için girdilerin üzerinde kayarak gezmesi gerekmektedir. Bunun için filtrenin her kaydırma için pencereyi ne kadar hareket ettireceği değerinin bilmesi gerekir. Filtrenin bu her kaydırma iterasyonunda ne kadar ilerleyeceğini belirten parametreye adım aralığı (stride) denilmektedir.

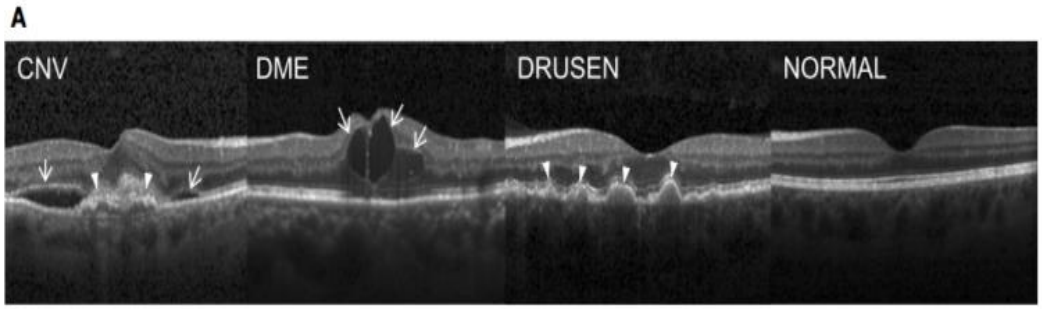
Adım aralığı boyunca katdırılıp filtreyle çarpılan girdilerin kenar kısımları orta kısımlara göre her zaman daha az işlem yapılmış olur. Bundan dolayı görüntü verisinin kenarlarında piksel kaybı yaşanmış olmaktadır. Bunu önlemek için doldurma (padding) denilen bir parametre bulunmaktadır. Doldurma, kenarlardan ekstra veri kaybını önlemek için, o kısımlara fazladan değerler atamaktadır. Böylelikle belitilen adım aralığı kadar kaydırılan pencere, her yere eşit miktarda işlem uygulamaktadır.

3.3. VERİ SETİ VE PROGRAMLAMA

“Göz tomografisi olarak da bilinen optik koherens tomografisi (OKT), ışık (lazer) vasıtasıyla göz dokularının kesitsel görüntülenmesine olanak sağlar. Retina bozukları, kornea bozuklukları, glokom hastalığı ve optik sinir hastalıklarının tanı ve takibinde kullanılır. İlk olarak 1990 başlarında kullanılmaya başlanan cihaz, göz alanında önemli gelişmelere yol açmıştır. Retina hastalıklarının tanı ve tedavi sürecinde temel cihazlardan biri haline gelmiştir.” [51]. Bu çalışmada kullanılan Optik koherens tomografi (OCT) verileri Kaggle adlı web sitesinden “Retinal OCT Images” adlı başlıktan alınmış olup 3 ana klasör ve 4’er tane alt klasörden oluşmaktadır [8] .Ana dosyalar kullanım amaçlarına göre sırasıyla “train”, “val” ve “test” olarak adlandırılmıştır. Bu klasörler de kendi içlerinde sınıflandırma yapılacak göz

hastalığı adına göre “CNV”, “DRUSEN” , “DME” ve bu hastalıkların hiçbirinin olmaması durumuna göre “NORMAL” klasörlerine ayrılmıştır. Anlaşılacağı üzere kullanılan eğitim verilerinin etiketli olmalarına göre bu çalışma bir Denetimli öğrenme (Supervised Learning) üzerine kuruludur.

“Koroidal neovasklarizasyon (CNV), vücudun gözün retinasına daha fazla besleyici maddeleri ve oksijeni temin etmek için yeni bir kan damarı ağını oluşturmaya çalışma şeklinde yanlış yönlendirilmesidir.” [52] . Drusen yine CNV gibi bir yaşa bağlı makula dejenarasyonunun bir sonucudur. Doğal sebeplerden ya da anormal bir retina hastalığının bir sonucu olabilir. “Diyabetik makula ödemi (DME) makulada veya retinanın orta kısmında sıvı birikmesinden kaynaklanan ve makulanın şişmesine neden olan bir hastalıktır. Diyabetin oluşturduğu komplikasyonlardan biridir.” [53] .



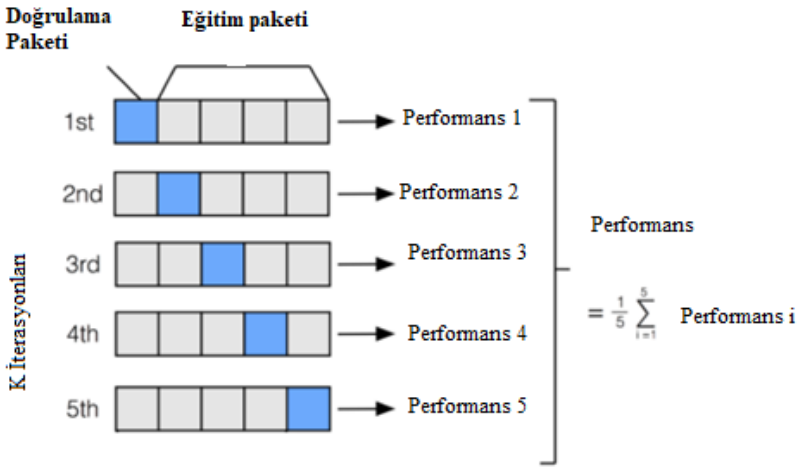
Şekil 3.10: Veri setinde bulunun sınıflara ait örnek görüntüler **[8]**.

Veri setimiz 3 ana klasör, bu 3 hastalık ve 1 nominal durum için oluşturulan 4 alt klasörden meydana gelmektedir. Eğitimde kullanılan “train” klasöründe 37205 CNV görüntü dosyası, 11348 DME görüntü dosyası, 8616 DRUSEN görüntü dosyası ve normal durum için 26315 görüntü dosyasından oluşmaktadır. Yani toplamda 83484 görüntü “train” klasöründe bulunmaktadır. Doğrulama için oluşturulmuş “val” klasörü her 4 durum için 8’er görüntü dosyasından yani 32 görüntüden oluşmaktadır. Son olarak sistemi test etmek için kullanılacak olan aynı adı taşıyan “test” klasörü her bir alt klasör için yani “CNV”, “DME”, “DRUSEN” ve “NORMAL” klasörleri için 242’şer görüntü dosyasından toplamda 968 görüntüden oluşmaktadır. Doğrulama klasörünün az veri içermesinden dolayı kodda görüleceği ImageDataGeneratör üzerinde “validation_split= 0,2” yapılarak eğitim klasöründeki verilerin bir kısmı modelin eğitiminde kullanılması için başka bir doğrulama klasörü varmışçasına ayırılmıştır. Bunun sonucunda modele girdi için 66,788 eğitim verisi, 16696 doğrulama verisi elde edilmiştir. Eğitim ve doğrulama için ayrı ayrı oluşturulmuş eğitim jeneratörü ve doğrulama

jeneratörlerinde de iki alt veri setinin bağlı olduğu “train_gen” kullanılmıştır. İki ayrı jeneratör içindeki “subset” kısmında ise hangi jeneratörün hangisi için çalıştırıldığı doğru bir şekilde ayarlanmıştır. Böylelikle elde modelin sağlıklı bir şekilde eğitim yapabileceği kadar veri oluşmuş olmaktadır. Test klasöründeki 968 test verisi ise yeterli miktarda olduğu için sabittir.

Elimizdeki veri seti eğitim ve test olarak 2’ye değil, eğitim, doğrulama ve test olarak 3’e ayrılmıştır. Model “train” yani eğitim setinde eğitilip, doğrulama setinde değerlendirilip, en uygun halini aldığında “test” setinde test edilmiştir. İki değil üç set kullanılmasının sebebi modele test verilerini göstermeden, katman sayılarını ya da parametrelerini değiştirerek veya farklı en iyileyiciler deneyerek başarımı en yüksek değeri çekebilmektir.

