T.C.
KIRIKKALE ÜNİVERSİTESİ
FEN BİLİMLERİ ENSTİTÜSÜ

BİLGİSAYAR MÜHENDİSLİĞİ ANABİLİM DALI

YÜKSEK LİSANS TEZİ

Makine Öğrenmesi ve Derin Öğrenme: Nesne Tanıma Uygulaması

Saliha Kevser KAVUNCU

ARALIK 2018

Şehit Özel Harekat Polisi Said Uslu'ya

ÖZET

MAKİNE ÖĞRENMESİ VE DERİN ÖĞRENME:
NESNE TANIMA UYGULAMASI

KAVUNCU, Saliha Kevser
Kırıkkale Üniversitesi
Fen Bilimleri Enstitüsü
Bilgisayar Mühendisliği Anabilim Dalı, Yüksek Lisans Tezi
Danışman: Dr. Öğr. Üyesi Bülent Gürsel EMİROĞLU
Aralık 2018, 157 sayfa

Büyük veri son yılların en popüler sorunlarından birisidir. İnternet üzerinden veya çevrimdışı olarak elde edilen verilerin saklanması oldukça zordur. Ancak mevcut veriler sayesinde karşılaşılan yeni veriler için doğru çıkarımlar yapmak bilim, ekonomi, tıp, savunma sanayi ve teknoloji gibi birçok disiplinlinler arası alan için oldukça önemlidir. Bu anlamda düzenlenerek bir araya getirilen veriler makine öğrenmesi ve derin öğrenme yöntemleri ile anlamlandırılarak yeni veya öngörülemeyen veriler için doğru tahminlerin yapılmasını sağlar.

Bu çalışmada derin öğrenme yöntemi ve diğer makine öğrenmesi yöntemleri anlatılmıştır. Yöntemler, gerçek görüntülerden oluşan CIFAR-10 ve MNIST veri setleri üzerinde uygulanmıştır. Yeni bir derin ağ modeli (BasitNet) geliştirilerek lojistik regresyon, naive bayes, rastgele orman ve k-en yakın komşu yöntemleri ile karşılaştırılmıştır. Önerilen BasitNet modeli ile 1/50 oranında daha az parametreyle AlexNet düzeyinde başarı elde edilmiştir. Elde edilen sonuçlar önerilen BasitNet diğer yöntemlerden daha başarılı olduğunu göstermiştir.

Anahtar kelimeler: Makine Öğrenmesi, Derin Öğrenme, Nesne Tanıma, CIFAR-10, ESA, Görüntü İşleme.

ABSTRACT

MACHINE LEARNING AND DEEP LEARNING:
APLICATIONS OF OBJECT RECOGNITION

KAVUNCU, Saliha Kevser
Kırıkkale University
Graduate School of Natural and Applied Sciences
Department of Computer Engineering, M. Sc. Thesis
Supervisor: Asst. Prof. Dr. Bülent Gürsel EMİROĞLU
December 2018, 157 pages

Big data is one of the most popular problems experienced in recent years. Storing data obtained as offline or over the Internet is very difficult. However, making the correct inferences for the new data originating from the existing data is very important for many interdisciplinary fields such as science, economics, medicine, defense industry and technology. In this sense, the data gathered by means of machine learning and deep learning methods is used to make accurate predictions for new or unpredictable data.

In this study, Machine Learning Methods and Deep Learning Method are explained. The methods were applied on CIFAR-10 and MNIST datasets composed of real images. A new deep network model (SimpleNet) was developed and compared with logistic regression, naive bayes, random forest and k-nearest neighbor methods. AlexNet level has been achieved with less than 1/50 recommended parameters. The results obtained have shown that the proposed SimpleNet is more successful than the other methods.

Key Words: Machine Learning, Deep Learning, Object Recognition, CIFAR-10, CNN, Image Processing.

TEŞEKKÜR

Bu çalışmanın gerçekleştirilmesinde, değerli bilgilerini benimle paylaşan, desteklerini esirgemeyen danışman Hocam Sayın Dr. Öğr. Üyesi Bülent Gürsel Emiroğlu'na içten teşekkürlerimi borç bilirim. Tezin birçok aşamasında yardımlarından ve cesaretlendirmelerinden dolayı Sayın Dr. Öğr. Üyesi Serkan Dişlitaş'a ve değerli arkadaşım Meryem Taşkesen'e teşekkürlerimi sunarım.

Hayatım boyunca, her daim varlıklarını çok güçlü hissettiğim değerli ailem; babam Erol, annem Huriye, ablam Sümeyye, enişte Nurettin ve kardeşim Şeyma Nur'a minnettarlığımı ve şükranlarımı sunarım.

İÇİNDEKİLER DİZİNİ

Sayfa

ÖZET ..... **i**

ABSTRACT ..... **ii**

TEŞEKKÜR ..... **iii**

İÇİNDEKİLER DİZİNİ ..... **iv**

ŞEKİLLER DİZİNİ ..... **vi**

ÇİZELGELER DİZİNİ ..... **ix**

KISALTMALAR DİZİNİ ..... **xi**

1. GİRİŞ **1**

1.1. Yapay Zekâ 2

1.2. Görüntü İşleme 8

1.3. Nesne Tanıma 10

1.4. Literatür Çalışmaları 11

2. MATERYAL VE YÖNTEM **12**

2.1. Makine Öğrenmesi 12

2.2. Öznitelik Çıkarma ve Seçme İşlemleri 14

2.3. Makine Öğrenmesi Yöntemleri 18

2.4. Denetimli Öğrenme 19

2.4.1.Sınıflandırma Tahmin Yöntemleri 20

2.4.1.1. Lojistik Regresyon 20

2.4.1.2. k-En Yakın Komşu 22

2.4.1.3. Destek Vektör Makineleri 24

2.4.1.4. Naive Bayes 26

2.3.1.5. Karar Ağaçları 28

2.4.2. Regresyon Tahmin Yöntemleri 35
2.4.2.1. Doğrusal Regresyon 35
2.4.2.2. Polinom Regresyon 37
2.4.2.3. Bayes Ağları 37
2.5. Denetimsiz Öğrenme 39
2.5.1. Kümeleme Tahmin Yöntemleri 39
2.5.1.1. K-Ortalamalar 40
2.5.1.2. Birliktelik Kuralları 42
2.5.1.3. Genetik Algoritmalar 45
2.5.1.4. Hiyerarşik Kümeleme 46
2.5. Yapay Sinir Ağlarından Derin Öğrenmeye 48
2.6. Derin Öğrenme 58
2.7. Transfer Öğrenme 84
3. UYGULAMA 87
3.1. Makine Öğrenmesi ve Derin Öğrenme Yöntemlerinde Ortak Olan Özellikler 89
3.2. Akış Şemaları 97
3.3. Makine Öğrenmesi Yöntemi Uygulamaları 100
3.4. Derin Öğrenme Yöntemi Uygulamaları 106
4. BULGULAR 113
4.1. Makine Öğrenmesi Yöntemleri Eğitim Sonuçları 113
4.2. Derin Öğrenme Yöntemi Eğitim Sonuçları 133
4.3. Makine Öğrenmesi ve Derin Öğrenme Yöntemleri Sonuçları 139
5. DEĞERLENDİRME VE ÖNERİLER 143
KAYNAKLAR 146

ŞEKİLLER DİZİNİ

ŞEKİL Sayfa

1.1. Görüntünün sayısallaştırılması 9
1.2. Nesne tanıma ve sınıflandırma aşamaları 10
2.1. Geleneksel programlama mantığı 13
2.2. Makine öğrenmesi yöntemi ile programlama mantığı 13
2.3. Boyut azaltma teknikleri hiyerarşisi 15
2.4. Öznitelik seçme işlemi süreci akış diyagramı 17
2.5. İki düzeyli kategorik verilerden oluşan lojistik regresyon grafiği 22
2.6. k=3 olduğu durumda tahmin verisinin k-en yakın komşu algoritmasına göre koordinat düzleminde gösterilmesi 23
2.7. k-En yakın komşu algoritması ile sınıflandırma işlemi sonrası tahmin verisinin koordinat düzleminde gösterilmesi 23
2.8. Doğrusal verilen hiperdüzlem ile birbirinden ayrılması 25
2.9. Doğrusal olmayan veri setleri için oluşturulan hiperdüzlem 26
2.10. X ve Y olaylarının gerçekleşme olasılığı, X∩Y venn diyagramı 27
2.11. Karar ağacı örneği (DÖ: Derin Öğrenme, GPU: Graphichs Processing Unit) . 29
2.12. Rastgele orman ağaç yapısı örneği 33
2.13.Karar ağaçları akış diyagramı (a) Sınıflandırma ve regresyon, entropiye dayalı (b) Rastgele orman 34
2.14. Doğrusal regresyon grafiği 35
2.15. Gradyan iniş tekniği grafiksek gösterimi 36
2.16. Polinom derecelerine göre regresyon grafiği 37
2.17. Beş değişkenli bayes ağ modeli 38
2.18. Alarm sistemi bayes olasılık ağ modeli 39
2.19. k-En yakın komşu yöntemi örneği (a) K-ortalamalar algoritmasında küme sayısının bulunması (b) K-ortalamalar algoritması ile kümeleme yöntemi 42
2.20. Apriori algoritması ile birliktelik kuralı oluşturulması (a) Birliktelik kuralı akış şeması (b) Örnek akış şeması 44
2.21. Genetik Algoritma akış şeması 46
2.22. Hiyerarşik kümeleme yöntemi (a) Birleştirici (b) Ayırıcı 47

2.23. Hiyerarşik şemayı dedrograma dönüştürme işlemi (a) Hiyerarşik kümeleme şeması (b) Hiyerarşik kümeleme dendrogramı 47
2.24. Basit bir biyolojik nöron yapısı 49
2.25. Biyolojik nöron ağı ve yapay sinir ağı karşılaştırması (a) Biyolojik nöron (b) Yapay nöron (c) Biyolojik nöron ağı (d) Yapay sinir ağları 50
2.26. Algılayıcı modeli 51
2.27. Çok katmanlı sinir ağı mimarisi 54
2.28. Aşırı uyum regresyon grafiği 56
2.29. İleri ve geri yayılım ağ yapısı örnek akış şeması 57
2.30. Yapay zekâ, makine öğrenmesi ve derin öğrenme ilişkileri şeması 58
2.31. Konvolüsyon katmanı öznitelik seçme ve çıkarma işlemi 59
2.32. Konvolüsyon işlemi 60
2.33. Gradyan iniş yöntemi grafiği 63
2.34. İleri ve geri beslemeli ağlarda konvolüsyon katmanı 63
2.35. Doğrusal aktivasyon grafiği 67
2.36. Atlama fonksiyonu aktivasyon grafiği 68
2.37. Sigmoid aktivasyon fonksiyonu grafiği 69
2.38. Tanh aktivasyon fonksiyonu grafiği 70
2.39. ReLU aktivasyon fonksiyonu grafiği 71
2.40. Softmax aktivasyon fonksiyonu olasılık sonuçları gösterimi 72
2.41. Bir katmanlı basit evrişimsel sinir ağı mimarisi 73
2.42. Görüntü işlemede max ve average pooling katmanı uygulaması 74
2.43. Dropout katmanı modeli (a) İki gizli katmanlı sinir ağı (b) Dropout uygulaması ile elde edilmiş sinir ağı modeli 75
2.44. Yapay sinir ağı yapısı 76
2.45. Veri arttırma işlemleri için seçilen görüntü 77
2.46. featurewise_center parametresi uygulanmış görüntü 78
2.47. samplewise_center parametresi uygulanmış görüntü 78
2.48. featurewise_std_normalization uygulanmış görüntü 79
2.49. samplewise_std_normalization uygulanmış görüntü 79
2.50. rotation_range parametresi uygulanmış görüntü 80
2.51. zoom_range parametresi uygulanmış görüntü 80
2.52. width_shift_range parametresi uygulanmış görüntü 81

2.53. height_shift_range parametresi uygulanmış görüntü 81
2.54. horizontal_flip parametresi uygulanmış görüntü 82
2.55. vertical_flip parametresi uygulanmış görüntü 82
2.56. Veri arttırma parametreleri kod bloğu..... 83
2.57. ImageNet Büyük Ölçekli Görsel Tanıma yarışmasının yıllara göre hata oranlarının dağılımı..... 85
3.1. Jupyter Notebook arayüzü ekran görüntüsü python kodu 87
3.2. Tez süreci işlem adımları akış şeması..... 88
3.3. CIFAR-10 veri seti örnek görüntüler 90
3.4. MNIST veri seti örnek görüntüler..... 90
3.5. Çalışmada kullanılan kütüphaneler ve açıklamaları 95
3.6. Makine öğrenmesi yöntemleri akış şemaları (a) Akış şeması (b) CIFAR-10 veri seti ile uygulama süreci..... 98
3.7. Derin öğrenme yöntemi akış şemaları (a) Akış şeması (b) CIFAR-10 veri seti ile uygulama süreci 99
3.8. k-En yakın komşu yöntemi uygulanan CIFAR-10 veri seti [107]..... 103
3.9. AlexNet ve BasitNet modelleri veri arttırma kodu ve ekran çıktısı..... 107
3.10. Ağırlıkları kaydetme 108
4.1. Lojistik regresyon yöntemi sınıflandırma raporları (a) CIFAR-10 (b) MNIST 115
4.2. Lojistik regresyon yöntemi karmaşıklık matrisleri (a) CIFAR-10 (b) MNIST. 117
4.3. k-En yakın komşu yöntemi sınıflandırma raporları (a) CIFAR-10 (b) MNIST 120
4.4. k-En yakın komşu yöntemi karmaşıklık matrisleri (a) CIFAR-10 (b) MNIST 122
4.5. Naive bayes yöntemi sınıflandırma raporları (a) CIFAR-10 (b) MNIST 124
4.6. Naive bayes yöntemi karmaşıklık matrisleri (a) CIFAR-10 (b) MNIST 126
4.7. Rastgele orman yöntemi sınıflandırma raporları (a) CIFAR-10 (b) MNIST.... 130
4.8. Rastgele orman yöntemi karmaşıklık matrisleri (a) CIFAR-10 (b) MNIST..... 132
4.9. BasitNet modeli, 100 epoch süresince doğruluk oranları değişim grafiği (a) CIFAR-10 (b) MNIST 134
4.10. BasitNet modeli sınıflandırma raporları (a) CIFAR-10 (b) MNIST..... 135
4.11. BasitNet modeli karmaşıklık matrisi (a) CIFAR-10 (b) MNIST..... 137
4.12. BasitNet modeli nesne tahmin sonuçları (a) CIFAR-10 (b) MNIST 138
4.13. En yüksek doğruluk oranı elde edilen modellerin sonuçlarının grafiksel gösterimi 142

ÇİZELGELER DİZİNİ

ÇİZELGE Sayfa

1.1. Yapay zekanın gelişim süreci 4
2.1. Makine öğrenmesi yöntemleri..... 18
3.1. Test verilerindeki her bir sınıfa ait görüntü sayıları..... 91
3.2. Karmaşıklık matrisi..... 96
3.3. Lojistik regresyon yöntemi modelleri ve parametre değerleri 101
3.4. k-En yakın komşu yöntemi modelleri ve parametre değerleri 104
3.5. Karar ağacı yöntemi modelleri ve parametre değerleri a. Sınıflandırma ve regresyon, entropiye dayalı b. Rastgele orman 105
3.6. Veri setlerine uyarlanan AlexNet model özeti (a) CIFAR-10 (b) MNIST 109
3.7. Veri setlerine uyarlanan BasitNet model özeti (a) CIFAR-10 (b) MNIST..... 111
3.8. Derin öğrenme modelleri katman özellikleri 112
4.1. Lojistik regresyon yöntemi sınıflandırma sonuçları 114
4.2. Lojistik regresyon yöntemi sınıflandırma raporlarına göre yapılan değerlendirmeler 115
4.3. Lojistik regresyon yöntemi karmaşıklık matrislerine göre yapılan değerlendirmeler 117
4.4. k-En yakın komşu yöntemi sınıflandırma sonuçları (a) CIFAR-10 (b) MNIST 118
4.5. k-En yakın komşu yöntemi sınıflandırma raporlarına göre yapılan değerlendirmeler 121
4.6. k-En yakın komşu yöntemi karmaşıklık matrislerine göre yapılan değerlendirmeler 123
4.7. Naive bayes yönteminin veri setleri üzerindeki sınıflandırma tahmin sonuçları 123
4.8. Naive bayes yöntemi sınıflandırma raporlarına göre yapılan değerlendirmeler125
4.9. Naive bayes modeli karmaşıklık matrislerine göre yapılan değerlendirmeler.. 126
4.10. Karar ağacı yöntemi sınıflandırma tahmin sonuçları (a) Sınıflandırma ve regresyon (b) Entropiye dayalı (c) Rastgele orman 127

4.11. Rastgele orman yöntemi sınıflandırma raporlarına göre yapılan değerlendirmeler 131
4.12. Rastgele orman yöntemi karmaşıklık matrislerine göre yapılan değerlendirmeler 132
4.13. Derin öğrenme yöntemi sınıflandırma tahmin sonuçları 133
4.14. BasitNet modeli sınıflandırma raporlarına göre yapılan değerlendirmeler..... 136
4.15. BasitNet modeli karmaşıklık matrislerine göre yapılan değerlendirmeler 137
4.16. Makine ve derin öğrenme modelleri ile eğitilen CIFAR-10 ve MNIST veri setlerinin eğitim sonuçları ve süreleri 139
4.17. Makine öğrenmesi ve derin öğrenme modellerinden elde edilen sınıflandırma raporlarına göre en iyi ve en kötü tahmin edilen sınıflar 140
4.18. Makine ve derin öğrenme modellerinden elde edilen karmaşıklık matrislerine göre birbirine benzetilen sınıflar (a) CIFAR-10 (b) MNIST 141

KISALTMALAR DİZİNİ

SIFT	Scale-Invariant Feature Transform
SURF	Speeded-Up Robust Features
HOG	Histogram of Oriented Gradients
RGB	Red Green Blue
AID	Automatic Interaction Detection
CART	Classification And Regression Trees
GPU	Graphichs Processing Unit
ID3	Iterative Dichotomiser 3
WCSS	Within Clusters Sum of Square
IDC	International Data Corporation
ILSVRC	ImageNet Large Scale Visual Recognition Competition
ReLU	Rectified Linear Units

1. GİRİŞ

Teknolojinin gelişmesi ve internetin hayatımıza girmesi ile birlikte bilginin saklanması ve veri büyüklüğünün sürekli artması depolama sorununu beraberinde getirmiştir. Market araştırmaları merkezi olan International Data Corporation (IDC) Data Age 2025 araştırması raporuna [1] göre, 2016 yılındaki çevrimiçi ve çevrimdışı üretilen veri miktarı 16.1 zettabayttır. Aynı araştırmaya göre 2025 yılındaki veri miktarının 163 zettabayt (yaklaşık 1 trilyon gigabeyt)'a çıkacağı öngörülmektedir. Verilerin büyük çoğunluğu bulut hizmetinde saklanır veya internette yayınlanır. Tüm bu verileri etkili bir şekilde yönetebilmek için verilerin anlamlandırılması önemlidir.

Verilerin önemli bir kısmı görüntülerden oluşmaktadır. Görüntü içeriğinin otomatik olarak işlenmesi, çeşitli görevler için kullanışlıdır. Otomatik olarak işlenen veriler, görüntü dosyalarında saklanan piksel seviyesindeki bilgilerin anlaşılmasını sağlar. İnsan mantığının bu bilgileri işlemesi ve anlamlandırması zordur. Bilgisayar vizyonu bu aradaki köprüyü kurmaya çalışmaktadır [2]. Bu anlamda yapay zekâ sistemlerinin geliştirilmesi hem büyük miktardaki verinin insana göre daha hızlı anlamlandırılmasını, hem de hata oranı insana göre daha kabul edilebilir düzeylerde olmasını sağlar.

Makine öğrenmesi yöntemleri yapay zekâ tekniklerinden bir tanesidir. Makinenin ezberden ziyade, insani özelliklerden bir tanesi olan *öğrenme*yi gerçekleştirebileceği düşüncesi makine öğrenmesini ortaya çıkarmıştır. Makine öğrenmesi yöntemleri, yapay zekanın en önemli konusu olan insan-makine etkileşiminde diğer tekniklere oranla daha başarılıdır ve bilinen verileri öğrenerek bilinmeyen veriler üzerinden doğru tahminler yapmaya odaklanır. Makine öğrenmesi yöntemlerinden bir tanesi olan derin öğrenme yöntemi ise, insanın beyin yapısından yola çıkarak öğrenme tekniklerini modelleyen bir sistem olan yapay sinir ağlarını kullanır. Derin öğrenme mimarisi yapay sinir ağlarının ara (gizli) katman sayısının arttırılması, derinleştirilmesi esasına dayanır.

1.1. Yapay Zekâ

Zekâ, yaygın olarak, karmaşık problemleri çözmek için bilgi çıkarımı yapma veya bilgi birikimine sahip olma yeteneği olarak düşünülür. Yapay zekâ ise bilgi toplayabilen, iletişim kurabilen, güncellenebilen ve algılayabilen akıllı makinelerin ve yazılımların incelenmesi ve geliştirilmesidir. Yapay zekâ sistemlerinin amacı insan zekâsına en yakın tasarımı oluşturmaktır. Yani makineler veya yazılım tarafından sergilenen zekâ olarak da düşünülebilir. Yapay zekanın kalıcı, tutarlı, ucuz, çoğaltma ve yayma kolaylığına sahip olması, belgelendirilebilmesi ve belirli görevleri insandan daha hızlı yapabilmesi sayesinde doğal zekaya göre daha avantajlıdır [3].

Yapay zekanın ilk ortaya çıkışı II. Dünya Savaşı döneminde olmuştur. İngiliz Hükümeti tarafından ünlü matematikçi Alan Turing'e, içinde savaş strateji bilgilerinin olduğu Alman yapımı Enigma adı verilen makinenin şifrelerini çözme görevi verilmiştir. Alan Turing şifre kırıcı genel bir makine tasarlar ve tasarladığı bu makine için "Makineler düşünebilir mi?" sorusu üzerine odaklanır. Bu soru yapay zekaya temel oluşturacak önemli bir adımdır. Alan Turing bu konu üzerine yazdığı ilk makalesinde, makine ve düşünme terimlerinin yaygın kullanılan anlamlarından ziyade bu terimlerle yakından ilişkili, nispeten kesin olmayan ifadelerle anlamlandırmaya çalışmıştır. Ve düşünceyi somutlaştırmak adına da "Yapay Oyun"u geliştirmiştir. Kendisi tarafından geliştirilen bir yarışma olan ve Turing adı verilen bu yapay oyun ile makinenin kanı ve canı olan bir insan olduğuna bilirkişileri inandırmayı başarmıştır.

Yıllar önce Alan Turing'in [4] makalesinde yazdığı, "Önümüzde sadece kısa bir mesafe görebiliyoruz, ancak yapılması gereken çok şey var." cümlesi bugün de hala geçerliliğini korumaktadır. Diğer taraftan aynı yıllarda Jefferson [5] çalışmasında sinir iletimlerinden yola çıkarak insanın düşünce sitemini keşfetmeye çalışırken; bu düşüncenin makinelere entegre edilmesi hakkındaki öngörüleri hayali ve boş teoriler olarak görmüş ve bunun da makineyi antropomorfize[1] eden yeni ve daha büyük tehlike olduğunu söylemiştir.

[1] Herhangi bir şeyi insanlaştırmak, insani özellikleri herhangi bir şeye aktarmak anlamına gelir.

Yapay zekanın geliştirilmesinde bilgisayar oyunlarının önemli bir yeri vardır. DeepMind şirketi AlphaGo adlı bir yazılım geliştirdi ve bu yazılım ile Go oyununda şampiyon oyuncu olan Ki Cie'yi yendi. Aynı şirket daha sonra bu öğrenen algoritmayı genelleştirdi. AlphaZero adı verilen bu sürüm, satrancı da benzer bir biçimde kurallar ve amaç tanımlandıktan sonra öğrenebilecek duruma geldi. AlphaZero, herhangi bir insan girdisi olmadan öğrenme konusunda genelleştirilmiş bir makine öğrenimi algoritmasıdır. AlphaZero öncekinin aksine dışarıdan yardım almadan öğrenme işlemini gerçekleştirdi. AlphaZero dört saatlik bir öğrenme sürecinin ardından 2017 bilgisayar satranç şampiyonu Stockfish 28 galibiyet, 72 beraberlik ve sıfır kayıp gibi olağanüstü bir skorla kazandı. Öğrenmenin bir algoritma aracılığı ile genelleştirilmesi, sadece oyun alanında değil, aynı zamanda insan doğasının pek de önemli olmayan yönlerini ele alarak insanlığı daha iyi hale getirebileceğini gösterir. Sonuç olarak yapay zekâ çevrimiçi alışverişlerden arabalara, fotoğraflarımızı düzenlemekten günlük rutinlerimizin çoğunda yaygın hale gelmiştir [3].

1.1.1. Tarihçesi ve Gelişim Süreci

John McCarthy 1956'da IBM'in düzenlediği bir konferansta "Yapay Zekâ" terimini literatüre kazandırmıştır. İnsani özellikleri bir makineye aktarma fikri ise çok eskilere dayanan bir düşüncedir. Felsefi çalışmalarda bu düşüncelerin analizi yapılmış; bilimsel çalışmalarda ise uygulanabilirliği deneysel çalışmalarla test edilmiş ve piyasaya hazır hale getirilmeye çalışılmıştır. Bu tür düşünceler, her ne kadar herhangi bir problemi çözme odaklı olarak ortaya çıkmış olsa da süreç içerisinde genellenerek birden fazla problemi tasarlayacak ürünler ortaya çıkarmıştır.

Özellikle yapay zekâ alanında yapılan bir çalışma diğer çalışmalara ışık tutabileceği gibi yeni problemleri ve çözümlerini de beraberinde getirmektedir. Bu da yapay zekâ alanına süreklilik ve dinamizm kazandırmaktadır. Yakın gelecekte akıllı makineler, birçok alanda insan yeteneklerinin yerini alacaktır.

Yapay zekanın ilk ortaya çıkışından derin öğrenmeye kadar olan süreç Çizelge 1.1.'deki gibi tablolaştırılabilir.

Çizelge 1.1. Yapay zekanın gelişim süreci ([6]'dan değiştirilerek)

Yıl	Yapay Zekâ Alanındaki Gelişmeler
1914	• A. Torres y Quevedo, satranç oyununun son hamlesi için elektromekanik makineler üretti.
1921	• Çek yazar Karel Capek Robot kelimesini ilk kez "Rossum's Universal Robots" isimli oyununda kullandı.
1937	• Alan Turing evrensel soyut bir hesaplama yapabilen bir makine tasardı.
1938	• Berlin'deki Z1 bilgisayarını geliştiren Konrad Zuse, teknolojinin sonunda yapay bir beyin haline geleceğini öngördü.
1940	• İngiltere' deki Bletchley Park'ta, Turing ve diğerleri Alman istihbarat mesajlarının şifresini otomatik olarak çözen bir makine tasardı.
1941	• McCulloch ve Pitts, "Etkin Sinirlerde Düşüncelerin Mantıksal Bir Analizini" yayınladı. • Rosenblueth, Wiener ve Bigelow, sibernetik terimini tanıtan "Davranış, Amaç ve Teleoloji" yayınladı.
1944	• Eckert ve Mauchly tarafından ENIAC (Electronic Numerator, Integrator and Computer), Pennsylvania Üniversitesi'nde geliştirildi.
1945	• Vannevar Bush, Memex'i bilgi depolamak ve dağıtmak için otomatik bir araç olarak önerdi.
1946	• Turing, öncü kabul edilen "Zeki Makinalar" isimli yayımlanmamış bir makale yazdı.
1947	• Norbert Wiener Sibernetik kavramını kullandı. • Grey Walter elektromekanik bir yapısı olan "turtle"ı tasardı
1949	• Manchester Üniversitesi'nde kayıt yapabilen ilk bilgisayar Mark I tarafından geliştirildi. • Turing ve meslektaşları satranç oyununu programlamaya çalıştı.
1950	• Alan Turing, Turing Testini öneren "Bilgi İşlem Makineleri ve Zeka"yı yayınladı. • Isaac Asimov, Runaround'da bir dizi kural olan "Üç Robot Kuralı"nı tasarladı. • Shannon, satranç oyununun detaylı bir analizini yayınladı.
1951	• 1945'te John von Neumann tarafından önerilen IAS makinesi, Princeton İleri Araştırma Enstitüsü'nde çalıştırıldı, • Marvin Minsky and Dean Edmunds ilk yapay sinir ağı, 40 nöron ağı taklit etmek için 3000 vakum tüpü kullanılan SNARC (Stokastik Sinirsel Analog Güçlendirme Hesaplayıcısı) icat edildi.
1952	• Arthur Samuel, öğrenen ve yarışan bir dama makinesinde çalışmaya başladı.
1956	• John McCarthy'nin "YZ" terimini Dartmouth Konferansı'nda önerdi ve Newell, Shaw ve Simon, ilk çalışan AI programını, Mantık Teorisini sergiledi.
1958	• John McCarthy, YZ araştırmasında kullanılan en popüler programlama dili olan Lisp'i geliştirdi.
1959	• Arthur Samuel, "machine lerning" terimini ortaya attı.
1961	• T. Evans'ın ANALOGY programı IQ testlerinde görülen aynı analojik problemleri çözdü. • James Slagle SAINT (Symbolic Automatic INTegrator) geliştirdi. Bu program, birinci sınıf hesaplamada simgesel entegrasyon problemlerini çözen sezgisel bir programdır.

Çizelge 1.1. (devam) Yapay zekanın gelişim süreci ([6]'dan değiştirilerek)

1962	• J. C. R. Licklider, coğrafi konumu ne olursa olsun, bireyler arasındaki iş birliği, bilgi yayma, yayın ve etkileşim için dünya çapında bir ortam olan İnternet'in ilk adımlarını attı. DARPA desteğini başlattı ve sürdürdü.
1964	• Daniel Bobrow, "Bir Bilgisayar Sorun Çözme Sistemi için Doğal Dil Girişleri" başlıklı MIT doktora tezini tanımladı ve doğal bir dil anlama bilgisayar programı olan STUDENT geliştirildi.
1965	• Lotfi Zadeh bulanık mantığı icat etti. • Joseph Weizenbaum ELIZA'yı geliştirdi, her hangi bir konu üzerinde ki İngilizce diyologlarla ilgilenen ilgi çekici bir programdı. Weizenbaum, insanlar ve bilgisayarlar arasındaki iletişimin yüzeyselliğini göstermeye çalışırken bilgisayar programı gibi düşünen insanların sayısına çok şaşırmıştır.
1967	• DENDRAL, bilimsel akıl yürütme için ilk başarılı bilgi tabanlı program; MACSYMA, matematikte ilk başarılı bilgi tabanlı program geliştirildi. • Bilgi tabanlı bir satranç oyunu programı olan MacHack, C sınıfı bir derecelendirme turnuvası oyunu gerçekleştirdi. • İnteraktif bir öğrenme ortamı olan LOGO'nun ilk sürümü ortaya çıktı.
1969	• İnternetin öncüsü ARPANET kuruldu. • Arthur Bryson and Yu-Chi Ho çok katlı bir dinamik sistem optimizasyon yöntemi olarak geri yayılımı tanımladılar. Çok tabakalı yapay sinir ağları için bir öğrenme algoritması, bilgi işlem gücü büyük ağların eğitimini karşılamak için yeterince gelişmiş olduktan sonra, 2000'li ve 2010'lı yıllarda derin öğrenmenin başarısına önemli ölçüde katkıda bulunmuştur.
1978	• Moravec'in arabası ilk bilgisayar kontrollü otonom araçtır.
1981	• Japon Uluslararası Ticaret ve Sanayi Bakanlığı, Beşinci Nesil Bilgisayar projesi için 850 milyon dolar bütçe ayırdı. Proje, konuşmaları devam ettirebilen, dilleri tercüme edebilen, resimlerini yorumlayabilen ve insan gibi sebepleri bulabilen bilgisayarlar geliştirmeyi amaçladı.
1982	• Newell genel zekâ için bir mimari olan SOAR'ı tasarladı. • ABD, YZ hedeflerine ulaşmak için Stratejik Bilgi İşlem Projesine başladı.
1985	• R. Brooks, otonom reaktif robotların ilki olan "Allen" i tanıttı.
1989	• Yann LeCun el yazısı posta kodlarını tanıyan çok katmanlı bir sinir ağına başarılı bir geri bildirim algoritması uyguladı.
1980'lerin sonu	• "The AI Winter" dönemi yaşandı.
1990	• Human Genome Project (HGP) başlatıldı.
1995	• Richard Wallace, Joseph Weizenbaum'un ELIZA programından esinlenerek chatbot A.L.I.C.'yi (Yapay Dil İnternet Bilgisayar Kurumu) geliştirici sadece Web'in ortaya çıkışı ile mümkün olan benzeri görülmemiş ölçekte doğal dil örnekleri veri toplama özelliklerini eklendi.
1997	• Deep Blue, dünyanın bir numaralı satranç şampiyonu olan Garry Kasparov'u yenerek ilk bilgisayar satranç oyun programı oldu. • Sepp Hochreiter ve Jürgen bugün el yazısı ve konuşma tanımada kullanılan tekrarlayan sinir ağının bir türü olan Uzun ve Kısa Vadeli Hafıza (LSTM) fikrini ortaya attı.
2000	• Yapay olarak akıllı bir insan robotu olarak tasarlanan Honda'nın ASIMO robotu, bir insan gibi hızlı bir şekilde yürüyebilme, bir restoran ortamında tepsileri müşterilere sunabilme özelliklerine sahipti.

Çizelge 1.1. (devam) Yapay zekanın gelişim süreci ([6]'dan değiştirilerek)

2003	• DARPA üç büyük yapay zeka projesini başlattı.
2006	• Oren Etzioni, Michele Banko ve Michael Cafarella "okuyabilen makine," terimini ortaya attılar, bu robot otomatik olarak yazıyı anlayabilme özelliğine sahipti. • Geoffrey Hinton Çoklu öğrenme tanımını ortaya attı bu terim özetle derin öğrenmeye ilk bakış olmuştur.
2007	• Fei Fei Li ve Princeton Üniversitesi'ndeki meslektaşları, görsel nesne tanıma yazılımı araştırmasına yardımcı olmak için tasarlanmış açıklama görüntülerinin büyük bir veri tabanını ImageNet'i oluşturmaya başladılar.
2009	• Rajat Raina, Anand Madhavan ve Andrew Ng, "modern grafik işlemcilerinin çok çekirdekli işlemcilerin yetersiz kaldığını ve denetimsiz öğrenme yöntemlerinin uygulanabilirliğinin devrim potansiyeline sahip olduğunu" savunarak, "Grafik İşlemcilerini Kullanarak Büyük Ölçekli Denetimsiz Öğrenme"yi yayınladılar." • Northwestern Üniversitesi Akıllı Bilişim Laboratuvarı'ndaki bilgisayar bilimcileri, insan müdahalesi olmaksızın spor haberleri yazan Stats Monkey programını geliştirdi.
2011	• Bir konvolüsyonel sinir ağı, Alman Trafik İşareti Tanıma yarışmasını %99,46 doğrulukla (%99,22 insanla karşılaştırıldığında) kazandı. • İsviçre'deki IDSIA araştırmacıları, konvolüsyonel sinir ağları kullanarak el yazısı tanımada %0.27 hata oranını rapor ettiler, önceki yıllarda %0.35-%0.40 hata oranı üzerinde önemli bir gelişme olmuştu. • Watson, hızlı cevap verebilen bir konuşma makinesi yaptı ve bu makine iki dil şampiyonunu yendi.
2012	• Toronto Üniversitesi araştırmacıları tarafından tasarlanan konvolüsyonel sinir ağı, bir önceki yıla göre en iyi girişin elde ettiği %25'lik hata oranının üzerinde önemli bir iyileşme olan Büyük Ölçekli Görsel Tanıma Yarışması'nda yalnızca %16'lık bir hata oranı elde etti. • Apple Siri'yi tanıttı ve cihazlarında kullanmaya başladı. Siri kelimeleri anlama ve cevap verme konusunda çok başarılı olsa da ses tonundan duygu ve düşünceleri anlama konusunda üzerinde çalışmalar hala devam etmektedir.
2014	• Google YZ şirketi Deepmind'ı satın aldı. 400 milyon dolar değerindeki bu anlaşmanın sebebi YZ ile öğrenme (machine learning) ve sinir ağlarını harmanlayarak genel amaçlı öğrenme algoritmaları tasarlamaktı. Deepmind'ı satın alması Google'ın YZ konusunda üst düzey yetenekleri bünyesine katma hamlesi olarak da görülüyor. • Facebook yüz tanıma konusunda yüksek doğruluk oranları ile insan zekâsı seviyesine ulaştı. • Vicarious'ın sekiz kişilik, dahilerden oluşan takımı, CAPTCHA'yı çözebilen (CAPTCHA testleri insan ve bilgisayarları ayırt etmekte kullanılıyor.) bir teknoloji gelişmesi duyurdu. Vicarious'ın algoritmaları denemelerin yüzde 90'ında testi geçmeyi başarmıştı. Mark Zuckerberg, Ashton Kutcher ve Elon Musk bu şirkete 40 milyon dolar yatırım yaptı. • Google sürücüsüz araç geliştirmeye başladı. 2014 yılında, Nevada'da ABD eyaletinde kendi kendine sürüş testi yapan ilk araba oldu.
2015	• Maryland Üniversitesi yemek pişirmeyi öğrenen robotlar geliştirdi. • Google'ın Project Wing ekibi Skydio dronelar için 'Yapay Zekâ' kullandı.
2016	• Ars Technica senaryosu tamamen yapay zekâ tarafından yazılan ilk kısa filmi yayınladılar. • Apple yüz ifadesi analizi yapan yapay zekâ yazılım şirketi olan Emotient'i satın aldı.

Çizelge 1.1. (devam) Yapay zekanın gelişim süreci ([6]'dan değiştirilerek)

2017	• NVIDIA PilotNet adını verdiği ve insanları gözlemleyerek bir otomobili sürmeyi öğrenen, sinir ağı tabanlı bir sistem geliştirdi. Bu sistemle yetinmeyen NVIDIA, sürüş kararları alırken öncelik sıralamasının ne olduğunu söyleyen bir yöntem geliştirmeyi başardı. Ağ, öncelik sıralamasını bu metoda göre yöneticilerle paylaşıyor [7], • DeepMind araştırmacıları geliştirdikleri yeni algoritmayla yapay zekaya hafıza eklemeyi başardılar.
2018	• Blockchain tabanlı bir yapay zekâ SingulartyNET, anti merkezcil bir yapay zekâ platformu. Gerzeli ve ekibi, görüntü tanımlamadan doğal dil işlemeye kadar birçok yapay zekâ algoritmasını etkin bir şekilde çalıştırabilen blok zincir tabanlı bir alt yapı oluşturmayı planlıyor. Sistem ayrıca hangi algoritmaların daha çok kullanıldığını takip etmek ve geliştiricilerin buna göre hareket etmesini sağlamak içinde kullanılabilecek.

1.1.2. Uygulama Alanları

Yapay zekanın çeşitli uygulama alanları;

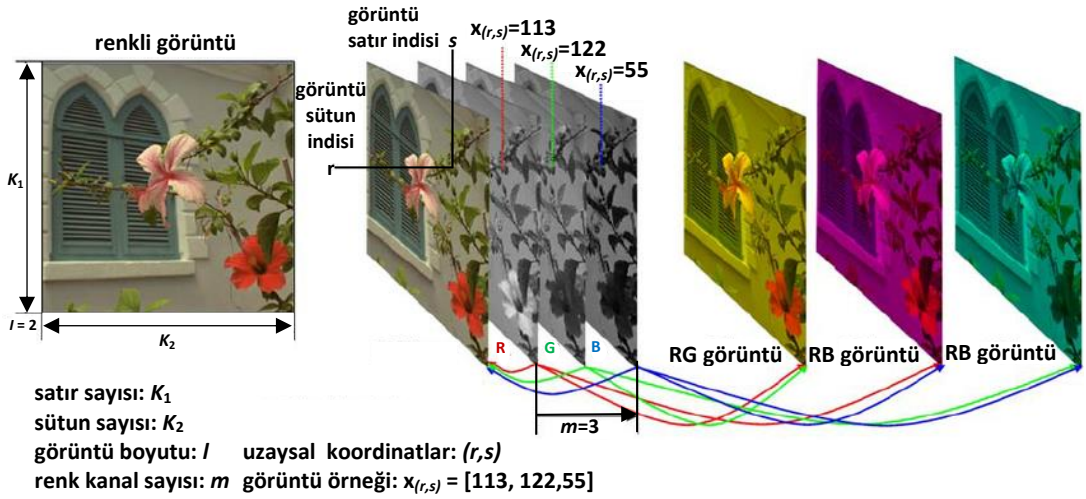
- Bulanık mantık,
- Tıpta evrimsel hesaplamalar,
- Hastane hasta bakımını iyileştirme,
- Tıbbi görüntü sınıflandırması,
- Tıbbi tanı,
- Muhasebe veri tabanları,
- Bilgisayar oyunları,
- Doğal dil işleme,
- Robotik ve duyusal sistemler,
- Havacılık, konuşma anlama,
- Bilgisayarlı görü ve nesne tanıma,
- Bilgisayar destekli öğretim,
- Nöral bilişim,
- İstatistik hukuk şeklinde sıralanabilir [8].

Yapay zekâ uygulamalarında uygulanan çeşitli teknikler ise sinir ağı, bulanık mantık, evrimsel hesaplama ve hibrittir.

Yapay zekâ ile birlikte insan-makine etkileşimi birçok alanda büyük değişimleri beraberinde getirmiştir. Bu etkileşimin içinde bulunduğu çağ, sibernetik çağ olarak adlandırılır. Genel itibariyle sibernetik; insan-makine arasında etkileşim kuran, bütün bilim dalları için ortak olan iletişim-kontrol süreçlerini içine alan, geri besleme için pratikler oluşturan mekanik sistem olarak tanımlanabilir [9].

1.2. Görüntü İşleme

Görüntüler karmaşık olma eğilimindedirler. Her ne kadar insan tarafından anlaşılması kolay olsa da bir bilgisayar modeline tanıtmak nispeten daha zordur. Ancak insanın görme sistemini oluşturan temel mekanizmanın anlaşılması görüntü işleme için önemlidir. Çünkü insan görme sistemi sayısal görüntü işleme sistemi olarak düşünülebilir. Görüntü işleme, görüntüden bilgi elde etmek veya görüntünün alternatif bir temsilini üretmek amacıyla bir görüntünün manipülasyonu olarak tanımlanabilir. Görüntülerin bilgisayar ortamında değerlendirilebilmeleri için görüntülerin bilgisayarın işlem yapabileceği hale dönüştürülmeleri gerekir. Bu işleme sayısallaştırma adı verilir. Görüntüdeki ışık, mikroişlemciler aracılığıyla önce elektriksel veriye daha sonra da sayısallaştırılarak sayısal veriye dönüştürülür [10]. Sayısallaştırılan görüntüler, *m* satır ve *n* sütundan oluşan *mxn* boyutunda bir matris görünümündedir. Siyah-beyaz, gri ve renkli olmak üzere üç tane görüntü türü vardır. Yalnızca siyah ve beyaz piksellerden oluşan görüntüler siyah-beyaz, gri tonlardan oluşan görüntüler gri tonlamalı ve RGB (Kırmızı, Yeşil, Mavi) katmanından oluşan görüntüler renkli görüntüler olarak sınıflandırılır. Şekil 1.1'de renkli görüntüye ait 0-255 arası üçlü tam sayılardan oluşan renkli görüntü matrisi verilmiştir.



Şekil 1.1. Görüntünün sayısallaştırılması [11]

Görüntü işleme iki önemli amaç için yapılabilir. Bunlar [12]:

(i) görüntüyü bozan, istenmeyen sinyal bileşenlerini kaldırmak

(ii) daha açık veya daha kullanışlı bir biçimde oluşturarak bilgi çıkarmak

Görüntünün değerlendirilebilmesi için, görüntüden doğru bilgi çıkarımının yapılması gerekir. Bu işlem öznitelik çıkarma işlemi olarak da tanımlanır. Görüntüdeki pikseller aracılığı ile öznitelik çıkarma işlemi gerçekleştirilir. Görüntünün karmaşıklıkları ve boyutsallıkları nedeniyle [13], görüntülerin içerisindeki uzaysal koordinatları bulmak ve öznitelik çıkarımı yapmak kolay değildir. Gabor filtresi, SIFT (Scale-Invariant Feature Transform), SURF (Speeded-Up Robust Features), Feature Matching ve HOG (Histogram of Oriented Gradients) gibi yöntemleri uygulanır. Görüntü işlemede bilgisayarlı görünün genel işlevi aşağıdaki aşamalarla özetlenebilir [14]:

• Görüntü yakalama- Görüntü yakalanır (kamera veya benzeri bir cihazla) ve dijitalleştirilir.

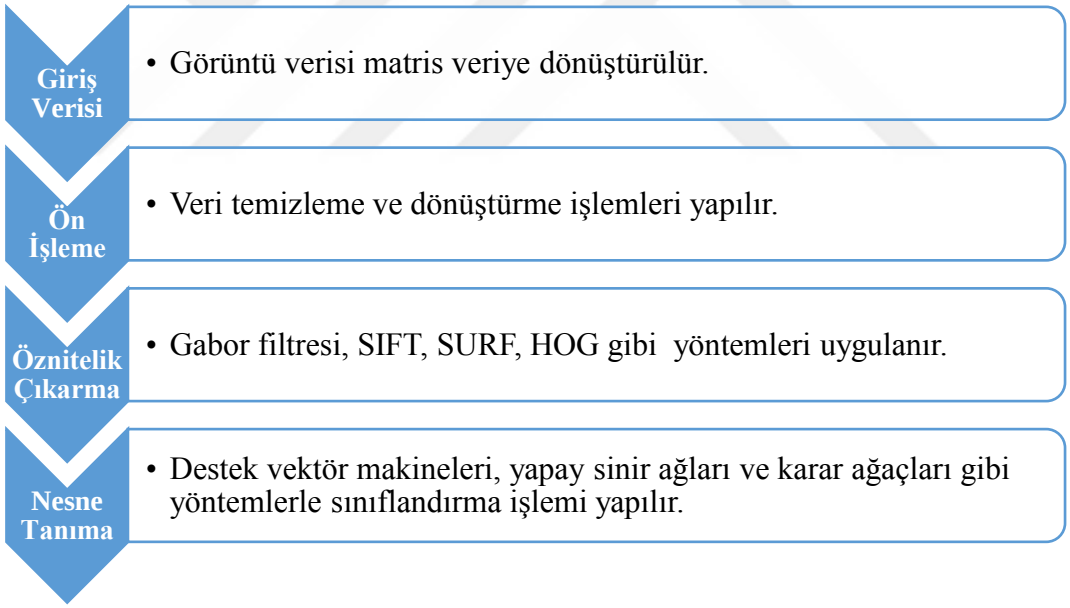
• Ön işleme- Sayısallaştırılmış görüntü önemli özellikleri vurgulamak için değiştirilir. (Örneğin, gürültü azaltma, kontrast normalleştirme gibi).

• Segmentasyon- İlginç öznitelikleri seçimi (kenarlar, benzer yüzeyler gibi).

1.3. Nesne Tanıma

Görme olayında nesnelerin ne olduklarının belirlenmesine nesne tanıma denir. Bir nesneyi tanımak, onu bir sınıflandırma içerisinde değerlendirmek anlamına gelir [15]. Görüntünün sınıflandırılması için de görüntü üzerinden renk, şekil gibi yüksek düzeyde bilgiler çıkartılabilecek görüntü işlemenin gerçekleştirilmesi gerekir. Trafik işaretleri, medikal görüntüleme, karakter tanıma, yüz tanıma ve parmak izi tanıma gibi birçok alanda yapılan çalışmalar nesne tanıma kullanım alanlarına örnek gösterilebilir.

Nesne tanıma süreci Şekil 1.2'de gösterildiği gibi gerçekleşir. Görüntüler sayısallaştırılarak matris verilere dönüştürülür. Sınıflandırmaya hazır hale getirilebilmesi için ön işlemeye tabi tutulur. Öznitelik çıkarma işlemi yapıldıktan sonra sınıflandırma yöntemlerinden birisi kullanılarak nesne tanıma işlemi gerçekleştirilir.



Şekil 1.2. Nesne tanıma ve sınıflandırma aşamaları

1.4. Literatür Çalışmaları

CIFAR-10 ve MNIST veri setleri literatürde oldukça sık kullanılmıştır. Veri setlerinin gerçek görüntülerden oluşması, ön işlem gerektirmemesi ve farklı yazılım platformlarında kütüphane desteği ile kolayca eklenebilmesi sayesinde farklı yöntemlerde tercih edilmiştir. Ayrıca yöntemler aynı çalışmada farklı veri setleri üzerinde uygulanarak karşılaştırılması sağlanmıştır.

Norouziy vd. [16] CIFAR-10 ve MNIST veri setleri üzerinde k-en yakın komşu yöntemi ile hamming mesafe ölçüsü kullanmışlardır. Basit yapıda oluşturdukları model ile diğer karmaşık modellerin performansına yakın bir doğruluk oranı elde etmişlerdir. Goyal vd. [17] çalışmasında, temiz ve gürültülü verileri karşılaştırılarak karmaşıklık matrisinde bilgi kaybını engellemek için bir hesaplama önermiştir. En iyi performansı naive bayes yönteminden elde etmişlerdir.

