Sosyal Bilimler Enstitüsü

T.C.
MARMARA ÜNİVERSİTESİ
SOSYAL BİLİMLER ENSTİTÜSÜ
EKONOMETRİ ANABİLİM DALI
İSTATİSTİK BİLİM DALI

GENETİK ALGORİTMA VE ALTERNATİF YÖNTEMLERLE DERİN ÖĞRENME MODELLERİNDE GİZLİ KATMAN VE NÖRON SAYILARININ ANALİZİ

Doktora Tezi

KUBİLAY ERİŞLİK

İstanbul, 2024

T.C.
MARMARA ÜNİVERSİTESİ
SOSYAL BİLİMLER ENSTİTÜSÜ
EKONOMETRİ ANABİLİM DALI
İSTATİSTİK BİLİM DALI

GENETİK ALGORİTMA VE ALTERNATİF YÖNTEMLERLE DERİN ÖĞRENME MODELLERİNDE GİZLİ KATMAN VE NÖRON SAYILARININ ANALİZİ

Doktora Tezi

KUBİLAY ERİŞLİK

Danışman: PROF. DR. DİLEK ALTAŞ KARACA

İstanbul, 2024

T.C.
MARMARA ÜNİVERSİTESİ
SOSYAL BİLİMLER ENSTİTÜSÜ

TEZ ONAY BELGESİ

Ekonometri Anabilim Dalı İstatistik Bilim Dalı ~~Yüksek Lisans~~ / Doktora öğrencisi Kubilay ERİŞLİK'in Genetik Algoritma ve Diğer Yöntemlerle Derin Öğrenme Modellerinde Gizli Katman ve Nöron Sayılarının Analizi adlı tez çalışması, Enstitümüz Yönetim Kurulunun 17.01.204 tarih ve 2024-3/19 sayılı kararıyla oluşturulan jüri tarafından oy birliği / ~~oy çokluğu~~ ile ~~Yüksek Lisans~~ / Doktora Tezi olarak kabul edilmiştir.

Tez Savunma Tarihi: 31/01/2024

Öğretim Üyesi Adı Soyadı			**İmzası**
1	Tez Danışmanı	Prof. Dr. Dilek ALTAŞ KARACA	
2	Jüri Üyesi	Prof. Dr. Ahmet Mete ÇİLİNGİRTÜRK	
3	Jüri Üyesi	Prof. Dr. Ünal Halit ÖZDEN	
4	Jüri Üyesi	Prof. Dr. Selay GİRAY YAKUT	
5	Jüri Üyesi	Prof. Dr. Münevver TURANLI	

ÖZET

Bu çalışma, derin öğrenme modellerinde gizli katman ve nöron sayılarının belirlenmesi konusunu ele almaktadır. Çalışmanın odak noktası, bu gizli katman ve nöron sayılarının belirlenmesi için genetik algoritma ve alternatif yöntemlerin incelenmesidir. Çalışma, genetik algoritma tarafından önerilen yöntemle diğer alternatif yöntemler arasında bir karşılaştırma yaparak bu modeller arasındaki farklılıkları analiz etmeyi amaçlamaktadır. Tezin teorik temelini, derin öğrenme ve genetik algoritma oluşturmaktadır. Derin öğrenme model yapıları, gizli katmanlar ve nöron sayıları hakkındaki temel kavramlar, detaylı bir şekilde incelenmiştir. Diğer taraftan çalışmada, genetik algoritma ve alternatif yöntemlerin teorik çerçevesi, bu algoritmaların nasıl çalıştığı ve derin öğrenme modellerindeki parametre optimizasyonu için nasıl kullanılabileceği açıklanmaktadır.

50.000 gözlemli teorik verilerle yapılan çalışmada genetik algoritmanın en düşük MSE değeri ile öne çıktığı görülmektedir. Gerçek verilere göre ise, genetik algoritmanın yine en düşük MSE değerine sahip olduğu, Heaton tarafından önerilen kuralların ise ikinci sırada yer aldığı gözlemlenmektedir.

Gerçek verilere göre elde edilen sonuçlar da benzer bir sıralamayı ortaya koymaktadır, ancak Huang tarafından önerilen tek katmanlı ağın teorik verilere göre daha iyi performans gösterdiği, çift katmanlı ağın ise daha düşük sonuçlar verdiği dikkat çekmektedir. Bu sonuçlar, teorik ve gerçek veriler arasında genetik algoritmanın istikrarlı performansını vurgulamaktadır.

Teorik ve gerçek verilere ait Kruskal-Wallis testleri, yöntemler arasında istatistiksel olarak anlamlı farklar olduğunu göstermektedir. Genetik algoritma, teorik ve gerçek verilere göre en düşük MSE değeri ile öne çıkmıştır. İstatistiksel anlamda benzer performans gösteren diğer yöntemler arasında ise Heaton kuralları, Faktör Analizi ve Huang'ın tek katmanlı ağ modeli yer almaktadır.

Sonuç olarak, bu çalışma derin öğrenme modellerinin uygulanmasında genetik algoritma, Heaton kuralları, Faktör Analizi ve Huang'un çift/tek katmanlı ağlarını karşılaştırmış ve genetik algoritmanın genelde en iyi performansı sergilediğini göstermiştir. Ancak, yöntemlerin performanslarının teorik ve gerçek veriler arasında bazı farklılıklar gösterdiği görülmektedir. Bu çalışma, gelecekteki araştırmalar için geniş bir perspektif sunacak ve derin öğrenme modellerinin hiper parametrelerinin belirlenmesinde kullanılan yöntemlerin etkisinin daha iyi anlaşılmasına katkı sağlayacaktır.

Anahtar Kelimeler: Derin Öğrenme, Genetik Algoritma, Optimizasyon, Yapay Sinir Ağları

ABSTRACT

This paper addresses the issue of determining the number of hidden layers and neurons in deep learning models. The focus of the study is to examine the genetic algorithm and alternative methods to optimize these parameters. The study aims to provide a comparison between the method proposed by the genetic algorithm and other alternative methods, analyzing their impact on the architecture of deep learning models. The theoretical basis of the thesis focuses on deep learning, genetic algorithm. The basic concepts of deep learning model structures, hidden layers and neuron counts are analyzed in detail. The theoretical framework of the genetic algorithm and alternative methods explains how these algorithms work and how they can be used for parameter optimization in deep learning models.

With 50000 observations of theoretical data, the genetic algorithm stands out with the lowest MSE value. According to real data, it is observed that the genetic algorithm has the lowest MSE value again, while the rules proposed by Heaton are in second place.

The results based on real data show a similar ranking, but it is noteworthy that the single-layer network proposed by Huang outperforms the theoretical data, while the double-layer network gives inferior results. These results emphasize the consistent performance of the genetic algorithm between theoretical and real data.

Kruskal-Wallis tests for the theoretical and real data show that there are statistically significant differences between the methods. The genetic algorithm stands out with the lowest MSE value compared to the theoretical and real data. Other methods with statistically similar performance include Heaton rules, factor analysis and Huang single layer network.

In conclusion, this study compares genetic algorithm, Heaton's rules, factor analysis and Huang's double/single layer network in the application of deep learning models and shows that genetic algorithm generally performs the best. However, the performance of the methods showed some differences between theoretical and real data. This study provides a broad perspective for future research and contributes to our further understanding of the impact of methods in determining the hyperparameters of deep learning models.

Keywords: Deep Learning, Genetic Algorithm, Optimization, Artificial Neural Network

İÇİNDEKİLER

ÖZET ..... **i**

ABSTRACT ..... **ii**

İÇİNDEKİLER ..... **iii**

TABLOLAR LİSTESİ ..... **vii**

ŞEKİLLER LİSTESİ ..... **viii**

KISALTMALAR ..... **x**

TEŞEKKÜR ..... **xi**

1. GİRİŞ **1**

2. DERİN ÖĞRENME **4**

2.1. Derin Öğrenme Kavramı 5

2.2. Derin Öğrenme Alanında Bibliyometrik Analiz 5

2.3. Derin Öğrenmenin Uygulama Alanları 12

2.3.1. Görüntü ve Ses Tanıma 12

2.3.2. Doğal Dil İşleme 13

2.3.3. Oyun Sektörü 14

2.3.4. Sağlık Sektörü 14

2.3.5. Otomotiv Sektörü 14

2.3.6. Finans Sektörü 15

2.3.7. Perakende ve E-Ticaret 15

2.4. Derin Öğrenmenin Yapısı ve Temel Elemanları 16

2.4.1. Derin Öğrenmenin Yapısı 17

2.4.1.1. Girdiler 17

2.4.1.2. Ağırlıklar 18

2.4.1.3. Toplama Fonksiyonu 18

2.4.1.4. Aktivasyon Fonksiyonu 19

2.4.1.5. Çıktılar 21
2.4.2. Derin Öğrenmenin Temel Elemanları 21
2.4.2.1. Girdi Katmanı 21
2.4.2.2. Gizli Katman 22
2.4.2.3. Çıktı Katmanı 22
2.5. Derin Öğrenme Türleri 23
2.5.1. Derin İleri Beslemeli Ağlar 23
2.5.2. Derin Geri Beslemeli Ağlar 24
2.6. Gizli Katman ve Nöron Sayısının Belirlenmesi 24
2.6.1. Huang Önerisi 25
2.6.2. Heaton Önerisi 26
2.6.3. Temel Bileşenler Analizi 27
2.6.4. Genetik Algoritma 28
3. GENETİK ALGORİTMA 29
3.1. Genetik Algoritma Kavramı 30
3.2. Genetik Algoritma Alanında Bibliyometrik Analiz 31
3.3. Genetik Algoritma Çeşitleri 37
3.3.1. Klasik Genetik Algoritma 37
3.3.1.1. Genetik Operatörler 39
3.3.1.2. Kodlama Şeması 40
3.3.1.3. Seçim Teknikleri 41
3.3.1.4. Çaprazlama Operatörleri 43
3.3.1.5. Mutasyon 46
3.3.2. Gerçek ve İkili Kodlu Genetik Algoritmalar 48
3.3.3. Çok Amaçlı Genetik Algoritmalar 48
3.3.4. Paralel Genetik Algoritmaları 49
3.3.5. Kaotik Genetik Algoritmalar 50

3.3.6. Karma Genetik Algoritmalar 50
3.4. Genetik Algoritmanın Uygulama Alanları 51
3.4.1. Operasyon Yönetimi 52
3.4.1.1. Tesis Yerleşim Planı 52
3.4.1.2. Çizelgeleme 52
3.4.1.3. Envanter Kontrolü 53
3.4.1.4. Tahmin ve Ağ Tasarımı 53
3.4.2. Multimedya 53
3.4.2.1. Bilgi Güvenliği 53
3.4.2.2. Görüntü İşleme 53
3.4.2.3. Video İşleme 54
3.4.2.4. Tıbbi Görüntüleme 54
3.4.2.5. Hassas Tarım 54
3.4.2.6. Oyun 54
3.4.3. Kablosuz Ağlar 55
3.4.3.1. Yük Dengeleme 55
3.4.3.2. Yerelleştirme 55
3.4.3.3. Bant Genişliği ve Kanal Tahsisi 55
4. TEORİK VERİLER VE EMLAK SEKTÖRÜ ÜZERİNE UYGULAMA 56
4.1. Araştırmanın Amacı 56
4.2. Araştırmanın Kapsamı 56
4.3. Araştırma Yöntemi 56
4.4. Gerçek ve Teorik Verilerin Elde Edilmesi 57
4.4.1. Gerçek Verilerin Elde Edilmesi 57
4.4.2. Teorik Verilerin Elde Edilmesi 57
4.5. Gerçek Verilere Ait Tanımsal İstatistikler 58
4.6. Analiz ve Bulgular 62

4.6.1. Gerçek Veri Analizi 63

4.6.2. Teorik Veri Analizi 73

SONUÇ 83

KAYNAKÇA 87

EKLER 93

Ek – 1 Teorik Verilere Ait Sonuçlar 93

Ek – 2 Gerçek Verilere Ait Sonuçlar 94

Ek – 3 Web Scrapping Kodları 95

Ek – 4 Derin Öğrenme ve Genetik Algoritma Kodları 101

TABLOLAR LİSTESİ

Tablo 1. Yayın Tipi (Derin Öğrenme) 6

Tablo 2. Alternatif Toplama Fonksiyonları 19

Tablo 3. Yayın Tipi (Genetik Algoritma) 32

Tablo 4. Farklı Kodlama Şemalarının Karşılaştırılması 40

Tablo 5. Farklı Şeçim Tekniklerinin Karşılaştırılması 42

Tablo 6. Farklı Çaprazlama Operatörlerinin Karşılaştırılması 44

Tablo 7. Farklı Mutasyon Operatörlerinin Karşılaştırılması 47

Tablo 8. Optimum Şartlar Altında Çeşitli Operatörlerin En İyi Kombinasyonu 47

Tablo 9. Değişkenlere Ait Frekans Dağılımı 59

Tablo 10. Fiyat, Brüt m^2, Net m^2 ve Aidat için Tanımlayıcı İstatistikler 62

Tablo 11. Yöntemlere Göre Ortalama Sonuçlar 66

Tablo 12. Kruskal Wallis Testi 68

Tablo 13. İkili Karşılaştırma Sonuçları 69

Tablo 14. Yöntemlere Ait Ranklar 70

Tablo 15. Kruskal Wallis Testi 71

Tablo 16. İkili Karşılaştırma Sonuçları 72

Tablo 17. Yöntemlere Ait Ranklar 73

Tablo 17. Yöntemlere Göre Ortalama Sonuçlar 76

Tablo 18. Kruskal Wallis Testi 78

Tablo 19. İkili Karşılaştırma Sonuçları 79

Tablo 20. Yöntemlere Ait Ranklar 80

Tablo 21. Mann Whitney U Testi 81

ŞEKİLLER LİSTESİ

Şekil 1. Yapay Zekâ, Makine Öğrenmesi ve Derin Öğrenme 4
Şekil 2. Kategorilere Göre Yayın Dağılımları (Derin Öğrenme 6
Şekil 3. Yıllara Göre Yayın Sayıları (Derin Öğrenme) 7
Şekil 4. Anahtar Kelime Sayısına Göre Yayın Sayısı Dağılımı (Derin Öğrenme) 8
Şekil 5. Anahtar Kelimelere Ait Sosyal Ağ Analizi (Derin Öğrenme) 8
Şekil 6. Yazarlara Ait Sosyal Ağ Analizi (Derin Öğrenme) 9
Şekil 7. Yazarların Almış Oldukları Atıflara Ait Sosyal Ağ Analizi (Derin Öğrenme) 10
Şekil 8. Yayınların Almış Oldukları Atıflara Göre Sosyal Ağ Analizi (Derin Öğrenme) 11
Şekil 9. Kullanılan Referanslara Ait Sosyal Ağ Analizi (Derin Öğrenme) 12
Şekil 10. Derin Öğrenme Yapısı ve Temel Elemanları 17
Şekil 11. Metasezgisel Algoritmalar (Katoch, Chauhan ve Kumar, 2021) 29
Şekil 12. Kategorilere Göre Yayın Dağılımları (Genetik Algoritma) 32
Şekil 13. Yıllara Göre Yayın Sayıları (Genetik Algoritma) 33
Şekil 14. Anahtar Kelime Sayısına Göre Yayın Sayısı Dağılımı (Genetik Algoritma) 33
Şekil 15. Anahtar Kelimelere Ait Sosyal Ağ Analizi (Genetik Algoritma) 34
Şekil 16. Yazarlara Ait Sosyal Ağ Analizi (Genetik Algoritma) 35
Şekil 17. Yazarların Almış Oldukları Atıflara Ait Sosyal Ağ Analizi (Genetik algoritma) 35
Şekil 18. Yayınların Almış Oldukları Atıflara Göre Sosyal Ağ Analizi (Genetik Algoritma) ... 36
Şekil 19. Kullanılan Referanslara Ait Sosyal Ağ Analizi (Genetik Algoritma) 37
Şekil 20. Genetik Algoritma Akış Diyagramı 39
Şekil 21. Genetik Algoritmada Kullanılan Operatörler (Katoch vd., 2021) 40
Şekil 22. Karma Yöntemlerin Genetik Algoritmalarda Uygulanma Nedenleri 51
Şekil 23. Genetik Algoritmanın Uygulama Alanları 52
Şekil 24. İlçelere Göre Ortalama Satılık Ev Fiyatları 58
Şekil 25. İlçelere Göre Satılık Ev Sayısı 59

Şekil 26. Huang Tek Katmanlı Derin Öğrenme Analizi Sonuç Dağılımı 63
Şekil 27. Huang Çift Katmanlı Derin Öğrenme Analizi Sonuç Dağılımı 64
Şekil 28. Heaton Kuralları Derin Öğrenme Analizi Sonuç Dağılımı 64
Şekil 29. Faktör Analizi Derin Öğrenme Sonuç Dağılımı 65
Şekil 30. Genetik Algoritma Derin Öğrenme Sonuç Dağılımı 66
Şekil 31. Yöntemlerin Karşılaştırılması 67
Şekil 32. Yöntemlerin MSE Değerlerine Ait Kutu Grafiği 68
Şekil 33. Yöntemlerin Süre Değerlerine Ait Kutu Grafiği 71
Şekil 34. Huang Tek Katmanlı Derin Öğrenme Analizi Sonuç Dağılımı 73
Şekil 35. Huang Çift Katmanlı Derin Öğrenme Analizi Sonuç Dağılımı 74
Şekil 36. Heaton Kuralları Derin Öğrenme Analizi Sonuç Dağılımı 75
Şekil 37. Faktör Analizi Derin Öğrenme Sonuç Dağılımı 75
Şekil 38. Genetik Algoritma Derin Öğrenme Sonuç Dağılımı 76
Şekil 39. Yöntemlerin Karşılaştırılması 77
Şekil 40. Yöntemlerin MSE Değerlerine Ait Kutu Grafiği 78
Şekil 41. Faktör Analizi ve Heaton Yöntemine Ait Süre Dağılımları 81

KISALTMALAR

WOS:	Web of Science
MSE:	Mean Square Error (Ortalama Hata Kareleri)

TEŞEKKÜR

Bu uzun soluklu süreçte bana desteklerini hiçbir zaman esirgemeyen, yol gösteren değerli hocam Sayın Prof. Dr. Dilek ALTAŞ KARACA; her zaman sorularıma sabırla cevap veren, doktora sürecimde her koşulda yanımda olan hocalarım Prof. Dr. Ahmet Mete ÇİLİNGİRTÜRK, Prof. Dr. Esen YILDIRIM, Prof. Dr. Ali KÖSE, Prof. Dr. Selay GİRAY YAKUT ve Doç. Dr. Özlem ERGÜT; hayatımın her aşamasında beni destekleyen ve yol gösteren Sayın Prof. Dr. Münevver TURANLI başta olmak üzere lisans öğrenciliğim ve beraber çalışma fırsatı yakaladığım dönemden itibaren her zaman yanımda olan hocalarım Prof. Dr. Ünal Halit ÖZDEN, Prof. Dr. Özlem DENİZ BAŞAR ve Doç. Dr. Seda BAĞDATLI KALKAN; değerli jüri üyelerine; her zorlukta yanımda olan oda arkadaşlarım Arş. Gör. Gurur Öyküm GÜVEN ve Arş. Gör. Deniz GERÇEKER; beni her zaman destekleyen aileme teşekkürü borç bilirim.

Bu süreçte sizlerin desteği ve rehberliği olmadan bu noktaya gelemezdim. Sizlerin katkılarıyla tamamlanan bu çalışma, sizlerin öğretileri ve desteğiyle zenginleşti. İlginiz, anlayışınız ve emeğiniz için içten teşekkür ederim.

1. GİRİŞ

Günümüzde derin öğrenme, yapay zekâ alanında önemli bir yer edinmiş, karmaşık veri setlerinin analizi, örüntü tanıma, tahminleme ve sınıflandırma gibi birçok uygulama alanında başarıyla kullanılan güçlü teknolojiler arasında yer almaktadır. Derin öğrenme modelleri, gizli katmanlar ve nöron sayıları gibi temel yapı taşları üzerine kurulmuş, ancak bu yapıların optimal konfigürasyonu konusunda halen birçok soru işareti bulunmaktadır. Çalışmanın giriş bölümünde, derin öğrenme genel olarak incelenmiş ve daha sonra bu alandaki temel kavramlar yani gizli katmanlar ve nöron sayılarının belirlenmesindeki temel zorluklar ele alınmıştır.

Derin öğrenme uygulamalarında karşılaşılan en büyük zorluklardan biri, gizli katmanların ve nöron sayılarının belirlenmesidir. Bu parametrelerin doğru bir şekilde seçilmemesi, modelin performansını etkileyebilir ve doğru sonuçlara ulaşmayı engelleyebilir. Bu nedenle çalışma, bu temel sorunlara çözüm getirme amacını taşımaktadır. Derin öğrenmenin geniş veri setleri üzerindeki etkili kullanımının daha iyi anlaşılabilmesi ve bu konudaki boşlukların doldurulabilmesi için bu parametrelerin optimize edilmesi gerekmektedir.

Derin öğrenmede kaç tane gizli katman olması gerektiği ve her katmanda kaç tane nöron kullanılması gerektiği, derin öğrenme uygulamaları sırasında cevaplanmakta zorlanılan konulardır. İlk derin öğrenme uygulamalarının üzerinden 30 yıldan fazla bir zaman geçmesine rağmen bu iki konuya günümüzde de net bir cevap verilmesi mümkün değildir (Mas ve Flores, 2008).

Bu çalışmanın amacı, yukarıda da belirtildiği üzere, günümüzde hala net olarak cevaplanamamış bu iki konu için sunulmuş yöntemleri incelemek, gerçek ve teorik veriler üzerinde sunulmuş yöntemleri karşılaştırarak en optimum sürede en az hata ile ağı oluşturabilecek gizli katman ve nöron sayısını belirleyecek yöntemi tespit etmektir.

Çalışmanın hedefleri arasında, literatürdeki yöntemlerin ayrıntılı bir analizi, gerçek ve teorik veriler üzerinde bu yöntemlerin karşılaştırılması, en iyi parametre konfigürasyonunun tespiti ve bu alandaki bilimsel ve uygulamaya yönelik katkıları bulunmaktadır.

Bu çalışma, derin öğrenme modellerinde gizli katmanların ve nöron sayılarının belirlenmesinde kullanılan mevcut yöntemleri sistematik bir şekilde incelemeyi ve bu yöntemleri gerçek ve teorik veriler üzerinde karşılaştırmayı içermektedir. Çalışma, belirli bir sıra ve zaman çizelgesine göre yürütülmekte, her aşamada elde edilen sonuçlar dikkate alınarak analiz edilmekte ve en sonunda bir yöntem önerisi sunmaktadır.

Araştırmada, gerçek veriler olarak İstanbul ili bazında Ekim 2022 yılında sahibinden.com üzerinden elde edilen 65726 adet satılık eve ait veriler kullanılmıştır. Teorik veriler olarak normal dağılımdan türetilmiş 50000 gözlemli bir veri seti elde edilmiştir. Çalışmada kullanılan veriler, her alana uyumluluk sağlayamayacağı için elde edilen sonuçların genellenmesi konusunda bazı sınırlamaların yapılması gerekmektedir. Veri seti boyutu, kullanılan algoritmanın sınırlamaları ve diğer faktörler, çalışmanın genel kapsamını ve derin öğrenme alanındaki hızla değişen teknolojik gelişmeler ise çalışmanın sonuçlarının geçerliliğini etkileyebilmektedir.

Bu araştırma, derin öğrenme modellerinde gizli katmanların ve nöron sayılarının belirlenmesindeki zorluklara ve sorunlara yönelik kapsamlı bir analiz sunarak, bu alandaki bilgi birikimine katkıda bulunmayı hedeflemektedir. Elde edilecek sonuçlar, derin öğrenme alanındaki araştırmalara yeni bir bakış açısı kazandırmakta ve uygulamada daha etkili modellerin geliştirilmesine katkı sağlamaktadır.

Çalışmanın tamamlanması sonucunda, derin öğrenme modellerinde gizli katmanların ve nöron sayılarının belirlenmesinde daha etkili ve optimize edilmiş bir yöntemin ortaya çıkması beklenmektedir. Bu sonuç, bilimsel literatürde yeni bir bakış açısı sunabilecek ve derin öğrenme modellerinin daha iyi anlaşılmasına ve kullanılmasına katkıda bulunacaktır. Bu çalışmanın önemi, derin öğrenme modellerinin performansını artırmak ve daha geniş bir uygulama yelpazesine yaymak isteyen araştırmacılara ve profesyonellere yeni bir çerçeve sunmasıdır.

İstanbul, sadece coğrafi konumuyla değil, aynı zamanda tarihi ve kültürel zenginlikleriyle de dünya genelinde benzersiz bir şehir olarak kabul edilmektedir. Boğaz Köprüsü ile iki kıtayı birleştiren ve tarihi dokusuyla göz kamaştıran İstanbul, tarihi İpek Yolu üzerinde stratejik bir konumda yer almaktadır. Bu stratejik konum, şehre hem Asya hem de Avrupa kültürlerini ve etkileşimini bir araya getirme özelliği kazandırmaktadır.

İstanbul, sadece Türkiye'nin değil, aynı zamanda bölgesinin ekonomik bir gücüdür. Türkiye'nin finans, ticaret, sanayi ve turizm alanlarında lider bir şehri olarak, İstanbul'un ekonomik büyümesi son yıllarda dikkat çekici bir ivme kazanmaktadır. Bu büyüme, şehrin emlak sektörünü de olumlu bir şekilde etkilemektedir. Yüksek istihdam potansiyeli, şehirdeki iş olanakları ve büyüyen ekonomi, İstanbul'u hem yerli hem de uluslararası yatırımcılar için cazip kılmaktadır.

Nüfus artışı da İstanbul'daki emlak piyasasını şekillendiren önemli faktörler arasında yer almaktadır. Hızla büyüyen nüfus, konut talebini artırmakta ve şehirdeki gayrimenkul projelerini teşvik etmektedir. Özellikle genç nüfusun şehre olan göçü, konut talebinde artışa neden olmakta, böylece yeni konut projelerinin ortaya çıkmasına yol açmaktadır.

Emlak piyasasında, bir evin fiyatını belirleyen faktörler arasında fiziksel özellikler önemli bir yer tutar. Evlerin fiziksel özellikleri, alıcıların tercihleri üzerinde belirleyici bir etkiye sahiptir. Bu kapsamda, evin büyüklüğü, oda sayısı, banyo sayısı, balkon gibi unsurlar, alıcıların satın alma kararlarını etkileyen önemli faktörler arasında yer almaktadır.

Bu çalışmada, İstanbul'daki emlak piyasasını analiz etmek ve ev fiyatlarını tahmin etmek amacıyla derin öğrenme modelinin tasarlanması amaçlanmaktadır. Model, geniş ve detaylı bir veri seti üzerinde eğitilmiş olup, evlerin fiziksel özellikleri ile fiyatları arasındaki ilişkiyi anlamak için güçlü bir analitik araç sunmaktadır.

Uygulama için kullanılan 2022 yılı veri seti, İstanbul genelindeki satılık konut ilanlarını kapsayan geniş bir örnekleme sahiptir. Bu zengin veri seti, çeşitli mahallelerdeki konut piyasasının dinamiklerini anlamak ve kullanıcılara güvenilir fiyat tahminleri sunmak amacıyla kullanılmaktadır. Modelin kullanıcı dostu bir arayüzle entegre edilmesi, potansiyel ev alıcılarının ve satıcılarının daha bilinçli kararlar vermelerine katkı sağlayacaktır.

Tezin birinci bölümü giriş bölümü olup, bu bölümde çalışmanın amacına ve önemine yer verilmiştir.

Tezin ikinci bölümünde, derin öğrenme kavramı ve derin öğrenme ile ilgili temel kavramlar açıklanmakta ve daha sonra derin öğrenmede kullanılan veri ve yöntemler ayrıntılı olarak incelenmektedir. Ayrıca bu bölümde bibliyometrik analiz ile derin öğrenme konusunda literatür taramasına yer verilmiştir.

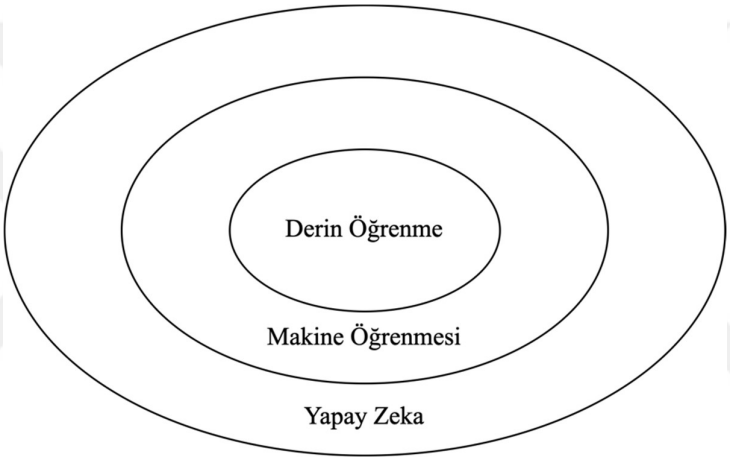
Üçüncü bölümünde, genetik algoritma kavramı ve genetik algoritma ile ilgili temel kavramlar açıklanmakta ve daha sonra genetik algoritmada kullanılan veri ve yöntemlerden ayrıntılı olarak bahsedilmektedir. Ayrıca genetik algoritma konularında bibliyometrik analiz ile elde edilen bulgular incelenmiştir.

Dördüncü bölümünde ise emlak sektörüne ait tanımlayıcı istatistikler yer almakta, teorik ve gerçek veriler üzerinde gerçekleştirilen analizlere ait karşılaştırmalar incelenmektedir.

Sonuç bölümünde ise çalışmanın genel bir değerlendirilmesi yapılarak, sonraki çalışmalar için önerilerde bulunulmuştur.

2. DERİN ÖĞRENME

Derin öğrenme, örnekten öğrenme fikrine dayanan makine öğrenmesi adı verilen daha genel bir yapay zekâ alanının bir alt kümesidir. Yapay zekâ, makine öğrenmesi ve derin öğrenmenin hiyerarşik yapısı Şekil 1'de verilmiştir. Makine öğrenmesinde, bir bilgisayar sorunu çözmek için devasa bir kurallar listesi öğrenmek yerine, ona verilen örnekleri değerlendirebileceği bir model ve hata yaptığında modeli değiştirmesi için küçük bir talimat verilmektedir. Zaman içinde, uygun bir modelin sorunu son derece doğru bir şekilde çözmesi beklenmektedir.



Şekil 1. Yapay Zekâ, Makine Öğrenmesi ve Derin Öğrenme

Bir veri modeline kaç katmanın katkıda bulunduğu modelin derinliği olarak adlandırılır. Modern derin öğrenme genellikle onlarca hatta yüzlerce ardışık temsil katmanı içermekte ve bunların hepsi eğitim verilerine uygulanarak otomatik olarak eğitilmektedir. Bu arada, makine öğrenimine yönelik diğer yaklaşımlar, verilerin yalnızca bir veya iki temsil katmanını öğrenmeye odaklanma eğilimindedir; bu nedenle, bazen sığ öğrenme olarak adlandırılırlar (Chollet, 2018).

Derin öğrenme, sinir ağı programlamada nispeten yeni bir gelişmedir ve derin sinir ağlarını eğitmenin bir yolunu temsil eder, Temelde ikiden fazla katman için tüm nöral ağlar derin sinir ağı olarak adlandırılmaktadır. Pitts'in (1943) çok katmanlı ağları keşfetmesinden bu yana, derin sinir ağlar oluşturulmaya başlanmıştır. Ancak, Hinton (1984) karmaşık sinir ağlarını başarılı bir şekilde eğiten ilk araştırmacı olana kadar, derin sinir ağları etkili bir şekilde eğitilmeyi başaramamıştır (Heaton, 2013: s.171).

Derin öğrenme, son yıllarda oldukça popüler hale gelmiştir. Bunun nedeni, derin öğrenme modellerinin birçok görevde insan seviyesinde veya insan seviyesine yakın performans elde

edebilmesidir. Derin öğrenme, görüntü tanıma, doğal dil işleme, ses tanıma, makine çevirisi, robotik ve finans gibi çeşitli alanlarda kullanılmaktadır.

2.1. Derin Öğrenme Kavramı

Goodfellow, Bengio ve Courville (2016) derin öğrenmenin tanımını; denetimli öğrenme için çok güçlü bir çerçeve sunan, daha fazla katman ve bir katman içinde daha fazla sayıda nöron ekleyerek oluşturulan derin bir ağ olarak tanımlamakta, artan karmaşıklıktaki işlevleri başarılı bir şekilde temsil edebilen ağlar olarak belirtmektedir.

Chollet (2018) derin öğrenmeyi; makine öğreniminin özel bir alt alanı olarak belirtmekte ve giderek daha anlamlı hale gelen ilişkilerin ardışık katmanlarını öğrenmeye vurgu yapan, verilerden ilişkileri öğrenmeye yönelik bir yöntem olarak tanımlamaktadır.

LeCun, Bengio ve Hinton (2015) temsili öğrenmeyi; bir makinenin ham verilerle beslenmesine ve algılama veya sınıflandırma için gereken ilişkileri otomatik olarak keşfetmesine olanak tanıyan bir dizi yöntem olarak belirtmekte ve derin öğrenme yöntemlerini, her biri bir seviyedeki ilişkiyi (ham girdiden başlayarak) daha yüksek, biraz daha soyut bir seviyedeki bir ilişkiye dönüştüren, basit ancak doğrusal olmayan modüllerin oluşturulmasıyla elde edilen çoklu ilişki seviyelerine sahip temsili öğrenme yöntemleri olarak tanımlamaktadır. Bu tür dönüşümlerin yeterli sayıda bir araya getirilmesiyle çok karmaşık fonksiyonlar öğrenilebilir.

Sejnowski (2018) derin öğrenmenin tanımını, matematik, bilgisayar bilimi ve nörobilime dayanan bir makine öğrenme dalı olarak belirtmektedir. Derin öğrenmenin kaynağı, yapay zekanın oluşturulması için iki farklı vizyonun olduğu 1950'lerde yapay zekânın doğuşuna dayanmaktadır: Vizyonlardan birincisi, yapay zekânın yıllardır baskın olduğu mantık ve bilgisayar programlarına dayanmakta, diğeri ise doğrudan verilerden öğrenme sistemine dayanmaktadır (Sejnowski, 2018).

2.2. Derin Öğrenme Alanında Bibliyometrik Analiz

WOS'ta "derin öğrenme" anahtar kelimesi kullanılarak yayınlanmış 88.717 adet akademik çalışma bulunmaktadır. Bu çalışmaların WOS tarafından belirlenmiş kategorilere göre dağılımları Şekil 2'de verilmiştir.



Şekil 2. Kategorilere Göre Yayın Dağılımları (Derin Öğrenme)

Şekil 2 incelendiğinde "Derin Öğrenme" anahtar kelimesini kullanan 88.717 çalışma birden fazla kategoride yer alabildiğinden dolayı toplam 199.349 adet veri üzerinden çalışma gerçekleştirilmiş ve bu çalışmaların %13,72'si Mühendislik Elektrik Elektronik kategorisinde, %9,68'i Bilgisayar Bilimi Yapay Zekâ kategorisinde, %8,30'u Bilgisayar Bilimi Bilgi Sistemleri kategorisinde, %6,48'i Bilgisayar Bilimi Teorisi Yöntemleri kategorisinde, %5,10'u Telekomünikasyon kategorisinde yer almaktadır. Bahsedilen 5 kategori yayınlanan çalışmaların %43,83'ünü kapsamaktadır.