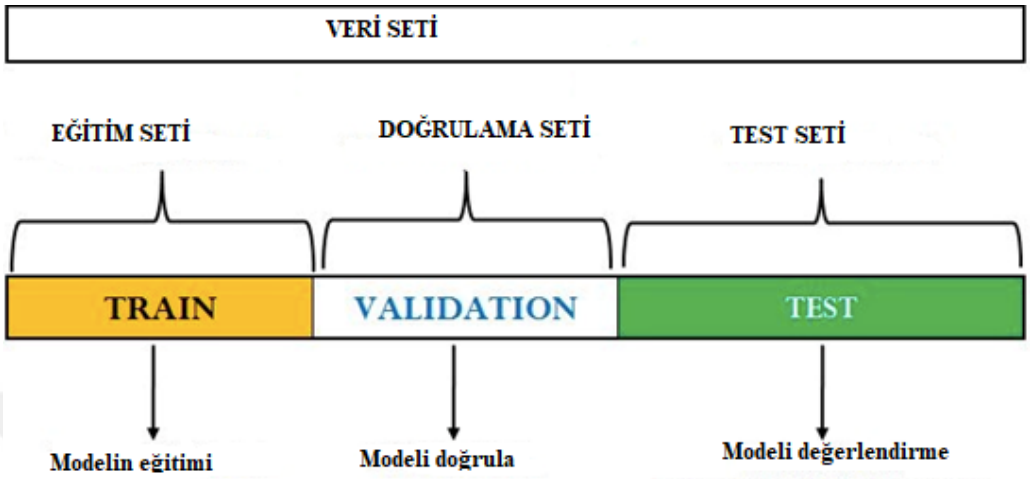
Verileri eğitim, doğrulama ve test olarak ayırırken elde bulunan verinin büyüklüğü çok önemlidir. Verinin büyüklüğüne göre doğrulama setini oluşturmanın bazı yöntemleri vardır. Bunlardan en önemli iki tanesi Cross validasyon yöntemi ve Hold-out validasyon yöntemidir. Cross validasyon ya da diğer adıyla K-Fold validasyon yönteminde eğitim verileri K eşit parçaya bölünür. Her i. parça doğrulama kabul edilerek geri kalan K-1 parçada eğitim yapılıp bir doğrulama skoru elde edilir. Her K parçası için bu yapıldığında elde edilen doğrulama skorlarının ortalaması alınarak nihai doğrulama skoru elde edilir. Bu yöntem elimizde az bir verinin olduğu durumlar için idealdir.



Şekil 3.11: Crossvalidation yöntemi **[54]**.

Bu çalışmada kullanılan Hold-out yöntemi ise elde büyük verilerin bulunduğu durumlar için idealdir. Bu yöntemde genellikle elde bulunan eğitim setinin %30’u doğrulama %70’i eğitim olarak ikiye ayrılır. Eğitim setinde eğitilip doğrulama setiyle en iyi haline gelen set en son test

setinde değerlendirilir. Böylelikle test setinden modele bilgi kaçağı yaşanmamış olur. Eğitim setinin ayrılma oranı verinin büyüklüğüne göre değişebilir. Bu çalışmada eğitim seti %80'e %20 olarak ayrılmıştır.



Şekil 3.12: Verilerin kullanımı **[55]**.

Kullanılan ImageDataGeneratör görüntü verilerini belirli bir dosta yolundan okunmasına yaramasından başka görüntü üzerinde çeşitlendirme yapılabilmesinde de kullanılmaktadır. Bu çeşitlendirilmenin amacı az sayıda bulunan görüntü verilerini kaydırarak, hizalayarak, ters çevirerek vb. uygulamalar ile 1 görüntü verisinden birden fazla veri üretilmesini sağlamaktadır. Üreteç anlamını taşımasının nedeni budur. Bu sayede az sayıda verinin kullanıldığı modellerin aşırı uydurma (overfitting) ya da az uydurma (underfitting) yapmasının önüne geçilmesidir. Yapılan bu tez çalışmasında ise ImageDataGenerator'un flow_from_directory metodu ile önceden oluşturulan klasör yolu ile okunması istenen verilerin oluşturduğu jeneratöre göre veriler modele getirilmiştir. Başlangıçta hazır olarak veri seti ile beraber gelen "train" adlı alt veri setinin yolu "os "modülü kullanılarak ilk olarak "base_dir" ve bu ana klasör yoluna ekleme yapılarak train_dir ve test_dir olarak oluşturulmuştur. Eğitim ve doğrulama için oluşturulan "train_datagen" jenaratörüne "train_dir", test için oluşturulan "test_datagen" jeneratörüne ise "test_dir" okunacak verilerin yolu olarak girilmiştir.

```
base_dir = os.path.join(r"C:\Users\arigu\OneDrive\Masaüstü\archive\OCT2017")
train_dir = os.path.join(base_dir , "train")
print("Train Directory --> ", os.listdir(train_dir))
test_dir=os.path.join(base_dir , "test")
print("Test Directory --> ", os.listdir(test_dir))
```

Şekil 3.13: Verilerin bulunduğu dizinleri hazırlama.

```
train_datagen = ImageDataGenerator(rescale=1./255,validation_split=0.2)

train_generator = train_datagen.flow_from_directory(train_dir,target_size=(150,150),
                                                    batch_size=64,
                                                    class_mode='categorical',
                                                    subset='training') # Eğitim setini ayarla

validation_generator = train_datagen.flow_from_directory(train_dir,target_size=(150,150),
                                                         batch_size=64,
                                                         class_mode='categorical',
                                                         subset='validation') # Doğrulama setini ayarla
```

Şekil 3.14: Eğitim ve doğrulama jeneratörlerinin kurulması.

Yukarıdaki görselde görüldüğü üzere eğitim jeneratörü ve doğrulama jeneratörünün verileri alacağı klasör yolu aynıdır. Eğitim ve doğrulamada kullanacak verilerin ayrımı ise "subset" ile belirlenmiştir. "Target_size= (150,150)" seçilerek verilerin boyutları ayarlanış, "batch size=64" seçilerek jeneratörün bir defada çekeceği veri sayısı belirlenmiştir. Çoklu sınıflandırma yapılacağı için ise "class_mode =categorical" olarak belirlenmiştir.

Veriler uygun bir şekilde ayrılıp modele çağrıldıktan sonra sıra ağ mimarisini kurmaya gelmektedir. Bunun için iki yol bulunmaktadır. Bunlardan ilki ve bu çalışmada kullanılmış olan Sequintial olarak katmanları oluşturmak ikincisi ise Functional API kullanmak. Lineer olarak art arda sıralanan, çoklu giriş ve çıkışı olamayan ve katman paylaşımına gerek olmayan ağlar için uygun olduğundan dolayı bu çalışmada Sequential modelin kullanımı uygun bulunmuştur.

Keras' dan çağrılan models modülü ile birlikte models.Sequential() yapılarak sıralı bir ağ yapısının tabanı oluşturulmuştur. Böylelikle bundan sonra oluşturulacak tüm katmanlar kendinden önceki katmandan tek bir giriş alıp kendinden sonraki katman için tek bir çıkış oluşturacak ve sıralı bir şekilde art arda gelecektir.

```
model=models.Sequential()
```

Şekil 3.15: Lineer bir katman yapısının alt yapısı oluşturuldu**.**

Ağ mimarisinin tabanı oluşturulduktan sonra sırada evrişim katmanlarını oluşturmak bulunmaktadır. Önceki başlıkta açıklandığı üzere evrişim katanları derin öğrenmede görüntü verilerini işlemek için kullanan katman yapılarıdır. Detaylı açıklamalar önceki bölümlerde verilmiştir. Aşağıdaki görselde bu çalışmada kullanılan evrişim katmanları gösterilmektedir.

```
model=models.Sequential()

model.add(layers.Conv2D(128,(3,3),activation="relu",input_shape=(150,150,3)))
BatchNormalization()
model.add(layers.MaxPooling2D((2,2)))


model.add(layers.Conv2D(128,(3,3),activation="relu"))
BatchNormalization()
model.add(layers.MaxPooling2D((2,2)))

model.add(layers.Conv2D(64,(3,3),activation="relu"))
BatchNormalization()
model.add(layers.MaxPooling2D((2,2)))

model.add(layers.Conv2D(64,(3,3),activation="relu"))
BatchNormalization()
model.add(layers.MaxPooling2D((2,2)))

model.add(layers.Conv2D(32,(3,3),activation="relu"))
BatchNormalization()
model.add(layers.MaxPooling2D((2,2)))
```

Şekil 3.16: Evrişim işleminin gerçekleştiği katmanlar.