Abouelnaga vd. [18], CIFAR-10 veri seti üzerinde farklı sınıflandırıcıların performanslarını incelemişlerdir. k-En yakın komşu ve evrişimsel sinir ağlarının farklı sınıflarda iyi performans gösterdiklerini tespit etmişler ve iki sınıflandırıcıyı bir araya getirerek geliştirilen algoritmanın hata oranını düşürdüğünü tespit etmiştir. Liu vd. [19], makine öğrenmesi yönteminde öznitelik çıkarımı için SIFT, HOG, GIST gibi yöntemleri kullanmak yerine AlexNet, VGG19 ve ResNet modellerindeki gizli katmanlarda bulunan öznitelikleri kullanmışlardır. Öznitelikleri çıkarılan veriler üzerinde naive bayes yöntemi uygulanmıştır. Aynı şekilde Li vd. [20] de derin sinir ağları ile birlikte naive bayes yöntemini kullanmıştır.

Graham [21], CIFAR-10 veri setinde pooling katmanında ayrık ve örtüşen pooling işlemlerini karşılaştırmışlar ve ağırlık matrisi için kesirli sayılar kullanmışlardır. Springenberg vd. [22] pooling ve aktivasyon işlemlerinin doğruluk oranına etkisini tespit etmeye çalışmışlar ve oluşturulan modelin basit yapıda olmasını amaçlamışlardır. Ayrıca çalışmada, pooling işleminin aşırı uyumu azalttığı tespit edilmiştir. Mishkin vd. [23] aktivasyon fonksiyonlarının öğrenmeye etkisini test etmişlerdir. Ayrıca ağırlık başlatma için basit bir yöntem önermişler ve karmaşık derin ağlardan daha iyi sonuçlar verdiğini saptamışlardır.

2. MATERYAL VE YÖNTEM

2.1. Makine Öğrenmesi

Yapay zekâ alanında öğrenme kavramı zeki sistemler için önemlidir. Zekânın var olabilmesi, belirli bir eşik değere kadar öğrenmenin gerçekleşmesine bağlıdır. Makine öğrenmesi araştırması, fizyoloji, bilişsel bilim yoluyla insanlığın çalışma mekanizmasına göre hesaplama modelini veya anlama (kavrama) modelini kurar, her türlü çalışma teorisini ve çalışma yöntemini geliştirir, genel algoritmayı inceler ve teorik olarak analiz eder. İstenilen göreve göre özel çalışma sistemini kurar.

Makine kabiliyetinin insanın ötesine geçip geçemeyeceğini düşünen pek çok kişinin ana argümanı, makinelerin insan yapımı olmalarından dolayı, performans ve hareketlerini yine insanların belirleyeceği şeklindedir. Bu nedenle makinenin kabiliyeti tasarımcıyı herhangi bir şekilde geçemez. Bu görüş, öğrenme yeteneğine sahip olmayan makineler için doğrudur, ancak makine öğrenmesi için dikkate değerdir [24]. Bu tür bir makinenin yeteneği uygulamada sürekli olarak artırılabildiği için, zaman geçtikçe tasarımcı bile makinenin öğrenme seviyesini bilmeyecektir. Bu açıdan makine öğrenmesi yöntemleri, yapay zekâ araştırmasında son derece önemli bir konuma sahiptir.

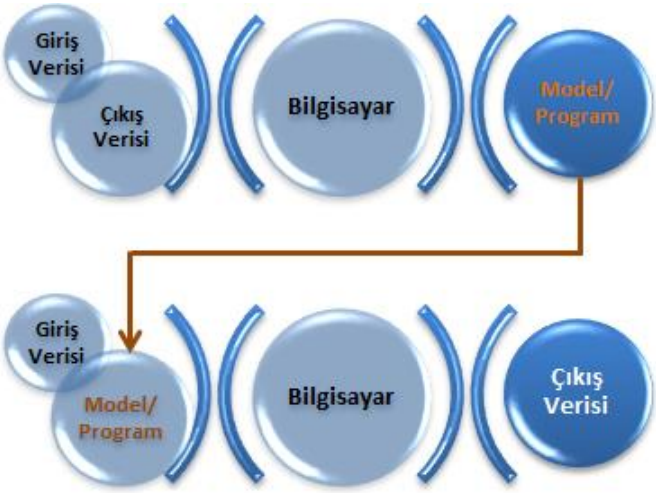
İnsanın etkileşimde olduğu hemen hemen her alanda veriler sürekli olarak değişmektedir. Verinin boyutu arttıkça insan tarafından anlamlandırılması zorlaşmaktadır. Makine öğrenmesi yöntemleri için ise veri boyutunun büyük olması avantajlı bir durumdur. Makine öğrenmesi yöntemleri yeni verilere adaptasyon sürecinde kolaylık sağlar. Karar verme süreci hızlı olması ile birlikte doğru sonuçlar sunar. Genel karar verme kapasitesini arttıran gelişmiş algoritmalar kullanır. Bu da yenilikçi iş hizmetlerine ve modelleri geliştirmeye yardımcı olur.

Şekil 2.1'de görüldüğü üzere geleneksel programlamanın çalışma mantığı, giriş verilerinden çıkış verileri elde edilmesine yöneliktir. Çıkış verilerinin elde edilebilmesi için problem türüne uygun program veya yazılım kullanılmalıdır.

Şekil 2.1. Geleneksel programlama mantığı

Makine öğrenmesi çalışma mantığında ise giriş ve çıkış verileri ile makine eğitilerek problem türüne uygun program veya yazılım elde edilir. Bu program veya yazılım yeni veriler üzerinde test edilir ve çıkarımda bulunulur. Yeni giriş verilerinden doğru veya hatalı çıkış verileri elde edilir ve yüzde olarak ifade edilir. Bu yüzdelik oranlarına doğruluk ve hata oranı denir. Doğruluk veya hata oranı ile modelin geçerliliği ölçülür. Şekil 2.2'de makine öğrenmesi çalışma mantığı gösterilmektedir.

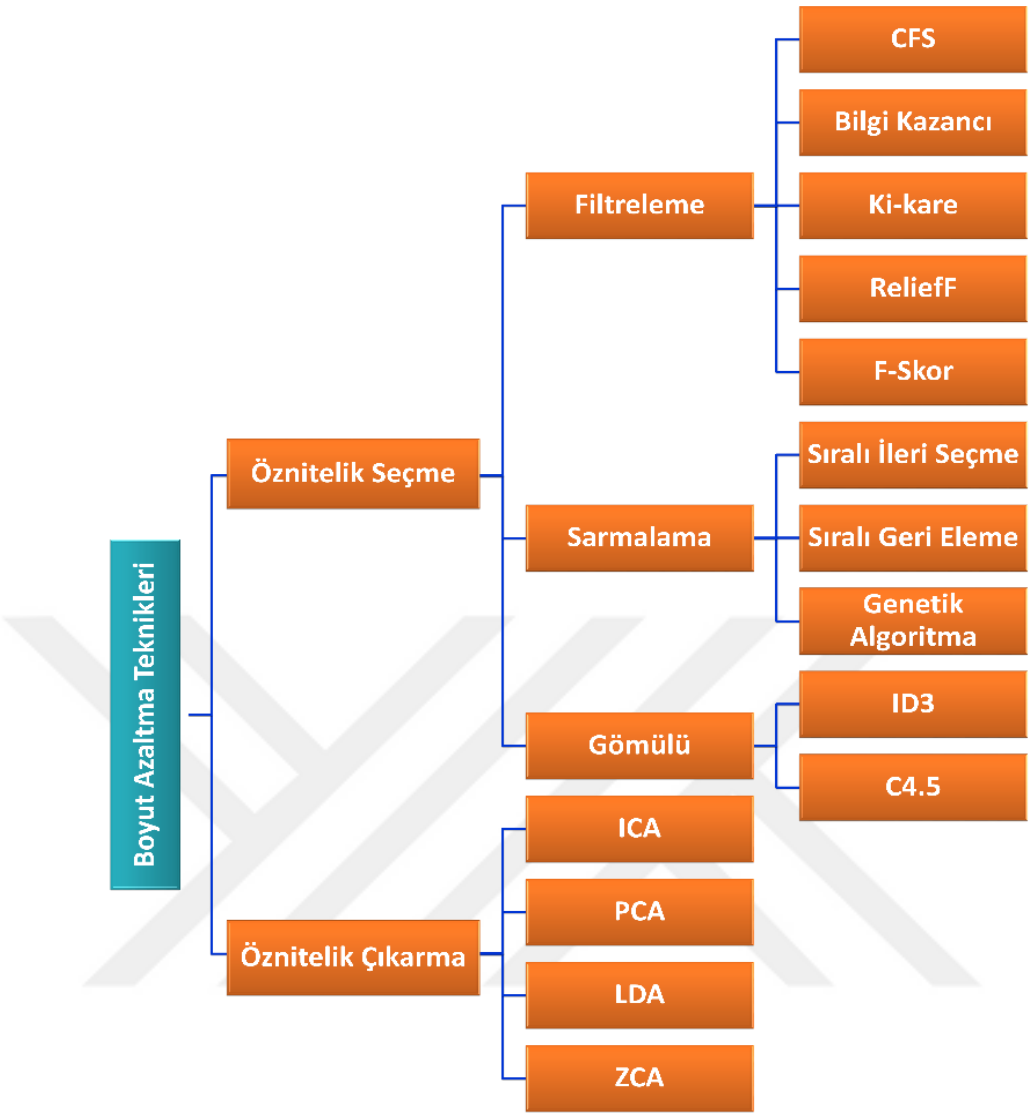
Makine öğrenmesi yöntemlerinde ortak olarak kullanılan 2 kavram vardır: eğitim ve test verileri. Eğitim verileri üzerinde makine öğrenmesi yöntemleri kullanılarak yeni bir model oluşturulur. Oluşturulan model, test verisi üzerinde uygulanarak hata oranı tespit edilir ve modelin performansı ölçülür.



Şekil 2.2. Makine öğrenmesi yöntemi ile programlama mantığı

2.2. Öznitelik Çıkarma ve Seçme İşlemleri

Veriler üzerinden doğru çıkarımlar yapılabilmesi için verilerin ayırt edilebilirliğini sağlayan öznitelikler adı verilen özelliklerin tespit edilmesi gerekir. Veriler üzerinden doğru özelliklerin çıkarılması işlemine öznitelik çıkarma denir. Öznitelik çıkarma işlemi, mevcut veriler daha küçük boyutlu bir alt uzaya dönüştürülür ve özellikler kaybolmadan küçük boyutta öznitelik kümesinin oluşturulması sağlanmış olur. Çıkarılan özniteliklerden doğru tahminler yapılmasını sağlayan özniteliklerin belirlenmesi işlemine öznitelik seçme denir. Öznitelik seçme işlemi ile çıkarılan öznitelikler dönüştürülmeden küçük boyutlu alt uzay oluşturulur. Böylece veri boyutu da azaltılmış olur. Her iki işlemde de amaç, dönüştürülen bilgileri kullanarak görüntüleri iyileştirmek ve tahmin sistemlerinde doğru çıkarımlar yapabilmektir. Bir diğer ifadeyle öğrenme algoritmalarına en fazla katkıda bulunabilecek en düşük boyutta bir alt uzayın oluşturulmasını sağlamak olarak da düşünülebilir. Her iki işlem için de en sık kullanılan yöntemler Şekil 2.3'te verilmiştir.



Şekil 2.3. Boyut azaltma teknikleri hiyerarşisi

Oluşan öznitelik kümesi öznitelik haritası olarak da adlandırılır. Öznitelik haritası oluşturulurken orijinal, özgün bilgilerin kaybedilmemesi önemlidir [25]. Yorumlanamayan ve kaybedilen orijinal bilgiler tahmin algoritmalarının doğruluk oranlarını etkiler.

2.2.2. Öznitelik Çıkarımı

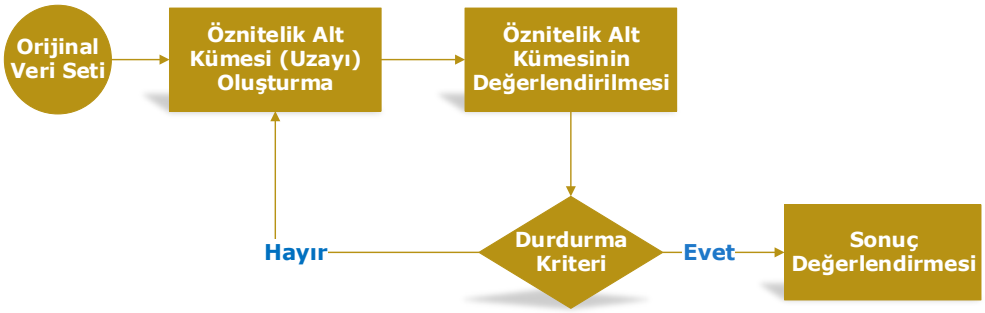
Herhangi bir problemin makine öğrenmesi yöntemleri ile çözülebilmesi için mevcut veriyi temsil eden öznitelik çıkarma işleminin yapılması ve öznitelik haritasının oluşturulması gerekir. Öznitelik haritasının sınıfları en iyi ayırabilecek şekilde oluşturulması beklenir [26]. Öznitelik çıkarma işlemi ile tüm öznitelikler çıkartılarak eğitilecek veri setinin boyutu küçültülmüş olur. Bu durum eğitim sürecini hızlandırabilir. Veya modelin daha az performansla daha hızlı bir şekilde sonuç elde etmesini sağlayabilir. Ancak tüm özniteliklerin boyutlarının aynı anda indirgenmesi sınıfları ayıran özniteliklerin tespit edilememesine sebep olur.

2.2.1. Öznitelik Seçimi

Öznitelik seçme işlemi, Şekil 2.3'te görüldüğü gibi filtreleme, sarmalama ve gömülü olmak üzere üç farklı yöntemle yapılmaktadır:

- **Filtreleme** : Filtreleme modeli, temel olarak istatistik hesaplamalar yaparak her bir özelliğin sonuca olan katkısını puanlar. Puanlar üzerinden eşikleme yaparak öznitelik haritası oluşturur. Herhangi bir sınıflandırıcı kullanmaz.
- **Sarmalama** : Sarmalama modeli, oluşturulan öznitelik haritasını bir sınıflandırıcı ile eğiterek modelin uygunluk değerini hesaplar.
- **Gömülü** : Gömülü modelde, sınıflandırıcı, kendi öznitelik haritasını oluşturur.

Öznitelik seçme işlemi sürecini oluşturan dört temel adım, Şekil 2.4'teki akış diyagramında gösterilmiştir. Bunlar; öznitelik alt kümesinin oluşturulması, öznitelik alt kümesinin değerlendirilmesi, durdurma kriteri, sonuç değerlendirmesi şeklinde sıralanır.



Şekil 2.4. Öznitelik seçme işlemi süreci akış diyagramı [27]

Şekil 2.4'te verilen öznitelik seçme işlemi 4 adımda özetlenebilir:

Adım 1. Öznitelik Alt Kümesi Oluşturma: Üç farklı yöntem ile öznitelik alt kümesi oluşturulur [28].

- Sezgisel fonksiyonlar kullanılarak
- Her seferinde bir öznitelik ekleyerek veya kaldırarak,
- Rastgele olarak öznitelik seçerek öznitelik alt kümesi oluşturulabilir.

Adım 2. Öznitelik Alt Kümesinin Değerlendirilmesi: Oluşturulan alt kümenin uygunluk değerlendirmesi yapılır. Öznitelik alt kümesinin uygunluğu, filtreme modelinde, modelin kendine özgü yöntemleri ile tespit edilirken sarmal modelde sınıflandırıcının eğitilmesi ve değerlendirilmesi ile tespit edilir. Bu adımdaki amaç öznitelik kümenin sınıflandırıcı ile uyumunu artırmaktır.

Adım 3. Durdurma Kriteri: Öznitelik kümesinin tamamlanmasının ya da durdurulmasının kriteri seçilen modele göre değişkenlik gösterir. İterasyon sayısının tamamlanması, öznitelik ekleme-kaldırma işleminin tamamlanması veya uygun öznitelik kümesinin oluşturulamaması durdurma kriteri olarak gösterilebilir.

Adım 4. Sonuç değerlendirmesi: Sınıflandırıcının performansı hata oranı ile ölçülür. Öznitelik seçme işlemi yapılmamış sınıflandırıcı ile öznitelik seçme işlemi yapılmış sınıflandırıcı performansı karşılaştırılır.

2.3. Makine Öğrenmesi Yöntemleri

Elektronik yayınlar, dijital kütüphaneler, elektronik kitaplar, e-posta mesajları, haber makaleleri ve web sayfaları gibi elektronik metin bilgilerinin sayısı hızla artmaktadır. Ancak, çevrimiçi metin bilgisinin hacmi arttıkça, ilgili bilginin çıkarılması zorluğu da artar [29]. Bu kaynakları bulmaya, filtrelemeye ve yönetmeye teşvik eden araçlara duyulan ihtiyaç artmıştır. Böylece, metin belgesi koleksiyonlarının otomatik organizasyonu önemli bir araştırma konusu haline gelmiştir. Metin verilerinin otomatik organizasyonunu geliştirmek için bir dizi makine öğrenmesi yöntemleri önerilmiştir.

Makine öğrenmesi yöntemleri, Çizelge 2.1'de gösterildiği gibi denetimli ve denetimsiz olmak üzere iki ana kategoride gruplanabilir. Hangi sınıfa ait olduğu belli olan veriler etiketli, belli olmayan veriler etiketsiz veriler olarak adlandırılır. Üzerinde eğitim yapılacak verilerin etiketli veya etiketsiz olma durumuna göre denetimli veya denetimsiz öğrenme yöntemlerinden bir tanesi kullanılır. Kullanılan yöntemler problemin türüne göre değişkenlik göstermektedir. En düşük hata oranı veren yöntem en uygun yöntem olarak gösterilebilir.

Çizelge 2.1. Makine öğrenmesi yöntemleri

Yapay Zekâ		
Denetimli Öğrenme		**Denetimsiz Öğrenme**
Sınıflandırma	**Regresyon**	**Kümeleme**
Lojistik Regresyon	Doğrusal Regresyon	K-Ortalamalar
K-En Yakın Komşu	Polinom Regresyon	Apriori/Birliktelik
Destek Vektör Makineleri	Bayes Ağları	Genetik
Naive Bayes		Hiyerarşik Kümeleme
Karar Ağaçları		Yapay Sinir Ağları
Yapay Sinir Ağları		

2.4. Denetimli Öğrenme

Denetimli öğrenme makine öğrenme yöntemlerinin en yaygın biçimidir. Gerçek hayatta hangi sınıfa ait olduğu belli olan (etiketli) ve belli olmayan (etiketsiz) veriler vardır. Denetimli öğrenmenin amacı etiketli veriler eğitilerek etiketsiz verilerin doğru olarak sınıflandırılmasıdır. Denetimli öğrenme ile her biri kendi sınıfıyla etiketlenmiş veriler makine öğrenme yöntemleri ile eğitilir oluşturulan model ile yeni verilerin istenen çıktıya en yakın sonucu üretilmesi amaçlanır. Model ile elde edilen sonuç ve istenen sonuç arasındaki fark hata oranı olarak nitelendirilir ve bu oran en aza indirilmeye çalışılır. Böylece etiketsiz veriler en yüksek doğruluk oranı ile sınıflandırılmış olur.

Denetimli öğrenme metotları Çizelge 2.1'de belirtildiği gibi sınıflandırma ve regresyon olmak üzere 2 başlık altında incelenir. Sınıflandırma yöntemlerinde çıktı veriler ayrık sayılabilir bir kümeden oluşur. Regresyonda ise sürekli değişen veya sürekli eklenen veriler üzerinden eğitim yapılır. Bu iki metodu somut bir örnekle açıklamak gerekirse; stok marketin verimi öğrenmek istenildiğinde sınıflandırma yöntemlerinden birisi kullanılır. Bu verimin artması sonucun +1 (artı bir) veya azalması sonucun -1 (eksi bir)'e yaklaşması durumu olarak nitelendirilebilir.

Stok marketin veriminin ne kadar arttığı veya azaldığı ise regresyon problemi olarak görülebilir. Bazı durumlarda artım ve azalım miktarının az olduğu aralık stok market için etkisiz kabul edilebilir. Bu aralığın üstünde veya altında bir değer çıkması durumunda sistemin sinyal vermesi istenebilir. Artım +1; azalım -1 ile ifade edilirse artım ve azalım miktarının az olduğu aralık 0 (sıfır) olarak ifade edilebilir. Bu tür problemler Regresyon yöntemleri ile çözülmeye çalışılır. Bir diğer örnekle gelen iletinin gereksizlere (spam) düşmesi veya gelen kutusuna düşmesi sınıflandırma problemi iken sosyal medya reklamlarının herhangi bir firmanın satışına etkisini tahmin etmek ise regresyon problemi olarak düşünülebilir.

2.4.1.Sınıflandırma Tahmin Yöntemleri

Nesneler üzerinde farklı görüntü değerlerinin tespiti yapılarak belirlenen özniteliklerin ilgili gruba veya sınıfa atama işlemine sınıflandırma denir. Sınıflandırma ile elde edilen görüntüler öznitelik haritası olarak isimlendirilirler [30]. Görüntü işlemede ise öncelikle öznitelik çıkartma ve seçme işlemi uygulanarak bir sınıfı diğer sınıftan ayırt edecek piksel gruplarına göre etiketleme işlemi yapılır. Bu yüzden aynı görüntüye ait veri sayısının çok olması sınıflandırma işlemini kolaylaştırır. Herhangi bir f fonksiyonu sınıflandırma tahmin modeli olarak düşünüldüğünde f(x)=y denkleminde x için girdi veriler, y için ise çıktı verilerdir. f(x) fonksiyonu herhangi bir sınıflandırma yöntemini temsil edebilir.

2.4.1.1. Lojistik Regresyon

Makine öğrenesinin sınıflandırma metotlarından birisi olan lojistik regresyonda bağımlı ve bağımsız değişkenler arasındaki korelasyonu çözümlemek ve ortaya çıkarmak amaçlanır. Buradaki girdi verisi bağımsız değişken olarak kabul edilirken çıktı verisi bağımlı değişken olarak kabul edilir. Aralarındaki bu korelasyonun yüksek olması beklenir. Elde edilen bu model sayesinde yeni verilerde en doğru tahminleri yapmak mümkün olmaktadır. Buradaki bağımlı değişen kategorili sırasız (rüzgârlı, güneşli, yağmurlu) olabileceği gibi kategorili sıralı (sıcak, ılık, soğuk) da olabilir. Bağımsız değişkenler bağımlı değişkenin yordayıcısıdır ve nominal, ordinal, aralık veya oran ölçeğinde ölçülebilir [31].

Gösterilen herhangi bir fotoğraf için cinsiyet tespiti yapılmak istendiğinde, saç, kaş, göz, ağız ve burun yapısının cinsiyet belirleme üzerindeki etkisinin bulunması lojistik regresyon problemidir. Bu örnekteki cinsiyet sonuç ve etkilenen değişken olup bağımlı değişkendir. Saç, kaş, göz, ağız ve burun yapısı ise açıklayan ve etkileyen değişken olan bağımsız değişkendir. Saç, kaş, ağız ve burun yapısına bakılarak cinsiyet durumu tahmin edilmeye çalışılır. Bağımsız değişkenlerin bağımlı değişkenler üzerine etkisi olasılıksal olarak ifade edilir. Bu oranlar başarı veya başarısızlık olasılığı olarak düşünülebilir. Başarı veya başarısızlık oranlarının birbirine oranlarının logaritması,

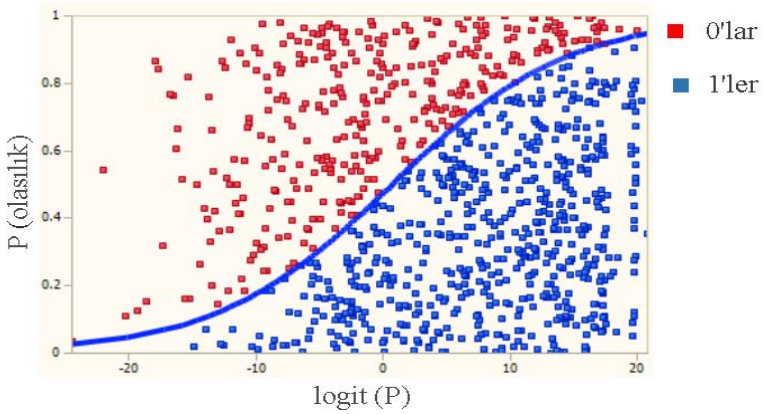
lojistik regresyon formülünü verir. Bahis oranı (odds) adı verilen logit fonksiyon 2.1 eşitliğinde verilmiştir.

$$\text{logit (P)} = \log \frac{\text{P}}{1 - \text{P}} \tag{2.1}$$

Bahis oranlarının 0 (sıfır) ve 1(bir)'e yakınlık durumlarına göre değerlendirme yapılır. Bahis oranının 1'e yakın çıkması durumu, bağımsız değişkenlerin bağımlı değişken üzerinde önemli bir katkısı olmadığını gösterir. Bahis oranının 1'den büyük çıkması ilgili bağımsız değişkenlerin bağımlı değişken üzerinde önemli bir katkısı olduğunu gösterir. 0'a yakın bir değere sahip olması ise bağımsız değişkenin bağımlı değişken üzerinde negatif bir etkiye sahip olduğunu gösterir. X değerlerini ise bağımsız değişkenler olarak düşünüldüğünde, bağımsız değişkenler X = (X_1, X_2, X_3, Xm) vektörü ile gösterilebilir. Bağımsız değişkenlerin regresyon katsayıları β ile ifade edilirse genelleştirilmiş regresyon logit fonksiyonu Eşitlik 2.2'teki gibi formüle edilir.

$$\text{logit (P)} = \log \frac{\text{P}}{1-\text{P}} = \beta_0 + \beta_1 \text{X}_1 + \ldots + \beta_m \text{X}_m \tag{2.2}$$

Fonksiyondaki P (olasılık) değeri 0 ile 1 arasında bir değer alabilir. P değeri arttıkça logit (P) değeri de artar. P değeri 0,5'ten küçük olduğunda, logit (P) değeri de 0'dan küçük olur. P değeri 0,5'ten büyük olduğunda logit (P) değeri de 0'dan büyük olur. Grafiksel gösterimi Şekil 2.5'teki gibidir. Şekilde 0 (kırmızı) ve 1 (mavi) olmak üzere 2 düzeyli kategorik verilerden oluşan LR grafiği verilmiştir.

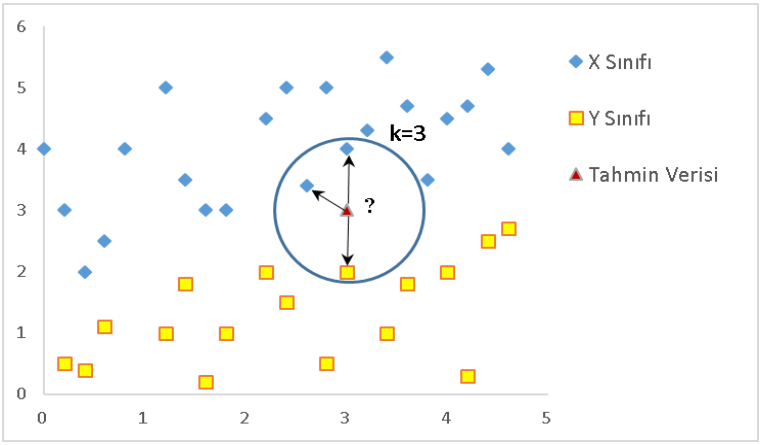


Şekil 2.5. İki düzeyli kategorik verilerden oluşan lojistik regresyon grafiği

2.4.1.2. k-En Yakın Komşu

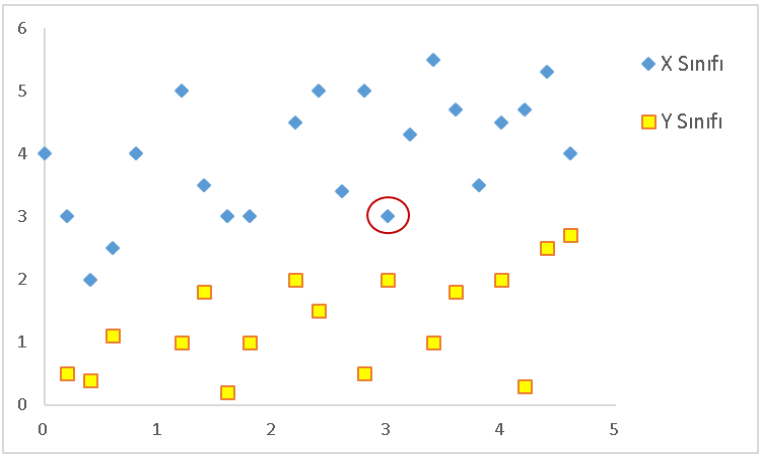
k-En en yakın komşu sınıflandırması, ilk kez Fix ve Hodes [32], tarafından öne sürülen, örüntü tanıma veya sınıflandırma problemlerinde sıkça kullanılan [33, 34] makine öğrenmesi yöntemlerinden bir tanesidir. k-En en yakın komşu algoritmasında verilerin birbiri arasındaki ilişkiler dikkate alınarak sınıflandırma işlemi yapılır. Bu sistem doğrusal ayrıştırma yöntemi ile koordinat düzlemi üzerinde çalışır. Orjinal k-en en yakın komşu algoritmasında sınıflandırılacak nesne, kendisine en yakın k tane komşularına göre çoğunluğun dahil olduğu sınıfa atanır [33].

Örneğin k=3 için sınıflandırma yapılmak istendiğinde önceden sınıflandırılmış 3 tane veriden çoğunluğun dahil olduğu sınıfa atama işlemi gerçekleştirilir. İlerleyen süreçlerde algoritma Dudani [35] tarafından geliştirilerek, k tane en yakın komşularının ağırlıklarını ve birbirleri arasındaki mesafesini dikkate alan model önerilmiştir. Şekil 2.6'da iki düzeyli kategorik verilerden oluşan, koordinat düzlemi verilmiştir. Herhangi X ve Y sınıflarına ait verilerin olduğu düzlemde komşu sayısı 3 (k=3) olarak belirlenmiştir. Şekil 2.6'da tahmin verisinin, kendisine en yakın 3 tane veriye göre konumu şekildeki gibidir.



Şekil 2.6. k=3 olduğu durumda tahmin verisinin k-en yakın komşu algoritmasına göre koordinat düzleminde gösterilmesi

Kendisine en yakın komşularından çoğunluğun dahil olduğu sınıfa atandığı bir sınıflandırma işlemi yapıldığı düşünülürse, tahmin verisine en yakın 3 veriden 2'si X sınıfına aittir. Bu durumda tahmin verisi X sınıfına atanır. Yapılan sınıflandırma işlemi sonrası son durum Şekil 2.7'deki gibi olur.



Şekil 2.7. k-En yakın komşu algoritması ile sınıflandırma işlemi sonrası tahmin verisinin koordinat düzleminde gösterilmesi

k-En en yakın komşu algoritmasında tahmin verisine en yakın k tane verinin uzaklıkları, uzaklık formülü ile bulunur. N boyutlu bir koordinat düzleminde birbirine komşu olan iki nokta (*x, y*) arasındaki uzaklık öklid, manhattan ve minkowski gibi yöntemlerinden birisi ile hesaplanır. Denklemler sırasıyla Eşitlik 2.3, 2.4 ve 2.5'te verilmiştir.

Öklid $:d(x,y) = \sqrt{\sum_{i=1}^{k}(x_i - y_i)^2}$ (2.3)

Manhattan $:\mathrm{d}(x,y) = \sum_{i=1}^{k}|x_i - y_i|$ (2.4)

Minkowski $:d(x,y) = \left(\sum_{i=1}^{k}|x_i - y_i|^n\right)^{1/n}$ (2.5)

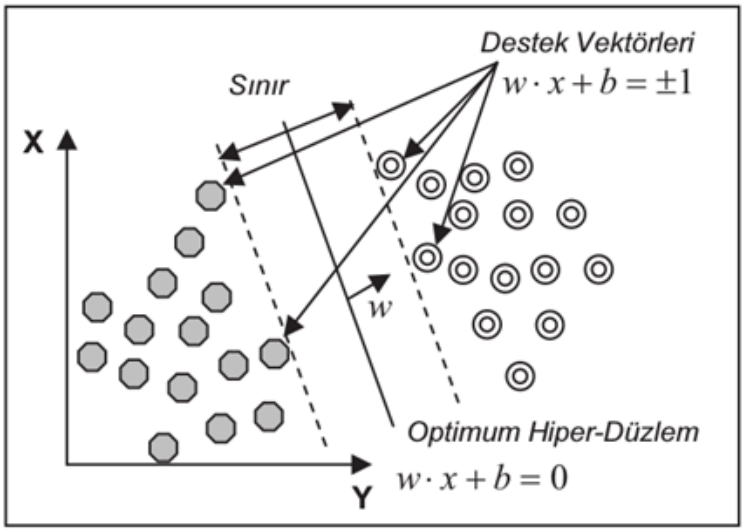
2.4.1.3. Destek Vektör Makineleri

Destek vektör makineleri hem sınıflandırma hem de regresyon problemleri için kullanılabilen makine öğrenmesi yöntemlerinden bir tanesidir. Ancak, çoğunlukla sınıflandırma problemlerinde kullanılır. Bu algoritmada, eğitim verisi üzerinde eğitim yapılarak yeni veriler üzerinde doğru tahminler yapılmaya çalışılır. Eğitim verileri N boyutlu uzayda bir nokta olarak çizilir. (N, özniteliklerin sayısıdır). Bu noktaların her birisi koordinat değerlerine sahiptir. Destek vektörleri ile eğitim verileri eğitilerek karar fonksiyonları oluşturulur. Elde edilen karar fonksiyonu yardımıyla iki veya daha fazla sınıfa ait veriler hiperdüzlem üzerinde gösterilir [36]. Hiperdüzlem iki veya daha fazla sınıfın birbirinden ayrılmasını sağlar.

Örneğin iki sınıflı bir düzlemde, her sınıf için ayrı ayrı kendisine en yakın olacak şekilde çizgiler çizilir. Çizgilere en yakın verilere destek vektörleri denir. Bu destek vektörler birbirine yaklaştırılarak her iki sınıfı birbirinden ayıran ortak bir sınır çizgisi çizilir. Bu şartları sağlayan birden fazla çizgi çizilir. Destek vektör makinelerindeki amaç yeni verilerle karşılaşıldığında hatayı en aza indirecek uygun sınır çizgisini çizmektir. Sınır çizgisi Eşitlik 2.6'da verilen eğim formülü ile oluşturulur.

$$\mathrm{f(x)} = \mathrm{wx} + \mathrm{b} \quad (2.6)$$

Formüldeki; *w*: Hiperdüzlemin normal vektörü (ağırlık vektörü), *x*: girdi verileri, veri noktaları, *b*: Hiperdüzleme dik olan normal vektörünün eğilim değeri, f*(x)*: veri noktalarının ait olduğu sınıflar (-1, +1). Destek vektör makinelerindeki amaç, hiperdüzlemin oluşturulabilmesi için giriş verileri ve ait olduğu sınıf verileri yardımıyla en uygun *w* ve *b* değerlerini bulmaktır. Bulunan formül ile oluşturulan hiperdüzlemde sınıflandırma işlemi yapılır. İki kategorili bir sınıflandırma problemi için oluşturulmuş hiperdüzlemin genel gösterimi Şekil 2.8'de verilmiştir.



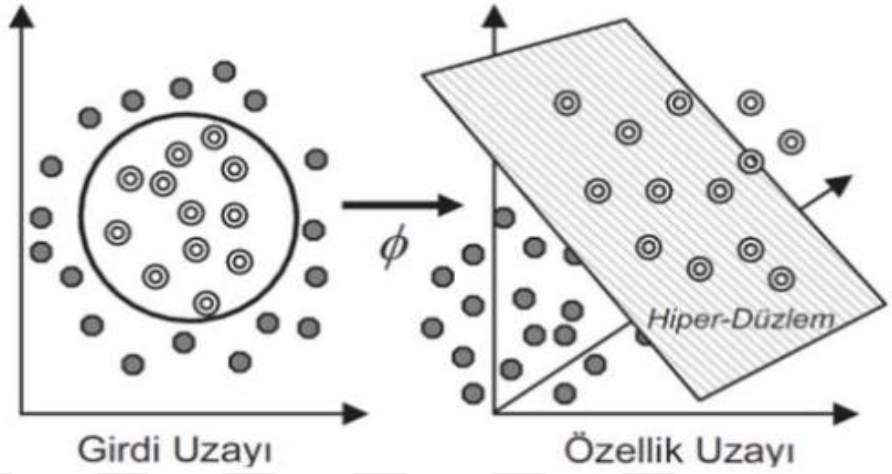
Şekil 2.8. Doğrusal verilen hiperdüzlem ile birbirinden ayrılması [30]

Veriler her zaman doğrusal olarak ayrılmayabilir. Böyle durumlarda iki boyutlu uzaydaki veriler doğrusal olmayan haritalama yaklaşımı ile üç boyutlu uzaya aktarılır. İki boyutlu olan girdi vektörleri çok boyutlu uzay vektörlerine dönüştürülür. İki boyutlu (x, y) bir koordinat düzlemindeki girdi vektörünün (x_1, y_1) üç boyutlu (x, y, z) uzaydaki dönüşümü eşitlik 2.7'deki gibidir.

$$\Phi : R^2 \rightarrow R^3 \quad (x_1, y_1) \rightarrow (x, y, z) \quad \Phi\ (x_1, y_1) = x_1^2, \sqrt{2}x_1 y_1, y_1^2 \qquad (2.7)$$

Böylece doğrusal olarak ayrılamayan veri seti iki boyutlu bir uzaydan üç boyutlu bir uzaya aktarılarak doğrusal haritalama işlemi gerçekleştirilir ve hiperdüzlem

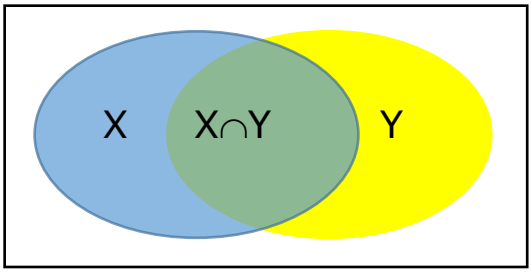
oluşturulur. Şekil 2.9'da iki boyutlu uzaydan üç boyutlu uzaya aktarılan verilerin hiperdüzlem yardımıyla sınıflandırılması gösterilmiştir.



Şekil 2.9. Doğrusal olmayan veri setleri için oluşturulan hiperdüzlem [37]

2.4.1.4. Naive Bayes

Naive bayes sınıflandırıcılar, basit yapıda olmasına karşın çok verimli olarak bilinen doğrusal sınıflandırıcılardır. Naive bayes sınıflandırıcılarının modeli bayes teoremine dayanmaktadır [38]. Bayes teoremi bağımlı değişkenler üzerinden hesaplamalar yapmaya dayanır. Hesaplamalar koşullu olasılıklar üzerinden yapılır. Koşullu olasılık ise bir durumun daha önce gerçekleşmiş bir duruma bağlı olarak gerçekleşme olasılığını temsil eder. Örneğin, futbol maçının galibiyet ve mağlubiyet durumları; maçın oynandığı gün, yer, seyirci sayısı ve hava durumuna göre tahmin edilebilir. Kendi sahasında, seyircisiz ve yağmurlu bir havada oynanan maçın galibiyetle sonuçlanma olasılığı daha önce oynanmış maçlara bakılarak tahmin edilebilir. Birbirinden bağımsız ve birbirinin ardı sıra gerçekleşen X ve Y olaylarından, X olayı gerçekleştiğinde Y olayının da gerçekleşme olasılığı P (X∩Y) ile ifade edilir ve şemasal gösterimi Şekil 2.10'daki gibidir.



Şekil 2.10. X ve Y olaylarının gerçekleşme olasılığı, X∩Y venn diyagramı

Şekil 2.10'da verilen venn diyagramından elde edilen eşitlik 2.8'deki gibidir.

$$P(X \cap Y) = P(X|Y)P(Y) = P(Y|X)P(X) \tag{2.8}$$

P(X) ifadesi problem durumunun girdi olasılığı, P(Y) ifadesi olası çıktı olasılığı, P(Y|X) ifadesi ise gerçekleşmiş olan X girdi durumuna bağlı olarak Y çıktı durumu olasılığını gösterir. P(Y|X) ifadesine karşılık gelen denklem 2.8'de verilen eşitlikten elde edilir ve eşitlik 2.9'daki gibidir.

$$P(Y|X) = \frac{P(X|Y)P(Y)}{P(X)} \tag{2.9}$$

P(X|Y): Y olayının gerçekleştiği sırada, X olayının gerçekleşme olasılığı
P(Y|X): X olayının gerçekleştiği sırada, Y olayının gerçekleşme olasılığı
P(X) : X olayının gerçekleşme olasılığı
P(Y): Y olayının gerçekleşme olasılığı

Bayes sınıflandırıcının mantığı basitçe bütün koşullu olasılıkların çarpımıdır. Elde edilen genel denklem Eşitlik 2.10'daki gibidir.

$$P(Y|X_1, \dots X_n) = \frac{P(X_1, \dots X_n|Y)\ P(X)}{P(X_1, \dots X_n)} \tag{2.10}$$

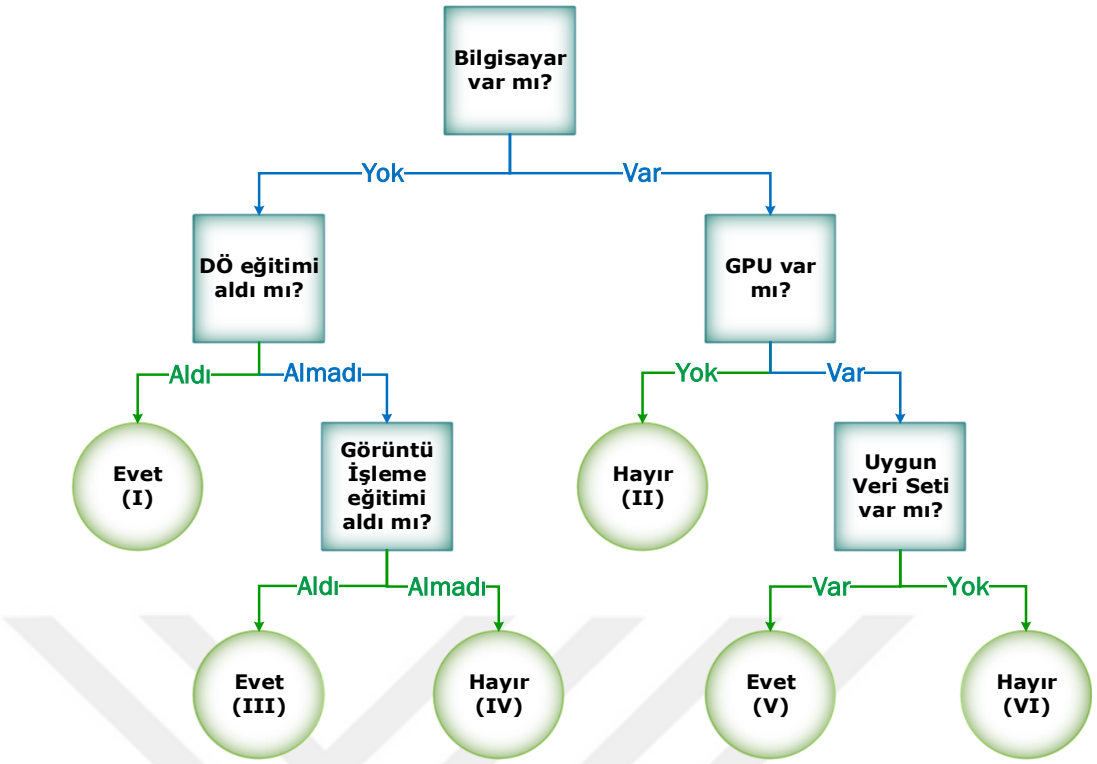
2.3.1.5. Karar Ağaçları

Karar ağacı algoritması ilk kez Morgan ve Sonquist [39] tarafından geliştirilmiştir. İlk olarak AID (Automatic Interaction Detection) adı verilen karar ağacı algoritması ile hata karesi toplamına göre alt bölünmeleri veya homojen grupları belirleyen ikili ağaçlar oluşturulmuştur. Daha sonra sırasıyla sınıflama ve regresyon ağaçları olan CART algoritmaları ve entropiye dayalı olan ID3, C4.5 algoritmaları geliştirilmiştir. Karar Ağaçlarının yapısı iki önemli kısımdan oluşur [40]:

1. Yapraklar: Sınıfları temsil eder.
2. Karar düğümleri (öznitelikler): Veriden çıkarılan özniteliklere göre ölçümler yapılarak uygun bölünmeleri temsil eder.

Karar ağaçlarının yapısı tipik bir ağaca benzetilebilir. Kullanılacak veri yığını kök düğüm olarak kabul edilir. Algoritmalara göre dallanma gerçekleştirilerek homojen yapıda düğümler elde edilmeye çalışılır. Düğümlerin her birisi dal olarak adlandırıldığında en son düğüm yaprak olarak adlandırılır. Yapraklar oluşturulan yapı için en homojen verilerin olduğu kısımdır. Ağacın büyümesinin temel amacı, sınıfı homojen alt sınıflara ayırmaktır. Ağaç oluşturma süreci sırasında, her bir düğümde optimal değişkeni elde etmek için bağımlı değişkeni bölüştürmek önemlidir. Bu şekilde dengeli veya doymuş bir ağaç elde edilir.

Karar ağacı, homojen gruplar oluşturulana kadar kök düğüme eklenen karar düğümlerin birleşmesinden oluşur. Şekil 2.11'de bir karar ağacı örneği verilmiştir. Bir şirket müdürü, şirket personellerinin derin öğrenme çalışmaları yapmalarında nelerin etkin rol oynadığını tespit etmek için personele uyguladığı anketler sonucunda Şekil 2.11'deki gibi bir karar ağacı oluşturmaktadır. Oluşturulan karar ağacında karar düğümleri (öznitelikler) mavi kare şekiller ile gösterilirken homojen gruplar yeşil yuvarlak şekiller ile gösterilmektedir. Kriterlere göre oluşturulan homojen gruplar roma rakamları ile belirtilmiştir.



Şekil 2.11. Karar ağacı örneği (DÖ: Derin Öğrenme, GPU: Graphichs Processing Unit)

Personelin belirtilen kriterlere göre derin öğrenme uygulaması yapıp yapmama durumu Şekil 2.11'e göre aşağıdaki gibi özetlenebilir:

I. Personelin bilgisayarı varsa ve derin öğrenme eğitimi almışsa derin öğrenme uygulaması yapmaktadır.

II. Personelin bilgisayarı var fakat GPUdonanımı yoksa derin öğrenme uygulaması yapmamaktadır.

III. Personelin bilgisayarı yoksa; derin öğrenme eğitimi almamışsa ve görüntü işleme eğitimi almışsa derin öğrenme uygulaması yapmaktadır.

IV. Personelin bilgisayarı yoksa; derin öğrenme eğitimi ve görüntü işleme eğitimlerini almamışsa derin öğrenme uygulaması yapmamaktadır.

V. Personelin bilgisayarı, GPU donanımı ve uygun veri seti varsa derin öğrenme uygulaması yapmaktadır.

VI. Personelin bilgisayarı, GPU donanımı var fakat uygun veri seti yoksa derin öğrenme uygulaması yapmamaktadır.

Karar Ağaçları problem durumlarına göre farklı algoritmalar kullanılarak oluşturulabilir. Karar Ağaçları kullanılan algoritmalara göre iki şekilde sınıflandırılır: Sınıflandırma ve regresyon karar ağaçları ve Entropiye dayalı karar ağaçları.

2.Sınıflandırma ve Regresyon Karar Ağaçları (CART)

CART algoritmaları 1984 yılında Breiman [41] tarafından geliştirilmiştir. Regresyon adı verilen yöntem ile belirli aralıktaki değerleri analiz ederek sınıflandırma yapılır. Aynı zamanda bu algoritmalar sürekli tipteki veriler üzerinde çalışır.