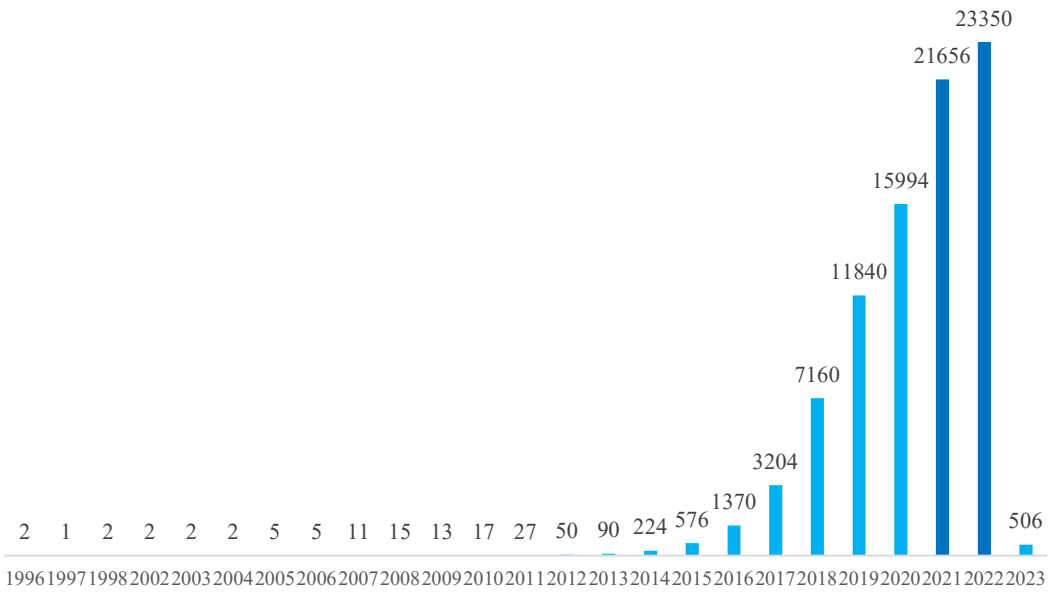
En çok yayın yapılan 20 kategori bu alanda yapılmış çalışmaların %75,59'unu kapsamaktadır. Bundan dolayı Şekil 1'de ilk 20 kategoriye yer verilmiş ve yayın türüne göre yayın dağılımları ise Tablo 1'de verilmiştir.

Tablo 1. Yayın Tipi (Derin Öğrenme)

Yayın Tipi	**Yayın Sayısı**	**Yüzde**
Makale	56944	64,16
Bildiri Metni	28050	31,60
İnceleme	3390	3,82
Editoryal Materyal	256	0,29
Toplantı Özeti	67	0,08
Düzeltme	26	0,03
Mektup	22	0,02
Yeni Baskı	2	0,00
Toplam	88757	100,00

Tablo 1'de görüldüğü gibi Derin Öğrenme hakkında WOS'ta yayınlanan 88.757 çalışmanın %64,16'sı makale türünde, %31,60'ı bildiri türünde ve %3,82'si inceleme türündedir. Bu üç tür tüm çalışmaların %99,58'ini oluşturmaktadır. Yıllara göre yayın dağılım grafiği ise Şekil 3'te verilmiştir.

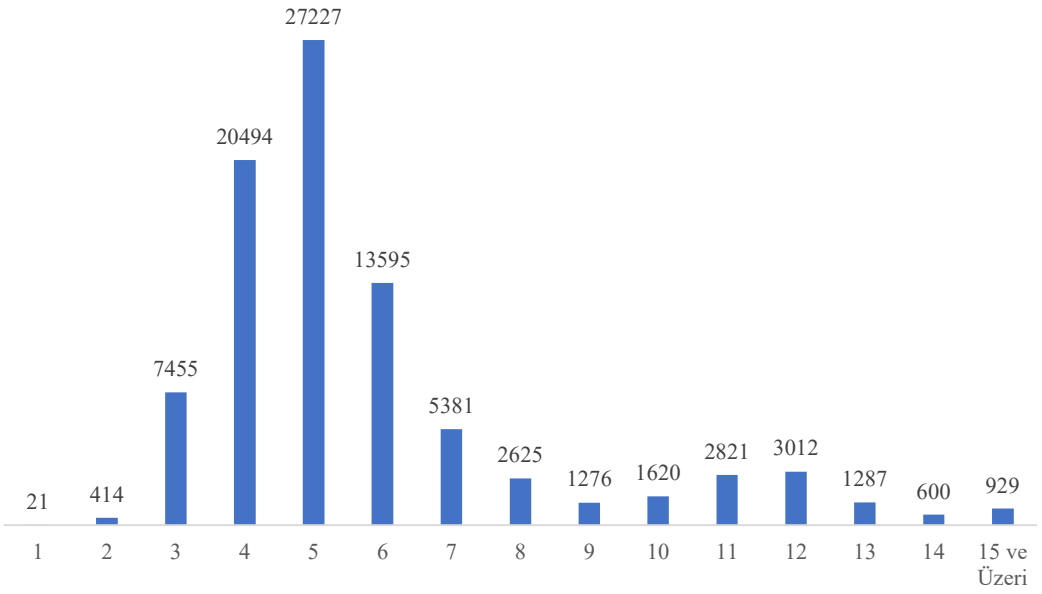
2633 adet çalışmanın yayın tarihi WOS'ta belirtilmediği için grafik 86.124 yayın için gerçekleştirilmiştir.



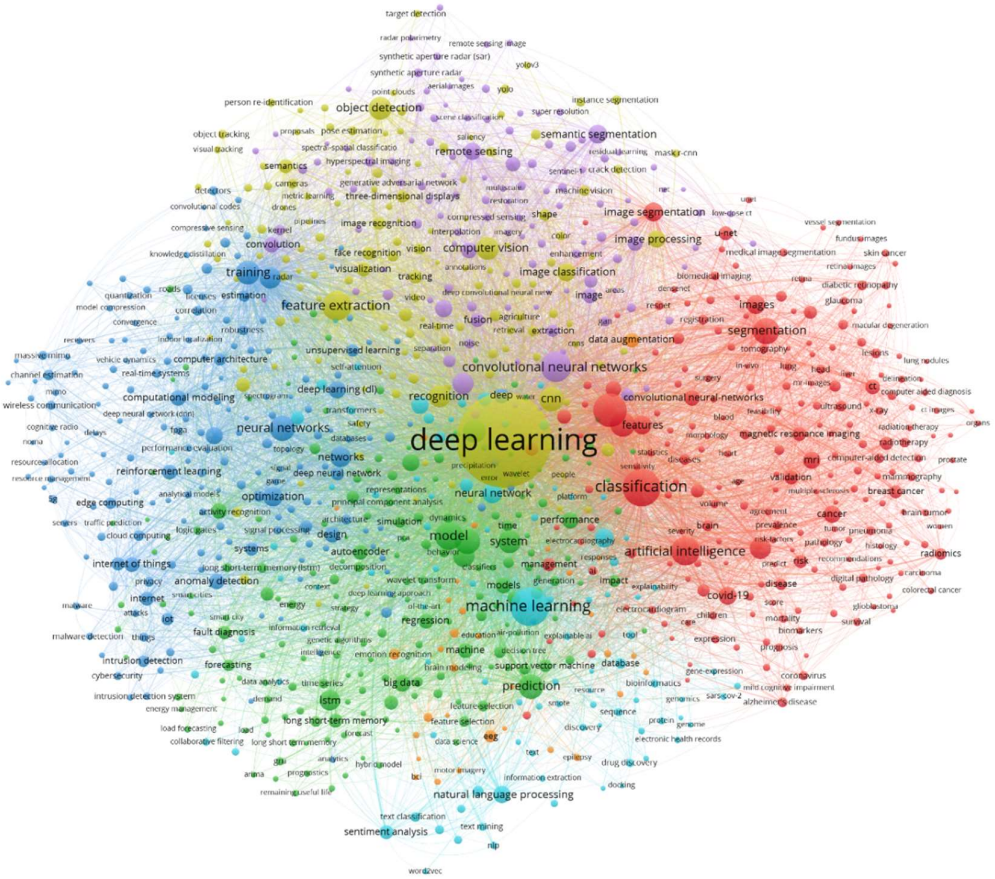
Şekil 3. Yıllara Göre Yayın Sayıları (Derin Öğrenme)

Şekil 3 incelendiğinde Derin Öğrenme alanında ilk çalışmaların 1996 yılında yapılmaya başlandığı görülmektedir. 2007 yılına kadar yıllık yayın sayısı tek hanelerde kalmıştır. Teknolojik gelişmeler ve bilgisayar donanımlarının güçlenmesi ile yayın sayısı artış göstermiştir. 2014 yılından itibaren çıkış gösteren derin öğrenme konusu son yıllarda çıkışına devam etmektedir. 2021 ve 2022 yılları arasında yapılan çalışmalar toplam çalışmaların %52,27'sini oluşturmaktadır. Anahtar kelime sayısına göre yayın dağılımları Şekil 4'te verilmiştir.

Şekil 4 incelendiğinde çalışmaların %23,09'unda 4 anahtar kelime kullanıldığı, %30,68'inde 5 anahtar kelime kullanıldığı %15,32'sinde 6 anahtar kelime kullanıldığı tespit edilmiştir. Çalışmaların %69,08'inde 4 ile 6 arasında anahtar kelime kullanılmıştır. Çalışmalarda kullanılan anahtar kelimelere ait sosyal ağ analizi Şekil 5'te verilmiştir.

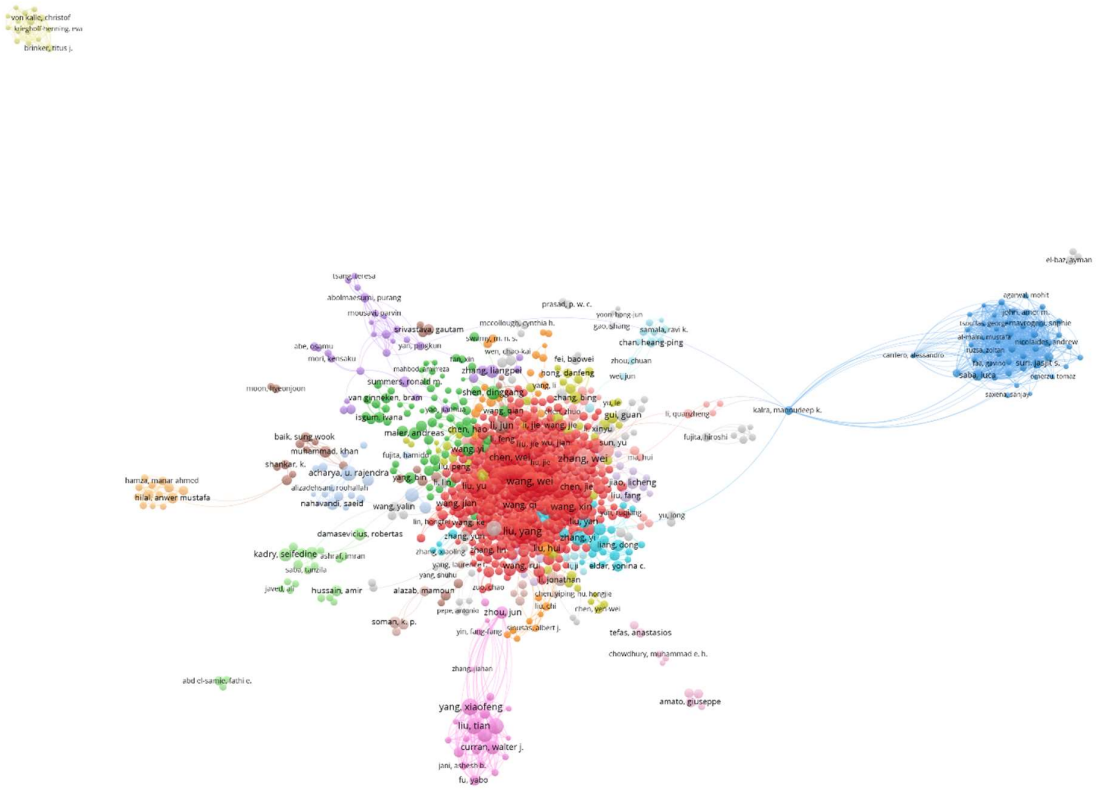


Şekil 4. Anahtar Kelime Sayısına Göre Yayın Sayısı Dağılımı (Derin Öğrenme)



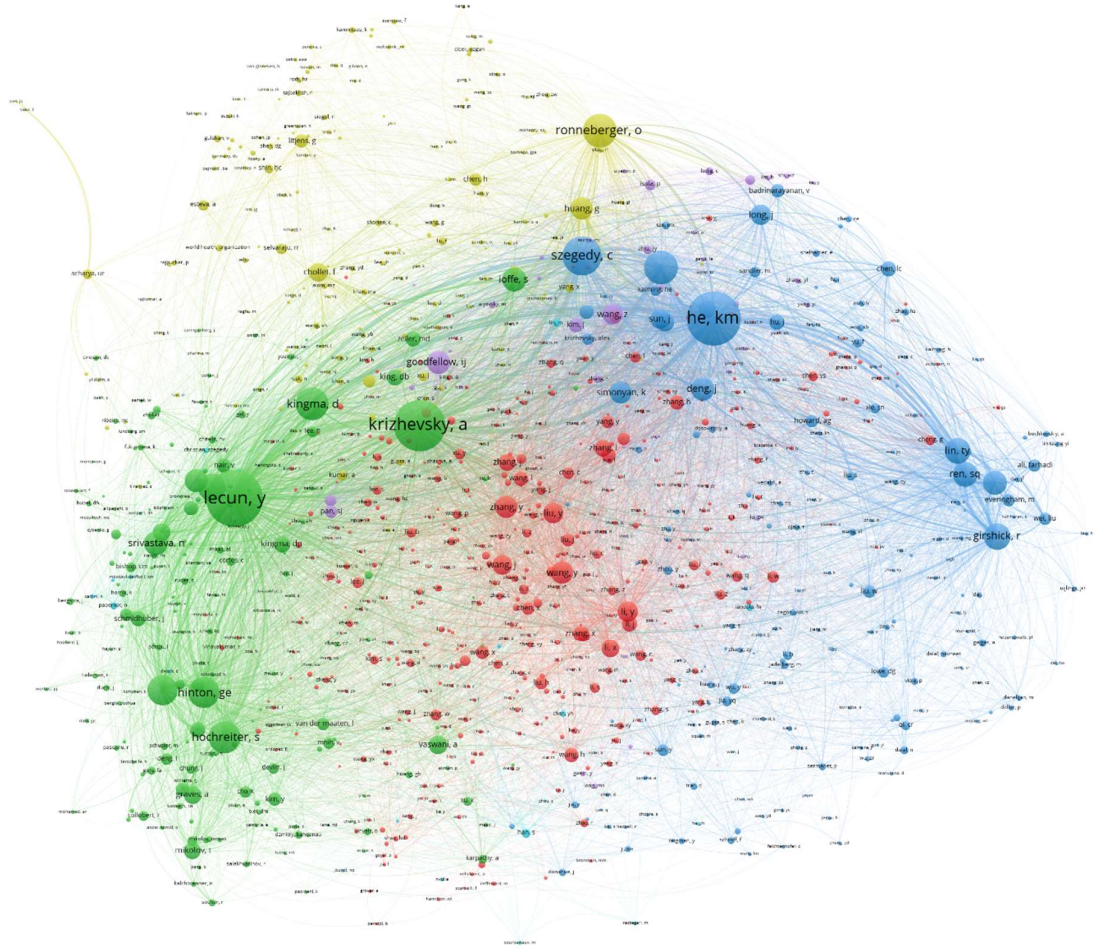
Şekil 5. Anahtar Kelimelere Ait Sosyal Ağ Analizi (Derin Öğrenme)

Şekil 5 incelendiğinde 88.717 adet çalışmada toplam 135.685 adet anahtar kelime kullanıldığı tespit edilmiştir. 135.685 adet anahtar kelime 224 kümede birleşmektedir. Şekilde kümeler renkler ile gösterilmiştir. En çok kullanılan anahtar kelime bibliyometrik analizde de kullanılan derin öğrenme anahtar kelimesidir. Derin öğrenme haricinde, makine öğrenmesi, sınıflandırma, modelleme, sinir ağları, yapay zekâ, segmentasyon, tahmin, eğitim ve özellik çıkartma, anahtar kelimelerinin de sıklıkla çalışmalarda kullanıldığı tespit edilmiştir. Yazarlar arasındaki ilişkileri incelemek için yazarlara ait sosyal ağ analizi gerçekleştirilmiş ve Şekil 6'da verilmiştir.



Şekil 6. Yazarlara Ait Sosyal Ağ Analizi (Derin Öğrenme)

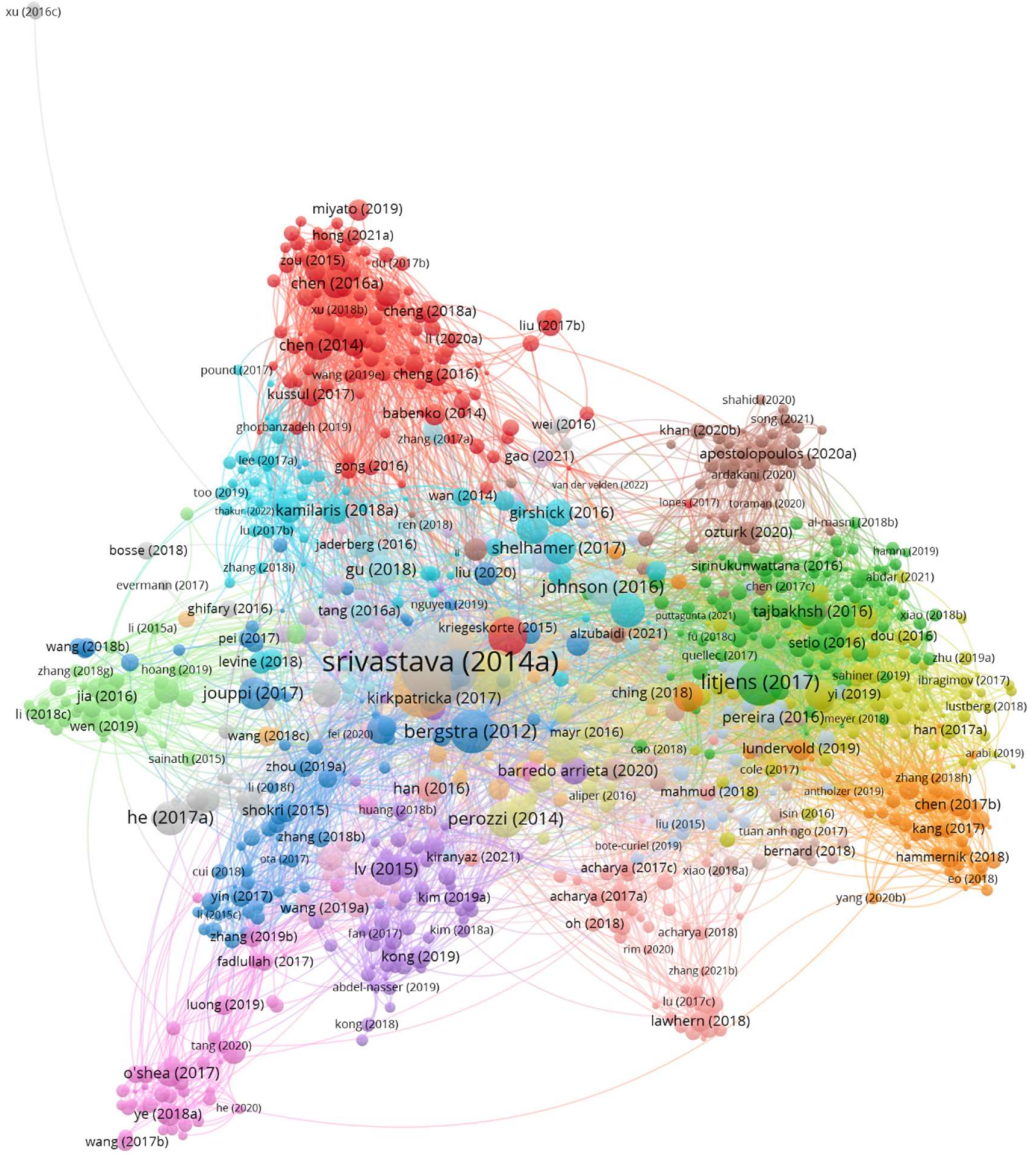
Şekil 6 incelendiğinde 88.717 adet çalışmada toplam 218.551 farklı yazarın yer aldığı ve bu 218.551 yazarın 34 kümede birleştiği tespit edilmiştir. Sosyal ağ analizi sonuçları incelendiğinde yazarların birbiri ile ilişkili bir yapıda olduğu ve çoğunlukla merkezde konumlandığı görülmektedir. Bunların haricinde bazı yazarların kendi aralarında gruplaşarak merkezden bağımsız olarak çalışmalar gerçekleştirdiği görülmektedir. Derin öğrenme anahtar kelimesi kullanan çalışmalar birleştirilerek yazarların bu alanda almış oldukları atıflar sosyal ağ analizi ile incelenmiş ve sonuçlar Şekil 7'de verilmiştir.



Şekil 7. Yazarların Almış Oldukları Atıflara Ait Sosyal Ağ Analizi (Derin Öğrenme)

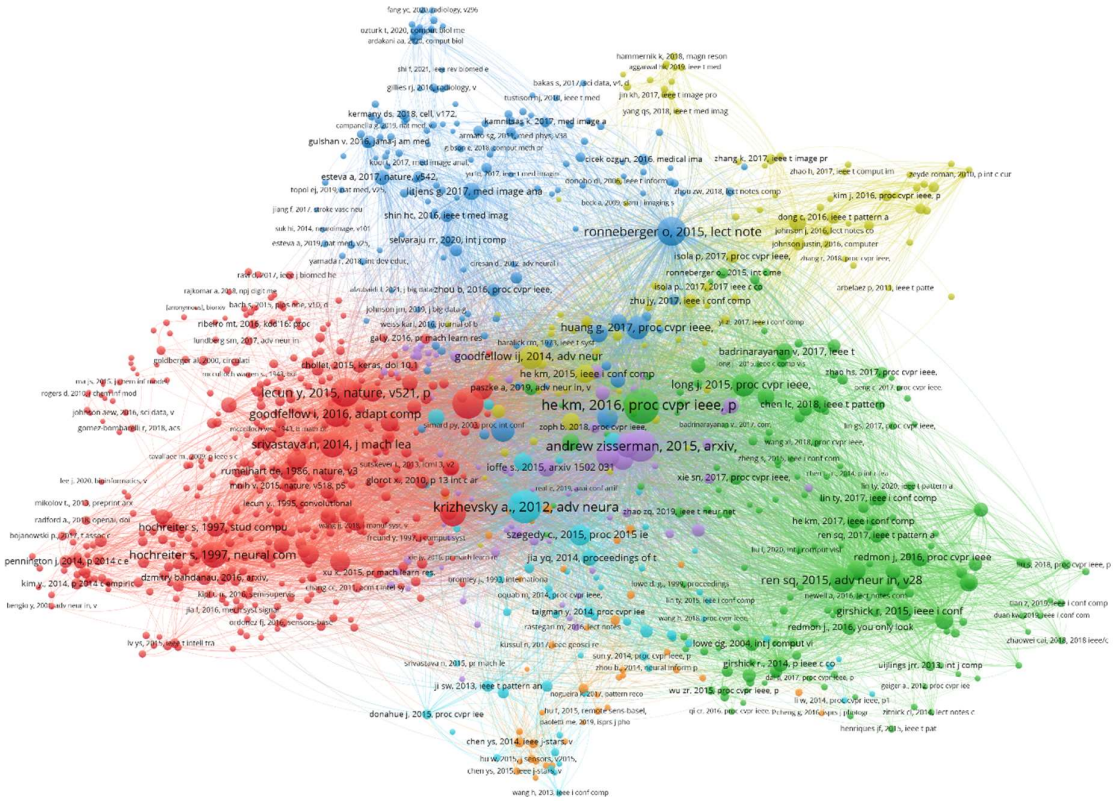
Şekil 7 incelendiğinde A.Krizhevsky, Y. Lecun, KM. He adlı yazarların, derin öğrenme alanında en fazla atıf alan yazarlar olduğu tespit edilmiştir. Derin öğrenme alanında yayın yapmış 218.551 yazar toplamda 565.966 adet atıf almıştır. Toplam atıf miktarı yazar sayısına oranlandığında yazar başına ortalama 2.59 atıf düşmektedir. Alınan atıflara göre yazarların 6 kümede birleştiği görülmektedir. Atıf alan çalışmalara ait sosyal ağ analizi Şekil 8'de verilmiştir.

Şekil 8 incelendiğinde 88.717 adet yayından 59.719 tanesi atıf almıştır. Atıf alan 59.719 adet yayının 21 kümede birleştiği tespit edilmiştir. Derin öğrenme alanında en çok atıf alan çalışma 18.668 adet atıf ile N.Srivastava ve arkadaşlarının (2014) yılında yayınlamış olduğu çalışmadır. İkinci sırada ise 8.434 adet atıf ile J.Schmidhuber'in (2015) yılında yayınlamış olduğu çalışma bulunmaktadır.



Şekil 8. Yayınların Almış Oldukları Atıflara Göre Sosyal Ağ Analizi (Derin Öğrenme)

Derin öğrenme anahtar kelimesine sahip 88.717 adet çalışmada kullanılan referanslara ait sosyal ağ analizi Şekil 9'da verilmiştir.



Şekil 9. Kullanılan Referanslara Ait Sosyal Ağ Analizi (Derin Öğrenme)

Şekil 9 incelendiğinde 88.717 adet çalışmada toplamda 1.438.528 adet referans kullanıldığı görülmektedir. Kullanılan referans miktarı yayın sayısına oranlandığında yayınların ortalama 16.21 adet referans kullandığı tespit edilmiş ve kullanılan 1.438.528 adet referansın 7 kümede toplandığı görülmüştür

2.3. Derin Öğrenmenin Uygulama Alanları

Derin öğrenme görüntü ve ses tanıma, doğal dil işleme, örüntü algılama işlemlerinde oldukça başarılı bir performans göstermesinden dolayı başta oyun, sağlık, otomotiv, finans, perakende ve e-ticaret olmak üzere birçok sektörde aktif bir şekilde kullanılmaktadır.

2.3.1. Görüntü ve Ses Tanıma

Günümüzde, derin öğrenme teknikleri, özellikle görüntü ve ses tanıma alanında önemli başarılar elde etmektedir. Gelişmiş yapay sinir ağları ve derin öğrenme algoritmaları, bu alanlarda hassas ve doğru sonuçlar elde etmekte kullanılmaktadır.

Derin öğrenme, görüntü tanıma alanında devrim niteliğinde sonuçlar üretmektedir. Convolutional Neural Networks (CNN) adı verilen özel bir derin öğrenme mimarisi, bu alanda özellikle etkili olmaktadır. Bu mimari, bir görüntünün farklı özelliklerini öğrenmek ve tanımak için katmanlarca

işlem uygulamaktadır. Örneğin, yüz tanıma, nesne tanıma ve otomatik sürücüsüz araçlardaki nesne algılama gibi uygulamalarda derin öğrenme, yüksek doğruluk oranlarına ulaşmaktadır.

Görüntü tanıma alanındaki bu gelişmeler, tıp sektöründe de büyük bir etki yaratmaktadır. Radyolojik görüntülerin analizi, kanser taraması ve tıbbi teşhislerde derin öğrenme modelleri, uzmanlık düzeyinde sonuçlar üretme kabiliyetine sahip olmaktadır.

Ses tanıma, derin öğrenmenin başarılı olduğu alanlar arasında yer almaktadır. Recurrent Neural Networks (RNN) ve Long Short-Term Memory (LSTM) gibi özel mimariler, ses verilerinin işlenmesinde etkili olmaktadır. Bu teknikler, konuşma tanıma, sesli komutları anlama ve konuşma sentezi gibi bir dizi uygulamada kullanılmaktadır.

Derin öğrenme modelleri, ses tanıma alanında da olağanüstü başarılar sağlamakta ve bu başarılar gün geçtikçe daha da artmaktadır. Derin öğrenme modelleri, sanal asistanlar, konuşma tabanlı kontrol sistemleri ve otomatik çeviri gibi uygulamaların gelişmesine olanak tanımaktadır.

2.3.2. Doğal Dil İşleme

Doğal Dil İşleme, bilgisayarların insan dilini anlaması ve işlemesi için geliştirilen bir dal olarak ifade edilmektedir. Derin öğrenme, doğal dil işlemenin temel görevlerini gerçekleştirmek için özellikle güçlü bir araç haline gelmektedir. Dilin karmaşıklığı ve çeşitliliği göz önüne alındığında, derin öğrenme modelleri büyük dil veri setleri üzerinde eğitilerek dilin derin yapılarından anlam çıkarma yeteneği kazandırılmaktadır.

Derin öğrenme, metin sınıflandırma görevlerinde etkileyici sonuçlar elde etmektedir. Örneğin, bir metnin belirli bir kategoriye ait olup olmadığını belirleme veya bir metindeki duygu tonunu analiz etme gibi görevlerde derin öğrenme modelleri kullanılmaktadır. Derin öğrenme modelleri, sosyal medya analizi, müşteri geri bildirimleri değerlendirmesi ve hisse senedi piyasası gibi birçok alanda kullanılmaktadır.

Çeviri uygulamalarında, derin öğrenme, bir dildeki metni başka bir dile anlamlı bir şekilde çevirme yeteneğiyle öne çıkmaktadır. Aynı zamanda, konuşma tanıma sistemlerinde de kullanılarak sesli komutları veya konuşma metinlerini anlamlandırmakta ve işlemektedir.

Dil tabanlı kişisel asistanlar (örneğin, Siri veya Alexa), otomatik metin düzeltme önerileri (klavye uygulamalarında), anlamak ve cevap vermek amacıyla geliştirilmiş sohbet robotları gibi günlük hayatta birçok derin öğrenme uygulaması bulunmaktadır.

2.3.3. Oyun Sektörü

Derin öğrenme, oyun endüstrisinde önemli bir etki yaratmakta ve oyun geliştiricilerine yeni ve güçlü araçlar sunarak oyun deneyimini zenginleştirmektedir. Bu teknoloji, birçok farklı alanı kapsayarak oyunları daha etkileyici, gerçekçi ve kişiselleştirilmiş hale getirmektedir.

Derin öğrenme algoritmaları, oyun karakterlerine daha karmaşık ve gerçekçi tepkiler kazandırmak için kullanılmaktadır. Bu sayede, oyun içindeki yapay zekâya sahip düşmanlar daha akıllı, oyun yer alan karakterler daha gerçekçi ve oyuncuların kararlarına daha adaptif bir şekilde tepki verebilecek düzeyde olabilmektedir.

Derin öğrenme aynı zamanda oyun içi reklamcılık ve kişiselleştirme konularında da kullanılmaktadır. Oyunlar, oyuncuların davranışlarına dayanarak reklam içeriklerini kişiselleştirebilir, böylece reklamlar daha ilgi çekici ve etkili olabilmektedir. Bu durum, oyun geliştiricileri ve yayıncılar için yeni gelir kaynakları ve oyuncular için daha özelleştirilmiş deneyimler sağlamaktadır.

2.3.4. Sağlık Sektörü

Derin öğrenme, sağlık ve tıp alanında çeşitli uygulamalarda önemli bir rol oynamaktadır. Bu yenilikçi teknoloji, büyük miktarda veriyi analiz edebilme ve karmaşık desenleri tanıyabilme yeteneği sayesinde tıbbi araştırmalarda, teşhislerde ve tedavilerde büyük bir potansiyel sunmaktadır.

Tıbbi veri analizi konusunda derin öğrenme, genomik ve proteomik veri setlerinin anlaşılması, hastalıkların genetik temellerinin belirlenmesi ve bireyselleştirilmiş tedavi planlarının oluşturulması gibi alanlarda da kullanılmaktadır. Genetik verilerin derinlemesine analizi, bireyler arasındaki genetik farklılıkları anlamamıza ve hastalıkların genetik kökenlerini aydınlatmamıza yardımcı olmaktadır.

Derin öğrenme aynı zamanda ilaç geliştirme süreçlerinde de kullanılmaktadır. Moleküler yapı analizi, ilaç etkileşimleri ve yan etkilerin değerlendirilmesi gibi konularda derin öğrenme, ilaç araştırmalarına hız kazandırmakta ve maliyetleri düşürmektedir.

2.3.5. Otomotiv Sektörü

Otomotiv endüstrisi, teknolojinin hızla ilerlemesiyle birlikte, derin öğrenme gibi yapay zekâ tekniklerini benimseyerek, önemli bir dönüşüm geçirmektedir. Derin öğrenme, özellikle sürücüsüz araç teknolojileri, güvenlik sistemleri ve otomotiv tasarımında önemli bir rol oynamaktadır.

Derin öğrenme, otomobillerin güvenlik sistemlerini daha etkili hale getirmektedir. Örneğin, sürücü dikkatini izleyen sistemlerde, kameralar tarafından yakalanan sürücünün yüz ifadeleri ve göz hareketleri, derin öğrenme algoritmaları ile analiz edilerek sürücünün dikkatinin dağılmış olup olmadığını belirlemektedir. Ayrıca, kaza önleme sistemlerinde de derin öğrenme, çevresel tehditleri tanımlama ve sürücüyü uyararak çarpışmaları azaltma potansiyeli sunmaktadır.

Derin öğrenme, otonom araçların çevresini daha iyi anlamalarına yardımcı olacak detaylı yol haritaları oluşturmada da kullanılmaktadır. Yüksek çözünürlüklü kameralar, lidar ve radar sensörlerinden gelen veriler, derin öğrenme algoritmalarıyla birleştirilerek, araçların anlık olarak çevrelerini haritalandırmalarına olanak tanımaktadır. Bu, otonom araçların güvenli bir şekilde seyahat etmelerini ve karmaşık yol koşullarına uyum sağlamalarını sağlamaktadır.

2.3.6. Finans Sektörü

Finans sektöründe derin öğrenmenin kullanılmasının en belirgin örneklerinden biri, risk yönetimi ve sahtecilik işlemlerini tespit etmektir. Derin öğrenme algoritmaları, büyük veri setlerini analiz ederek potansiyel sahtekarlık işlemlerini tanımlama konusunda etkili olmaktadır. Bu, finansal kurumların sahtecilikle mücadelesini güçlendirmekte ve müşteri varlıklarını daha iyi korumalarına olanak tanımaktadır.

Ayrıca, portföy yönetimi ve ticaret stratejilerinin geliştirilmesinde de derin öğrenme kullanılmaktadır. Bu algoritmalar, piyasa verilerini analiz ederek gelecekteki fiyat hareketlerini tahmin etmeye çalışmaktadır. Bu sayede, yatırım profesyonelleri daha bilinçli kararlar alabilmekte ve portföylerini daha etkili bir şekilde yönetebilmektedir.

Derin öğrenme, aynı zamanda kredi değerlendirmesi süreçlerinde de kullanılmaktadır. Bankalar, müşteri kredi başvurularını değerlendirirken derin öğrenme algoritmalarını kullanarak müşteri kredi riskini daha doğru bir şekilde belirleyebilmektedir. Derin öğrenme modelleri, kredi değerlendirmesi sürecini iyileştirmekte ve finansal kuruluşların daha güvenilir kararlar almasına yardımcı olmaktadır.

2.3.7. Perakende ve E-Ticaret

Perakende sektöründe derin öğrenme, müşteri davranışlarını anlamak ve öngörmek için kullanılmaktadır. Müşteri sadakati ve satın alma alışkanlıkları üzerinde etkili olan bir dizi faktör, derin öğrenme modelleri tarafından analiz edilebilmektedir. Bu analizler, işletmelerin kişiselleştirilmiş kampanyalar oluşturmalarına, ürün önerileri sunmalarına ve müşteri sadakatini artırmalarına yardımcı olmaktadır.

E-ticaret platformları, derin öğrenme sayesinde daha etkili öneri sistemleri geliştirebilmektedir. Müşterilerin alışveriş geçmişleri, tıklama davranışları ve beğenileri üzerinden özelleştirilmiş öneriler sunmak, müşteri memnuniyetini artırmanın yanı sıra, işletmelerin gelirlerini de optimize etmelerine yardımcı olmaktadır. Bu öneri sistemleri, kullanıcının ilgi alanlarına göre dinamik olarak uyum sağlayabilmekte, böylece zamanla daha hassas hale gelmektedir.

Derin öğrenme modelleri, talep tahmini, perakende ve e-ticaret sektörlerinde stok yönetimi ve tedarik zinciri planlamasında kritik faktörler arasında yer almaktadır. Derin öğrenme modelleri, geçmiş satış verileri, mevsimsel etkiler ve pazar trendleri gibi bir dizi faktörü değerlendirerek gelecekteki talepleri tahmin edebilmektedir. Bu sayede işletmeler, stoklarını optimize ederek müşteri taleplerini karşılamak için doğru miktarda ürüne sahip olabilmektedir.

2.4. Derin Öğrenmenin Yapısı ve Temel Elemanları

Derin öğrenmenin yapısında girdiler, ağırlıklar, toplama ve aktivasyon fonksiyonları ve çıktılar yer almaktadır. Girdi katmanı, gizli katman ve çıktı katmanı ise derin öğrenmenin temel elemanlarını oluşturmaktadır. Derin öğrenmenin yapısı Şekil 10'da gösterilmektedir.

Girdi Katmanı, ağa dış dünyadan gelen bilgileri almakta ve modelin temel girişini oluşturmaktadır. Bu katman, genellikle veri setindeki özelliklere karşılık gelen nöronlardan oluşmakta ve başlangıç noktasını belirlemektedir.