Görüldüğü üzere ilk olarak evrişim işleminin gerçekleştirildiği Conv2D katmanları eklenmiştir. Keras kütüphanesine ait olan Conv2D katmanının aldığı ilk parametre çıkışta hesaplanacak filtre sayısıdır. Bu değer ilk katmanda 128, ikinci sırada yine 128 daha sonra 64, 64, 32 şeklinde devam etmektedir. Bu seçilen filtre sayısı tamamen kullanıcıya bağlı olarak değişmektedir. Modelin test aşamasına geçmeden eğitim esnasında değiştirilerek en uygun değerin bulunmasıyla elde edilmektedir. Oluşturulan evrişim ağı art arta gelen toplam beş adet Conv2D katmanından oluşmaktadır.

Conv2D katmanının aldığı ikinci parametre (3,3) ise adım aralığını belirlemektedir. Bu değer görüntü üzerinden çıkarılıp filtre ile çarpılan pencerelerin boyutuna belirtmektedir. Genellikle (3x3) olarak seçilir, lakin (5x5)'lik seçildiği durumlarda vardır.

Evrişim işleminde diğer bir önemli konu katmanda kullanılacak aktivasyon fonksiyonunun belirlenmesidir. Daha önceki YSA adlı başlıkta aktivasyon fonksiyonları daha detaylı belirtilmiştir. Bu tez çalışmasında ise ReLu aktivasyon fonksiyonu kullanılmıştır. ReLu fonksiyonunun açılımı Doğrultulmuş Lineer Birim (Rectified Linear Unit) olup negatif değerler için sıfır çıktısını verirken pozitif a değeri için a çıktısını vermektedir. Böylelikle negatif değerler yok edilmiş olup çıkış sınırlandırılmıştır.

Bir diğer evrişim katmanı girdisi input_shape'dir. Giriş verilerinin boyutlarını belirlemektedir. Bu girdi sadece ilk katmanına verilmektedir. Diğer evrişim katmanları bu girdiden yararlanarak kendileri girdi boyutlarını belirleyecektir. Bu çalışmada giriş verilerinin boyutu (150,150,3) olarak belirlenmiştir.

Buna evrişim işleminde doldurma denilmektedir. Evrişim ve havuzlama katmanları arasında yığın normalleştirilmesi yapılmıştır. Bu Keras kütüphanesine ait BatchNormalization() ile yapılmıştır. Yığın normalleştirmesinin yapılmasının amacı evrişim katmalarına giren veya çıkan verilerin korelasyonuna bakarak veriyi normalize edilmesini sağlamaktadır. Böylelikle modelin eğitim işlemini daha kısa sürede tamamlanması sağlamaktadır.

Evrişim katmanından çıkan nitelik haritalarının boyutu çok büyük olduğu için havuzlama katmanı kullanılmaktadır. Bunun sebebi evrişim katmanından çıkan verilerden havuzlama yöntemiyle boyut azaltmaktadır. Havuzlama katmanının çalışma şekli 3.2.2. bölümünde anlatılmıştır. Buraya kadar olan kısım ağın evrişim bölümü olarak geçmektedir. Bundan sonraki kısımda ağın sınıflandırma yapabilmesi gerek katmanlar eklenmektedir.

```
model.add(layers.Flatten())

model.add(layers.Dropout(0.5))


model.add(layers.Dense(64,activation="relu"))
model.add(layers.Dense(4,activation="softmax"))
model.summary()
```

Şekil 3.17: Sınıflandırmanın gerçekleştiği katmanlar.

Yukarıdaki kod parçasında görüldüğü gibi evrişim işleminden çıkan veriler ilk kez Flatten katmanına gelmektedir. Flatten katmanı düzleştirme katmanı olarak geçmektedir. Kullanılmasındaki amacı, evrişim kısmından çıkan verilerin çok boyutlu olmasına karşın sınıflandırma işleminin yapıldığı tam bağlı katmanlarda verilen tek boyutlu olmasıdır. Flatten katmanı bu çok boyutlu verileri tek boyuta indirip, tam bağlı yani Dense katmanlarına işlenebilecek hale getirmektedir. Yani evrişim ile sınıflandırıcı arasında köprü görevi görmektedir.

Düzleştirme katmanından sonra Dropout katmanı kullanılmıştır. Dropout katmanının amacı modelin aşırı uydurmasını yani ezberlemesini engellemek için katmanlardaki nöronların bazılarını belirtilen oranda etkisiz hale getirmektir. Bu çalışmada bu oran yüzde elli olarak belirlenmiştir.

Dropout katmanından sonra sıra sınıflandırıcı kısmını kodlamaya gelmiştir. Buraya gelen veri tek boyutlu olarak Dense katmanlarına giriş yapmaktadır. İki adet tam bağlı katman belirlenmiştir. Bunlardan ilkinin parametre sayısı 64 ikincisinin yani çıkış katmanının ise 4'tür. Çıkış katmanındaki parametre sayısının 4 olmasının sebebi, sınıflandırma yapılması istenen görevin dört sınıftan oluşmasıdır. Ayriyeten görüldüğü üzere çıkış katmanının aktivasyon

fonksiyonu diğer katmanlarınkinde farklıdır. Kullanılan softmax fonksiyonu çoklu sınıflandırma problemlerinde kullanılan sıfır ile bir arasında bir değer verebilen fonksiyondur. Yukarıdaki kod öbeğinde görülen son satırda yer alan model.summary() ise python terminal ekranında katmanların gösterilmesini sağlamaktadır. Buraya kadar olan kısım oluşturulan modelin ağ kısmıdır.

```
model.compile(loss="categorical_crossentropy",optimizer = optimizers.Adam(lr=0.001, beta_1=0.9, beta_2=0.999)  ,metrics = ["accuracy" ])
```

Şekil 3.18: En iyileyici ve kayıp fonksiyonu belirlendi.

Yukarıda olan kod satırında modelin öğrenme işleminin gerçekleştiği katman ağırlıklarının güncellenmesini sağlayan en iyileyici kısmının belirlendiği kısmıdır. Yukarıda gösterilen loss parametresi kayıp fonksiyonunun belirlendiği kısımdır. Kayıp fonksiyonun ağın eğitim esnasında hesapladığı değerlere gerçek değer arasındaki farkı hesaplayan buna göre ağırlıkları güncelleyen fonksiyondur. Çoklu sınıflandırmanın yapılmasından dolayı bu çalışmadan kayıp fonksiyonu "categorical_crossentropy" olarak belirlenmiştir.

İkinci parametre olarak Adam eniyileyici seçilmiştir. Adam eniyileyici önceki kısımlarda belirtildiği gibi, derin öğrenme problemlerinde sıklıkla kullanılan gürültüye karşı dayanıklı, gradyanların birincil ve ikincil momentumlarını kullanan bir en iyileyicidir. Birincil momentum beta_1, ikincil momentum beta_2 ve öğrenme oranı lr ile temsil edilmektedir. Kullanılan bu değerler Adam en iyileyicisinin önden tanımlı değerleridir. Son olarak metrics parametresi ile en iyileyicinin performansının iyi mi kötü mü olduğunu aradığı kısımdır. Burada "accuracy" ile başarım kıstas alınmıştır. Buraya kadar olan kısım ile modelin altyapısı kurulması sağlanmıştır. En iyileyeciler konusu bölüm 3.1.2'de detaylı anlatılmıştır.

```
history = model.fit_generator(train_generator,
                              steps_per_epoch = train_generator.samples // 64,
                              validation_data = validation_generator,
                              validation_steps = validation_generator.samples // 64,
                              epochs = 10,

                              )
```

Şekil 3.19: Model eğitiminin gerçekleştirildiği kısım.

Yukarıda görüldüğü üzere sıra modeli eğitmeye kalmıştır. Bunu için başlangıçta olduğu gibi jeneratörlerden yararlanmıştır. Keras'a ait olan fit_generator metodu ile beraber train_generator eğitim verilerinin alınacağı yer olarak verilmiş steps_per_epoch ile epok başına adım sayısı belirlenmiş, doğrulama verisi validation_data parametresine başlangıçta elle ayarlanan "validation_generator" ile verilmiş ve en son olarak epok sayısı 10 olarak belirlenmiştir. Burada

eğitilen 10 epok boyunca eğitilen modelin eğitim ve doğruluk başarım ve kayıplarını görmek için aşağıdaki kodlamalar yapılmıştır. Bunları yaparken Python'ın en çok kullanılan görselleştirme kütüphanesi olan matplot.lib' den faydalanılmıştır.

```
#Eğitim ve doğrulama için başarı ve kayıp grafiklerini çizdirelim
acc=history.history["accuracy"]
val_acc=history.history["val_accuracy"]

val_loss = history.history['val_loss']
lss = history.history['loss']


epochs=range(1,len(acc)+1)
plt.plot(epochs,acc,"bo",label="Eğitim Başarımı")
plt.plot(epochs,val_acc,"b",label="Doğruluk Başarımı")
plt.title("Eğitim ve Doğruluk Başarımı")
plt.legend()
plt.grid()
plt.figure()


epochs=range(1,len(acc)+1)
plt.plot(epochs,lss,"bo",label="Eğitim Kaybı")
plt.plot(epochs,val_loss,"b",label="Doğruluk Kaybı")
plt.title("Eğitim ve Doğruluk Kaybı")
plt.legend()
plt.grid()
plt.figure()
```

Şekil 3.20: Eğitim ve doğrulama için başarı ve kayıp grafiklerinin görselleştirilmesi.