İstenen ağaç yapısının oluşturulması için dallanmayı belirleyen kriter veya özelliğin doğru belirlenmesi önemlidir. Karar düğümleri karar ağacının performansını için önemlidir. Bu yüzden sınıflandırma veya regresyon problemleri için geliştirilmiş Karar ağaçlarında ayrılma kriteri önemlidir. Bunlardan twoing kuralı ve gini indeksi en çok kullanılan karar ağacı kriterleridir.

a) Twoing Kuralı

İsminden de anlaşılacağı üzere bu algoritmada ilk olarak veriler iki sınıfa bölünür [42]. Twoing kuralının amacı optimal bir ayırma kuralı türetmektir. Algoritmada ikiye bölünme işlemi gerçekleştirilse de ikiden fazla değişkenin olduğu durumlarda her bir değişken diğer değişkenlerle karşılaştırılarak düğüm oluşturulabilirliği tespit edilmeye çalışır. Yani tüm eşleşmeler dikkate alınır. Böylece dinamik bir veri setinde hangi değişken değişirse değişsin algoritma istenilen yönde hesaplama yapacaktır. Algoritmada tüm kriterler için uygunluk değerleri bulunur. Bulunan en yüksek uygunluk değerine sahip düğüm dallanmayı gerçekleştirecek düğüm olarak seçilir. Örneğin hava durumunun güneşli olduğu günlerde pikniğe gidildiği ve yağmurlu olduğu günlerde ise gidilmediği varsayıldığında oluşturulan karar ağacında 1 düğüm 2 ihtimal (sınıf) vardır. Hava durumunun güneşli ve bulutlu olduğu günlerde pikniğe gidildiği ve yağmurlu olduğu günlerde gidilmediği varsayıldığında ise yine 1 düğüm 2 ihtimal (sınıf) vardır. Yani her durumda düğümler ikiye bölünme eğilimindedir.

b) Gini Kuralı

Bir Gini Kuralı, bölünmenin ne kadar iyi olduğuna ve sınıfların bölünme tarafından oluşturulan iki grupta ne kadar heterojen olduğuna dair bir fikir verir. Uygun düğümlerin belirlenebilmesi için tüm özniteliklerin gini indeksleri hesaplanır. Verilere ait öznitelikler aynı sınıfı gösteriyorsa Gini indeksi 0 (sıfır)'a eşit olacaktır. Belirli bir sınıfı temsil etmiyorsa Gini indeksi en yüksek değerini alacaktır. Dolayısı ile gini indeksi en düşük olan kriter karar düğümü olarak belirlenir. Gini algoritmasında amaç mümkün olan en kısa adımda en büyük ve homojen veri kümesini elde etmektir. İstenen düğüm elde edildiğinde bir sonraki adıma kadar heterojen veriler izole edilir.

2.3.1.5.2. Entropiye Dayalı Karar Ağaçları

Entropiye dayalı olan ilk sezgisel hesaplamalar Fayyad ve Irani [43] tarafından geliştirilmiştir. CART algoritmaları her düğümde bir kritere göre ikili bölünmeler yapar. İki veya daha fazla bölünmelerde ve karmaşık problemlerde entropiye dayalı algoritmalar kullanılmaktadır. Entropi gürültü içerisindeki anlamlı bilginin çıkarması miktarını ve karmaşıklığını ölçmek için 1948 yılında Shannon tarafından geliştirilmiştir. Entropi herhangi bir sistem için belirsizlik ölçütüdür. Karar Ağaçları için karar vermeyi etkileyecek olan olasılık değerlerini bulur. Entropi, bir rastgele değişkenin değerini tahmin ederken belirsizliği sayısallaştırır [44]. Belirsizliğin en az olduğu değişken üzerinden bölümleme yapılır.

a) ID3 (Iterative Dichotomiser 3) Algoritması

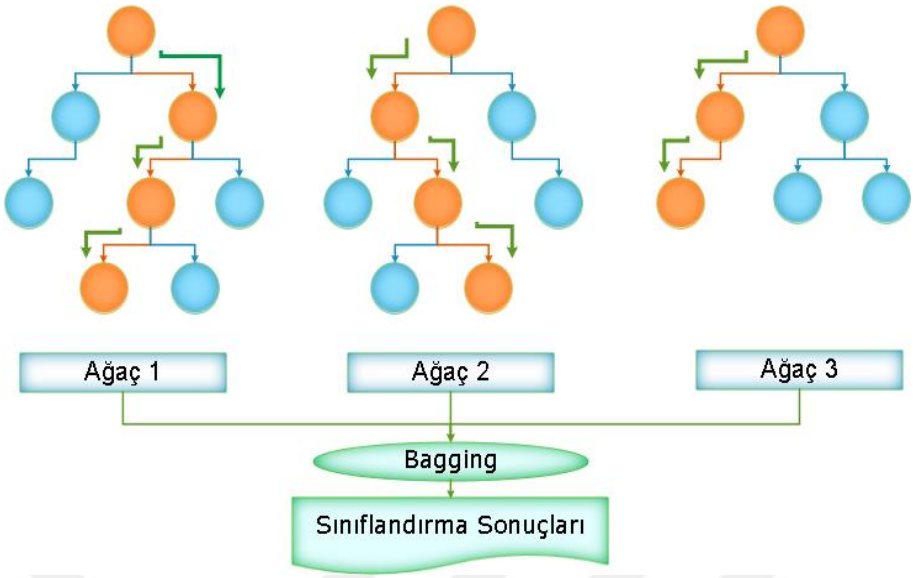
J. Ross Quinlan [45] tarafından 1979 yılında geliştirilmiştir. ID3, her adımda bir alt kümenin bir değerlendirmesi olan yinelemeli bir algoritmadır. Entropi adı verilen metriğe dayanarak karar düğümü oluşturulur [46]. Böylece, en yüksek bilgi kazancına sahip olan öznitelik (maksimum seviyedeki entropi azaltımı) dallanmayı gerçekleştirecek düğüm olarak seçilebilir [47]. Bu işlemler oluşturulacak her düğüm için tekrarlanır. Böylece alt düğümlerdeki heterojenlik en aza indirilir ve elde edilen en alt düğüm mümkün olduğunca aynı sınıfı belirten özniteliklerden oluşur.

b) C4.5 Algoritması

C4.5 algoritması, dezavantajlarının üstesinden gelmek için kullanılan ID3 algoritmasının bir uzantısıdır [48]. Bu iyileştirmeler, sayısal öznitelikler, eksik değerler, gürültülü veriler ve ağaçlardan kural oluşturma ile ilgili yöntemleri içerir [49]. Bir karar ağacı oluşturma genel süreci şu şekildedir. Bir dizi eğitim verisi göz önüne alındığında, en iyi ayırma özniteliğini bulmak için tüm özniteliklere bir ölçüm işlevi (entropi) uygulanır. Ayrılma özniteliği belirlendikten sonra, örnek uzayı çeşitli bölümlere ayrılır. Her bölüm içinde, tüm eğitim örnekleri tek bir sınıfa aitse, algoritma sona erer. Aksi takdirde, bölme işlemi, tüm bölüm aynı sınıfa atanana kadar ardışık olarak gerçekleştirilir. Bir karar ağacı oluşturulduktan sonra, sınıf etiketleri olan fakat sınıflandırma kuralları bilinmeyen yeni örneklerin sınıflandırılması için kolayca kullanılabilir [50].

2.3.1.5.3. Rastgele Orman

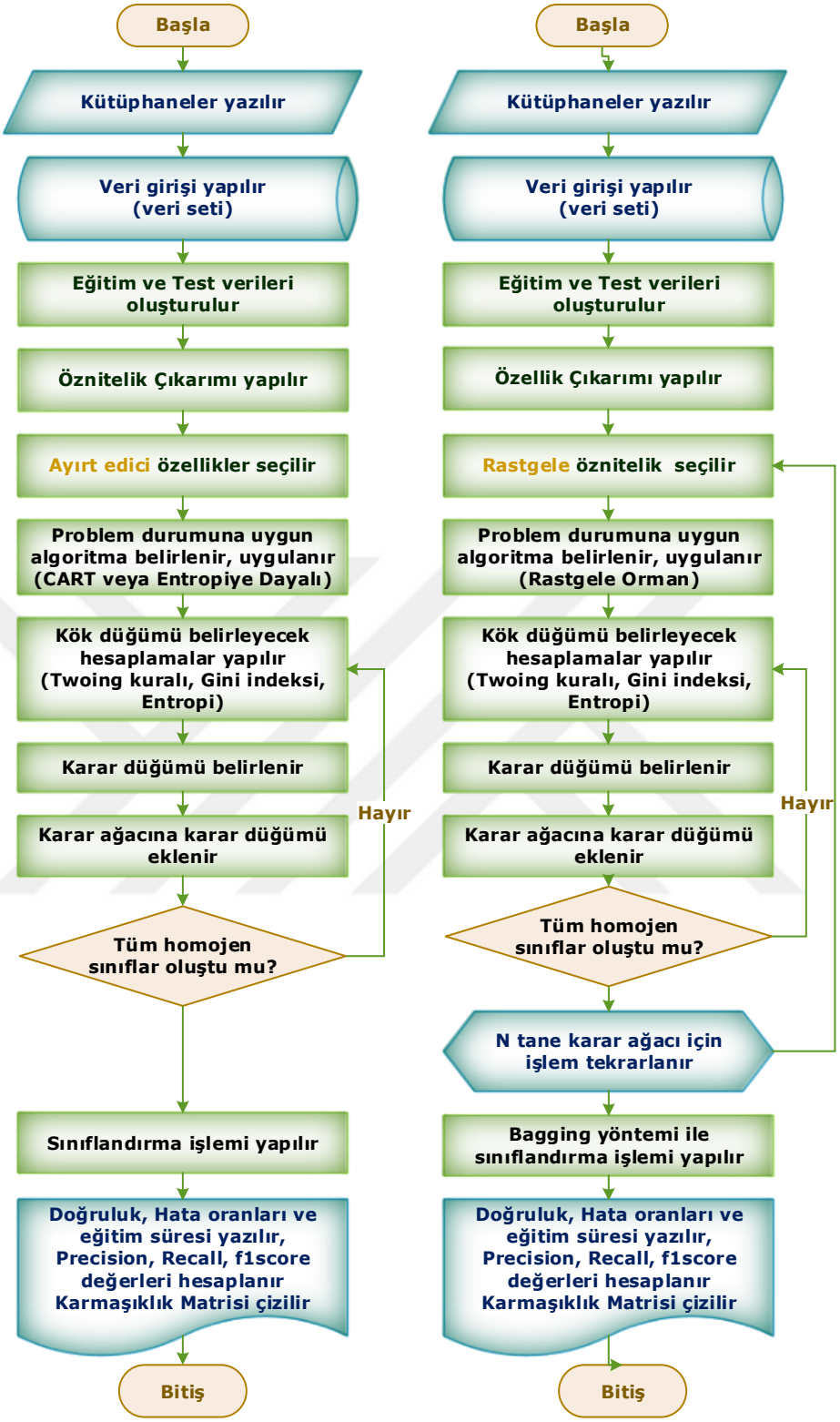
Karar ağacının derinliğinin az olması sınıflandırmanın gerçekleştirilememesine sebep olurken derinliğinin çok olması sınıflandırmayı zorlaştırmaktadır. Rastgele ormanda ağaç sayısı arttırılarak ağaç derinliği azaltılmaktadır. Rastgele orman yöntemi Breiman [51] tarafından 2001 yılında geliştirilmiştir. Rastgele ormanda ağaçlar arasında doğru sınıflandırılanların tespit edilebilmesi bagging işlemi ile mümkün olmaktadır. Bagging işlemi Breiman [52], tarafından geliştirilmiştir. Bagging işlemi sınıflandırma problemlerinde oylama adı verilen hesaplamalar ile gerçekleştirilirken regresyon problemlerinde ortalama adı verilen hesaplamalar ile gerçekleştirilir. Rastgele ormanda kök düğümü belirleyecek olan öznitelikler karar ağaçlarından farklı olarak rastgele seçilir. Rastgele seçme yöntemi Dietterich [53] tarafından geliştirilmiştir. Ho [54], her bir ağacın büyümesi için kullanılacak özellik alt kümesinin rastgele seçimini yapan "rastgele alt uzay" yöntemi üzerine bir dizi makale yazmıştır [51]. Bir rastgele karar ağacının yüksek performans gösterebilmesi, kullanılan uygun ağaç sayısı ve ağaçlar arasında yapılacak doğru oylama ile mümkün olmaktadır. Şekil 2.12'de rastgele orman örneği verilmiştir.



Şekil 2.12. Rastgele orman ağaç yapısı örneği

Örnekte ağaç sayısı 3 adet iken ağaç derinliği 4'tür. Bagging işlemi ile ağaçlar arasında oylama işlemi yapılarak sınıflandırma işlemi gerçekleştirilmiştir.

Rastgele orman ve diğer karar ağacı yöntemleri akış diyagramları Şekil 2.13'te verilmiştir. Rastgele ormanda öznitelikler rastgele seçilirken diğer karar ağaçlarında homojen bölünmeyi sağlayan öznitelikler seçilir. Bununla birlikte rastgele ormanda birden fazla ağaç ile sınıflandırma yapılırken diğer karar ağaçlarında bir ağaç ile sınıflandırma yapılır.



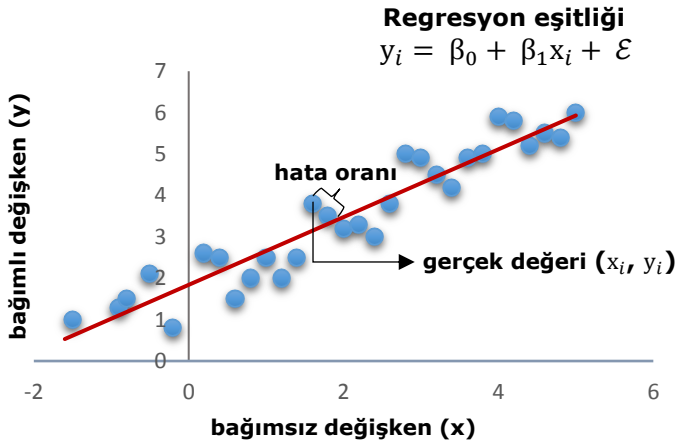
Şekil 2.13.Karar ağaçları akış diyagramı (**a)** Sınıflandırma ve regresyon, entropiye dayalı (**b)** Rastgele orman

2.4.2. Regresyon Tahmin Yöntemleri

Regresyon analizinin amacı iki veya daha fazla değişken arasındaki ilişkiyi, istatistik çözümleme tekniği ile matematiksel bir modele dönüştürmektir [55]. Sınıflandırma yöntemlerinde olduğu gibi eğitim verileri makine öğrenmesi algoritmaları ile eğitilip regresyon modeli oluşturulur. Bu model ile yeni veriler doğru tahmin edilmeye çalışılır. Sınıflandırmadan farklı olarak sonuçlar sınıfsal değil sayısaldır. Sınıflandırma yöntemlerinde kesikli çıktı ile veriler üzerinden tahmin modeli oluşturulurken regresyon yöntemlerinde sürekli çıktı ile veriler üzerinde tahmin modeli oluşturulur.

2.4.2.1. Doğrusal Regresyon

Girdi veriler ile çıktı veriler arasında doğrusal bir ilişki varsa doğrusal regresyon yöntemini kullanır [56]. Daha açık ifade etmek gerekirse girdi veriler ile çıktı veriler arasındaki doğrusal bir kombinasyon ile bir fonksiyon oluşturulur ve oluşturulan fonksiyon sayesinde sürekli tahmin işlemi gerçekleştirilir. Etiketli x verilerine ait iki boyutlu koordinat düzleminde oluşturulan regresyon grafiği Şekil 2.14'teki gibidir.



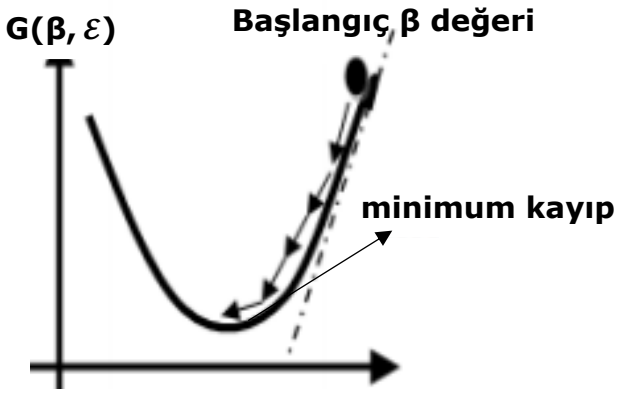
Şekil 2.14. Doğrusal regresyon grafiği

Kullanılan algoritmada sayısal veriler kullanılır. Basit bir doğrusal regresyon modeli için denklem eşitlik 2.11'de gösterildiği gibidir. Buradaki x girdi verilerini y ise çıktı verilerini gösterir. β_0 ve β_1 değerleri regresyon katsayıları olarak adlandırılır. $\mathcal{E}$ ise rastgele bir hata bileşenidir.

$$y = \beta_0 + \beta_1 x + \mathcal{E} \tag{2.11}$$

Etiketli bir x_i verisine karşılık gelen değer y_i; doğrusal regresyon fonksiyonu ile oluşturulan değer y_i' ise, iki değer arasındaki farkın mutlak değeri $|y_i - y_i'|$ kaybı (loss veya cost) yani hata oranını verir. Hata oranını en aza indirebilmek için fonksiyondaki regresyon katsayıları (β) ve hata bileşeninin ($\mathcal{E}$) en uygun değeri alması beklenir. Doğrusal bir regresyon modelinde hatayı en aza indiren katsayıları bulmak için de bazı teknikler kullanılır. Makine öğrenmesi problemlerinde en yaygın kullanılan teknik "En dik (Gradyan) İnişi" tekniğidir. Bu teknikte bir veya daha fazla giriş verisi olduğunda hata oranını en aza indirmek için regresyon katsayıları rastgele değerlerden başlayarak optimize edilir. Eğitim verisi üzerinde optimize edilen fonksiyon test verisi üzerinde uygulanır.

Gradyan iniş tekniğinin grafiksel gösterimi Şekil 2.15'te verilmiştir. G(β, $\mathcal{E}$) minimum kayıp fonksiyonunu göstermektedir. Gerçek sonuçlar ile eğitilmiş modelden elde edilen sonuçların arasındaki karesel hataların toplamı kullanılarak β değerleri optimize edilir.



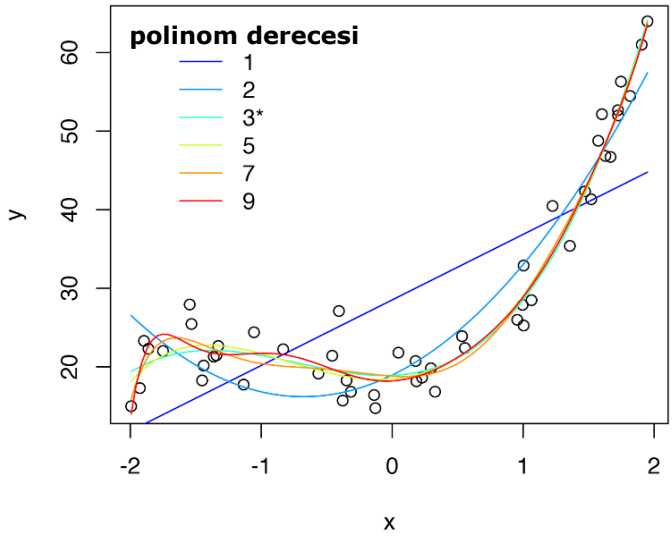
Şekil 2.15. Gradyan iniş tekniği grafiksek gösterimi [57]

2.4.2.2. Polinom Regresyon

Girdi ve çıktı değişkenler arasındaki ilişkinin doğrusal olmadığı durumlarda polinom regresyon kullanılır. x girdi değişkenlerine ve y çıktı değişkenine sahip bir polinom regresyon için modeli için eşitlik 2.12'de gösterilmiştir.

$$y = \beta_0 + \beta_1 x + \beta_2 x^2 + \cdots + \beta_m x^m + \mathcal{E} \quad (2.12)$$

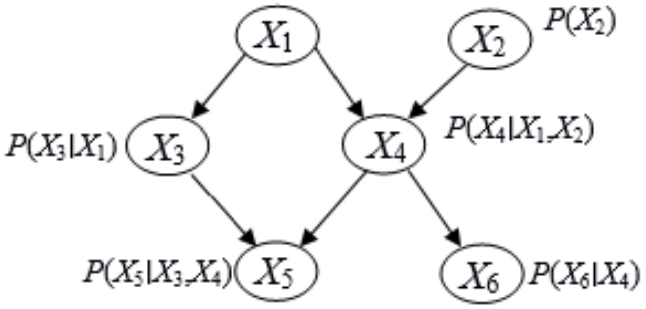
Eşitlikteki m değeri polinom derecesi olarak adlandırılır. Derecelere göre bulunan regresyon grafiği Şekil 2.16'da gösterildiği gibidir.



Şekil 2.16. Polinom derecelerine göre regresyon grafiği [58]

2.4.2.3. Bayes Ağları

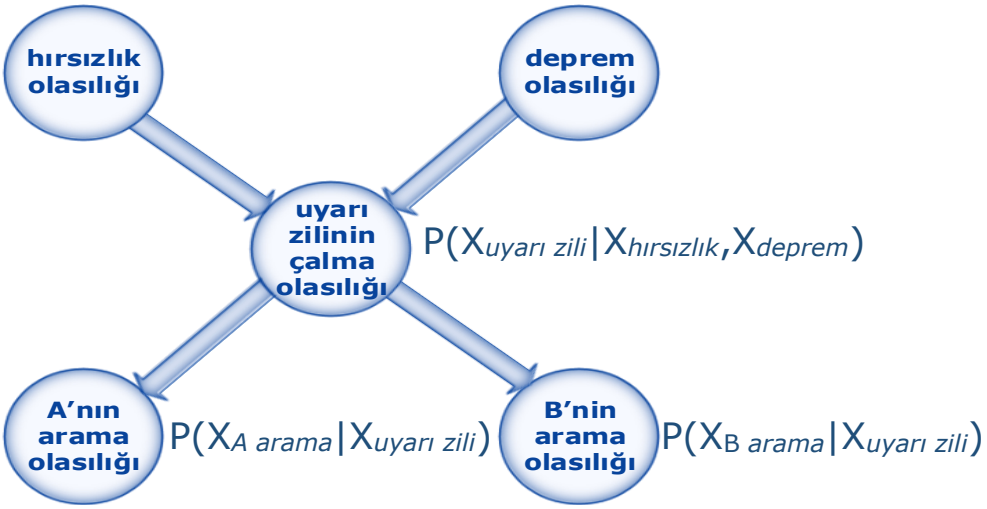
Bayes ağları, bir karar verme durumunda çıkarsama yapmak için kullanılır. Olasılık ağ modelidir. Bayes ağlarında tüm değişkenler düğümler olarak gösterilir. Değişkenler arasındaki ilişkiler, ilişkinin yönünü gösteren oklar ile gösterilir ve grafiksel model oluşturulur [59]. Böylelikle değişkenler arasındaki ilişkiler, Şekil 2.17'de olduğu gibi görsel olarak sunulur.



Şekil 2.17. Beş değişkenli bayes ağ modeli

Bayes ağları diğer modellerin aksine tek bir değişkene bağlı kalınmadan çıkarımlar yapılabilir ve yeni gözlemler ile tekrar düzenlenebilir. Herhangi bir bayes ağında X_1 ve X_2 ağda yer alan iki değişken olarak varsayılırsa X_1 değişkeninden X_2 değişkenine bir ok olması durumunda X_1 değişkeni X_2 değişkeninin ebeveyni, X_2 değişkeni ise X_1 değişkeninin çocuk değişkeni olarak adlandırılır [60]. Bir düğümün olasılığı biliniyorsa diğer bağlı düğümlerinde olasılıkları hesaplanabilir. Düğümler arasındaki bu bağımlılık koşullu olasılık değerleri olarak ifade edilir. Değerlerin oluşturduğu olasılık dağılım tablolarına koşullu olasılık tablosu adı verilir. Tabloda herhangi bir koşullu durumda bir düğüme ait ebeveyn düğümlerinin alabileceği değerler gösterilir. Bir düğümün k tane ebeveyni varsa 2^k adet olasılığı vardır [61].

Örneğin [62], evde uyarı alarm sisteminin olduğunu varsayalım. Sistem, olası bir hırsızlık durumunda veya ufak depremlerde uyarı versin. A ve B komşularından A komşusu, telefon zili ile uyarı zilini karıştırırken; B komşusu ise yüksek sesle müzik dinlediğinden dolayı uyarı zilini çoğunlukla duymamaktadır. Çalan telefonun A veya B komşularından geldiği bilindiğine göre arayan kişiye göre hırsızlık durumunu tahmin etme olasılığı bayes yaklaşımı ile çözülebilir. Böyle bir problemde deprem, hırsızlık veya uyarı sisteminin bozuk olma durumlarına göre uyarı zilinin çalma ihtimalleri birer düğümü temsil eder. Aynı zamanda uyarı zilinin çalması durumunda A ve B komşularının arama ihtimalleri de birer düğüm olarak kabul edilir. Naive Bayes yaklaşımından farklı olarak uyarı zilinin çalma olasılıkları veya haberin doğru olma olasılıkları olasılıksal (sayısal) olarak ifade edilir. Verilen örneğin Bayes olasılık ağ modeli Şekil 2.18'de verilmiştir.



Şekil 2.18. Alarm sistemi bayes olasılık ağ modeli

2.5. Denetimsiz Öğrenme

Denetimsiz öğrenme, girdi verileri üzerinden çıkarım yapılması esasına dayanır. Girdi değişkenler arasındaki ilişkiye dayalı olarak verileri homojen gruplara ayırır ve çıktı değişkenlerini oluşturur. Girdi verileri etiketsiz verilerden oluşur. Veriler arasındaki benzerliklere göre sınıflandırma tahmin işlemi yapılır.

2.5.1. Kümeleme Tahmin Yöntemleri

Kümeleme tahmin yöntemi sınıf sayısı belli olmayan ve sınıflandırılmamış verilerin özelliklerine göre sınıflandırılması esasına dayanır [63]. Kümelenen çıktıların diyagramda gösterilmesi mantığına dayanır. Kendi içindeki benzerlik oranı maximum; farklı kümeler arasındaki benzerlik oranının minimum olması beklenir. Yalnızca benzerliğin kriter olarak alınmadığı kümeleme analizinde girdi verileri arasındaki uzaklık da dikkate alınır. Kümeleme analizinde en önemli sorun küme sayısıdır. Çünkü küme sayısı küme kalitesini gösterir. Uygun küme sayısın belirlenmesi amacıyla birçok yöntem geliştirilmiştir. Ancak günümüzde yayınlanan birçok bilimsel makalede küme sayısının belirlenmesinde net bir yöntem yoktur [64].

Kümeleme teknikleri sayesinde büyük hacimli veriler özelliği kaybolmadan daha anlamlı homojen yapılara indirgenebilirler. Okuyucuya göre haber önerisi sunmak isteyen online bir gazete, haber başlıklarını politika, spor, eğitim ve teknoloji gibi kategorilere kullanıcı müdahalesi olmaksızın kümeleme işlemi ile gerçekleştirebilir.

2.5.1.1. K-Ortalamalar

K-ortalamalar yöntemi hiyerarşik olmayan kümeleme algoritmaları arasında en çok bilinen yöntem [65] olması ile birlikte görüntü analizinde kullanılan önemli algoritmalardan birisidir [66]. K-ortalamalar yöntemi öznitelik çıkarımı yapılmış bir veri seti üzerinde k adet kümeye ayırma işlemi olarak tanımlanabilir. Kümeleme işlemi veriler arası benzerlik veya uzaklık kullanılarak gerçekleştirilir. Uzaklık veya benzerlik ölçüsü belirlenerek işleme başlanır. Bir veri seti üzerinde uzaklık ölçüsü ile yapılan kümeleme işleminin işlem adımları aşağıdaki gibi gerçekleştirilir:

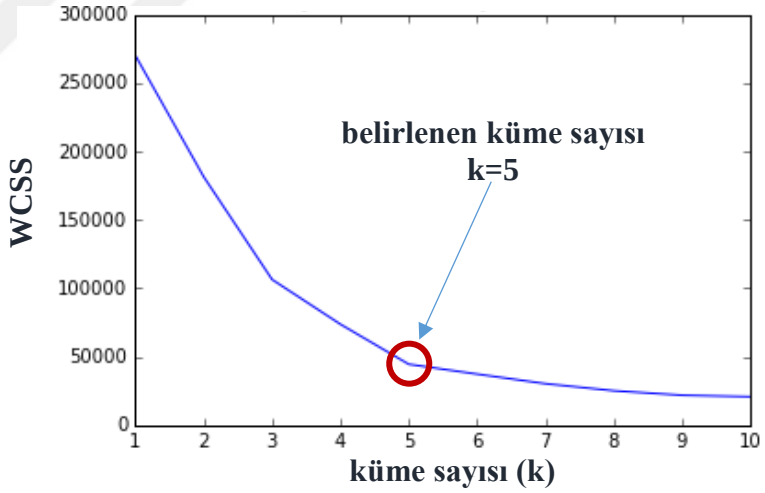
1. Veri setinde rastgele k adet nokta seçilir veya veri setinin ortalaması alınarak başlangıç noktaları belirlenir. Bu noktalar veri seti için küme merkezi olarak belirlenir.
2. Küme merkezlerinin verilere uzaklıkları (öklid) hesaplanır.
3. Veriler küme merkezlerine en yakın olduğu veri kümesine atanır.
4. Veri atama işlemi yapıldıkça hesaplamalar yinelenerek yeni merkez noktaları ve atama işlemleri gerçekleştirilir.

Küme merkezi olarak atanan noktaların verilere olan uzaklıkları genellikle Within Clusters Sum of Square (WCSS) eşitliği ile hesaplanır. Seçilen merkezi noktaların verilere olan öklid uzaklıklarının toplamı WCSS eşitliğini verir. Küme sayısı, K; kümeye ait elemanlar S_k; veri setindeki tüm elemanlar p ile gösterilmek üzere WCSS formülü eşitlik 2.13'te verilmiştir.

$$WCSS(K) = \sum_{k=1}^{K} \sum_{i \in S_k} \sum_{j=1}^{p} (x_{ij} - \bar{x}_{kj})^2 \tag{2.13}$$

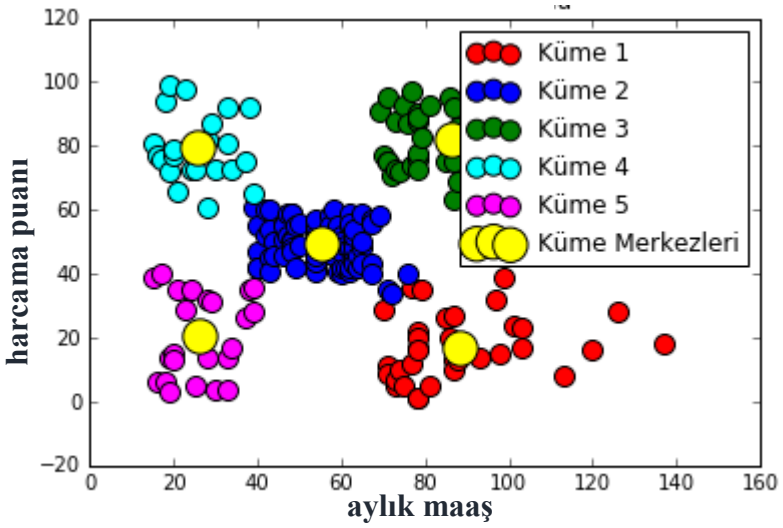
K-ortalamalar yöntemi ile belirlenen küme sayısı K değeri ile gösterilir. K değerinin çok olması istenilenin aksine büyük veri sorununa sebep olurken, az olması ise küme içerisindeki homojenliği düşürecektir. Bu sebeple en uygun K değeri belirlenmelidir. Kümeleme algoritmalarında oluşturulacak kümenin özellikleri önemlidir. Bu özellikler kümenin şekli, yapısı, uzaydaki yerleşimleri ve hacmi olarak gösterilebilir. Özelliklerin değişkenlik göstermesi farklı sonuçlar üretir, dolayısı ile geçerli sonuçlar üretmek zorlaşır [67].

Örneğin bir iş yerinde çalışan personelin, aylık para harcama durumlarına göre 100 üzerinden puanla yapalım. Personellerin yaptıkları harcamaları, aldıkları aylık maaşlara göre kümelendirebiliriz. Yani personelleri az veya çok harcama yapma durumlarını kümeleme yöntemi ile belirleyebiliriz. Personel bilgilerini içeren veri seti hazırlandıktan sonra WCSS denklemi ile en uygun küme sayısı belirlenir. Şekil 2.19.a'da optimum küme sayısını gösteren grafik verilmiştir.



(a)

Küme sayısının k=10 olarak belirlendiği grafikte, küme sayısı k=5'e kadar WCSS değeri hızlı bir düşüş gösterirken k=5'ten sonra yatay seyrine devam etmiştir. Bu da küme sayısı için optimum değerin 5 olduğunu gösterir. WCSS denklemi le belirlenen k değeri 5'tir. Yapılan hesaplamalar sonucu belirlenen küme merkezleri ve veriler Şekil 2.19.b'de gösterildiği gibidir.



(b)

Şekil 2.19. k-En yakın komşu yöntemi örneği **(a)** K-ortalamalar algoritmasında küme sayısının bulunması **(b)** K-ortalamalar algoritması ile kümeleme yöntemi [68]

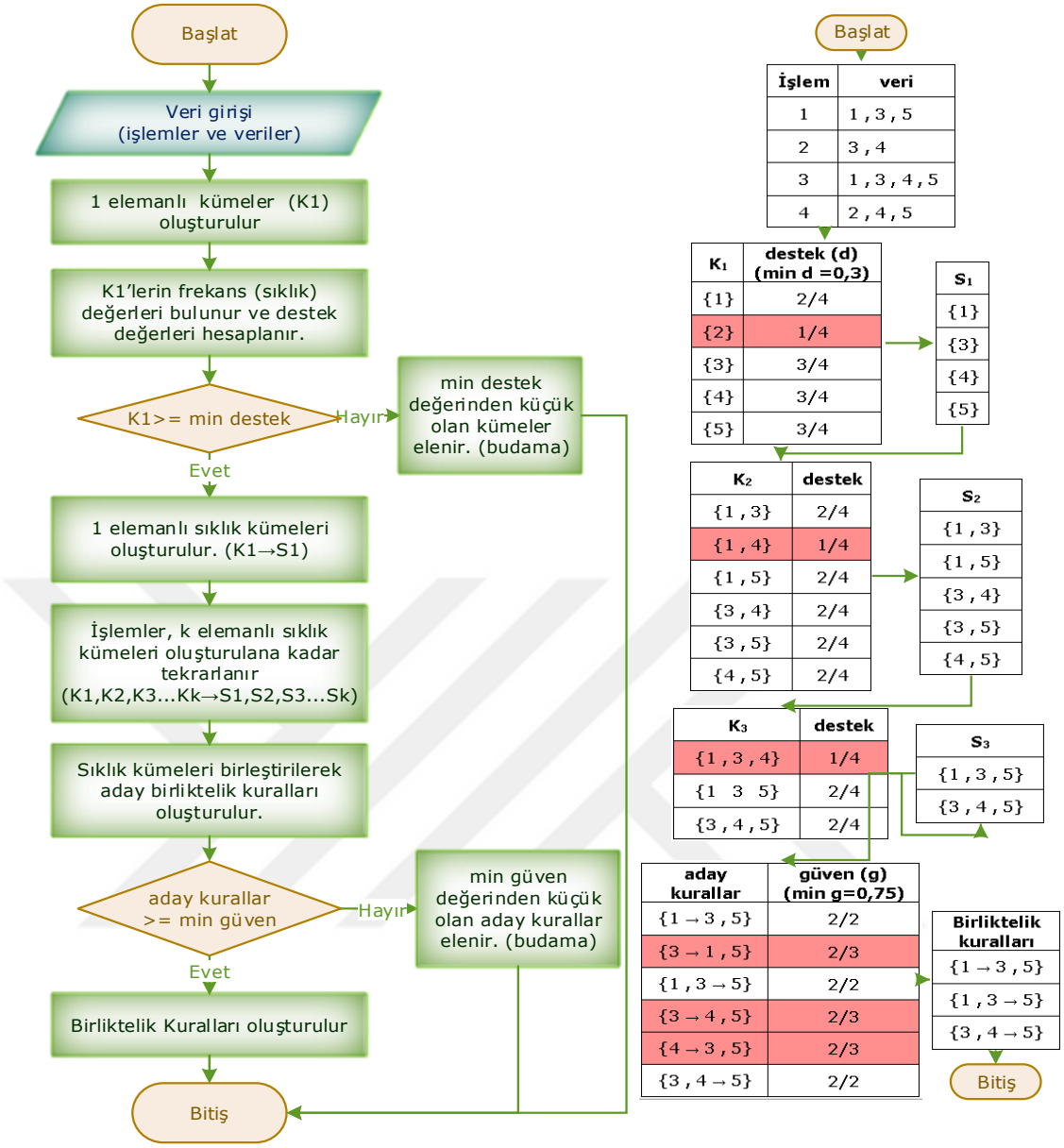
2.5.1.2. Birliktelik Kuralları

Öncelik sonralık ilişki analizi yapan birliktelik kural madenciliğinde apriori algoritması kullanılır. Agrawal ve Srikant [69] tarafından büyük ölçekli veri tabanları üzerinde apriori algoritması geliştirilmiştir. Algoritma akış diyagramı Şekil 2.20.a.'da verilmiştir. Algoritmanın çalışma mantığı şu şekilde açıklanabilir: Veri gruplarının her birisine işlem denir. Şekil 2.20.b'deki örnekte 4 işlem bulunmaktadır. Algoritmada ilk olarak tüm işlemlerdeki veriler tekrarsız olacak şekilde 1'er elemanlı kümelere (K_1) ayrılır. K_1'lerin tüm işlemlerde bulunma sıklıkları (frekansları) bulunur. Her bir kümenin bulunma sıklığının işlem sayısına bölünmesi ile destek değeri elde edilir. Bir diğer ifadeyle destek değeri, verinin küme içindeki bulunma yüzdesidir. Destek değeri denklemi eşitlik 2.14'te verilmiştir.

$$(A \rightarrow B) \quad \text{olmak üzere;} \quad \text{destek} = \frac{\text{sıklık}(A, B)}{\text{işlem sayısı}(N)} \tag{2.14}$$

Tüm kümelerin destek değerleri bulunur. Destek değeri, min destek değerinden küçük olan kümeler elenir(budama). Yeni oluşturulan kümelere 1 elemanlı Sıklık (Frequent) kümesi (S_1) denir. 1 elemanlı verilerden tekrarsız olacak şekilde 2 elemanlı veriler (K_2) oluşturulur. Bir önceki aşamada yapılan işlemler 2 elemanlı kümeler için de tekrarlanır ve 3 elemanlı kümeler oluşturulur. Max elemanlı kümelere ulaşana kadar işlemler döngüsel olarak tekrar eder. Max eleman sayısı, eleman sayısı en çok olan işlemden büyük olamaz. Örnekte en fazla 4 elemanlı işlem bulunmaktadır. Her Oluşturulan yeni kümelerden aday kurallar oluşturulur. Kurallar, küme elemanlarının öncül ve ardıl olacak şekilde gruplanmasıyla oluşur (Şekil 2.20.b: {1,3,5} →{1→3,5} gibi). Oluşan yeni kümeler aday birliktelik kurallarıdır. Kuralların güven değerleri bulunarak min güven değerinden küçük olan kurallar elenir (budama). Kalan kurallar istenen birliktelik kurallarıdır. Güven değeri Eşitlik 2.15'teki gibi hesaplanır.

$$(\mathrm{A} \rightarrow \mathrm{B}) \quad \text{olmak üzere;} \quad \text{güven} = \frac{\text{sıklık}(\mathrm{A})}{\text{sıklık}(\mathrm{A}, \mathrm{B})} \tag{2.15}$$



(a) **(b)**

Şekil 2.20. Apriori algoritması ile birliktelik kuralı oluşturulması **(a)** Birliktelik kuralı akış şeması **(b)** Örnek akış şeması

Apriori algoritmasının uygulandığı en tipik örneklerden birisi market sepet analizidir [70]. Tüketicilerin alışveriş alışkanlıklarının incelenmesi markette ürünlerin rafa dizilimini veya e-ticaret web sitelerinde ürün dizilimini kolaylaştırarak daha çok ürün satışını sağlayabilir. İnternet üzerinden bir ürünü inceleyen veya alan bir tüketiciye yeni ürünlerin önerisinin sunulmasına olanak tanır.

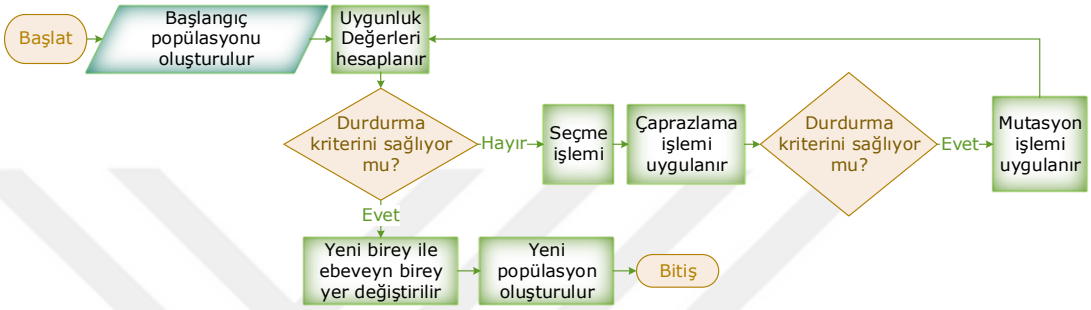
Örneğin herhangi bir E-ticaret web sitesinden zeytin alanların %80'i peynir de almışsa, zeytin alan yeni tüketicilere peynir önerisi sunulabilir. Yani tüketicilerin hangi ürünleri bir arada aldıklarının analiz edilmesi tüketicilerin yeni satış stratejileri geliştirmelerine olanak sağlar. Apriori algoritması tıp, eğitim, finans, telekomünikasyon, pazarlama, bankacılık ve E-ticaret gibi uygulama alanlarında kullanılır.

2.5.1.3. Genetik Algoritmalar

Genetik algoritmalar evrim benzetimini kullanarak en uygun stratejinin devamlılığını sağlamak amacıyla en uygun fonksiyonun çözüm uzayını araştırır. Genel olarak, herhangi bir popülasyonun en güçlü bireyleri, bir sonraki nesile çoğalma ve hayatta kalma eğilimindedirler, böylece, birbirini izleyen kuşakları geliştirirler [71]. Genetik algoritma, güçlü bireyler oluşturabilmek için ebeveynden gelen bilgileri farklı yöntemler ile birleştirir. Genetik algoritmalar doğrusal veya doğrusal olmayan problemlerin çözümünde kullanılır. Genetik algoritmada, genler üzerinde genetik bilgi taşıyan birimlerdir. Birden fazla gen birleşerek kromozomları oluşturur. Kromozomlar herhangi bir problem için aday çözümleri temsil eder. Nesildeki kromozomların hayatta kalma olasılığını belirlemek için aday çözümlere uygunluk fonksiyonu uygulanır. Uygunluk fonksiyonu yüksek çıkan aday çözümler hayatta kalma gücü yüksek bireylerden oluşur. Kromozomlar bir araya gelerek de popülasyonu oluşturur [72]. Yeni nesil, önceki popülasyonun farklı kombinasyonlarından oluşan, en yüksek uygunluğa sahip olan ve sağ kalan yeni bireylerden oluşmaktadır [73]. K-ortalamalar algoritmasının gelişmiş bir versiyonu olarak da düşünülebilir.

Algoritmanın çalışma şekli şu şekilde açıklanabilir: Genetik algoritmada çözümlerin ilk popülasyonu rastgele oluşturulur. En uygun çözümü veren iterasyon sayısı ve durdurma kriteri belirlenir. Kromozomların uygunluk değerleri hesaplanır. Uygunluk değeri yüksek olan bireyler eşleştirilir. Eşleşen bireylere çaprazlama işlemi uygulanır. Çaprazlama, iki kromozomun bir araya gelerek gen değişimi yapması esasına dayanır. Gen değişimi sonucu yeni bireyler oluşur.

Belirli bir olasılık düzeyindeki (mutasyon oranı) bireylere mutasyon işlemi uygulanır. Mutasyon işlemi bir kromozom diziliminde bazı değerlerin, rastgele seçilmiş bir değerle değiştirilmesi esasına dayanır. Çaprazlama veya mutasyona uğrayan uygunluk değeri yüksek çıkan yeni bireyler ebeveyn bireylerle değiştirilir. Böylece uygunluk değeri yüksek yeni bireyler elde edilmiş olur. Genetik algoritma akış şeması Şekil 2.21'de verilmiştir.



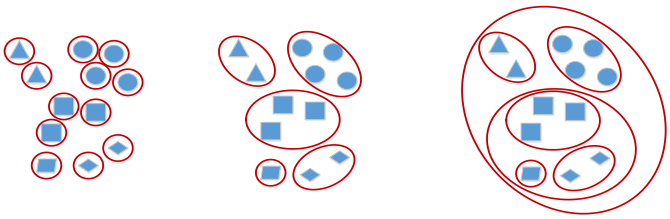
Şekil 2.21. Genetik Algoritma akış şeması

2.5.1.4. Hiyerarşik Kümeleme

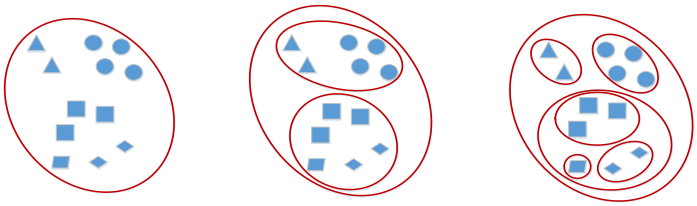
Hiyerarşik kümelemede, iç içe geçirilmiş kümelerin en dışta tek bir kümede toplanması ile kümeler dizisi üretilir. Kümelerin oluşum şekillerine göre birleştirici ve ayırıcı olmak üzere iki yöntem kullanılır.

- Birleştirici Yöntem: Başlangıçta tüm elemanlar birer küme olarak atanır. Birbirine en yakın elemanların birleştirilerek kümeleme yapılır. Bu yöntemde her birleştirme işleminden sonra küme sayısı 1 azaltılmış olur.
- Ayırıcı Yöntem: Başlangıçta tüm veriler bir kümeye atanır. Benzer olmayan veriler kümeden atılarak yeni kümeler oluşturulur. Bu yöntemde ise küme sayısı her seferinde 1 arttırılır.

Veriler arası uzaklık veya benzerlikler kullanılarak kümeleme işlemi gerçekleştirilir. Birleştirici ve ayırıcı hiyerarşik kümeleme yöntemini gösteren şemalar Şekil 2.22.a. ve Şekil 2.22.b'de gösterildiği gibidir.



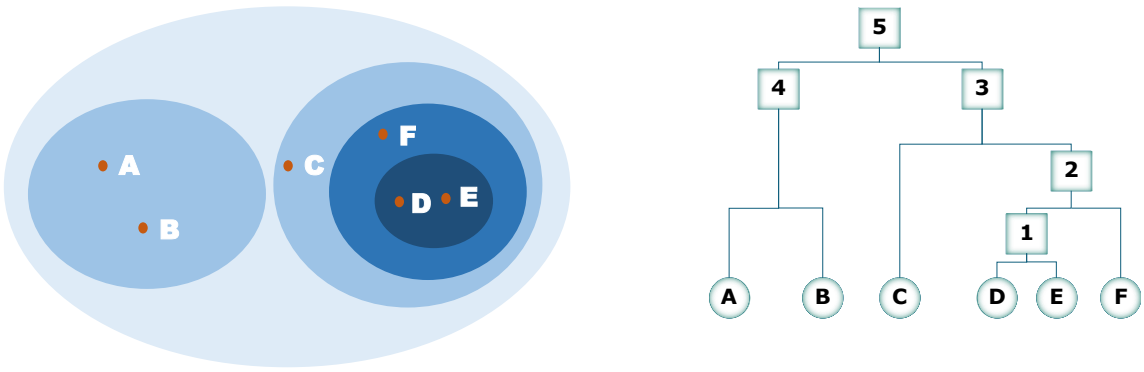
(a)



(b)

Şekil 2.22. Hiyerarşik kümeleme yöntemi **(a)** Birleştirici **(b)** Ayırıcı

Kümeler arası hiyerarşik ayrıştırma dendrogram ağaç diyagramı ile görselleştirilir. Şekil 2.23'te hiyerarşik kümeleme şeması ve dendrogramı verilmiştir.



(a) **(b)**

Şekil 2.23. Hiyerarşik şemayı dedrograma dönüştürme işlemi **(a)** Hiyerarşik kümeleme şeması **(b)** Hiyerarşik kümeleme dendrogramı

Ağaç diyagramı sayesinde istenen hiyerarşik ayrıştırma görselleştirilmiş olur [74]. Şekil 2.23.b'de verilen dendrogram şeması aşağıdaki yorumlanabilir:

D ve E	verileri birleşerek	1. küme	{D,E}
1. küme ve F	verisi birleşerek	2. küme	{{D,E}F}
2. küme ve C	verisi birleşerek	3. küme	{{{D,E}F},C}
A ve B	verileri birleşerek	4. küme	{A,B}
3. ve 4.	kümeler birleşerek	5. küme	{{A,B},{{{D,E}F},C}}

oluşmuştur.

Hiyerarşik kümeleme yönteminde oluşturulacak küme sayısı önceden bilinemez. Her ne kadar algoritmanın çalışma mantığı K-ortalamalar ile benzer olsa da bu özelliği ile farklılık gösterir. Ayrıcı yönteme oranla daha çok kullanılan birleştirici hiyerarşi yöntemi dört adımdan oluşan bir algoritma ile ifade edilebilir. Bunlar [75]:

1. Birey sayısı kadar küme sayısı ile işe başlanır.
2. Düzlem üzerinde birbirine en yakın olan iki küme birleştirilir.
3. Birleştirilerek indirgenen küme sayısı ile uzaklıklar matrisi hesaplanır.
4. İkinci ve üçüncü adımda yapılan işlemler küme birey sayısının bir eksiği kez tekrarlanır.