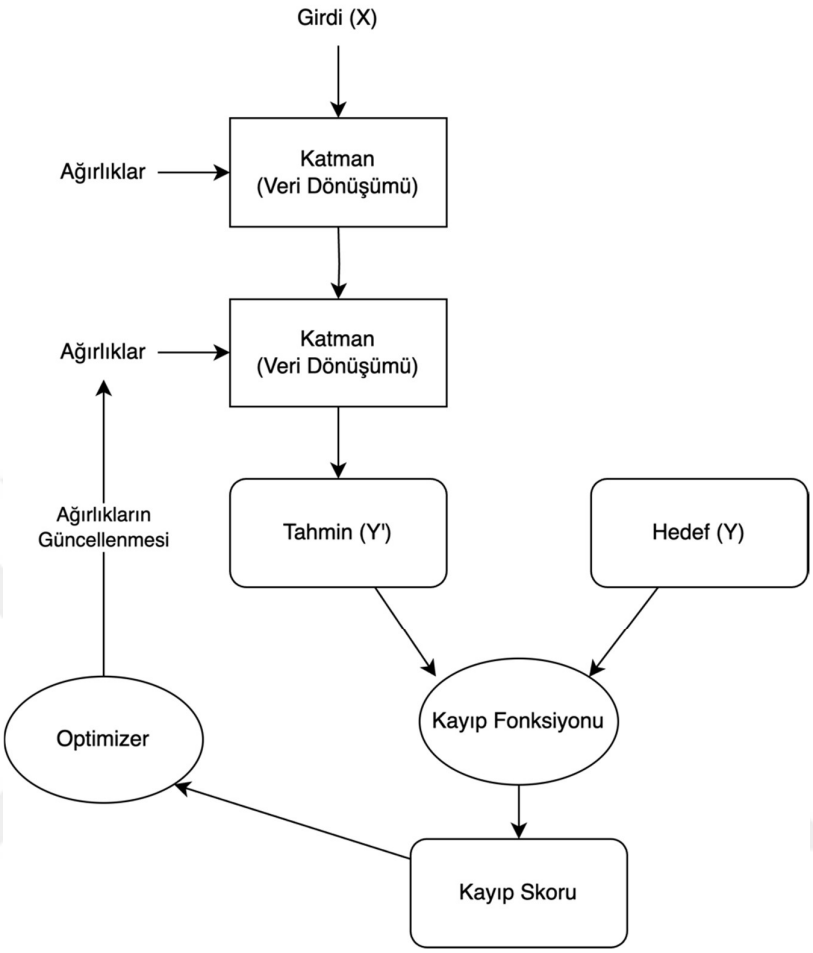
Veri Dönüşümü, girdi katmanındaki verileri modelin anlayabileceği bir forma çevirmektedir. Diğer bir ifade ile veri dönüşümü veri setindeki özelliklerin standartlaştırılması, normalleştirilmesi veya dönüştürülmesi gibi işlemleri içermektedir. Bu adım, ağın öğrenme sürecini daha etkili kılmaktadır.

Ağırlıklar, sinir ağındaki bağlantıları temsil etmekte ve her bir nöronun girdilerini ağırlıklandırarak çıktı üretmesine katkıda bulunmaktadır. Bu ağırlıklar, modelin öğrenmesini sağlamak adına eğitim sürecinde güncellenmektedir.

Kayıp Fonksiyonu, modelin tahminlerini gerçek değerlerle karşılaştırarak modelin performansını değerlendirmektedir. Kayıp Skoru, bu fonksiyonun çıktısını temsil etmekte ve modelin ne kadar yanıldığını belirlemektedir. Optimizer, modelin kaybını minimize etmek için ağırlıkları güncellemektedir.

Ağırlıkların Güncellenmesi, geri yayılım algoritması aracılığıyla elde edilen gradyanlara dayanarak ağırlıkların yeniden ayarlaması sürecini ifade etmektedir. Bu süreç, modelin doğru sonuçlara daha yakınlaşmasını sağlamaktadır.

Bu temel elemanlar, derin öğrenme sürecinin temelini oluşturmakta ve modelin karmaşıklığını, doğruluğunu ve genelleme yeteneğini belirlemektedir.



Şekil 10. Derin Öğrenme Yapısı ve Temel Elemanları

2.4.1. Derin Öğrenmenin Yapısı

Derin öğrenmenin öğrenme sürecinde, temel elemanlar modelin yapısını oluşturmaktadır. Girdi, veri setindeki bilgiyi ifade etmektedir. Ağırlıklar, girdi verileriyle ilişkilendirilen önem derecelerini belirlemekte ve öğrenme sürecinde güncellenmektedir. Toplama fonksiyonu, ağırlıklı girdi değerlerini birleştirip ve sapma eklemektedir. Aktivasyon fonksiyonu, bu toplamı sıkıştırarak ve belirli bir aralığa taşıyarak çıkışı üretmektedir. Çıktı ise modelin sonucunu ifade etmektedir.

2.4.1.1. Girdiler

Derin öğrenme modelleri, girdiler olarak adlandırılan verilerle eğitilmektedir. Girdiler, genellikle sayılar, metinler veya görüntüler biçiminde sisteme tanımlanabilir. Girdilerin biçimi, modelin uygulanacağı göreve bağlı olarak değişiklik göstermektedir.

Sayısal girdiler, genellikle sayılar, kümeler veya vektörler olarak ifade edilmekte ve görüntü tanıma, doğal dil işleme ve makine öğreniminin diğer alanlarında kullanılmaktadır. Metin girdileri, genellikle kelimeler, cümleler veya paragraflar olarak ifade edilmekte ve doğal dil işleme, istem

yanıtlama ve makine çevirisi gibi alanlarda kullanılmaktadır. Görüntü girdileri ise pikseller dizisi olarak ifade edilir ve görüntü tanıma, yüz tanıma ve nesne tespiti gibi alanlarda kullanılmaktadır.

2.4.1.2. Ağırlıklar

Derin öğrenme modellerinin temel elemanlarından biri ağırlıklar olarak ifade edilmektedir. Ağırlıklar, yapay sinir ağlarının öğrenme sürecinde büyük bir rol oynamaktadır. Her bir ağırlık, girdi verileriyle ilişkilendirilen önem derecesini temsil etmekte ve öğrenme sürecinde adaptasyon geçirmektedir. Bu ağırlıklar, modelin karmaşık ilişkileri anlamasını ve öğrenmesini sağlamaktadır. Özellikle, geri yayılım algoritması kullanılarak, modelin çıktısındaki hata hesaplanmakta ve bu hata, ağırlıkların güncellenmesi için kullanılmaktadır. Bu adaptasyon süreci, modelin veri setindeki örüntüleri anlamasını ve genelleme yeteneğini geliştirmesini sağlamaktadır.

Ağırlıkların doğru ayarlanması, modelin başarısını ve performansını belirleyen kritik faktörlerden biridir. Bu nedenle, derin öğrenme uygulamalarında, ağırlıkların etkili bir şekilde öğrenilmesi ve güncellenmesi, modelin genel kalitesini belirleyen önemli bir süreç olarak ifade edilmektedir. Ağırlıkların artı veya eksi olması etkinin pozitif ya da negatif olduğunu sıfır olması ise herhangi bir etkinin olmadığını göstermektedir (Öztemel, 2012).

2.4.1.3. Toplama Fonksiyonu

Derin öğrenme modellerinde toplama fonksiyonu, ağırlıklarla çarpılmış girdi değerlerini birleştiren kritik bir bileşen olarak ifade edilmektedir. Bu fonksiyon, her bir girdi ile ona atanmış olan ağırlıkları çarparak ve ardından bu değerleri toplayarak ağırlıklı toplamı üretmektedir. Toplama fonksiyonu, derin öğrenmedeki katmanlar arasında bilgi taşınmasını sağlamaktadır. Genellikle sapma terimi de eklenerek modelin daha esnek bir yapıda olması sağlanmakta ve öğrenme kapasitesi genişletilmektedir. Farklı toplama fonksiyonları, sinir ağı modellerinde farklı davranışları temsil etmektedir. Bu fonksiyonlar, derin öğrenmenin karmaşıklığını artırabilmekte ve belirli türdeki problemlere daha iyi uyan modellerin oluşturulmasına olanak tanımaktadır. Toplama fonksiyonları, derin öğrenme modelinin içsel işleyişinde temel bir taşıyıcı rol oynayarak bilgi akışını yönetmektedir.

Toplama fonksiyonu, bir hücreye gelen net girdiyi hesaplamaktadır. Bunun için değişik fonksiyonlar kullanılmaktadır. En yaygın olanı ağırlıklı toplamı bulmaktadır. Burada her gelen girdi değeri kendi ağırlığı ile çarpılarak toplanmaktadır. Böylece ağa gelen net girdi bulunmuş olmaktadır. Toplama Fonksiyonu Denklem 2.1'de gösterildiği gibi formülize edilmektedir (Öztemel, 2012).

$$NET = \sum_{i=1}^{n} G_i A_i \tag{2.1}$$

Burada G girdileri, A ağırlıkları, n ise hücreye gelen toplam girdi sayısını göstermektedir. Literatürde yapılan çalışmalarda toplama fonksiyonu olarak değişik formüllerin kullanıldığı görülmektedir. Tablo 2'de diğer toplama fonksiyonlarının örnekleri verilmiştir.

Tablo 2. Alternatif Toplama Fonksiyonları (Öztemel, 2012)

Net	Açıklama
$NET = \prod_{i=1} G_i A_i$	Ağırlık değerleri girdiler ile çarpılır ve daha sonra bulunan değerler birbirleri ile çarpılarak net girdi değeri hesaplanmaktadır.
$NET = Max(G_i A_i) \qquad i = 1,\ldots,N$	Ağırlık değerleri ile girdiler çapılır ve daha sonrasında N adet değerden en büyük olanı net girdi değeri olarak kabul edilmektedir.
$NET = Min(G_i A_i) \qquad i = 1,\ldots,N$	Ağırlık değerleri ile girdiler çapılır ve daha sonrasında N adet değerden en küçük olanı net girdi değeri olarak kabul edilmektedir.
$NET = \sum_{i=1}^{n} \text{sgn}(G_i A_i)$	Ağırlık değerleri ile girdiler çapılır ve daha sonrasında N adet değer pozitif, negatif olarak ayrılır ve büyük olan sayı hücrenin net girdisi olarak kabul edilmektedir.
$NET = NET(eski) + \sum(G_i A_i)$	Hücreye gelen bilgiler ağırlıklı olarak toplanır ve daha önce gelen bilgilere eklenerek hücrenin net girdisi hesaplanmaktadır.

2.4.1.4. Aktivasyon Fonksiyonu

Derin öğrenme süreçlerinde aktivasyon fonksiyonları, yapay sinir ağlarının öğrenme kapasitesini artıran kritik unsurlar olarak yer almaktadır. Bu fonksiyonlar, her bir nöronun çıkışını belirlemekte ve modelin karmaşıklığını kontrol etmektedir. İkili basamak fonksiyonu, doğrusal, sigmoid, tanh, ReLU, sızıntı ReLU, parametrize edilmiş ReLU, Üstel Doğrusal Birim, Swish (Kendinden Geçitli) ve SoftMax aktivasyon fonksiyonları sıklıkla kullanılan aktivasyon fonksiyonlarının başında gelmektedir (Agostinelli, Hoffman, Sadowski ve Baldi, 2014). Aktivasyon fonksiyonları, sinir ağlarının öğrenme sürecini hızlandırmakta, modelin genelleme yeteneğini artırmakta ve aşırı uydurmayı önlemektedir. Bu fonksiyonlar, derin öğrenme modellerinin güçlü ve esnek öğrenme yapısını şekillendirmektedir.

İkili basamak aktivasyon fonksiyonu, girişin eşik değerini aştığında 1, aşamadığında 0 yanıtını veren basit bir aktivasyon fonksiyonu olarak belirtilmektedir. Matematiksel olarak Denklem 2.2'deki gibi ifade edilmektedir.

$$f(x) = \begin{cases} 0, & x < EşikDeğer \\ 1, & x \geq EşikDeğer \end{cases} \tag{2.2}$$

Doğrusal aktivasyon fonksiyonu, girişi olduğu gibi ileten basit bir fonksiyon olarak ifade edilmekte ve matematiksel olarak Denklem 2.3'teki şekilde gösterilmektedir. Bu fonksiyon, özellikle regresyon problemlerinde kullanılmaktadır.

$$f(x) = x \tag{2.3}$$

Sigmoid aktivasyon fonksiyonu, girişi 0 ile 1 arasına sıkıştırmakta ve genellikle ikili sınıflandırma problemlerinde çıkış üretmek için kullanılmaktadır. Denklem 2.4'teki gibi gösterilmektedir.

$$f(x) = \frac{1}{1+e^{-x}} \tag{2.4}$$

Tanh aktivasyon fonksiyonu, girişi -1 ile 1 arasına sıkıştırmakta ve genellikle derin öğrenmenin orta katmanlarında kullanılmaktadır. Denklem 2.5'teki gibi gösterilmektedir.

$$f(x) = \frac{e^{x} - e^{-x}}{e^{x} + e^{-x}} \tag{2.5}$$

ReLU aktivasyon fonksiyonu, pozitif giriş değerlerini olduğu gibi bırakırken, negatif değerleri sıfıra dönüştürmekte kullanılmaktadır. Denklem 2.6'daki gibi gösterilmektedir.

$$f(x) = \max(0, x) \tag{2.6}$$

Sızıntı ReLU aktivasyon fonksiyonu, ReLU fonksiyonundan farklı olarak negatif değerleri küçük bir eğimle (0,01) çarpmaktadır. Denklem 2.7'deki gibi gösterilmektedir.

$$f(x) = \max(0,01 \times x, x) \tag{2.7}$$

Parametrize edilmiş ReLU aktivasyon fonksiyonu sızıntı ReLU fonksiyonundan farklı olarak negatif değerleri α gibi bir eğimle çarpmaktadır. Denklem 2.8'deki gibi gösterilmektedir.

$$f(x) = \max(\alpha \times x, x) \tag{2.8}$$

Üstel Doğrusal Birim aktivasyon fonksiyonu, negatif girişler için daha yumuşak bir sıfır değeri sağlamak amacıyla kullanılmakta ve negatif değerler için log eğrisi kullanmaktadır. Denklem 2.9'daki gibi gösterilmektedir.

$$f(x) = \begin{cases} \alpha\left(e^{x} - 1\right), & x < 0 \\ x, & x \geq 0 \end{cases} \tag{2.9}$$

Swish aktivasyon fonksiyonu, sigmoid ve ReLU aktivasyon fonksiyonlarının kombinasyonu olan bir fonksiyon olarak ifade edilmektedir. Denklem 2.10'daki gibi gösterilmektedir.

$$f(x) = x \times \frac{1}{1+e^{-x}} \tag{2.10}$$

SoftMax aktivasyon fonksiyonu, bir vektörün elemanlarını (genellikle sınıf olasılıkları olarak yorumlanan) 0 ile 1 arasına sıkıştırmakta ve toplamı 1 olacak şekilde normalize etmektedir. Genellikle çok sınıflı sınıflandırma problemlerinde kullanılmaktadır. Denklem 2.11'deki gibi gösterilmektedir.

$$f(x)_i = \frac{e^{x_i}}{\sum_j e^{x_j}} \tag{2.11}$$

2.4.1.5. Çıktılar

Derin öğrenmedeki çıktı, bir modelin öğrenme sürecinin son aşamasını temsil etmekte ve genellikle modelin hedeflediği görevle ilgili bilgileri içermektedir. Bu aşamada amaç, girdi verilerinin karmaşıklığını anlamlandırarak, modelin eğitildiği veri setinden elde ettiği bilgileri kullanarak çıktı üretmektedir. Örneğin, bir görüntü sınıflandırma modeli ise çıktı, görüntünün hangi sınıfa ait olduğunu belirten bir olasılık dağılımı olabilmektedir. Dil işleme uygulamalarında ise çıktı, verilen metinle ilgili anlamı içermektedir. Derin öğrenme modellerinde çıktı, geniş bir uygulama yelpazesi içinde bilgi öğrenme ve karmaşık görevleri gerçekleştirme yeteneğiyle önemli bir rol oynamaktadır.

2.4.2. Derin Öğrenmenin Temel Elemanları

Derin öğrenmenin temel elemanları girdi katmanı, gizli katmanlar ve çıktı katmanından oluşmaktadır.

2.4.2.1. Girdi Katmanı

Derin öğrenmenin girdi katmanı, bir modelin eğitildiği temel adımı temsil etmekte ve veri setinden alınan bilgileri modele sunmaktadır. Girdi katmanı, ağa çeşitli formatlardaki girdileri kabul edebilmektedir. Bu katman, genellikle veri setinin özelliklerini ve boyutlarını yansıtan düğümlerden oluşmaktadır.

Girdi katmanı, modelin algılaması ve öğrenmesi gereken desenleri içeren veriyi almaktadır. Girdi verileri genellikle önceden işlenmekte ve standartlaştırılmaktadır, böylece modelden, öğrenme sürecinde daha tutarlı ve etkili sonuçlar elde edebilmektedir.

Bu katman, öğrenilecek ağırlıkları içeren bir matrisi de içermektedir. Bu ağırlıklar, girdi verilerinin model tarafından anlaşılmasını sağlamak için kullanılmaktadır. Her düğüm, girdi verileriyle ilişkilendirilmiş bir ağırlığa sahiptir ve bu ağırlıklar öğrenme sürecinde güncellenmektedir.

Girdi katmanı, modelin genel performansını etkileyen önemli bir bileşen olarak ifade edilmektedir. Doğru veri seti seçimi ve girdi katmanının uygun boyutlandırılması, modelin başarılı öğrenme yeteneğini belirleyebilmektedir. Ayrıca, girdi katmanının esnekliği, farklı türdeki verilerle çalışabilme yeteneği açısından da kritiktir, bu da derin öğrenme modellerini çeşitli uygulama alanlarında kullanılabilir kılmaktadır. Girdi katmanı, derin öğrenme sürecinde bilgi akışının başlangıç noktası olarak ifade edilmekte ve modelin geri kalan katmanlarda daha karmaşık özellikleri öğrenmesine olanak tanımaktadır.

2.4.2.2. Gizli Katman

Bir derin öğrenme ağında ne kadar çok gizli katman olursa, veride bulunan karmaşık ilişkiler daha fazla temsil edilebilir (Mitchell, 1997). Girdi katmanına yakın olan katmanlar daha basit özellikleri algılarken, çıktı katmanına yakın olan katmanlar karmaşık ilişkileri algılayabilir (Di, Bhardwaj ve Wei, 2018).

Özellikle, gizli katmanlar, girdi verilerini temsil eden ağırlık matrisleri ve aktivasyon fonksiyonları arasında karmaşık bir etkileşimle çalışmaktadır.

Gizli katmanlar, modelin öğrenme sürecinde genellikle modelin karmaşıklığını belirleyen önemli bir rol oynamaktadırlar. Derin öğrenme modeli, bu katmanlar aracılığıyla veri setindeki özellikleri hiyerarşik bir şekilde öğrenerek daha karmaşık ve soyut ilişkiler oluşturmaktadırlar. Her bir gizli katman, önceki katmanın çıktılarından gelen bilgileri almaktadırlar.

Bu katmanlar genellikle ağırlık matrisleri sayesinde özelliklerin doğrusal kombinasyonlarını çıkartmakta ve ardından aktivasyon fonksiyonları, bu doğrusal çıktıları daha karmaşık ve non-lineer ilişkilere dönüştürmektedir. Gizli katmanlar, modelin genelleme yeteneğini artırarak aşırı uyuma (overfitting) karşı direncini arttırmaktadırlar. Ancak, gizli katmanların sayısı ve büyüklüğü, modelin karmaşıklığı ve hesaplama maliyeti üzerinde etkili olan önemli tasarım parametreleri olarak ifade edilmektedir.

2.4.2.3. Çıktı Katmanı

Çıktı katmanı, aktivasyon fonksiyonları ve hata fonksiyonları kullanılarak modelin performansını değerlendiren önemli bir aşama olarak ifade edilmektedir. Doğru çıktı, modelin başarı derecesini belirlemekte ve modelin genelleme yeteneğini değerlendirmek için sıklıkla test verileri ile karşılaştırılmaktadır.

Derin öğrenmedeki çıktı katmanları, bir modelin öğrenme sürecinde elde ettiği bilgileri final bir çıktıya dönüştüren kritik bileşen olarak ifade edilmektedir. Bu katmanlar, modelin amaçladığı görev türüne bağlı olarak çeşitlilik gösterebilmektedir. Sınıflandırma görevlerinde, softmax aktivasyon fonksiyonları genellikle kullanılmakta ve modelin girdileri farklı sınıflar arasında olasılık dağılımına dönüştürülmektedir. Regresyon görevlerinde ise doğrudan bir çıktı üretme eğilimindedirler, bu da modelin sürekli bir tahmin yapmasına olanak tanımaktadır. Ayrıca, çıktı katmanları, transfer öğrenme gibi konseptlerle birleştirilerek, önceden eğitilmiş modellerin daha spesifik görevlerde kullanılmasına olanak tanımaktadırlar. Bu sayede, çıktı katmanları, derin öğrenme modellerinin geniş bir uygulama yelpazesinde başarıyla kullanılmasını sağlamaktadırlar.

2.5. Derin Öğrenme Türleri

İleri beslemeli ağlar ve geri beslemeli ağlar, derin öğrenme alanında öne çıkan temel yapı taşları arasında yer almaktadır. İleri beslemeli ağlar, veriyi bir yönde işleyen ve girişten çıkışa doğru ilerleyen katmanlardan oluşmaktadır. Gelişmiş örüntü tanıma ve sınıflandırma görevlerinde sıklıkla kullanılmaktadır. Öte yandan, geri beslemeli ağlar, geçmiş bilgileri saklama yeteneğine sahip olup, zamanla değişen veri üzerinde çalışmada avantaj sağlamaktadır. Dil işleme ve zaman serisi analizi gibi görevlerde etkin bir şekilde kullanılmaktadır. Bu iki tür, derin öğrenmenin geniş uygulama yelpazesinde önemli roller üstlenmektedir.

2.5.1. Derin İleri Beslemeli Ağlar

İleri beslemeli sinir ağları veya çok katmanlı algılayıcılar olarak da adlandırılan derin ileri beslemeli ağlar, derin öğrenme modellerinin başında gelmektedir. İleri beslemeli bir ağın amacı, belirli bir f^* fonksiyonuna yaklaşmaktır. Örneğin bir sınıflandırma için $y = f^*(x)$, bir x girdisini bir y kategorisiyle eşlemektedir. İleri beslemeli bir ağ, $y = f(x;\ \theta)$ eşlemesini tanımlamakta ve en iyi fonksiyon yaklaşımıyla sonuçlanan θ parametresinin değerini öğrenmektedir. Bu modeller ileri beslemeli olarak adlandırılır çünkü bilgi, x'ten başlayarak f'i tanımlamak için kullanılan ara hesaplamalardan geçmekte ve son olarak y çıktısına kadar değerlendirilen fonksiyon üzerinden ilerlemektedir (Goodfellow ve diğerleri, 2016: 168).

İleri beslemeli ağlar, girdi katmanından başlayarak bilgiyi bir yönde ileterek çalışmakta, birbirine doğrudan bağlı olmaları sayesinde, ardışık katmanlardan elde edilen çıktıları kullanarak belirli bir çıktı üretmektedir. Öğrenme sürecinde, her katmanın içerdiği öğrenilebilir ağırlıklar ve güncellenen sapma terimleri, ağın esnek ve adaptif olmasını sağlamaktadır.

İleri beslemeli ağlar, geniş bir uygulama yelpazesinde etkili bir şekilde kullanılmaktadır. Görüntü tanıma alanında, özellikle nesne tespiti ve yüz tanıma gibi görevlerde büyük başarı elde

etmektedirler. Finans sektöründe risk analizi, hisse senedi tahmini gibi karmaşık veri analizlerinde de ileri beslemeli ağlar etkin bir şekilde kullanılmaktadır.

2.5.2.Derin Geri Beslemeli Ağlar

Geri beslemeli ağları, giriş verilerini işlerken önceki bilgileri kullanabilen yapay zekâ modeli olarak ifade edilmektedir. Geleneksel ileri beslemeli ağlarından farklı olarak, geri beslemeli ağlar, döngüler aracılığıyla bilgiyi saklamakta ve bu sayede zaman içindeki ilişkileri modelleme yeteneğine sahip olmaktadır.

Geri beslemeli ağları, bilgiyi geçmişe doğru taşıma yeteneğine sahip olmalarıyla öne çıkmakta bu sayede, önceki adımlarda elde edilen bilgilerin sonraki adımlarda kullanılabilmesi sağlamaktadır. Bu özellik ile geri beslemeli ağlar, zamanla değişen veri setlerinde veya birbiriyle ilişkili olayları içeren görevlerde sıklıkla tercih edilmektedir.

Geri beslemeli sinir ağlarının birçok avantajı bulunmaktadır. Özellikle dil modelleme, metin üretimi, konuşma tanıma gibi doğal dil işleme görevlerinde başarıyla kullanılmaktadır. Ayrıca, zaman serisi verilerinin analizi, finansal tahminler ve çeşitli sinyal işleme uygulamalarında da etkin bir biçimde kullanılmaktadır.

2.6. Gizli Katman ve Nöron Sayısının Belirlenmesi

Bu bölümde literatürde gizli katman ve nöron sayısının belirlenmesinde önerilen yöntemler ve literatürde yapılmış çalışmalar ele alınmıştır.

Syaharuddin, Fatmawati ve Suprajitno (2022) gerçekleştirdikleri çalışmada gizli nöron sayılarının belirlenmesinde Denklem 2.12'de yer alan formüllerin performanslarını MSE, R ve Doğruluk kriterlerine göre karşılaştırmışlardır. N_z gizli katmandaki nöron sayısını, N_x girdi katmanındaki nöron sayısını, N_y çıktı katmanındaki nöron sayısını, N_t eğitim setindeki veri adetini belirtmektedir.

Çalışmada Denklem 2.12'de belirtilmiş 7 formülü sıcaklık verilerine uygulanmış, 1 numaralı formülün eğitim verisi için %97,79 ve test verisi için %99,99 doğruluk sağlayarak en başarılı olduğu tespit edilmiştir.

$$
\begin{aligned}
&1 \quad N_z = 2 \times N_x + 1 \\
&2 \quad N_z = \log_2 N_x \\
&3 \quad N_z = \frac{N_x + N_y}{2} \\
&4 \quad N_z = \frac{2}{3} N_x \\
&5 \quad N_z = 2 \times N_x \\
&6 \quad N_z = \frac{N_x}{2} + 1 \\
&7 \quad N_z = \frac{1}{2}\left(N_x + N_y\right) + \sqrt{N_t}
\end{aligned}
\tag{2.12}
$$

Vujicic, Matijevic, Ljucovic, Balota ve Sevarac (2016) gerçekleştirdikleri çalışmada gizli nöron sayısının belirlenmesinde Denklem 2.13'de yer alan formüllerin performanslarını MSE kriterine göre karşılaştırmışlardır. N_h gizli katmandaki nöron sayısını, N_i girdi katmanındaki nöron sayısını, N_o çıktı katmanındaki nöron sayısını, N_t eğitim setindeki veri adetini belirtmektedir.

$$
\begin{aligned}
&1 \quad N_h = \frac{\sqrt{1 + 8N_i} - 1}{2} \\
&2 \quad N_h = N_i - 1 \\
&3 \quad \begin{cases} N_h = \dfrac{N_t}{N_i \log N_t}, & \dfrac{N_t}{N_i} > 30 \\ N_h = \dfrac{N_t}{N_i}, & \dfrac{N_t}{N_i} \le 30 \end{cases} \\
&4 \quad N_h = \sqrt{N_i N_o} \\
&5 \quad N_h = \log\left(N_i + 1\right) - N_o \\
&6 \quad N_h = \frac{\left(4N_i^2 + 3\right)}{N_i^2 - 8}
\end{aligned}
\tag{2.13}
$$

Çalışmada LAMS projesi kapsamında Karadağ'daki Lovćen dağında yıldırım araştırması için kurulan izleme istasyonundan alınan veriler kullanılmış, elde edilen sonuçlar kapsamında kullanılan yöntemlerin farklı veri setlerinde farklı sonuçlar verdiği tespit edilmiştir. Kullanılan formüllerin az sayıda girdi nöron sayısı için iyi sonuçlar verdiği belirtilirken, fazla sayıda girdi nöron sayısında karşılaştırılabilir sonuçlar vermediği belirtilmiştir.

2.6.1. Huang Önerisi

Huang ve Babri 1997 yılında gerçekleştirdikleri çalışmada sigmoid fonksiyonu gibi düzenli aktivasyon fonksiyonları için tek bir gizli katmanın kullanımın yeterli olduğunu ancak nöron sayısının eğitim örneği sayısı kadar olabileceğini belirtmiştir. Huang 2003'te gerçekleştirdiği çalışmada nöron

sayısı azaltmak için ikinci bir gizli katmanın kullanılabileceğini, tek gizli katmanlı sistemlerde nöron sayısının belirlenmesinde Denklem 2.14'ün, iki gizli katmanlı sistemlerde ilk gizli katmanda Denklem 2.15'in ve ikinci gizli katmanda Denklem 2.16'nın kullanılmasıyla düşük bir hata ile nöron sayısının belirlenebileceğini tespit etmiştir (Stathakis, 2009).

Tek gizli katmanlı sistemde m çıktı nöron sayısı ve N örnek boyutu ile gizli katmandaki nöron sayısı belirlenmektedir.

$$2\sqrt{(m+2)N} \tag{2.14}$$

Huang iki gizli katmanlı sistemde ilk gizli katmandaki nöron sayısının belirlenmesinde,

$$\sqrt{m+2N}+2\sqrt{N/(m+2)} \tag{2.15}$$

İkinci katmandaki gizli nöron sayısının belirlenmesinde ise,

$$m\sqrt{N/(m+2)} \tag{2.16}$$

ifadesini önermektedir.

2.6.2. Heaton Önerisi

Heaton 2008 yılında yayınladığı çalışmada, derin öğrenmede gizli nöron sayısının belirlenmesinin, gizli katman sayısının belirlenmesinden daha önemli olduğunu belirmiştir. Heaton çalışmasında doğrusal fonksiyonların kullanıldığı ağlarda gizli katman kullanılmasına gerek olmadığını, sonlu bir alanda sürekli değerler alan bir fonksiyon için tek bir gizli katmanın yeterli olacağını ve rasyonel aktivasyon fonksiyonunun kullanıldığı ve isteğe bağlı bir karar sınırı olan çalışmalarda ise iki gizli katman kullanılabileceğini belirtmiştir.

Heaton'a göre (2008) gizli nöron sayısının saptanması amacıyla aşağıda belirtilen üç kural kullanılabilir.

- Gizli nöron sayısı, giriş katmanı boyutu ile çıktı katmanının boyutu arasında olmalıdır.
- Gizli nöron sayısı için Denklem 2.17 kullanılabilir. i girdi katman boyutunu o ise çıktı katman boyutunu temsil etmektedir.

$$\frac{2}{3}i+o \tag{2.17}$$

- Gizli nöron sayısı, girdi katman boyutunun 2 katından az olmalıdır.

Ayrıca Heaton çalışmasında, yukarıda belirtilen üç kuralın göz önünde bulundurulması gereken birer başlangıç noktası olduğunu belirtmekte ve bu üç kurala göre ağın en iyi performansına ulaşabilmesi

için bu başlangıç noktalarından yararlanarak ağı oluşturan kişinin deneme yanılma yöntemini uygulaması gerektiğini belirtmiştir.

2.6.3. Temel Bileşenler Analizi

Ibnu Choldun, Santoso ve Surendro'nun 2020 yılında gerçekleştirmiş oldukları çalışmada gizli katman sayısının belirlenmesinde Temel Bileşenler Analizinin kullanılabileceğini belirtmişlerdir. Giderek daha karmaşık hale gelen özellik, alınan daha yüksek bilgi içeriğini temsil etmektedir. Temel bileşenler analizindeki yüksek bilgi, yüksek değişkenliği olan ana bileşen ile temsil edilmektedir. Başka bir deyişle, gizli katmanlarda gittikçe daha karmaşık hale gelen ilişkiler, daha yüksek değişkenlik olan temel bileşenlere uygundur. Sonuç olarak, sinir ağındaki gizli katmanların sayısı temel bileşen analizindeki temel bileşenlerin sayısına göre belirlenebilir (Ibnu Choldun R., Santoso ve Surendro, 2020).

Temel Bileşenler Analizinin adımları aşağıdaki gibidir.

1. Veri matrisi $\left(X = \left[x_1, x_2, \ldots, x_N\right]\right)$, N toplam örnek sayısını ve x_i i örneğine ait değeri temsil eder.
2. Tüm örneklere ait ortalama aşağıdaki gibi hesaplanır:

$$\mu = \frac{1}{N}\sum_{i=1}^{N} x_i \tag{2.18}$$

3. Tüm örnek değerleri ortalamadan çıkartılır:

$$D = \left\{d_1, d_2, \ldots, d_N\right\} = \sum_{i=1}^{N}\left(x_i - \mu\right) \tag{2.19}$$

4. Kovaryans matrisi aşağıdaki gibi hesaplanır:

$$\Sigma = \frac{1}{N-1} D \times D^T \tag{2.20}$$

5. Özvektörler V ve özdeğerleri λ, kovaryans matrisinden hesaplanır.
6. Özvektörleri karşılık gelen özdeğere göre sıralanır.
7. En büyük özdeğerleri $W = \left\{v_1, \ldots, v_k\right\}$ olan özvektörleri seçin. Seçilen özvektörler $\left(W\right)$ temel bileşenler analizi projeksiyon alanını temsil eder.
8. Tüm örnekler temel bileşen analizinden $\left(W\right)$ daha düşük boyutsal alana aşağıdaki gibi yansıtılır:

$$Y = W^T D \tag{2.21}$$

Oluşturulan temel bileşenlerin sayısı orijinal değişkenlerin sayısına eşittir. Kullanılan temel bileşen sayısının seçiminde farklı yöntemler bulunmaktadır. Derin öğrenmede temel bileşen sayısının seçimi yani gizli katman sayısının belirlenmesinde, açıklanan varyans miktarına göre karar verilebilir (Ibnu Choldun R. vd., 2020).

2.6.4. Genetik Algoritma

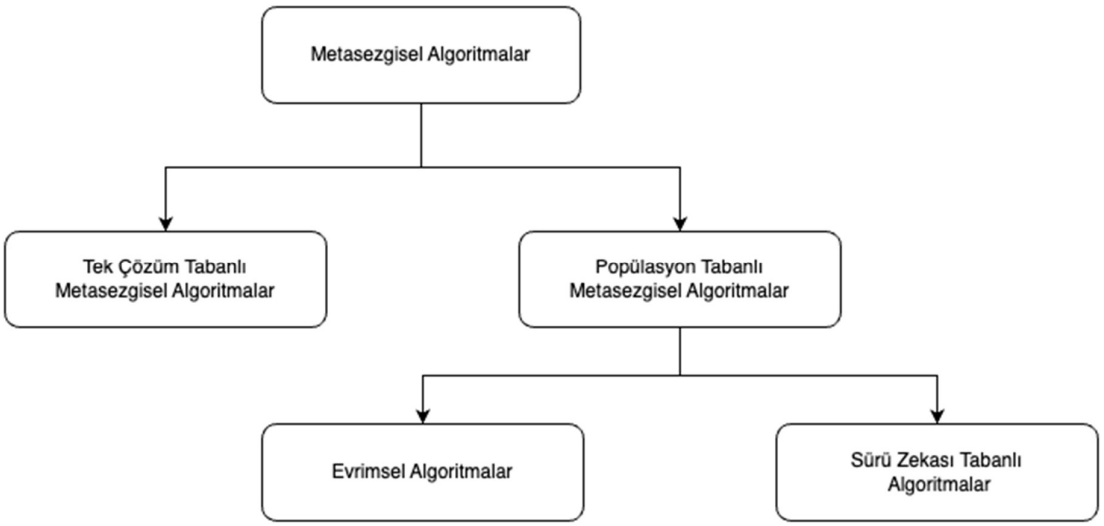
Derin öğrenme yöntemlerinde, amaç fonksiyonunun minimizasyonu, ağırlıkların, gizli katman ve gizli nöron sayılarının belirlenmesi problemlerinde genetik algoritmalar kullanılabilmektedir. Genetik algoritma ile derin öğrenme ağı eğitilerek amaç fonksiyonundaki hata miktarı en aza indirgenebilir.

Genetik algoritmalar bir sonraki bölümde detaylı olarak ele alınmaktadır.

3. GENETİK ALGORİTMA

Bu bölümde Genetik Algoritma optimizasyon yöntemi detaylı bir şekilde ele alınmış ve Genetik Algoritma kavramı, klasik genetik algoritma, reel sayılı genetik algoritma, çok amaç fonksiyonlu genetik algoritma yöntemleri ayrıntılı olarak açıklanmıştır.