Böylelikle buraya kadar olan kısımla birlikte veriler hazırlanmış modele çağrılmış ağ mimarisi kurulmuş model eğitilmiş, her epok için eğitilen modelin doğruluk ve kayıp değerleri hem gösterilmiş hem de grafik yardım ile görselleştirilmiştir. Ortaya çıkan değerler bulgular kısmında detaylı olarak açıklanmıştır.

Bir derin öğrenme modelinin asıl başarısı ise daha önce görmediği veriler üzerine gösterdiği performansa göre değerlendirilmektedir. Verileri ezberlemeden ilk kez karşılaştığı verilerde de iyi değerlendirmeler yapmalıdır. Bunu sağlamak için başlangıçta eğitim verileri için yapılan dizin ve jeneratör oluşturma test verileri için de yapılmıştır.

```
test_dir=os.path.join(base_dir , "test")
print("Test Directory --> ", os.listdir(test_dir))
```

Şekil 3.21: Test verileri için dizininin hazırlanması.

```
test_datagen = ImageDataGenerator()

test_generator = test_datagen.flow_from_directory(test_dir,target_size=(150, 150),
                                                  batch_size=64,
                                                  class_mode='categorical')
```

Şekil 3.22: Test için gerekli jeneratörün hazırlanması.

Daha sonradan modelin test skoru belirlemesi için başka bir jeneratör olan evaluate_generator metodunda faydalanılmıştır. Test verisi olarak da test_dir dizininden verileri alan test_generator verilmiştir.

```
test_loss,test_acc =model.evaluate_generator(test_generator,steps=test_generator.samples // 64)
print("Test Başarımı : ", test_acc)
```

Şekil 3.23: Model nihai olarak değerlendirilmesi.

En sonunda model bir sonraki aşama olan tahmin için sürekli eğitilmek zorunda kalmaması için kaydedilmiştir. Böylelikle modelin eğitim ve test işlemi gerçekleştirilmiştir.

```
model.save("Final_Egitilmis_model.h5")
```

Şekil 3.24: Modelin kaydedilmesi.

Modelin başarımının detaylı bir şekilde öğrenmenin en önemli yollarından biri konfüzyon matrisinin çizilmesidir. Konfüzyon matrisi modelin yaptığı tahminlerden yararlanarak sınıflandırmadaki başarımını ayrı ayrı göstermektedir. Bunu için ilk olarak ayrı bir Python dosyasında uygun diziler ayarlanmalı, kaydedilen model geri çağrılmalı ve tahmin için bir jeneratör belirlenmelidir. Daha sora konfüzyon matrisi oluşturmak için tahmin jeneratöründen uygun değişkenler çekilmelidir. Bununla birlikte aşağıda görüleceği üzere konfüzyon matrisin elde edilebilmesi için aşağıda verilen fonksiyon tanımlanmıştır. Hem heatmap şeklinde hem de Python terminalinde konfüzyon matris elde edilmiştir.

```
base_dir = os.path.join(r"C:\Users\arigu\OneDrive\Masaüstü\archive\OCT2017")
test_dir=os.path.join(base_dir , "test")
print("Test Directory --> ", os.listdir(test_dir))



# Prediction için ayrı bir generatör yapılır


loaded_model=load_model("Final_Egitilmis_model.h5")


pred_datagen = ImageDataGenerator()

pred_generator = pred_datagen.flow_from_directory(test_dir,target_size=(150,150),
                                                  batch_size=1,
                                                  class_mode='categorical',
                                                  shuffle = False)
```

Şekil 3.25: Tahmin için jeneratörün hazırlanması

```
pred_generator.reset()
thmn = loaded_model.predict_generator(pred_generator,steps = 968) # 968 Test klösöründeki veri sayısı
test = pred_generator.classes[pred_generator.index_array]
thmn1 = np.argmax(thmn, axis=-1)
```

Şekil 3.26: Modelin gerçek değerler ve kendi tahminlerini çekmesi.

```
def plot_confusion_matrix(cm, classes,
                          normalize=False,
                          title='Konfüzyon matris',
                          cmap=plt.cm.Blues):
    if normalize:
        cm = cm.astype('float') / cm.sum(axis=1)[:, np.newaxis]

    plt.imshow(cm, interpolation='nearest', cmap=cmap)
    plt.title(title)
    plt.colorbar()
    tick_marks = np.arange(len(classes))
    plt.xticks(tick_marks, classes, rotation=90)
    plt.yticks(tick_marks, classes)

    fmt = '.2f' if normalize else 'd'
    thresh = cm.max() / 2.
    for i, j in itertools.product(range(cm.shape[0]), range(cm.shape[1])):
        plt.text(j, i, format(cm[i, j], fmt),
                 horizontalalignment="center",
                 color="white" if cm[i, j] > thresh else "black")

    plt.tight_layout()
    plt.ylabel('Gerçek Etiketler')
    plt.xlabel('Tahmin Edilen Etiketler')

from sklearn.metrics import confusion_matrix
cm = confusion_matrix(test, thmn1)


print('Konfüzyon Matris')

print('\n\n {}'.format(cm))
```

Şekil 3.27: Hata matrisinin çizimi.

Konfüzyon matrisi başarılı şekilde elde edildikten sonra modelin performansı rastgele seçilen verilerle tek tek denenmiştir. Bunun için tek bir veriyi çekebilmek load_image adında ayrı bir fonksiyon tanımlanmıştır.

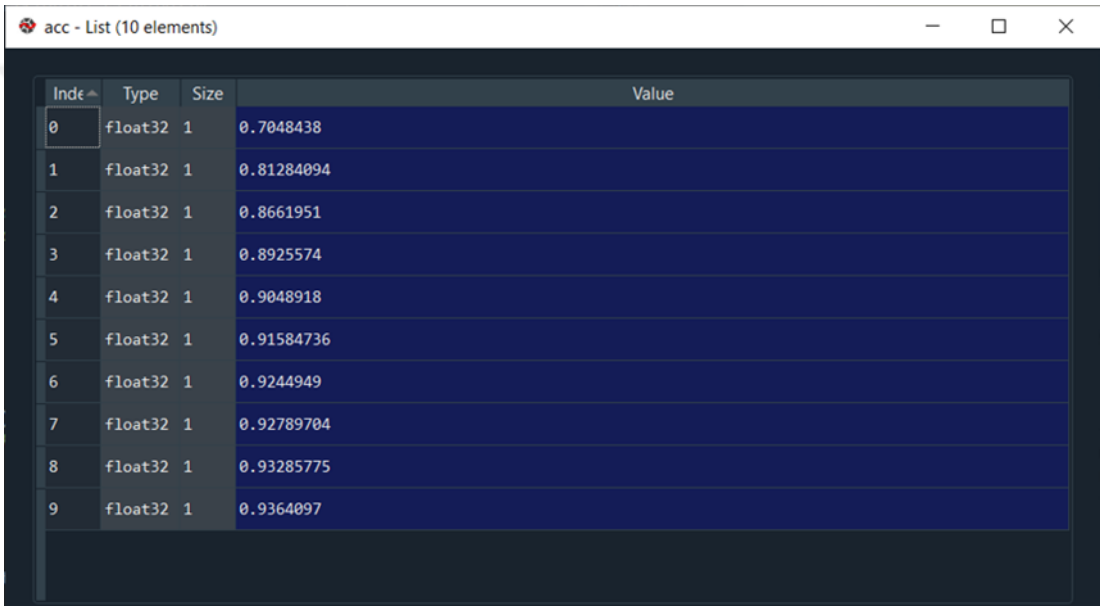
```python
def load_image(img_path, show=False):

    veri = image.load_img(img_path, target_size=(150, 150))
    veri_tensor = image.img_to_array(veri)
    veri_tensor = np.expand_dims(veri_tensor, axis=0)
    veri_tensor /= 255.
    if show:
        plt.imshow(veri_tensor[0])
        plt.axis('off')
        plt.show()

    return veri_tensor


if __name__ == "__main__":

    # modeli geri yükle
    reloaded_model = load_model("Final_Egitilmis_model.h5")


    # istenilen görseli yükle
    yeni_veri = load_image(r"C:\Users\arigu\OneDrive\Masaüstü\NORMAL-5246808-1.jpeg")

    # sonuç tahmin
    pred = reloaded_model.predict(yeni_veri)
```

Şekil 3.28: Tekli olarak veri çağırma ve tahmin etme

Koddan anlaşılacağı üzere NORMAL sınıfına ait bir veri modele verilmiş ve model Bulgular kısmında yer alan çıktıyı vermiştir. Sınıflandırmanın sırası dizinde CNV-DME-DRUSEN-NORMAL şeklinde bulunmaktadır.