2.5. Yapay Sinir Ağlarından Derin Öğrenmeye

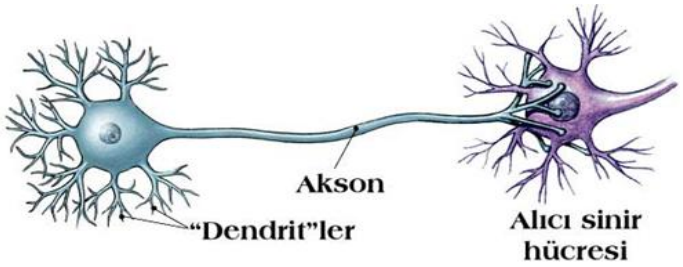
2.5.1. Yapay Sinir Ağları

Makine düşünebilir mi sorusu, insanın beyin yapısının yapay bir sistem olarak modellenebilirliğinin önünü açmıştır. Disiplinler arası bir yaklaşım olan yapay sinir ağları biyolojik bir yapının mekatronik bir yapı ile etkileşimini mümkün kılmıştır. Bu etkileşim sibernetik yaklaşımının ana konusudur. İnsanın beyin yapısından esinlenerek oluşturulan yapay sinir ağları görüntü işleme, doğal dil işleme, ses işleme, haberleşme ve tıp alanı gibi birçok alanda kullanılmaktadır.

Biyolojik Sinir Ağları

Sinir sistemi dışsal ve içsel çevreyi algılayarak bilgi elde eder. Bu bilgiyi analiz ederek vücuttaki hücre ağını kullanan sinyaller vasıtası ile kas hareketlerini düzenler [76]. Sinir sistemi vücudu saran milyonlarca sinirden meydana gelir. İnsan vücudunun sinir sistemi üç aşamadan oluşur: Reseptörler, bir sinir ağı ve efektörler. Reseptörler, içsel olarak veya dış dünyadan uyaranları alırlar, sonra bilgiyi elektriksel impulslar biçiminde nöronlara geçirirler. Nöral ağ daha sonra girdileri işler ve daha sonra çıktıların doğru kararını verir. Son olarak, efektörler elektriksel impulsları sinir ağından dış çevreye verilen yanıtlara dönüştürür. Nöral ağın temel unsuru bir nöron olarak adlandırılır. Nöronlar Şekil 2.24'te görüldüğü üzere üç kısımdan oluşur:

1. Gövdesi: Nöronun çekirdek, sitoplazma ve organellerinin bulunduğu kısımdır.
2. Dendrit: Her birisinin bir nörona bağlı olduğu, çevresindeki nöronlardan gelen sinyali alan ağaç benzeri bir yapıdır.
3. Akson: Sinyali bir nörondan başkalarına ileten ince bir silindirdir. Aksonun sonunda, dendritlere temas bir sinaps ile yapılır. Sinapstaki nöronlar arası sinyal genellikle kimyasal difüzyon, bazen de elektriksel impulslardır. Bir nöron, sadece belirli bir durumla karşılaşıldığında elektriksel bir dürtü ateşler.

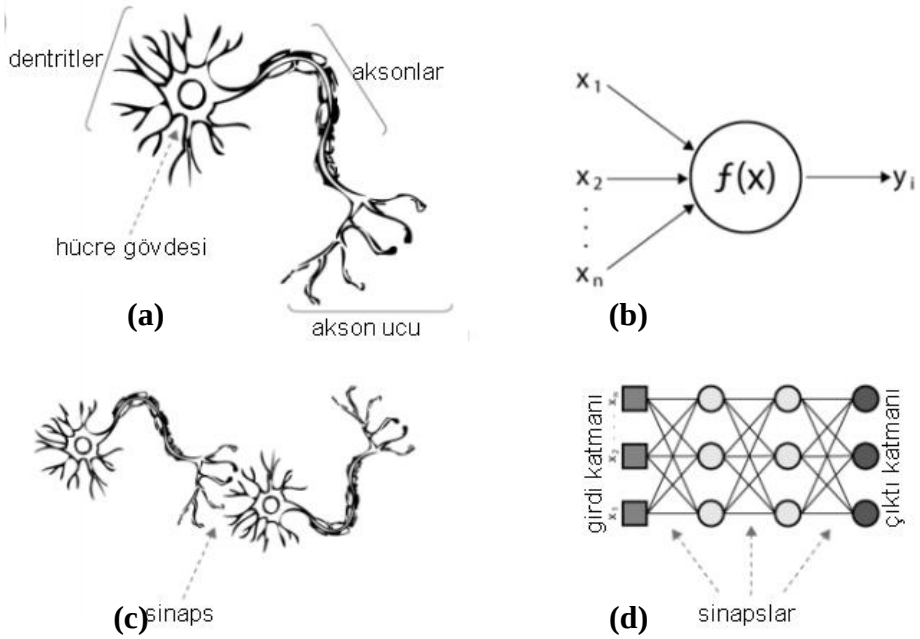


Şekil 2.24. Basit bir biyolojik nöron yapısı

Bir insan beyni muazzam miktarda sinir hücresi ve nöron içerir. Bu hücrelerin her biri, diğer benzer hücrelere bağlanır ve çok karmaşık bir sinyal iletimi ağı oluşturur. Her hücre, bağlı olduğu tüm diğer nöral hücrelerden gelen girdileri toplar ve belirli bir eşiğe ulaşırsa, bağlı olduğu tüm hücrelere sinyal verir [77].

Her sinapstan nörona gelen impuls sinyali, eksitatör veya inhibitördür, bu da ateşlemeye yardım etmek veya engellemek anlamına gelir. Ateşlemeye neden olma koşulu, uyarıcı sinyalin, kısa bir süre içinde gizli toplama periyodu olarak adlandırılan belirli bir miktarda engelleyici sinyali aşmasıdır [78].

Yapay sinir ağları biyolojik sinir sisteminden etkilenerek geliştirilmiştir. Biyolojik sinir hücreleri birbirleri ile sinapslar vasıtası ile iletişim kurarlar. Bir sinir hücresi işlediği bilgileri aksonları yolu ile diğer hücrelere gönderirler. Benzer şekilde yapay sinir hücreleri dışarıdan gelen bilgileri bir toplama fonksiyonu ile toplar ve aktivasyon fonksiyonundan geçirerek çıktıyı üretip ağın bağlantılarının üzerinden diğer hücrelere (proses elemanlarına) gönderir [79]. Şekil 2.25'te biyolojik sinir sistemi ve yapay sinir ağları arasındaki karşılaştırma gösterilmiştir.



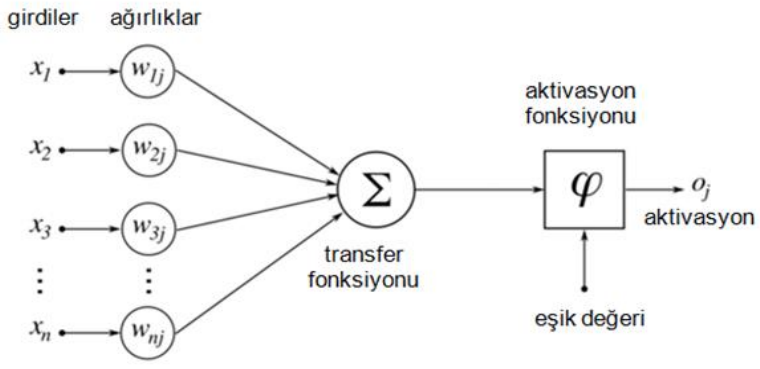
Şekil 2.25. Biyolojik nöron ağı ve yapay sinir ağı karşılaştırması **(a)** Biyolojik nöron **(b)** Yapay nöron **(c)** Biyolojik nöron ağı **(d)** Yapay sinir ağları [80]

Algılayıcılar

İnsan sinir sisteminin temel yapı birimi olan nöronların makine tasarımı algılayıcılar olarak adlandırılır. Algılayıcılar daha önceki çalışmalardan esinlenerek Frank Rosenblatt [81] tarafından geliştirilmiştir. Basit bir algılayıcı, giriş verileri, giriş ve ağırlıkların çarpımı, çarpımların toplamı, aktivasyon fonksiyonunun hesaplanması ve çıktı verisinin elde edilmesi olmak üzere 5 aşamadan oluşur. Şekil 2.26'da verilen basit bir algılayıcıda x_1, x_2, x_3, ..., xn olmak üzere n adet giriş verisi önem derecelerine göre belirlenmiş ağırlıklar ile çarpılır ve her birisi ağırlıklarla çarpılan değerler toplanır. Toplam değer belirli bir eşik değerin üzerinde ise veri iletimi bir sonraki nörona veya katmana iletilir. Basit bir yapay sinir ağı ağın en küçük parçası olan algılayıcı için doğrusal bir fonksiyonla tanımlanır. Formülün genel gösterimi:

$$f(x) = \sum_{i=1}^{n} w_i \, x_i \tag{2.16}$$

şeklindedir. *x* değerleri giriş verilerini veya özniteliği, *w* değerleri ağırlığı ifade etmektedir. Ağırlıklar çıktı verilerine etki eden girdi verilerinin önem derecelerini ifade eden gerçek sayılardır. Örneğin öğrencinin başarı durumuna etki eden faktörler girdi verilerini (*x*) ifade ederken, faktörlerin önem dereceleri ağırlıkları (*w*) ifade eder.



Şekil 2.26. Algılayıcı modeli

Biyolojik sinir iletiminde sinyallerin sinaps boşluklarından iletimi için sinyalin belirli bir eşik değerinin üstünde olması gerekir. Örneğin belirli bir eşik değerinin üstünde olan görüntüleri görebilir, sesleri işitebilir veya nesneleri hissedebiliriz. Algılayıcılar için yapay nöron iletimi için Eşitlik 2.17'de olduğu gibi f(x) fonksiyonunun belirlenen eşik değerinden yüksek olması beklenir.

$$\text{Çıkış birimi} = \begin{cases} 0 & \text{eğer} \quad \sum_j w_j x_j \leq \text{eşik} \\ 1 & \text{eğer} \quad \sum_j w_j x_j \geq \text{eşik} \end{cases} \tag{2.17}$$

Algılayıcıların bilgi iletimini somut bir örnekle açıklamak gerekirse; 2018 Geleneksel Okçuluk turnuvasına katılmak için kararsız kaldığınızı varsayalım. Aşağıdaki 4 duruma göre karar vermek istiyorsunuz.

1. Turnuva yapılacak yere toplu taşıma ile gidilebilir mi?
2. Turnuvaya katılacakların sayısı 100'ün altında mı?
3. Atış uzaklığı 30 m'nin altında mı?
4. Turnuva gününe 1 haftadan fazla vakit var mı?

Örnekte Turnuvaya katılımını yani sonucu etkileyen 4 durum yani 4 giriş mevcuttur. Giriş verilerini x_1, x_2, x_3 ve x_4 olarak gösterebiliriz. 4 durumda da Evet veya Hayır olmak üzere ikişer ihtimal mevcuttur. x_1 durumu için eğer turnuva yapılacak yere toplu taşıma ile gidilebiliyorsa $x_1 = 1$ gidilmiyorsa $x_1 = 0$ olur Aynı şekilde ikinci durum için turnuvaya katılacakların sayısı 100'ün altında ise $x_2=1$ üstünde ise $x_2=0$ olur. Üçüncü ve dördüncü durumlar için aynı işlemler tekrarlanır. Turnuvaya katılımı etkileyen 4 durumun önem derecelerini 10 üzerinde puanlarsak birinci durumun önem derecesi $w_1= 6$, ikinci durumun önem dereceği $w_2=5$, üçüncü durumun önem derecesi $w_3=8$ ve dördüncü durumun önem derecesi $w_4=4$ olarak belirlendiğini varsayalım. Eşik değerinin 10 olarak belirlendiği problemde 2.18'de eşitlikler verilmiştir.

Giriş verileri ilk durumda (*i.*) sırasıyla '0, 1, 0, 1' iken ikinci durumda (*ii.*) sırasıyla '0, 1, 1, 0' olarak verilmiştir. Ağırlıkların değişmediği algılayıcı modelinde sonuç ilk durumda turnuvaya katılım sağlanmazken ikinci durumda katılım sağlanır.

i. $\left|\begin{matrix} x_1 = 0 \\ x_2 = 1 \\ x_3 = 0 \\ x^4 = 1 \end{matrix}\right|$ $\left|w_1=6\ w_2=5\ w_3=8\ w_4=4\right|$ ise $f(x) = x_1 w_1 + x_2 w_2 + x_3 w_3 + x_4 w_4$

$=0*6 + 1*5 + 0*8 + 1*4 = 9$

(2.18)

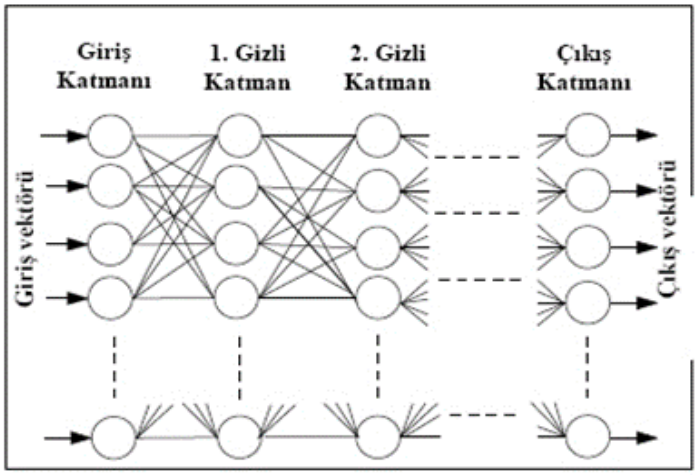
ii. $\left|\begin{matrix} x_1 = 0 \\ x_2 = 1 \\ x_3 = 1 \\ x_4 = 0 \end{matrix}\right|$ $\left|w_1=6\ w_2=5\ w_3=8\ w_4=4\right|$ ise $f(x) = x_1 w_1 + x_2 w_2 + x_3 w_3 + x_4 w_4$

$=0*6 + 1*5 + 1*8 + 0*4 = 13$

Çok Katmanlı Algılayıcılar

Yapay sinir ağlarının günümüzde en yaygın olarak kullanılan modeli çok katmanlı algılayıcı ağlarıdır. Bu ağlar, özellikle mühendislik problemlerinin %95'ine çözüm üretebilecek nitelikte bir ağdır. Çok katmanlı algılayıcı ağları XOR problemini çözebilmek için yapılan çalışmalar neticesinde ortaya çıkmıştır. Bu ağlar 3 katmandan oluşurlar

1. Girdi katmanı :Herhangi bir bağlantının hedefi olmayan düğümlerin oturduğu katmana giriş katmanı denir. N boyutundaki girdi modellerine uygulanması gereken birçok katmanlı algılayıcı, her bir boyut için bir tane n giriş nöronuna sahip olmalıdır.
2. Ara katmanlar :Girdi katmanından gelen bilgileri işlerler. Bir adet ara katman ile birçok problemi çözmek mümkündür. Eğer ağın öğrenmesi istenilen problemin girdi/çıktı arasındaki ilişkisi doğrusal olmaz ve karmaşıklık artarsa birden fazla sayıda ara katman kullanılabilir. Ara katmanlar gizli katmanlar olarak da adlandırılır.
3. Çıktı katmanı : Herhangi bir bağlantının kaynağı olmayan düğümlerin oluşturduğu katmana çıkış katmanı denir. Birçok katmanlı algılayıcı, bir katmanda birden fazla çıkış nöronuna sahip olabilir. Çıkış nöronlarının sayısı, eğitim modellerinin hedef değerlerinin (istenen değerler) açıklanma şekline bağlıdır. İletim giriş katmanından başlayarak ara katmanda devam eder ve çıkış katmanında son bulur [79]. Şekil 2.27'de çok katmanlı sinir ağı gösterilmiştir.



Şekil 2.27. Çok katmanlı sinir ağı mimarisi [82]

2.5.2. Evrişimsel Sinir Ağı

Derin öğrenme mimarilerinden birisi olan evrişimsel sinir ağı [83], çok katmanlı algılayıcıların bir türevidir. Makinenin ezberlemesini önlemek ve makine öğrenmesinin daha iyi gerçekleşmesi için eklenen katmanlar (pooling, dropout, tam bağlantılı) evrişimsel sinir ağı mimarisini çok katmanlı algılayıcılardan ayıran en önemli farklardan birisidir.

2.5.2.1. İleri Yayılımlı (Beslemeli) Evrişimsel Sinir Ağı

İleri yayılımlı ağlarda iletim tek yönlüdür. İletim girdi katmandan başlar, ağırlıklar vasıtası ile ara katmanlarda iletilerek çıktı katmanına ulaşır. [84].

2.5.2.2. Geri Yayılımlı (Beslemeli) Evrişimsel Sinir Ağı

Yapay sinir ağları birçok durumda bir kara kutu olarak kullanılır; belirli bir girdi istenen bir çıktı üretmelidir, ancak ağın bu sonuca nasıl ulaştığı kendi kendini organize eden bir sürece bırakılır [85]. Derin öğrenme süreci iki aşamadan oluşur. İlk aşamada

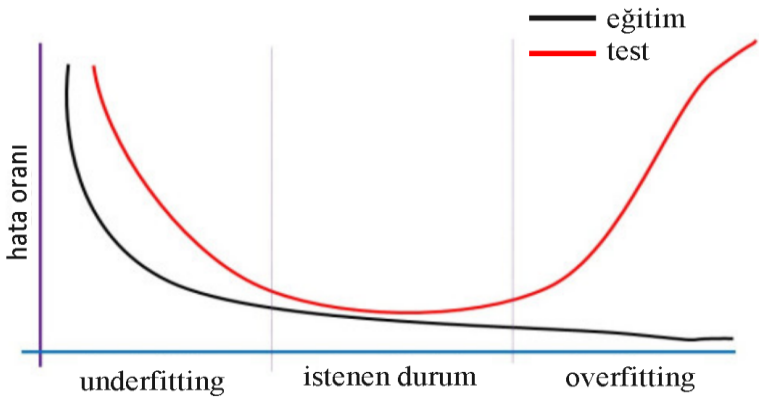
mevcut veriler üzerinden eğitim aşaması gerçekleştirilir. Bu aşamada etiketli veriler üzerinde evrişimsel sinir ağı mimarisi geliştirilir. Girdi katmanından alınan veriler gizli katmanda eğitildikten sonra çıktı katmanına iletilir. Burada tahmin sonucu ile gerçek sonuç karşılaştırılır. Karşılaştırma sonucu elde edilen hata oranının düşürülmesi için mimari çıkış biriminden geri beslenerek geriye doğru çalışır. Bu esnada ağırlıklar, öğrenme oranları değişir ve mimari tekrar eğitilir. Elde edilen hata oranının düşmesi uygun evrişimsel sinir ağı mimarisinin geliştirildiğini gösterir. Doğru mimarinin oluşturulması eğitim süresini kısaltabilir veya hata oranını en aza indirebilir. İkinci aşama; eğitim aşamasında oluşturulan evrişimsel sinir ağı mimarisi test aşamasında daha önce eğitilmemiş veriler üzerinden yapılır. Yeni veriler üzerinden yapılan tahminlerin doğruluk oranı evrişimsel sinir ağı mimarisinin başarısını gösterir. Evrişimsel sinir ağı mimarisi giriş, gizli ve çıktı katmanı olarak üç bölümden oluşur. Giriş katmanından alınan veriler gizli katmanda eğitilerek hata oranı tespit etmek üzere çıktı katmanına iletilir. Geliştirilen evrişimsel sinir ağı mimarisinde sistemin öğrenmesi istenir. Eğer oluşturulan mimari mevcut verileri ezberlerse *aşırı uyum* meydana gelir.

Aşırı Uyum, Ezberleme

Ezber ve öğrenme insanın çevreye uyum sağlama yeteneğine etki eden iki önemli kavramdır. Başka bir deyişle tek bir problem için formül ezberleyen bir öğrencinin benzer başka bir problemde başarılı olması düşük bir ihtimaldir. Zira formülün tüm parametrelerini özümsemek ve anlamlandırmak, o formülle çözülebilen tüm problemlerde daha yüksek bir ihtimalle başarılı sonuçlar verebilir. Makine öğrenmesi yöntemlerinde amaç mevcut veriler üzerinden yeni problemler hakkında doğru tahminlerde bulunabilmektir. Bu amaçla ilk olarak veri seti, eğitim ve test verileri olmak üzere ikiye ayrılır. Eğitim verisi ile öğrenme gerçekleştirilirken test verisi ile tahmin işlevi gerçekleştirilir. Geliştirilen evrişimsel sinir ağı mimarisinde test verisi üzerinde elde edilen doğruluk oranının, eğitim verisi üzerinden elde edilenle aynı veya daha yüksek olması beklenir. Hata oranında tersi durum söz konusudur. Test verisi üzerinden elde edilen doğruluk oranı, eğitim verisi üzerinden elde edilen doğruluk oranına yakın veya daha düşük ise iki özel durum söz konusudur: underfitting ve overfitting.

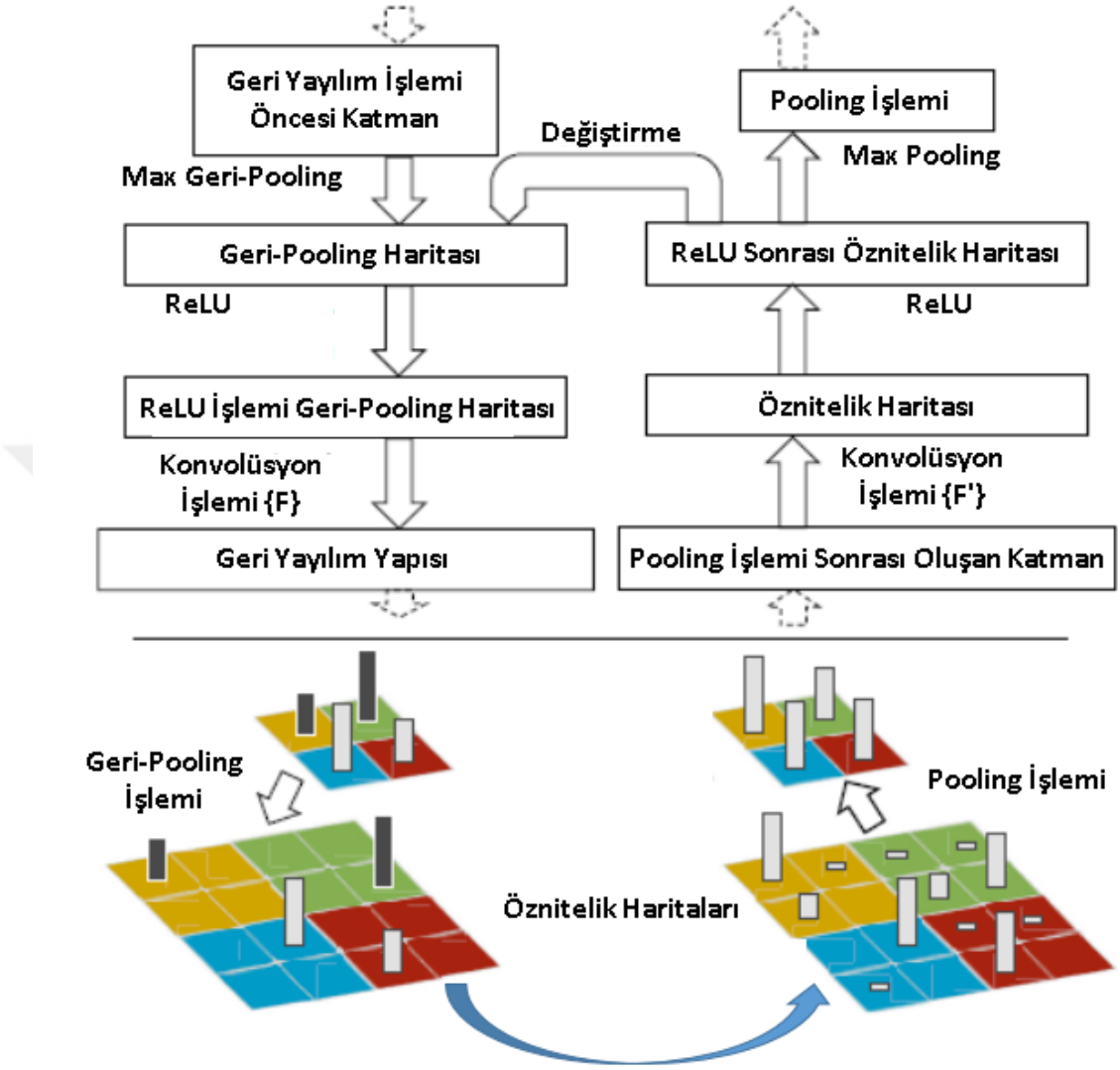
1. Underfitting : Geliştirilen evrişimsel sinir ağı mimarisi, test verisi üzerinde gerçeğe yakın tahminlerde bulunuyor ise underfitting'ten söz edilebilir. Modelin en iyi performansı gösterebilmesi için bazı değişikliklerin yapılması gerekir. Örneğin mimaride kullanılan bazı parametre değerlerinde veya veri seti üzerinde değişiklikler yapılabilir.
2. Overfitting : Geliştirilen evrişimsel sinir ağı mimarisi eğitim verisi üzerinde yüksek doğruluk elde ediyorken test verisi üzerinde düşük doğruluk oranı elde ediyorsa sistem ezberleme yapmış denilir. Yani sistem eğitim verilerini öğrenmeden ziyade ezberlemiştir. Böyle bir durumda parametreler üzerinden değişiklik yapılabileceği gibi katman sayısı azaltılarak model biraz daha basite indirgenebilir.

Şekil 2.28'de underfitting, istenen durum ve overftting durumlarını gösteren grafik verilmiştir. Grafikte görüldüğü üzere underfitting durumunda, test verisi üzerinden elde edilen hata oranı eğitim verisinden elde edilen hata oranına yakındır. İstenilen durumda ise oranlar birbirine yakındır. Oferfitting durumunda ise model eğitim verisini ezberlediğinden düşük hata oranı elde ediyorken test verisi üzerinde yüksek hata oranı elde eder. Makine öğrenmesinin amacına ters olan bu her iki durum için de bazı yöntemler geliştirilmiştir: çapraz doğrulama, dropout, normalleştirme, ensembling, erken durdurma ve veri arttırma gibi.



Şekil 2.28. Aşırı uyum regresyon grafiği [86]

İleri ve geri besleme ağ yapısı Şekil 2.29'da verilmiştir.



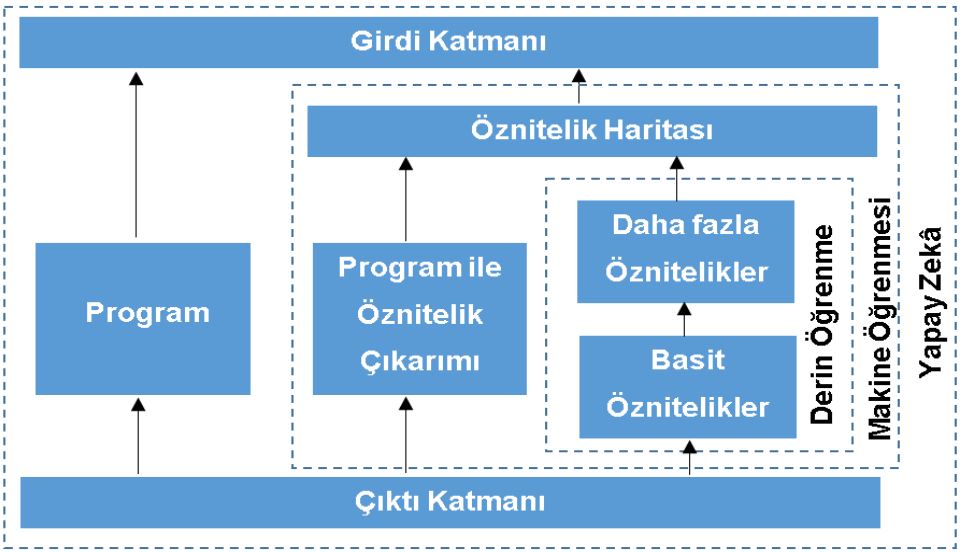
Şekil 2.29. İleri ve geri yayılım ağ yapısı örnek akış şeması

Şekil 2.29'da 1 epoch'ta gerçekleşen ileri ve geri yayılım aşaması gerçekleşmiştir. Şekildeki matrise ilk olarak pooling işlemi uygulanmıştır. Sonrasında yapılan konvolüsyon işlemi sonrası ReLU aktivasyonu işlemi gerçekleştirilerek öznitelik haritası oluşturulmuştur. En son yapılan pooling işlemi sonrasında geri yayılım aşamasında sırasıyla pooling, ReLU, konvolüsyon ve poolng işlemlerinin tersi gerçekleştirilmiştir.

2.6. Derin Öğrenme

Derin öğrenme yöntemi, konuşma veya nesne tanıma, tahminde bulunma ve video analizi yapma gibi insani işlerin makinenin yapabilmesini sağlayan makine öğrenmesi yöntemidir. Çok katmanlı yapay sinir ağlarından oluşur. Nesne tanıma için makine öğrenmesi yöntemlerinin hepsi kullanılabilir. Ancak son zamanlarda yapılan derin öğrenme yöntemi çalışmalarında başarı oranının yüksek; hata oranının düşük olması bu alana yönelimi hızlandırmıştır. Öyle ki yapılan çalışmalar, makinenin insandan daha az hata oranıyla nesne tanıma yaptığını göstermektedir.

Nesne tanımanın kuşkusuz en önemli aşamalarından birisi öznitelik çıkarımıdır. Makine öğrenmesi algoritmalarından önce hog, gabor, shift gibi yöntemlerle öznitelik çıkarımı yapılırken derin öğrenmede eğitim sırasında gizli katmanlarda gerçekleşmektedir. Bu da ham veriden sınıflandırılmış veriye kadar olan sürecin insan müdahalesi olmaksızın gerçekleştiğini gösterir. Bu durum derin öğrenme yöntemini diğer öğrenme yöntemlerinden ayıran en temel farktır. Makine öğrenmesi ve derin öğrenme arasındaki ilişki Şekil 2.30'daki gibi özetlenebilir.



Şekil 2.30. Yapay zekâ, makine öğrenmesi ve derin öğrenme ilişkileri şeması

Evrişimsel sinir ağı mimarisi, modelin performansını etkileyen farklı katmanlardan oluşur. Katmanlar: konvolüsyon, pooling, tam-bağlantılı olmak üzere dört farklı işlevi yerine getirir.

2.6.1. Konvolüsyon Katmanı

Konvolüsyon katmanı öznitelik çıkarımı ve seçiminin yapıldığı katmandır. Giriş veriler veya bir önceki katmandan alınan veriler üzerinde ağırlık adı verilen *nxn* matrisindeki filtreler gezdirilerek konvolüsyon işlemi uygulanır. Bu işlem ile öznitelik çıkarma işlemi yapılmış olur. Öznitelik haritası çıkarılarak bilgi çıkarımı sağlanır. Doğru bilgi çıkarımının yapılması sınıflandırma için önemlidir. Bu katmanda yapılan konvolüsyon işlemi derin öğrenme yöntemini diğer makine öğrenmesi yöntemlerinden ayırır. Yani farklı bir uygulama yapmaksızın aynı sistem içerisinde hem öznitelik çıkarma işlemi hem de öznitelik seçme işlemi gerçekleşmiş olur.

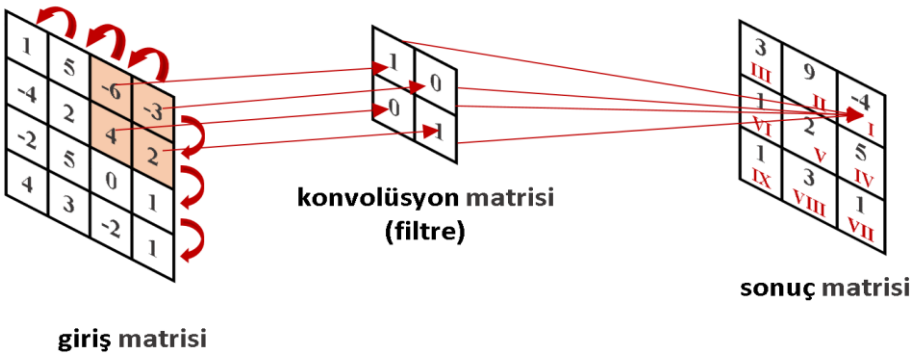
Şekil 2.31'de konvolüsyon katmanı öznitelik çıkarımı örneği gösterilmiştir. Şekilde tahmin verisi olarak araba verilmiştir. Tahmin verisi üzerinde yapılan öznitelik seçme işleminden sonra olasılıklı olarak sınıflandırma işlemi gerçekleştirilir. Bu katman araba sınıfına ait öznitelikleri seçme işlevi yapar ve tahmin verisini bir olasılık ile ilgili sınıfa atar.

Şekil 2.31. Konvolüsyon katmanı öznitelik seçme ve çıkarma işlemi [87]

Derin öğrenme yönteminde kurulan sistemin modeli geri yayılımlı olarak çalışır. Ağın ileri yayılım aşamasında, konvolüsyon katmanında kullanılan parametre değerleri rastgele oluşturulabileceği gibi istenilen değerler de atanabilir. Bu parametrelerle eğitilen sistemin hata oranını düşürmek amacıyla çıkış biriminden giriş birimine doğru olan geri yayılım aşamasında parametreler genellikle gradyan inişi yöntemi ile değiştirilir. Değişen parametreler hata oranı, ağırlık, öğrenme oranı ve bias değerleridir.

Şekil 2.32'de konvolüsyon işlemi verilmiştir. 4x4 boyutundaki giriş matrisine 2x2 boyutunda filtre uygulanarak 3x3 boyutunda sonuç matrisi elde edilmiştir. Filtre giriş matrisinin üzerinde gezdirilerek sonuç matrisi oluşturulur. Şekilde her seferinde 1 (bir) adım atlanarak konvolüsyon işlemi yapılmıştır. Atlama değeri 1'den farklı olabilir. Konvolüsyon işlemi temelde çarpımların toplamından oluşur. Konvolüsyon işlemi sonucunda oluşan sonuç matris değerleri roma rakamları ile gösterilmişti. Elde edilen 9 tane sonuç değeri hesaplamaları aşağıdaki gibidir:

I.(-3)*0+(-6)*1+2*1+4*0=-4
II.(-6)*0+5*1+4*1+2*0=9
III.5*0+1*1+2*1+(-4)*0=3
IV.2*0+4*1+1*1+0*0=5
V.4*0+2*1+0*1+5*0=2
VI.2*0+(-4)*1+5*1+(-2)*0=1
VII.1*0+0*1+1*1+(-2)*0=1
VIII.0*0+5*1+(-2)*1+3*0=3
IX.5*0+(-2)*1+3*1+4*0=1



Şekil 2.32. Konvolüsyon işlemi

Verilen örnekte görüldüğü üzere giriş verisi ile filtre adı verilen matrisi çarpılır ve sonuç matrisi oluşturulur. Bu yapılan işlemin genel gösterimi Eşitlik 2.16'da verilmiştir. Burada giriş matrisi, *x*; konvolüsyon matrisi *w* ile gösterilirken sonuç matrisi f(*x*) ile gösterilir.

Hata (Loss) Fonksiyonu

Tahmin verisi üzerinden elde edilen sonuç ile istenilen sonuç arasındaki fark hata oranını verir. Evrişimsel sinir ağı mimarisinde ortalama kare hatası olarak bilinen hata fonksiyonu kullanılır. Hata fonksiyonunun sıfıra yakın çıkması beklenir. Hataların kareler toplamı fonksiyonu Eşitlik 2.19'da verilmiştir.

$$\text{Hata} = \text{E} = \text{loss}(\text{d}, \text{p}) = \frac{1}{\text{c}} \sum_{\text{i}=1}^{\text{c}} (\text{d}_\text{i} - \text{p}_\text{i})^2 \tag{2.19}$$

Denklemde *d* gerçek değer ile *p* olasılık değer arasındaki farkın kareleri toplamı hesaplanır. Denklemdeki *c* değeri sınıf sayısını belirtir. Evli, bekar; kadın, erkek, başarılı, başarısız gibi iki sınıfa (etikete) sahip veri seti için *c* değişkeni 2 değerini alacaktır. Hata fonksiyonu tespit edilen bir evrişimsel sinir ağı mimarisinin gradyan iniş metodu ile hata fonksiyonunu minimize edecek ağırlık, bias ve öğrenme oranları tespit edilir.

Ağırlıklar (Weights)

Bir ağda çıktı değerleri ile istenen değerler arasındaki oranın (hata) en düşük olması beklenir. Makine daha sonra bu hatayı azaltmak için dahili ayarlanabilir parametrelerini değiştirir. Sıklıkla ağırlıklar olarak adlandırılan bu ayarlanabilir parametreler, makinenin giriş-çıkış fonksiyonunu tanımlayan "düğmeler" olarak görülebilen gerçek sayılardır. Bu sayılar veri tipine göre oluşturulabileceği gibi rastgele olarak da atanabilir.

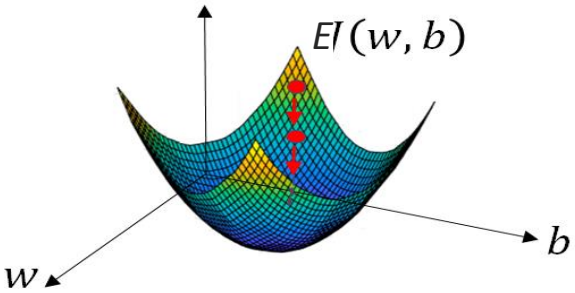
Doğru ağırlıklar bulunana kadar ağ eğitilir. İstenen çıktılar elde edilinceye kadar işlem tekrarlanır. Eğitim sürecinde bulunan uygun ağırlıklar test verilerine uygulanır ve

istenen sonuçların elde edilmesi beklenir. Eğer istenen sonuç elde edilirse ağ 'eğitilmiş model ağ' olarak kabul edilir. Ağırlıklar tek tek bakıldığında ne anlama geldikleri anlaşılmayabilir. Ağırlıklar girdiler ile ilgili analizin yapılmasına olanak sağladığından ağın zekasının ağırlıklarla yakından ilişkili olduğu söylenebilir [79]. Ağırlıklar problem türüne göre (ses, veri, görüntü gibi) belirlenir.

Görüntü verilerinde uygulanan derin öğrenme yönteminde ağırlıklar matris (*n*x*n*) görünümündedir. Matristeki değerlerin doğru belirlenmesi sınıflandırma için önemlidir. Ağırlık değerleri öznitelik matrisi olarak da adlandırılır. Oluşturulan öznitelik matrisi görüntü verisi üzerinde gezdirilmek suretiyle bir dizi işlemler uygulanır ve öznitelik haritası oluşturulur. Öznitelik haritaları test verileri üzerinde de uygulanarak sınıflandırma işlemi gerçekleştirilir. Dolayısı ile öğrenmeye yardımcı olan doğru özellik haritaların oluşturulabilmesi için ağırlıkların tespiti önemlidir. Geri beslemeli ağlarda ağırlık parametresi çıkış biriminden giriş birimine doğru güncellenir. Ağırlık üzerindeki değişim miktarı eşitlik delta kuralına göre hata fonksiyonunun ağırlığa göre türevi alınarak bulunur. Ağırlık değişim miktarı Eşitlik 2.20'de verildiği gibidir.

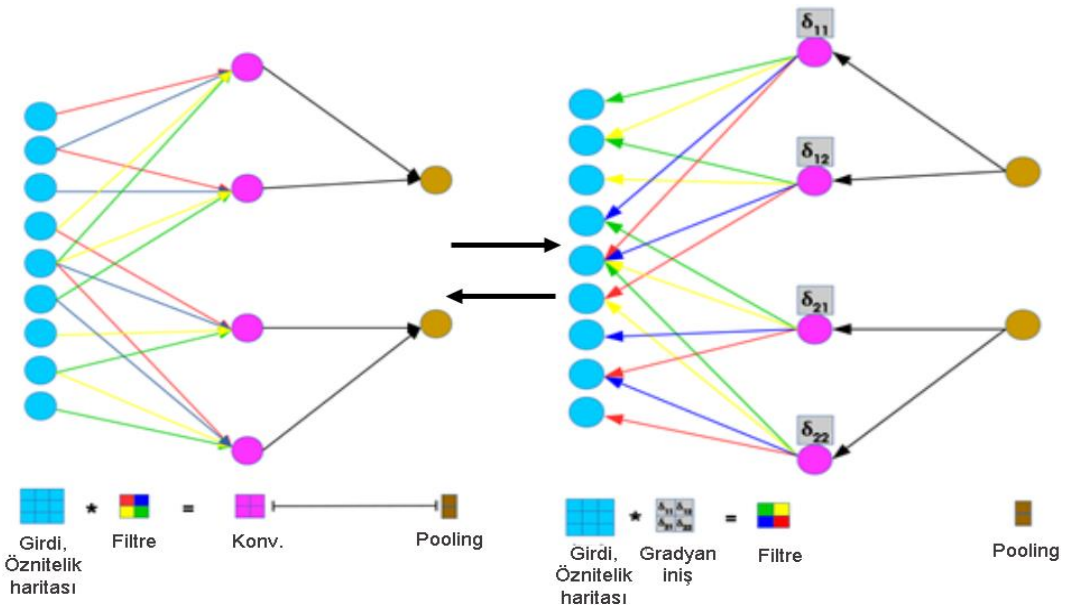
$$\Delta W_{ji} = -\eta \frac{\partial E}{\partial W_{ji}} \tag{2.20}$$

Eşitlik 2.20'de verilen η değeri öğrenme oranıdır. Bir evrişimsel sinir ağı mimarisinde kaç adet bağlantı varsa o kadar ağırlık değeri vardır. Eğitimde rastgele girilen ağırlık değeri doğru sonuç bulununcaya kadar ara katmanlarda güncellenir. Buradaki amaç en düşük hata oranı ile en uygun ağırlık değerlerini bulmaktır. Tespit edilen ağırlıklar genellikle [0,1] aralığındaki değerlerdir. Öğrenme sonunda her bir iterasyonda ağırlık değerleri hata değerine göre geri yayılımlı (backpropagation) ile "Gradyan İniş" yöntemine göre tekrar hesaplanıp güncellenir. Şekil 2.33'te bir *E(w, b)* noktası belirleyelim. *w* ağırlık *b* ise bias değerleridir. Bu nokta gradyan iniş yöntemi için başlangıç noktası olsun. *E* değerinin azaltılabilmesi için *w* ve *b* değerlerinde küçük değişiklikler yapılır. Eğitim sırasında her iterasyonda gradyana bağlı olarak ağırlıklar azaltılır. İşlemlerin sonucunda *E* değerini minimum yapacak *w* ve *b* değerleri elde edilir.



Şekil 2.33. Gradyan iniş yöntemi grafiği [88]

Şekil 2.34'te girdi verisi ile pooling katmanı arasında uygulanan konvolüsyon katmanı verilmiştir. 2x2 boyutunda ağırlık matrisinin her birisi farklı değer alacak şekilde farklı renkteki oklarla gösterilmiştir. Girdi verisi ile ağırlık matrisi çarpılıp oluşturulan öznitelik haritası pooling katmanına iletilmiştir. Ağın geri besleme aşamasında çıkış birimindeki hata oranı dikkate alınarak gradyan iniş yöntemi ile ağırlık matrisinde değişiklik yapılmış ve bu değişim sonucunda oluşturulan filtre farklı renklerle gösterilmiştir. Ağın geri yayılım aşamasında yeni ağırlık değerleri kullanılır. Ağ, bu şekilde ileri ve geri yayılarak döngüsel olarak eğitilir.



Şekil 2.34. İleri ve geri beslemeli ağlarda konvolüsyon katmanı [89]

Bias

Bias değeri, verilere daha iyi uyum sağlamak için aktivasyon işlevinin sola veya sağa kaydırılmasına izin verir. Bu nedenle, ağırlıklardaki değişiklikler sigmoid eğrisinin yüksekliğini değiştirirken, bias onu dengeleyerek, eğriyi çözüm için düzenli konuma getirir. Ayrıca bias (b) yalnızca çıkış değerlerini etkiler, gerçek girdi verileriyle etkileşime girmez. Bunun dışında bias değeri fonksiyonun sonsuz sıfır döngüsüne girmesini önler.

Girdi (x) veya ağırlık (w) değerinin sıfır olduğu durumlarda bias değeri fonksiyon çıktısını öteleyerek sonraki iterasyonlarda öğrenmeyi gerçekleştirmesini sağlar. Bias değeri genellikle belli aralıklardaki değerler dikkate alınarak ağırlıkla birlikte güncellenir ancak sabit olarak da belirlenebilir. Her katmanda kaç adet ağırlık varsa o kadar bias değeri vardır ve algılayıcının azaltıp arttırılması bias değerini de arttırıp azaltır.

Momentum Katsayısı

Ağın eğitilme hızını arttırmak için momentum katsayısı kullanılır. Ağ eğitilirken kullanılan parametrelerden elde edilen sonucun, bir sonraki eğitimde bir kısmının eklenmesi esasına dayanır. Ağın sürekli tekrarını önleyerek öğrenmesini sağlar [90]. Yani öğrenme sırasında ağın yerel bir optimuma takılmaması için ağırlıktaki değişim değerinin bir kısmını bir sonrakine ekler.

Öğrenme Oranı

Ağ eğitilirken ağırlık değerleri doğru sonuca erişebilmek için değişir. Bu değişim miktarı öğrenme oranı olarak tanımlanır. Öğrenme oranı düşükse, eğitim daha güvenilirdir, ancak optimizasyon çok zaman alacaktır, çünkü minimum kayıp fonksiyonu için adımlar çok küçüktür. Öğrenme oranı yüksekse, Eşitlik 2.20'de verilen denklemde olduğu gibi ağırlıktaki değişim miktarı artar dolayısı ile de hata fonksiyonu artar. Bu da oluşturulan ağ için istenmeyen bir durumdur.

Örneğin bir çocuğun bitki yapraklarını ilk kez gördüğünü varsayalım ve 100 çeşit yalnızca geniş yapraklı bitki gösterelim. Çocuk tüm bitki yapraklarının geniş yapraklı olduğunu düşünür. Çam ağacı gördüğünde ise iğne yapraklı bitkilerin de olabileceğini öğrenir. Öğrenme durumunu bir öğrenme oranı ile ölçeklendirebiliriz. Çok küçük bir öğrenme oranıyla yaprakların hala geniş yapraklı bitkiler olduğunu ve çam ağacının bir istisna olduğunu düşünebilir. Çok büyük bir öğrenme oranıyla ise tüm yaprakların iğne yapraklı olduğunu düşünebilir. Bu sebeple öğrenme oranının optimum düzeyde tutulması önemlidir. Genel olarak, ağın yararlı bir şeye yakınlaştığı kadar düşük bir öğrenme oranı bulmak istiyoruz, ancak bunu eğitmek için uzun zaman harcamak gerekebilir.

Macêdo [91] tarafından yapılan çalışmada öğrenme oranının doğruluk oranını etkilediği tespit edilmiştir. Çalışmada iki kez derin öğrenme yöntemi uygulanmıştır. İlk çalışmada öğrenme oranı 0,1; ikinci çalışmada ise 0,01olarak belirlenmiş ve diğer parametreler sabit tutulmuştur. MNIST veri seti üzerinde 0,1; CIFAR-10 veri setinde 0,01; CIFAR-100 veri setinde ise 0,01 değeri daha yüksek performans göstermiştir. Bu da öğrenme oranının veri setine uygun olarak belirlenmesi gerekliğini gösterir.

Düşük öğrenim oranları ile gelişmeler doğrusal olacaktır. Yüksek öğrenme oranları ile de daha fazla üssel görünmeye başlayacaklar. Daha yüksek öğrenme oranları hata oranını daha hızlı düşürür, ancak daha kötü hata oranında (yeşil çizgi) sabitlenir. Bunun nedeni, optimizasyonda çok fazla "enerji" bulunması ve parametrelerin düzensiz bir şekilde değişmesi, optimizasyon ortamındaki güzel bir noktaya yerleşememesidir [92].

Aktivasyon Fonksiyonu

Konvolüsyon işleminden sonra elde edilen yapay sinir ağları modeli, basit bir doğrusal regresyon modelidir. Doğrusal regresyon modelinin çözümü kolay olmakla birlikte karmaşık problemler üzerinde etkin çözümler sunmayabilir. Sınırlı işlem gücüne sahip ve çoğu zaman istenen performansı gösteremeyen bir modeldir. Daha açık bir ifadeyle, doğrusal bir modelin resim, video ses gibi karmaşık problemleri öğrenip modellemesi oldukça zordur. Bununla birlikte konvolüsyon işleminden sonra elde edilen çıktılar bir

sonraki katmana iletilmeden önce belirli bir eşik değerine tabi tutulur. Eşik değeri nöronu aktif veya pasif eder. Aktivasyon fonksiyonu, algılayıcının doğru bilgileri elde edip etmediğini test ederek bir nöronun aktif olup olmayacağını belirler. Daha açık bir ifadeyle aktivasyon fonksiyonu ile öznitelik seçme işlemi gerçekleştirilir. Yani konvolüsyon işlemi ile oluşturulan öznitelik haritası içerisinden sınıflandırmayı etkileyen öznitelikler seçilmiş olur.