Optimizasyon, bir takım matematiksel modelleri kullanarak bir sistemdeki "en iyi" çıktıyı verecek girdi değerlerini bulmayı amaçlamaktadır. Optimizasyon problemleri, birden fazla çözümü olan ve bu çözümlerin farklı kalitede olduğu durumlar olarak açıklanabilir. Metasezgisel algoritmalar, geleneksel yöntemler kullanılarak çözülemeyen karmaşık problemler için en uygun çözümü bulmak amacıyla kullanılan optimizasyon algoritmalarıdır (M. Almufti, Ahmad Shaban, Arif Ali, Ismael Ali ve A. Dela Fuente, 2023). Metasezgisel algoritmalar, ekonomi, mühendislik, politika, yönetim gibi farklı alanlardaki optimizasyon problemlerinin çözümünde kullanılmaktadır (V. Kumar, Chhabra ve Kumar, 2014). Metasezgisel algoritmaların çoğu biyolojik evrim sürecinden, sürü davranışından ve fizik kanunlarından esinlenmektedir. Bu algoritmalar genel olarak tek çözüm ve popülasyon tabanlı metasezgisel algoritmalar olmak üzere iki kategoriye ayrılır. Metasezgisel algoritmaların kategorik yapısı Şekil 19'da verilmiştir.



Şekil 11. Metasezgisel Algoritmalar (Katoch, Chauhan ve Kumar, 2021)

Tek çözüm tabanlı metasezgisel algoritmalar tek bir aday çözüm kullanır ve bu çözümü yerel arama kullanarak iyileştirir. Ancak tek çözüm tabanlı metasezgisel algoritmalar ile elde edilen çözüm, yerel optimumlara takılabilmektedir (Chahar ve Kumar, 2017). En sık kullanılan tek çözüm tabanlı metasezgisel algoritmalar; benzetilmiş tavlama (simulated annealin), tabu arama (tabu search), mikrokanonik tavlama ve güdümlü yerel aramadır.

Popülasyon tabanlı metasezgisel algoritmalar, arama işlemi sırasında birden fazla aday çözüm kullanmaktadır. Bu metasezgisel algoritmalar, popülasyondaki çeşitliliği korur ve çözümlerin yerel optimumlara takılmasını önlemektedir. En sık kullanılan popülasyon tabanlı metasezgisel algoritmalar, genetik algoritma, parçacık sürüsü optimizasyonu, karınca kolonisi optimizasyonu, benekli sırtlan optimize edici, imparator penguen optimize edici ve martı optimizasyonudur.

Genetik algoritma doğadaki evrim sürecini taklit ederek, popülasyon tabanlı aramalarla global optimum çözümleri bulma başarısını yükselten ve her türlü optimizasyon problemine uygulanabilen güçlü ve esnek bir optimizasyon yöntemidir.

Genetik algoritma kavramının tarihçesine bakıldığında 1960'lı yıllarda Racherberg tarafından ortaya atılan evrimsel hesaplama adımları ile ortaya çıkmıştır. 1975'lerde Holland ve arkadaşları tarafından "Genetik Algoritma" kavramı literatüre kazandırılmıştır. 1992'de John Koza "Genetik Programlama" adını verdiği yöntemle genetik algoritma kavramına yeni bir boyut kazandırmıştır.

Genetik Algoritmalar ilk kez Holland (1992) tarafından önerilmiştir. Holland, kromozom olarak adlandırdığı ikili kodlanmış dizilerinden oluşan bir popülasyon üzerinde uyum ile oranlı seçme, rastlantısal çaprazlama ve bit mutasyonu işlecini kullanarak evrimsel ilkeleri taklit eden ilk genetik algoritmayı önermiştir. Canlıların üreme ve kalıtım mekanizmalarını taklit eden ve doğal seçilim prensiplerine dayanan evrimsel hesaplama algoritmaları "Genetik Algoritma" olarak tanımlamıştır (Cebeci, 2021; Katoch vd., 2021).

Genetik algoritmalar, sayısal optimizasyon ve gezgin satıcı problemi, devre tasarımı, iş atölyesi çizelgeleme ve video ve ses kalitesi optimizasyonu gibi kombinasyonel optimizasyon problemleri de dahil olmak üzere çok çeşitli optimizasyon görevlerinde kullanılmıştır (Nandal ve Nandal, 2013).

3.1. Genetik Algoritma Kavramı

Holland (1992) gerçekleştirdiği çalışmada, genetik algoritmaların evrimsel süreçlerden ilham alınarak tasarlanmış olan bir yapay zeka tekniği olarak ifade etmektedir. Bu algoritmalar, bir doğal seçilim sürecine benzer şekilde, bir popülasyon içerisindeki bireylerin genetik materyallerini mutasyon, çaprazlama gibi operatörlerle değiştirerek yeni bireyler üretmektedirler. Bu yeni bireyler, önceden belirlenmiş bir amaç fonksiyonunu en iyi şekilde karşılayacak biçimde seçilmektedirler. Bu şekilde popülasyonun giderek daha iyiye doğru evrimleşmesi sağlanmaktadır. Genetik algoritmaların bu evrimsel süreçleri, birçok optimizasyon, tasarım ve öğrenme probleminin çözümünde etkili bir şekilde kullanıldığını belirtmektedir.

Goldberg (1989) genetik algoritmaları matematiksel bir modelleme aracı olarak tanımlamaktadır. Ona göre, genetik algoritmalar, bir optimizasyon problemi çözme tekniği olarak

kullanılabilir ve doğal seçilim mekanizmasından esinlenen bir yaklaşımdır. Genetik algoritma, bir başlangıç popülasyonu oluşturur ve bu popülasyondaki bireylerin genetik materyallerini mutasyon, çaprazlama gibi operatörlerle değiştirerek yeni bireyler oluşturur. Bu yeni bireyler, uygunluk fonksiyonları kullanılarak ölçülen bir uygunluk seviyesine göre sıralanır ve en iyi bireyler bir sonraki jenerasyon için seçilmektedir. Bu işlem, belirlenen bir duruma kadar tekrar edilir ve sonuçta en iyi çözüm bulunur. Goldberg, genetik algoritmaların özellikle karmaşık, çok boyutlu ve belirsizlik içeren optimizasyon problemlerinin çözümünde başarılı olduğunu vurgulamaktadır.

Genetik algoritmalar, canlıların yaşam sürecini örnek almaktadırlar ve iyi nesillerin yaşamları koruması, kötü nesillerin yok olması ilkesine dayanmaktadır. Matematiksel modellemenin yapılamadığı veya kesin çözümün olmadığı problemlerde genetik algoritmadan yararlanılır (Elmas, 2018).

Genetik algoritmalar, canlılardaki evrim süreci ve doğal seçilim ilkesinin yanında genetik süreçlerin taklit edilmesi yoluyla çalışır. Doğanın taklit edilmesi yoluyla geliştirildikleri için yapay zeka algoritmalarına dahil edilirler, ayrıca genetik algoritmalar optimizasyon tekniklerinin de bir parçasıdır (Hillier ve Lieberman, 2021).

Genetik algoritmalar, optimizasyon süreci boyunca amaç fonksiyonuna gönderilen aday çözümlerin kalitesi hakkında aldığı bilgiyi daha iyi aday çözümler üretmekte kullanmakta ve hataları deneme yanılma yoluyla gidermektedir. Yüzlerce aday çözümün yine yüzlerce kez amaç fonksiyonuna gönderilmesi ve geri dönen değerlere bağlı olarak genetik operatörlerin işletilmesi, birçok klasik optimizasyon yönteminden daha fazla işlem süresini gerektirmektedir (Satman, 2016).

3.2. Genetik Algoritma Alanında Bibliyometrik Analiz

WOS'ta "genetik algoritma" anahtar kelimesi kullanılarak yayınlanmış 158.884 adet akademik çalışma bulunmaktadır. Literatür taramasında veriler WOS'tan elde edilirken bir konuda maksimum 100.000 adet çalışmanın verilerine ulaşılabilmektedir. 100.000 üzerindeki verilere ulaşımı WOS tarafından engellenmektedir. Bu çalışmaların WOS tarafından belirlenmiş kategorilere göre dağılımları Şekil 12'de verilmiştir.



Şekil 12. Kategorilere Göre Yayın Dağılımları (Genetik Algoritma)

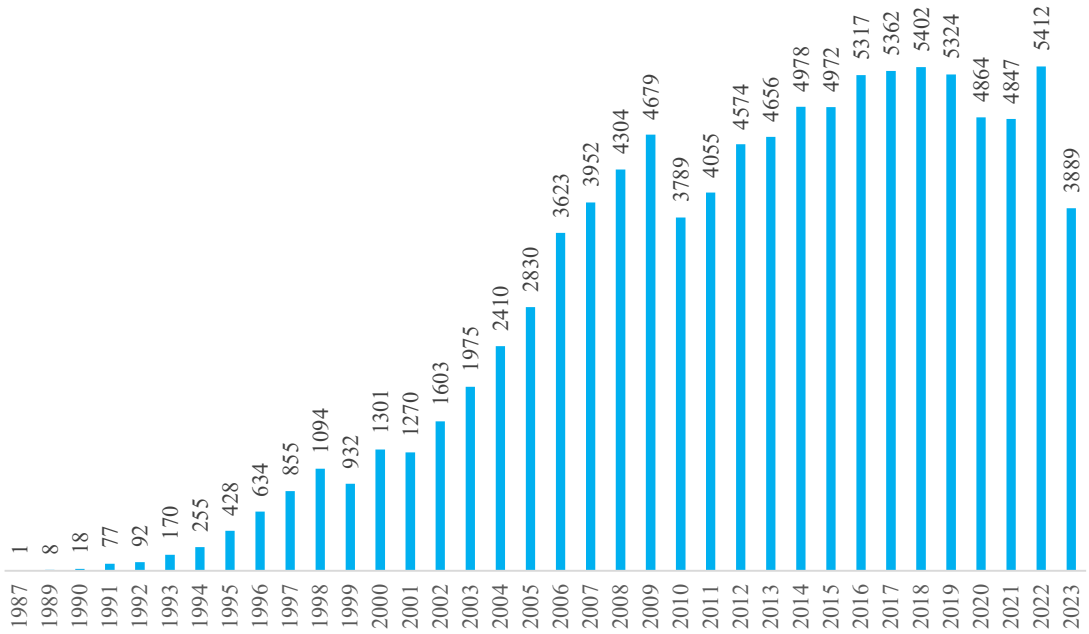
Şekil 12 incelendiğinde "Genetik Algoritma" anahtar kelimesini kullanan 158. 884 çalışma birden fazla kategoride yer alabildiğinden dolayı toplam 329.105 adet veri üzerinden çalışma gerçekleştirildiği ve bu çalışmaların %11,54,'ü Elektrik Elektronik Mühendisliği kategorisinde, %9,79'u Bilgisayar Bilimi Yapay Zekâ kategorisinde, %5,44'ü Bilgisayar Bilimi Teori Yöntemleri kategorisinde, 4,78'i Multidisipliner Bilgisayar Bilimi kategorisinde, 4,33'ü Bilgisayar Bilimi Bilişim Sistemleri kategorisinde yer aldığı görülmektedir. Bahsedilen 5 kategori yayınlanan çalışmaların %35,89'unu kapsamaktadır.

En çok yayın yapılan 25 kategori bu alanda yapılmış çalışmaların %75,34'ünü kapsamaktadır. Bundan dolayı Şekil 12'de ilk 25 kategoriye yer verilmiştir Yayın türüne göre yayın dağılımları ise Tablo 3'te verilmiştir.

Tablo 3. Yayın Tipi (Genetik Algoritma)

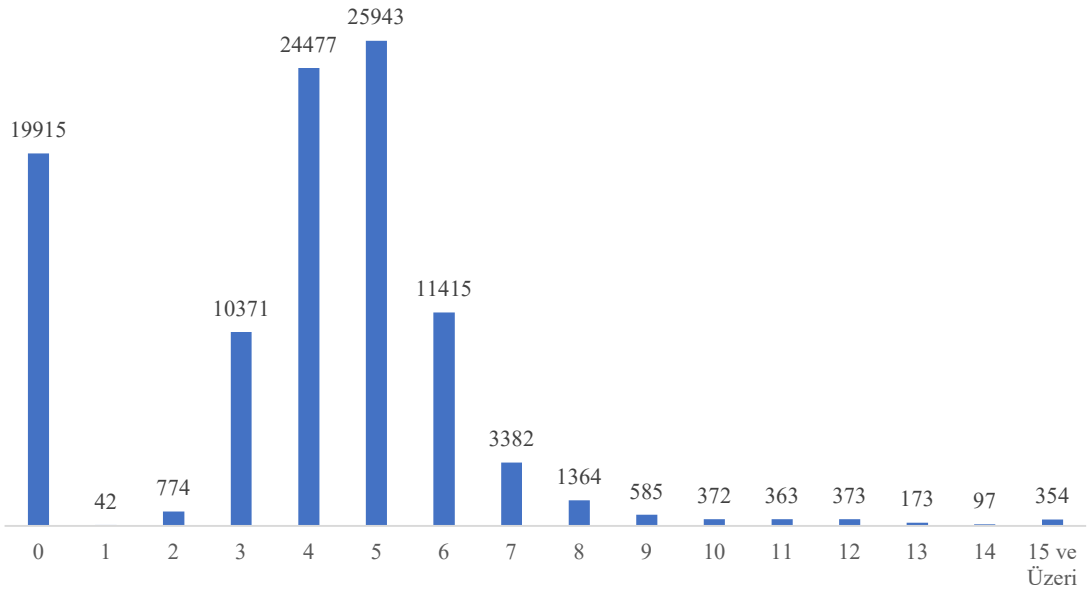
Yayın Tipi	Yayın Sayısı	Yüzde
Makale	57397	57,40
Bildiri Metni	41315	41,32
İnceleme	667	0,67
Toplantı Özeti	201	0,20
Düzeltme	183	0,18
Editoryal Materyal	125	0,13
Mektup	82	0,08
Kitap	30	0,03
Genel Toplam	100000	100,00

Tablo 3'te görüldüğü gibi Genetik Algoritma hakkında WOS'ta yayınlanan 100000 çalışmanın %57,40'ı makale türünde, %41,32'si bildiri türünde ve %0,67'si inceleme türündedir. Bu üç tür tüm çalışmaların %99,38'ini oluşturmaktadır. Yıllara göre yayın dağılım grafiği Şekil 12'de verilmiştir.



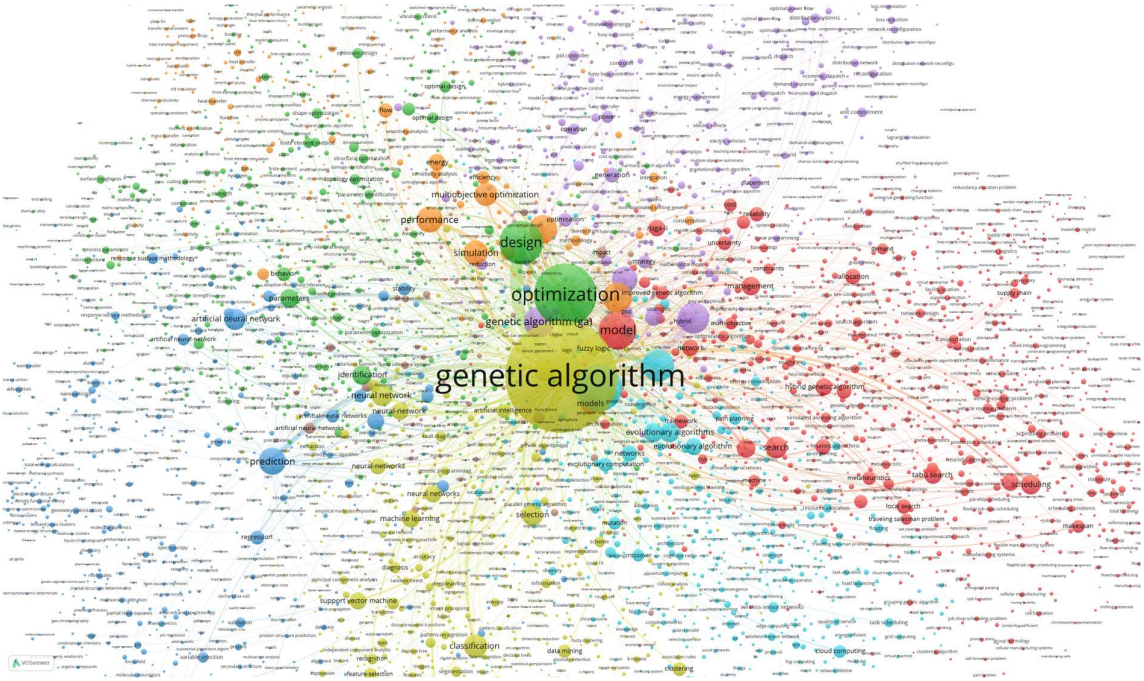
Şekil 13. Yıllara Göre Yayın Sayıları (Genetik Algoritma)

Şekil 13 incelendiğinde Genetik Algoritma alanında ilk çalışmaların 1987 yılında yapılmaya başlandığı görülmektedir. Teknolojik gelişmeler ve bilgisayar donanımlarının güçlenmesi ile yayın sayısı artış göstermiştir. Ortaya çıktığı günlerden beri çıkış gösteren derin öğrenme konusu son yıllarda artışını kaybetmekte ve durağanlığa geçiş yapmaktadır. Anahtar kelime sayısına göre yayın dağılımları Şekil 14'te verilmiştir.



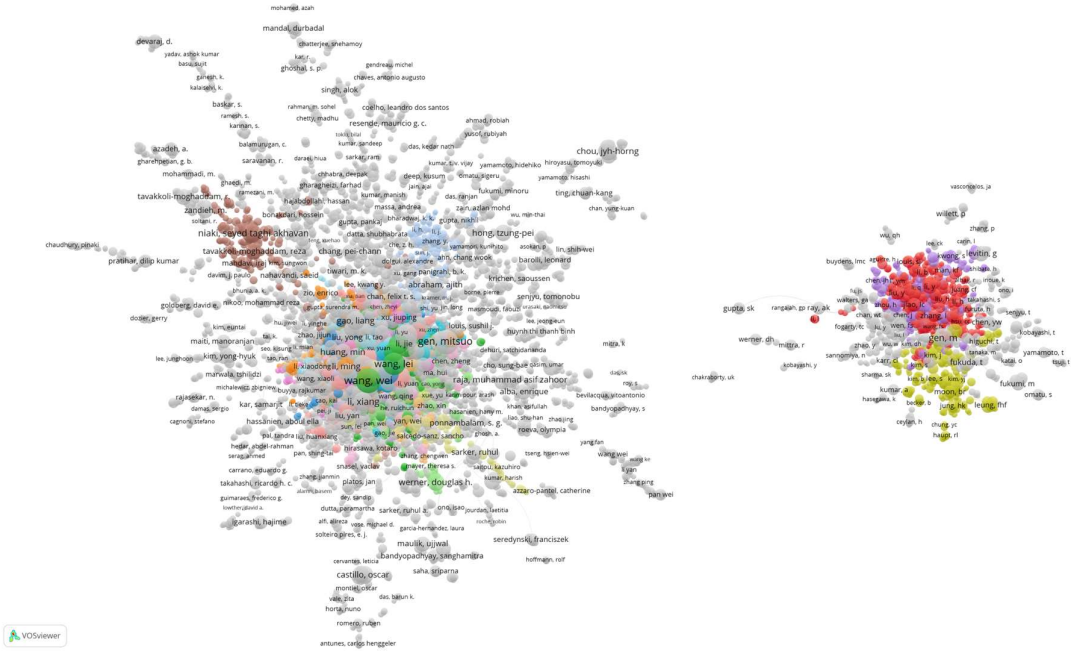
Şekil 14. Anahtar Kelime Sayısına Göre Yayın Sayısı Dağılımı (Genetik Algoritma)

Şekil 14 incelendiğinde çalışmaların %10,37'sinde 3 anahtar kelime kullanıldığı, %24,48'sinde 4 anahtar kelime kullanıldığı, %25,94'ünde 5 anahtar kelime kullanıldığı %11,42'sinde 6 anahtar kelime kullanıldığı tespit edilmiştir. Çalışmaların %72,21'inde 3 ile 6 arasında anahtar kelime kullanılmıştır. Çalışmaların %19,92'sinde genetik algoritma haricinde herhangi bir anahtar kelime kullanılmamıştır. Genetik Algoritma anahtar kelimesi kullanılan çalışmalarda anahtar kelimelere ait sosyal ağ analizi Şekil 15'te verilmiştir.



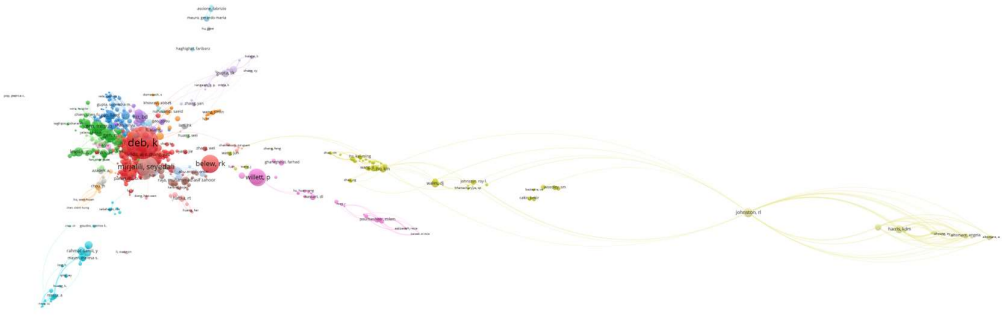
Şekil 15. Anahtar Kelimelere Ait Sosyal Ağ Analizi (Genetik Algoritma)

Şekil 15 incelendiğinde 100.000 adet çalışmada toplam 411.857 adet anahtar kelime kullanıldığı ve bu 411.857 adet anahtar kelimenin 1173 kümede birleştiği görülmektedir. Şekilde kümeler renkler ile gösterilmiştir. En çok kullanılan anahtar kelime bibliyometrik analizde de kullanılan genetik algoritma anahtar kelimesidir. Genetik algoritma haricinde, optimizasyon, tasarım, model, sinir ağları, evrim algoritmaları, simülasyon, performans, tahmin, sınıflandırma, anahtar kelimelerinin de sıklıkla çalışmalarda kullanıldığı tespit edilmiştir. Yazarlar arasındaki ilişkileri incelemek için yazarlar arasında da sosyal ağ analizi gerçekleştirilmiş ve Şekil 16'da verilmiştir.



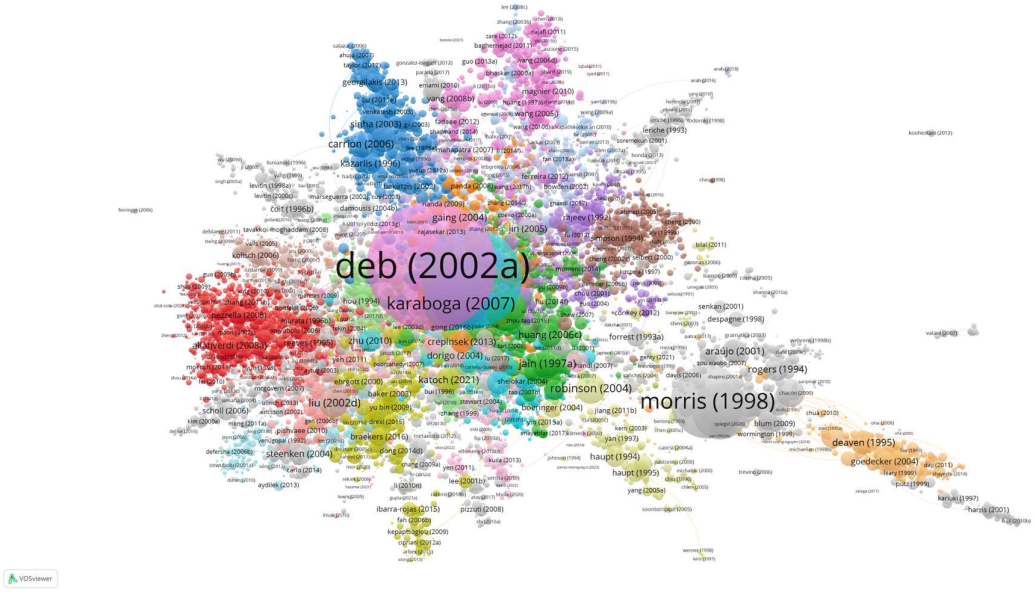
Şekil 16. Yazarlara Ait Sosyal Ağ Analizi (Genetik Algoritma)

Şekil 16 incelendiğinde 100.000 adet çalışmada toplam 329.021 adet farklı yazarın yer aldığı tespit edilmiştir. 329.021 adet yazarın 110 kümede birleştiği tespit edilmiştir. Sosyal ağ analizi sonuçları incelendiğinde yazarların iki ayrık kümede yoğunlaştığı ve ayrık kümelerin birbiri ile ilişkili bir yapıda olduğu ve çoğunlukla merkezde konumlandığı görülmektedir. Genetik algoritma anahtar kelimesi kullanan çalışmalar birleştirilerek yazarların bu alanda almış oldukları atıflar sosyal ağ analizi ile incelenmiş ve sonuçlar Şekil 17'de verilmiştir.



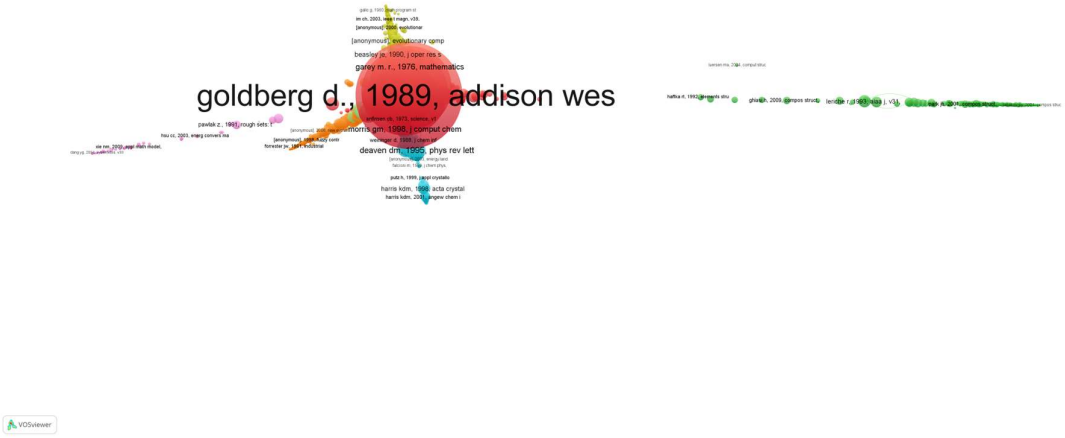
Şekil 17. Yazarların Almış Oldukları Atıflara Ait Sosyal Ağ Analizi (Genetik algoritma)

Şekil 17 incelendiğinde K. Deb'in genetik algoritma alanında 29558 adet atıfla en fazla atıf alan yazar olduğu tespit edilmiştir. Genetik algoritma alanında yayın yapmış 329.021 adet yazar toplamda 2.533.421 adet atıf almıştır. Toplam atıf miktarı yazar sayısına oranlandığında yazar başına ortalama 7,70 atıf düşmektedir. Alınan atıflara göre yazarların 26 kümede birleştiği görülmektedir. Atıf alan çalışmalara ait sosyal ağ analizi Şekil 18'de verilmiştir.



Şekil 18. Yayınların Almış Oldukları Atıflara Göre Sosyal Ağ Analizi (Genetik Algoritma)

Şekil 18 incelendiğinde 100.000 adet yayından 73.739 tanesi atıf almıştır. Atıf alan 73.739 adet yayının 35 kümede birleştiği tespit edilmiştir. Derin öğrenme alanında en çok atıf alan çalışma 29.790 adet atıf ile K.Deb ve arkadaşlarının 2002 yılında yayınlamış olduğu çalışmadır. İkinci sırada ise 16.831 adet atıf ile Storn ve Price'ın 1997 yılında yayınlamış olduğu çalışma bulunmaktadır. Üçüncü sırada ise G. M. Morris ve arkadaşlarının 1998 yılında yayınlamış oldukları çalışma yer almaktadır. 100.000 adet çalışmada kullanılan referanslara ait sosyal ağ analizi Şekil 19'da verilmiştir.



Şekil 19. Kullanılan Referanslara Ait Sosyal Ağ Analizi (Genetik Algoritma)

Şekil 19 incelendiğinde 100.000 adet çalışmada toplamda 1.331.301 adet referans kullanılmıştır. Kullanılan referans miktarı yayın sayısına oranlandığında yayınların ortalama 13.31 adet referans kullandığı tespit edilmiştir. Kullanılan 1.331.301 adet referansın 16 adet kümede birleştiği görülmektedir. D. Golberg'in 1989 yılında yayınlamış olduğu yayın, 10.299 çalışmanın kaynakçasında yer almaktadır.

3.3. Genetik Algoritma Çeşitleri

Genetik algoritmalar, gerçek ve ikili kodlu genetik algoritmalar, çok amaçlı genetik algoritmalar, paralel genetik algoritmalar, kaotik genetik algoritmalar ve karma genetik algoritmalar olmak üzere çeşitlere ayrılmaktadır.

3.3.1. Klasik Genetik Algoritma

Klasik Genetik Algoritma, doğal seçilimden esinlenen bir optimizasyon algoritmasıdır. En uygun olanın hayatta kalması prensibine dayanan popülasyon tabanlı bir arama algoritmasıdır (Michalewicz, 1996). Yeni popülasyonlar, popülasyonda bulunan bireyler üzerinde genetik operatörlerin iteratif kullanımı ile üretilmektedir (Katoch vd., 2021).

Kromozom gösterimi, seçim, çaprazlama, mutasyon ve uygunluk fonksiyonu hesaplaması genetik algoritmanın temel unsurlarını oluşturmaktadır. Genetik algoritmanın çalışma prosedürüne ait akış diyagramı Şekil 20'de verilmiştir.

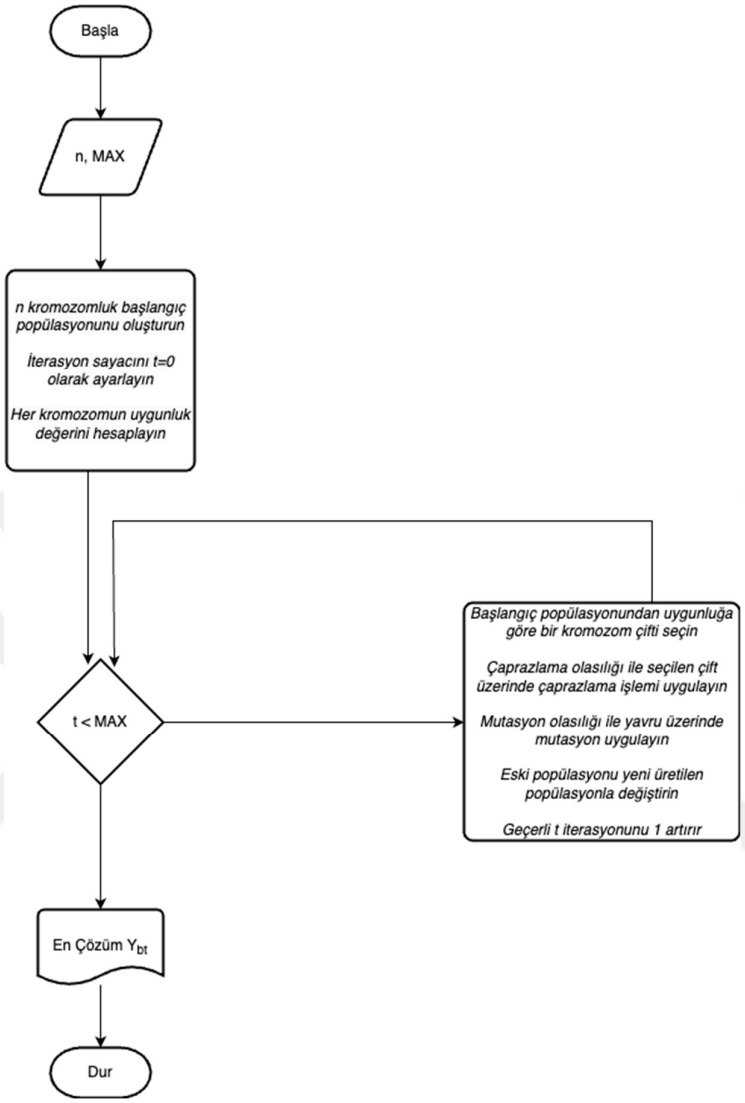
N kromozomdan oluşan bir popülasyon *(Y)* rastgele başlatılmaktadır. *Y*'deki her kromozomun uygunluğu hesaplanmaktadır. Uygunluk değerine göre *Y* popülasyonundan *C1* ve *C2* olmak üzere iki kromozom seçilmektedir. *C1* ve *C2* üzerinden çaprazlama olasılığı *p(C)* olan tek noktalı çaprazlama operatörü uygulanarak *O* yavrusu üretilmektedir. Daha sonra, üretilen yavru *(O)* üzerinden mutasyon olasılığı *p(M)* olan tek tip mutasyon operatörü uygulanarak O' üretilmektedir. Yeni yavru O' yeni popülasyona yerleştirilmektedir. Seçim, çaprazlama ve mutasyon işlemleri yeni popülasyon tamamlanana kadar mevcut popülasyon üzerinden tekrarlanmaktadır.

Genetik algoritma, çaprazlama ve mutasyon olasılıkları aracılığıyla arama sürecini dinamik olarak değiştirerek optimum çözüme ulaşmaktadır. Genetik algoritma kodlanmış genleri değiştirilebilmekte, birden fazla bireyi değerlendirebilmekte ve birden fazla optimum çözüm üretebilmektedir. Bu nedenle genetik algoritma diğer yöntemlere göre daha iyi bir optimum çözüm arama yeteneğine sahiptir.

Ebeveyn kromozomlarının çaprazlanmasından üretilen yavruların genetik şemaları, ebeveyn kromozomlarını ortadan kaldırmakta ve çaprazlama formülü aşağıdaki gibi tanımlanmaktadır.

$$R = \left(G + 2\sqrt{g}\right) \Big/ 3G \tag{3.1}$$

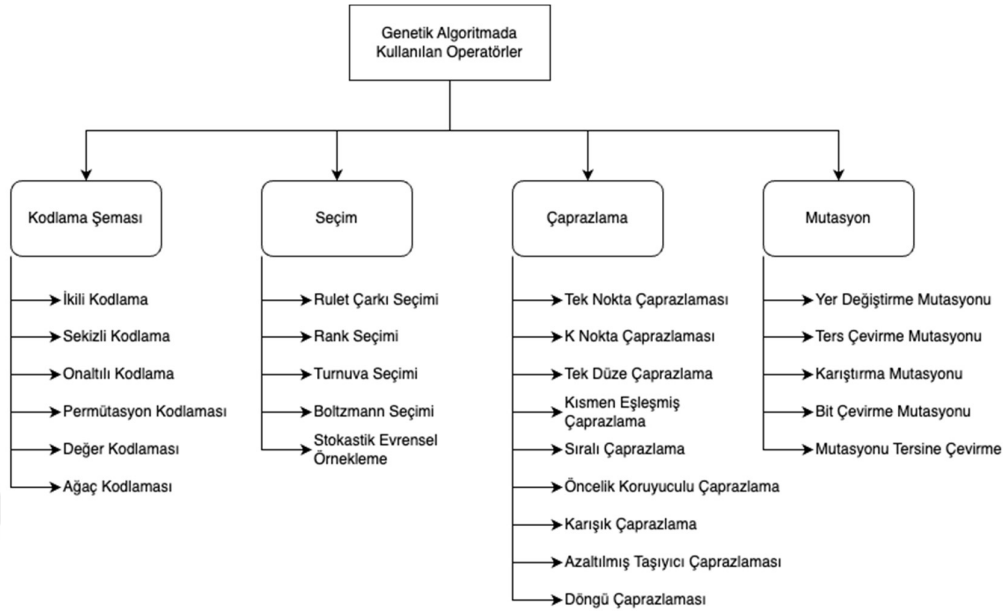
Burada g nesil sayısı, G ise popülasyon tarafından belirlenen toplam evrimsel nesil sayısıdır. Denklem 3.1'den R'nin dinamik olarak değiştiği ve evrimsel nesil sayısı arttıkça arttığı görülmektedir. Genetik algoritmanın ilk aşamasında bireyler arasındaki benzerlik çok düşüktür. Yeni popülasyonun bireylerin mükemmel genetik şemasını yok etmemesini sağlamak için R değeri düşük olmalıdır. Evrimin sonunda, bireyler arasındaki benzerlik çok yüksek olduğun gibi R değeri de yüksek olmalıdır (D. Liu, 2019).