4. BULGULAR

Öğrenme işlemini gerçekleştiren derin öğrenme modelinde elde dilen eğitim başarımı alttaki görselde de anlaşılacağı üzere on epok için %93.6 seviyesindedir. Başlangıçta rastgele seçilen katman ağırlıklarından dolayı düşük bir eğitim başarımı elde eden model, on epokluk döngü boyunca ağırlıkları güncellemeye başlamıştır. Aşağıdaki görselde görüleceği üzere, art arda gelen epoklar arsındaki eğitim başarımı farkı, ağırlıkların en iyi değere ulaşıncaya kadar zamanla azalmış en son olarak 0.9364097 değerine ulaşmıştır. Bu değer nihai eğitim başarımı değeridir.



Şekil 4.1: On epok boyunca gerçekleşen eğitim başarımı.

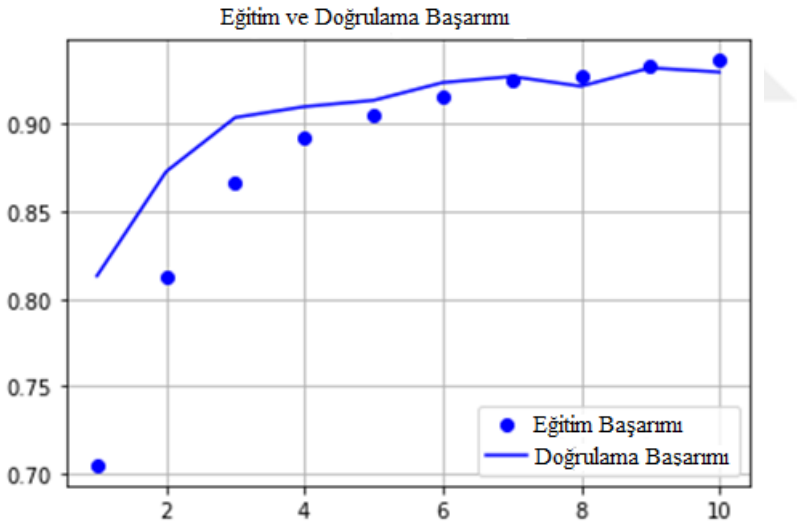
Model eğitim esnasında kendini eğitirken bir yandan da doğrulama veri setinde kendini denemektedir. Doğrulama veri seti, modelin hiper parametreleri ayarlanırken yani model son halini alana kadar test verisi görevi görmektedir. Doğrulama işlemi eğitim ile beraber gerçekleşmekte dolayısıyla on epok halinde çalışmaktadır. Eğitim esnasında olduğu gibi başlangıçta rastgele seçilen ağırlıklardan dolayı, diğer başarımlara göre uzak bir değerde başlayan doğruluk değeri, ağırlıkları en iyiledikçe uygun bir değere ulaşmıştır. Aşağıdaki görselde görüldüğü üzere doğruluk başarımı bu model için %92,9'dur.

val_acc - List (10 elements)

Inde	Type	Size	Value
0	float64	1	0.8130408525466919
1	float64	1	0.8727152347564697
2	float64	1	0.9036796689033508
3	float64	1	0.9098725318908691
4	float64	1	0.9134800434112549
5	float64	1	0.9236411452293396
6	float64	1	0.9271284341812134
7	float64	1	0.9215368032455444
8	float64	1	0.9320586919784546
9	float64	1	0.9296537041664124

Şekil 4.2: On epok boyunca gerçekleşen doğrulama başarımı.

Eğitim ve doğrulama için çalışan modelin her iki veri setinde gösterdiği doğruluk performansını gösteren grafik değer toplam on epok için aşağıdaki grafikte belirtilmektedir.



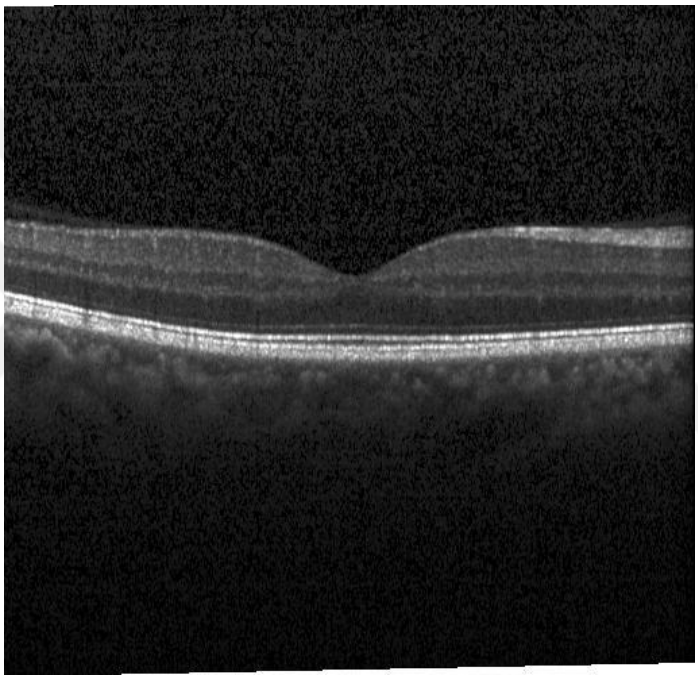
Şekil 4.3: Eğitim ve doğrulama başarım grafiği.

Eğitimden sonra sıra modelin daha önce karşılamadığı verilerle test edilmesine gelmektedir. Daha önce karşılaşmadığı veriler olmasının önemi modelin tekrar tekrar görüp ezber yapması engellemektir. Böylelikle gerçek performansı ortaya çıkmaktadır. Yani test işlemi modelin gerçek performansını gösterdiği kısımdır. Aşağıdaki görselde görüldüğü üzere model test işlemini başarılı bir şekilde gerçekleştirmiş ve %82'lik bir başarım elde etmiştir.

```
Test Başarımı :  0.8208333253860474
```

Şekil 4.4: Modelin test skoru.

Denetimli öğrenme algoritmalarında sınıflandırma ve tahmin görevleri beraber gerçekleşebilmektedir. Tahmin görevi, eğitilmiş modelin kendisine verilen tek bir görüntü için sınıf tahmininde bulunmasıdır. Sınıf sıralama sıfırıncı indeksten başlayıp üçüncü yani son indekse olacak şekilde CNV-DME-DRUSEN-NORMAL şeklinde bulunmaktadır. Bu tez çalışmasında gerçekleştirilen model için aşağıda verilen NORMAL sınıfına ait OKT verisi modeli girdi olarak verilmiş ve modelden sınıf tahmini yapması istenmiştir.



Şekil 4.5: NORMAL sınıfına ait hastalıksız bir göze ait OKT görüntüsü **[8]**.

Bunun sonucunda altta görülen sonuç elde edilmiştir. Bu sonuca göre model, her sınıf için sıfır ve bir arasındaki değerler atayarak verinin hangi sınıfa ait olduğunu göstermektedir. En yüksek değer ile belirttiği sınıf sonuç olarak kabul edilmektedir. Görselden de anlaşılacağı üzere model kendisine verilen girdinin üçüncü indekste yer alan NORMAL sınıfına ait olduğunu belirterek doğru tahminde bulunmuştur.

Şekil 4.6: Modelin tahmin görevi için verdiği sonuç.