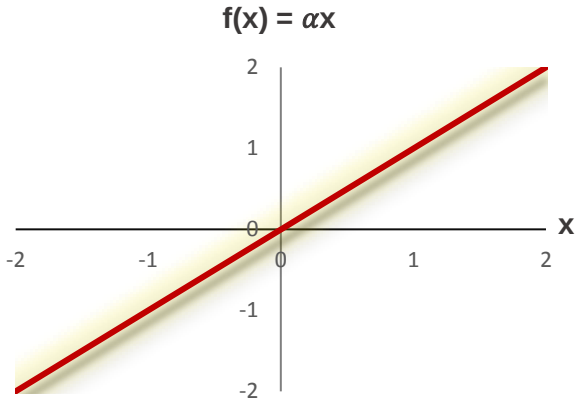
Aktivasyon fonksiyonu, giriş sinyali üzerinden yaptığımız doğrusal olmayan bir dönüşümdür. Bu dönüştürülmüş çıktı, daha sonra bir sonraki nöron katmanına girdi olarak iletilir. Doğrusal olmayan dönüşüm sayesinde ağ daha karmaşık görevleri öğrenip gerçekleştirebilir. Dil çevirileri ve görüntü sınıflandırmaları gibi karmaşık görevler üzerinde çalışabilir. 'Evet, hayır', 'başarılı, başarısız', 'kadın, erkek' gibi ikili sınıflandırıcılar için son katmandaki aktivasyon işlemi kolaydır. Zira sınıflandırma işlemi için nöronun birinin aktif diğerinin pasif olması beklenir. Ancak birden fazla etiketli sınıflandırıcılar için işlem daha zordur. Çünkü tek bir nöronun aktif diğerlerinin pasif olması gerekirken, tüm nöronların aktif olması büyük bir sorundur. Örneğin %80 oranında aktif, %20 oranında pasif olması daha tercih edilebilir bir durumdur. Yaygın olarak kullanılan aktivasyon fonksiyonları açıklanmıştır. Toplama fonksiyonundan çıkan sonuç için x değişkeni kullanılırsa aktivasyon fonksiyonu f(x) ile gösterilir. Buradaki x değeri Eşitlik 2.16'da verilen denklemin sonucuna eşittir.

a. Doğrusal Aktivasyon Fonksiyonu

Doğrusal problemlerin çözümünde doğrusal bir aktivasyon fonksiyonu seçilebilir. Konvolüsyon işleminde elde edilen sonuç ile sabit bir sayı çarpılarak aktivasyon işlemi gerçekleştirilir. Doğrusal bir fonksiyon Eşitlik 2.21'de gösterildiği gibi birinci dereceden bir polinomdur.

$$doğrusal(x) = f(x) = \alpha x \tag{2.21}$$

Eşitlikteki x değeri konvolüsyon işleminden sonra elde edilen çıktıyı temsil ederken, α değeri sabit bir değerdir ve doğrunun çıkış ekseni ile yaptığı açıyı değiştirir. Eşitlik 2.21'deki α değeri 1 (bir) olarak alındığında edilen doğrusal aktivasyon grafiği Şekil 2.35'teki gibidir.



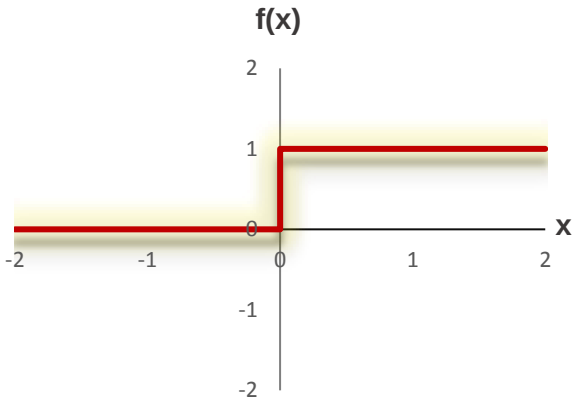
Şekil 2.35. Doğrusal aktivasyon grafiği

Doğrusal aktivasyon işleminden sonra nöronlardan bir tanesi aktif olur. Eğer birden fazla nöron aktif edilirse en yüksek değer alan nörona göre değerlendirme yapılır. Geri yayılımlı bir yapay sinir ağları modelinde, ileri yayılım aşamasında kullanılan doğrusal bir aktivasyon fonksiyonunun geri yaylım aşamasında türevi alınır. Doğrusal fonksiyonlar birinci dereceden polinomlar olduğundan türev işleminden sonra x değeri silinerek yalnızca α değeri kalır. Bu da ağırlık ve bias gibi parametrelerin silinmesi anlamına gelir ve istenmeyen bir durumdur. Bir sonraki yayılım işleminde ise, sabit bir değer olan α üzerinden işlem yapılır. Derin öğrenmenin amacı olan hata fonksiyonunun indirilmesini sağlayan gradyan iniş yöntemi x değeri olmaksızın uygulanamaz. Dolayısı ile doğrusal aktivasyon fonksiyonu karmaşık problemlerin çözümünde yetersiz kalır. Bu yüzden aktivasyon işlemi için doğrusal olmayan fonksiyonlar tercih edilebilir.

b. Adım (Step) Aktivasyon Fonksiyonu

Adım fonksiyonu, toplam fonksiyonundan gelen sonucu sıfırdan büyük olup olmamasına göre değerlendirerek -1 veya 1 olmak üzere iki farklı çıktı verir. Sonucun kontrolü yalnızca sıfır değer ile yapılmayabilir. Problem türüne uygun eşik değeri belirlenerek de aktivasyon işlemi yapılabilir. Adım aktivasyon işlemi Eşitlik 2.22'deki gibi tanımlanır. Fonksiyona göre çizilen grafik Şekil 2.36'daki gibidir.

$$f(x) = \begin{cases} 0 \text{ için } x < \text{eşik değer} \\ 1 \text{ için } x \geq \text{eşik değer} \end{cases} \quad (2.22)$$

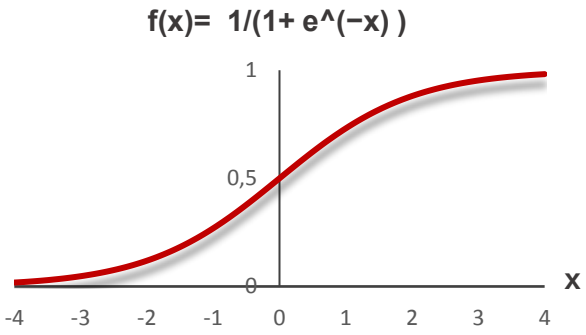


Şekil 2.36. Atlama fonksiyonu aktivasyon grafiği

c. Sigmoid Aktivasyon Fonksiyonu

Sigmoid aktivasyon fonksiyonu, sürekli türevlenebilir bir fonksiyondur. Değişim aralığı 0 ile 1 arasında olan fonksiyon Eşitlik 2.23'te verilmiştir. Sigmoid fonksiyonu ile çizilen grafik Şekil 2.37'deki gibidir.

$$sigmoid(x) = f(x) = \frac{1}{1 + e^{-x}} \quad (2.23)$$



Şekil 2.37. Sigmoid aktivasyon fonksiyonu grafiği

Sigmoid fonksiyonu türevlenebilir olmasına rağmen bazı sınırlılıkları vardır Bunlar:

1. Gradyanın Sıfıra Yakınması Sorunu: Aktivasyon fonksiyonunun ürettiği sonuçlar [0,1] aralığında olduğundan geri yayılım sırasında gradyan iniş yöntemi ile elde dilen değerler sıfıra yakınsar. Diğer bir ifadeyle, aktivasyon fonksiyonuna gelen bilgiler aktivasyon işlemi sonrasında kaybolur veya azalır. Önemli değişiklikler olmadığından öğrenme gerçekleşmez.
2. Gradyan Güncelleme Sorunu: Fonksiyon merkezi 0 (sıfır) olmadığından 0 ile 1 arasındaki optimizasyon zorlaşır.
3. Yavaş Yakınsama Sorunu: Derin öğrenme için hata oranının azaltılması önemlidir. Hata oranının azalma hızı düşüktür.

d. Tanh Aktivasyon Fonksiyonu

Tanh aktivasyon fonksiyonu sigmoid aktivasyon fonksiyonuna benzer. Sigmoid ve Tanh aktivasyon fonksiyonu arasındaki ilişki Eşitlik 2.24'te verilmiştir.

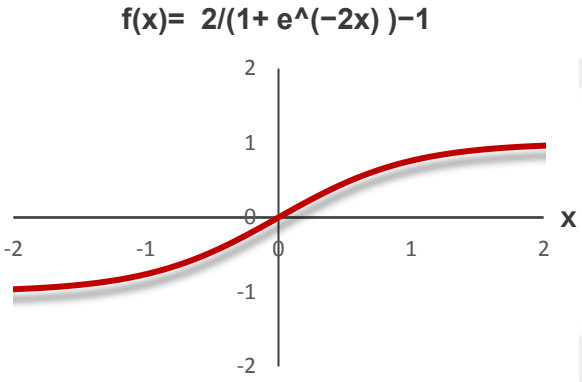
$$tanh(x) = f(x) = 2sigmoid(2x)\text{-}1 \qquad (2.24)$$

Eşitlik 2.38'de görüldüğü gibi fonksiyon sigmoid fonksiyonuna göre daha dik bir eğim gösterir. Gradyan inişinin sigmoid fonksiyonuna göre daha dik olması tanh fonksiyonunu avantajlı hale getirmiştir [93]. Ancak Gradyanın Sıfıra Yakınsaması

sorunu tanh için de söz konusudur. İki sınıflı problemler için kullanışlı bir fonksiyondur. Tanh aktivasyon fonksiyonu Eşitlik 2.25'te verildiği gibidir.

$$f(x) = \frac{2}{1 + e^{-2x}} - 1 \tag{2.25}$$

Eşitlikte 2.24'te verilen fonksiyonun değişim aralığı [-1,1]'dir. Dolayısı ile negatif girdiler negatif çıktılarla eşleşir. Fonksiyonun grafiği Şekil 2.38'de verilmiştir.



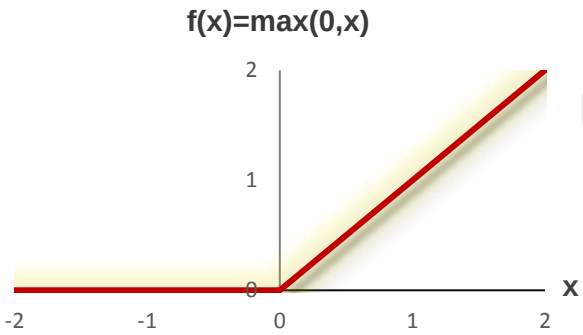
Şekil 2.38. Tanh aktivasyon fonksiyonu grafiği

e. ReLU (Rectified Linear Unit) Aktivasyon Fonksiyonu

En çok kullanılan aktivasyon fonksiyonu ReLU fonksiyonudur. Doğrusal fonksiyonlarda genellikle sıfır veya bire yaklaşma eğilimlerine göre sınıflandırma yapılır. Doğrusal olmayan fonksiyonlarda ise 2 asimptot arasına sınırlandırmaksızın [0, ∞] arasında çıkış değeri üretilir. ReLU doğrusal bir fonksiyon olmadığından dolayı geri beslemeli ağlarda parametre değişimine olanak tanıdığından başarılı sonuçlar elde eder.

Eşitlik 2.26'da gösterildiği gibi, *x* negatif değerler aldığında fonksiyon 0 (sıfır) değeri üretirken, *x* pozitif değerler aldığında fonksiyon max değerleri döndürecektir. Bu da max değer alan nöronların aktif olacağı anlamına gelir. Eşitlik 2.26'da göre verilen koşula göre çizilen grafik Şekil 2.39'daki gibidir.

$$f(x) = \begin{cases} 0 \text{ için } x < 0 \\ x \text{ için } x \geq 0 \end{cases} \quad (2.26)$$



Şekil 2.39. ReLU aktivasyon fonksiyonu grafiği

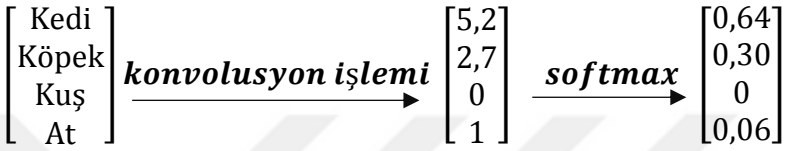
ReLU aktivasyon fonksiyonu derin öğrenme yönteminde gizli katmanlarda kullanılır. Genellikle sondan bir önceki katmanda kullanılan aktivasyon fonksiyonudur. ReLU ağırlık güncellemesi için kullanılan ideal bir fonksiyondur [94]. Son katmanda sınıflandırma işleminin gerçekleştirilebilmesi için genellikle softmax fonksiyonu kullanılır.

f. Softmax Aktivasyon Fonksiyonu

Softmax aktivasyon fonksiyonu ile sigmoid fonksiyonunda olduğu gibi [0,1] arasında değerler üretilir. Aynı zamanda tüm çıkışların toplamı 1 olacak şekilde bölünmesini sağlar ve sınıfların herhangi birinin doğru olma olasılığını belirlenir. Örneğin kedi,

köpek, kuş ve at resimlerinin olduğu bir veri setinde nesne tanıma uygulaması yapıldığını varsayalım.

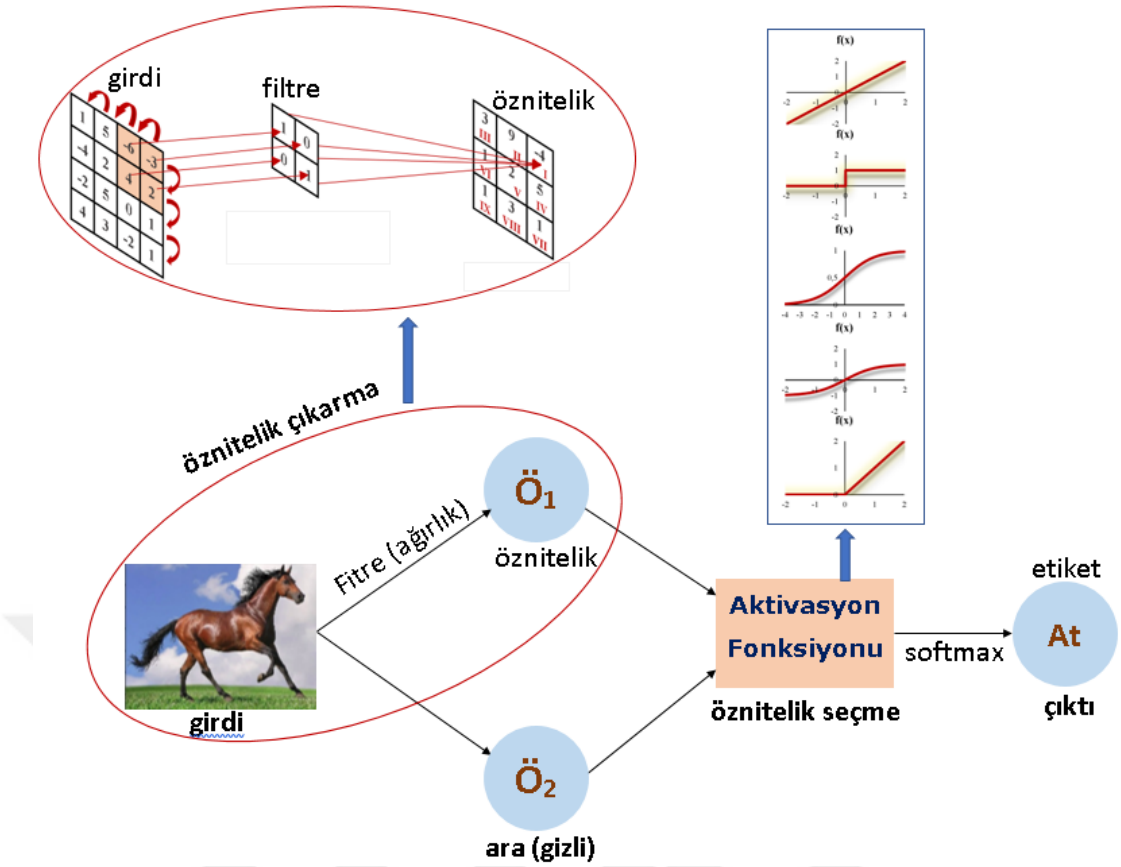
Kedi resmi için softmax aktivasyon fonksiyonu Şekil 2.40'taki gibi bir sonuç üretebilir. Uygulanan softmax aktivasyon fonksiyonu ile %64 kedi, %30 köpek %0 kuş ve %6 at resmi olma ihtimali elde edilerek sınıflandırma işlemi yapılır. Sonuç olarak en yüksek ihtimal (%64) ile resmin kedi sınıfına ait olduğu söylenebilir.



Şekil 2.40. Softmax aktivasyon fonksiyonu olasılık sonuçları gösterimi

Softmax aktivasyon fonksiyonunda diğer fonksiyonların aksine sınıflandırma işlevi yerine getirdiğinden bir nöronun aktif olması beklenir. Diğer aktivasyon fonksiyonlarında öznitelik seçimi yapıldığından birden fazla nöron aktif olabilir. Herhangi bir aktivasyon fonksiyonunun tüm problemler için aynı oranda başarı elde etmesi mümkün değildir. Bir problem için doğru belirlenen aktivasyon fonksiyonu yüksek oranda başarı elde edebileceği gibi daha kısa zamanda öğrenme gerçekleşecektir. Örneğin iki boyutlu sınıflandırma için sigmoid aktivasyon fonksiyonu daha açıklayıcı sonuç verebileceği gibi ReLU aktivasyon fonksiyonu da çok boyutlu sınıflandırma için doğru sonuçlar verebilir.

Bir tane ara katmana sahip basit yapıda evrişimsel sinir ağı Şekil 2.41'de verilmiştir.



Şekil 2.41. Bir katmanlı basit evrişimsel sinir ağı mimarisi

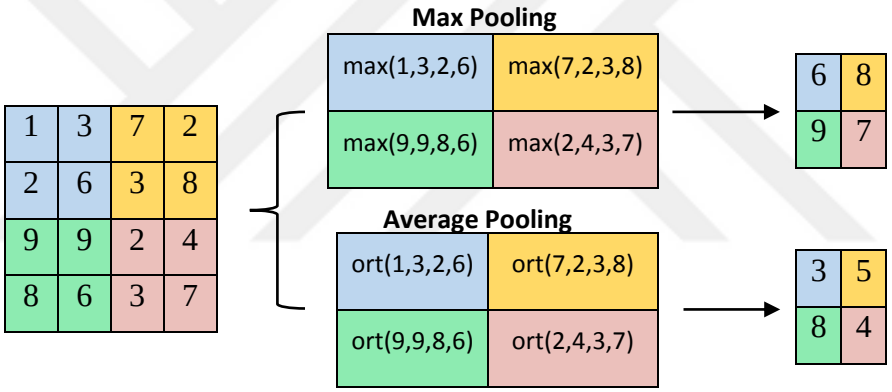
4x4x3 boyutundaki görüntü üzerinde 2x2 boyutundaki filtre ile konvolüsyon işlemi yapılmıştır. Öznitelik seçme işleminin yapıldığı aktivasyon işlemi sonrası softmax ile sınıflandırma yapılmış ve çıktı elde edilmiştir.

2.6.2. Pooling Katmanı

Aktivasyon katmanından sonra elde edilen aktivasyon haritasının özelliğini kaybetmeyecek şekilde uzamsal boyutunun azaltılması pooling katmanı ile mümkün olur. Daha az uzamsal bilgi daha az parametre anlamına gelir ve bu da modelin hesaplama performansını arttırır. Bir diğer faydası da aşırı uyumu (ezber) engeller. Konvolüsyon katmanında olduğu gibi veri üzerinde filtre gezdirme esasına dayanır.

Örneğin aktivasyon fonksiyonundan sonra elde edilen veri üzerinde 2x2 matrisinde bir filtre gezdirilsin. 2x2 boyutundaki filtre giriş verisinin üzerinde uygulandığında filtrelenen 4 değerden en yüksek olanı (max pooling) veya 4 değerin ortalaması (average pooling) yeni matrise aktarılır. Filtrelenen bölgeyi temsil eden değer aktarılmış olur. Böylece 2x2 matris boyutundaki veri 1x1 matris boyutuna indirgenmiş olur. Evrişimsel sinir ağı mimarisinde konvolüsyon katmanından sonra pooling katmanı eklemek yaygındır.

Şekil 2.42'de 4x4 matris görüntü üzerine 2x2 matris boyutunda max(en yüksek) ve average (ortalama) pooling filtresi uygulanmıştır. Atlama sayısı (stride)'nın 2 olduğu pooling işlemi sonucu görüntü boyutu 2x2 matris boyutuna indirgenmiştir.

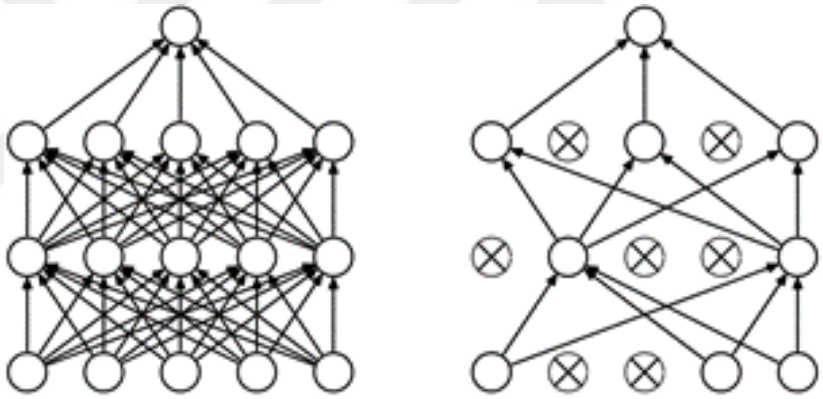


Şekil 2.42. Görüntü işlemede max ve average pooling katmanı uygulaması

Çok fazla pooling katmanının zararlı olabileceği açıktır. Önemli bir soru, belirli bir görüntü sınıflandırma problemi için kaç tane pooling katmanına ihtiyaç olduğudur. Pooling katmanlarının sayısı konvolüsyon katman sayısından daha büyük olamaz. Her bir konvolüsyon katmanının pooling katmanı ile devam ettirilmesi gerekip gerekmediği, tartışmaya açık başka bir konudur [95].

2.6.3. Dropout Katmanı

Dropout, aşırı uyumu önler ve üstel olarak birçok farklı sinir ağı mimarisini verimli bir şekilde birleştirme olanağı sağlar. “Dropout” terimi, bir sinir ağında birimlerin (gizli ve görünür) bırakılması anlamına gelir. Bir birimi düşürerek, tüm gelen ve giden bağlantılar ile birlikte ağdan geçici olarak kaldırmak kastedilir [96]. Dropout ile ağın ezberlemesine neden olan düğümlerin her iterasyonda farklı düğümler olacak şekilde geçici olarak devre dışı bırakılması esasına dayanır. Genellikle 0 ile 1 arasında değişen olasılık değerlerine göre katmanda bazı düğümler devre dışı kalır. Dropout ile aslında ağırlıklar sıfırlanır ve bu şekilde bazı nöronlar pasif olur. Şekil 2.43’te iki gizli katmana sahip bir sinir ağında üç katmana dropout katmanı uygulanmış bir sinir ağı modeli verilmiştir.



(a) **(b)**

Şekil 2.43. Dropout katmanı modeli **(a)** İki gizli katmanlı sinir ağı **(b)** Dropout uygulaması ile elde edilmiş sinir ağı modeli [96]

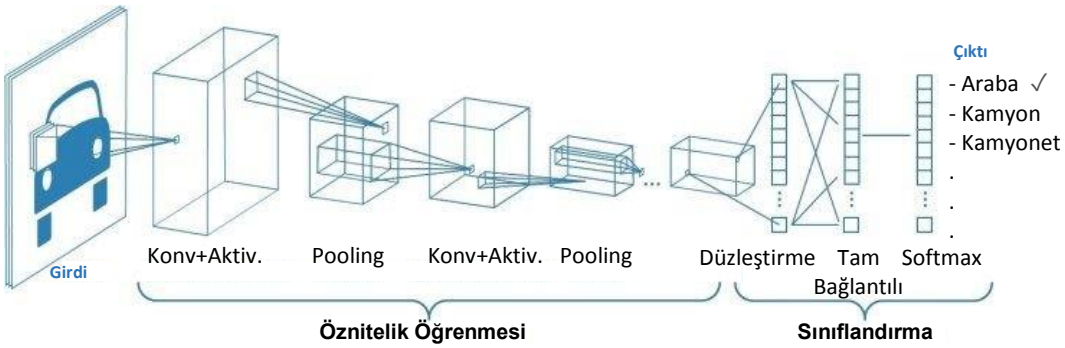
2.6.4. Tam-Bağlantılı (Full-Connected) Katmanı

Tam bağlantılı katmanı sınıflandırmayı gerçekleştiren katmandır. Bu katman girdi verilerini (ReLu, Pooling, Dropout vb.) alır ve k sınıflı bir sınıflandırıcı için k boyutlu vektör çıktısı verir. Gelen girdiler arasında hangi özelliğin hangi sınıfa ait olduğunu

belirler. Örneğin tüy, gaga, kanat gibi öznitelikleri kuş olarak sınıflandıracaktır. Tam-Bağlantılı katmanı, belirli bir sınıfla en güçlü şekilde ilişkili olan yüksek seviyeli özniteliklere bakar ve sınıflar için doğru olasılıklar elde edilir.

Tam-bağlantılı katmanda bir nöron kendinden önceki tüm nöronlara bağlıdır. Evrişimsel sinir ağı mimarisinde ard arda gelen konvolüsyon, aktivasyon ve havuzlama katmanından sonra tam bağlantılı katman gelir. Bu katman kendinden önceki katmanın tüm alanlarına bağlıdır. Farklı mimarilerde bu katmanın sayısı değişebilir. Evrişimsel sinir ağı mimarisinde en son katmanın üretmiş olduğu matris boyutu 25x25x256=160000x1 ve tam bağlantılı katmandaki matris boyutu 4096x1 olarak seçilirse, toplamda 160000x4096 ağırlık matrisi oluşur. Yani her bir 160000 nöron 4096 nöron ile bağlanmaktadır. Bu sebepten dolayı bu katmana tam bağlantılı katman denilmektedir [97]. Şekil 2.44'te birçok konvolüsyon katmanına sahip bir ağ örneği verilmiştir.

Filtreler, her çözünürlük görüntüsüne farklı çözünürlüklerde uygulanır ve her bir katlanmış resmin çıkışı, sonraki katmana girdi olarak kullanılır. Konvolüsyon ve pooling işlemlerinden sonra düzleştirme işlemi uygulanmıştır. Tam bağlantılı katmandan sonra softmax işlemi uygulanarak sınıflandırma gerçekleştirilmiştir.



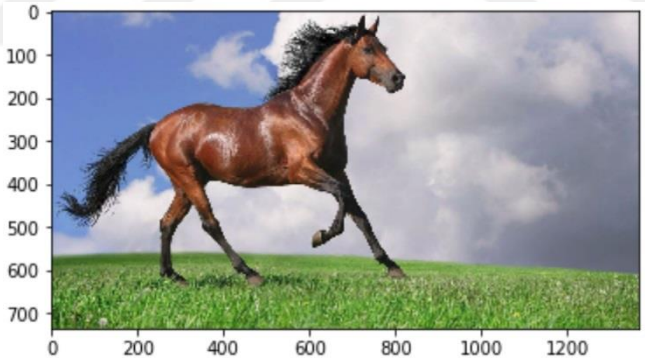
Şekil 2.44. Yapay sinir ağı yapısı [98]

2.6.5. Diğer Kavramlar

Yapay sinir ağı modeli belirlendikten sonra verilere ve ağa bazı işlemler uygulanır. Uygulanacak işlemler için kullanılan parametrelerde yapılacak küçük değişiklikler sistemin başarı düzeyini arttırabilir.

Veri Arttırma (Data Augmentation)

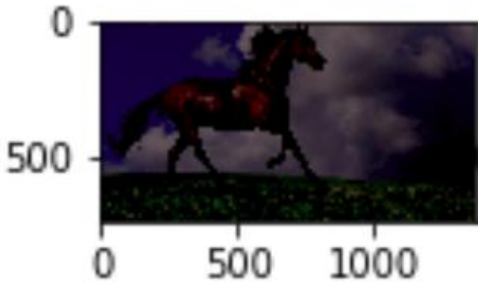
Herhangi bir nesnenin doğru olarak sınıflandırılabilmesi için kullanılacak veri sayısının sistemin öğrenebileceği yeterlilikte olması gerekir. Mevcut verilerin eğitim için yeterli olmadığı durumlarda veri arttırma işlemi uygulanır. Mevcut veriler üzerinde bir nevi filtreleme işlemleri (çevirme, uzatma, bulanıklaştırma gibi) uygulayarak veri sayısının arttırılması işlevi, veri arttırma olarak adlandırılır. Yapay sinir ağlarının bu kısımda, uygulanmak istenen filtreler yazılarak eğitime eklenir. Veri arttırma işlemleri uygulanacak görüntü Şekil 2.45'te verildiği gibidir.



Şekil 2.45. Veri arttırma işlemleri için seçilen görüntü

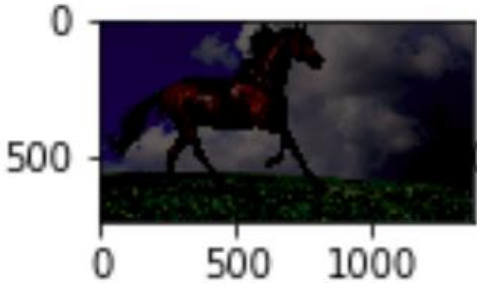
Şekil 2.45'te verilen görüntüye aşağıdaki parametreler uygulanmış ekran çıktıları verilmiştir.

a. **featurewise_center parametresi**: Tüm girdi verilerinin ortalama piksel değerlerini 0'a eşitler Parametre değerleri *boolean* veri tipindedir. Seçilen görüntüye *featurewise_center* parametresinin uygulanması sonucu elde edilen görüntüler Şekil 2.46'daki gibidir.



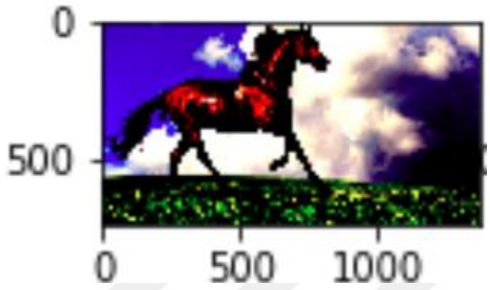
Şekil 2.46. featurewise_center parametresi uygulanmış görüntü

b. **samplewise_center parametresi**: Her bir örnek girdi verisinin piksel ortalamasını ayrı ayrı 0'a eşitler. Parametre değerleri *boolean* veri tipindedir. Seçilen görüntüye *samplewise _center* parametresinin uygulanması sonucu elde edilen görüntüler Şekil 2.47'deki gibidir.



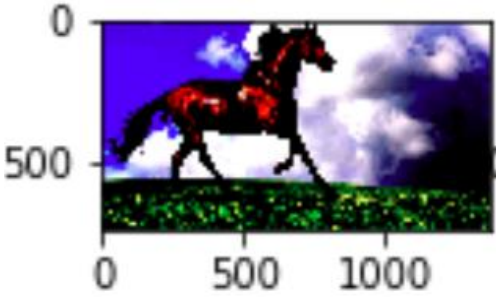
Şekil 2.47. samplewise_center parametresi uygulanmış görüntü

c. **featurewise_std_normalization**: Giriş verileri pikselleri veri setinin standart varyans değerine bölünür. Parametre değerleri *boolean* veri tipindedir. Seçilen görüntüye *featurewise_std_normalization* parametresinin uygulanması sonucu elde edilen görüntüler Şekil 2.48'deki gibidir.



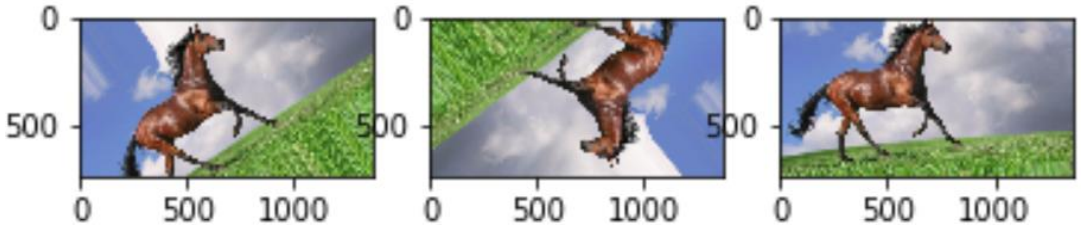
Şekil 2.48. featurewise_std_normalization uygulanmış görüntü

d. **samplewise_std_normalization**: Her bir örnek girdi verisi pikselleri veri setinin standart varyans değerine bölünür. Parametre değerleri *boolean* veri tipindedir. Seçilen görüntüye *samplewise_std_normalization* parametresinin uygulanması sonucu elde edilen görüntüler Şekil 2.49'daki gibidir.



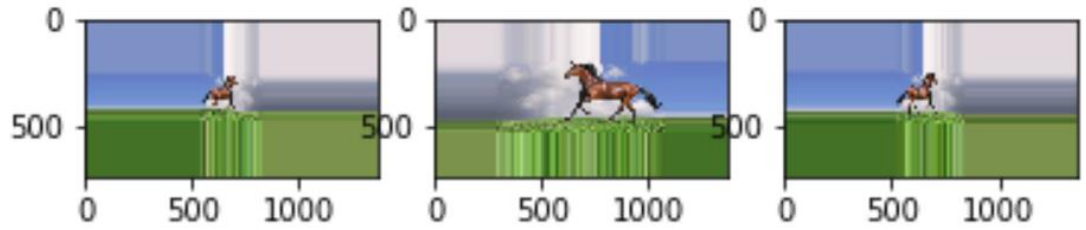
Şekil 2.49. samplewise_std_normalization uygulanmış görüntü

e. **rotation_range parametresi**: Görüntüler belirli bir değer ile rastgele olarak döndürülür. Değerler *integer* veri tipindedir. Seçilen görüntüye *rotation_range* parametresinin uygulanması sonucu elde edilen görüntüler Şekil 2.50'deki gibidir.



Şekil 2.50. rotation_range parametresi uygulanmış görüntü

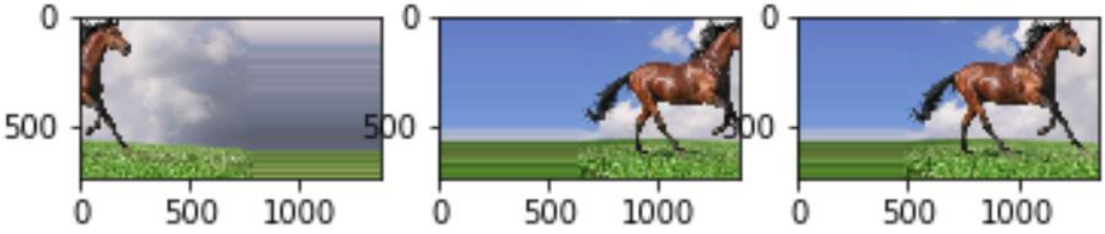
f. **zoom_range**: Giriş verileri rastgele olarak yakınlaştırılır veya uzaklaştırılır. *Float* veri tipinde değerler seçilebileceği gibi [*lower*, *upper*] ifadelerinden birisi yazılabilir. Seçilen görüntüye *zoom_range* parametresinin uygulanması sonucu elde edilen görüntüler Şekil 2.51'deki gibidir.



Şekil 2.51. zoom_range parametresi uygulanmış görüntü

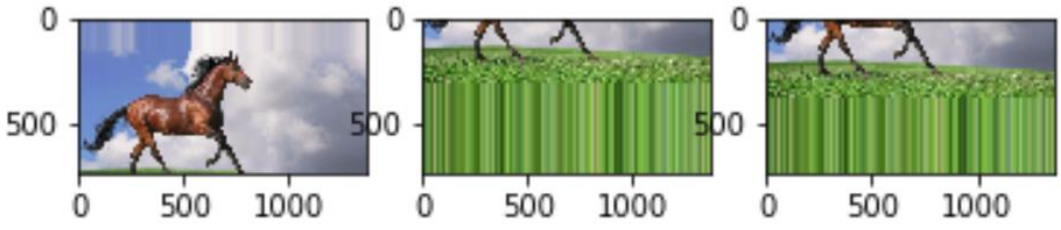
g. **width_shift_range parametresi**: Görüntülerdeki nesnelerin konumları her zaman görüntünün merkezinde olmayabilir. Görüntüler rastgele olarak yatay eksende kaydırılır. Değerler *integer* ve *float* veri tipinde olabilir. Veya width_shift_range=2 değerini aldığında [-1,0,1] değerlerini alabildiği anlamına

gelir. Yani width_shift_range= [-1,0,1] ifadesiyle eşdeğerdir. Aynı şekilde width_shift_range=1 değerini aldığında ise width_shift_range= [-1,1] ifadesiyle eş değer olduğu anlamına gelir. Seçilen görüntüye *width_shift_range* parametresinin uygulanması sonucu elde edilen görüntüler Şekil 2.52'deki gibidir.



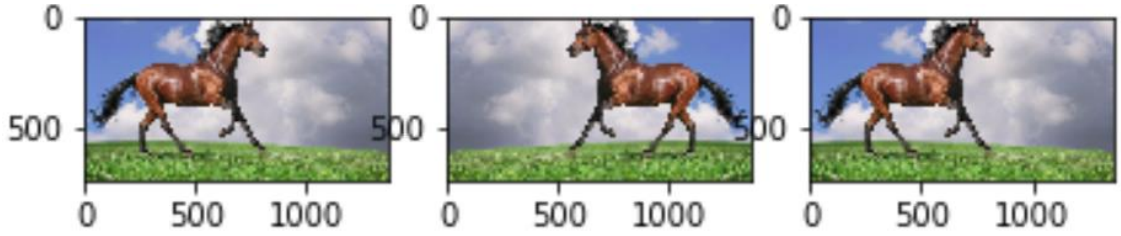
Şekil 2.52. width_shift_range parametresi uygulanmış görüntü

h. **height_shift_range parametresi**: Görüntülerdeki nesnelerin konumları her zaman görüntünün merkezinde olmayabilir. Görüntüler rastgele olarak dikey eksende kaydırılır. Değerler *integer* ve *float* veri tipinde olabilir. Veya *width_shift_range*=2 değerini aldığında [-1,0,1] değerlerini alabildiği anlamına gelir. Yani *width_shift_range*=[-1,0,1] ifadesiyle eşdeğerdir. Aynı şekilde *width_shift_range*=1 değerini aldığında ise *width_shift_range*= [-1,1] ifadesiyle eş değer olduğu anlamına gelir. CIFAR-10 veri setinde seçilen görüntüye *height _shift_range* parametresinin uygulanması sonucu elde edilen görüntüler Şekil 2.53'teki gibidir.



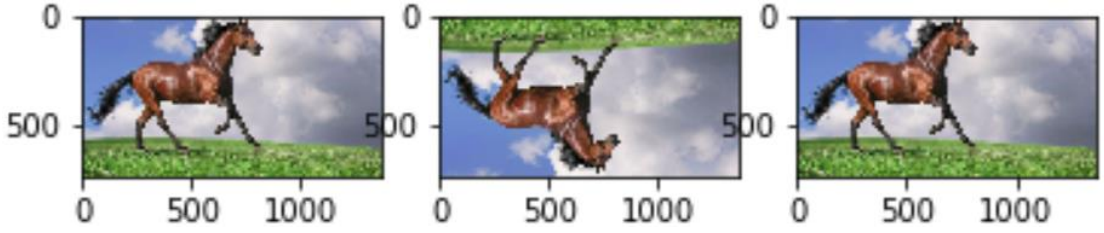
Şekil 2.53. height_shift_range parametresi uygulanmış görüntü

i. **horizontal_flip parametresi**: Giriş verileri yatay olarak rastgele çevrilir. Parametre değerleri *boolean* veri tipindedir. CIFAR-10 veri setinde Şekil 47'deki görüntüye *horizontal_flip parametresinin* uygulanması sonucu elde edilen görüntüler Şekil 2.54'teki gibidir.



Şekil 2.54. horizontal_flip parametresi uygulanmış görüntü

j. **vertical_flip parametresi**: Giriş verileri dikey olarak rastgele çevrilir. Parametre değerleri *boolean* veri tipindedir. Seçilen görüntüye *vertical _flip parametresinin* uygulanması sonucu elde edilen görüntüler Şekil 2.55'teki gibidir.



Şekil 2.55. vertical_flip parametresi uygulanmış görüntü

Veri arttırma işlemi için parametre değerleri girildikten sonra, Şekil 2.56'da görüldüğü gibi tüm değerler eğitime dahil edilmek üzere herhangi bir değişkene (veri arttırma) aktarılır.

```
veri_arttirma= ImageDataGenerator(
                                featurewise_center=False,
                                samplewise_center=False,
                                featurewise_std_normalization=False,
                                samplewise_std_normalization=False,
                                rotation_range=0,
                                zoom_range=False,
                                width_shift_range=0,
                                height_shift_range=0,
                                horizontal_flip=0,
                                vertical_flip=0,
                                data_format=K.image_data_format())

veri_arttirma.fit(img)

i=0
for img_batch in veri_arttirma.flow(img, batch_size=6, shuffle=False):
    for img in img_batch:
        plt.subplot(330 + 1 + i)
        plt.imshow(img)
        i=i+1
    if i >= batch_size:
        break
plt.show()
```

Şekil 2.56. Veri arttırma parametreleri kod bloğu

Batch boyutu

Veri setinin büyüklüğü makine öğrenmesi için önemlidir. Ancak veri setinin aynı anda eğitimi işlemci, bellek ve zaman açısından maliyetlidir. Veri seti küçük gruplara ayrılarak eğitim ve öğrenme işlevi yapılmaktadır. Yani eğitim setinden tek seferde alınan veri sayısıdır. Veri boyutu mevcut bellek ortamında eğitilebilecek kadar büyük, local optimuma takılmayacak kadar küçük olmalıdır. Bununla birlikte küçük batch boyutunun seçilmesi modelin veri setindeki gürültüyü öğrenmesini zorlaştıracaktır. Batch boyutu genellikle GPU belleğine uyacak şekilde 2, 4, 8, 16, 32,...2^n gibi 2'nin kuvvetleri olacak şekilde belirlenebilir. Bu sayılar veri setinin boyutuna ve türüne göre değişkenlik göstermektedir. Radiuk [99], CIFAR-10 veri seti üzerinde 5 tane konvolüsyon katmanı olan yapay sinir ağlarını kullanarak veri boyutunun doğruluk değerine bağımlılığını ölçülmüştür. Çalışmaya göre 16 ile 1024 arasında 12 farklı veri boyutu kullanılarak veri boyutunun doğruluk değeri ile doğru orantılı olduğunu tespit etmiştir.

Epoch sayısı

İleri ve geri yayılım eğitim sayısı epoch sayısını verir. Yani epoch sayısının 1(bir) olması demek 1 tam ileri ve geri yayılım eğitimi gerçekleştirmek anlamına gelir. En yüksek doğruluk oranı elde edecek parametrelerin tespiti için en uygun epoch sayısı belirlenmesi önemlidir. Ağırlık, öğrenme oranı ve diğer parametrelerin güncellenerek hata oranını en aza indiren değerlerin bulunması epoch sayısı arttıkça mümkün olur. Ancak epoch sayısının çok yüksek olması modelin ezberlemesine (ovetfitting) sebep olmaktadır.

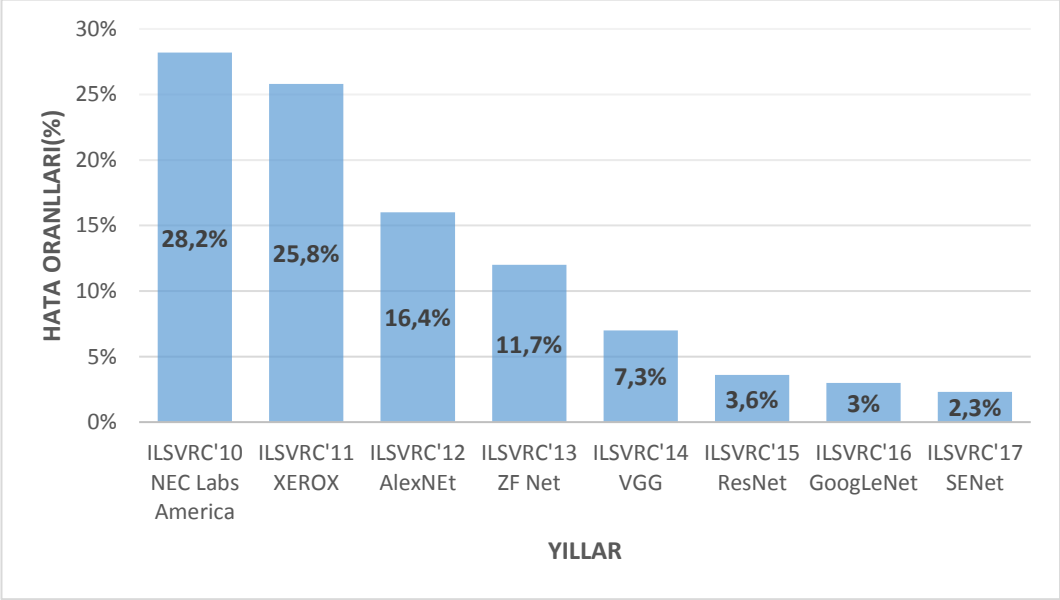
İterasyon

Veri seti batch boyutlarına ayrılmış gruplar halinde eğitilir. Bir epoch'ta eğitilen veri boyutu sayısı iterasyonu verir. Yani 12.800 etiketli veri içeren bir veri seti için veri boyutu 512 olarak belirlendiğinde 1 epoch ile eğitiminde iterasyon sayısı 25'tir.

2.7. Transfer Öğrenme

Herhangi bir problemin çözümü için yeni modeller oluşturulabileceği gibi daha önce denenmiş modeller de kullanılabilir. Bir problem üzerinde eğitilmiş bir modelin başka bir problem üzerinde tasarlanarak yeniden uygulanması işlemine transfer öğrenme denir [100]. Derin öğrenmede doğru sonuçlar veren eğitilmiş bir modelin oluşturulabilmesi için veri miktarının büyük olması önemlidir. Veri miktarının az olduğu durumlarda transfer öğrenme daha iyi sonuçlar verebilir. Her yeni problem için sıfırdan bir model oluşturmak zordur. Derin öğrenmede veri büyüklüğü, öğrenmenin uzun zaman alması ve doğru parametrelerin belirlenmesinin zorluğu transfer öğrenmeyi daha çok tercih edilir hale getirmiştir. Öğrenmenin aktarılması ancak problem türüne uygun olduğunda mümkün olmaktadır. Bir model tüm problem durumları için uygun bir çözüm olmayabilir. Transfer öğrenmede katmanların keyfi olarak kaldırılması veya eklenmesi beklenemez. Ancak belirlenen problemin sınıf sayısına göre son katmanın değiştirilmesi tercih edilen bir durumdur.

Her yıl North Carolina Üniversitesi, Stanford Üniversitesi ve Michigan Üniversitesi ev sahipliğinde makine öğrenmesi ve örüntü tanıma alanlarında çalışma yapmak ve geliştirmek adına nesne tanıma ve görüntü sınıflandırması yarışması olan ImageNet Büyük Ölçekli Görsel Tanıma Yarışması (ILSVRC) [101] yapılmaktadır. Yarışma ImageNet veri tabanı üzerinden yapılmaktadır. Her yıl geliştirilen ve erişilebilir bir görüntü veri tabanı olan ImageNet yaklaşık 22.000 kategoriye ait 1,200 milyondan fazla etiketli yüksek çözünürlüklü görüntüden oluşan veri kümesidir. Her 3 ay önce sunulan veriler ile en yüksek performans gösteren model mimarisi oluşturulması beklenir. Şekil 2.57'da 2010-2017 yılları arası yarışmada elde edilen en düşük hata oranlarının yıllara göre dağılımının grafiği gösterilmektedir. İnsan hatasının %5 olduğu kabul edilirse 2015 yılı ve sonrası makinenin insan hatasından daha düşük oranla hata tespiti yapabildiğini göstermektedir. Şekilde görüldüğü gibi derin sinir ağlarıyla birlikte hata oranında hızlı bir düşüş gözlemlenmiştir. Bu da büyük ölçekli verilerde derin sinir ağlarının diğer yöntemlere oranla daha başarılı olduğunu gösterir.



Şekil 2.57. ImageNet Büyük Ölçekli Görsel Tanıma yarışmasının yıllara göre hata oranlarının dağılımı

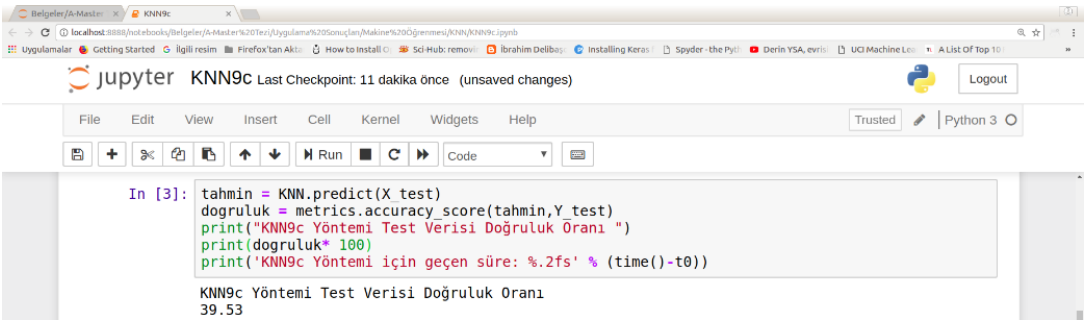
Yarışmada geliştirilen düşük hata oranı elde edilmiş modeller ile farklı veri setleri üzerinde transfer öğrenme işlemi gerçekleştirilebilir. Katmanlar veya parametreler üzerinde ince ayarlamalar yapılarak modelin performansı arttırılabilir. Yarışma dışında geliştirilen bazı derin öğrenme mimarileri ile de farklı veri setlerinde düşük oranda hata oranı elde edilmiştir. Bir model farklı veri setlerinde düşük hata oranı elde ediyorsa bu model, transfer öğrenme için uygun diyebiliriz.