Şekil 20. Genetik Algoritma Akış Diyagramı

3.3.1.1. Genetik Operatörler

Bu bölümde genetik algoritmada kullanılan operatörler açıklanmaktadır. Şekil 21'de genetik algoritmada kullanılan operatörler ve operatörlerde en sık kullanılan yöntemler verilmiştir.



Şekil 21. Genetik Algoritmada Kullanılan Operatörler (Katoch vd., 2021)

3.3.1.2. Kodlama Şeması

Hesaplama problemlerinin çoğunda kodlama şeması (belirli bir forma dönüştürme) önemli bir rol oynar. Verilen bilgi belirli bir bit dizisi içinde kodlanmalıdır. Kodlama şemaları problem alanına göre farklılaşmaktadır. Sıklıkla kullanılan kodlama şemaları ikili, sekizli, on altılı, permütasyon, değer tabanlı ve ağaç kodlamadır. Kodlama şemalarına ait artılar, eksiler ve uygulama alanları Tablo 4'te verilmiştir.

Tablo 4. Farklı Kodlama Şemalarının Karşılaştırılması (Katoch vd., 2021)

Kodlama Şeması	Artıları	Eksileri	Uygulama Alanları
İkili Kodlama	Uygulaması Kolay Daha Hızlı Çalışma	Ters Çevirme Operatörünü Desteklemez	İkili Kodlamayı Destekleyen Problemler
Sekizli Kodlama	Uygulaması Kolay	Ters Çevirme Operatörünü Desteklemez	Sınırlı Kullanım
Onaltılı Kodlama	Uygulaması Kolay	Ters Çevirme Operatörünü Desteklemez	Sınırlı Kullanım
Permütasyon Kodlaması	Ters Çevirme Operatörünü Destekler	İkili Operatörü Desteklemez	Görev Sıralama Problemleri
Değer Kodlaması	Değer Dönüştürmeye Gerek Yoktur	Özel Çaprazlama ve Mustasyon Gerektirir	Sinir Ağları Problemleri
Ağaç Kodlaması	Operatör Uygulanması Kolay	Bazı Problemler İçin Ağaç Tasarlamak Zordur	Evrim Problemleri

Genetik algoritmalarda ikili kodlama, içerilen bilgiyi temsil etmenin en yaygın yoludur. Yaygın olarak kullanılma nedenlerinin başında basitliği gelmektedir. İkili kodlamada her kromozom bir bit dizisidir: 0 veya 1 olarak kodlanmaktadır. Tek bir x_i değişkeni sırasıyla a_i ve b_i alt ve üst sınırlarına sahipse, x_i'yi n bit kullanarak ikili kodlamanın standart yolu, $0 < k < 2$ için tamsayı k için standart ikili kodlamaya karşılık gelen $a_i + k\frac{b_i - a_i}{2^n}$ ve $a_i + (k+1)\frac{b_i - a_i}{2^n}$ arasındaki gerçek değerleri kullanmaktır. Örneğin $a_i = 0$, $b_i = 4$ ve $n = 5$ ise $\frac{3}{8}$ ile $\frac{1}{2}$ arasındaki gerçek değerler, ikili kod olarak 00011'e karşılık gelmektedir (Wright, 1991).

Genetik algoritmaların kullanımı sırasında karşılaşılan ilk sorun, değerlerin doğru bir şekilde temsil edilip edilmediğidir. Sürekli değişkenlerin fonksiyonlarının optimize edilmesi için ikili kodlamanın kullanılması, temsil edilememe sorununa sebep olmaktadır. Bu nedenle değer kodlamasında, x kromozomu gerçek sayıların bir vektörü olarak kodlanmaktadır. Bu sayede x'in her bileşeni f fonksiyonunun bir değişkenini temsil etmektedir (Bessaou ve Siarry, 2001).

Permütasyon kodlaması, çoğunlukla çizelgeleme problemlerini çözmek amacıyla geliştirilmiş olup, tasarım uzayının boyutunun azalttığı için optimum permütasyon arayışını daha verimli bir şekilde ele almaktadır. Permütasyon kodlaması, çoğunlukla belirli bir şehir listesi için, seyahat maliyetini en aza indirmeyi amaçlayan gezgin satıcı problemleri için geliştirilmiş ve dizinin nerede başladığına duyarsızdır, permütasyon kodlamasında döngüsel permütasyonlar önemli değildir (B. Liu, T. Haftka, A. Akgün ve Todoroki, 2000). Permütasyon kodlamasında, her kromozom, bir dizideki bir konumu temsil eden bir sayı dizisidir.

Ağaç kodlaması esas olarak, gelişen programlar veya ifadeler için kullanılır. Ağaç kodlamasında her kromozom, fonksiyonlar veya komutlar gibi bazı nesnelerden oluşan bir ağaçtır. Cebirsel ifadeler keyfi boyutta ve çeşitli formlarda bulunmaktadır. Bunları sabit uzunluklu ikili diziler olarak kodlamak, arama alanını kısıtlar ve sonuçta algoritma yalnızca belirli bir problem yerine genel bir sınıfa uygulanabilir hale gelmektedir. Bu nedenle, sayısal tahminlerde cebirsel ifadeleri temsil etmek için ağaç kodlama kullanmak kullanışlı bir alternatif olarak ortaya çıkmaktadır (McKay, 1995).

3.3.1.3. Seçim Teknikleri

Seçim, genetik algoritmalarda belirli bir dizinin üreme sürecine katılıp katılmayacağını belirleyen önemli bir adımdır. Seçim adımı bazen "üreme operatörü" olarak da adlandırılmaktadır (Fox ve McMahon, 1991; Jebari, 2013). Seçim, çaprazlama için popülasyondan iki ebeveyn seçme işlemidir. Seçimin amacı, popülasyondaki mükemmel bireyleri seçerek yavruların daha mükemmel olmasını sağlamaktır. Genetik algoritmanın yakınsama oranı seçimde belirlenen yavrulara bağlıdır. Genetik algoritmalarda seçim aşamasında doğru yavruların seçimi optimal veya neredeyse optimal çözümlerin

elde edilmesini sağlayabilir. Ancak, yanlış bir yavrunun seçimi sonucunda yakınsama hızı yavaş olacak ve genetik algoritmanın optimum çözüme ulaşması daha uzun zaman alacaktır (Sivanandam ve Deepa, 2007). Sıklıkla kullanılan seçim teknikleri, rulet çarkı (roulette wheel), rank, turnuva (tournament), Boltzmann ve stokastik evrensel (stochastic universal sampling) örneklemedir. Seçim tekniklerine ait artılar ve eksiler Tablo 5'te verilmiştir.

Tablo 5. Farklı Şeçim Tekniklerinin Karşılaştırılması (Katoch vd., 2021)

Seçim Tekniği	Artıları	Eksileri
Rulet Çarkı	Uygulaması Kolay Basit Yanlılıktan Arınmış	Erken Yakınsama Riski Uyum Fonksiyonunda Mevcut Olan Varyansa Bağlıdır
Rank	Çeşitliliği Korur Yanlılıktan Arınmış	Yavaş Sıralama Gerekli Hesaplama Açısından Pahalı
Turnuva	Çeşitliliği Korur Paralel Uygulama Sıralama Gerekmez	Turnuva Boyutu Büyük Olduğunda Çeşitlilik Kaybı
Boltzmann	Global Optimum Elde Edilir	Hesaplama Açısından Pahalı
Stokastik Evrensel Örnekleme	Hızlı Yanlılıktan Arınmış	Erken Yakınsama

İdeal bir seçim stratejisi, genetik algoritmada arama performansına ince ayar yapmak için bireyler arasındaki uyum farkını ve popülasyon çeşitliliğini ayarlayabilecek şekilde olmalıdır. Rulet çarkı seçim tekniğinde kümülatif olasılık, tüm bireylerin seçim olasılıkları ile hesaplanmaktadır. Denklem 3.2 ile k numaralı bireyin kümülatif olasılığı hesaplanabilir. 0 ile 1 arasında olan rastgele e sayısı oluşturulur ve $p_x(a_k)$ ile karşılaştırılarak seçim bireyi belirlenir. Eğer $a_{k-1} < e < a_k$ ise birey seçilir. Yavru neslin n bireyini oluşturmak için n tur işlem tekrarlanmaktadır.

$$p_x(a_k) = \sum_{j=1}^{k} p_x(a_j) \tag{3.2}$$

Rulet çarkı yönteminde seçim tamamen rastgele sayılara bağlı olarak yapılmakta, bu da seçim belirsizliğini arttırmaktadır. $n = 100$'lük bir popülasyon ölçeği için ortalama seçim olasılığı sadece 0,01 olmakla birlikte popülasyon ölçeğinin daha da artması ile optimal bir bireyin seçilme olasılığı daha da düşmektedir. Rulet çarkı yöntemindeki bu rastgelelik mükemmel bir bireyin kaybına neden olabilmektedir (Yu, Fu, Li ve Dong, 2016).

Rank seçimi, rulet çarkı seçiminin hatalarını ortadan kaldırmak için seçim prosedürüne determinizm eklenerek, değiştirilmiş bir şeklidir. Her bireyin kendi sıralamasına göre seçilme şansı elde etmesi için uyum değerine göre sıralanmakta, bu da çözümün zamanından önce yerel bir minimuma

yakınsama olasılığını azaltmaktadır. Rank seçimi popülasyonu uyum değerine göre sıralar, en kötünün uyum değeri 1 ve en iyinin uyum değeri N'dir. Rank seçiminde uyum varyansı düşük olduğundan bireyler arasındaki uyum farkını sürdürmekte, çeşitliliği korumakta ve başarılı bir arama sağlamaktadır (Sivanandam ve Deepa, 2007).

Rulet çarkı seçiminin aksine turnuva seçim stratejisi N_u bireyleri arasında bir turnuva yarışması düzenleyerek bireyler arasındaki farkı sağlar. Turnuvadaki en iyi birey, N_u'nun kazananı olan en yüksek uyuma sahip bireydir. Turnuva yarışmalarını kazanan bireyler sonra çiftleşme havuzuna eklenir, bu işlem çiftleşme havuzu dolana kadar tekrarlanır. Turnuva galibini içeren çiftleşme havuzu daha yüksek ortalama popülasyon uyumuna sahiptir. Uyum farkı, genetik algoritmada sonraki genlerin uyumunu iyileştirerek optimum çözüme ulaşmayı sağlamaktadır (Sivanandam ve Deepa, 2007).

Benzetimli tavlama, bir fonksiyonun minimizasyonu veya maksimizasyonu yöntemidir. Bu yöntem, bir minimizasyon probleminde minimum fonksiyon değerine ulaşmak için erimiş metalin yavaş soğuma sürecini simüle etmektedir. Boltzmann, olasılık dağılımı kavramı ile tanıtılan sıcaklık benzeri bir parametrenin kontrol edilmesi, soğutma olayını simüle etmektedir. Boltzmann seçiminde sürekli değişken bir sıcaklık, önceden ayarlanmış bir programa göre seçim oranını kontrol etmektedir. Sıcaklık yüksek olarak başlar, bu da bireyler arasındaki uyum farkının düşük olduğu anlamına gelmektedir. Sıcaklık kademeli olarak düşürülerek bireyler arasındaki farklılık kademeli olarak arttırılır böylece genetik algoritmanın uygun çeşitlilik derecesini korurken, arama uzayının en iyi çözüm kümesine daha yakın bir şekilde daralmasına izin vermektedir (Sivanandam ve Deepa, 2007). Boltzmann seçimi, Monte Carlo simülasyonunda kullanılan entropi ve örnekleme yöntemlerine dayanmakta ve erken yakınsama sorununun çözülmesine yardımcı olan bir seçim yöntemidir (Chang-Yong Lee, 2003).

Stokastik evrensel örnekleme seçim tekniği, rulet çarkı seçim yönteminin bir uzantısıdır. Bir popülasyonun bireyler listesinden bir başlangıç noktası belirlenir ve yeni bireyler eşit aralıklarla seçilir. Stokastik evrensel örnekleme bir sonraki nesil için çaprazlamaya katılmak üzere seçilecek tüm bireylere eşit seçilme imkanı sunmaktadır (Katoch vd., 2021).

3.3.1.4. Çaprazlama Operatörleri

Genetik algoritma, verilen problemin cevabına eşit veya yakın bir sonucu bir önceki popülasyondan üretilen yeni popülasyonlar ile aramaktadır. Genetik algoritmalarda en iyi çözümleri / yavruları oluşturmak için kullanılan temel strateji, ebeveyn genlerini çaprazlamaktır (Umbarkar ve Sheth, 2015). Çaprazlama operatörleri iki veya daha fazla ebeveynin genetik bilgilerini birleştirerek yavruları oluşturmak için kullanılmaktadır. Optimum çözümü minimum nesilde mümkün olduğunca erken elde etmek için çeşitli çaprazlama teknikleri geliştirilmiştir. Sık kullanılan çaprazlama operatörleri: tek noktalı, iki noktalı, k noktalı, tek düze, kısmen eşleşen, sıralı, öncelik koruyuculu

çaprazlama, karışık, indirgenmiş vekil ve döngü çaprazlamasıdır. Çaprazlama operatörlerine ait artılar ve eksiler Tablo 10'da verilmiştir.

Tablo 6. Farklı Çaprazlama Operatörlerinin Karşılaştırılması (Katoch vd., 2021)

Çaprazlama Operatörü	Artıları	Eksileri
Tek Nokta	Uygulaması Kolay Basit	Daha Az Çeşitli Sonuçlar
İki ve K Nokta	Uygulaması Kolay	Daha Az Çeşitli Sonuçlar Küçük Alt Kümelere Uygulanabilir
İndirgenmiş Vekil	Küçük Optimize Problemlerinde Daha İyi Performans	Erken Yakınsama
Tek Düze	Yansız Keşif Büyük Alt Kümelerde Uygulanabilir Daha İyi Tekrar Kombinasyon Potansiyeli	Daha Az Çeşitli Sonuçlar
Öncelik Koruyuculu	Daha İyi Yavru Üretimi	Artık Sorunu
Sıralı	Daha İyi Keşif	Önceki Bireylerden Gelen Bilgi Kaybı
Döngü	Yansız Keşif	Erken Yakınsama
Kısmen Eşleşen	Daha İyi Yakınsama Oranı Diğer Çaprazlama Operatörlerinden Daha Üstün	-

Tek noktalı çaprazlama, genetik algoritma uygulamaları için kullanılan basit çaprazlama teknikleri arasında yer almaktadır. Bu çaprazlama, ebeveynlerin tek noktadan parçalanmasını kullanmakta ve ardından yavru oluşturmak için ebeveynlerin çaprazlama noktalarını birleştirmektedir. Tek noktalı çaprazlama, önce çaprazlama için kullanılacak iki ebeveyni seçmekte ve ardından rastgele herhangi bir çaprazlama noktası $p_i (i = 0$ ile $n - 1)$ seçmektedir. Ebeveynler çaprazlama noktalarından birleştirilerek iki yavru oluşturmaktadır (Umbarkar ve Sheth, 2015). Aşağıda iki ebeveyn ile oluşturulan, tek nokta çaprazlama gerçekleştirilen basit bir örnek gösterilmektedir.

Ebeveyn 1: (1 0 1 0 | 1 0 0 1 0)
Ebeveyn 2: (1 0 1 1 | 1 0 1 1 0)
Yavru 1: (1 0 1 0 | 1 0 1 1 0)
Yavru 2: (1 0 1 1 | 1 0 0 1 0)

Yukarıdaki örnekte dördüncü ve beşinci genler arası çaprazlama noktası olarak seçilmiştir.

İki noktalı ve k noktalı çaprazlamada, iki veya daha fazla rastgele çaprazlama noktası seçilir ve ebeveynlerin genetik bilgileri oluşturulan bölümlere göre değiştirilir (Katoch vd., 2021). Aşağıda iki ebeveyn ile oluşturulan k nokta çaprazlama gerçekleştirilen basit bir örnek gösterilmektedir.

Ebeveyn 1: (1 0 | 1 0 | 1 0 0 | 1 0)
Ebeveyn 2: (1 0 | 0 0 | 1 0 1 | 1 0)
Yavru 1: (1 0 | 0 0 | 1 0 0 | 1 0)
Yavru 2: (1 0 | 1 0 | 1 0 1 | 1 0)

Yukarıdaki örnekte ikinci ve üçüncü, dördüncü ve beşinci, yedinci ve sekizinci genler arası çaprazlama noktası olarak seçilmiştir.

Tek düze çaprazlama, kromozomları yeni kombinasyonlar oluşturmak için parçalara ayırmamaktadır. Yavrulardaki her gen, ebeveyn kromozomlarının uzunluğu ile aynı uzunluktaki ikili çaprazlama maskesindeki ilgili bite göre seçilen ebeveynden kopyalanarak oluşturulmaktadır. Çaprazlama maskesindeki bit 1 ise, ortaya çıkan gen ilk ebeveynden kopyalanmakta ve çaprazlama maskesindeki bit 0 ise, ortaya çıkan gen ikinci ebeveynden kopyalanmaktadır. Çaprazlama noktasının miktarı sabit olmadığından dolayı yavrular her iki ebeveynden gelen genlerin bir karışımına sahiptir (Kora ve Yadlapalli, 2017). Aşağıda iki ebeveyn ile oluşturulan tek düze çaprazlama gerçekleştirilen basit bir örnek gösterilmektedir.

Ebeveyn 1:	(1 1 1 0 1 0 0 1 0)
Ebeveyn 2:	(1 0 0 0 1 0 1 1 0)
Yavru 1:	(1 1 0 0 1 0 1 1 0)
Yavru 2:	(1 0 1 0 1 0 0 1 0)

Kısmen eşleştirilmiş çaprazlama en sık kullanılan çaprazlama operatörlerinin başında gelmektedir. Diğer operatörlere göre daha kısa sürede ve daha iyi bir performans ile optimuma ulaşmayı sağlamaktadır. Kısmen eşleştirilmiş çaprazlamada çiftleşme için iki ebeveyn seçilmekte ve ebeveynlerden biri genetik materyalinin bir kısmını yavrusuna bağışlamakta ve diğer ebeveynin karşılık gelen kısmı yavrunun genlerine dahil edilmektedir. Bu süreç tamamlandığında, dışarıda kalan genler ikinci ebeveynden kopyalanmaktadır. Aşağıda iki ebeveyn ile oluşturulan kısmen eşleştirilmiş çaprazlama gerçekleştirilen basit bir örnek gösterilmiştir.

Ebeveyn 1:	⟨1 2 3 | 4 5 6 | 7 8⟩
Ebeveyn 2:	⟨3 7 5 | 1 6 8 | 2 4⟩
Yavru 1:	⟨4 2 3 | 1 6 8 | 7 5⟩
Yavru 2:	⟨3 7 8 | 4 5 6 | 2 1⟩

Yukarıdaki örnekte, üçüncü ve dördüncü genler ile altıncı ve yedinci genler arasında yer alan kesme noktalarının arasındaki alt dizi "eşleme" olarak adlandırılmaktadır. Örnekte $4 <-> 1$; $5 <-> 6$ ve $6 <-> 8$ eşlemeleri tanımlanmaktadır. İlk ebeveynin eşleme bölümü ikinci yavruya, ikinci ebeveynin eşleme bölümü birinci yavruya kopyalanmaktadır: Yavru 1 = $\langle X\ X\ X \mid 1\ 6\ 8 \mid X\ X\rangle$, Yavru 2 = $\langle X\ X\ X \mid 4\ 5\ 6 \mid X\ X\rangle$. Ardından, Yavru 1'de kalan boşluklar Ebeveyn 1 tarafından, Yavru 2'de kalan boşluklar Ebeveyn 2 tarafından doldurulmaktadır. Ebeveynler tarafından doldurulan genler, eşleme bölümü dikkate alınarak doldurulmaktadır. Örneğin; Ebeveyn 1'in ilk geninde yer alan "1" eşleme bölümünde yer almasından dolayı Yavru 1'e "4" olarak aktarılmaktadır.

Sıralı çaprazlama, ebeveynin bir veya daha fazla bölümünü seçilen kesim noktalarından yavruya kopyalamakta ve kalan alanı kopyalanan bölümünde yer alanlardan farklı değerlerle doldurmaktadır (Katoch vd., 2021). Sıralı çaprazlama, tek bir adımda gerçekleştirilebilen basit bir permütasyon

çaprazlamasıdır (R. Kumar vd., 2022). Süreç kesme noktası seçilerek kısmen eşleştirilmiş çaprazlamaya benzer şekilde başlamaktadır. Sıralama çaprazlama, kısmen eşleştirilmiş çaprazlamada olduğu gibi nokta değişimler kullanılmak yerine eşlenen konumları göndererek dışarıda kalan boşlukları doldurmak için kayma hareketi uygulamakta ve çaprazlama noktaları arasındaki permütasyon elemanlarının bir kısmını kesilen diziden doğrudan yavrulara kopyalamakta ve aynı konuma yerleştirmektedir (Kora ve Yadlapalli, 2017).

Öncelik koruyuculu çaprazlama, çaprazlama uygulamadan önce yavruların ebeveynlerinde mevcut olan bireysel çözümlerin sıralamasını korumaktadır. Öncelik koruyucu çaprazlamada, her iki ebeveynden yavruya bireylerin seçilip seçilmeyeceğine rastgele 1'ler ve 0'lardan oluşan bir dizi yardımıyla karar verilmektedir (Katoch vd., 2021).

Karışık çaprazlama, diğer operatörlerin getirdiği yanlılığı azaltmak için önerilmektedir. Çaprazlama işleminden önce bireysel bir çözümü değerlerini karıştırmakta ve çaprazlama işlemi gerçekleştirildikten sonra karıştırmayı çözerek çaprazlama noktasının çaprazlamada herhangi bir yanlılık oluşturmasını engellemektedir . Karışık çaprazlamada, tek noktalı çaprazlama tekniği kullanılır ek olarak karıştırma işlemi uygulanmaktadır (Umbarkar ve Sheth, 2015).

İndirgenmiş vekil çaprazlama, ebeveynlerin genetik kompozisyonlarında yeterince çeşitlilik olması durumunda genetik algoritmanın daha iyi bireyler ürettiği varsayımına dayanarak gereksiz çaprazlamaları azaltmakta ancak aynı bileşime sahip ebeveynler için daha iyi yavrular üretememektedir (Katoch vd., 2021). İndirgenmiş vekil çaprazlama ilk olarak ebeveynlerin bireysel genlerini kontrol etmekte ve her iki ebeveynin genlerinin farklı olduğu tüm olası çaprazlama noktalarının listesini oluşturmaktadır. Bu listeleri oluşturduktan sonra, eğer listede bir eleman bulunmuyorsa bir işlem gerçekleştirmez. Ancak, ebeveynlerin 1'den fazla gende farklılık göstermesi durumunda, listeden rastgele bir çaprazlama noktası seçmek kaydıyla yavruları oluşturmak için tek noktalı çaprazlama gerçekleştirmektedir (Umbarkar ve Sheth, 2015).

Döngü çaprazlama, her bir elemanın ebeveynlerinin konumuna atıfta bulunarak pozisyon aldığı ebeveynleri kullanarak bir yavru oluşturmaya çalışmaktadır. İlk döngüde, ilk ebeveynden bazı genleri almakta ardından ikinci döngüde ikinci ebeveynden geri kalan genleri almaktadır. Döngü çaprazlama, permütasyon kodlamasına sahip kromozomlar için kullanılmaktadır (Kora ve Yadlapalli, 2017).

3.3.1.5. Mutasyon

Mutasyon, bir popülasyondan bir sonraki popülasyona genetik çeşitliliği korumak amacıyla kullanılan bir operatördür (Katoch vd., 2021). Mutasyon süreci, genetik algoritmaların erken yakınsama sorununu önlemek için yerel veya optimum altında kalan çözümlerden kaçınmasını sağlamaktadır (Lim

vd., 2017). Sıklıkla kullanılan mutasyon operatörleri; Yer değiştirme, basit ters çevirme ve karıştırma mutasyonudur. Çeşitli mutasyon operatörlerinin artıları ve eksileri Tablo 11'de verilmektedir.

Tablo 7. Farklı Mutasyon Operatörlerinin Karşılaştırılması (Katoch vd., 2021)

Mutasyon Operatörü	Artıları	Eksileri
Yer Değiştirme	Uygulaması Kolay Küçük Problemler için Uygulanabilir	Erken Yakınsama Riski
Basit – Ters Çevirme	Uygulaması Kolay	Erken Yakınsama
Karıştırma	Çok Sayıda Geni Etkiler Büyük Problemler için Uygulanabilir	Popülasyonda Bozulma Bazı Problemlerde Çözüm Kalitesinin Bozulması

Yer değiştirme mutasyonu, rastgele bir alt dizi seçmekte ve seçmiş olduğu diziyi rastgele bir konuma yerleştirmektedir. Yer değiştirme mutasyonu rastgele seçilen alt dizinin bir pozisyon sayısı kadar sağa ya da sola kaydırılması olarak da açıklanabilmektedir (Nearchou, 2004).

1 2 3 4 5 6 7 8 9
1 2 7 8 9 3 4 5 6

Basit – ters çevirme mutasyon operatörü, tek bir çözümde belirtilen herhangi iki konum arasındaki alt diziyi tersine çevirmektedir. Basit – ters çevirme mutasyon operatörü ters çevirdiği alt diziyi rastgele bir konuma yerleştirmektedir (Katoch vd., 2021).

1 2 3 4 5 6 7 8 9
1 2 3 4 8 7 6 5 9

Karıştırmalı mutasyon operatörü, bireysel çözümün belirli bir aralığındaki elemanları rastgele bir sırayla yerleştirir ve yeni üretilen çözümün uygunluk değerinin iyileştirilip iyileştirilmediğini kontrol etmektedir.

1 2 3 4 5 6 7 8 9
1 2 3 6 4 7 5 8 9

Optimum şartlar altında kullanılabilecek olan çeşitli operatörlerin en iyi kombinasyonları Tablo 12'de verilmektedir.

Tablo 8. Optimum Şartlar Altında Çeşitli Operatörlerin En İyi Kombinasyonu (Katoch vd., 2021)

Kodlama Şeması	Çaprazlama	Mutasyon
İkili	Tek Düze Aritmetik Tek Nokta K Nokta	Basit – Ters
Permütasyon	Kısmen Eşleştirilmiş Döngü Sıralı	Basit – Ters
Değer	Tek Düze Aritmetik Tek Nokta K Nokta	Yer Değiştirme

Ağaç	Tek Düze Tek Nokta	Karıştırmalı

Tablo 12 incelendiğinde tek düze çaprazlama yöntemi farklı kodlama şemaları ve farklı mutasyon yöntemleri ile kullanılabilmektedir. Kısmen eşleştirilmiş çaprazlama yöntemi, permütasyon kodlama şeması ve basit – ters mutasyon ile kullanıldığında optimum sonuca ulaşmaktadır. Değer kodlama şeması tercih edildiğinde yer değiştirme mutasyon yöntemi ile optimum sonuca ulaşmayı sağlamaktadır.

3.3.2. Gerçek ve İkili Kodlu Genetik Algoritmalar

İkili kodlu genetik algoritmalar, özellikle bir özelliğin var olup olmadığını belirten karar değişkenlerinin olduğu problemlerde kullanılmaktadır. Bununla birlikte, kesikli bir arama uzayı problemini ele alırken de ikili kodlu genetik algoritmalar kullanılmaktadır. Kesikli bir arama uzayı problemini ele alırken ikili kodlu genetik algoritmaları kullanmak için önemli nedenler bulunmaktadır.

Holland (1975) ve Goldberg (1989), popülasyon büyüklüğü N olan bir genetik algoritmanın her nesilde N adet dizgiyi işlemine rağmen şema uzayında $O(N^3)$ kadar şemanın işlendiğini öne sürmektedir. Bu kaldıraç, genetik algoritmaya örtük paralellik özelliği sağlamaktadır. Goldberg, işlenen toplam şema sayısına dayanarak, ikili kodlu genetik algoritmanın bilgi biti başına maksimum şema sayısını sunduğunu belirtmektedir (Deb, 2001).

Sürekli bir arama uzayına sahip problemleri ele almak için ikili kodlu genetik algoritmaların kullanılması gerektiğinde, birçok zorluk ortaya çıkmaktadır. Zorluklardan biri, komşu bir çözüme (gerçek uzayda) geçilmesi sırasında ikili kodlu genetik algoritmada birçok bit değerin gerektiği belli dizilerde ortaya çıkan Hamming uçurumlarıdır. İkili kodlamalarda bulunan Hamming uçurumları, sürekli arama uzayında kademeli bir arama için engel oluşturmaktadır (Deb, 2001). Bundan dolayı sürekli bir arama uzayına sahip problemlerin çözümüne imkân sağlamak için D. E. Goldberg (1990) tarafından gerçek kodlu genetik algoritmalar kullanılmaya başlanmıştır. Gerçek kodlu genetik algoritmada, bir çözüm doğrudan gerçek parametreli karar değişkenlerinin bir vektörü olarak gösterilmektedir. Bu durum çözümlerin, problemin doğal formülasyonuna yakın bir şekilde temsil edilmesini sağlamaktadır (Patel ve Raghuwanshi, 2010).

3.3.3. Çok Amaçlı Genetik Algoritmalar

Birçok gerçek problem, birden fazla amacın eşzamanlı optimizasyonunu içermektedir. Tek amaçlı optimizasyonda amaç, global optimum olarak adlandırılan en iyi tasarım çözümünü bulmaktır. Buna karşılık, birbiriyle çelişen hedefleri bulunan çok kriterli bir optimizasyonda, tek bir optimum çözüm bulunmamaktadır. Bu durumda farklı hedefler arasındaki etkileşim, çoğunlukla Pareto optimal çözümler olarak bilinen bir dizi uzlaşılmış çözüme ulaşılmasını sağlamaktadır (Prasad ve Park, 2004).

Çok amaçlı genetik algoritmaların temel amacı, diğer amaç fonksiyonlarını bozmadan herhangi bir amaç fonksiyonunda daha fazla iyileştirme yapılamayacak şekilde amaç uzayında optimum Pareto verimliliğini sağlamaktır (Katoch vd., 2021).

Klasik arama ve optimizasyon yöntemlerin, çok kriterli optimizasyon problemlerin çözümünde kullanılması verimli değildir, çünkü;

1. Klasik yöntemler tek seferde birden fazla çözüm bulamaz, dolayısıyla istenen Pareto optimum çözüm sayısı kadar tekrar gerektirmektedir.
2. Yöntemlerin birden fazla kez uygulanması, farklı Pareto optimum çözüm sayısına ulaşılmasını garanti etmemektedir.
3. Kesikli değişkenli ve birden fazla optimum çözümü olan problemleri verimli şekilde ele alamamaktadır (Prasad ve Park, 2004).

3.3.4. Paralel Genetik Algoritmaları

Paralel genetik algoritmalar, Holland (1975) tarafından geliştirilen genetik algoritmaların paralel bilgisayarlara uyarlanmış halidir. Paralel genetik algoritmalar, dağıtılmış kontrol yapısı ile tamamen asenkron olarak çalışmaktadır. Asenkron yapı, uzamsal bir popülasyon yapısı ve aktif bireylerin tanıtılması ile elde edilmektedir (Mühlenbein, Schomisch ve Born, 1991).

Üç ana paralel genetik algoritma türü bulunmaktadır;

1. Global tek popülasyonlu ana – köle genetik algoritmalar
2. Tek popülasyonlu ince taneli genetik algoritmalar
3. Çok popülasyonlu iri taneli genetik algoritmalar

Global tek popülasyonlu ana – köle genetik algoritmada tek bir popülasyon bulunmaktadır ve uygunluk değerlendirmesi birkaç istemci arasında dağıtılmaktadır. Bu tür paralel genetik algoritmalarda seçim ve çaprazlama tüm popülasyonu dikkate aldığından, global paralel genetik algoritmalar olarak da bilinmektedir (Cantú-Paz, 2000).

İnce taneli genetik algoritmalar büyük ölçekli paralel bilgisayarlar için uygundur ve uzamsal olarak yapılandırılmış bir popülasyondan oluşmaktadır. Seçim ve çaprazlama küçük bir komşulukla sınırlıdır ancak komşuluklar üst üste binerek tüm bireyler arasında bir miktar etkileşime izin vermektedir (Cantú-Paz, 2000).

Çok popülasyonlu genetik algoritmalar ara sıra bireyleri değiştirilen alt popülasyonlardan oluştuğundan dolayı daha karmaşık bir yapıda bulunmaktadır. Göç adı verilen birey değiştirme işleminden dolayı diğer paralel genetik algoritmalara göre daha az tercih edilmektedir (Katoch vd., 2021).

3.3.5. Kaotik Genetik Algoritmalar

Genetik algoritmaların dezavantajlarından biri de erken yakınsamadır. Bu dezavantajı ortadan kaldırmak için kaotik sistemler genetik algoritmalara dahil edilebilmektedir. Genetik algoritmalardan farklı olarak kaotik genetik algoritmalarda çaprazlama ve mutasyon operatörleri kaotik haritalarla değiştirilerek erken yakınsama problemi ortadan kaldırılmaktadır (Katoch vd., 2021). Koatik değerler keşfe, yakınsak değerler ise sömürüye yardımcı olmaktadır. Kaotik ve yakınsak değerlerin bir arada kullanılması erken yakınsamayı ortadan kaldırmaktadır.

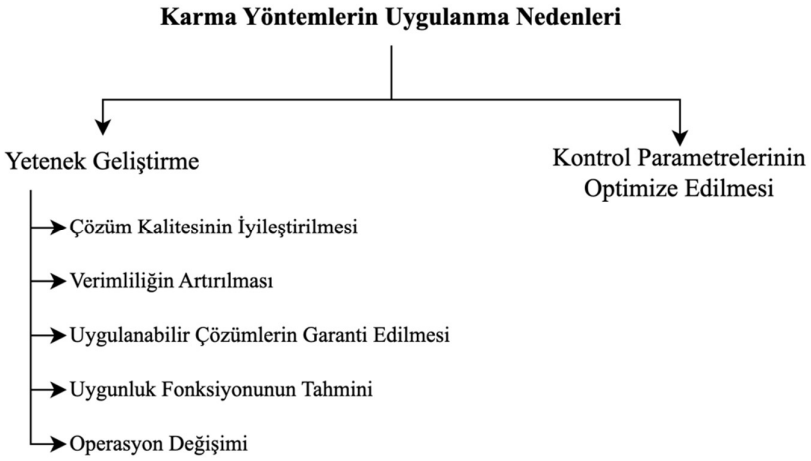
Fonksiyon optimizasyonu ile ilgili gerçek kodlu genetik algoritmalarda genler sürekli arama uzayında gösterilmektir. Bu tür kodlama yöntemlerinin kullanıldığı durumlarda çaprazlama ve mutasyon operatörlerinde bazı değişikliklerin yapılması gerekmektedir. Gerçek kodlu ve sürekli arama uzaylı problemlerde mutasyon operatörlerini oluşturmak için stokastik tekniklerden yararlanılmaktadır. Kaotik sistemler, deterministik stokastik özelliklere sahip olduklarından dolayı, gerçek kodlu ve sürekli arama uzayına sahip genetik algoritmalar ise stokastik yöntemlere göre daha iyi performans göstermesinden dolayı tercih edilmektedir (Li-Jiang ve Tian-Lun, 2002).

3.3.6. Karma Genetik Algoritmalar

Genetik algoritmalar ve diğer optimizasyon yöntemleri, bir optimizasyon hedefine ulaşmak için bir araya getirilebilen tamamlayıcı araçlar olarak görülmektedir. Karma genetik algoritmalar, optimizasyon performansını artırmak için bir veya daha fazla yöntemi içerebilmektedir.

Genetik algoritmalar, görüntü arındırma yöntemleri, kimyasal reaksiyon optimizasyonu ve çok daha fazlasında yöntemlerin performanslarını artırmak için diğer optimizasyon yöntemleriyle kolayca birleştirilebilir. Genetik algoritmanın diğer yöntemlerle birleştirilmesinin başlıca avantajları daha iyi çözüm kalitesi, daha iyi verimlilik, uygulanabilir çözümlerin garantisi ve optimize edilmiş kontrol parametreleri olarak belirtilmektedir (Katoch vd., 2021).

Karma yöntemlerin genetik algoritmalara uygulanmasının birçok nedeni bulunmaktadır. Bu nedenler 2 grup altında toplanabilir. Karma yöntemlerin uygulama nedenleri Şekil 22'de verilmektedir.