Konfüzyon matrisi, tahmin edilen değerler ve gerçek değerler arasında yapılan doğru pozitif, doğru negatif, yanlış pozitif ve yanlış negatif çıktılarını inceleyip, bunların oranlarına göre değerlendirme metriklerine sahiptir. Bu sebepten konfüzyon matrisi, derin öğrenme modellerini detaylıca incelemede önemli rol oynamaktadır. Konfüzyon matrisinin bu çalışmada kullanılan dört adet metriği vardır.

Precision metriği, kesinlik metriği olarak geçmektedir. Modelin doğru pozitif olarak yaptığı değerlendirmelerin, tüm pozitif değerlere ait çıktıların oranını vermektedir.

$$Precison = \frac{TP}{TP+FP} \quad (4.1)$$

Recall metriği hassasiyet metriği olarak geçmektedir. Modeldeki pozitif olan değerlerin, kaç tanesinin doğru tahmin edildiğini göstermektedir. Sınıflandırma görevlerinde önemli bir metriktir.

$$Recall = \frac{TP}{TP+FN} \quad (4.2)$$

F1-Score metriği recall ve precision metriklerinin harmonik ortalamasını vermektedir.

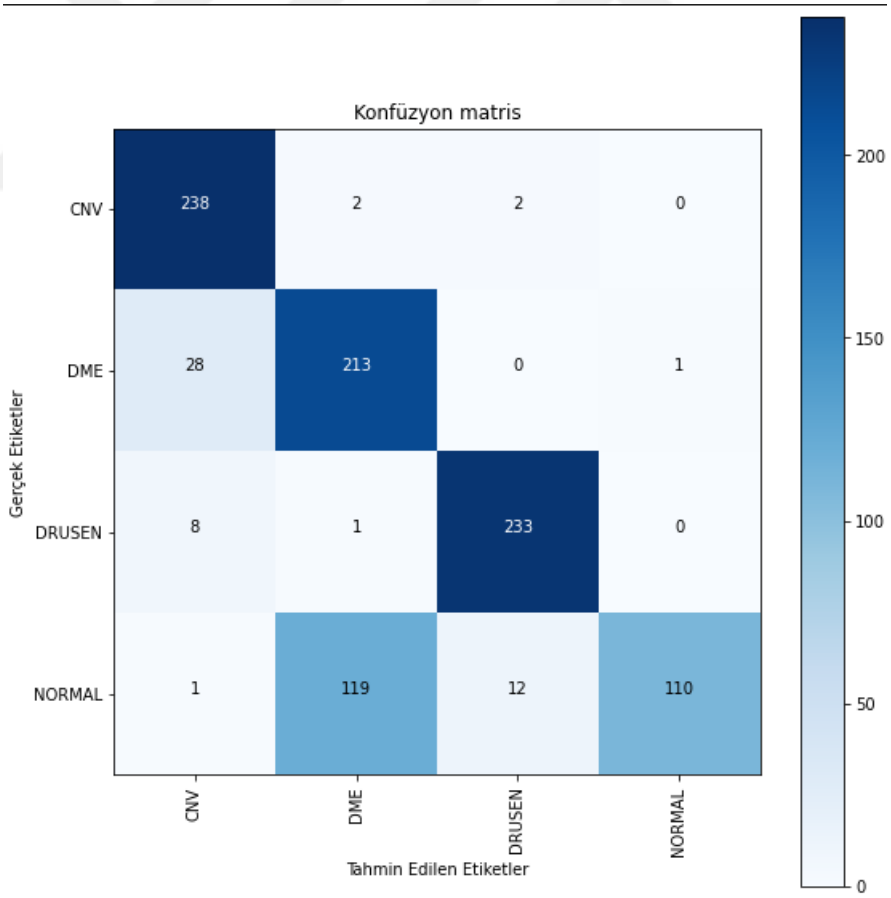
$$F1Score = \frac{2.Precision.Recall}{Precision+Recall} \quad (4.3)$$

Accuracy metriği, modelin doğru olarak yaptığı tahminlerin, tüm tahminlere oranını verdiği metriktir.

$$Accuracy = \frac{TP+TN}{TP+TN+FP+FN} \quad (4.4)$$

```
Konfüzyon Matris


 [[238   2   2   0]
 [ 28 213   0   1]
 [  8   1 233   0]
 [  1 119  12 110]]
              precision    recall  f1-score   support

         CNV       0.87      0.98      0.92       242
         DME       0.64      0.88      0.74       242
      DRUSEN       0.94      0.96      0.95       242
      Normal       0.99      0.45      0.62       242

    accuracy                           0.82       968
   macro avg       0.86      0.82      0.81       968
weighted avg       0.86      0.82      0.81       968
```

Şekil 4.7: Sonuç konfüzyon matrisi.



Şekil 4.8: Heatmap şeklinde olan konfüzyon matrisi.

Modelin test edilmesi, konfüzyon matrisinin çizilmesi ve tekli görsel veriler için tahminde bulunmasından sonra tez çalışması sonuçlandırılmış olmaktadır. Elde edilen sonuçlar değerlendirilmek istendiğinde ilk olarak modelin test başarımına bakılmalıdır. Modelin elde ettiği test başarımı %82 çıkmıştır. Sağlık alanında kullanılması için modellenmiş bir derin öğrenme modeli için bu oran yeterli bir orandır. Bu yeterlilik irdelenmek istendiğinde, konfüzyon matrisi diğer adıyla hata matrisi önemli bir araç olmaktadır.

Oluşturulan konfüzyon matris incelendiğinde model 242 CNV verisinden 238'ini doğru tahmin edip CNV olduğunu anlamış, dört tane veride hata yapıp iki tanesini DME iki tanesini DRUSEN hastalığına ait olduğu belirtmiştir. Bu değerler yüksek değerlerdir. Yani model CNV verisini tanıyabilmektedir.

DME verileri için konfüzyon matris incelendiğinde, model 242 veriden 213'ünü doğru tahmin etmiş, 28 tane verinin CNV sınıfına ait, bir tanesinin de NORMAL sınıfına ait bir veri olduğunu belirterek yanlış tahminde bulunmuştur.

Modelin, DRUSEN sınıfına ait verilerdeki performansı incelendiğinde 242 adet veriden 233'ü doğru tahmin edilmiş, sekiz tanesi CNV ve bir tanesi DME sınıfına ait oldu çıktısını vererek yanlış cevap vererek en iyi performansını bu sınıfta göstermiştir.

NORMAL sınıfına bakıldığında ise modelin kötü bir performans gösterdiği 242 veriden sadece 110'ununda doğru tahminde bulunduğu, 119 veriyi DME sınıfına 12'sini de DRUSEN sınıfına ait olduğu tahminde bulunduğu görülmektedir. Bu değerlere bakarak modelin NORMAL verileri tahmin etmede ise istenilen düzeyde değildir.

Daha detaylı inceleme için konfüzyon matrisinin performans metriklerine bakıldığında precision (kesinlik) metriğinin belirtilen dört sınıf için sırasıyla 0.87, 0.64, 0.94 ve 0.99 olduğu görülmektedir. Precision metriği modelin, o sınıf için yaptığı doğru veya yanlış pozitif tahminlerin kaç tanesinin gerçekten doğru pozitif olduğu oranını vermektedir. Bu oranlar incelendiğinde en düşük kesinlik değerinin DME sınıfına en yüksek değerinde NORMAL sınıfına ait olduğu görülmektedir.

Diğer bir konfüzyon matrisi metriği olan recall (hassasiyet) incelendiğinde dört sınıf için sırasıyla 0.98, 0.88, 0.96 ve 0.45 değerlerinin elde edildiği görülmektedir. Recall metriği modelin pozitif olarak değerlendirilmesi gereken verilerin kaç tanesinin pozitif olarak kabul

edildiğini göstermektedir. Bu değerler incelendiğinde model NORMAL sınıfına ait verilerin çoğunluğunu doğru tahmin etmediği görülmüştür.

Bu iki metriğin karışımı olup modelin genel performansını değerlendirmek için çoğunlukla kullanılan F1-Score metriğine bakıldığında sınıf sırasına göre 0.92, 0.74, 0.95 ve 0,62 değerleri elde edilmiştir. Bu değerler incelendiğinde modelin CNV ve DRUSEN verilerini tahmin etmede çok başarılı olduğu, DME ve özellikle NORMAL sınıfına ait verilerde çok iyi tahminlerde bulunmadığı görülmüştür. Bu eksikleri önceki bölümlerde anlatıldığı üzere katman sayıları, parametreleri veya kullanılan en iyileyicilerle iyileştirilebilmektedir.