3. UYGULAMA

Makine öğrenmesi ve derin öğrenme yöntemleri ile toplamda 90 faklı model CIFAR-10 ve MNIST veri setleri üzerinde uygulanarak nesne tanıma çalışması yapılmıştır. Yöntemlerin performans karşılaştırmasının yapılması amaçlanmıştır.

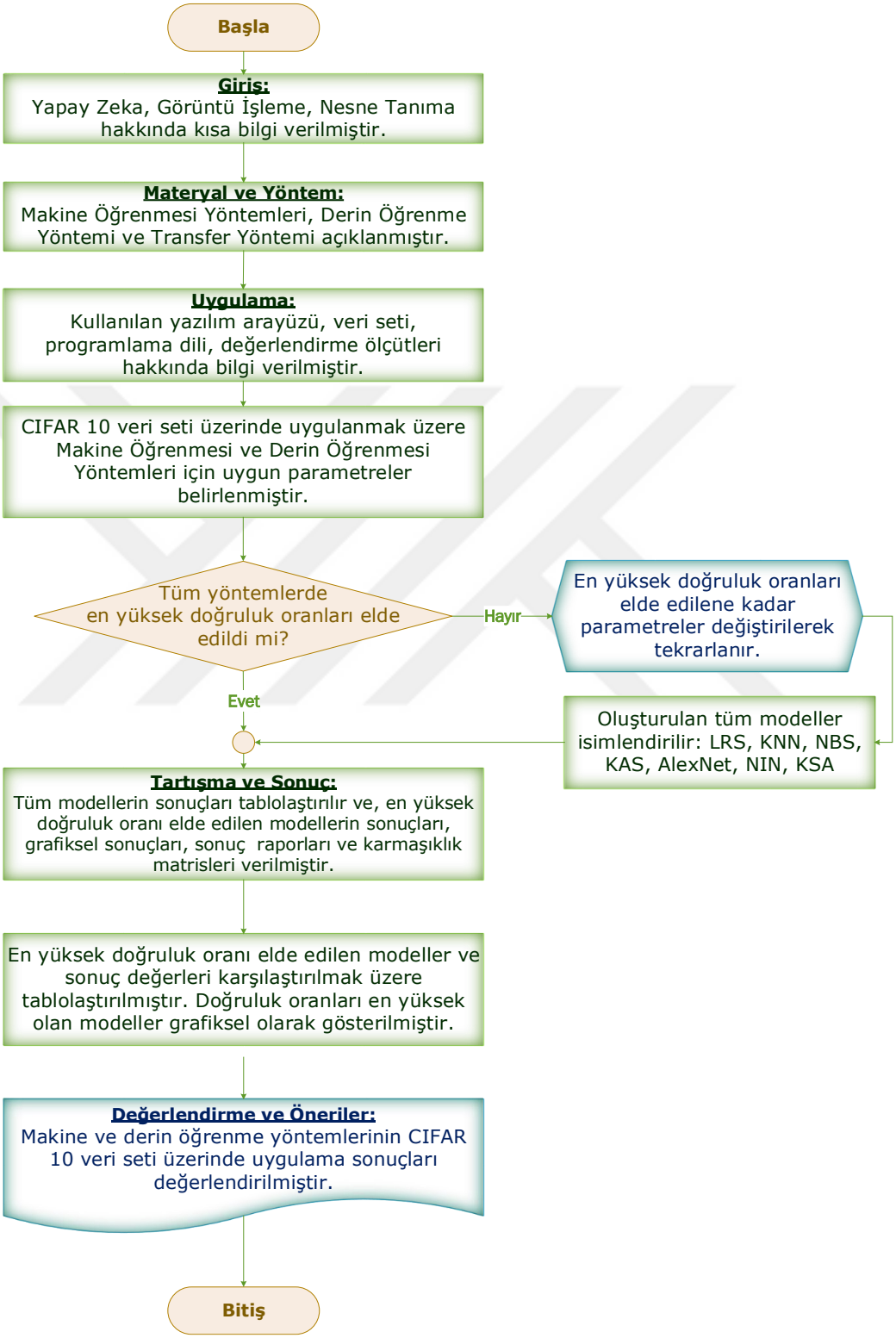
- Lojistik regresyon: Max İterasyon sayısı değiştirilerek en yüksek en yüksek doğruluk oranı elde edilmeye çalışılmıştır. 12 farklı lojistik regresyon eğitim modeli oluşturulmuştur.
- k-En yakın komşu: Komşuluk, uzaklık ve algoritma parametre değiştirilerek en yüksek doğruluk oranı elde edilen model tespit edilmeye çalışılmıştır. Toplamda 21 farklı k-en yakın komşu modeli ile eğitim yapılmıştır.
- Naive bayes: Naive bayes yöntemi ile eğitim yapılmıştır.
- Karar ağacı: Regresyon ve entropiye dayalı karar ağaçları ile rastgele karar ağaçlarında ağaç derinlikleri değiştirilerek en yüksek doğruluk oranı elde edilen karar ağacı modeli tespit edilmeye çalışılmıştır. 54 farklı karar ağacı modeli ile eğitim yapılmıştır.
- Derin öğrenme: İki farklı derin öğrenme mimarisi ile eğitim yapılmış ve en yüksek doğruluk oranı tespit edilmeye çalışılmıştır.

Jupiter notebook arayüzü üzerinde örnek makine öğrenmesi ve derin öğrenme python kodu ekran görüntüsü Şekil 3.1'de verilmiştir.



Şekil 3.1. Jupyter Notebook arayüzü ekran görüntüsü python kodu

Çalışmada izlenen akış diyagramı şekilde verilmiştir.



Şekil 3.2. Tez süreci işlem adımları akış şeması

3.1. Makine Öğrenmesi ve Derin Öğrenme Yöntemlerinde Ortak Olan Özellikler

Çalışmada veri setleri, uygulama yazılımı ve dili, kullanılan bilgisayar donanımı, kütüphaneler ve değerlendirme ölçümleri makine öğrenmesi ve derin öğrenme yöntemleri için ortak olarak kullanılan özelliklerdir.

3.1.1 Veri Seti

Yöntemlerin görüntü verileri üzerindeki sınıflandırma başarısını ölçmek için görüntü veri setlerinde makine öğrenmesi ve derin öğrenme yöntemleri uygulanmıştır. CIFAR-10 ve MNIST veri setleri üzerinde sınıflandırma yapılarak yöntemlerin görüntüler üzerindeki performans tutarlılığının değerlendirilmesi sağlanmıştır.

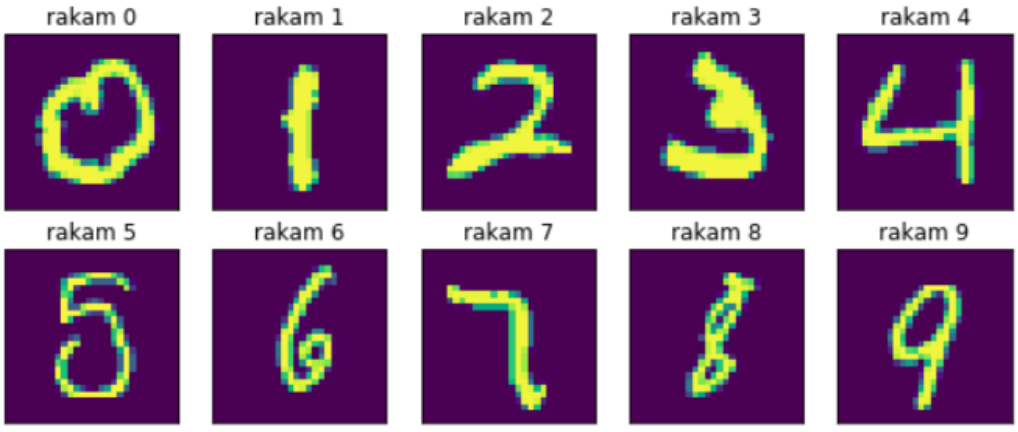
CIFAR-10

CIFAR-10 veri seti [79] Alex Krizhevsky, Vinod Nair ve Geoffrey Hinton tarafından oluşturulmuştur. CIFAR-10 veri seti, sınıf başına 6000 görüntü ile toplam 10 sınıfta 60000 adet 32x32 boyutunda renkli görüntülerden oluşur. Veri seti, eğitim ve test verisine ayrılmıştır. Eğitim verileri her sınıftan 5000 adet görüntü olmak üzere rastgele olarak seçilmiş 50000 adet görüntü içerir. Test verileri ise, her sınıftan 1000 adet olmak üzere rastgele olarak seçilmiş 1000 adet görüntüden oluşur. Şekil 3.3'teki veri kümesinde sınıfların yanı sıra, her biri için 1(bir) adet rastgele seçilmiş görüntü verilmiştir.

Şekil 3.3. CIFAR-10 veri seti örnek görüntüler

MNIST

MNIST veri seti, 0'dan 9'a kadar rakamlardan oluşan 60000 adet 28x28 boyutundaki siyah beyaz görüntülerden oluşur. Veri seti, 50000 adet görüntüden oluşan eğitim ve 10000 adet görüntüden oluşan test verisine ayrılmıştır. Test veri setini oluşturan her bir sınıfın görüntü sayıları birbirinden farklıdır. 980 tane 0 rakamı, 1135 tane 1 rakamı, 1032 tane 2 rakamı, 1010 tane 3 rakamı, 982 tane 4 rakamı, 892 tane 5 rakamı, 958 tane 6 rakamı, 1028 tane 7 rakamı, 974 tane 8 rakamı ve 1009 tane 9 rakamından oluşur. Şekil 3.4'teki veri setinde sınıfların yanı sıra, her biri için 1(bir) adet rastgele seçilmiş görüntü vardır.



Şekil 3.4. MNIST veri seti örnek görüntüler

Çizelge 3.1'de test verilerindeki her bir sınıfa ait görüntü sayıları verilmiştir.

Çizelge 3.1. Test verilerindeki her bir sınıfa ait görüntü sayıları

	CIFAR-10		**MNIST**
Sınıf	**Görüntü Sayısı**	**Sınıf**	**Görüntü Sayısı**
uçak	1000	rakam 0	980
araba	1000	rakam 1	1135
kuş	1000	rakam 2	1032
kedi	1000	rakam 3	1010
geyik	1000	rakam 4	982
köpek	1000	rakam 5	892
kurbağa	1000	rakam 6	958
at	1000	rakam 7	1028
gemi	1000	rakam 8	974
kamyon	1000	rakam 9	1009

3.1.2. Uygulama Programlama Dili

Uygulamada yazılım dili olarak python dili kullanılmıştır. Python dili derin öğrenme yöntemi için uygun yapısal içeriğe sahip olmasından dolayı en çok tercih edilen programlama dilidir. Python programlama dilinin bir diğer avantajı da hem akademik hem ticari uygulamalar için kullanılabilmesidir [102]. Nesne yönelim bir dil olması ve derin öğrenme yöntemi çalışmalarının python dilinde yazılması python tercihini zorunlu hale getirmektedir. Özellikle görüntü işleme uygulamalarında python'un matris çarpımı kolaylığı diğer dillere göre avantajlı bir durumdur.

3.1.3. Kullanılan Program

Ubuntu özgür bir yazılım olması ve yapay zekâ çalışmalarında yüksek kararlılık, ölçeklenebilirlik ve kütüphane desteği sağlaması açısından avantajlı bir yazılımdır. Çalışmada Linux işletim sistemi çekirdeği dağıtımlarından birisi olan Ubuntu üzerinde Anaconda yazılımı kullanılmıştır. Python yaygın olarak kullanılan yüksek seviyeli, genel amaçlı, yorumlanmış ve dinamik bir programlama dilidir. Tasarım felsefesi, kod okunabilirliğini vurgular ve sözdizimi, programcıların C ++ veya Java gibi dillerde mümkün olandan daha az kod satırındaki kavramları ifade etmelerine olanak tanır. Dil hem küçük hem de büyük ölçekte açık programları etkinleştirmeyi amaçlayan yapılar sağlar [103].

Anaconda, bilimsel uygulamalar ve yapay zekâ gibi konulara uygun olarak hazırlanmış python üzerinde çalışmak isteyenler için hazırlanmış bir dağıtımdır. Yapılan çalışmaları kolaylaştıran sıkça kullanılan kütüphaneleri bünyesinde barındırır ve conda isimli paket yöneticisiyle gereli kütüphaneleri kurma olanağı sağlar. Bununla birlikte jupyter notebook ve spyder gibi araçlar da Anaconda ile tümleşik olarak kurulabilmektedir. Jupyter notebook, python ile daha esnek bir çalışma ortamı sağlar. Beraberinde dil ve kütüphane desteği ile kurulur.

3.1.4. Donanım Özellikleri

Derin öğrenme uygulamaları işlem gücü yüksek makinelerde daha iyi performans gösterir. Donanım desteği yüksek bir makine ile daha çok katmana sahip deri öğrenme mimarileri oluşturulabilir. Uygun parametrelerin test edilmesi donanım ile yakından ilişkilidir. Katman sayısı ve epoch sayısı yüksek; veri boyutu düşük olan bir mimaride veri setinin eğitimi günlerce hatta aylarca sürebilir. Bu yüzden bu derin öğrenme ile yapılan çalışmalar makine öğrenmesi ile yapılan çalışmalara oranla donanımsal olarak daha maliyetli denilebilir. Aşağıdaki özelliklere sahip bir bilgisayar ile nesne tanıma çalışması yapılmıştır:

1. İşlemci : Intel Core i7, 2.5 GHz
2. RAM : DDR3L, 16GB
3. Ekran Kartı : Paylaşımsız, 4GB
4. Grafik Kartı : NVIDIA, GTX860M

3.1.5. Kullanılan Kütüphaneler

Çok sık kullanılan kod parçacıkları daha sonra kullanılmak üzere dosyalar halinde saklanır. Birçok sınıfı ve fonksiyonu daha sonra kullanılmak üzere bir arada bulundurur. Python dilinde kullanılan bu dosyalar kütüphaneler olarak adlandırılır. Modelde veri setinin, katmanların, matematiksel işlemlerin, grafiklerin, resimlerin, veri arttırmanın ve model için belirlenen birçok özelliğin eklenmesinde kütüphaneler kullanılır.

Mevcut kütüphanelerin Anaconda yazılımına eklenmeden önce bilgisayara ortamına eklenmesi gerekir. Ubuntu'da kütüphaneler terminal üzerinden eklenir. Terminale eklenen kütüphanelerin kod yazılımı ve örnek gösterimi aşağıdaki gibidir:

Genel gösterim :sudo pip install *kütüphane* veya pip install *kütüphane*

Örnek :sudo pip install *keras*: Ubuntu'ya keras kütüphanesi eklenir.

Ubuntu'ya terminal üzerinden eklenen kütüphaneler Anaconda yazılımına python dilinde *import* kodu ile eklenir. Ve kütüphaneye *as* komutu ile kısaltma (alias) verilebilir. Kısaltma adı ile daha sonra program içerisinde çağrılabilir. Genel kod yazılımı ve örnek kod aşağıdaki gibidir:

Genel gösterim :import *kütüphane* as *kısaltma*

Örnek :from *keras* as *k*; Anaconda yazılımına keras kütühanesi eklenir. Keras yerine k kısaltması kullanılarak daha sonra programda çağrılabilir.

Kütüphanelere ait alt modüller ve fonksiyonlar da Anaconda yazılımına dahil edilebilir. *From* kodundan sonra kütüphane yazılır ve *import* kodu ile alt modül veya fonksiyon eklenir. Genel kod yazılımı ve örnek kod aşağıdaki gibidir:

Genel gösterim :from *kütüphane* import *alt modül veya fonksiyon*

Örnek :import *Conv*, *Dropout*; Anaconda yazılımına Konvolüsyon ve Dropout katmanları eklenir.

Python dilinde açıklama satırları # işareti ile eklenir. Çalışmada kulanılan kütüphaneler ve açıklamaları Şekil 3.5'te jupyter notebook ekran görüntüsü verilmiştir.

```
------- MAKİNE ÖĞRENMESİ - KÜTÜPHANELER --------

---Zaman İşlemleri--- (ortak)
import time
---Dosya İşlemleri---
import pickle
---Hesaplamalar--- (ortak)
import numpy
---Veri Analizi---
import pandas
---Grafik İşlemleri--- (ortak)
from matplotlib import pyplot
---Veri Görselleştirme--- (ortak)
import seaborn
import yellowbrick
---Sklearn: Makine Öğrenmesi İşlemleri---
# Lojistik Regresyon
from sklearn.linear_model import LogisticRegression
# K En Yakın Komşu
from sklearn.neighbors import KNeighborsClassifier
# Naive Bayes
from sklearn.naive_bayes import GaussianNB
# Karar Ağaçları Sınıfı
from sklearn.tree import DecisionTreeClassifier
# Rastgele Karar Ağaçları Sınıfı
from sklearn.ensemble import RandomForestClassifier
# Değerlendirme Tablosu ve Karmaşıklık Matrisi (ortak)
from sklearn import metrics
from sklearn.metrics import classification_report, confusion_matrix


------- DERİN ÖĞRENME - KÜTÜPHANELER -------

---Veri Analizi---
import pandas
---Gradyan İniş Yöntemi Optimizasyonu---
import SGD
---Grafik İşlemleri---
from matplotlib import pyplot
---Dönüşümler---
from scipy import ndimage #görüntü-dizi
from scipy.misc import toimage #görüntü dönüşümleri
---Keras: Yüksek Seviyeli Sinir Ağı Uygulama Arayüzü (API)---
import keras
#cifar10 Veri Seti Yükleme
from keras.datasets import cifar10
#Sinir Ağı ModeliİçinDizi Ekleme
from keras.models import Sequential
#Derin Öğrenme Katmanları
from keras.layers import Conv2D, Activation, MaxPooling2D
from keras.layers import Dropout, Flatten, Dense
# Vektör-Matris
import np_utils
# Ağırlık Hesaplamaları
from keras.constraints import maxnorm
# Gradyon İniş Optimizasyonu
from keras.optimizers import SGD
# Vektör-Matris Dönüşümleri
from keras.utils import np_utils
#Sinir Ağı Model Özeti
from keras_sequential_ascii import sequential_model_to_ascii_printout
#Veri Arttırma
from keras.preprocessing.image import ImageDataGenerator
```

Şekil 3.5. Çalışmada kullanılan kütüphaneler ve açıklamaları

3.1.6. Değerlendirme Ölçümleri

Değerlendirme ölçümleri Çizelge 3.2'de verilen karmaşıklık matrisi üzerinden hesaplanır. Karmaşıklık matrisi modelin doğru ve yanlış tahminlerinin sayısını verir. Karmaşıklık matrisi değerlendirme bakımından aşağıdaki anlama sahiptir:

DP (Doğru-Pozitif) : Bir modelin pozitif olan doğru tahminlerin sayısıdır.

YP (Yanlış-Pozitif) : Bir modelin pozitif olan yanlış tahminlerin sayısını verir.

YN (Yanlış-Negatif) : Bir modelin negatif olan yanlış tahminlerin sayısını verir.

DN (Doğru-Negatif) : Bir modelin doğru olan yanlış tahminlerin sayısını verir.

Karmaşıklık matrisinde yer alan değerlendirme ölçütleri doğruluk, kesinlik, duyarlılık, f1-skor'dur.

Çizelge 3.2. Karmaşıklık matrisi

		GERÇEK	
		Pozitif	Negatif
TAHMİN	Pozitif	DP	YP
	Negatif	YN	DN

Doğruluk : Bir modelin tahminlerinin ne kadarının doğru olduğunu ölçer. Eşitlik 3.1. doğruluk denklemini tanımlar.

$$\text{Doğruluk} = (\text{TP} + \text{TN})/(\text{TP} + \text{FP} + \text{FN} + \text{TN}) \quad (3.1)$$

Kesinlik : Bir modelin doğru olarak tahmin ettiklerinin gerçek doğrular içerisindeki oranını ölçer. Eşitlik 3.2. kesinlik denklemini tanımlar.

$$\text{Kesinlik} = \text{DP}/(\text{DP} + \text{YP}) \quad (3.2)$$

Duyarlılık : Bir modelin doğru olarak tahmin ettiklerinden ne kadarının gerçekten doğru olduğunu ölçer. Eşitlik 3.3. duyarlılık denklemini tanımlar.

$$\text{Duyarlılık} = \text{DP}/(\text{DP} + \text{YN}) \tag{3.3}$$

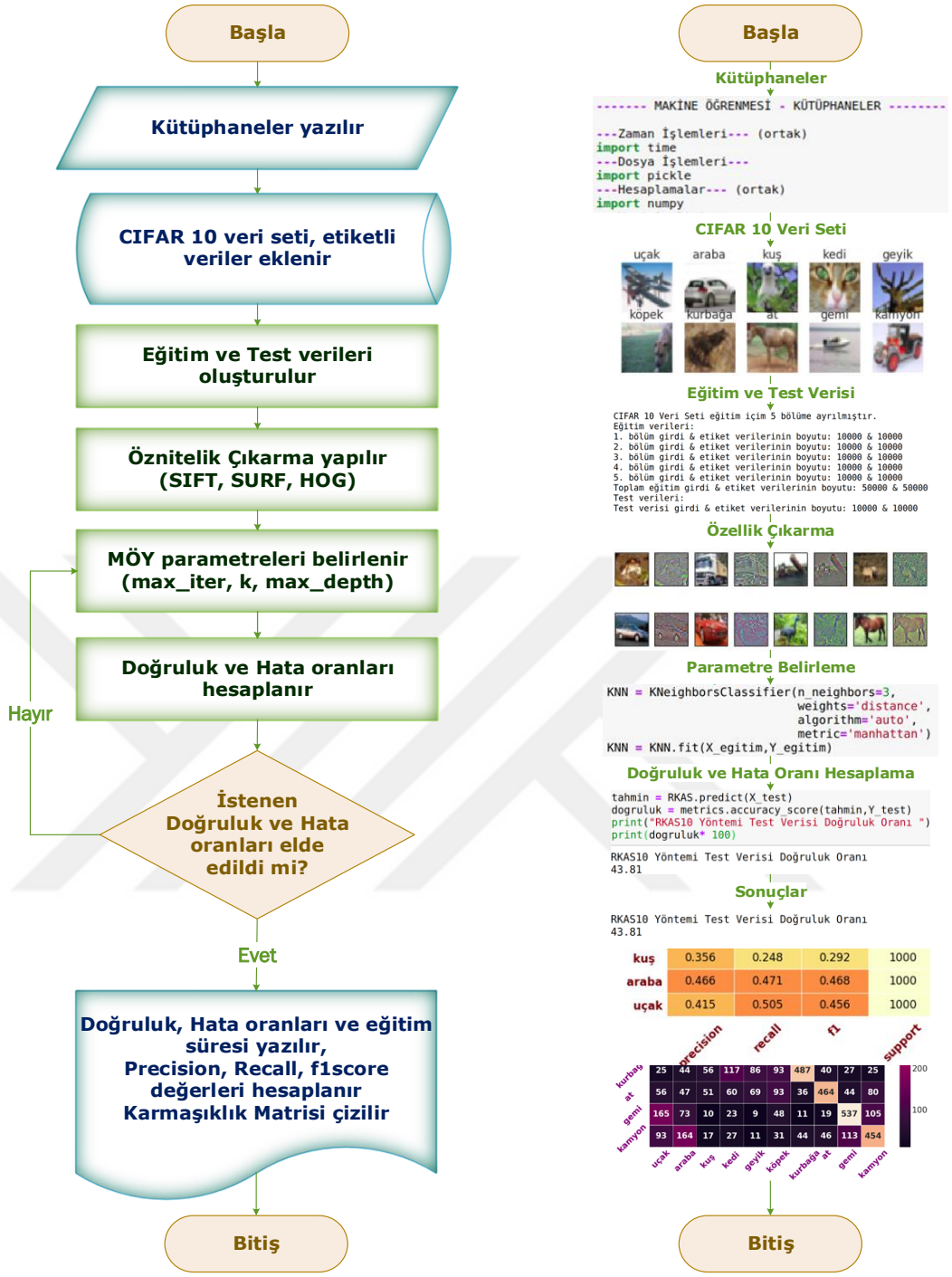
Kesinlik, doğruluk ya da kalite ölçüsü olarak görülebilirken, duyarlılık bütünlük ya da nicelik ölçüsüdür. [104]

F1-skor : Kesinlik ve Duyarlılık değerlerinin harmonik ortalaması f1-skor değerini verir. F1-skor değeri eşitlik 3.4'te gösterilmiştir.

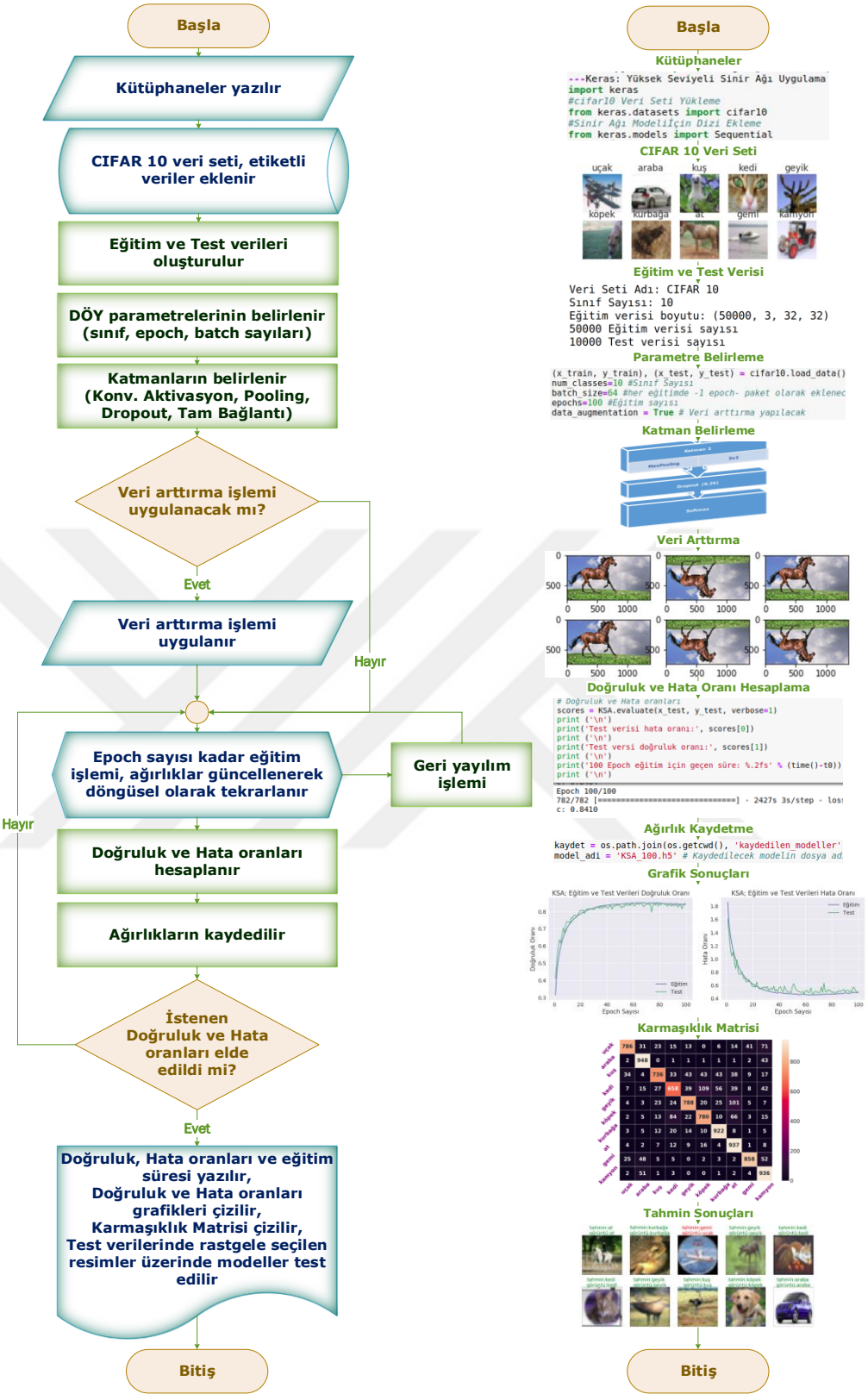
$$\text{f1} - \text{skor} = (2 * \text{Kesinlik} * \text{Duyarlılık})/(\text{Kesinlik} + \text{Duyarlılık}) \tag{3.4}$$

3.2. Akış Şemaları

Makine öğrenmesi ve derin öğrenme yöntemlerinin CIFAR-10 veri seti üzerinde uygulama adımları Şekil 3.6 ve 3.7'de verilmiştir. Ayrıca işlem adımlarının daha iyi anlaşılabilmesi için örnek ekran çıktıları verilerek akış şemaları oluşturulmuştur.



Şekil 3.6. Makine öğrenmesi yöntemleri akış şemaları **(a)** Akış şeması **(b)** CIFAR-10 veri seti ile uygulama süreci



(a) **(b)**

Şekil 3.7. Derin öğrenme yöntemi akış şemaları **(a)** Akış şeması **(b)** CIFAR-10 veri seti ile uygulama süreci

3.3. Makine Öğrenmesi Yöntemi Uygulamaları

CIFAR-10 ve MNIST veri setleri üzerinde lojistik regresyon, k-en yakın komşu, naive bayes ve karar ağaçları modelleri ile nesne tanıma çalışması yapılmıştır. Tüm yöntemlerde, parametreler değiştirilerek en yüksek doğruluk oranı elde edilmeye çalışılmıştır. Jupyter Notebook veri setleri uygulamaya eklenmiştir. Her bir veri setinde 50000'adet görüntü verisi var ken 10000 adet görüntü verisi vardır. Ayrılan her bölümde denetimli öğrenme için gerekli olan girdi ve etiket verileri vardır.

Makine öğrenmesi yöntemleri için scikit-learn kütüphanesinde bulunan sınıf ve fonksiyonlar kullanılmıştır. Kullanılan yöntemlerin fonksiyonları şekilde verilmiştir. *fit* parametresi ile çalıştırılan makine öğrenmesi modellerinin eğitim süreleri *time ()* parametresi hesaplanmıştır. Makine öğrenmesi yöntemlerinden birisi olan lojistik regresyon yöntemi için örnek kod gösterimi aşağıdaki gibidir:

```
from time import time
baslangic = time ( )
from sklearn LogisticRegression
LR = LogisticRegression (max_iter=1)
LR_fit = LR.fit (E_egitim, Y_Egitim)
```

3.3.1. Lojistik Regresyon Yöntemi

Çalışmada, python dilinde Sklearn kütüphanesinde yer alan LogisticRegression() sınıfı kullanılmıştır. Sınıftaki varsayılan parametreler aşağıdaki gibidir [105]:

*LogisticRegression(*penalty='l2', dual=False, tol=0.0001, C=1.0,
fit_intercept=True, intercept_scaling=1, class_weight=None,
random_state=None, solver='warn', max_iter=100, multi_class= 'warn',
verbose=0, warm_start=False, n_jobs=None*)*

Çalışmada kullanılan parametre ve değerler aşağıda verilmiştir:

max_iter: (*varsayılan=100* veya *int*) :Gradyan iniş max iterasyon sayısı

Lojistik regresyon yöntemi ile veri setleri eğitilerek görüntü sınıflandırma yapılmıştır. En yüksek doğruluk oranının elde edilmesi amacı ile modellerin max *iterasyon* sayıları değiştirilmiştir. Çalışmada parametre değerlerinin doğruluk oranına etkisini gösteren modellere yer verilmiştir. Parametre değerleri değiştirilerek elde edilen lojistik regresyon modelleri,

Lojistik Regresyon (LR) + max_ iter

şeklinde isimlendirilmiştir. Max iterasyon sayısı için x değeri aldığında örnek model gösterimi Çizelge 3.3'te verilmiştir.

Çizelge 3.3. Lojistik regresyon yöntemi modelleri ve parametre değerleri

Model	Max İterasyon
LR	Varsayılan
LR**x**	**x**

3.3.2. K-En Yakın Komşu Yöntemi

Çalışmada, python dilinde Sklearn kütüphanesinde yer alan KNeighborsClassifier () sınıfı kullanılmıştır. Sınıftaki varsayılan parametreler aşağıdaki gibidir [106]:

*KNeighborsClassifier(*n_neighbors=5, weights='uniform', algorithm='auto', leaf_size=30, p=2, metric='minkowski', metric_params=None, n_jobs=None*)*

k-En yakın komşu yöntemi ile yapılan çalışmada yüksek performans elde edilmesi için bazı parametre değerlerinde değişiklikler yapılmıştır. Yapılan değişikliklerin doğruluk oranını etkileyip etkilemediği test edilerek en yüksek doğruluk oranı elde edilen model tespit edilmeye çalışılmıştır. k-En yakın komşu yönteminde kullanılan parametreler aşağıda verildiği gibidir:

- **k** :Komşu sayısı, tahmin verisinin kendisine en yakın komşu sayısıdır.
- **weight** :Belirlenen komşuların ağırlık değerleridir:
  - uniform: Tüm komşular eşit ağırlık değerlerine sahiptir.

- distance: Kkomşuların uzaklıklarına göre ağırlık değerleri belirlenir. Belirlenen ağılıklar uzaklıkla ters orantılıdır.
- callable: Komşulara ağırlık değeri bir dizi ile atanır.

- **algorithm** :Tahmin verisine yakın k tane eğitim verisinin yapısı belirlenir:
  - ağaç tipi: Algoritma için ağaç tipi belirlenir:
    - kd tree: k boyutlu sıralı hiyerarşik ikili ağaç yapısını oluşturur.
    - ball tree: kd tree algoritmasının çok boyutlu veriler için geliştirilmiş şeklidir.
  - brute force: Tüm noktalar arası uzaklıklar hesaplanır.
  - auto: Veri tapısına uygun algoritma belirler.
  - leaf_size: Ağaç yapısının dallanma sayısı
- **metric** :Tahmin değerine yakın komşuların (k) tahmin değerine uzaklıklarının hesaplanması
  - p: Minkowski uzaklığının üssel kuvveti

k=10 olarak belirlenen bir k-en yakın komşu modelinde tahmin verisine en yakın 10 komşu Şekil 3.8'de gösterilmiştir. Şeklin solunda tahmin verileri yer alırken sağında en yakın 10 komşu verilmiştir.

Şekil 3.8. k-En yakın komşu yöntemi uygulanan CIFAR-10 veri seti [107]

En yüksek doğruluk oranının elde edilmesi amacı ile *komşu sayısı (n_neigbors), ağırlık belirleme yöntemi(weights), algoritma(algorithm)* ve *komşular arası uzaklık belirleme yöntemi (metric)* parametreleri değiştirilmiştir. k-En yakın komşu yöntemi yöntemi ile eğitim yapılmış ve çalışmada parametre değerlerinin doğruluk oranına etkisini gösteren modellere yer verilmiştir. Parametre değerleri değiştirilerek elde edilen k-en yakın komşu yöntemi modelleri,

KNN + komşu sayısı+ parametre değişiklikleri

şeklinde isimlendirilmiştir. Modeller için komşu sayısı **x** olarak ifade edildiğinde oluşturulan modeller ve parametre değerleri Çizelge 3.4'teki gibidir.

Çizelge 3.4. k-En yakın komşu yöntemi modelleri ve parametre değerleri

Sınıflandırıcı	k	Weight	Algorithm	Komşu Uzaklık
KNN			Varsayılan	
KNN**x**a	**x**	uniform	auto	manhattan
KNN**x**b	**x**	distance	auto	manhattan
KNN**x**c	**x**	distance	ball_tree	manhattan
KNN**x**d	**x**	distance	kd_tree	manhattan
KNN**x**e	**x**	distance	auto	eulidean
KNN**x**f	**x**	distance	auto	minkowski

3.3.4. Naive Bayes Yöntemi

Çalışmada, python dilinde Sklearn kütüphanesinde yer alan GaussianNB () sınıfı kullanılmıştır. Sınıftaki varsayılan parametreler aşağıdaki gibidir [108]:

*GaussianNB(*priors=None, var_smoothing=1e-09*)*

3.3.4. Karar Ağacı Yöntemi

Çalışmada, python dilinde Sklearn kütüphanesinde yer alan DecisionTreeRegressor () ve RandomForestRegressor() sınıfları kullanılmıştır. Sınıflardaki varsayılan parametreler aşağıdaki gibidir [109], [110]:

- *DecisionTreeRegressor(*criterion='mse', splitter='best', max_depth=None, min_samples_split=2, min_samples_leaf=1, min_weight_fraction_leaf=0.0, max_features=None, random_state=None, max_leaf_nodes=None, min_impurity_decrease=0.0, min_impurity_split=None, presort=False*)*
- *RandomForestRegressor(*n_estimators='warn', criterion='mse', max_depth= None, min_samples_split=2, min_samples_leaf=1, min_weight_fraction_leaf =0.0, max_features='auto', max_leaf_nodes=None, min_impurity_decrease =0.0, min_impurity_split=None, bootstrap=True, oob_score=False, n_jobs =None, random_state=None, verbose=0, warm_start=False*)*

Çalışmada kullanılan parametre ve değerler aşağıda verilmiştir:

criterion: *(varsayılan=gini veya entropy)* :Kök düğümü belirleyecek kural belirlenir.

max_depth: *(varsayılan=yok veya int)* :Ağacın maximum derinliği, tüm yapraklar saf olana kadar düğümler genişletilir.

n_estimators: *(varsayılan=yok veya int)* : Rastgele ormandaki ağaç sayısı

CIFAR-10 ve MNIST veri setleri regresyon, entropi ve rastgele orman yöntemleri ile eğitilmiştir. En yüksek doğruluk oranının elde edilmesi amacı ile *ağaç derinlikleri (max_dept)* değiştirilmiştir. Rastgele ormanda ağaç sayısı da belirlenmiştir. Karar ağacı yöntemi ile eğitim yapılmış ve çalışmada parametre değerlerinin doğruluk oranına etkisini gösteren modellere yer verilmiştir. Parametre değerleri değiştirilerek elde edilen Karar Ağacı modelleri,

Regresyon Karar Ağacı (KASR) + ağaç derinliği

Entropi Karar Ağacı (KAE) + ağaç derinliği

Rastgele Orman (RO) + ağaç derinliği isimlendirilmiştir.

Model isimleri ve parametre değerleri Çizelge 3.5'te verilmiştir. Ağaç sayısının 1000 olarak belirlendiği rastgele orman yönteminde ağaç derinliği Çizelge 3.5.b'de verilmiştir.

Çizelge 3.5. Karar ağacı yöntemi modelleri ve parametre değerleri **a.** Sınıflandırma ve regresyon, entropiye dayalı **b.** Rastgele orman

(a)

Model	Kriter	Ağaç Derinliği
KA		Varsayılan
KASRx	gini	x
KAEx	entropy	5

(b)

Model	Ağaç Sayısı	Ağaç Derinliği
RO	1000	Varsayılan
ROx	1000	x

3.4. Derin Öğrenme Yöntemi Uygulamaları

3.4.1. Derin Öğrenme Mimarilerinde Kullanılan Ortak Özellikler:

- ✓ **Veri setini değişkenlere atama:** 32x32x3 boyutunda 10 sınıfa ait 60000 veriden 50000'i eğitim 10000'i test verisi olarak ayrılmıştır. CIFAR-10 ve MNIST veri setleri keras kütüphanesi ile jupyter notebook yazılımına yüklendikten sonra, *load_data ()* parametresi ile veri setindeki veriler aşağıda ifade edilen değişkenlere atanır:

  (x_egitim, y_egitim), (x_test, y_test)=CIFAR-10.load_data()

  x_egitim :eğitim verilerinin girdileri x_train değişkenine,
  y_egitim :eğitim verilerinin etiketleri y_train değişkenine,
  x_test :test verilerinin girdileri x_test değişkenine,
  y_test :test verilerinin etiketleri y_test değişkenine aktarılır.
- ✓ **Sınıf atama:** CIFAR-10 ve MNIST veri setlerindeki görüntüler 10 adet sınıfa ayrılmıştır. *sinif_sayisi* isimli değişkenine 10 değeri atanmıştır. Kod satırı aşağıda verildiği gibidir:

  sinif_sayisi=10
- ✓ **Epoch sayısı:** Modeller farklı epoch sayıları ile test edilmiştir. En yüksek yüzeyde performans alınan epoch sayısı belirlenmiştir. GPU kullanımı durumunda epoch sayıları arttırılabilir. Tüm modellerde epoch sayısı 100 olarak belirlenmiştir. Kod satırı aşağıda verildiği gibidir:

  epochs=100
- ✓ **Batch Sayısı:** Modeller için farklı batch sayıları test edilmiş en yüksek doğruluk oranının elde edildiği batch değeri belirlenmiştir.Batch sayısının 64 olarak belirlendiği modellerde iterasyon sayısı (50000/64) 782'dir.

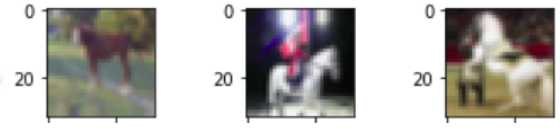
  batch size=64

- ✓ **Veri Arttırma:** Tüm modellerde veri arttırma yapılmıştır. Veri arttırma işlemine onay vermek için *data_augmentation* komutuna true parateresi atanırken, onay verilmediğinde ise false parametresi atanır. Kod satırı aşağıda verildiği gibidir:

  data_augmentation=True

Derin öğrenme modelleri için yapılan veri arttırma işlemi sonrası oluşan görüntü örnekleri Şekil 3.9'daki gibidir.

```
datagen = ImageDataGenerator(rotation_range=0,
                             width_shift_range=0.1,
                             height_shift_range=0.1,
                             horizontal_flip=True)
datagen.fit(x_train)
for X_batch, y_batch in datagen.flow(x_train, y_train, batch_size=9):
    for i in range(0, 9):
        pyplot.subplot(330 + 1 + i)
        pyplot.imshow(X_batch[i].reshape(img_rows, img_cols, 3))
    pyplot.show()
    break
```



Şekil 3.9. AlexNet ve BasitNet modelleri veri arttırma kodu ve ekran çıktısı

- ✓ **Değerlendirme:** Değerlendirme derin öğrenme modelleri için 3 farklı şekilde yapılmıştır :
  1. Eğitim ve test verileri üzerinden elde edilen doğruluk ve hata oranları grafiksel olarak gösterilmiştir.
  2. Karmaşıklık matrisi oluşturulmuştur.
  3. Sınıflandırma raporu oluşturulmuştur.
- ✓ **Ağırlıkların Kaydedilmesi:** Ağırlıklar, başlangıçta rastgele olarak belirlenmiş eğitim süresince geri yayılım yöntemi ile güncellemiştir. Şekil 3.10'da verildiği gibi daha sonra kullanılabilmesi için kaydedilmiştir. Örnek olarak BasitNet modelinin ağırlıkları kaydedildiği kod bloğu verilmiştir.

```
#Modelin kaydedileceği yer belirlenir.
kaydet = os.path.join(os.getcwd(), 'kaydedilen_modeller')
#Kaydedilecek modelin dosya adı belirlenir.
model_adi = 'AlexNet_100.h5'
```

Şekil 3.10. Ağırlıkları kaydetme

Çalışmada iki farklı derin öğrenme mimarisi kullanılmıştır.

3.4.2. AlexNet Mimarisi

1998 yılında Krizhevsky ve ark. [111] ilk kez çok katmanlı derin sinir ağları hakkında makale yazmıştır. Ancak derin öğrenmenin benimsenmesinin hızlanması ve dünya çapında duyulması 2012 yılında yapılan ILSVRC'12 yarışması ile mümkün olmuştur. Bu çalışma daha sonra yapılacak çalışmalar için devrim niteliğindedir. AlexNet modeli 227x227x3 boyutundaki görüntüler için geliştirilmiştir. CIFAR-10 veri setindeki görüntü boyutları 32x32x3; MNIST veri setindeki görüntü boyutları 28x28x3 boyutunda olduğundan, AlexNet modeli veri setlerine uygun olarak iyileştirilmiştir.

Çizelge 3.6'da CIFAR-10 ve MNIST veri setleri üzerinde uygulanan AlexNet model özeti verilmiştir.

Çizelge 3.6. Veri setlerine uyarlanan AlexNet model özeti **(a)** CIFAR-10 **(b)** MNIST

(a)

katman	filtre/değer sayı	filtre/değer boyut	adım	padding	çıktı	parametre
Konv1, ReLU	96	2x2	1	5	32x32x96	1248
Konv2, ReLU	256	2x2	1	2	32x32x256	98560
MaxPooling1	-	2x2	2	-	16x16x256	0
Dropout1	0,5	-	-	-	-	0
Konv3, ReLU	384	2x2	1	1	16x16x384	393600
Konv4, ReLU	384	2x2	1	1	16x16x384	590208
Dropout2	0,5	-	-	-	-	0
Konv5, ReLU	2x2	256	1	1	16x16x256	39472
MaxPooling2	2x2	-	2	-	8x8x256	0
Tam Bağlantılı	-	-	-	-	16384	0
Dense1, ReLU	-	4096	-	-	4096	67112960
Dense2, ReLU	-	4096			4096	16781312
Dropout3	0,5	-	-	-	-	0
Dense3, Softmax	-	10	-	-	10	40970
Toplam						**85,4M**

(b)

katman	filtre/değer sayı	filtre/değer boyut	adım	padding	çıktı	parametre
Konv1, ReLU	96	2x2	1	5	28x28x96	480
Konv2, ReLU	256	2x2	1	2	28x28x256	98560
MaxPooling1	-	2x2	2	-	14x14x256	0
Dropout1	0,5	-	-	-	-	0
Konv3, ReLU	384	2x2	1	1	14x14x384	393600
Konv4, ReLU	384	2x2	1	1	14x14x384	590208
Dropout2	0,5	-	-	-	-	0
Konv5, ReLU	2x2	256	1	1	14x14x256	393472
MaxPooling2	2x2	-	2	-	7x7x256	0
Tam Bağlantılı	-	-	-	-	12544	0
Dense1, ReLU	-	4096	-	-	4096	51384320
Dense2, ReLU	-	4096	-	-	4096	16781312
Dropout3	0,5	-	-	-	-	0
Dense3, Softmax	-	10	-	-	10	40970
Toplam						**69,6M**

Çizelge 3.6'da verilen AlexNet modeli 5 tane konvolüsyon, 2 tane maxpooling, 3 tane dropout ve tam bağlantılı katmandan oluşmaktadır. CIFAR-10 ve MNIST veri setleri için uyarlanan AlexNet modeli katman değerleri aynıdır. Ancak veri setlerindeki görüntü boyutları farklı olduğundan kullanılan parametreler ve son katmandan sonra elde edilen görüntü boyutları farklıdır.

Kullanılan toplam parametre sayısı CIFAR-10 veri seti için 85,4M ve MNIST veri seti için 69,6M'dir. Son konvolüsyon katmanından sonra elde edilen görüntü boyutları CIFAR-10 veri seti için 8x8x256 ve MNIST veri seti için 7x7x256'dır. CIFAR-10 veri seti 10 farklı sınıftan oluştuğundan son dense katmanı için düğüm sayısı 10 olarak belirlenmiştir.

3.4.3. BasitNet Mimarisi

Çizelge 3.7'de CIFAR-10 ve MNIST veri setleri üzerinde uygulanan BasitNet model özeti verilmiştir.

Çizelge 3.7. Veri setlerine uyarlanan BasitNet model özeti **(a)** CIFAR-10 **(b)** MNIST

(a)

katman	filtre/değer sayı	filtre/değer boyut	adım	padding	çıktı	parametre
Konv1, ReLU	128	3x3	1	aynı	32x32x96	3584
Konv2, ReLU	256	2x2	1	-	30x30x256	295168
MaxPooling1	-	2x2	1	-	15x15x256	0
Dropout1	0,25	-	-	-	-	0
Konv3, ReLU	96	3x3	1	aynı	15x15x96	221280
Konv4, ReLU	128	3x3	1	-	13x13x128	110720
Dropout2	0,25	-	-	-	-	0
MaxPooling2	2x2	-	1	-	6x6x256	0
Tam Bağlantılı	-	-	-	-	4608	0
Dense1, ReLU	-	128	-	-	128	589952
Dense2, ReLU	-	4096	-	-	4096	528384
Dropout3	0,5	-	-	-	-	0
Dense3, Softmax	-	10	-	-	10	40970
Toplam						**1,79M**

(b)

katman	filtre/değer sayı	filtre/değer boyut	adım	padding	çıktı	parametre
Konv1, ReLU	128	3x3	1	aynı	28x28x96	1280
Konv2, ReLU	256	2x2	1	-	26x26x256	295168
MaxPooling1	-	2x2	1	-	13x13x256	0
Dropout1	0,25	-	-	-	-	0
Konv3, ReLU	96	3x3	1	aynı	13x13x96	221280
Konv4, ReLU	128	3x3	1	-	11x11x128	110720
Dropout2	0,25	-	-	-	-	0
MaxPooling2	2x2	-	1	-	5x5x256	0
Tam Bağlantılı	-	-	-	-	3200	0
Dense1, ReLU	-	128	-	-	128	409728
Dense2, ReLU	-	4096	-	-	4096	528384
Dropout3	0,5	-	-	-	-	0
Dense3, Softmax	-	10	-	-	10	40970
Toplam						**1,60M**

Çizelge 3.8. Derin öğrenme modelleri katman özellikleri

Model	Öğrenme Oranı	Konv.	Pooling	Dropout	Tam Bağlantılı	Parametre
AlexNet	0.5e-3	5	2	3	2	25.455.818
BasitNet	0.0001	4	2	3	2	1.790.058

4. BULGULAR

Çalışmada, CIFAR-10 ve MNIST veri setleri üzerinde makine öğrenmesi ve derin öğrenme yöntemleri ile sınıflandırma tahmini yapılmıştır. Sonuçlar grafikler, sınıflandırma raporları ve karmaşıklık matrisleri üzerinden değerlendirilmiştir.