Şekil 22. Karma Yöntemlerin Genetik Algoritmalarda Uygulanma Nedenleri

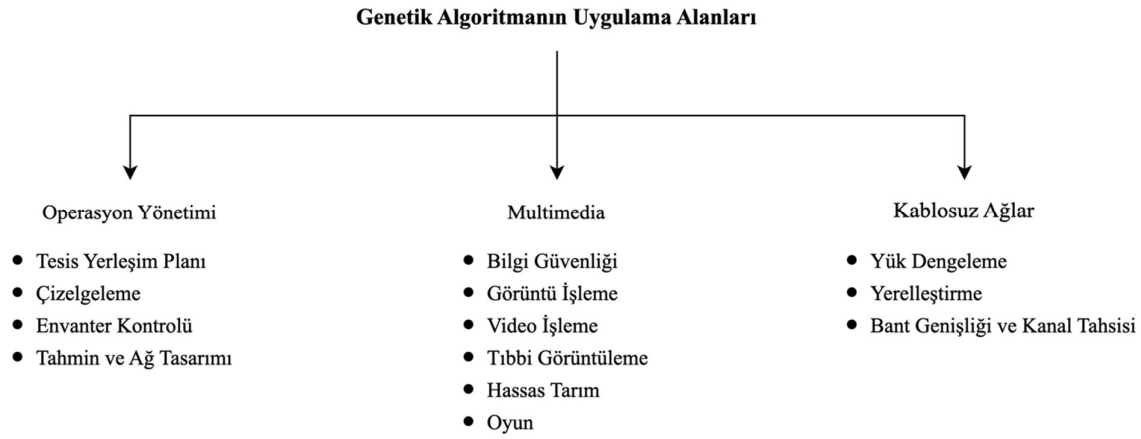
İki grup genetik algoritmaların yeteneklerini geliştirmek ve kontrol parametrelerinin optimize edilmesi olarak ifade edilmektedir. Optimizasyon problemlerinde genetik algoritmalar genel çözüme ulaşmak amacıyla kullanılmaktadır. Problem bazlı yöntemler ile birleştirilerek kullanıldığı takdirde genetik algoritmaların çözüme ulaşma yetenekleri geliştirilebilmektedir.

Farklı karma yöntemlerin kullanılması genetik algoritmaların çözüm kalitesini iyileştirebilmekte, verimliliğini arttırabilmekte, uygulanabilir çözümleri garanti edebilmekte, uygunluk fonksiyonunu tahmin edebilmekte ve operasyon değişimini gerçekleştirebilmektedir.

Optimizasyon performansını iyileştirmek amacıyla kontrol parametrelerini optimize etmek ve genetik algoritmanın davranışını izlemek amacıyla genetik algoritmalarla birlikte başka teknikler de kullanılabilmektedir. Bulanık mantığın bilgiyi kesin olmayan yollarla temsil etme yeteneği, açıkça tanımlanmamış veya tamamen anlaşılmamış bilgi üzerinde akıl yürütmek için kullanılmasını sağlamaktadır. Bu yetenek, bulanık mantığı bir genetik algoritmanın kontrol parametrelerini optimize etmek için uygun bir seçim haline getirmektedir (El-Mihoub, Hopgood, Nolle ve Battersby, 2006).

3.4. Genetik Algoritmanın Uygulama Alanları

Genetik algoritmanın uygulama alanları Şekil 23'te kategorize edilmiş bir halde verilmektedir. Genetik algoritmaların uygulama alanları operasyon yönetimi, multimedya ve kablosuz ağlar olmak üzere üç ana başlık altında toplanmaktadır.



Şekil 23. Genetik Algoritmanın Uygulama Alanları

3.4.1.Operasyon Yönetimi

Başta tesis yerleşim planı, çizelgeleme, envanter kontrolü, tahmin ve ağ tasarımı olmak üzere operasyon yönetiminde genetik algoritmalar sıklıkla kullanılmaktadır.

3.4.1.1. Tesis Yerleşim Planı

Tesis yerleşim planlaması, üretim tesislerinin verimli ve etkin bir şekilde işletilebilmesi için tesislerin konumunun ve şeklinin belirlenmesi sürecidir. Tesis yerleşim planlaması, üretim maliyetlerini, malzeme taşıma maliyetlerini, işgücü verimliliğini ve müşteri memnuniyetini önemli ölçüde etkileyebilir.

Tesis yerleşim problemleri, genellikle NP-Hard sınıfına giren zor problemlerdir. Bu nedenle, klasik optimizasyon teknikleri ile çözüm bulmak zordur. Genetik algoritmalar, bu tür problemlerin çözümü için etkili bir yaklaşımdır. Fabrikaların, depoların, ofislerin ve hastanelerin yerleştirilmesinin planlama aşamasında genetik algoritmalardan yararlanılmaktadır.

3.4.1.2. Çizelgeleme

Çizelgeleme, kaynakların sınırlı olduğu bir ortamda, birden fazla işi veya görevi verimli bir şekilde tamamlamak için bir dizi karar verme sürecidir. Çizelgeleme problemleri, üretim, nakliye, sağlık hizmetleri, eğitim ve finans gibi çok çeşitli alanlarda karşımıza çıkmaktadır.

Ders, sınav, üretim, nakliyat, hastane gibi alanlarda çizelgeleme işleminde genetik algoritmalardan yararlanılmaktadır.

3.4.1.3. Envanter Kontrolü

Envanter kontrolü, bir işletmenin elinde bulundurduğu stok miktarını yönetme sürecidir. Bu süreç, işletmenin stok maliyetlerini en aza indirirken, müşteri talebini de karşılamayı amaçlamaktadır. Envanter kontrolü, işletmenin karlılığını ve rekabet gücünü önemli ölçüde etkileyen bir süreçtir.

Genetik algoritmalar, karmaşık ve çok amaçlı envanter kontrol problemlerinin çözümünde etkili bir şekilde kullanılabilmektedir. Bu algoritmalar, geleneksel analitik yöntemlerle çözülmesi zor olan problemlerin çözümünde başarılı sonuçlar elde edebilmektedir.

Tek dönemli, çok dönemli envanter problemlerinde ve yasadışı stok problemlerinin çözümünde genetik algoritmalardan sıklıkla yararlanılmaktadır.

3.4.1.4. Tahmin ve Ağ Tasarımı

Genetik algoritmalar, tahmin problemlerinde, en iyi tahmin modelini bulmak için kullanılabilir. Genetik algoritmalar, farklı model parametreleri ve yapıları ile çok sayıda model oluşturabilir ve ardından bu modelleri performanslarına göre değerlendirebilmektedir.

Genetik algoritmalar, ağ tasarımında, en iyi ağ yapısını bulmak için kullanılabilir. Genetik algoritmalar, farklı ağ topolojileri ve parametreleri ile çok sayıda ağ oluşturabilir ve ardından bu ağları performanslarına göre değerlendirebilmektedir.

3.4.2. Multimedya

Genetik algoritmalar, bilgi güvenliği, görüntü işleme, video işleme, tıbbi görüntüleme, hassas tarım ve oyun başta olmak üzere birçok farklı multimedya alanında kullanılmaktadır.

3.4.2.1. Bilgi Güvenliği

Genetik algoritmalar, bilgi güvenliği alanındaki çeşitli zorlukların çözümüne yardımcı olabilecek güçlü bir araçtır. Genetik algoritmaların, kullanımı, bilgi güvenliğinin iyileştirilmesine ve siber saldırılara karşı korunmanın güçlendirilmesine yardımcı olabilmektedir.

Güvenlik açıklarının tespiti, güvenlik duvarlarının ve diğer güvenlik sistemlerinin tasarımı, siber saldırıların tespiti ve önlenmesi amacıyla bilgi güvenliği sektöründe genetik algoritmalardan yararlanılmaktadır.

3.4.2.2. Görüntü İşleme

Genetik algoritmalar, görüntüdeki bir nesnenin başka bir görüntüdeki eşleşmesini bulmak, görüntünün farklı bölümlerini ayırmak, görüntülerdeki gürültüyü azaltmak, hasarlı veya bozuk görüntüleri onarmak, tarihi belge restorasyonu veya güvenlik kamerası görüntülerinin kurtarılması için kullanılmaktadır.

Genetik algoritmalar, görüntü işleme alanındaki birçok zorlu sorunu çözmek için güçlü bir potansiyele sahiptir. Bu algoritmaların gelecekte bu alanda daha fazla kullanılacağı öngörülmektedir.

3.4.2.3. Video İşleme

Video işleme uygulamalarında, genellikle videodan elde edilen özelliklerin seçilmesi gerekmektedir. Bu özellikler, videodan elde edilen bilgileri temsil eder ve video işleme işleminde önemli bir rol oynamaktadır. Ancak, videodan elde edilebilecek çok sayıda özellik bulunmakta ve bu özelliklerin tümü uygulama için her zaman gerekli değildir. Bu nedenle, video işleme uygulamalarında, otomatik özellik seçimi önemli bir süreçtir. Genetik algoritmalar, otomatik özellik seçimi için etkili bir yöntemdir. Genetik algoritmada belirli işlemler sonucunda, video işleme için en uygun özellikler seçilebilmektedir.

Genetik algoritmalar, otomatik video segmentasyonu için de etkili bir yöntemdir. Genetik algoritmalar kullanılarak, videodaki belirli nesneler veya olaylar, seçilim, çaprazlama ve mutasyon gibi işlemler yoluyla sürekli olarak iyileştirebilmektedir. Bu işlemler sonucunda, videodan belirli nesneler veya olaylar doğru bir şekilde ayırt edebilmektedir.

Videodaki belirli nesneleri veya olayları tanımak için genetik algoritmalar kullanılabilir. Bu işlem, video işleme uygulamalarında önemli bir rol oynamaktadır. Örneğin, videolardan yüzleri tanımak, videolardan araçları tanımak veya videolardan sahneleri tanımak için genetik algoritmalar kullanılabilmektedir.

3.4.2.4. Tıbbi Görüntüleme

Genetik algoritmalar, tıbbi görüntü verilerinden anlamlı bilgiler çıkarmak için kullanılabilmektedir. Örneğin, kanser tespiti, patoloji teşhisi, radyolojik görüntü analizi için ve gerçekçi tıbbi görüntüler oluşturmak için genetik algoritmalardan yararlanılmaktadır. Genetik algoritmalar, ameliyat simülasyonu, cerrahi planlama ve tıbbi eğitim için de kullanılabilmektedir.

3.4.2.5. Hassas Tarım

Hassas tarım, mahsul verimini ve kalitesini artırmak için veri bilimi ve makine öğrenimi tekniklerini kullanan bir tarım uygulamasıdır. Genetik algoritmalar, mahsul yetiştirme modeli geliştirmek, hastalık ve zararlıları kontrol etmek, mahsul hasadını, mahsul rotasyonunu, sulama programlarını ve gübre uygulamalarını optimize etmek için kullanılabilmektedir.

3.4.2.6. Oyun

Oyun geliştirmede, genetik algoritmalar, çeşitli amaçlar için kullanılabilir. Örneğin oyun seviyelerinin ve seviye tasarımlarının oluşturulmasında, oyun karakterlerinin ve düşmanlarının

yaratılmasında, oyun kurallarının ve mekaniklerinin tasarımında genetik algoritmalardan yararlanılmaktadır.

3.4.3. Kablosuz Ağlar

Genetik algoritmalar, yük dengeleme, yerelleştirme, bant genişliği ve kanal tahsisi alanları başta olmak üzere kablosuz ağlarda sıklıkla kullanılmaktadır.

3.4.3.1. Yük Dengeleme

Yük dengeleme, bir ağdaki kaynakların verimli bir şekilde kullanılmasını sağlamak için kullanılan bir tekniktir. Yük dengeleme, ağdaki kaynakların, ağdaki trafik gereksinimlerine göre dağıtılmasını sağlamaktadır. Bu sayede, ağdaki performans sorunlarının önüne geçilebilmektedir. Kablosuz ağlarda yük dengeleme için genetik algoritmalar kullanılarak, ağdaki kaynakların daha verimli bir şekilde dağıtılması sağlanabilir. Genetik algoritmalar, ağdaki kaynakların, ağdaki trafik gereksinimlerine göre en uygun şekilde dağıtılmasını sağlayabilmektedir.

Ağdaki kaynakların, ağdaki trafik gereksinimlerine göre en uygun şekilde dağıtılmasını sağlayacak bir çözüm kümesi oluşturmak veya rotalama algoritması oluşturmak için genetik algoritmalardan yararlanılmaktadır.

3.4.3.2. Yerelleştirme

Kablosuz ağlarda yerelleştirme, bir cihazın bir kablosuz ağdaki konumunu belirleme işlemidir. Yerelleştirme, konum tabanlı hizmetleri desteklemek için önemli bir bileşendir. Genetik algoritmalar, gerçek zamanlı yerelleştirme, düşük güçlü yerelleştirme, karmaşık ortamlarda yerelleştirme için kullanılabilmektedir.

3.4.3.3. Bant Genişliği ve Kanal Tahsisi

Kablosuz ağlarda, kullanıcılar arasında veri aktarımını sağlamak için bant genişliği ve kanallar kullanılır. Kullanıcıların ihtiyaçlarına göre bant genişliği ve kanalları verimli bir şekilde tahsis etmek, kablosuz ağların esnekliğini artırmak için genetik algoritmalardan yararlanılmaktadır.

4. TEORİK VERİLER VE EMLAK SEKTÖRÜ ÜZERİNE UYGULAMA

Bu bölümde araştırmanın amacı, araştırmanın kapsamı gerçek ve teorik bilgilerden elde edilen veriler açıklanmaktadır.

4.1. Araştırmanın Amacı

Derin öğrenme yöntemlerinde kullanılan ağ modellerinde gizli nöron ve katman sayılarının belirlenmesi amacıyla kesin olarak kullanılan bir yöntem bulunmamakla beraber literatürdeki çalışmaların çoğunluğunda deneme yanılma yöntemi kullanılmaktadır.

Literatür incelendiğinde derin öğrenme yöntemlerinde gizli katman ve gizli nöron sayılarının belirlenmesinde belli öneriler bulunmaktadır. Huang Önerisi, Heaton Önerisi, Temel Bileşenler Analizi ve Genetik Algoritmalar bu önerilerden bazılarıdır.

Çalışmada:

- Yukarıda belirtilmiş olan önerileri aynı veri seti üzerine uygulayarak, önerilerin başarı sonuçlarının karşılaştırılması,
- Önerileri teorik veri seti üzerinde uygulayarak, belirli istatistiksel dağılımlara uygunluk gösteren veri setlerindeki önerilerin başarılarını incelemek amaçlanmıştır.

4.2. Araştırmanın Sınırlılıkları

Araştırmanın anakütlesini İstanbul'da yer alan ve fiyatı 750.000₺ ile 12.500.000₺ olan satılık evlerin tamamı oluşturmaktadır. Araştırmada Türkiye'nin en büyük satılık ev ilan sitesi olan sahibinden.com sitesinde 15.10.2022 tarihinde yer alan ve yukarıdaki özelliklere sahip evler örneklem olarak kullanılmıştır. Sahibinden.com 15.10.2022 tarihinde incelendiğinde ilgili fiyat kriterine uyan ve İstanbul'da yer alan 65.726 adet ev olduğu tespit edilmiş ve web kazıma yöntemi ile 65.726 adet eve ait "Fiyat", "İlçe", "Mahalle", "Emlak Tipi", "Brüt m^2", "Net m^2", "Oda Sayısı", "Bina Yaşı", "Bulunduğu Kat", "Kat Sayısı", "Isıtma", "Banyo Sayısı", "Balkon", "Eşyalı", "Kullanım Durumu", "Site İçerisinde", "Aidat", "Krediye Uygun", "Tapu Mülkiyeti" bilgileri toplanmıştır.

4.3. Araştırma Yöntemi

Araştırmada, gerçek verilerin elde edilmesinde web kazıma yöntemi, teorik verilerin oluşturulmasında istatistiksel dağılımlar ve modellerin oluşturulmasında derin öğrenme yöntemi kullanılmıştır.

Web kazıma, web'den yararlı bilgileri otomatik olarak toplama işlemidir. Prolog Sunucu Sayfaları (PSS) olarak bilinen bir betik dili kullanılarak uygulanabilen HTML sayfalarından faydalı bilgileri çıkarma işlemidir (Malik ve Rizvi, 2011). Araştırmada web kazıma işleminin gerçekleştirilmesinde Python programlama dili, beautifulsoup ve selenium kütüphanelerinden yararlanılmıştır.

Çalışmada derin öğrenmede kullanılan yöntemlerin MSE (Mean Square Error) (Ortalama Hata Kareleri) ve sürelerinin analizi için Kruskal Wallis ve Mann Whitney U testi kullanılmıştır. ANOVA testi, bir değişken için kategorilerin aynı anakütleden gelip gelmediğini belirlemek için güçlü bir yöntemdir. Anakütle ortalamaları karşılaştırıldığında ve anakütlelerin eşit varyanslara sahip olmadığı veya anakütlelerin normal dağılmadığı bilindiğinde Kruskal-Wallis testi kullanılmaktadır. (Montgomery ve Runger, 2014; Ostertagová ve Ostertag, 2013; Ostertagová, Ostertag ve Kováč, 2014).

Wilcoxon Sıra Toplamı Testi olarak da bilinen Mann Whitney U Testi, belirli bir dağılımı olmayan tek bir sıralı değişken üzerindeki iki grup arasındaki farklılıkları test etmek amacıyla kullanılmaktadır (Mann ve Whitney, 1947; McKnight ve Najab, 2010).

4.4. Gerçek ve Teorik Verilerin Elde Edilmesi

Gerçek ve teorik verilerin elde edilme süreçleri, bilimsel araştırmalarda önemli bir rol oynamaktadır. Gerçek ve teorik verilerin elde edilme süreçleri bu bölümde ele alınmaktadır.

4.4.1. Gerçek Verilerin Elde Edilmesi

Çalışmada kullanılacak gerçek verilerin elde edilmesi amacıyla sahibinden.com üzerinden İstanbul'da yer alan ve 750.000₺ ile 12.500.000₺ arasında bulunan 65.726 adet satılık ev ilanına ait bilgiler web kazıma işlemi uygulanarak elde edilmiştir.

Ev ilanlarına ait "Fiyat", "İlçe", "Mahalle", "Emlak Tipi", "Brüt m^2", "Net m^2", "Oda Sayısı", "Bina Yaşı", "Bulunduğu Kat", "Kat Sayısı", "Isıtma", "Banyo Sayısı", "Balkon", "Eşyalı", "Kullanım Durumu", "Site İçerisinde", "Aidat", "Krediye Uygun", "Tapu Mülkiyeti" bilgilerinin elde edilmesi amacıyla Ek-2'de yer alan Python kodlarından yararlanılarak web kazıma işlemi gerçekleştirilmiştir.

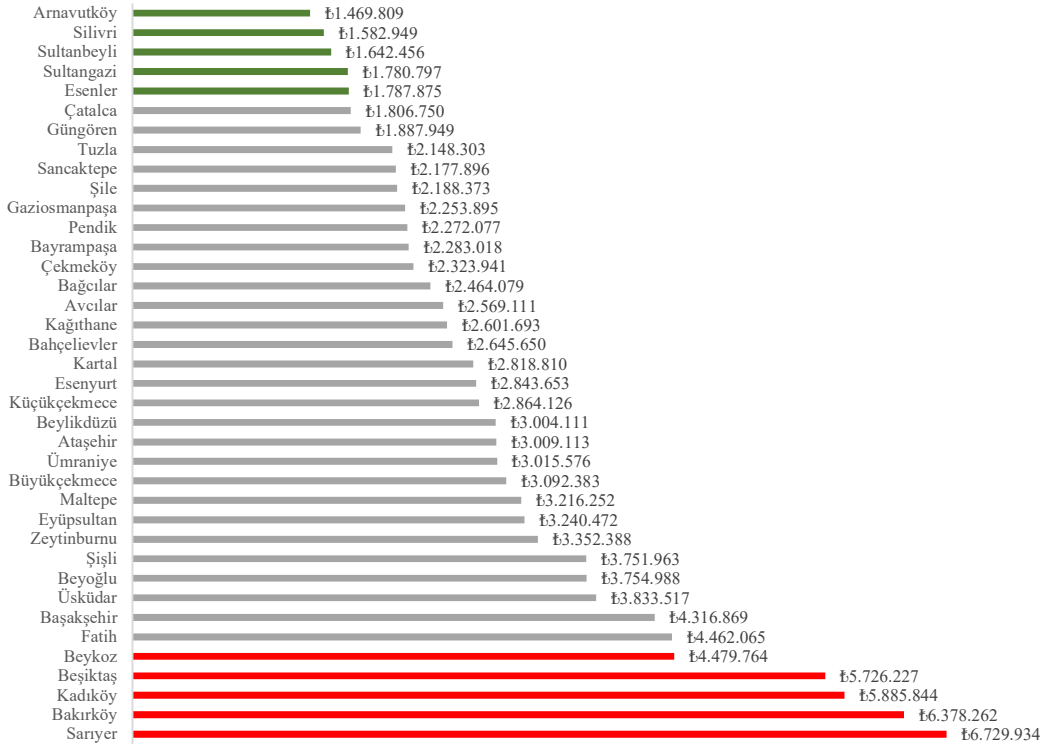
4.4.2. Teorik Verilerin Elde Edilmesi

Çalışmada kullanılacak teorik verilerin elde edilmesinde Python programlama dili kullanılarak ortalaması 100 ve standart sapması 25 olan normal dağılıma sahip 13 adet değişkenin her biri için 50.000 adet veri üretilmiştir. Yapay sinir ağlarında bu 13 değişkenden 1 tanesi bağımlı değişken (çıktı) diğer 12 tanesi ise bağımsız değişken (girdi) olarak kullanılmıştır.

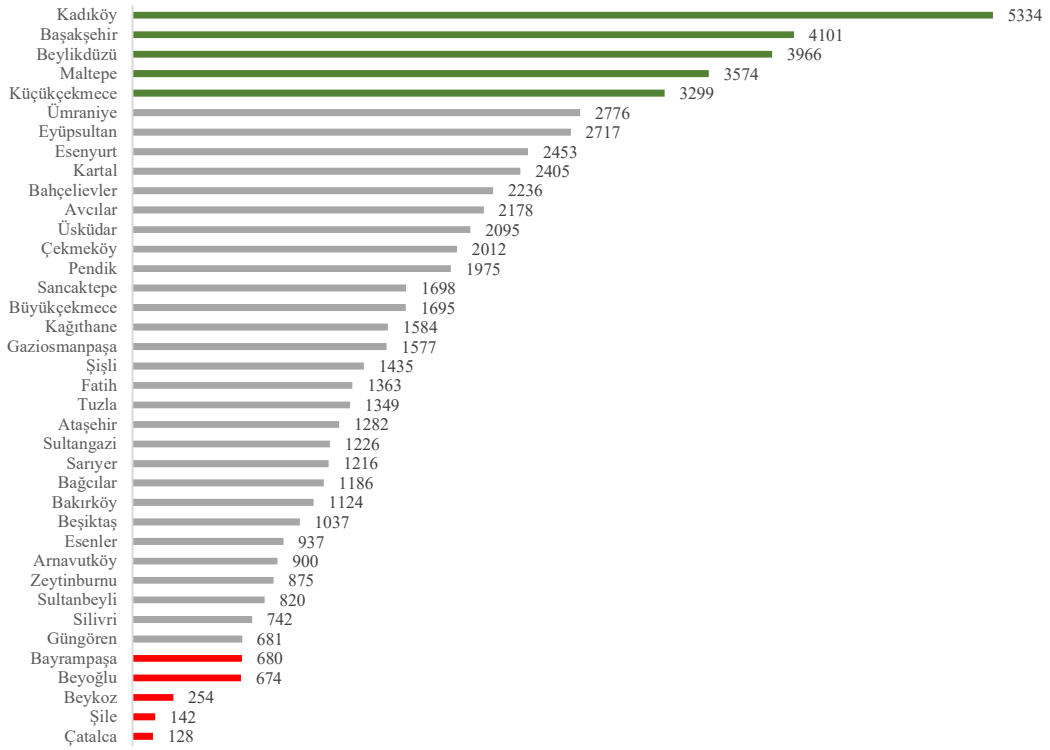
4.5. Gerçek Verilere Ait Tanımsal İstatistikler

Bu bölümde çalışmada kullanılacak olan gerçek verilere ait tanımsal istatistikler yer almaktadır. İstanbul'daki ilçelere göre ortalama satılık ev fiyatları Şekil 9'da yer almaktadır.

Şekil 24 incelendiğinde İstanbul'da ev fiyatları açısından en ucuz olan ilçe 1.469.809₺ ortalama fiyat ile Arnavutköy ilçesidir. En ucuz ikinci ilçe ise 1.582.949₺ ortalama fiyat ile Silivri. Bu iki ilçe İstanbul merkezine oldukça uzak bir konumda bulunduğundan dolayı diğer ilçelere göre daha ucuzdur. İstanbul'da ev fiyatları açısından en pahalı ilçe 6.729.934₺ ortalama fiyat ile Sarıyer ilçesidir. En pahalı ikinci ilçe ise 6.378.262₺ ortalama fiyat ile Bakırköy ilçesidir. En pahalı 5 ilçe incelendiğinde hepsinin Marmara Denizi'ne sahili bulunmasından dolayı ortalama ev fiyatlarının diğer ilçelere göre daha pahalı olduğu görülmüştür. İlçelere göre satılık ev fiyatları Şekil 24'te verilmiştir.



Şekil 24. İlçelere Göre Ortalama Satılık Ev Fiyatları



Şekil 25. İlçelere Göre Satılık Ev Sayısı

Şekil 25 incelendiğinde en çok satılık evin bulunduğu ilçenin 5.334 adet ev ile Kadıköy ilçesi olduğu tespit edilmiştir. Kadıköy ilçesinde özellikle Fikirtepe bölgesinde gerçekleştirilen kentsel dönüşümden dolayı satılık ev sayısı diğer ilçelere göre daha fazladır. Başakşehir 4.101 ev ile ikinci sırada ve Beylikdüzü ise 3.966 ev ile üçüncü sırada yer almaktadır. Başakşehir ve Beylikdüzü ilçeleri son 10 yıl içerisinde gelişimi hız kazanmış ve gelişimi devam eden ilçeler olduğundan dolayı satılık ev sayısı diğer ilçelere göre daha fazladır. Satılık evin en az bulunduğu ilçelerde, 128 adet ev ile Çatalca ilçesi birinci sırada, 142 adet ev ile Şile ilçesi ikinci sırada bulunmaktadır. Çatalca ve Şile ilçeleri İstanbul'un şehir merkezine en uzak ilçeler arasında yer almakta ve çoğunlukla tatil bölgesi olarak kullanıldıkları için satılık ev sayısı diğer ilçelere göre daha düşüktür. Bina Yaşı, Tapu Durumu, Yatak Odası Sayısı, Isıtma Sistemi, Kredi Durumu, Site İçinde, Kullanım Durumu, Eşyalı, Balkon ve Banyo Sayısı değişkenlerine ait frekans dağılımları Tablo 9'da verilmiştir.

Tablo 9. Değişkenlere Ait Frekans Dağılımı

Değişken	Kategori	Ev Sayısı	Yüzde
Bina Yaşı	0	14400	21,9
	1	1959	3,0
	2	2302	3,5
	3	3199	4,9
	4	6361	9,7

	5 – 10 arası	**15107**	**23,0**
	11 – 15 arası	6037	9,2
	16 – 20 arası	2837	4,3
	21 – 25 arası	4077	6,2
	26 – 30 arası	4301	6,5
	31 ve üzeri	5146	7,8
	Toplam	65726	100,0
Tapu Durumu	Arsa Tapulu	1563	2,4
	Kat İrtifaklı	14483	22,0
	Kat Mülkiyetli	**47385**	**72,1**
	Hisseli	979	1,5
	Bilinmiyor	1312	2,0
	Belirtilmemiş	4	,0
	Total	65726	100,0
Yatak Odası	1 Yatak Odası	5747	8,7
	2 Yatak Odası	**25843**	**39,3**
	3 Yatak Odası	24861	37,8
	4 Yatak Odası	4533	6,9
	5 ve Üzeri Yatak Odası	4742	7,2
	Total	65726	100,0
Isınma Sistemi	**Doğalgaz**	**45397**	**69,1**
	Merkezi Sistem	18210	27,7
	Yerden Isıtma	1545	2,4
	Diğer	451	0,7
	Yok	123	0,2
	Total	65726	100,0
Kredi Durumu	Belirtilmemiş	1140	1,7
	Evet	**57385**	**87,3**
	Hayır	7201	11,0
	Total	65726	100,0
Site İçinde	Belirtilmemiş	12	0,0
	Evet	23823	36,2
	Hayır	**41891**	**63,7**
	Total	65726	100,0
Kullanım Durumu	**Boş**	**28072**	**42,7**
	Kiracılı	23549	35,8
	Mülk Sahibi	14105	21,5
	Total	65726	100,0
Eşyalı	Belirtilmemiş	1948	3,0
	Evet	2747	4,2
	Hayır	**61031**	**92,9**
	Total	65726	100,0
Balkon	**Var**	**54549**	**83,0**

	Yok	11177	17,0
	Total	65726	100,0
Banyo Sayısı	**1**	**35050**	**53,3**
	2	27435	41,7
	3	2832	4,3
	4	372	0,6
	5	31	0,0
	6	6	0,0
	Total	65726	100,0

Tablo 9 incelendiğinde İstanbul'da satılık evlerin %23 ile büyük çoğunluğunun bina yaşlarının 5 – 10 arasında olduğu tespit edilmiştir. Satılık evlerin %66'sı 10 yaşında veya 10 yaşından daha genç binalarda bulunmaktadır. Satılık evlerin %20,5'lik bir kısmı 20 yaş ve daha büyük olduğu görülmektedir. İstanbul'un bir deprem bölgesi olduğu düşünüldüğünde ve geçmişte yaşanan 1999 yılı depremi ve sonuçları anımsandığında, bu oranın çok daha az olması gerekmektedir.

Evlere ait yatak odası sayıları incelendiğinde evlerin %39,3'ünün 2 yatak odasına sahip olduğu, %37,8'inin ise 3 yatak odasının bulunduğu tespit edilmiştir. Son yıllarda tek odalı evlerin sayısında bir artış olmasına rağmen 1 yatak odalı evlerin oranı çalışmada %8.7 olarak bulunmuştur. Evlere ait ısınma sistemleri incelendiğinde satılık evlerin %69,1'inde doğalgaz sistemi bulunurken %27,7'sinde merkezi sistem bulunmaktadır. Balkon durumu incelendiğinde; evlerin %83'ünde balkon bulunurken, %17'sinde balkon bulunmamaktadır. Evlerde bulunan banyo sayısı incelendiğinde ise evlerin %53,3'ünde 1 adet banyo bulunurken %41,7'sinde 2 adet banyo bulunmaktadır.

65.726 adet satılık ev incelendiğinde; bu evlerin %72'sinin kat mülkiyetli tapuya sahip olduğu, %22'sinin ise kat irtifaklı tapuya sahip olduğu tespit edilmiştir. Satışta olan evlerin krediye uygunlukları incelendiğinde; evlerin %87,3'ünün krediye uygun olduğu, %11'inin ise krediye uygun olmadığı tespit edilmiştir. Satılık evlerin site içerisinde olup olmadığı incelendiğinde; %63,7'sinin site içerisinde bulunmadığı, %36,2'sinin ise site içerisinde bulunduğu tespit edilmiştir. Evler kullanım durumlarına göre incelendiğinde; %42,7'sinin boş olduğu, %35,8'inin kiracı tarafından ve %21,5'inin mülk sahibi tarafından kullanıldığı tespit edilmiştir. Evlerin eşyalı olarak mı yoksa eşyasız olarak mı satışa sunulduğu incelendiğinde; evlerin %92,9'u eşyasız olarak satışa sunulduğu, %4,2'sinin ise eşyalı olarak satışa sunulduğu tespit edilmiştir. Satılık evlere ilişkin fiyat, brüt m^2 ve net m^2 değişkenlerine ait tanımlayıcı istatistikler Tablo 10'da verilmiştir.

Tablo 10. Fiyat, Brüt m^2, Net m^2 ve Aidat için Tanımlayıcı İstatistikler

Tanımlayıcı İstatistikler	Fiyat	Brüt m^2	Net m^2	Aidat
Ev Sayısı	65.726	65.726	65.726	26989*
Ortalama	3.321.779	131,82	112,34	266,83
Standart Sapma	1.740.368	52,37	45,84	330,66
Çarpıklık	159,29	1,88	1,83	2,58
Basıklık	26.670	9,80	9,64	18,52
Minimum	750.000	20	15	0
Maksimum	12.500.000	10000	5555	5000

* 38.737 adet evin aidat bilgisi belirtilmemiştir.

Tablo 10 incelendiğinde İstanbul'da satışa sunulan 65.726 adet evin ortalama fiyatının 3.321.779₺ olduğu, satışa sunulan evlerin büyüklüğü incelendiğinde ortalama brüt m^2 132 olarak bulunurken ortalama net m^2 112 olarak tespit edilmiştir. Satışa sunulan evlerin aidat bilgisi incelenmek istendiğinde 65.726 adet evden 38.737 tanesinin aidat bilgisini belirtmediği tespit edilmiş ve aidat bilgisi belirtilen 26.989 adet evin ortalama aidatının ise 266,83₺ olduğu görülmüştür.

4.6. Analiz ve Bulgular

Derin öğrenme modelleri Python programlama dili kullanılarak modellenmiş ve veri analizinde derin öğrenmenin temelinde bulunan başlangıç ağırlıklarının rasgele olmasının sonuçlara etkisini azaltmak için her öneri 30 kez tekrarlanmış ve bu tekrara ait ortalamalar sonuç olarak kabul edilmiştir. Sonuçlar IBM SPSS 28 programı kullanılarak analiz edilmiştir.

Derin öğrenme modelinde aşırı öğrenmeyi engellemek amacıyla dropout miktarı 0,2, iterasyon miktarı (epoch) 10000 olarak belirlenmiştir.

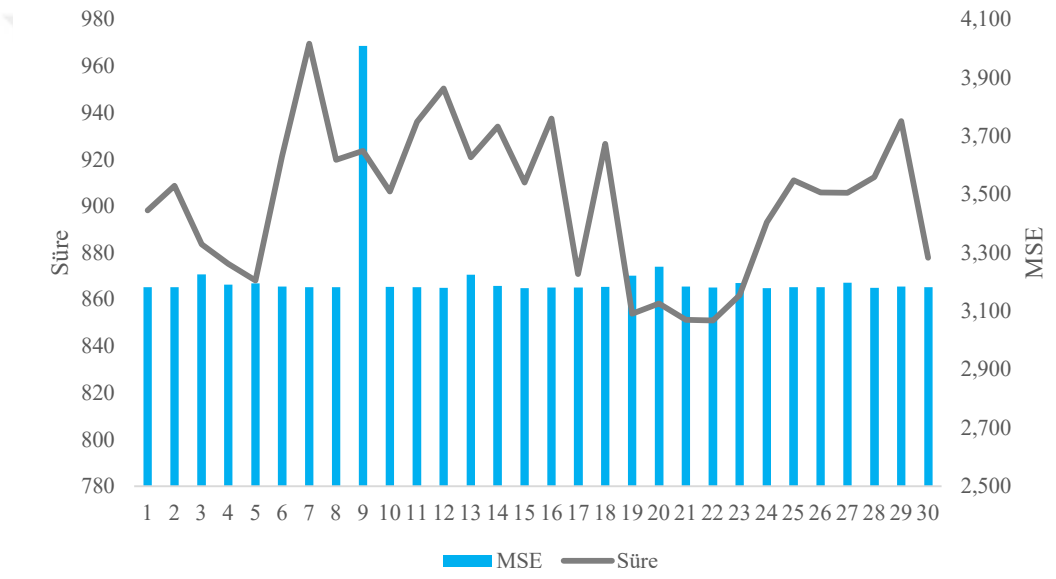
Genetik algoritmanın oluşturulma aşamasında popülasyon değeri 20, jenerasyon değeri 10 olarak belirlenmiştir. Genetik algoritma ile oluşturulacak derin öğrenme ağına ait nöron sayıları iki tabanlı sisteme uygun olarak 64, 128, 256, 512, 768 ve 1024 olarak belirlenmiştir. Gizli katmanlarda aktivasyon fonksiyonu olarak relu, elu, tanh ve sigmoid fonksiyonlarının kullanılabileceği belirtilmiştir. Optimize edici olarak rmsprop, adam, sgd, adagrad, adadelta, adamax ve nadam parametrelerinin kullanılabileceği belirtilmiştir.

Mutasyon işleminde koruma (retain) (her nesilden sonra elde tutulacak nüfus yüzdesi) değeri %20, reddedilen bir derin öğrenme modelinin popülasyonda kalma olasılığı %10, mutasyon şansı %20 olarak belirlenmiştir.