Tablo 4.1: Hata matrisi metrikleri.

	Precision	Recall	F1-Score
CNV	0.87	0.98	0.92
DME	0.64	0.88	0.74
DRUSEN	0.94	0.96	0.95
NORMAL	0.99	0.45	0.62

5. SONUÇ VE TARTIŞMA

Bu alanda yapılmış diğer çalışmalar incelendiğinde ise genellikle transfer fonksiyonlarından yararlanıldığı görülmektedir. Transfer fonksiyonları önceden eğitilmiş sıralı evrişim katmanlarının, kullanılmak istenilen ver setlerine göre, genellikle ilk birkaç katmanı olmak üzere modele getirilmesi, sonlarda bulunan katmanların ise dondurularak etkisiz hale getirilmesiyle olmaktadır.

Rashaad Meyer'in aynı isimli veri setiyle yaptığı ResNET18 önden eğitimli transfer fonksiyonu kullanarak yaptığı çalışmada %99,6 başarı sağlanmıştır [56]. Bu çalışmada doğrulama klasörü kod ile ayrılmamış veri setinin kendisinde bulunan doğrulama klasörüne ait 32 adet veri ile çalışılmıştır. Ayriyeten bu tez çalışmasından farklı olarak keras ve ilgili kütüphanelerle değil PyTorch kütüphanesinden faydalanılmıştır. Adam en iyileyici iki çalışma için de ortaktır.

Diğer bir çalışma olan Mostafa Ashraf'in yaptığı çalışmada farklı bir katman mimarisi kullanılarak art arda gelen evrişim katmanları arasına değil evrişim katmanları grupları arasına havuzlama katmanları kurulmuştur [57]. Model yirmi beş eğitim epoğuyla eğitilmiş ve test edilmiştir ve kullanılan dropout değerleri 0,2 olarak belirmiştir. Bu değer bu tez çalışmasında 0,5 'tir.

Hamza Münir'in yine aynı veri setini temel alarak yaptığı çalışmada, bu tez çalışmasında olduğu gibi beş adet evrişim katman farklı katman parametreleri ile kullanılmıştır [58]. Giriş verisi 150'ye 150 değil 128'e 128 belirtilmiştir. Ayriyeten adım aralığı önden tanımlı değer olan bir değil iki olarak kullanılmıştır. Kullanılan tam bağlı katmanların sayısı da tez

çalışmasından farklı olarak dörttür. Dropout katmanı kullanılmamış ve model yirmi epok ile eğitilmiştir.

Bunun sonucunda model %92 başarı elde etmiştir. Yukarıda bahsedilen tüm çalışmalar veri setinin kendisinden gelen sadece 32 veriden oluşan doğrulama klasörü ile yapılmıştır.

Daha önce yapılmış ilgili çalışmalar incelenip bu tez çalışma ile kıyaslandığında önceden eğitilmiş ağların test başarımında etkili olduğu görülmüştür. Bunun yanında önceden eğitilmiş olmayan, baştan oluşturulan ağların benzerlik oranları diğerlerine göre çok düşüktür.

Yine diğer çalışmalarla kıyaslandığında eğitim epoğu sayısının diğer çalışmalara göre daha düşük olduğu görülmektedir. Bu model eğitim süresini kısaltsa da modelin öğrenebileceği maksimum seviyeye daha gelmediğini göstermektedir.

Diğer bir performansı etkileyen kriter ise Dropout katmandır. Dropout katmanı modelinaşırı uydurmaya girmemesi için katmanlarda buluna nöronları, verilen oranda rastgele olarak söndürmeye yaramaktadır. Yukarıda bahsedilen çalışmalarda ya bu katman kullanılmamış ya da söndürme oranı 0,2 olarak belirlenmiştir. Tez çalışmasında ise bu değer 0,5 olarak belirleniştir. Anlaşılacağı üzere bu değer model için yüksek bir değer olmuş ve bilgi kaybına neden olmuştur.

Bu çalışmayı diğer çalışmalardan ayıran en önemli fark ise kullanılan veri miktarlarıdır. Diğer çalışmalar doğrulama için toplamda 32 veri ile çalışırken, bu çalışmada gerçekleştirilen modelde doğrulama seti eğitim setinden özel olarak ayırılmıştır. Bu modelin eğitim esnasında gördüğü veri sayısının diğerlerine göre %20 az olduğunu göstermektedir.

Bütün bu değerlendirmelere rağmen model kendinden istenen eğitim, test ve tahmin görevlerini başarı ile gerçekleştirmiştir. Yapılan bu tez çalışmasında evrişimli sinir ağlarının dolayısıyla derin öğrenmenin, artan veri akışı ile birlikte büyüyen donanımsal imkanlarla beraber hem hasta hem de görevli uzmanlar için avantaj sağladığı ispatlanmıştır. Özellikle görüntü verilerinin

işlendiği oftalmoloji, dermatoloji ve diğer tıbbi görüntüleme cihazlarının kullanıldığı sağlık birimlerinde kullanımı belirtilmiş ve kanıtlanmıştır.

KAYNAKÇA

[1] Z. Kapran. [Çevrimiçi]. Available: 1. https://ziyakapran.com/optik-koherens-tomografi.

[2] İstanbul Retina, [Çevrimiçi]. Available: 2. https://www.istanbulretina.com/makaleler-goz-hastaliklari-drusen-cesitleri-ve-goruntulenme-yontemleri.php.

[3] A. Geron, %1 içinde *Scikit-Learn, Keras ve TensorFlow ile Uygulamalı Makine Öğrenmesi* , 2021, p. 9.

[4] F. Ç. Gündoğan. [Çevrimiçi]. Available: https://gozdoktor.net/gozun-yapisi/.

[5] S. AKAR, %1 içinde *Göz Hastalıkları*, İstanbul Üniversitesi Cerrahpaşa Tıp Fakültesi, 1997.

[6] [Çevrimiçi]. Available: https://gormekguzeldir.com/diyabetik-makuler-odem#:~:text=Diyabetik%20Mak%C3%BCler%20%C3%96dem%2C%20diyabet%20hastalar%C4%B1nda,makula%20ad%C4%B1%20verilen%20g%C3%B6rme%20merkezinde.

[7] MUMCUOĞLU, ERDURMAN ve DURUKAN, «Optik koherens tomografi prensipleri ve uygulamadaki yenilikler,» *Turk J Ophthalmol,* 2008.

[8] P. Mooney, «KAGGLE,» [Çevrimiçi]. Available: https://www.kaggle.com/datasets/paultimothymooney/kermany2018. [Erişildi: 2021].

[9] P. S. O. F. R. U. &. T. M. T. Grewal, «Deep learning in ophthalmology: a review,» *Canadian Journal of Ophthalmology*, 2018.

[10] Y. X. D. W. K. W. T. Y. W. a. J. L. X. Chen, «Glaucoma detection based on deep convolutional neural network,» %1 içinde *37th Annual International Conference of the IEEE Engineering in Medicine and Biology Society (EMBC)*, 2015.

[11] A. J. T. N. P. D. Y. W. A. R. a. A. Y. L. Cecilia S. Lee, «Deep-learning based, automated segmentation of macular edema in optical coherence tomography,„» *Biomed. Opt. Express,* 2017.

[12] J. Markoff. [Çevrimiçi]. Available: https://www.nytimes.com/2011/11/08/science/computer-experts-building-1830s-babbage-analytical-engine.html.

[13] «Tarihteki İlk Bilgisayar Programcısı,,» [Çevrimiçi]. Available: https://www.emo.org.tr/ekler/51558fc12fe4df1_ek.pdf?dergi=1151.

[14] [Çevrimiçi]. Available: https://www.fundacionaquae.org/wiki/ada-lovelace-madre-de-la-programacion/.

[15] I. GOODFELLOW, Deep Learning, 2015.

[16] Wikipedia, [Çevrimiçi]. Available: https://tr.wikipedia.org/wiki/Dartmouth_Konferans%C4%B1.

[17] S. Russel ve P. Norvig.

[18] A. Öncü, «Uzman Sistem Yaklaşımı ile Web Tabanlı Öğretim Dğerelendirme Sisteminin Geliştirilmesi,» 2006.

[19] [Çevrimiçi]. Available: https://www.javatpoint.com/expert-systems-in-artificial-intelligence.