Modellere ait sınıflandırma raporunun ekran görüntüleri verilmiştir. Şeklin solunda sonuçlar yer alırken sağında ise sayıların ayırt edilebilirliğini gösteren renk baremi verilmiştir. Baremde 0 (sıfır) ile 1 (bir) arasındaki değerler için renk geçişi verilmiştir. Rengin koyu olması doğruluğun arttığını gösterirken, açık olması azaldığını gösterir. Sınıflandırma raporunda sınıflara göre kesinlik, duyarlılık ve f1-skor değerleri gösterilmektedir. F1-skor değeri, kesinlik ve duyarlılık değerlerinin harmonik ortalaması olduğundan tahmin edilen sınıflar için gerçeğe yakın bilgi verir. Dolayısı ile en iyi ve en kötü tahmin edilen sınıfların ve doğruluk oranının tespiti f1-skor değerleri ile yapılabilir.

Modellere ait karmaşıklık matrisinin ekran görüntüleri de verilmiştir. Sınıfların tahmin sonuçları farklı renlerle gösterilmiştir. Şeklin solunda sonuçlar yer alırken sağında ise sayıların ayırt edilebilirliğini gösteren renk baremi verilmiştir. 0 ile 1000 arasındaki sayısal değerler renk bareminde koyudan açığa doğru verilmiştir. Rengin açık olması doğruluğun arttığını gösterirken, koyu olması azaldığını gösterir. Makine öğrenmesi ve derin öğrenme eğitim sonuçları verilmiştir.

4.1. Makine Öğrenmesi Yöntemleri Eğitim Sonuçları

Çalışmada, lojistik regresyon, k-en yakın komşu, naive bayes ve rastgele orman yöntemlerinden elde edilen modellerin sonuçları verilmiştir. En yüksek doğruluk elde edilen modellerin sonuçları ise detaylı değerlendirme yapılması amacıyla sınıflandırma raporları ve karmaşıklık matrisleri ekran görüntüleri olarak verilmiştir.

4.1.1. Lojistik Regresyon Yöntemi Eğitim Sonuçları

Çalışmada CIFAR-10 ve MNIST veri setleri üzerinde parametreler değiştirilerek lojistik regresyon yöntemi uygulanmıştır. Modellerin eğitim süreleri ve doğruluk oranları Çizelge 4.1'de verilmiştir.

Çizelge 4.1. Lojistik regresyon yöntemi sınıflandırma sonuçları

Model	Eğitim Süresi		Doğruluk Oranı (%)	
	CIFAR-10	MNIST	CIFAR-10	MNIST
LR	5 saat 40 dakika	42 dakika	35,25	91,70
LR2	24 sn	3 saniye	23,02	78,09
LR5	3 dakika	10 saniye	35,76	90,32
LR10	4 dakika	21 saniye	40,11	91,77
LR11	4 dakika	26 saniye	40,15	91,78
*LR12	5 dakika	32 saniye	40,48	91,91
LR13	7 dakika	45 saniye	39,82	91,86
LR14	5 dakika	55 saniye	40,15	91,68
LR15	7 dakika	67 saniye	39,73	91,72
LR30	32 dakika	8 dakika	37,67	91,84
LR50	1 saat 38 dakika	20 dakika	36,22	91,78
LR60	2 saat 8 dakika	23 dakika	35,66	91,75

(*En yüksek doğruluk oranı elde edile model)

Her iki veri setinde de max iterasyon sayısı 12 olduğunda en iyi sınıflandırma tahmini elde edilmiştir.

a. Sınıflandırma Raporu

LR12 modelinin CIFAR-10 ve MNIST veri setleri üzerindeki sınıflandırma raporları Şekil 4.1'de verilmiştir.

(a)

(b)

Şekil 4.1. Lojistik regresyon yöntemi sınıflandırma raporları **(a)** CIFAR-10 **(b)** MNIST

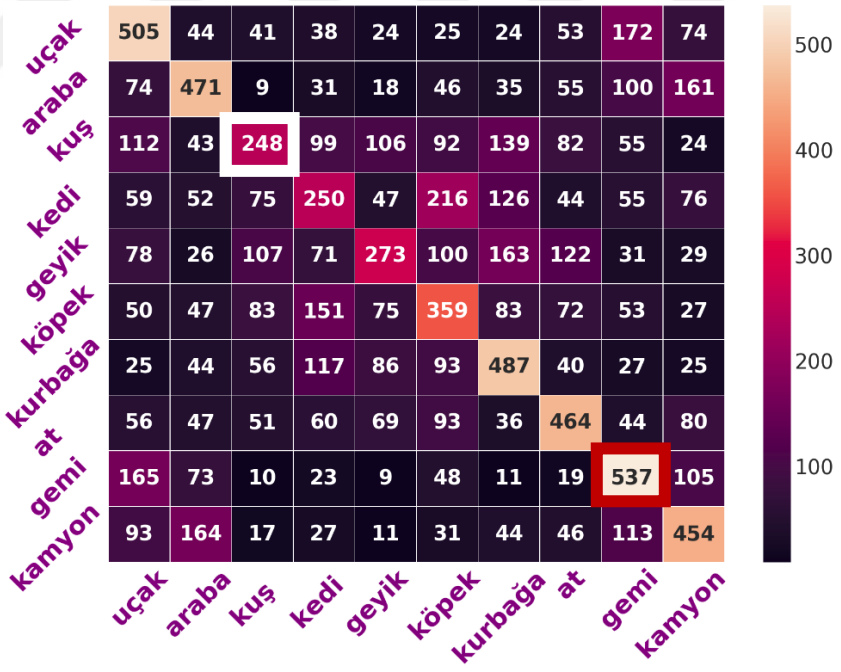
Şekil 4.1'de verilen tahmin sonuçlarına göre yapılan değerlendirmeler Çizelge 4.2'de verildiği gibidir.

Çizelge 4.2. Lojistik regresyon yöntemi sınıflandırma raporlarına göre yapılan değerlendirmeler

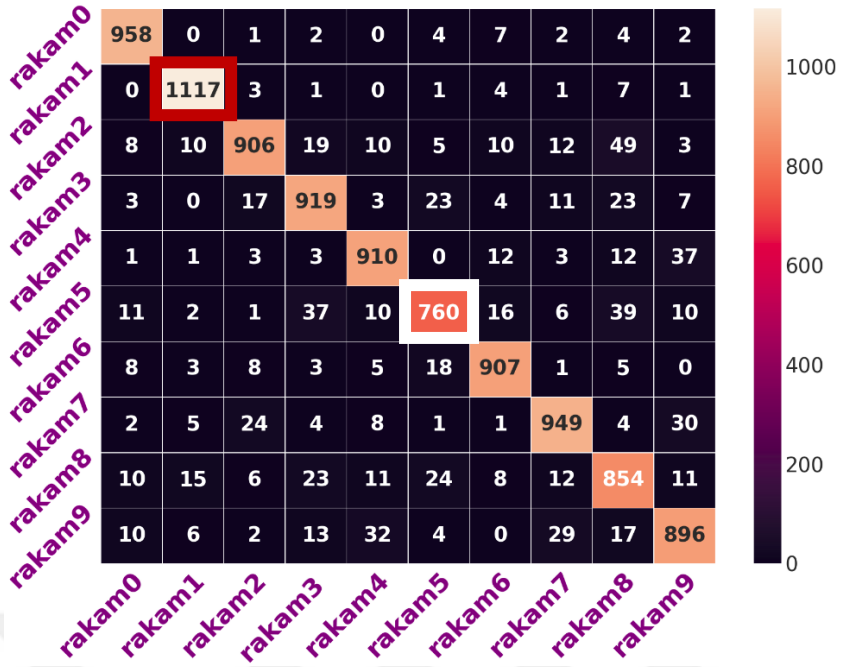
	CIFAR-10	MNIST
f1-skor	En iyi: gemi sınıfı En kötü: kedi sınıfı	En iyi: rakam 1 sınıfı En kötü: rakam 8 sınıfı
kesinlik	Gemi sınıfı olarak tahmin edilen görüntülerin tüm gemi görüntülerine oranı %45,2'dir.	Rakam 1 sınıfı olarak tahmin edilen görüntülerin tüm 1 rakamı görüntülerine oranı %96,4'tür.
duyarlılık	Gemi olarak tahmin edilen görüntülerin %53,7'si doğrudur.	1 rakamı olarak tahmin edilen görüntülerin %98,4'ü doğrudur.

b. Karmaşıklık Matrisi

LR12 modelinin CIFAR-10 ve MNIST veri setleri üzerindeki karmaşıklık matrisleri Şekil 4.2'de verilmiştir.



(a)



(b)

Şekil 4.2. Lojistik regresyon yöntemi karmaşıklık matrisleri **(a)** CIFAR-10 **(b)** MNIST

Şekil 4.2'de verilen tahmin sonuçlarına göre yapılan değerlendirmeler Çizelge 4.3'te verilmiştir.

Çizelge 4.3. Lojistik regresyon yöntemi karmaşıklık matrislerine göre yapılan değerlendirmeler

CIFAR-10	**MNIST**
• En iyi tahmin edilen sınıf 537 tane doğru sınıflandırma ile gemidir.	• En iyi tahmin edilen sınıf 1117 tane doğru sınıflandırma 1 rakamıdır.
• Gemi görüntüleri 165 tane ile en çok uçağa; 9 tane ile en az geyiğe benzetilmiştir.	• 1 rakamı görüntüleri 7 tane ile en çok 8 rakamına benzetilirken; 0 ve 4 rakamı olarak hiç sınıflandırılmamıştır.
• En kötü tahmin edilen sınıf 248 tane doğru sınıflandırma ile kuştur.	• En kötü tahmin edilen sınıf 760 tane doğru sınıflandırma ile 5 rakamıdır.

4.1.2. K-En Yakın Komşu Yöntemi Eğitim Sonuçları

Çalışmada CIFAR-10 ve MNIST veri setleri üzerinde parametreler değiştirilerek k-en yakın komşu yöntemi uygulanmıştır. Modellerin eğitim süreleri ve doğruluk oranları Çizelge 4.4'te verilmiştir.

Çizelge 4.4. k-En yakın komşu yöntemi sınıflandırma sonuçları **(a)** CIFAR-10 **(b)** MNIST

(a)

Model Adı	Eğitim Süresi	Doğruluk Oranı (%)
KNN	36 dakika	33,98
KNN3a	41 dakika	36,25
KNN3b	35 dakika	39,39
KNN3c	25 dakika	39,39
KNN3d	35 dakika	39,39
KNN3e	35 dakika	25,69
KNN3f	39 dakika	39,5
KNN5a	37 dakika	37,7
KNN5b	39 dakika	39,5
KNN5c	29 dakika	39,5
KNN5d	38 dakika	39,5
KNN5e	35 dakika	35,69
KNN7c	29 dakika	39,46
KNN8c	28 dakika	39,32
*KNN9c	29 dakika	39,53
KNN10c	27 dakika	39,26
KNN11c	30 dakika	39,19
KNN12c	40 dakika	33,98
KNN20c	29 dakika	38,51
KNN35c	26 dakika	38,51
KNN50c	27 dakika	36,16

(b)

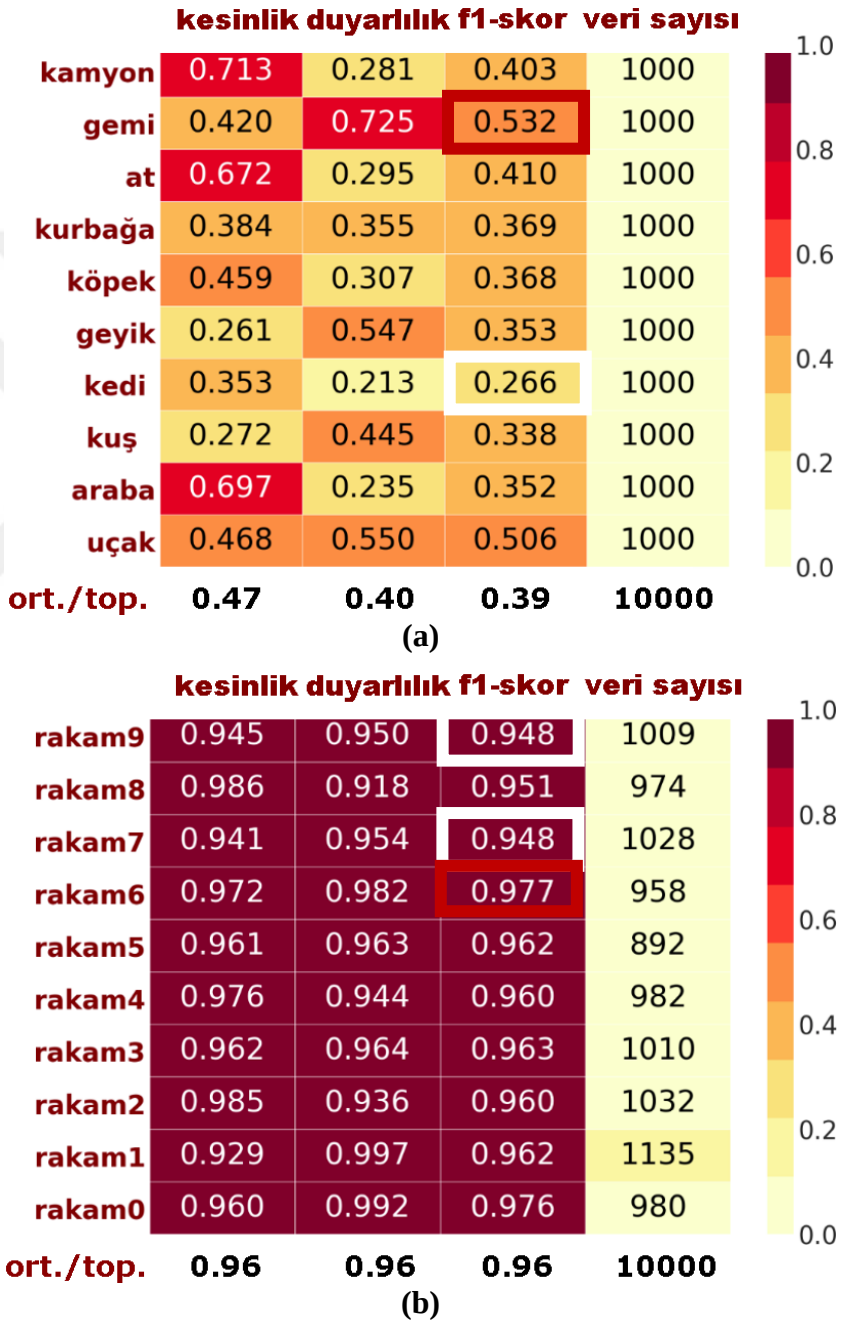
Model Adı	Süre	Doğruluk Oranı (%)
KNN	11 dakika	95,88
KNN3a	13 dakika	95,33
KNN3b	11 dakika	95,4
KNN3c	9 dakika	95,4
KNN3d	11 dakika	95,4
KNN3e	12 dakika	96,17
KNN3f	3 saat 22 dakika	96,28
KNN5a	10 dakika	95,18
KNN5b	10 dakika	95,29
KNN5c	8 dakika	96,29
KNN5d	10 dakika	96,29
KNN5e	20 dakika	95,91
*KNN5f	3 saat 44 dakika	96,35
KNN7f	4 saat 11 dakika	96,22
KNN9f	5 saat	96,19
KNN10f	5 saat	96,14
KNN11f	5 saat 22 dakika	96,07
KNN12f	5 saat 39 dakika	96,01
KNN20f	6 saat	95,54
KNN35f	8 saat 24 dakika	95,06
KNN50f	10 saat 47 dakika	94,69

(*En yüksek doğruluk oranı elde edile model)

Çizelge 4.4'te görüldüğü üzere parametre değerleri arttırıldıkça eğitim süreleri de artmaktadır. Ancak bir noktadan sonra doğruluk oranının azaldığı görülmektedir. CIFAR-10 veri seti için komşu sayısı 9 ve uzaklık ölçüsü manhattan (KNN9c) olduğunda en yüksek doğruluk oranı elde edilirken MNIST veri setinde komşu sayısı 5 ve uzaklık ölçüsü minkowski (KNN5f) olduğunda en yüksek doğruluk oranı elde edilmiştir.

a. Sınıflandırma Raporu

KNN9c modelinin CIFAR-10 veri seti ve KNN5f modelinin MNIST veri seti üzerindeki sınıflandırma raporları Şekil 4.3'te verilmiştir. Sonuçlara göre yapılan değerlendirmeler Çizelge 4.5'te verildiği gibidir.



Şekil 4.3. k-En yakın komşu yöntemi sınıflandırma raporları **(a)** CIFAR-10 **(b)** MNIST

Şekil 4.3’te verilen sonuçlara göre yapılan değerlendirmeler Çizelge 4.5’te verildiği gibidir.

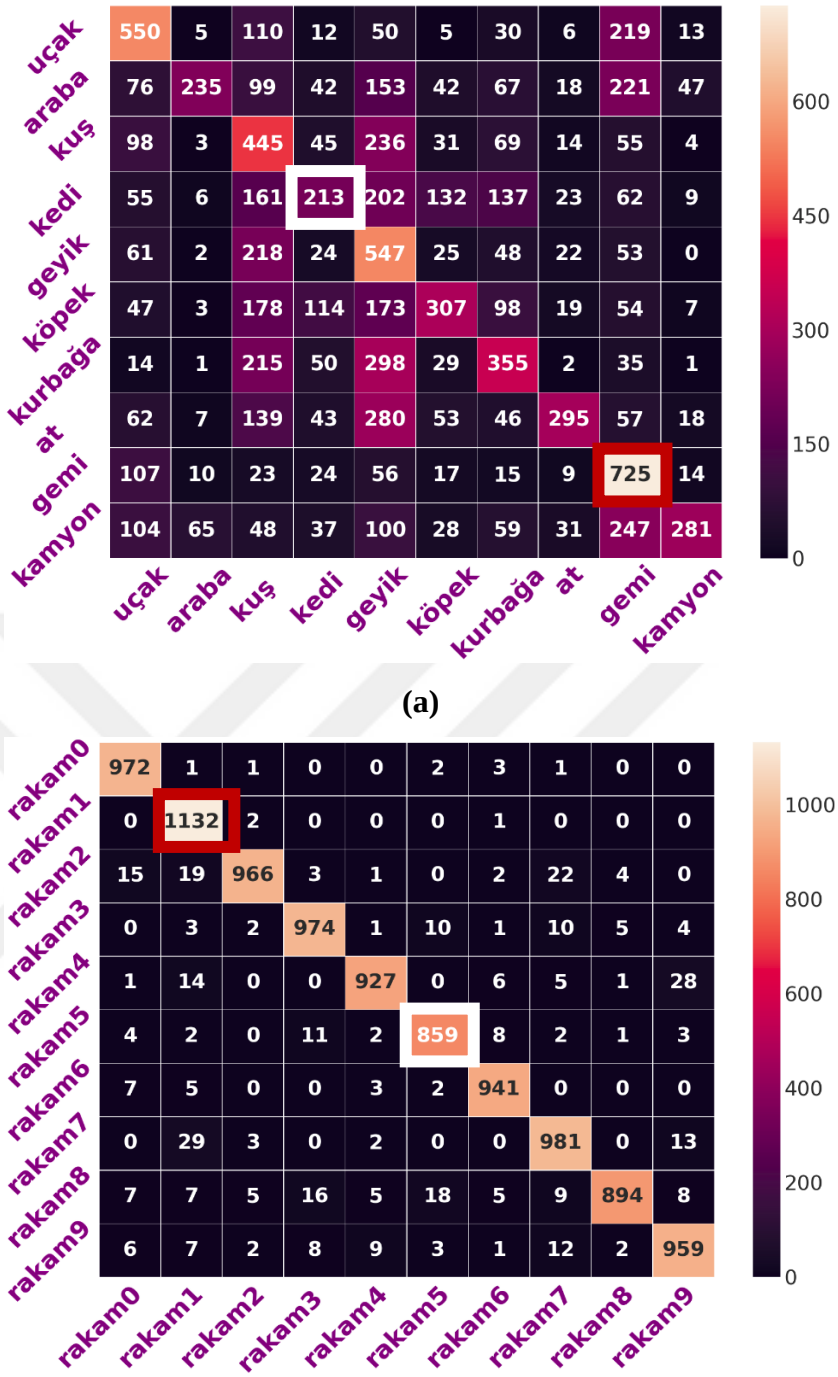
Çizelge 4.5. k-En yakın komşu yöntemi sınıflandırma raporlarına göre yapılan değerlendirmeler

f1-skor	En iyi: gemi sınıfı En kötü: kedi sınıfı	En iyi: rakam 6 sınıfı En kötü: rakam 7 ve 9 sınıfları
kesinlik	Gemi sınıfı olarak tahmin edilen görüntülerin tüm gemi görüntülerine oranı %42’dir.	Rakam 6 sınıfı olarak tahmin edilen görüntülerin tüm 6 rakamı görüntülerine oranı %97,2’dir.
duyarlılık	Gemi olarak tahmin edilen görüntülerin %72,5’i doğrudur.	6 rakamı olarak tahmin edilen görüntülerin %98,2’si doğrudur.

b. Karmaşıklık Matrisi

KNN9c modelinin CIFAR-10 veri seti ve KNN5f modelinin MNIST veri seti üzerindeki karmaşıklık matrisleri Şekil 4.4’te verilmiştir.

.



Şekil 4.4. k-En yakın komşu yöntemi karmaşıklık matrisleri **(a)** CIFAR-10 **(b)** MNIST

Şekil 4.4'te verilen sonuçlara göre yapılan değerlendirmeler Çizelgede 4.6'da verildiği gibidir.

Çizelge 4.6. k-En yakın komşu yöntemi karmaşıklık matrislerine göre yapılan değerlendirmeler

CIFAR-10	MNIST
• En iyi tahmin edilen sınıf 725 tane doğru sınıflandırma ile gemidir. • Gemi görüntüleri 107 tane ile en çok uçağa; 9 tane ile en az ata benzetilmiştir. • En kötü tahmin edilen sınıf 213 tane doğru sınıflandırma ile kedidir.	• En iyi tahmin edilen sınıf 1132 tane doğru sınıflandırma 1 rakamıdır. • 1 rakamı görüntüleri 2 tane ile en çok 2 rakamına benzetilirken; 3, 4, 5, 6, 7 ve 8 rakamı olarak hiç sınıflandırılmamıştır. • En kötü tahmin edilen sınıf 859 tane doğru sınıflandırma ile 5 rakamıdır.

4.1.3. Naive Bayes Yöntemi Eğitim Sonuçları

Naive bayes yöntemi varsayılan değerler üzerinden veri setlerine uygulanmıştır. Sınıflandırma tahmin sonuçları Çizelge 4.7'de verilmiştir.

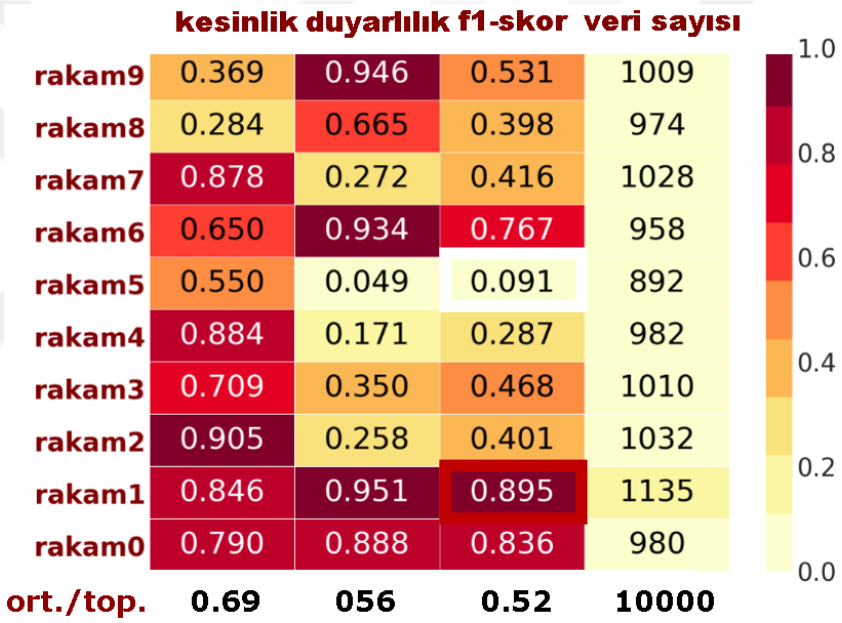
Çizelge 4.7. Naive bayes yönteminin veri setleri üzerindeki sınıflandırma tahmin sonuçları

Model	Eğitim Süresi		Doğruluk Oranı (%)	
	CIFAR-10	MNIST	CIFAR-10	MNIST
NB	6 saniye	2 saniye	29.76	55.58

b. Sınıflandırma Raporu

Naive bayes yönteminin CIFAR-10 ve MNIST veri setleri üzerindeki sınıflandırma raporları Şekil 4.5'te verilmiştir.

(a)



(b)

Şekil 4.5. Naive bayes yöntemi sınıflandırma raporları **(a)** CIFAR-10 **(b)** MNIST

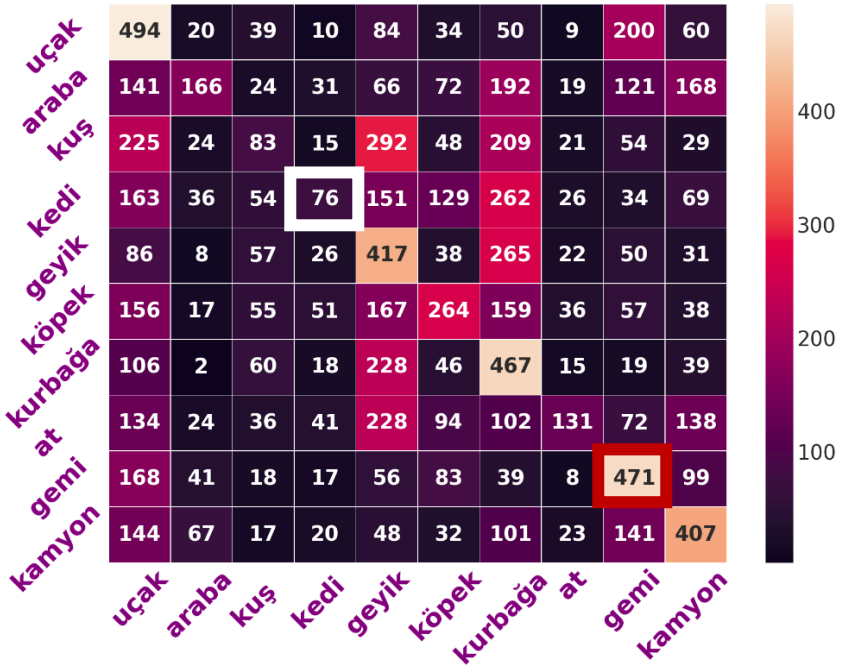
Şekil 4.5'te verilen sonuçlara göre yapılan değerlendirmeler Çizelge 4.8'de verildiği gibidir.

Çizelge 4.8. Naive bayes yöntemi sınıflandırma raporlarına göre yapılan değerlendirmeler

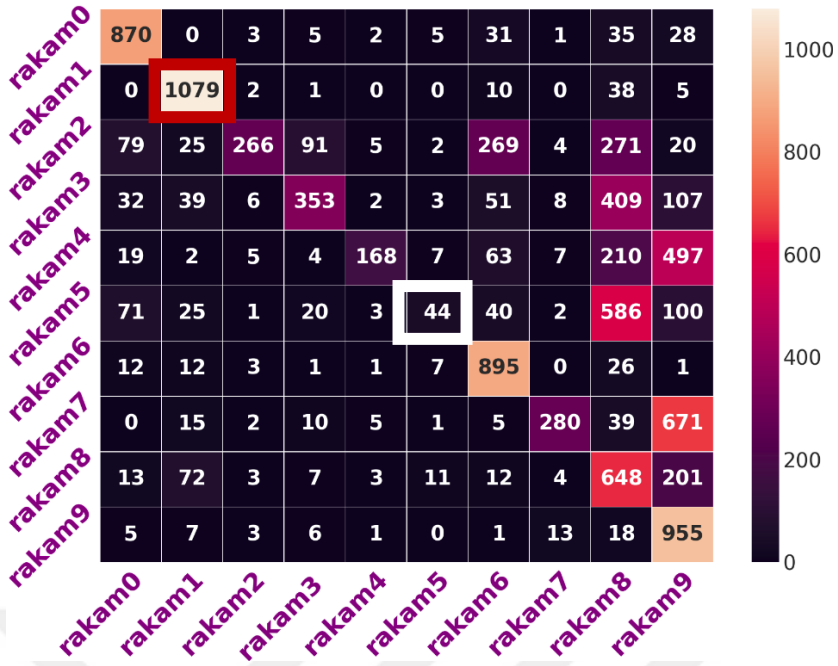
	CIFAR-10	**MNIST**
f1-skor	En iyi: gemi sınıfı En kötü: kuş sınıfı	En iyi: rakam 1 sınıfı En kötü: rakam 5 sınıfı
kesinlik	Gemi sınıfı olarak tahmin edilen görüntülerin tüm gemi görüntülerine oranı %38,6'dır.	Rakam 1 sınıfı olarak tahmin edilen görüntülerin tüm 1 rakamı görüntülerine oranı %84,6'dır.
duyarlılık	Gemi olarak tahmin edilen görüntülerin %47,1'i doğrudur.	1 rakamı olarak tahmin edilen görüntülerin %95,1'i doğrudur.

c. Karmaşıklık Matrisi

Naive bayes yönteminin CIFAR-10 ve MNIST veri setleri üzerindeki karmaşıklık matrisleri Şekil 4.6'da verilmiştir.



(a)



(b)

Şekil 4.6. Naive bayes yöntemi karmaşıklık matrisleri **(a)** CIFAR-10 **(b)** MNIST

Şekil 4.6'da verilen tahmin sonuçlarına göre yapılan değerlendirmeler Çizelge 4.9'da verilmiştir.

Çizelge 4.9. Naive bayes modeli karmaşıklık matrislerine göre yapılan değerlendirmeler

CIFAR-10	MNIST
• En iyi tahmin edilen sınıf 471 tane doğru sınıflandırma ile gemidir.	• En iyi tahmin edilen sınıf 1079 tane doğru sınıflandırma 1 rakamıdır.
• Gemi görüntüleri 168 tane ile en çok uçağa; 8 tane ile en az ata benzetilmiştir.	• 1 rakamı görüntüleri 38 tane ile en çok 8 rakamına benzetilirken; 0, 4, 5 ve 7 rakamı olarak hiç sınıflandırılmamıştır.
• En kötü tahmin edilen sınıf 76 tane doğru sınıflandırma ile kedidir.	• En kötü tahmin edilen sınıf 44 tane doğru sınıflandırma ile 5 rakamıdır.

4.1.4. Karar Ağacı Yöntemi Eğitim Sonuçları

Çalışmada CIFAR-10 ve MNIST veri setleri üzerinde parametreler değiştirilerek karar ağaçları yöntemi uygulanmıştır. Doğruluk oranları Çizelge 4.10'da verilmiştir.

Çizelge 4.10. Karar ağacı yöntemi sınıflandırma tahmin sonuçları **(a)** Sınıflandırma ve regresyon **(b)** Entropiye dayalı **(c)** Rastgele orman

(a)

Model Adı	Eğitim Süresi		Doğruluk Oranı (%)	
	CIFAR-10	MNIST	CIFAR-10	MNIST
KA	3 dakika	23 saniye	26,7	87,77
KASR3	47 saniye	3 saniye	23,94	49,53
KASR5	50 saniye	7 saniye	26,69	67,47
KASR7	69 saniye	9 saniye	29,29	78,53
KASR9	89 saniye	10 saniye	30,38	85,09
KASR10	3 dakika	11 saniye	30,51	86,55
KASR11	2 dakika	12 saniye	30,21	87,35
KASR15	78 saniye	15 saniye	23,35	88,24
KASR30	59 saniye	21 saniye	26,49	87,92
KASR50	4 dakika	23 saniye	29,54	87,83
KASR58	4 dakika	22 saniye	26,82	87,9
KASR59	4 dakika	24 saniye	26,92	87,92
KASR60	3 dakika	22 saniye	26,84	87,92
KASR61	3 dakika	24 saniye	26,8	87,64
KASR62	4 dakika	25 saniye	26,61	87,66
KASR63	4 dakika	24 saniye	27,32	87,88
KASR65	4 dakika	24 saniye	26,82	87,72
KASR69	4 dakika	25 saniye	26,81	87,64
KASR70	4 dakika	23 saniye	26,84	87,62
KASR75	3 dakika	24 saniye	26,88	87,6
KASR80	3 dakika	23 saniye	26,53	87,58

(b)

Modeller	Eğitim Süresi		Doğruluk Oranı (%)	
	CIFAR-10	**MNIST**	**CIFAR-10**	**MNIST**
KAE3	31 saniye	4 saniye	23,35	49,18
KAE5	62 saniye	7 saniye	26,49	69,95
KAE7	2 dakika	12 saniye	28,31	79,85
KAE9	8 dakika	15 saniye	26,47	86,03
KAE10	3 dakika	16 saniye	29,75	87,14
KAE11	5 dakika	18 saniye	29,59	88,31
KAE15	7 dakika	20 saniye	27,71	88,61
KAE30	8 dakika	21 saniye	26,47	88,7
KAE50	8 dakika	21 saniye	26,35	88,74
KAE58	8 dakika	21 saniye	26,52	88,39
KAE59	8 dakika	21 saniye	26,38	88,46
KAE60	8 dakika	22 saniye	26,68	88,85
KAE61	8 dakika	21 saniye	25,98	88,78
KAE62	8 dakika	22 saniye	26,32	88,64
KAE63	7 dakika	22 saniye	26,43	88,36
KAE65	8 dakika	21 saniye	26,87	88,44
KAE69	7 dakika	21 saniye	26,42	88,56
KAE70	7 dakika	21 saniye	26,08	88,64
KAE75	9 dakika	21 saniye	26,5	88,51
KAE80	9 dakika	21 saniye	26,04	88,49

(c)

Model Adı	Eğitim Süresi		Doğruluk Oranı (%)	
	CIFAR-10	MNIST	CIFAR-10	MNIST
RO	22 saniye	4 saniye	36,05	94,75
RO10	20 dakika	5 dakika	43,48	94,98
RO11	25 dakika	6 dakika	44,72	95,62
RO12	63 dakika	6 dakika	45,75	96,14
RO13	69 dakika	6 dakika	46,43	96,46
RO20	34 dakika	7 dakika	48,54	97,07
RO50	34 dakika	8 dakika	49,2	97,12
RO60	33 dakika	8 dakika	49,43	97,13
*RO62	34 dakika	7 dakika	49,56	97,24
RO63	36 dakika	7 dakika	49,40	97,24
RO65	35 dakika	7 dakika	49,32	97,24
RO70	34 dakika	7 dakika	49,28	97,24

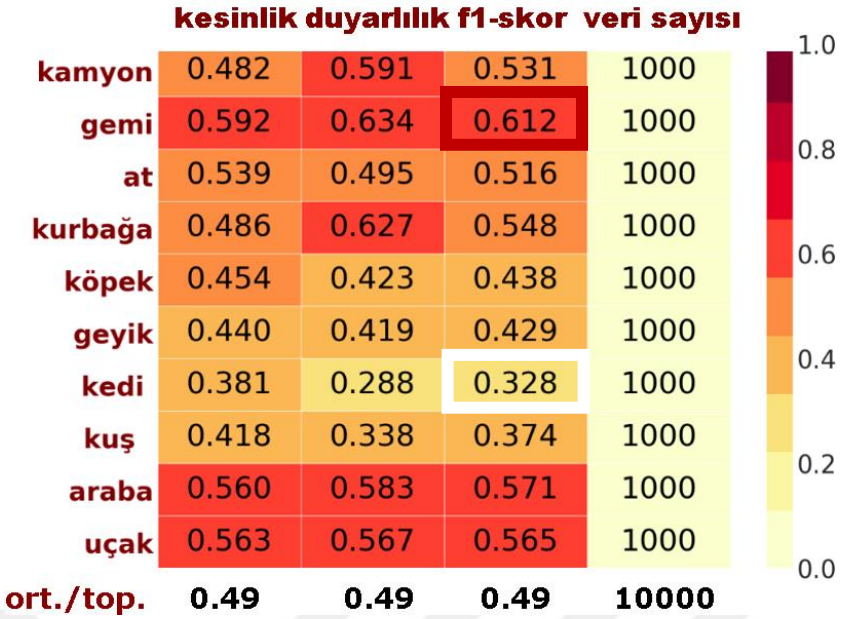
(*En yüksek doğruluk oranı elde edile model)

Şekil 4.13'te ağaç derinliğinin doğruluk oranına etkisinin gösteren grafikler verilmiştir. Şekil 4.13.a'da verilen grafikte, Sınıflandırma ve Regresyon, Entropiye Dayalı karar ağaçlarında en uygun ağaç derinliğinin 10 olduğu modellerde ağaç derinliği değiştikçe doğruluk oranlarının da değiştiği görülmektedir. Rastgele ormanda tüm modellerde ağaç sayısı 1000 olarak belirlenmiştir.

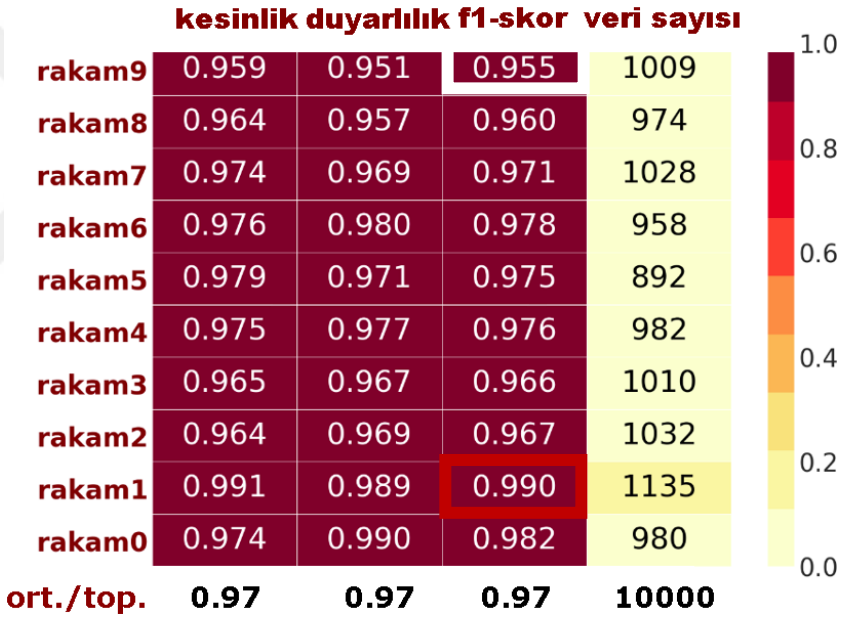
CIFAR-10 ve MNIST veri setlerinde en yüksek doğruluk oranı RO62 modelinden elde edilmiştir.

a. Sınıflandırma Raporu

RO62 modelinin CIFAR-10 ve MNIST veri setleri üzerindeki sınıflandırma raporları Şekil 4.7'de verilmiştir.



(a)



(b)

Şekil 4.7. Rastgele orman yöntemi sınıflandırma raporları **(a)** CIFAR-10 **(b)** MNIST

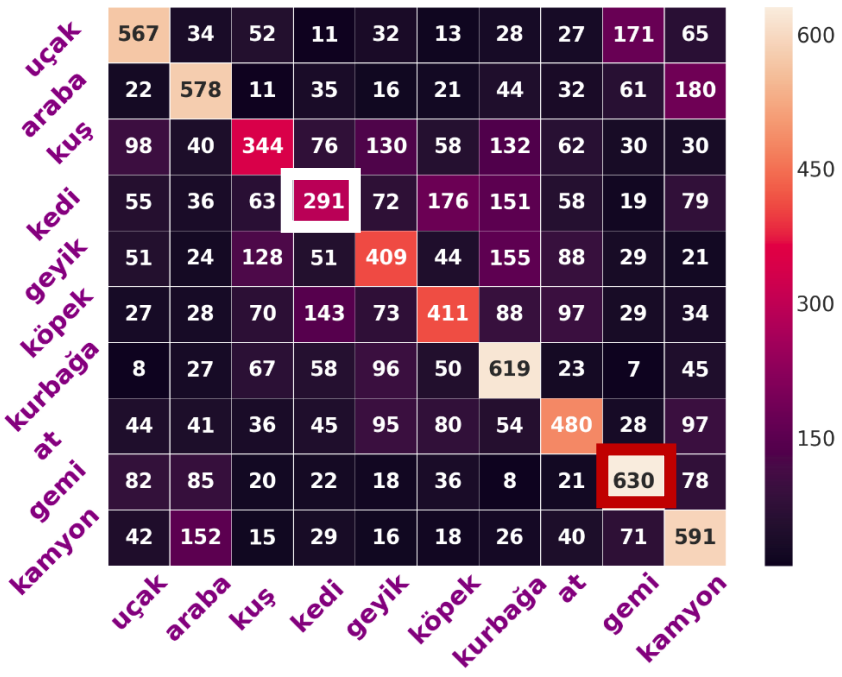
Şekil 4.7'de verilen sonuçlara göre yapılan değerlendirmeler Çizelge 4.11'de verildiği gibidir.

Çizelge 4.11. Rastgele orman yöntemi sınıflandırma raporlarına göre yapılan değerlendirmeler

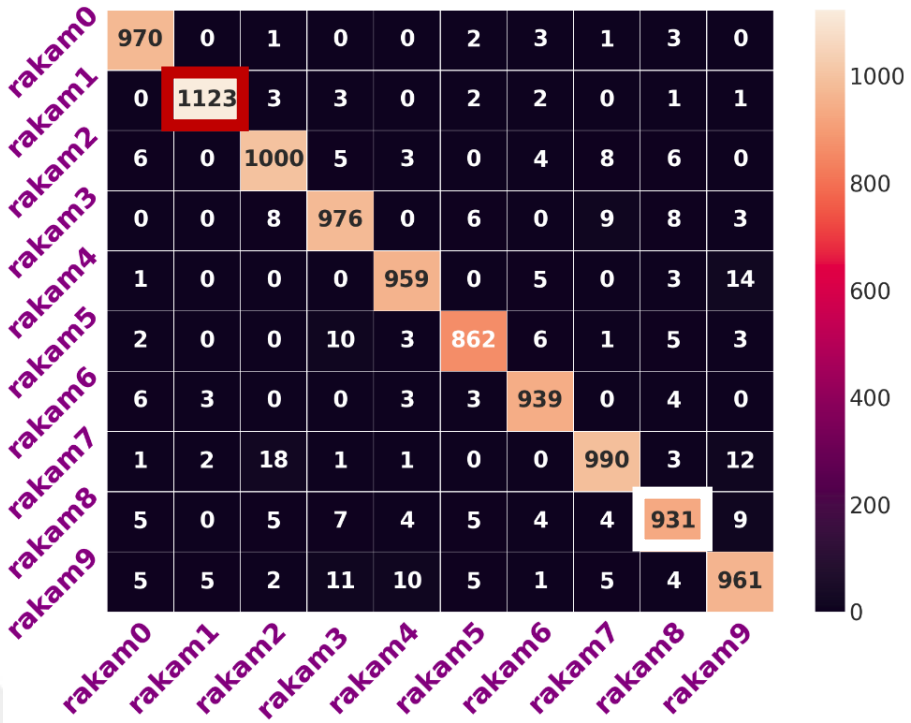
	CIFAR-10	**MNIST**
f1-skor	En iyi: gemi sınıfı En kötü: kedi sınıfı	En iyi: rakam 1 sınıfı En kötü: rakam 9 sınıfı
kesinlik	Gemi sınıfı olarak tahmin edilen görüntülerin tüm gemi görüntülerine oranı %59,2'dir.	Rakam 1 sınıfı olarak tahmin edilen görüntülerin tüm 1 rakamı görüntülerine oranı %99,1'dur.
duyarlılık	Gemi olarak tahmin edilen görüntülerin %63,4'ü doğrudur.	1 rakamı olarak tahmin edilen görüntülerin %98,9'i doğrudur.

c. Karmaşıklık Matrisi

RO62 modelinin CIFAR-10 ve MNIST veri setleri üzerindeki karmaşıklık matrisleri Şekil 4.8'de verilmiştir.



(a)



(b)

Şekil 4.8. Rastgele orman yöntemi karmaşıklık matrisleri **(a)** CIFAR-10 **(b)** MNIST

Şekil 4.8'de verilen tahmin sonuçlarına göre yapılan değerlendirmeler Çizelge 4.12'de verilmiştir.

Çizelge 4.12. Rastgele orman yöntemi karmaşıklık matrislerine göre yapılan değerlendirmeler

CIFAR-10	MNIST
• En iyi tahmin edilen sınıf 630 tane doğru sınıflandırma ile gemidir.	• En iyi tahmin edilen sınıf 1123 tane doğru sınıflandırma 1 rakamıdır.
• Gemi görüntüleri 85 tane ile en çok arabaya, 8 tane ile en az kurbağaya benzetilmiştir.	• 1 rakamı görüntüleri 3 tane ile en çok 2 ve 3 rakamına benzetilirken; 0, 4 ve 7 rakamı olarak hiç sınıflandırılmamıştır.
• En kötü tahmin edilen sınıf 291 tane doğru sınıflandırma ile kedidir.	• En kötü tahmin edilen sınıf 862 tane doğru sınıflandırma ile 5 rakamıdır.

4.2. Derin Öğrenme Yöntemi Eğitim Sonuçları

Çalışmada, AlexNet ve BasitNet modelleri ile CIFAR-10 ve MNIST veri setleri üzerinde sınıflandırma tahmini yapılmıştır. AlexNet modelinin CIFAR-10 ve MNIST veri setleri üzerindeki eğitim süreleri ve sınıflandırma sonuçları Çizelge 4.13'te verilmiştir.

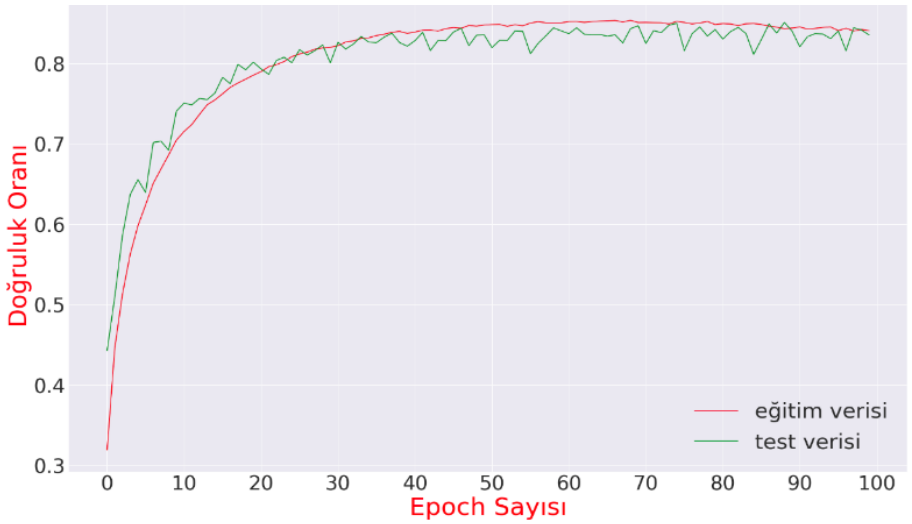
Çizelge 4.13. Derin öğrenme yöntemi sınıflandırma tahmin sonuçları

Model	Eğitim Süresi		Doğruluk Oranı (%)	
	CIFAR-10	MNIST	CIFAR-10	MNIST
AlexNet	19 saat 46 dakika	9 saat 24 dakika	72,43	93,67
BasitNet	67 saat	29 sat	84,1	98,75

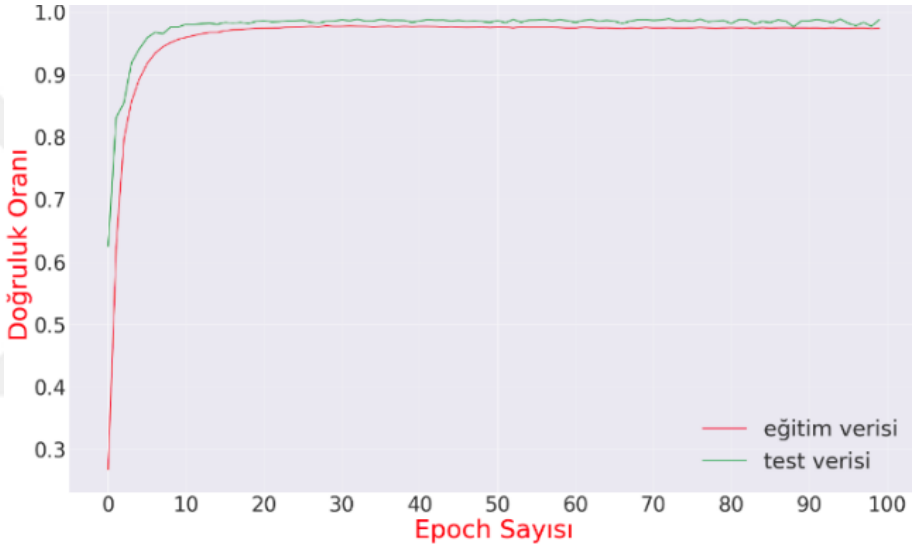
4.2.1. BasitNet Modeli

a. Grafik Sonuçları

100 epoch'ta eğitim ve test verileri üzerinde gerçekleştirilen sınıflandırma işleminin doğruluk oranlarını gösteren grafik Şekil 4.9'da verilmiştir.