4.6.1.Gerçek Verilerin Analizi

Huang tarafından ortaya atılmış tek katmanlı derin öğrenme formülasyonuna göre nöron sayısı 775 olarak hesaplanmıştır. 50.000 birimlik veriye tek katman ve 775 nöron sayılı derin öğrenme ağı oluşturulmuştur.

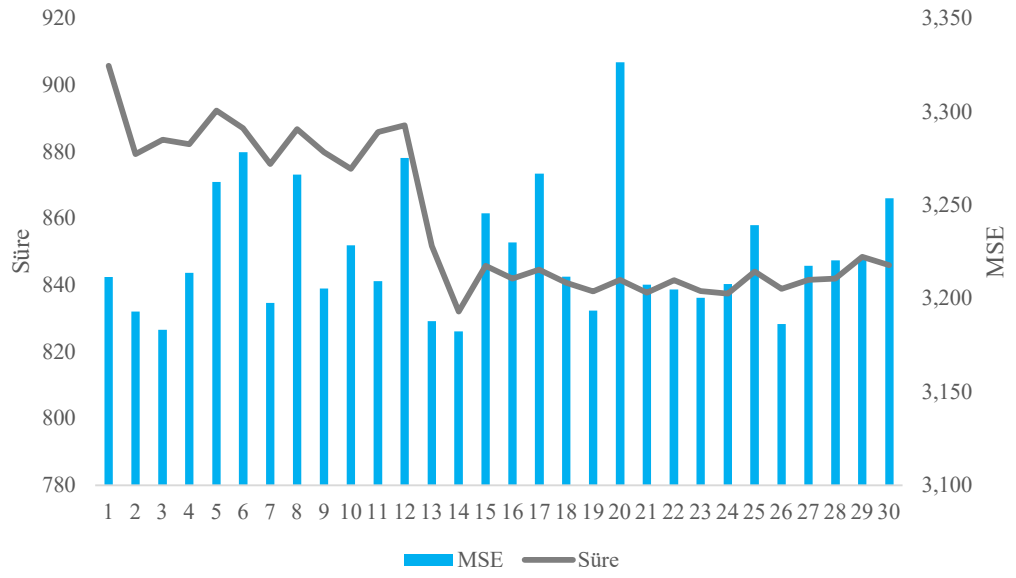
Oluşturulan ağ yapısı test verisi ile test edildiğinde 30 tekrar sonucunda ortalama MSE 3,2243 olarak hesaplanmış ve her bir ağın işlemi ortalama 902,71 saniye sürmüştür. 30 tekrara ait dağılım grafiğine Şekil 26'da yer verilmiştir.



Şekil 26. Huang Tek Katmanlı Derin Öğrenme Analizi Sonuç Dağılımı

Huang tarafından ortaya atılmış çift katmanlı derin öğrenme formülasyonuna göre ilk gizli katmandaki nöron sayısı 574, ikinci gizli katmandaki nöron sayısı 387 olarak hesaplanmıştır. 50.000 birimlik veriyle belirtilen derin öğrenme ağı oluşturulmuştur.

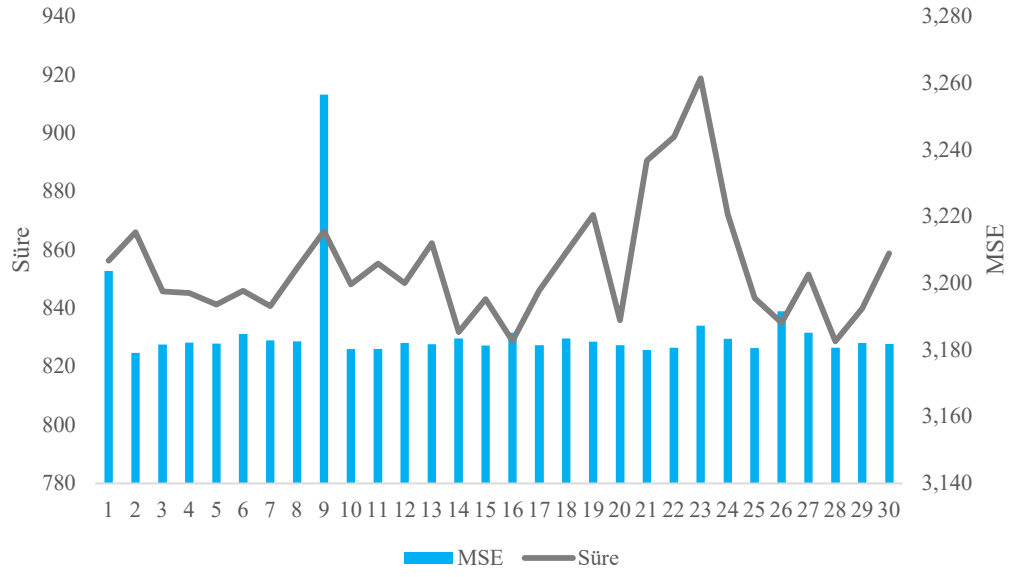
Oluşturulan ağ yapısı test verisi ile test edildiğinde 30 tekrar sonucunda ortalama MSE 3,2243 olarak hesaplanmış ve her bir ağın işlemi ortalama 859,18 saniye sürmüştür. 30 tekrara ait dağılım grafiğine Şekil 27'de yer verilmiştir.



Şekil 27. Huang Çift Katmanlı Derin Öğrenme Analizi Sonuç Dağılımı

Heaton tarafından önerilen kurallar çerçevesinde katmandaki nöron sayısı 604 olarak hesaplanmıştır. 50.000 birimlik veriyle belirtilen derin öğrenme ağı oluşturulmuştur.

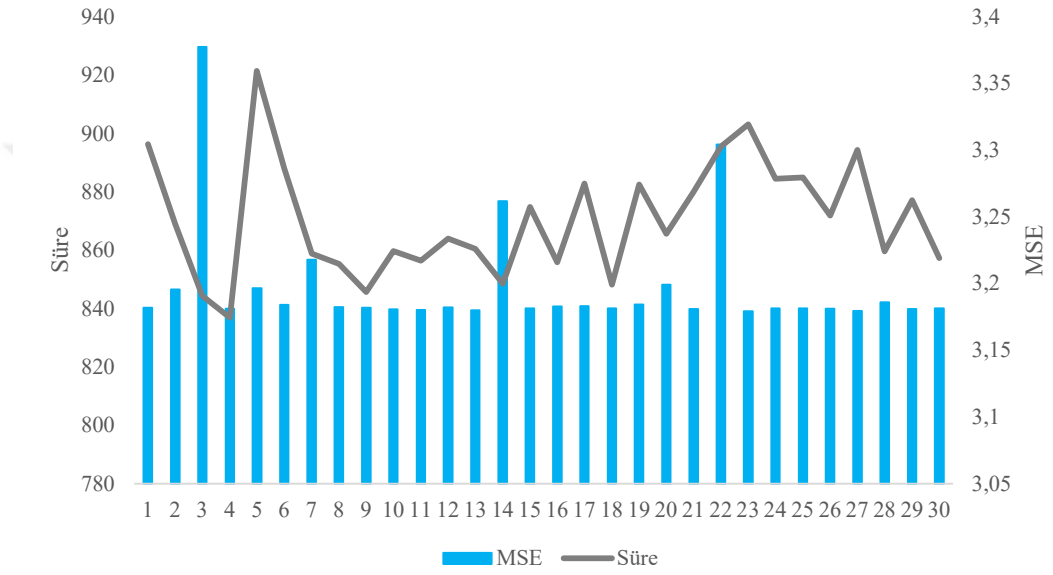
Oluşturulan ağ yapısı test verisi ile test edildiğinde 30 tekrar sonucunda ortalama MSE 3,1857 olarak hesaplanmış ve her bir ağın işlemi ortalama 854,33 saniye sürmüştür. 30 tekrara ait dağılım grafiğine Şekil 28'de yer verilmiştir.



Şekil 28. Heaton Kuralları Derin Öğrenme Analizi Sonuç Dağılımı

Temel Bileşenler Analizi sonucunda katmandaki nöron sayısı 751 olarak ve varyans açıklama oranı ise %88 olarak hesaplanmıştır. 50.000 birimlik veriyle belirtilen derin öğrenme ağı oluşturulmuştur.

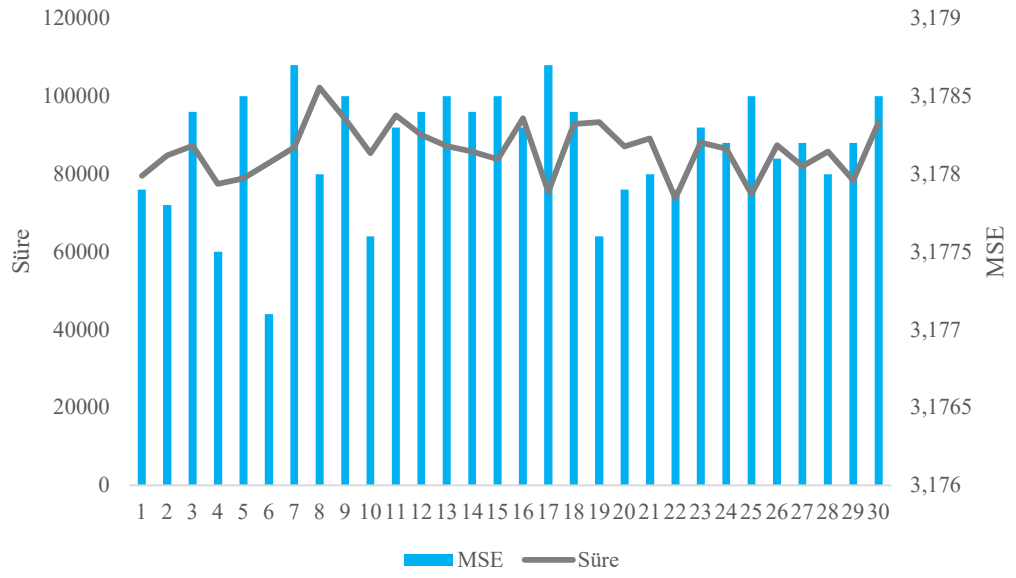
Oluşturulan ağ yapısı test verisi ile test edildiğinde 30 tekrar sonucunda ortalama MSE 3,1979 olarak hesaplanmış ve her bir ağın işlemi ortalama 879,83 saniye sürmüştür. 30 tekrara ait dağılım grafiğine Şekil 29'da yer verilmiştir.



Şekil 29. Faktör Analizi Derin Öğrenme Sonuç Dağılımı

Genetik Algoritma sonucunda gerçekleştirilen öğrenme sürecinde farklı ağ sonuçları elde edilmiştir. En iyi sonucu veren ağ tek katmanlı ve 128 nöron sayısına sahip olarak hesaplanmıştır. 50.000 birimlik veriyle belirtilen derin öğrenme ağı oluşturulmuştur.

Oluşturulan ağ yapısı test verisi ile test edildiğinde 30 tekrar sonucunda ortalama MSE 3,1782 olarak hesaplanmış ve her bir ağın işlemi ortalama 86.127,92 saniye sürmüştür. 30 tekrara ait dağılım grafiğine Şekil 30'da yer verilmiştir.



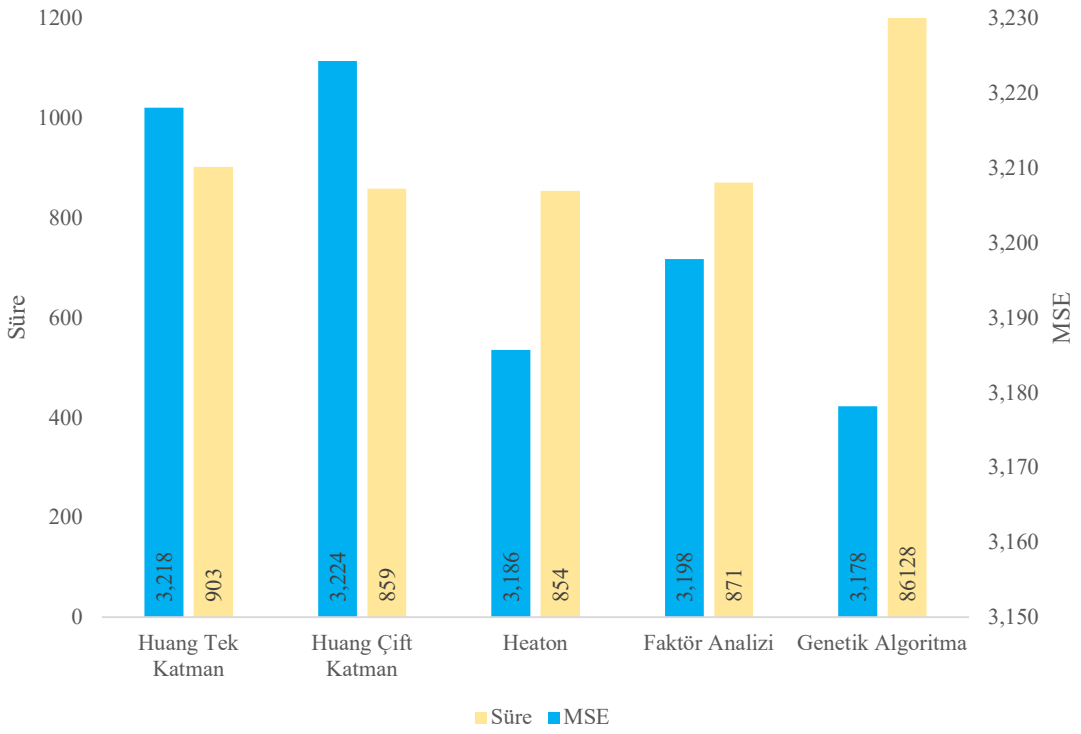
Şekil 30. Genetik Algoritma Derin Öğrenme Sonuç Dağılımı

Çalışmada gerçekleştirilen yöntemlerin ortalama hata miktarları ve ortalama sürelerine ilişkin veriler Tablo 11'de gösterilmiştir.

Tablo 11. Yöntemlere Göre Ortalama Sonuçlar

	Sinir Yapısı	**MSE**	**Süre**
Genetik Algoritma	(128)	3,1782	86127,90
Heaton Kuralları	(604)	3,1857	854,33
Faktör Analizi	(751)	3,1979	870,83
Huang Tek Katman	(775)	3,2181	902,71
Huang Çift Katman	(574-387)	3,2243	859,18

Tablo 11'e göre yapılan analizde farklı sinir ağı yapılarına sahip modellerin performansı değerlendirilmiştir. Her bir modelin sinir yapısı, ortalama kare hata (MSE) değeri ve hesaplama süresi kaydedilmiştir. Yöntemlere ait MSE değerleri karşılaştırmalı olarak Şekil 31'de verilmiştir



Şekil 31. Yöntemlerin Karşılaştırılması

İlk olarak, "Genetik Algoritma" yöntemi kullanılarak oluşturulan modelde 128 düğümden oluşan bir sinir ağı yapısı kullanılmıştır. Bu modelin MSE değeri 3,1782 olarak hesaplanmış ve hesaplama süresi 86.127,90 saniye olarak kaydedilmiştir.

Diğer bir yöntem olan "Heaton Kuralları" modelinde ise 604 düğümden oluşan bir sinir ağı yapısı kullanılmıştır. MSE değeri 3,1857 olarak hesaplanmış ve hesaplama süresi 854,33 saniye olarak belirlenmiştir.

"Faktör Analizi" yönteminde ise sinir ağı yapısı 8 düğümden oluşmaktadır. Bu modelin MSE değeri 3,1982 olarak hesaplanmış ve hesaplama süresi 870,83 saniye olarak kaydedilmiştir.

"Huang Tek Katman" modelinde 775 düğümden oluşan bir sinir ağı yapısı kullanılmıştır. MSE değeri 3,2181 olarak hesaplanmış ve hesaplama süresi 902,71 saniye olarak kaydedilmiştir.

Son olarak, "Huang Çift Katman" modelinde ilk katmanda 574 düğüm ve ikinci katmanında 387 düğüm kullanılmıştır. MSE değeri 3,2243 olarak hesaplanmış ve hesaplama süresi 859,18 saniye olarak belirlenmiştir.

Elde edilen sonuçlar arasındaki farklılıkların tespiti amacıyla Kruskal Wallis testi uygulanmıştır. Test sonucu Tablo 12'de gösterilmiştir. Kruskall Wallis testine ait hipotez aşağıdaki gibi oluşturulmuştur.

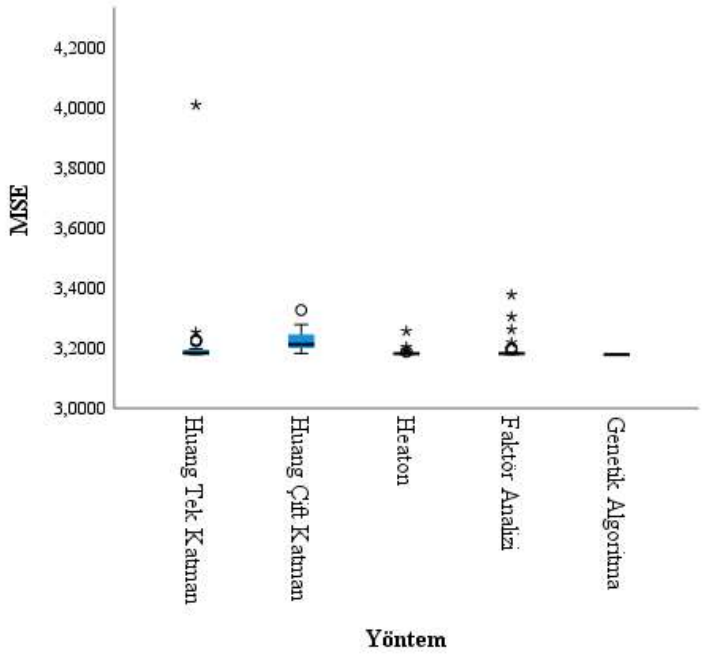
H_0 : Yöntemlere ait MSE değerleri arasında fark yoktur.

H_1 : Yöntemlerden birine ait MSE değeri diğerlerinden farklıdır.

Tablo 12. Kruskal Wallis Testi

Gözlem Sayısı	150
Test İstatistiği	97,429
Serbestlik Derecesi	4
P Değeri (Çift Taraflı)	>0,000

Tablo 12'de Kruskal Wallis testine ait değerler incelenmiş ve p değeri 0,000'dan küçük olarak bulunmuş, elde edilen p değeri, kritik değer olan 0,05'ten küçük olduğu için H_0 hipotezi reddedilmiş ve yöntemlerden birine ait MSE değerinin diğer yöntemlerden farklı olduğu tespit edilmiştir. Yöntemlerin MSE değerlerine ait kutu grafiği Şekil 26'da gösterilmektedir.



Şekil 32. Yöntemlerin MSE Değerlerine Ait Kutu Grafiği

Şekil 32 incelendiğinde yöntemlerin medyan değerlerinin benzerlik gösterdiği ve Huang Tek Katmanlı yöntem, Heaton yöntemi, Faktör Analizi yöntemi ve genetik algoritma yöntemlerine ait değişim aralıklarının oldukça küçük olduğu tespit edilmiştir.

Huang'a ait Tek Katmanlı yöntem ile elde edilen MSE değerlerine ait dağılımda yukarı yönlü uç değerler tespit edilmiştir.

Huang'a ait Çift Katmanlı yöntem ile elde edilen MSE değerlerine ait dağılımın, diğer yöntemlere göre daha geniş bir aralıkta dağılım gösterdiği tespit edilmiştir.

Heaton'a ait yöntem ile elde edilen MSE değerlerine ait dağılımda 1 tane yukarı yönlü uç değerin olduğu, Faktör Analizi ile elde edilen MSE değerlerine ait dağılımda 4 tane yukarı yönlü uç değerin olduğu tespit edilmiştir.

Genetik algoritma ile elde edilen MSE değerine ait dağılımın diğer yöntemlere göre değişim aralığı oldukça küçük olduğu tespit edilmiştir. Elde edilen bu sonuçlar genetik algoritma yönteminin diğer yöntemlere göre daha tutarlı sonuçlar elde ettiğini göstermektedir.

Yöntemlere ait MSE değerlerinin arasındaki farkın ikili karşılaştırmalar ile test edilip yöntemler arasındaki farkın tespiti için post-hoc testi gerçekleştirilmiş ve sonuçlar Tablo 13'te gösterilmiştir. İkili karşılaştırmalara ait örnek hipotez aşağıdaki gibi oluşturulmuştur.

Tablo 13. İkili Karşılaştırma Sonuçları

Yöntem 1	Yöntem 2	Test İstatistiği	Standart Hata	Standart Test İstatistiği	P Değeri
Genetik Algoritma	Heaton	58,000	11,217	5,171	**>,001**
Genetik Algoritma	Faktör Analizi	63,033	11,217	5,619	**>,001**
Genetik Algoritma	Huang Tek Katman	69,767	11,217	6,220	**>,001**
Genetik Algoritma	Huang Çift Katman	109,200	11,217	9,735	**>,001**
Heaton	Faktör Analizi	-5,033	11,217	-,449	,654
Heaton	Huang Tek Katman	11,767	11,217	1,049	,294
Heaton	Huang Çift Katman	51,200	11,217	4,565	**>,001**
Faktör Analizi	Huang Tek Katman	6,733	11,217	,600	,548
Faktör Analizi	Huang Çift Katman	46,167	11,217	4,116	**>,001**
Huang Çift Katman	Huang Tek Katman	-39,433	11,217	-3,516	**>,001**

H_0 : İki yönteme ait MSE değerleri arasında fark yoktur.

H_1 : İki yönteme ait MSE değerleri arasında fark vardır.

Tablo 13 incelendiğinde Genetik Algoritma yöntemi ile diğer yöntemlere ait karşılaştırmalar sonucunda p değerleri 0,000'dan küçük olarak bulunmuştur. Elde edilen p değerleri, kritik değer olan 0,05'ten küçük olduğu için H0 hipotezi reddedilmiş ve genetik algoritma yöntemine ait MSE değerleri diğer yöntemlerden farklı olduğu tespit edilmiştir.

Heaton yöntemine ait sonuçlar incelendiğinde Faktör Analizi ile yapılan karşılaştırma sonucunda p değeri 0,654 olarak, Huang tek katmanlı yöntem ile yapılan karşılaştırma sonucunda p

değeri 0,294 olarak elde edilmiştir. Diğer taraftan Faktör Analizi yöntemine ait sonuçlar incelendiğinde Huang Tek Katmanlı yöntem ile yapılan karşılaştırma sonucunda p değeri 0,548 olarak elde edilmiş, elde edilen p değeri, kritik değer olan 0,05'ten büyük olduğu için H_0 hipotezi kabul edilmiş ve yöntemlere ait MSE değerleri arasında fark bulunmadığı saptanmıştır.

Heaton yöntemi ile Huang Tek Katman yöntemlerine ait karşılaştırmalar sonucunda p değerleri 0,001'den küçük olarak bulunmuştur. Elde edilen p değeri, kritik değer olan 0,05'ten küçük olduğu için H_0 hipotezi reddedilmiş ve Heaton yöntemi ile Huang Çift Katman yöntemi arasında istatistiksel olarak anlamlı bir fark bulunduğu sonucuna varılmıştır.

Faktör analizine ait sonuçlar incelendiğinde Huang'a Çift Katman ile yapılan karşılaştırma sonucunda p değerleri sırasıyla 0,001'den küçük olarak bulunmuştur. Elde edilen p değerleri, kritik değer olan 0,05'ten küçük olduğu için H_0 hipotezi reddedilir, Faktör analizi yöntemi ile Huang Çift Katman yöntemleri arasında istatistiksel olarak anlamlı bir fark bulunduğu sonucuna varılmıştır.

Huang'a ait çift katmanlı analize ait sonuçlar incelendiğinde tek katmanlı yöntem ile yapılan karşılaştırma sonucunda p değeri 0,001 olarak bulunmuştur. Elde edilen p değerleri, kritik değer olan 0,05'ten küçük olduğu için H_0 hipotezi reddedilmiş ve Huang'a ait çift katmanlı yöntem ile tek katmanlı yöntem arasında istatistiksel olarak anlamlı bir fark bulunduğu sonucuna varılmıştır. Yöntemler için ortalama ranklara ait sonuçlar Tablo 14'te verilmiştir.

Tablo 14. Yöntemlere Ait Ranklar

Yöntem	Rank
Genetik Algoritma	15,50
Heaton	73,50
Faktör Analizi	78,53
Huang Tek Katman	85,27
Huang Çift Katman	124,70

Tablo 14 incelendiğinde genetik algoritma yöntemine ait ortalama rank 15,50 olarak elde edilmiş olup bu değer diğer yöntemlere ait ortalama ranklara göre oldukça düşüktür. Heaton yöntemine ait ortalama rank değeri 73,50 olarak, Faktör Analizine ait ortalama rank değeri 78,53 olarak ve Huang Tek Katman yöntemine ait ortalama rank 85,27 olarak hesaplanmıştır. Yöntemlere ait ortalama rankların birbirine yakın olduğu tespit edilmiştir. Huang'a ait çift katmanlı yönteme ait ortalama rank değeri 124,70 olarak olarak elde edilmiştir.

Genetik algoritma yönteminin diğer yöntemlere ait sonuçlara göre daha iyi sonuçlar verdiği tespit edilmiştir. Ancak diğer yöntemlere ait süreler 871 ile 903 saniye arasında değişirken genetik algoritma işlemine ait süre 86.128 saniye olarak elde edilmiştir. Bu sürenin diğer yöntemlere göre oldukça yüksek olduğu görülmektedir.

Heaton, Faktör Analizi ve Huang Tek Katman yöntemlerine ait MSE değerleri arasında fark olmadığı tespit edilmiştir. Bundan dolayı bu yöntemlere ait sonuçların süreleri karşılaştırılarak aralarında fark olup olmadığını incelemek amacıyla Kruskal Wallis testi kullanılmış, sonuçlar Tablo 15'te belirtilmiştir.

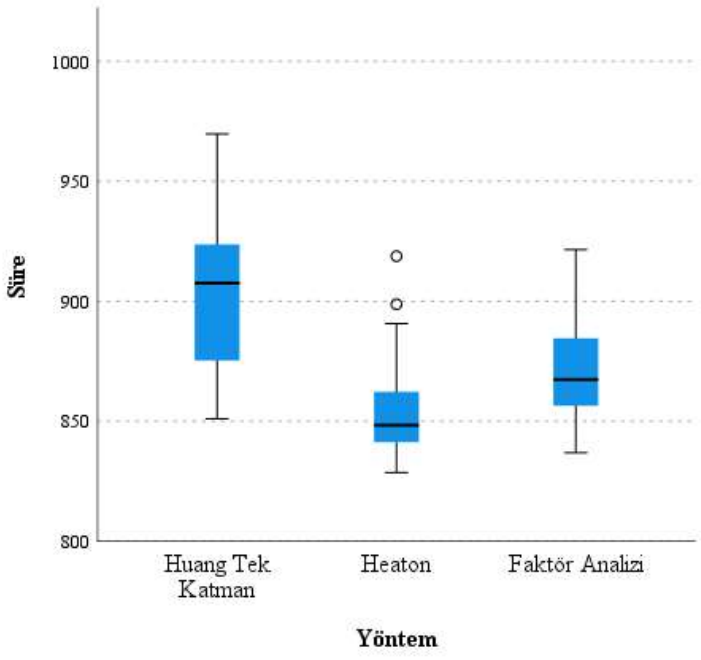
H_0 : Yöntemlere ait süre değerleri arasında fark yoktur.

H_1 : Yöntemlerden birine ait süre değeri diğerlerinden farklıdır.

Tablo 15. Kruskal Wallis Testi

Gözlem Sayısı	90
Test İstatistiği	35,838
Serbestlik Derecesi	2
P Değeri (Çift Taraflı)	>0,001

Tablo 15'te Kruskal Wallis testine ait değerler incelenmiş ve p değeri 0,001'den küçük olarak bulunmuştur. Elde edilen p değeri, kritik değer olan 0,05'ten küçük olduğu için H_0 hipotezi reddedilmiş ve yöntemlerden en az birine ait süre değerinin diğer yöntemlerden farklı olduğu tespit edilmiştir. Yöntemlerin süre değerlerine ait kutu grafiği Şekil 33'te gösterilmektedir.



Şekil 33. Yöntemlerin Süre Değerlerine Ait Kutu Grafiği

Şekil 33 incelendiğinde Huang'a ait Tek Katmanlı yöntem ile elde edilen süre değerlerine ait dağılımın medyan değerinin diğer yöntemlere göre yüksek olduğu ve değişim aralığının diğer yöntemlere göre daha büyük olduğu tespit edilmiştir.

Heaton yöntemi ile elde edilen süre değerlerine ait dağılımın medyan değerinin diğer yöntemlere göre daha küçük olduğu, sonuçların diğer yöntemlere göre daha küçük bir aralık arasında dağılım gösterdiği ve 2 adet yukarı yönlü aykırı değerin olduğu tespit edilmiştir.

Faktör analizi yöntemi ile elde edilen süre değerine ait dağılımın medyan değerinin Heaton yöntemine göre daha büyük ancak Huang Tek Katman yöntemine göre daha küçük olduğu tespit edilmiştir.

Yöntemlere ait süre değerleri arasındaki farkın ikili karşılaştırmalar ile test edilip yöntemler arasındaki farkın tespiti için post-hoc testi gerçekleştirilmiş ve sonuçlar Tablo 16'da gösterilmiştir. İkili karşılaştırmalara ait örnek hipotez aşağıdaki gibi oluşturulmuştur.

H_0 : İki yönteme ait süre değerleri arasında fark yoktur.

H_1 : İki yönteme ait süre değerleri arasında fark vardır.

Tablo 16. İkili Karşılaştırma Sonuçları

Yöntem 1	Yöntem 2	Test İstatistiği	Standart Hata	Standart Test İstatistiği	P Değeri
Heaton	Faktör Analizi	-18,467	6,745	-2,738	**0,006**
Heaton	Huang Tek Katman	40,333	6,745	5,979	**>0,001**
Faktör Analizi	Huang Tek Katman	21,867	6,745	3,242	**0,001**

Tablo 16 incelendiğinde Heaton yöntemi ile diğer yöntemlere ait karşılaştırmalar sonucunda p değerleri sırasıyla 0,006 ve 0,001'den küçük olarak bulunmuştur. Elde edilen p değerleri, kritik değer olan 0,05'ten küçük olduğu için H_0 hipotezi reddedilmiş ve Heaton yöntemine ait süre değerlerinin diğer yöntemlerden farklı olduğu tespit edilmiştir.

Faktör analizi yöntemi ile Huang Tek Katman yöntemlerine ait karşılaştırma sonucunda p değeri 0,001 olarak bulunmuştur. Elde edilen p değeri, kritik değer olan 0,05'ten küçük olduğu için H0 hipotezi reddedilmiş ve Faktör Analizi yöntemine ait süre değerlerinin, Huang Tek Katman yönteminden farklı olduğu saptanmıştır. Yöntemler için ortalama ranklara ait sonuçlar Tablo 17'de verilmiştir.

Tablo 17. Yöntemlere Ait Ranklar

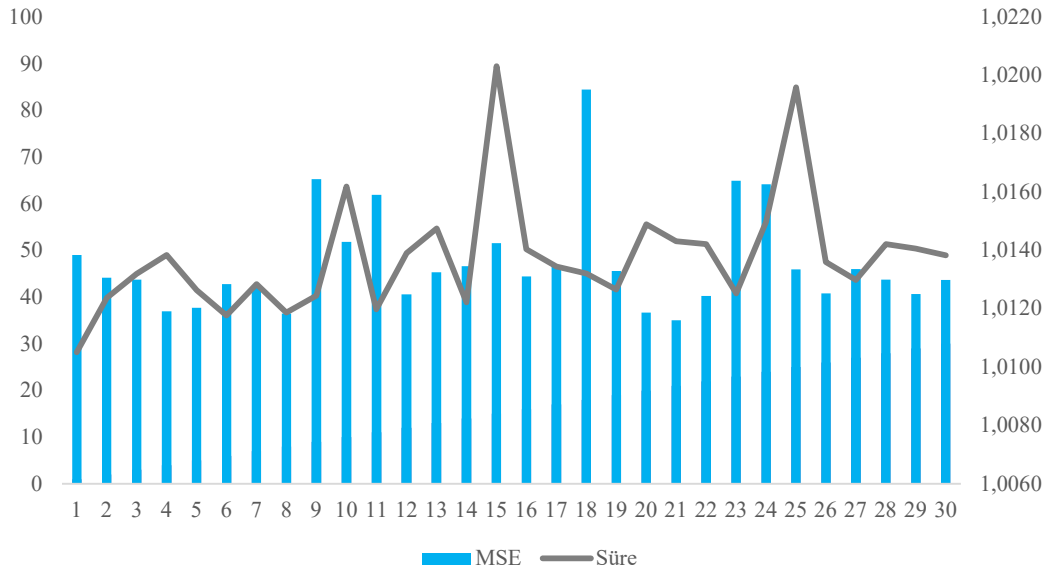
Yöntem	Rank
Heaton	25,90
Faktör Analizi	44,37
Huang Tek Katman	66,23

Tablo 17 incelendiğinde Heaton yöntemine ait ortalama rank 25,90 olarak elde edilmiş olup bu değer diğer yöntemlere ait ortalama ranklara göre oldukça düşüktür. Faktör analizi yöntemine ait ortalama rank değeri 44,37 olarak, Huang Tek Katman analizine ait ortalama rank değeri 66,23 olarak hesaplanmıştır. Sürelere ilişkin veriler incelendiğinde benzer MSE değeri sonuçlarını veren yöntemler arasında Heaton yönteminin en hızlı şekilde hesaplama işlemini gerçekleştirdiği, sonrasında sırasıyla Faktör Analizinin ve Huang Tek Katman yönteminin geldiği tespit edilmiştir.

4.6.2. Teorik Verilerin Analizi

Huang tarafından ortaya atılmış tek katmanlı derin öğrenme formülasyonuna göre nöron sayısı 775 olarak hesaplanmıştır. 50.000 birimlik veri için tek katman ve 775 nöron sayısına sahip derin öğrenme ağı oluşturulmuştur.

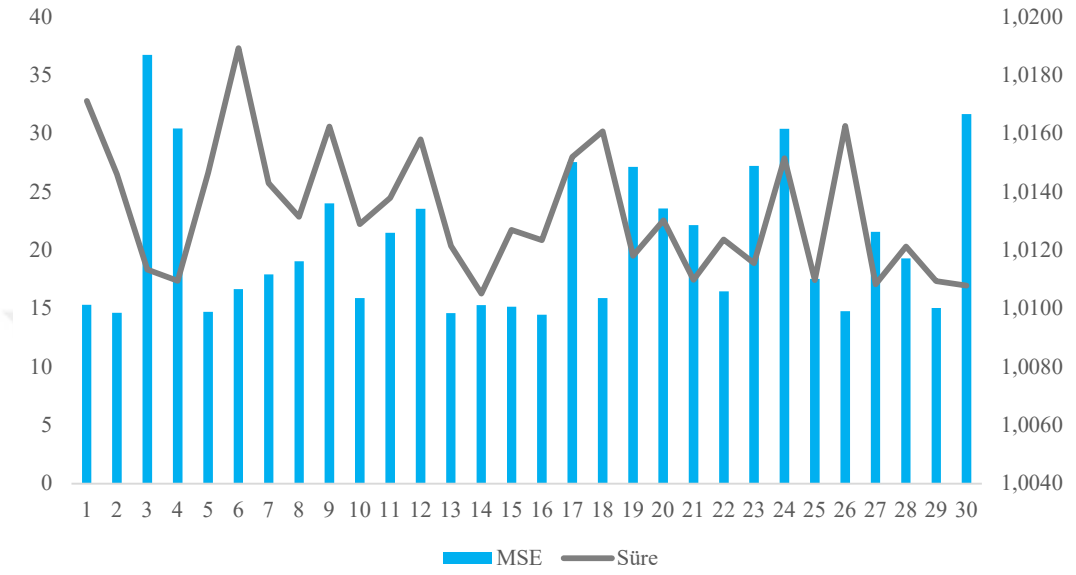
Oluşturulan ağ yapısı test verisi ile test edildiğinde 30 tekrar sonucunda ortalama MSE 1,0135 olarak hesaplanmış ve her bir ağın işlemi ortalama 48,64 saniye sürmüştür. 30 tekrara ait dağılım grafiği Şekil 34'te yer verilmiştir.



Şekil 34. Huang Tek Katmanlı Derin Öğrenme Analizi Sonuç Dağılımı

Huang tarafından ortaya atılmış çift katmanlı derin öğrenme formülasyonuna göre ilk gizli katmandaki nöron sayısı 574, ikinci gizli katmandaki nöron sayısı 387 olarak hesaplanmıştır. 50.000 birimlik veriyle belirtilen derin öğrenme ağı oluşturulmuştur.