[20] J. Prosise. [Çevrimiçi]. Available: https://www.atmosera.com/blog/supervised-learning-with-k-nearest-neighbors/.

[21] F. Azank. [Çevrimiçi]. Available: https://medium.com/turing-talks/clustering-conceitos-b%C3%A1sicos-principais-algoritmos-e-aplica%C3%A7%C3%A3o-ace572a062a9.

[22] L. Arnold. [Çevrimiçi]. Available: https://ludovicarnold.com/teaching/optimization-machine-learning/unsupervised-example-clustering-k-means/.

[23] T. Alade. [Çevrimiçi]. Available: https://blog.cambridgespark.com/how-to-determine-the-optimal-number-of-clusters-for-k-means-clustering-14f27070048f.

[24] A. Banerji. [Çevrimiçi]. Available: https://www.analyticsvidhya.com/blog/2021/05/k-mean-getting-the-optimal-number-of-clusters/.

[25] [Çevrimiçi]. Available: http://www.nlpca.org/pca_principal_component_analysis.html.

[26] S. Hussein. [Çevrimiçi]. Available: https://www.kdnuggets.com/2019/11/tips-class-imbalance-missing-labels.html.

[27] R. ALİYEV ve C. TEKSÖZ, «Reinforcement Learning and Optimization Algorithm (Pekiştirmeli Öğrenme ve Optimizasyon Algoritmaları)».

[28] N. SÜT, M. TÜRE ve M. ŞENOCAK, «Sağlık alanında karar vermede döngüsel süreçlerin kullanımı: Bir markov model uygulaması,» *Trakya Üniversitesi Tıp Fakültesi Dergisi.,* 2007.

[29] ÇINAR.

[30] C. Engin, *Geleneksel Sistem Tasarımları.*

[31] A. Kharwal. [Çevrimiçi]. Available: https://thecleverprogrammer.com/2020/07/31/voting-classifier-in-machine-learning/.

[32] L. Sepeda. [Çevrimiçi]. Available: https://medium.com/turing-talks/como-machine-learning-consegue-diferenciar-heter%C3%B4nimos-de-fernando-pessoa-156d0d52a478.

[33] O. Akhlaq. [Çevrimiçi]. Available: https://livecodestream.dev/post/top-ml-algorithms-for-classification/.

[34] K. R. Shrimali. [Çevrimiçi]. Available: https://learnopencv.com/svm-using-scikit-learn-in-python/.

[35] M. F. AKCA, «Medium,» [Çevrimiçi]. Available: https://medium.com/deep-learning-turkiye/her-%C5%9Feyiyle-lineer-regresyon-makine-%C3%B6%C4%9Frenmesi-serisi-1-1ee2aec10b74.

[36] K. Ülgen, «Medium,» [Çevrimiçi]. Available: https://medium.com/@k.ulgen90/lojistik-regresyon-makine-%C3%B6%C4%9Frenimi-b%C3%B6l%C3%BCm-7-c6bc685a4084.

[37] A. G. Zucco, «Computational analysis of pMHC - TCR interactions.,» 2017.

[38] O. ÖĞÜCÜ, *Yapay Sinir Ağğları ile Sistem Tanıma,* 2006.

[39] [Çevrimiçi]. Available: https://velog.io/@jhdai_ly/%EB%94%A5%EB%9F%AC%EB%8B%9DDL%ED%8D%BC%EC%85%89%ED%8A%B8%EB%A1%A0Perceptron2-%EB%8B%A4%EC%B8%B5-%ED%8D%BC%EC%85%89%ED%8A%B8%EB%A1%A0-XOR-%EA%B2%8C%EC%9D%B4%ED%8A%B8-%EA%B5%AC%ED%98%84-NAND-%EC%8B%A0%EA%B2%BD%EB%A7%9D-%EB%8F%84%E.

[40] [Çevrimiçi]. Available: https://lonepatient.top/2018/09/25/Cyclical_Learning_Rate.

[41] N. Çarkacı, «Medium,» [Çevrimiçi]. Available: https://medium.com/deep-learning-turkiye/derin-ogrenme-uygulamalarinda-en-sik-kullanilan-hiper-parametreler-ece8e9125c4.

[42] V. Gudımalla, «Medium,» [Çevrimiçi]. Available: https://varshithagudimalla.medium.com/concept-of-gradient-descent-algorithm-in-machine-learning-44f587ac16ac.

[43] «Analytics Vidhya,» [Çevrimiçi]. Available: https://www.analyticsvidhya.com/blog/2021/05/tuning-the-hyperparameters-and-layers-of-neural-network-deep-learning/.

[44] D. Chang, «Towards Data Science,» [Çevrimiçi]. Available: https://towardsdatascience.com/effect-of-gradient-descent-optimizers-on-neural-net-training-d44678d27060.

[45] Y. Z. Gomez, «Linkedin,» [Çevrimiçi]. Available: https://www.linkedin.com/pulse/pros-cons-some-machine-learning-algorithms-yulieth-zuluaga-g%C3%B3mez?trk=portfolio_article-card_title.

[46] E. SEYYARER, T. UÇKAN, F. AYATA ve A. KARCİ, «Derin Öğrenmede Kullanilan Optimizasyon Algoritmalarinin Uygulanmasi Ve Kiyaslanmasi,» *Computer Science,* pp. 90-98, 2020.

[47] M. D. Zeiler, «Adadelta: an adaptive learning rate method,» 2012.

[48] M. D. Zeiler, «Adadelta: an adaptive learning rate method,» 2012.

[49] A. İleri, «Medium,» [Çevrimiçi]. Available: https://abdulsamet-ileri.medium.com/2d-convolution-ced5d339aa5.

[50] M. S. MUDARAGADDA. [Çevrimiçi]. Available: https://analyticsindiamag.com/max-pooling-in-convolutional-neural-network-and-its-features/.

[51] ANKARA İL SAĞLIK MÜDÜRLÜĞÜ, «AĞLIK BİLİMLERİ ÜNİVERSİTESİ DIŞKAPI YILDIRIM BEYAZIT EĞİTİM VE ARAŞTIRMA HASTANESİ,» [Çevrimiçi]. Available: https://diskapieah.saglik.gov.tr/TR,403564/optik-koherens-tomografi.html. [Erişildi: 10 12 2022].

[52] İ. ŞAHBAZ, «All About Vision,» [Çevrimiçi]. Available: https://www.allaboutvision.com/tr/ko%C5%9Fullar/mak%C3%BCler-dejenerasyon/. [Erişildi: 10 12 2022].

[53] O. Musat, C. Cernat ve M. Labib , «DIABETIC MACULAR EDEMA,» *Romanian Journal Of Ophtalmalogy,* cilt 59, no. 3, pp. 133-136, 2015.

[54] V. Nguyen, «Medium,» [Çevrimiçi]. Available: https://vannguyen-8073.medium.com/home-credit-default-risk-based-on-lightgbm-part-2-1-8b5fc26e192f.

[55] M. A. Çifçi, «Derin Öğrenme Metodu Kullanarak BT Görüntülerinden,» *Dokuz Eylül Üniversitesi Mühendislik Fakültesi Fen ve Mühendislik Dergisi,* 2022.

[56] R. Meyer, «Kaggle,» [Çevrimiçi]. Available: https://www.kaggle.com/code/rashaadmeyer/retinal-oct-pytorch-99-8-resnet18.

[57] M. Ashraf, «Kaggle,» [Çevrimiçi]. Available: https://www.kaggle.com/code/mostafaashraf1/retina-damage-detection-cnn#Test-cases.

[58] H. Munir, «Kaggle,» 2020. [Çevrimiçi]. Available: https://www.kaggle.com/code/hammuneer/normal-cnv-dme-drusen-classification.

ETİK KURUL İZİN YAZISI

Uyarı: Canlı denekler üzerinde yapılan tüm araştırmalar için Etik Kurul Belgesi alınması zorunludur.

☐ **Etik Kurul izni gerekmektedir.**

☒ **Etik Kurul izni gerekmemektedir.**

Gülsüm ARI
(İmza)

KURUM İZNİ YAZILARI

Uyarı: Canlı ve cansız deneklerle yapılan tüm çalışmalar için kurum izin belgelerinin eklenmesi zorunludur. Gizlilik ve mahremiyet içeren durumlarda kurum adı kapatılmalıdır.

☐ **Kurum izni gerekmektedir.**

☒ **Kurum izni gerekmemektedir.**

Gülsüm ARI
(İmza)