(a)



(b)

Şekil 4.9. BasitNet modeli, 100 epoch süresince doğruluk oranları değişim grafiği **(a)** CIFAR-10 **(b)** MNIST

Şekil 9'da verilen grafikte 100 epoch süresince elde edilen doğruluk oranlarının arttığı görülmektedir. Test ve eğitim verilerinden elde edilen doğruluk oranlarının birbirine yakın olması aşırı uyumun olmadığını göstermektedir.

b. Sınıflandırma Raporu

BasitNet modeli ile CIFAR-10 ve MNIST veri setleri üzerindeki sınıflandırma raporları Şekil 4.10'da verilmiştir.

sınıf	kesinlik	duyarlılık	f1-skor
uçak	0.77	0.92	0.84
araba	0.95	0.94	0.94
kuş	0.79	0.76	0.77
kedi	0.65	0.75	0.70
geyik	0.88	0.77	0.82
köpek	0.94	0.60	0.73
kurbağa	0.75	0.94	0.83
at	0.90	0.86	0.88
gemi	0.92	0.91	0.91
kamyon	0.92	0.91	0.92
ortalama/toplam	0.85	0.84	0.83

(a)

sınıf	kesinlik	duyarlılık	f1-skor
Rakam 0	0.99	1.00	0.99
Rakam 1	0.99	1.00	1.00
Rakam 2	0.97	0.98	0.97
Rakam 3	0.99	1.00	0.99
Rakam 4	1.00	0.99	0.99
Rakam 5	0.98	0.97	0.98
Rakam 6	0.99	0.98	0.98
Rakam 7	0.98	0.99	0.99
Rakam 8	1.00	0.99	0.99
Rakam 9	0.99	0.99	0.99
ortalama/toplam	0.99	0.99	0.99

(b)

Şekil 4.10. BasitNet modeli sınıflandırma raporları **(a)** CIFAR-10 **(b)** MNIST

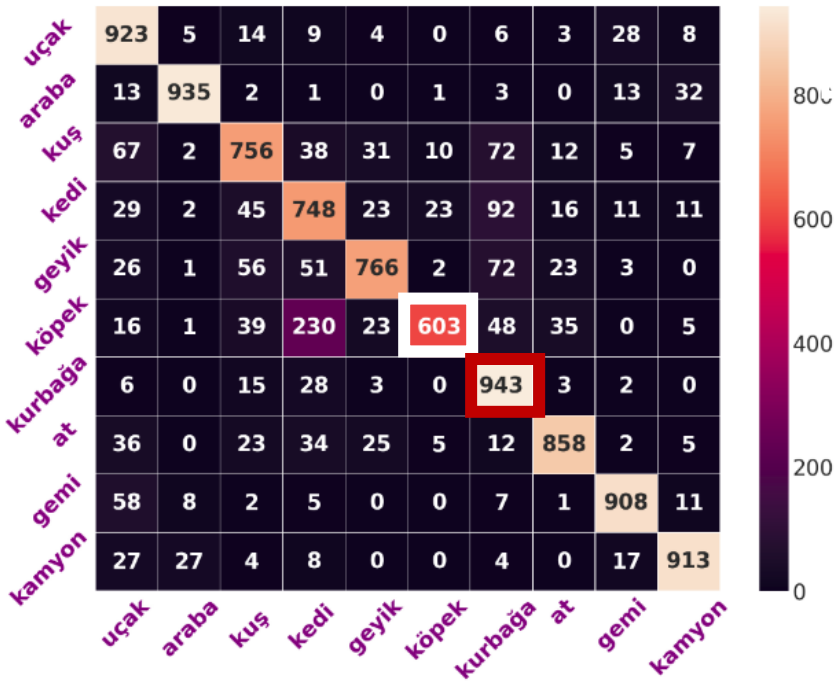
Şekil 4.10'da verilen sonuçlara göre yapılan değerlendirmeler Çizelge 4.14'te verildiği gibidir.

Çizelge 4.14. BasitNet modeli sınıflandırma raporlarına göre yapılan değerlendirmeler

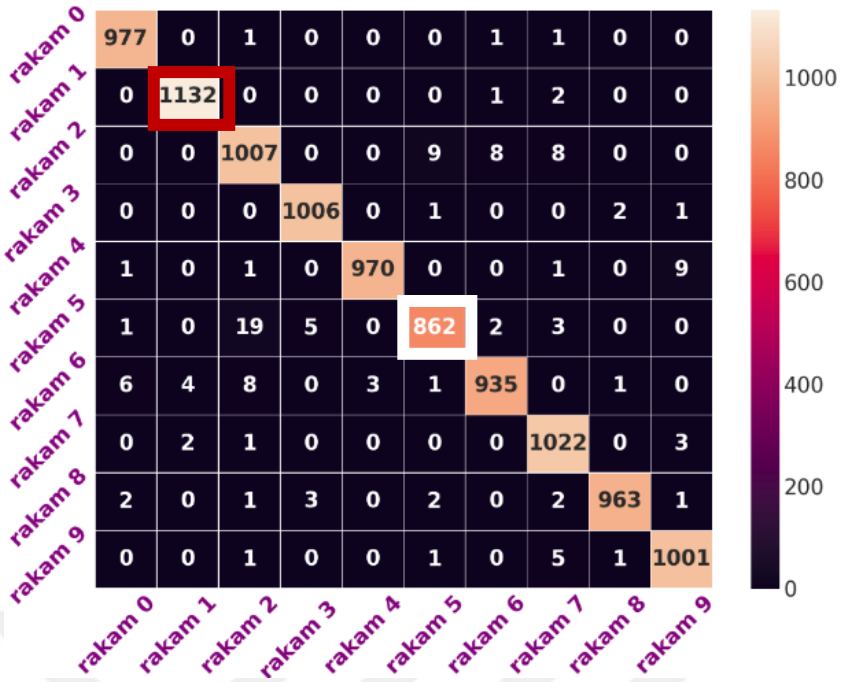
	CIFAR-10	**MNIST**
f1-skor	En iyi: araba sınıfı En kötü: kedi sınıfı	En iyi: rakam 1 sınıfı En kötü: rakam 2 sınıfı
kesinlik	Araba sınıfı olarak tahmin edilen görüntülerin tüm araba görüntülerine oranı %99'dır.	Rakam 1 sınıfı olarak tahmin edilen görüntülerin tüm 1 rakamı görüntülerine oranı %99'dur.
duyarlılık	Araba olarak tahmin edilen görüntülerin tamamı doğrudur.	1 rakamı olarak tahmin edilen görüntülerin tamamı doğrudur.

c. Karmaşıklık Matrisi

BasitNet modeline ait sonuçları gösteren karmaşıklık matrisinin ekran görüntüsü Şekil 4.11'de verilmiştir.



(a)



(b)

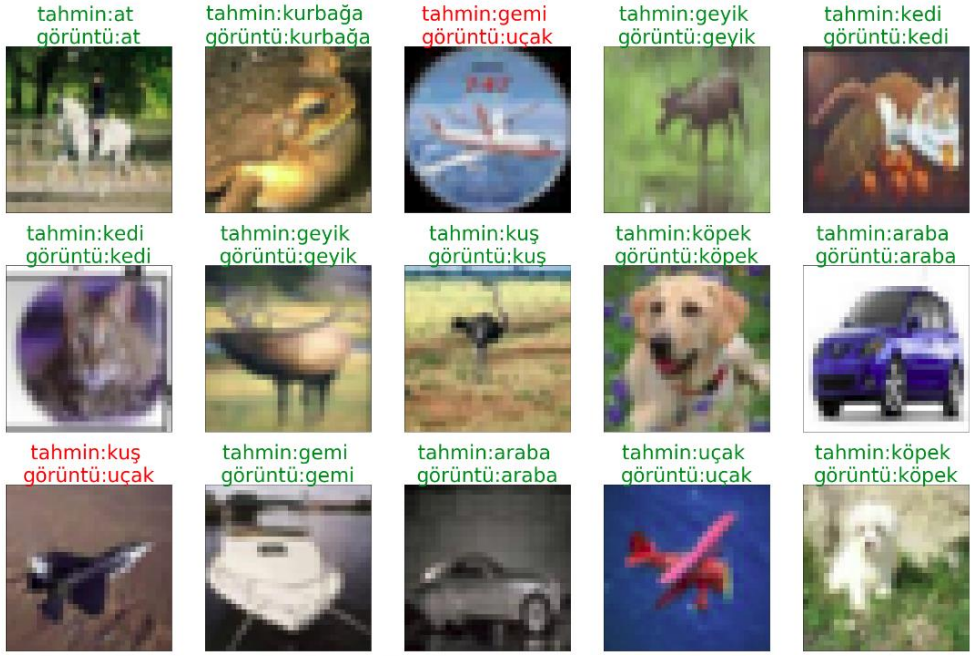
Şekil 4.11. BasitNet modeli karmaşıklık matrisi **(a)** CIFAR-10 **(b)** MNIST

Şekil 4.11'de verilen sonuçlara göre Çizelge 4.15'teki değerlendirmeler yapılabilir.

Çizelge 4.15. BasitNet modeli karmaşıklık matrislerine göre yapılan değerlendirmeler

CIFAR-10	**MNIST**
• En iyi tahmin edilen sınıf 943 tane doğru sınıflandırma ile kurbağadır.	• En iyi tahmin edilen sınıf 1132 tane doğru sınıflandırma 1 rakamıdır.
• Gemi görüntüleri 28 tane ile en çok kediye benzetilirken; araba, köpek ve kamyon sınıfı olarak hiç sınıflandırılmamıştır.	• 1 rakamı görüntüleri 2 tane ile en çok 7 rakamına benzetilirken; 0, 2, 3, 4, 5, 8 ve 9 rakamı olarak hiç sınıflandırılmamıştır.
• En kötü tahmin edilen sınıf 603 tane doğru sınıflandırma ile köpektir.	• En kötü tahmin edilen sınıf 862 tane doğru sınıflandırma ile 5 rakamıdır.

En yüksek doğruluk oranı elde edilen BasitNet modeli ile CIFAR-10 ve MNIST veri setleri (test verileri) üzerine tahmin yapılmıştır. Şekil 4.12' verilen sonuçlara göre CIFAR-10 veri setinde gemi ve uçak görüntüleri kuş olarak tahmin edilirken MNIST veri setinde yanlış tahmin edilen test verisi olmamıştır.



(a)

(b)

Şekil 4.12. BasitNet modeli nesne tahmin sonuçları **(a)** CIFAR-10 **(b)** MNIST

4.3. Makine Öğrenmesi ve Derin Öğrenme Yöntemleri Sonuçları

Makine öğrenmesi ve derin öğrenme yöntemleri ile eğitilen CIFAR-10 ve MNIST veri setlerinin eğitim sonuçları ve süreleri Çizelge 4.16'da verilmiştir.

Çizelge 4.16. Makine ve derin öğrenme modelleri ile eğitilen CIFAR-10 ve MNIST veri setlerinin eğitim sonuçları ve süreleri

	CIFAR-10	MNIST	CIFAR-10	MNIST
Model		**Eğitim Süreleri**		**Doğruluk Oranları**
LR	5 dakika	32 saniye	40,48	91,91
KNN	29 dakika	3 saat 44 dakika	39,53	96,35
NB	6 saniye	2 saniye	29,76	55,58
RO	34 dakika	7 dakika	49,56	97,24
AlexNet	19 saat 46 dakika	9 saat 24 dakika	72,43	93,67
BasitNet	67 saat	29 saat	**84,1**	**98,75**

Çizelge 4.16'da verilen sonuçlara göre CIFAR-10 ve MNIST veri setlerinde en iyi tahminler derin öğrenme modeli olan BasitNet'ten elde edilmiştir.

Makine ve derin öğrenme modellerinden elde edilen sınıflandırma raporlarına göre en iyi ve en kötü tahmin edilen sınıflar Çizelge 4.17'de verilmiştir.

Çizelge 4.17. Makine öğrenmesi ve derin öğrenme modellerinden elde edilen sınıflandırma raporlarına göre en iyi ve en kötü tahmin edilen sınıflar

Modeller	**En İyi Tahmin Edilen Sınıf**		**En Kötü Tahmin Edilen Sınıf**	
	CIFAR-10	**MNIST**	**CIFAR-10**	**MNIST**
LR	gemi	rakam 1	kedi	rakam 8
KNN	gemi	rakam 0, 1 ve 6	kedi	rakam 9
NB	gemi	rakam 1	kuş	rakam 5
RO	gemi	rakam 1	kedi	rakam 9
AlexNet	kurbağa	rakam 1	kedi	rakam 5
BasitNet	kurbağa	rakam 1	kedi	rakam 5
Sonuç	**gemi, kurbağa**	**rakam1**	**kedi**	**rakam 5**

Çizelge 4.17'de verilen sonuçlara göre CIFAR-10 veri setinde makine öğrenmesi yöntemleri sonuçlarına göre en iyi tahmin edilen sınıf gemi iken derin öğrenme yöntemi sonuçlarına göre ise kurbağadır. MNIST veri setinde makine öğrenmesi yöntemleri ve derin öğrenme yöntemi sonuçlarına göre en iyi tahmin edilen sınıf genelde rakam 1'dir. CIFAR-10 veri setinde her iki yöntemde de en kötü tahmin edilen sınıf genelde kedi iken MNIST veri setinde genelde rakam 5'tir.

Makine ve derin öğrenme modellerinden elde edilen karmaşıklık matrislerine göre birbirine benzetilen sınıflar Çizelge 4.18'de verilmiştir.

Çizelge 4.18. Makine ve derin öğrenme modellerinden elde edilen karmaşıklık matrislerine göre birbirine benzetilen sınıflar **(a)** CIFAR-10 **(b)** MNIST

(a)

			Makine Öğrenmesi		Derin Öğrenme	
Sınıf	**LR**	**KNN**	**NB**	**RO**	**AlexNet**	**BasitNet**
***uçak**	gemi	gemi	gemi	gemi	gemi	gemi
***araba**	kamyon	gemi	kurbağa	kamyon	kamyon	kamyon
kuş	kurbağa	geyik	geyik	kurbağa	kurbağa	kurbağa
kedi	köpek	geyik	kurbağa	köpek	kurbağa	kurbağa
***geyik**	kurbağa	kuş	kurbağa	kurbağa	kurbağa	kurbağa
köpek	kedi	kuş	geyik	kedi	kurbağa	kedi
kurbağa	kedi	geyik	geyik	geyik	kedi	kedi
at	köpek	geyik	geyik	kamyon	kurbağa	uçak
***gemi**	uçak	uçak	uçak	araba	kamyon	uçak
kamyon	araba	gemi	uçak	araba	kurbağa	uçak,araba

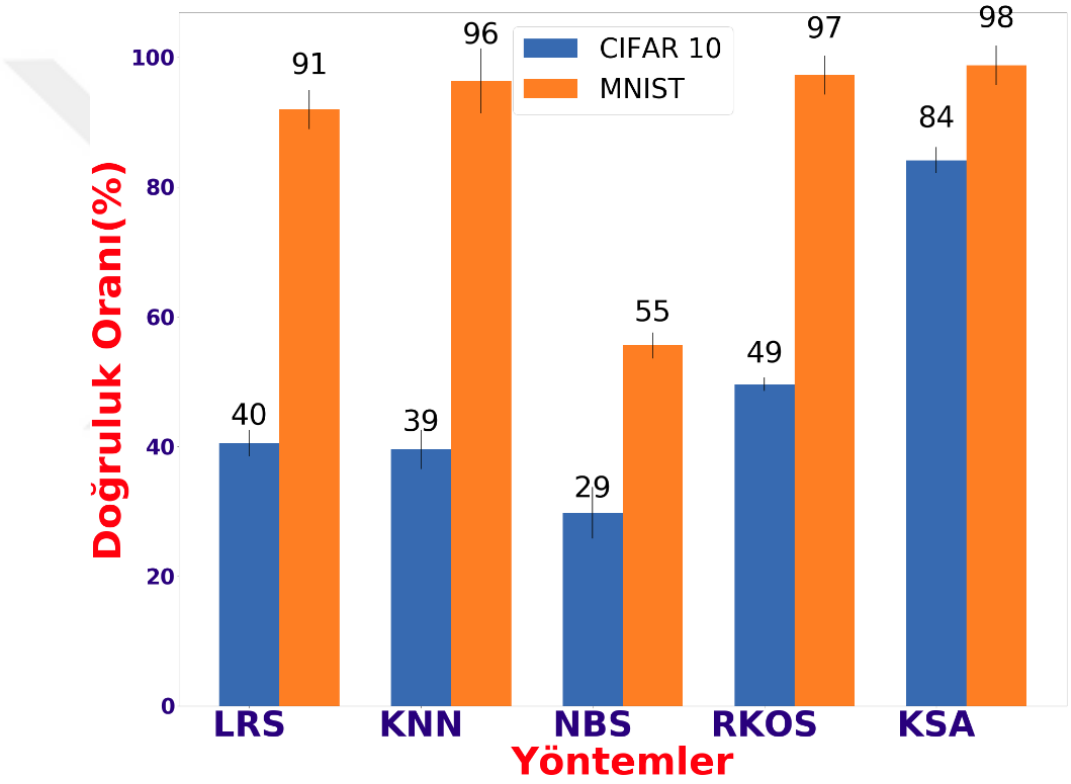
(b)

			Makine Öğrenmesi		Derin Öğrenme	
Sınıf	**LR**	**KNN**	**NB**	**RO**	**AlexNet**	**BasitNet**
***rakam 0**	rakam 6	rakam 6	rakam 8	rakam 6,8	rakam 8	rakam 2,6,7
rakam 1	rakam 8	rakam 2	rakam 8	rakam 2,3	rakam 8	rakam 7
rakam 2	rakam 8	rakam 7	rakam 8	rakam 7	rakam 5	rakam 5
rakam 3	rakam 5,8	rakam 5	rakam 8	rakam 7	rakam 8	rakam 8
***rakam 4**	rakam 9	rakam 9	rakam 9	rakam 9	rakam 9	rakam 9
rakam 5	rakam 8	rakam 3	rakam 8	rakam 3	rakam 2	rakam 2
rakam 6	rakam 5	rakam 0	rakam 8	rakam 0	rakam0,1,2	rakam 2
***rakam 7**	rakam 9	rakam 1	rakam 9	rakam 2	rakam 9	rakam 9
rakam 8	rakam 5	rakam 3	rakam 9	rakam 9	rakam 5	rakam 3
rakam 9	rakam 4	rakam 7	rakam 8	rakam 3	rakam 8	rakam 7

(*Farklı yöntemlerden elde edilen sonuçlara göre birbirine en çok benzeyenler)

Karmaşıklık matrislerinden elde edilen sonuçlara göre CIFAR-10 veri setinde, uçak görüntüleri genellikle gemiye, araba görüntüleri kamyona, geyik görüntüleri kurbağaya ve gemi görüntüleri de uçağa benzetilmiştir.

MNIST veri setinde genellikle rakam 0 görüntüleri 6 rakamına, rakam 3 görüntüleri 8 rakamına, rakam 4 ve 7 görüntüleri 9 rakamına benzetilmiştir. Şekil 4.13'te çalışmada kullanılan Makine öğrenmesi ve derin öğrenme yöntemlerine ait doğruluk oranlarının grafiksel sonuçları verilmiştir.



Şekil 4.13. En yüksek doğruluk oranı elde edilen modellerin sonuçlarının grafiksel gösterimi

Şekil 4.13'te verilen grafikte, CIFAR-10 ve MNIST veri setlerinde en yüksek doğruluk oranının BasitNet modelinden elde edildiği görülmektedir. Ayrıca ikinci en iyi tahmin sonucu her iki veri setinde de rastgele orman yönteminden elde edilmiştir.

5. DEĞERLENDİRME VE ÖNERİLER

Çağın gereklerine uygun olarak birçok ülke, yapay zekâ alanında ulusal strateji çalışmaları yapmaktadır. Yapay zekanın ülkeler bazında önemli bir yere sahip olması gelişimini de hızlandırmaktadır. Son yıllarda yapılan yapay zekâ alanındaki çalışmalarda en başarılı sonuçlar Makine öğrenmesi yönteminden elde edilmiştir. Makine öğrenmesi yöntemi, doğal dil işleme, bilgisayarlı görü, konuşma tanıma, veri madenciliği ve duygu analizi gibi birçok alanda oldukça iyi sonuçlar vermektedir.

Çalışmada, makine öğrenmesi ve derin öğrenme yöntemlerini karşılaştırmak amacıyla ayrı olarak ele alınmıştır. Ayrıca makine öğrenmesi ve derin öğrenme yöntemlerinin algoritma yapıları birbirinde farklıdır. Makine öğrenmesi yönteminde öznitelik çıkarma ve seçme, algoritma dışında yapılırken derin öğrenme yönteminde gizli katmanlarda tahmin işlemi ile birlikte yapılmaktadır. Geri yayılımlı bir derin öğrenme mimarisinde sonuçlar iyileştirilene kadar döngüsel olarak tekrarlanır. Bu da derin öğrenme yönteminin diğer yöntemlere göre daha avantajlı olmasını sağlar. Bununla birlikte hızla artan veriler, depolama maliyeti ve zaman kaybı gibi problemlerden dolayı, büyük veri çağın önde gelen sorunlarındandır. Makine öğrenmesi yönteminin mantığı, mevcut verilerden yola çıkarak olası yeni durumlar için doğru tahminler yapmaya dayandığından verilerin büyük boyutta olması avantajlı bir durumdur. Veri miktarının çok fazla olması makine öğrenmesi yönteminde analizi güçleştirirken, Derin öğrenme yöntemlerinin her iterasyonunda farklı özniteliklerin çıkarılmasını sağladığından doğru kararlar vermeyi kolaylaştırmaktadır. Ayrıca derin öğrenme, makine öğrenmesi yöntemlerinden bir tanesi olmasına rağmen, algoritma yapısının farklı olması ve diğer yöntemlerden çok daha yüksek doğrulukta tahmin yapabilmesi sayesinde en çok tercih edilen yöntem olmuştur.

Çalışmada, CIFAR-10 ve MNIST veri setleri ile nesne tanıma çalışması yapılmıştır. Makine öğrenmesi ve derin öğrenme yöntemleri uygulanarak elde edilen sonuçlar ekran görüntüsü olarak gösterilmiştir. Elde edilen sonuçlara göre aşağıdaki değerlendirmeler ve öneriler yapılabilir:

- En kısa eğitim süresi naive bayes yönteminden elde edilirken, en uzun eğitim süresi BasitNet modelinden elde edilmiştir (Çizelge 4.16). Çalışmada makine öğrenmesi yöntemlerinden elde edilen eğitim süreleri incelendiğinde, kullanılan parametre sayıları ile doğru orantılı olduğu görülmektedir. Lojistik regresyonda max iterasyon sayısı, k-en yakın komşu yönteminde komşu sayısı, karar ağacında ağaç derinliği ve sayısı ile eğitim süreleri doğru orantılı olarak artmaktadır (Çizelge 4.1, 4.4. ve 4.10).
- En düşük doğruluk oranı naive bayes modelinden elde edilirken en yüksek doğruluk oranı BasitNet modelinden elde edilmiştir. Makine öğrenmesi ve derin öğrenme yöntemleri karşılaştırıldığında CIFAR-10 veri setinde derin öğrenme yöntemlerinden elde edilen doğruluk oranlarının makine öğrenmesi yöntemlerine göre yaklaşık 2 kat daha fazla olduğu görülmektedir (Çizelge 4.7). MNIST veri setinde ise derin öğrenme yönteminden yüksek doğruluk oranları elde edilirken elde edilen doğruluk oranları
- CIFAR-10 v e MNIST veri setlerinde %70 in üzerinde doğruluk oranları, insan beynini taklit eden bir yapıya sahip olan derin öğrenme yönteminden elde edilmiştir. Bu da çok derin öğrenmenin makine öğrenmesine göre daha iyi sonuçlar verdiğini göstermektedir (Çizelge 4.19).
- Her bir model için sonuç raporları ve karmaşıklık matrisleri karşılaştırıldığında; sonuç raporlarında erilen f1-skor değerleri sınıfların doğru tahmin performanslarını ve tahminlerin kalite ölçüsünü verirken karmaşıklık matrisleri sınıfların doğru ve yanlış tahmin sayılarını verir. Doğru ve yanlış tahmin sayıları sınıfların benzerlikleri konusunda değerlendirme yapılmasını sağlar. Sınıflandırma raporlarına göre makine öğrenmesi yöntemlerinde en iyi tahmin edilen sınıflar gemi ve rakam 1 iken derin öğrenme yönteminde kurbağa ve rakam 1'dir. En kötü tahmin edilen sınıflar her iki yöntemde de kedi ve rakam 5 sınıflarıdır. (Çizelge 4.20) Bu sonuçlar söz konusu sınıflar için modellerin geliştirilmesini sağlayabilir. Örneğin otonom araçlar için yayaların trafik lambalarının ve araçların doğru sınıflandırılması, hastalar için beyin kanaması veya tümörün doğru sınıflandırması önem taşır. Dolayısı ile geliştirilen modelden yüksek önem derecesine sahip sınıflarda hatasız performans göstermesi beklenir. Bu durumda modelin, daha iyi öğrenmesini

sağlayacak öznitelik çıkarma yöntemi geliştirilebileceği gibi, sınıf için öğrenmeyi sağlayacak örnek sayısı ve çeşitliliğinin arttırılması düşünülebilir.

- Karmaşıklık matrislerinden elde edilen sonuçlara göre sınıfların doğru tahmin sayıları diğer tahmin sayılarından genellikle fazla olur. Çalışmada, bazı sınıflar için ikinci olarak yüksek tahmin edilen sınıfların farklı modellerde aynı olduğu görülmüştür. (Çizelge 4.20). Eğer bir sınıftaki görüntüler yüksek sayıda başka bir sınıf olarak yanlış tahmin ediliyor ise kuvvetle muhtemele o sınıfa benzetiliyordur. Çalışmada 6 farklı modelden elde edilen ortak sonuçlara göre CIFAR-10 veri setinde uçak görüntüleri genellikle gemiye, araba görüntüleri kamyona, geyik görüntüleri kurbağaya ve gemi görüntüleri de uçağa benzetilmiştir. MNIST veri setinde genellikle rakam 0 görüntüleri 6 rakamına, rakam 3 görüntüleri 8 rakamına, rakam 4 ve 7 görüntüleri 9 rakamına benzetilmiştir.
- Nesne tanımayı etkileyen en önemli etkenlerden bir tanesi öznitelik haritalarıdır. Makine öğrenme yöntemlerinin benzer öznitelik haritaları oluşturmasının yöntemlerde benzer sonuçlara sebep olduğu söylenebilir. Aynı şekilde derin öğrenme yönteminde öznitelik haritası oluşturma yöntemi modellerde aynı olduğundan birbirine yakın sonuçların oluştuğu düşünülebilir. Örneğin makine öğrenmesi yöntemleri sınıflandırma tahmin sonuçlarına göre en iyi tahmin edilen sınıf gemi iken derin öğrenme yönteminde kurbağadır.
- Önerilen BasitNet modeli ile 1/50 oranında daha az parametreyle AlexNet düzeyinde başarı elde edilmiştir. Elde edilen sonuçlar önerilen BasitNet diğer yöntemlerden daha başarılı olduğunu göstermiştir. Ayrıca rastgele orman yöntemi her iki veri setinde de diğer makine öğrenmesi yöntemlerine göre daha iyi tahmin yapmıştır.
- Önerilen BasitNet modeli parametreleri üzerinde değişiklik yaparak veya katman ekleyip çıkartılarak daha yüksek oranda doğruluk oranı elde edilebilir.
- GPU desteğinin olmaması sebebi ile eğitimler uzun sürmüştür. Eğitim süresinin kısaltılması ile daha çok deneme yapılarak yüksek doğruluk oranları elde edilebilir.

KAYNAKLAR

[1] Reinsel D., Gantz J., Rydning J., The evolution of data to life-critical, International Data Corporation. http://www.webcitation.org/70ITr6RPw (Erişim tarihi: 25.04.2018)

[2] O. Stenroos, Object Detection From Images Using Convolutional Neural Networks. Master's Thesis. Aalto University, Finland, 2017.

[3] Pannu, A., Artificial intelligence and its application in different areas. International Journal of Engineering and Innovative Technology (IJEIT). 4(10): 79-84, 2015.

[4] Turing, M. A., Computing machinery and intelligence. Mind. LIX(236): 433-460, 1950.

[5] Jefferson, G., The mind of mechanical man. British Medical. 1(4616): 1105-1110, 1949.

[6] McCorduck, P., Machine who think: A personel injury into the history and prospects of artificial intelligence. 523-533. A K Peters Limit, Canada , 2004.

[7] Bojarski, M., Testa, D. D., Dworakowski, D., Firner, B., Flepp, B., Goyal, P., Jackel, L. D., Monfort, M., Muller, U., Zhang, J., Zhang, X., Zhao, J., and Zieba, Explaining how a Deep Neural Network trained with end to end learning for self-driving cars. ArXiv: 1704.07911v1. 2016.

[8] Kumar, J., Applications of artificial intelligence. International Journal of Research in Engineering and Applied Sciences. 6(4): 42-49, 2016.

[9] Sezgin, M., Talaz, L., Bilişim devrimi, sibernetik iletişim ve stratejik halkla ilişkiler. Karabük Üniversitesi Sosyal Bilimler Enstitüsü Dergisi. 6(2): 559-571, 2016.

[10] Dipova, N., Görüntü analizi yöntemlerinin geoteknik mühendisliğinde kullanımı, Mehmet Akif Ersoy Üniversitesi Fen Bilimleri Enstitüsü Dergisi. 9(1):33-44, 2018.

[11] Lukac, R., Plataniotis, K. N., Single-sensor camera image processing. In Color Image Processing: Methods and Applications. 435-469. Ed: By Lukac, R., Plataniotis, K. N. CRC Press, NewYork, London, 2007.

[12] V. Musoko, Biomedical Signal and Image Processing. PhD Thesis. Institute of Chemical Technology in Prague, Czech, 2005.

[13] B. Wicht, Deep Learning Feature Extraction for Image Processing. PhD Thesis. University of Fribourg, Switzerland, 2017.

[14] O. Zapletal, Image Recognition by Convolutional Neural Networks. Master's Thesis. Brno University, Czech, 2017.

[15] Nabiyev V. V., Yapay Zeka. Sözkesen Matbaacılık, Ankara, 2005.

[16] Norouziy, M., Fleety, D., Salakhutdinovy, R., Hamming distance metric learning, Advances in Neural Information Processing Systems (NIPS), Canada, s. 1-9, December 2012.

[17] Goyal, A., Mehta, R., Performance comparison of naïve bayes and j48 classification algorithms. International Journal of Applied Engineering Research. 7(11):1-5, 2012.

[18] Abouelnaga, Y., Ola, S. A., H. Rady, Moustafa, M., CIFAR-10: KNN-based ensemble of classifiers. International Conference on Computational Science and Computational Intelligence (CSCI), California-USA s. 1192-1195, December 2016.

[19] Liu, X., Zhang, R., Meng, Z., Hong, R., Liu, G., On fusing the latent deep CNN feature for image classification. World Wide Web. (Special Issue on Deep vs. Shallow: Learning for Emerging Web-scale Data Computing and Applications). 2018. ArXiv: 1806.00667. 2018.

[20] Li, Y., Bradshaw, J., Sharma, Y., Are generative classifiers more robust to adversarial attacks?.

[21] B. Graham, Fractional max-pooling. ArXiv:1412.6071. 2014.

[22] Springenberg, J. T., Dosovitskiy, Brox, A. T., ve Riedmiller, M., Striving for simplicity: the all convolutional net. ArXiv:1412.6806. 2015.

[23] Mishkin, D., Matas, J., All you need is a good init. International Conference on Learning Representations, US, s. 1-13, May 2016

[24] Xue M., Zhu, C., A study and aplication on machine learning of artificial ıntelligence, International Joint Conference on Artificial Intelligence, Langkawi, Malaysia, s. 272-274, January, 2009.

[25] Khalid, S., Khalil, T., Nasreen, S., A Survey of feature selection and feature extraction techniques in machine learning. Science and Information Conference, London, s.372-378, August, 2014.

[26] Zheng, Y., Vanderbeek, B., Daniel, E., Stambolian, D, Maguire, M., Brainard, D., Gee, J. An automated drusen detection system for classifying age-related macular degeneration with color fundus photographs. IEEE 10th International Symposium on Biomedical Imaging, San Francisco, April, 2013.

[27] Liu, H., Yu, L., Toward ıntegrating feature selection algorithms for classification and clustering. IEER Transactions on Knowledge and Data Engineering.17(4):491-502, 2005.

[28] Uzer, M., S., Örüntü Tanıma Uygulamalarında Yapay Zekâ ve Öznitelik Dönüşüm Metotları Kullanılarak Geliştirilen Öznitelik Seçme Algoritmaları. Doktora Tezi. Selçuk Üniversitesi, Konya, 2014.

[29] A. Özgü, Supervised and Unsupervised Machine Learning Techniques for Text Document Categorization. Master's Thesis. Boğaziçi University, İstanbul, 2004.

[30] Kavzoğlu T., Çölkesen İ., Destek vektör makineleri ile uydu görüntülerinin sınıflandırılmasında kernel fonksiyonlarının etkilerinin incelenmesi. Harita Dergisi. 144, 73-82, 2010.

[31] Yeşilnacar E., Topal, T., Landslide susceptibility mapping: a comparison of logistic regression and neural networks methods in a medium scale study, Hendek region (Turkey). Engineering Geology. 79(3-4): 251-266, 2005.

[32] Fix E., Hodges, J. L., Discriminatory analysis, nonparametric discrimination, consistency properties, Usaf School of Aviation Medicine Randolph Field, Report Number:4, Project No. 21-49-004, 24s.,1951.

[33] Wu, Y., Ianakiev, K., Govindaraju, V., Improved k-nearest neighbor classification. Pattern Recognition. 35, 2311-2318, 2002.

[34] Denoeux, T., A k-nearest neighbor classification rule based on dempster–Shafer Theory. IEEE Transaction on Systems Man Cybernetics. 25(5): 804-813, 1995.

[35] Dudani, S. A., The distance-weight k-nearest neighbor rule. IEEE Trans. Syst. Man. Cybern. SMC-6, 325-327, 1976.

[36] Vapnik, V., The Nature of Statistical Learning Theory. Springer-Verlag, New York, 1995.

[37] Küçüksille, E.U., Ateş, N., Destek vektör makineleri ile yaramaz elektronik postaların filtrelenmesi. Türkiye Bilişim Vakfı Bilgisayar Bilimleri ve Mühendisliği Dergisi. 6(1):1-7, 2013.

[38] Rish, I. An emprical study of the naive bayes. IJCAI ,Workshop on Empirical Methods In Artificial Intelligence. 3(22): 41-46, 2001.

[39] Şahiner, S, En küçük kareler yöntemi ile dogrusal regresyon modeli oluşturmanın prensipleri, Mustafa Kemal Üniversitesi Ziraat Fakültesi Dergisi. 5(1-2): 57-73, 2000.

[40] Montgomery, D. C., Peck, E. A., Vining, G. G., Introduction to Linear Regression Analysis. Wiley, Canada, 2012.

[41] Lakshmi, J. V. N., Stochastic gradient descent using linear regression with python, International Journal of Advanced Engineering Research and Applications (IJA-ERA). 2(8):519-524, 2016.

[42] Anonim, Evaluating model performance - A practical example of the effects of overfitting and data size on prediction, Çevrimiçi. http://www.webcitation.org/70IVVzwMM (Erişim tarihi: 12.12.2017).

[43] Judea, P., Probabilistic Reasoning in Intelligent Systems: Networks of Plausible Inference. Morgan Kaufmann, Los Angeles, 1988.

[44] Cinicioğlu E. N.,Shenoy, P. P.,A new heuristic for learning bayesian networks from limited datasets: a real-time recommendation system application with RFID systems in grocery stores. Annals of Operations Research. 244(2):385-405, 2016.

[45] Yücebaş, S. C., Hipokrat-I: Bayes Ağı Tabanlı Tıbbi Teşhis Destek Sistemi. Yüksek Lisans Tezi. Başkent Üniversitesi.Ankara 2006.

[46] Pearl J. Reverned Bayes on inference engines:A ditributed approach. AAAI-82 Conference, Pittsburgh, s. 133-136, August 1982.

[47] Çakmak, Z., Uzgören N., Keçek, G., Kümeleme analizi teknikleri ile illerin kültürel yapılarına göre sınıflandırılması ve değişimlerinin incelenmesi. Dumlupınar Ünivesitesi Sosyal Bilimler Dergisi. 12:15-36, 2015.

[48] A. C. Günay Atbaş, Kümeleme Analizinde Küme Sayısının Belirlenmesi Üzerine Bir Çalışma. Yüksek Lisans Tezi. Ankara Üniversitesi, Ankara, 2008.

[49] Lin, G. F., Chen, L. H., Identification of homogeneous regions for regional frequency analysis using the self-organizing map. Journal of Hydrology. 324,1-9, 2005.

[50] Najarian, K., Splinter, R., Biomedical Signal and Image Processing. 135. MIT Press, New York, 2005.

[51] Breiman, L., Random forest. Machine Learning. 45(1):5-32. 2001.

[52] Breiman, L., Bagging predictors. Machine Learning. 24(2):123-140. 1996.

[53] Dietterich, T., an Experimental comparison of three methods for constructing ensembles of decision trees: bagging, boosting, and randomization random forest. bagging predictors. Machine Learning. 40(2):139-157. 2000.

[54] Ho, T, K., The random subspace method for constructing decision forests. IEEE Transactions on Pattern Analysis and Machine Intelligence. 20(8):832-844.1998.

[55] Kodinariya, T. M., Makwana, P. R.,Review on determining number of cluster in k-means clustering. Int. J. of Advance Research in Computer Science &Management Studies. 1(6): 90-95, 2013.

[56] Şirin, E., K-Ortalamalar Tekniği (K-Means Clustering) İle Kümeleme: Python Uygulaması.Çevrimiçi.http://www.datascience.istanbul/2017/08/05/kumeleme-notlari-4-k-ortalamalar- python- uygulama/ (Erişim tarihi: 17.08.2018).

[57] G. S. Özcan, Bütünleştirici Modül Ağlarıyla Gen Düzenleme Analizi. Yüksek Lisans Tezi. Başkent Üniversitesi, Ankara, 2014.

[58] Agrawal R., Srikant, R., Fast algorithms for mning association rules. Expanded version available as IBM Research Report RJ9839, In Proc. of the VLDB Conference, Santiago-Chile, s. 487-499, September 1994.

[59] Erce, S., Apriori Algoritması, Çevrimiçi. http://www.webcitation.org/ (Erişim tarihi: 14.04.2018).

[60] Feng, Z., Data Clustering using Genetic Algorithm. Evolutionary Computation: Project Report CSE484, Michigan State University, USA, 2012.

[61] Houck C. R., Joines J. A., ve Kay, M. G., A Genetic Algorithm for Function Optimization: A Matlab Implementation. Technical Report NCSU-IE-TR-95-09, North Carolina State University, Rleigh, 14s., 1995.

[62] Maulik U., Bandyopadhyay, S., Genetic Algorithm-Based Clustering Technique. Pattern Recognition. 33, 1455-1465, 2000.

[63] Tatlıdil, H. Uygulamalı Çok Değişkenli İstatistiksel Analiz. Akademi Matbaası, Ankara, 1996.

[64] Morgan, J. N., Sonquist, J, N., Problems in the analysis of survey data and a proposal. Journal of the American Statistical Association, 58(302):415-484, 1963.

[65] Quinlan, J. R., C4.5: Programs for Machine Learning. 5. Morgan Kaufmann Publishers, California, 1993.

[66] Breiman, L., Friedman, J., Stone C. J., Olshen, R. A., Classification and Regression Trees. 1-15. Chapman&Hall, California, 1984.

[67] Oğuzlar, A., CART analizi ile hanehalkı işgücü anketi sonuçlarının özetlenmesi. Atatürk Üniversitesi İktisadi ve İdari Bilimler Fakültesi Dergisi, 18(3-4):79-90, 2004.

[68] Fayyad, U. M., ve Irani, K. B., Multi Interval Discretization of Continuous-Valued Attributes for Classification Learning. Proceedings of the Thirteenth International Joint Conference on Artificial Intelligence. San Francisco, 1022-1027s., 1993.

[69] Bulut, F., Different mathematical models for entropy in information theory Bilge International Journal of Science and Technology Research. 1(2):167-174, 2017.

[70] Quinlan, J. R., Discovering rules by induction from large collections of examples. 168-201. Ed: by D. Michie, Expert systems in the micro electronic age. Edinburgh University Press, Edinburg, 1979.

[71] Ahire, P. G., Kolhe, S., Kirange, K., Implementation improved ID3 algorithm to obtain more optimal decision tree. International Journal of Engineering Research and Development. 11(2):44-47, 2015.

[72] Jin, C., De-lin, L., Fen-xiang, M., An Improved ID3 Decision Tree Algorithm, Proceedings of 2009 th International Conference on Computer on Computer Science & Education ICCSE'09, London, July 2009.

[73] Adhatrao, K., Gaykar, A., Dhawan, A., Jha, R., Honrao, V., Predicting Students' Performance Using ID3 and C4.5 Classification Algorithms. International

Journal of Data Mining & Knowledge MAnagement Process (IJDKP). 3(5):39-52, 2013.

[74] Witten I. H., Frank, E., Data mining: practical machine learning tools and techniques. Morgan Kaufmann, San Francisco, 2005.

[75] Dai W., Ji, W., MapReduce ımplementation of C4.5 decision tree algorithm. International Journal of Database Theory and Application. 7(1):49-60, 2014.

[76] Anonim, Sinir sistemi nedir?, Çevrimiçi. http://www.webcitation.org/70IVaZtWV (Erişim tarihi: 15.10.2017).

[77] Borglin, J., Classification of hand movements using multi-channel EMG. Master's Thesis. Chalmers University of Technology, Sweden, 2011.

[78] Arbib, M. A., Brains, Machines, and Mathematics, Springer-Verlag, New York, 1987.

[79] Öztemel, E., Yapay Sinir Ağları, Papatya Yayıncılık, İstanbul, 2012.

[80] Maltarollo, V. G., Honorio K. M., Ferreira da Silva, A. B., Aplications of artificial neural networks chemicial problems. 203-223. Ed: Kenji Suzuki, Artificial neural networks-architectures and applications E-book, America, 2013.

[81] Rosenblatt, F., A Perceiving and recognizing automaton, Cornell Aeronautical Laboratory, Inc. Rapor No: 85-460-1, 33s., 1957.

[82] Serhatllıoğlu S., Hardalaç, F., Yapay zeka teknikleri ve radyolojiye uygulaması, Fırat Tıp Dergisi, 14(1):1-6, 2009.

[83] S. B. Driss, M. Soua, R. Kachouri ve M. Akil, A comparison study between MLP and convolutional neural network models for character recognition, SPIE

Conference on Real-Time Image and Video Processing, United States, April 2017.

[84] Ataseven, B., Yapay sinir ağları ile öngörü modellemesi, Öneri, 10(39):101-115, 2013.

[85] Rojas, R., Neural Networks - A Systematic Introduction. 29. Springer-Verlag, New York, 1996.

[86] Anonim, Overfitting in machine learning: What it is and how to prevent it, Çevrimiçi. http://www.webcitation.org /70IVu0tGF (Erişim tarihi: 01.02.2018).

[87] Aronsson, D., Deep learning made Accessible, Çevrimiçi). http://www.webcitation.org/70IW4mDDS (Erişim tarihi: 15.04.2018).

[88] Bozarık, E. https://medium.com/deep-learning-turkiye/sinir-a%C4%9Flar%C4%B1-ve-derin-%C3%B6%C4%9 Frenme-v-grandyan-d%C3%BC%C5%9F%C3%BC%C5%9F%C3%BC-8c6d15a3d965 (Erişim tarihi:29.09.2018).

[89] Anonim, Backpropagation in convolutional neural networks, Çevrimiçi. http://www.webcitation.org/70IW8UJqK (Erişim tarihi: 10.02.2018).

[90] Taşgetiren, F., Çok katmanlı yapay sinir ağlar, Çevrimiçi. http://www.elektrik.gen.tr/2015/08/cok-katmanli-yapay-sinir-aglar/549 (Erişim tarihi: 16.03.2018).

[91] Macêdo, D., Zanchettin, C., Ludermir, T., Simple fast convolutional feature learning. ICLR, Canada, May 2018.

[92] CS231n Convolutional Neural Network for Visual Recognition, Çevrimiçi. http://CS231n.github.io/neural-networks-3/. (Erişim tarihi: 01.04.2018).

[93] Valia, A. S., Activation functions and it's types-Which is better?, https://towardsdatascience.com/activation-functions-and-its-types-which-is-better-a9a5310cc8f (Erişim tarihi: 02.10.2018).

[94] Agarap, A. F. M., Deep learning using Rectified Linear Units (ReLU). arXiv:1803.08375v1. 2018.

[95] Romanuke, V. V., Appropriate number of standard 2 × 2 max pooling layers and their allocation in convolutional neural networks for diverse and heterogeneous datasets. Information Technology and Management Science. 20(1):12-19, 2017.

[96] Srivastava, N., Hinton, G., Krizhevsky, A. Sutskever I., Salakhutdinov, R., Dropout: a simple way to prevent neural networks from overfitting. Journal of Machine Learning Research. 6 (14) :1929-1958, 2014.

[97] İnik Ö., Ülker, E., Derin Öğrenme ve görüntü analizinde kullanılan Derin Öğrenme modelleri. Gaziosmanpaşa Bilimsel Araştırma Dergisi. 6(3):85-104, 2017.

[98] Anonim, What is a convolutional neural network?, Çevrimiçi. https://www.mathworks.com/solutions/deep-learning/convolutional-neural-network.html. Erişim tarihi: 21.12.2017).

[99] Radiuk, P. M., Impact of training set batch size on the performance of convolutional neural networks for diverse datasets. Information Technology and Management Science. 20, 20-24, 2017.

[100] Bengio, Y., Deep learning of representations for unsupervised and transfer learning. JMLR: Workshop and Conference Proceedings. 27:17-3, 2012.

[101] Krizhevsky, A., https://www.cs.toronto.edu/~kriz/cifar.html. Çevrimiçi.. (Erişim tarihi: 23.01.2018).

[102] Kızrak A., Yapay Zeka ve Derin Öğrenmeye Başlama Rehberi, Çevrimiçi. http://www.webcitation.org/mainframe.php (Erişim tarihi: 02.05.2018).

[103] Ergül, E., Mevkisel ve Anlamsal Göreceli Nitelikler Yardımıyla Görüntü Tanıma. Doktora Tezi. Kocaeli Üniversitesi, Kocaeli, 2016.

[104] Goyal, A., Mehta, R., Performance comparison of naïve Bayes and J48 classification algorithms. International Journal of Applied Engineering Research. 7(11), 2012.

[105] Cournapeau, D., http://scikit-learn.org/stable/modules/generated/sklearn.linear_model.LogisticRegression.html. (Erişim Tarihi: 02.11.2018).

[106] Cournapeu, D., http://scikit-learn.org/stable/modules/generated/sklearn.neighbors.KNeighborsClassifier.html. (Erişim Tarihi: 02.11.2018).

[107] Johnson, J., Karpathy, A., http://cs231n.github.io/classification/. (Erişim tarihi: 30.10.2018)

[108] Cournapeu, D., http://scikit-learn.org/stable/modules/generated/sklearn.naive_bayes.GaussianNB.html. (Erişim Tarihi: 02.11.2018).

[109] Cournapeu, D., http://scikit-learn.org/stable/modules/generated/sklearn.tree.DecisionTreeRegressor.html. (Erişim Tarihi: 02.11.2018).

[110] Cournapeu, D., http://scikit-learn.org/stable/modules/generated/sklearn.tree.DecisionTreeClassifier.html. (Erişim Tarihi: 02.11.2018).

[111] Krizhevsky, A., Sutskever, I., and Hinton, G. E., Imagenet classification with deep convolutional neural networks. Neural Information Processing Systems (NIPS). 1097–1105, 2012.