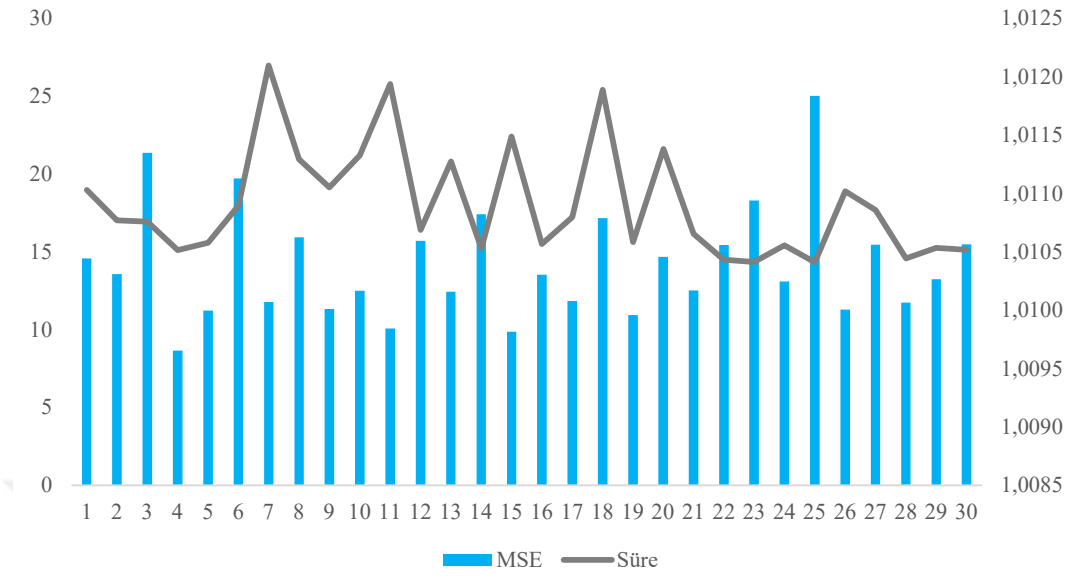
Oluşturulan ağ yapısı test verisi ile test edildiğinde 30 tekrar sonucunda ortalama MSE 1,0123 olarak hesaplanmış ve her bir ağın işlemi ortalama 23,29 saniye sürmüştür. 30 tekrara ait dağılım grafiğine Şekil 35'te yer verilmiştir.



Şekil 35. Huang Çift Katmanlı Derin Öğrenme Analizi Sonuç Dağılımı

Heaton tarafından önerilen kurallar çerçevesinde katmandaki nöron sayısı 10 olarak hesaplanmıştır. 50.000 birimlik veriyle belirtilen derin öğrenme ağı oluşturulmuştur.

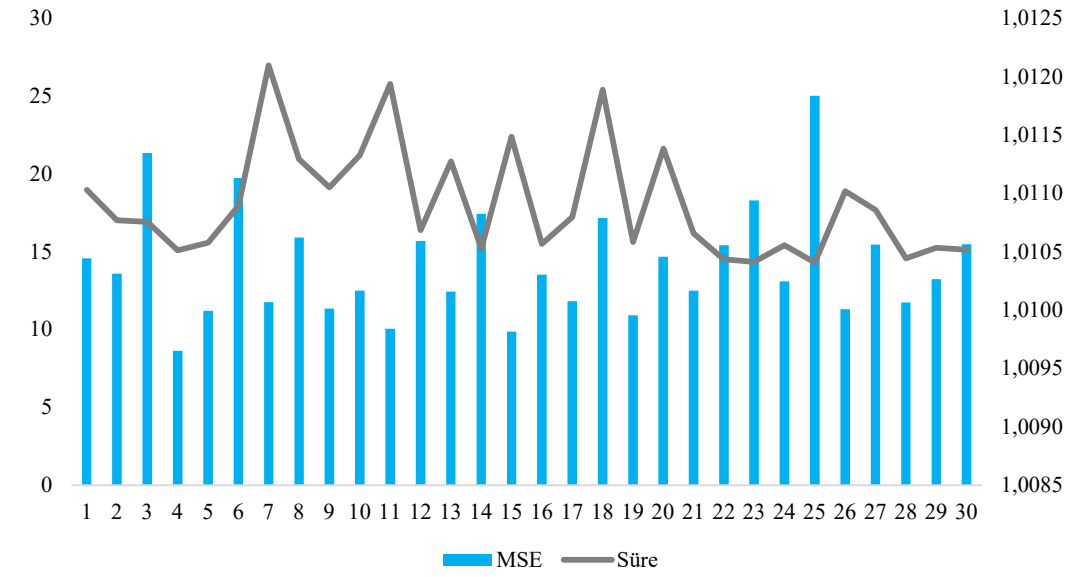
Oluşturulan ağ yapısı test verisi ile test edildiğinde 30 tekrar sonucunda ortalama MSE 1,0104 olarak hesaplanmış ve her bir ağın işlemi ortalama 18,07 saniye sürmüştür. 30 tekrara ait dağılım grafiğine Şekil 36'da yer verilmiştir.



Şekil 36. Heaton Kuralları Derin Öğrenme Analizi Sonuç Dağılımı

Temel Bileşenler Analizi sonucunda katmandaki nöron sayısı 8 olarak ve varyans açıklama oranı ise %54 olarak hesaplanmıştır. 50.000 birimlik veriyle belirtilen derin öğrenme ağı oluşturulmuştur.

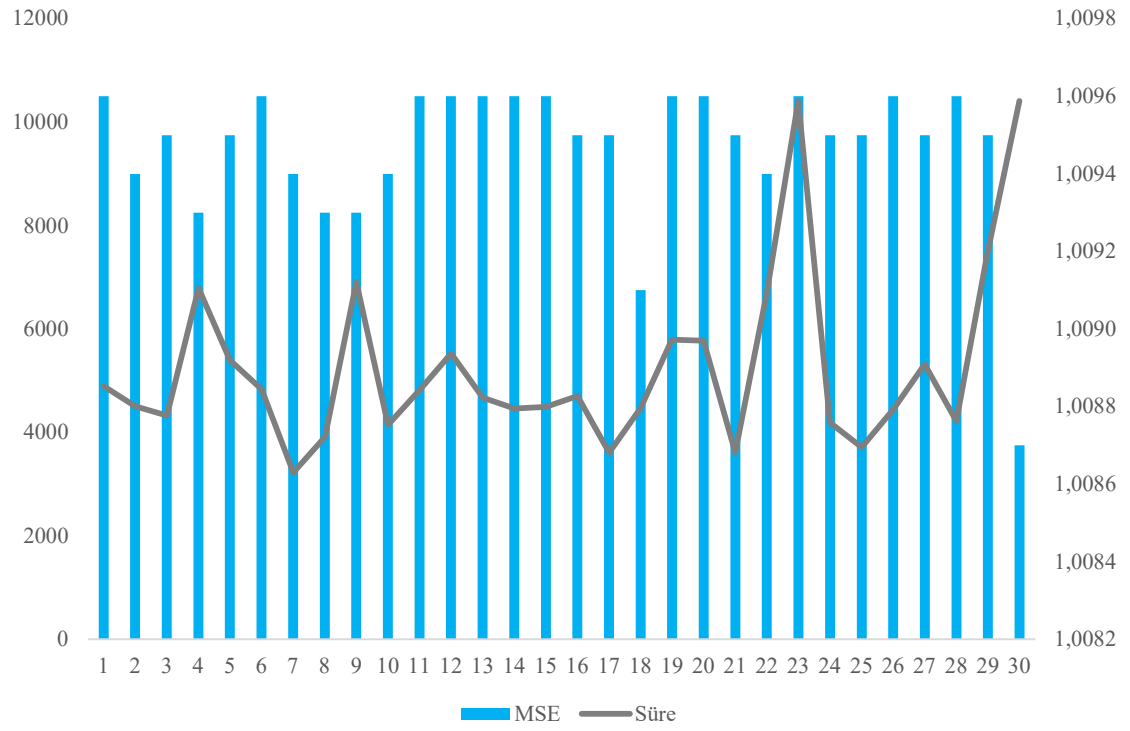
Oluşturulan ağ yapısı test verisi ile test edildiğinde 30 tekrar sonucunda ortalama MSE 1,0105 olarak hesaplanmış ve her bir ağın işlemi ortalama 17,82 saniye sürmüştür. 30 tekrara ait dağılım grafiğine Şekil 37'de yer verilmiştir.



Şekil 37. Faktör Analizi Derin Öğrenme Sonuç Dağılımı

Genetik Algoritma sonucunda gerçekleştirilen öğrenme sürecinde farklı ağ sonuçları elde edilmiştir. En iyi sonucu veren ağ tek katmanlı ve 64 nöron sayısına sahip olarak hesaplanmıştır. 50.000 birimlik veriyle belirtilen derin öğrenme ağı oluşturulmuştur.

Oluşturulan ağ yapısı test verisi ile test edildiğinde 30 tekrar sonucunda ortalama MSE 1,0095 olarak hesaplanmış ve her bir ağın işlemi ortalama 5.255,90 saniye sürmüştür. 30 tekrara ait dağılım grafiğine Şekil 38'de yer verilmiştir.



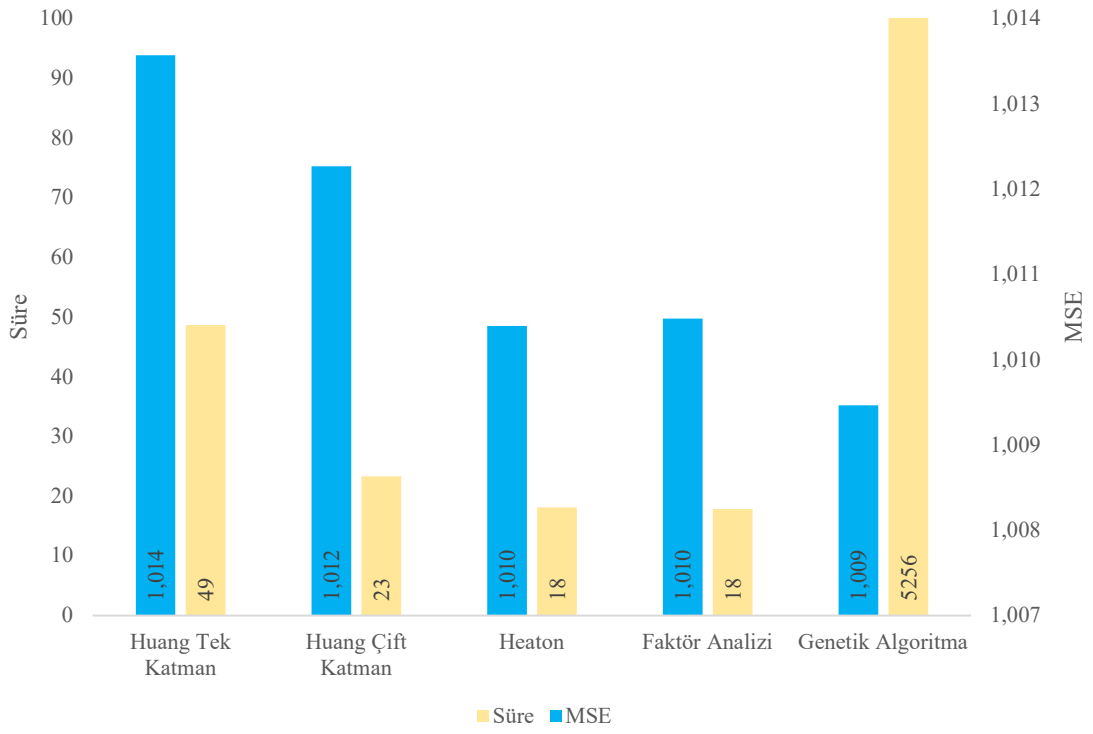
Şekil 38. Genetik Algoritma Derin Öğrenme Sonuç Dağılımı

Çalışmada gerçekleştirilen yöntemlerin ortalama hata miktarları ve ortalama sürelerine ilişkin veriler Tablo 18'de verilmiştir.

Tablo 18. Yöntemlere Göre Ortalama Sonuçlar

	Sinir Yapısı	**MSE**	**Süre**
Genetik Algoritma	(64)	1,0095	5255,9000
Heaton Kuralları	(10)	1,0103	18,0742
Faktör Analizi	(8)	1,0104	17,8164
Huang Çift Katman	(574-387)	1,0122	23,2850
Huang Tek Katman	(775)	1,0135	48,6359

Tablo 18'e göre yapılan analizde farklı sinir ağı yapılarına sahip modellerin performansı değerlendirilmiştir. Her bir modelin sinir yapısı, ortalama kare hata (MSE) değeri ve hesaplama süresi kaydedilmiştir. Yöntemlere ait MSE değerleri karşılaştırmalı olarak Şekil 39'da verilmiştir.



Şekil 39. Yöntemlerin Karşılaştırılması

İlk olarak, Genetik Algoritma yöntemi kullanılarak oluşturulan modelde 64 düğümden oluşan bir sinir ağı yapısı kullanılmıştır. Bu modelin MSE değeri 1,0095 olarak hesaplanmış ve hesaplama süresi 5.255,9000 saniye olarak kaydedilmiştir.

Diğer bir yöntem olan "Heaton Kuralları" modelinde ise 10 düğümden oluşan bir sinir ağı yapısı kullanılmıştır. MSE değeri 1,0103 olarak hesaplanmış ve hesaplama süresi 18,0742 saniye olarak belirlenmiştir.

"Faktör Analizi" yönteminde ise sinir ağı yapısı 8 düğümden oluşmaktadır. Bu modelin MSE değeri 1,0104 olarak hesaplanmış ve hesaplama süresi 17,8164 saniye olarak kaydedilmiştir.

"Huang Çift Katman" modelinde 574 giriş düğümü ve 387 çıkış düğümü kullanılmıştır. MSE değeri 1,0122 olarak hesaplanmış ve hesaplama süresi 23,2850 saniye olarak belirlenmiştir.

Son olarak, "Huang Tek Katman" modelinde 775 düğümden oluşan bir sinir ağı yapısı kullanılmış ve MSE değeri 1,0135 olarak hesaplanmış, hesaplama süresi 48,6359 saniye olarak kaydedilmiştir.

Elde edilen sonuçlar arasındaki farklılıkların tespiti amacıyla Kruskal Wallis testi uygulanmıştır. Test sonucu Tablo 19'da gösterilmiştir. Kruskal Wallis testine ait hipotez aşağıdaki gibi oluşturulmuştur.

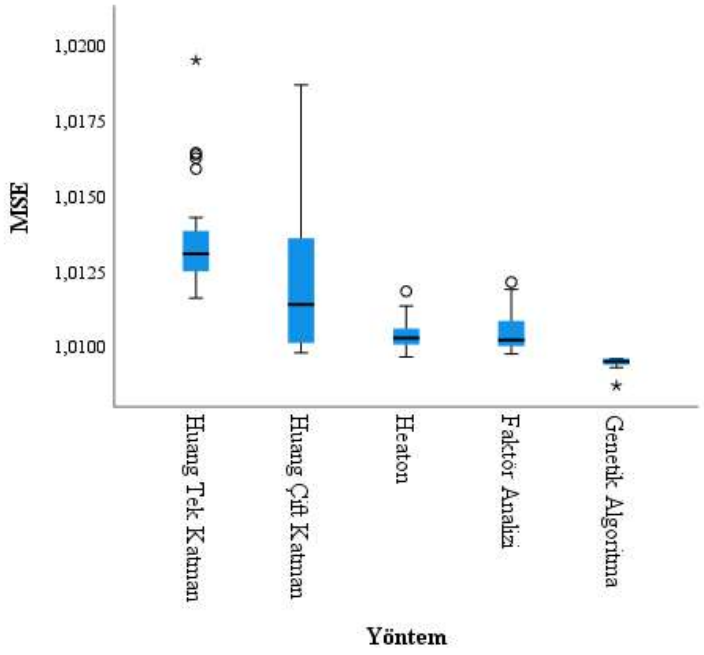
H_0 : Yöntemlere ait MSE değerleri arasında fark yoktur.

H_1 : Yöntemlerden birine ait MSE değeri diğerlerinden farklıdır.

Tablo 19. Kruskal Wallis Testi

Gözlem Sayısı	150
Test İstatistiği	105,758
Serbestlik Derecesi	4
P Değeri (Çift Taraflı)	>0,000

Tablo 19'da Kruskal Wallis testine ait değerler incelenmiş ve p değeri 0,000'dan küçük olarak bulunmuştur. Elde edilen p değeri, kritik değer olan 0,05'ten küçük olduğu için H_0 hipotezi reddedilmiş ve yöntemlerden birine ait MSE değerinin diğer yöntemlerden farklı olduğu tespit edilmiştir. Yöntemlerin MSE değerlerine ait kutu grafiği Şekil 40'ta gösterilmektedir.



Şekil 40. Yöntemlerin MSE Değerlerine Ait Kutu Grafiği

Şekil 40 incelendiğinde Huang'a ait Tek Katmanlı yöntem ile elde edilen MSE değerlerine ait dağılımın medyan değerinin diğer yöntemlere göre yüksek olduğu ve yukarı yönlü uç değerlerin olduğu tespit edilmiştir.

Huang'a ait Çift Katmanlı yöntem ile elde edilen MSE değerlerine ait dağılımın medyan değerinin Tek katmanlı yönteme göre daha küçük olduğu ancak diğer yöntemlere göre daha yüksek

olduğu görülmektedir. Ayrıca Huang'a ait Çift Katmanlı yönteme ait sonuçların geniş bir aralık arasında dağılım gösterdiği de Şekil 40'ta görülmektedir.

Heaton'a ait yöntem ve Faktör Analizi ile elde edilen MSE değerlerine ait dağılımların birbirine benzerlik gösterdiği görülmektedir. Medyan değerleri birbirlerine oldukça yakın olmakla birlikte her iki yöntemde de birer adet yukarı yönde aykırı değer tespit edilmiştir.

Genetik algoritma ile elde edilen MSE değerine ait dağılımın diğer yöntemlere göre değişim aralığının oldukça küçük olduğu tespit edilmiş olup bu da genetik algoritma yönteminin diğer yöntemlere göre daha tutarlı sonuçlar elde ettiğini göstermektedir. Genetik algoritma ile elde edilen dağılımda bir adet aşağı yönde uç değer bulunmaktadır.

Yöntemlere ait MSE değerlerinin arasındaki farkı ikili karşılaştırmalar ile test edilip yöntemler arasındaki farkın tespiti için post-hoc testi gerçekleştirilmiş ve sonuçlar Tablo 20'de gösterilmiştir. İkili karşılaştırmalara ait örnek hipotez aşağıdaki gibi oluşturulmuştur.

Tablo 20. İkili Karşılaştırma Sonuçları

Yöntem 1	Yöntem 2	Test İstatistiği	Standart Hata	Standart Test İstatistiği	P Değeri
Genetik Algoritma	Heaton	53,883	11,213	4,805	**>0,000**
Genetik Algoritma	Faktör Analizi	54,633	11,213	4,872	**>0,000**
Genetik Algoritma	Huang Çift Katman	81,117	11,213	7,234	**>0,000**
Genetik Algoritma	Huang Tek Katman	110,367	11,213	9,842	**>0,000**
Heaton	Faktör Analizi	-,750	11,213	-,067	,947
Heaton	Huang Çift Katman	27,233	11,213	2,429	**,015**
Heaton	Huang Tek Katman	56,483	11,213	5,037	**>0,000**
Faktör Analizi	Huang Çift Katman	26,483	11,213	2,362	**,018**
Faktör Analizi	Huang Tek Katman	55,733	11,213	4,970	**>0,000**
Huang Çift Katman	Huang Tek Katman	29,250	11,213	2,609	**,009**

H_0 : İki yönteme ait MSE değerleri arasında fark yoktur.

H_1 : İki yönteme ait MSE değerleri arasında fark vardır.

Tablo 20 incelendiğinde, Genetik Algoritma yöntemi ile diğer yöntemlere ait karşılaştırmalar sonucunda p değerleri 0,000'dan küçük olarak bulunmuştur. Elde edilen p değerleri, kritik değer olan 0,05'ten küçük olduğu için H_0 hipotezi reddedilirmiş ve genetik algoritma yöntemine ait MSE değerlerinin diğer yöntemlerden farklı olduğu tespit edilmiştir.

Heaton yöntemine ait sonuçlar incelendiğinde Faktör Analizi ile yapılan karşılaştırma sonucunda p değeri 0,947 olarak elde edilmiştir. Elde edilen p değeri, kritik değer olan 0,05'ten büyük olduğu için H_0 hipotezi kabul edilmekte ve iki yönteme ait MSE değerleri arasında fark bulunmadığı sonucuna varılmaktadır. Heaton yöntemi ile Huang yöntemlerine ait karşılaştırmalar sonucunda p değerleri sırasıyla 0,015 ve 0,000'dan küçük olarak bulunmuştur. Elde edilen p değerleri, kritik değer olan 0,05'ten küçük olduğu için H_0 hipotezi reddedilmekte ve Heaton yöntemi ile Huang yöntemleri arasında istatistiksel olarak anlamlı bir fark bulunduğu sonucuna varılmaktadır.

Faktör Analizine ait sonuçlar incelendiğinde Huang'a ait diğer yöntemler ile yapılan karşılaştırmalar sonucunda p değerleri sırasıyla 0,018 ve 0,000'dan küçük olarak bulunmuştur. Elde edilen p değerleri, kritik değer olan 0,05'ten küçük olduğu için H_0 hipotezi reddedilmekte, Faktör analizi yöntemi ile Huang yöntemleri arasında istatistiksel olarak anlamlı bir fark bulunmaktadır.

Huang'a ait çift katmanlı analize ait sonuçlar incelendiğinde tek katmanlı yöntem ile yapılan karşılaştırma sonucunda p değeri 0,009 olarak bulunmuştur. Elde edilen p değeri, kritik değer olan 0,05'ten küçük olduğu için H_0 hipotezi reddedilmekte ve Huang'a ait çift katmanlı yöntem ile tek katmanlı yöntem arasında istatistiksel olarak anlamlı bir fark bulunmaktadır. Yöntemlere göre ortalama ranklara ait sonuçlar Tablo 21'de verilmiştir.

Tablo 21. Yöntemlere Ait Ranklar

Yöntem	Rank
Genetik Algoritma	15,50
Heaton	69,38
Faktör Analizi	70,13
Huang Çift Katman	96,62
Huang Tek Katman	125,87

Tablo 21 incelendiğinde genetik algoritma yöntemine ait ortalama rank 15,50 olarak elde edilmiş olup bu değer diğer yöntemlere ait ortalama ranklara göre oldukça düşüktür. Heaton yöntemine ait ortalama rank değeri 69,38 olarak, Faktör Analizine ait ortalama rank değeri ise 70,13 olarak hesaplanmıştır. İki yönteme ait ortalama rankların birbirine oldukça yakın olduğu tespit edilmiştir. Huang'a ait çift katmanlı yönteme ait ortalama rank değeri 96,62 olarak tek katmanlı yönteme ait ortalama rank 125,87 olarak elde edilmiştir.

Genetik algoritma yönteminin diğer yöntemlere ait sonuçlara göre daha iyi sonuçlar verdiği tespit edilmiştir. Ancak diğer yöntemlere ait süreler 18 ile 48 saniye arasında değişirken genetik algoritma işlemine ait süre 5.255 olarak elde edilmiştir. Bu sürenin diğer yöntemlere göre oldukça yüksek olduğu görülmektedir.

Heaton yöntemi ile Faktör Analizine ait MSE değerleri arasında fark olmadığı tespit edilmiştir. Bundan dolayı bu iki yönteme ait sonuçların süreleri karşılaştırılarak aralarında fark olup olmadığını incelemek amacıyla Mann Whitney U testi kullanılmış, sonuçlar Tablo 22'de belirtilmiştir.

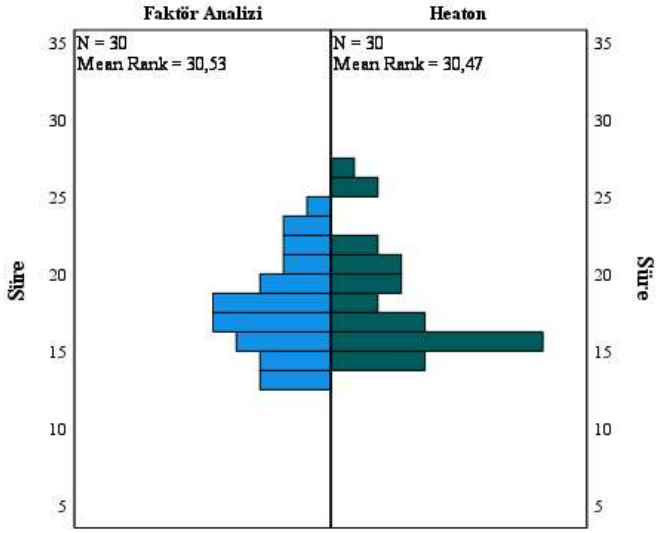
H_0 : İki yönteme ait süreler arasında fark yoktur.

H_1 : İki yönteme ait süreler arasında fark vardır.

Tablo 22. Mann Whitney U Testi

Toplam N	60
Mann-Whitney U	451,000
Wilcoxon W	916,000
Test İstatistiği	451,000
Standart Hata	67,639
Standartlaştırılmış Test İstatistiği	,015
P Değeri (Çift Taraflı)	,988

Tablo 22 incelendiğinde Heaton yöntemi ile Faktör Analizine ait karşılaştırma sonucunda p değeri 0,988 olarak bulunmuştur. Elde edilen p değeri, kritik değer olan 0,05'ten büyük olduğu için H_0 hipotezi kabul edilmiş ve Heaton yöntem ile FAk ait sonuçların süreleri arasında istatistiksel olarak anlamlı bir farklılık olmadığı tespit edilmiştir. Yöntemlerin sürelerine ait dağılımlar ve ortalama ranklar Şekil 41'de verilmiştir.



Şekil 41. Faktör Analizi ve Heaton Yöntemine Ait Süre Dağılımları

Şekil 41 incelendiğinde Faktör Analizine ait ortalama rank 30,53 olarak Heaton yöntemine ait ortalama rank 30,47 olarak elde edilmiştir.

Teorik veri üzerinde yapılan çalışmalar sonucunda genetik algoritma yönteminin en iyi sonucu verdiği, Faktör Analizi ve Heaton yönteminin hem MSE değeri hem de süre olarak birbirine benzer sonuçlar elde ettiği tespit edilmiştir. Diğer taraftan Huang'ın çift katmanlı yönteminin tek katmanlı yönteme göre daha iyi sonuçlar verdiği görülmüştür. Elde edilen sonuçların gerçek veriler üzerinde test edilip benzer sonuçlar verip vermediği bir sonraki bölümde incelenmiştir

SONUÇ

Bu çalışmanın temel amacı, derin öğrenme modellerinde gizli katman ve nöron sayılarının belirlenmesi için önerilen yöntemlerin başarı sonuçlarını aynı veri seti üzerinde uygulayarak karşılaştırmaktır. Huang Önerisi, Heaton Önerisi, Temel Bileşenler Analizi ve Genetik Algoritmalar gibi literatürde bulunan öneriler, pratikte nasıl performans gösteriyor ve hangi durumlarda daha etkili olabiliyor, bu çalışma kapsamında detaylı bir şekilde incelenmektedir. Bu kapsamda çalışmada, derin öğrenme modellerinde gizli katman ve nöron sayılarının belirlenmesi geliştirilmiş mevcut yöntemler ayrıntılı olarak incelenmiş, gerçek ve teorik veriler kullanılarak en etkili ve optimize edilmiş yaklaşımın hangisi olduğu ortaya çıkarılmaya çalışılmıştır. Ayrıca gerçek ve teorik veriler üzerinde bu yöntemlerin performansları da karşılaştırılmıştır.

Araştırmanın veri seti, İstanbul'da bulunan 750.000₺ ile 12.500.000₺ arasındaki satılık evlerin tamamını içermektedir. Bu evlere ait geniş bir veri seti, Türkiye'nin en büyük satılık ev ilan sitesi olan sahibinden.com üzerinden web kazıma yöntemi kullanılarak elde edilmiştir. Ayrıca belirli bir dağılıma sahip verilerde yöntemlerin performansını incelemek amacıyla 50.000 gözleme sahip teorik bir veri seti oluşturulmuştur.

Araştırma yöntemi, gerçek verilerin web kazıma ile elde edilmesi, teorik verilerin istatistiksel dağılımlar kullanılarak oluşturulması ve derin öğrenme yöntemlerinin uygulanması adımlarını içermektedir. Analiz aşamasında, elde edilen sonuçların istatistiksel olarak değerlendirilmesi için Kruskal Wallis ve Mann Whitney U testi kullanılmıştır.

Bu çalışmanın tamamlanmasıyla, derin öğrenme modellerinde gizli katman ve nöron sayılarının belirlenmesi konusunda daha etkili bir yöntemin ortaya çıkması beklenmektedir. Elde edilen bulguların bilimsel literatüre yeni bir perspektif sunması ve derin öğrenme modellerinin daha geniş bir uygulama alanına yayılmasına katkı sağlaması hedeflenmektedir.

İstanbul'da satışta bulunan evler incelendiğinde 5.334 adet ev ile Kadıköy ilçesi satılık evin en fazla olduğu, 4.101 adet ev ile Başakşehir ikinci sırada, 3.966 ev ile Beylikdüzü üçüncü sırada yer almaktadır. En az satılık olan evlere göre ilçeler incelendiğinde 128 adet ev ile Çatalca birinci sırada, 142 adet ev ile Şile ikinci sırada yer alırken almaktadır. En çok satılık ev bulunan ilçeler Kentsel dönüşümün olduğu veya gelişimi hız kazanmış ilçeler olduğu görülmektedir. En az satılık ev olan ilçelerin ise daha çok yazlık amacıyla tercih edilen ilçeler olduğu görülmektedir.

Satılık evlerin özelliklerine bakıldığında büyük çoğunluğunun 5 – 10 yıl arasında, kat mülkiyetli, 2 yatak odasına sahip, doğalgaz ile ısınma sistemi olan, krediye uygun, site içerisinde olmayan, boş durumda olan, eşyasız, balkonlu ve 1 adet banyosu bulunan evler olduğu görülmektedir.

65.726 satılık ev incelenmiş ve elde edilen verilere göre ortalama fiyatın 3.321.779₺ olduğu ortalama 131 metre kare brüt 112 metre kare net genişliğe sahip olduğu görülmektedir. 65.726 adet evden 26.989 tanesinin aidat bilgisine ulaşılmış olup ortalama aidatın 266,83₺ olduğu görülmektedir.

50.000 gözlemli teorik veriden elde edilen sonuçlara göre en düşük MSE değeri (1,0095) ve 5255,90 saniye ile genetik algoritma ilk sırada, Heaton tarafından önerilen kurallarla oluşturulan ağ 1,0103 MSE değeri ve 18,07 saniye ile ikinci sırada, 1,0104 MSE değeri ve 17,82 saniye ile Faktör Analizi üçüncü sırada, Huang tarafından önerilen çift katmanlı ağ 1,0122 MSE değeri ve 23,29 saniye ile dördüncü sırada ve Huang tarafından önerilen tek katmanlı ağ 1,0135 MSE değeri ve 48,64 saniye ile beşinci sırada yer almaktadır.

50.000 gözlemli gerçek veriden elde edilen sonuçlara göre en düşük MSE değeri (3,1782) ve 86.127,90 saniye ile genetik algoritma ilk sırada, Heaton tarafından önerilen kurallarla oluşturulan ağ 3,1857 MSE değeri ve 854,33 saniye ile ikinci sırada, 3,1979 MSE değeri ile 870,83 saniye ile Faktör Analizi üçüncü sırada, Huang tarafından önerilen tek katmanlı ağ 3,2181 MSE değeri ve 902,71 saniye ile dördüncü sırada ve Huang tarafından önerilen çift katmanlı ağ 3,2243 MSE değeri ve 859,18 saniye ile beşinci sırada yer almaktadır.

Gerçek ve teorik veriler karşılaştırıldığında sıralamadaki ilk üç yöntemin sırasıyla genetik algoritma, Heaton önerisi ve Faktör Analizinin değişmediği ancak Huang tarafından önerilen çift katmanlı yöntemin teorik veriler için daha iyi sonuçlar verirken gerçek verilerde tek katmanlı yönteme göre daha düşük sonuçlar verdiği tespit edilmiştir.

Gerçek ve teorik veriler üzerinde her yöntem 30 kez tekrar edilmiş ve elde edilen sonuçlar birleştirilerek Kruskal – Wallis yöntemi ile aralarında istatistiksel olarak anlamlı bir fark olup olmadığı incelenmiştir. İncelenen sonuçlar kapsamında hem teorik hem de gerçek verilere ait Kruskal – Wallis testleri sonucunda yöntemlerden en az birinin diğer yöntemlerden farklı olduğu görülmüştür. Farklı yöntemlerin tespit edilmesi amacıyla Post-Hoc analizine geçilmiş ve yöntemler ikili olarak karşılaştırılmıştır.

Teorik verilere ait Post-Hoc testlerinin sonuçlarına göre MSE değeri bakımından en düşük tespit edilen genetik algoritmaya ait sonuçlarda p değerlerinin %5 anlam düzeyinin altında olduğu tespit edilmiştir. Genetik algoritma ile oluşturulan ağın performansı diğer yöntemlere göre istatistiksel açıdan farklıdır. Heaton yöntemi ile diğer yöntemler karşılaştırıldığında Faktör Analizi ile istatistiksel olarak benzer sonuçlar gösterdiği, Huang'a ait çift ve tek katmanlı yöntemlerden istatistiksel açıdan daha iyi

bir performans gösterdiği görülmüştür. Faktör analizi ile Huang'a ait yöntemler karşılaştırıldığında istatistiksel açıdan daha iyi performans verdiği tespit edilmiştir. Huang'a ait yöntemler de birbiri ile karşılaştırıldığında çift katmanlı yöntemin istatistiksel açıdan tek katmanlı yönteme göre daha iyi performans verdiği belirlenmiştir.

Teorik veriler göre, genetik algoritma en iyi performans ile birinci sırada, Heaton yöntemi ve Faktör Analizi benzer sonuçları göstererek ikinci sırada, Huang çift katmanlı ağ üçüncü sırada ve Huang tek katmanlı ağ dördüncü sırada olduğu belirlenmiştir.

Teorik veriler üzerinde benzer sonuçları gösteren Heaton yöntemi ve Faktör Analizine ait süreler Mann Whitney U Testi ile karşılaştırılmış ve süreleri arasında istatistiksel açıdan fark olmadığı tespit edilmiştir.

Gerçek verilere ait Post-Hoc testlerine ait sonuçlara göre MSE değeri bakımından en düşük tespit edilen genetik algoritmaya ait sonuçlarda p değerleri %5'in altında elde edilmiştir. Genetik algoritma ile oluşturulan ağın performansı diğer yöntemlere göre istatistiksel açıdan farklıdır. Heaton yöntemi ile diğer yöntemler karşılaştırıldığında Faktör Analizi ve Huang'a ait tek katmanlı yöntem ile istatistiksel olarak benzer sonuçlar gösterdiği, Huang'a ait çift katmanlı yöntemden istatistiksel açıdan daha iyi bir performans gösterdiği tespit edilmiştir. Faktör analizi yöntemi ile diğer yöntemler karşılaştırıldığında Huang'a ait tek katmanlı yöntem ile istatistiksel olarak benzer sonuçlar gösterdiği, Huang'a ait çift katmanlı yöntemden istatistiksel açıdan daha iyi bir performans gösterdiği tespit edilmiştir. Huang'a ait yöntemler de birbiri ile karşılaştırıldığında tek katmanlı yöntemin istatistiksel açıdan çift katmanlı yönteme göre daha iyi performans verdiği belirlenmiştir.

Gerçek veriler göre, genetik algoritmanın en iyi performans ile birinci sırada, Heaton yöntemi, Faktör Analizi ve Huang tek katmanlı yöntemlerin benzer sonuçları göstererek ikinci sırada, Huang çift katmanlı yönteminin üçüncü sırada olduğu belirlenmiştir.

Gerçek veriler üzerinde benzer performans gösteren Heaton yöntemi, Faktör Analizi ve Huang tek katmanlı yöntemlere ait süreler Kruskal Wallis analizi ile karşılaştırılmış ve süreler açısından istatistiksel olarak bir fark tespit edilmiştir. İkili karşılaştırmalar sonucunda süreler açısından yöntemlerin hepsinin birbirinden istatistiksel olarak farklı olduğu, Heaton yönteminin diğer yöntemlerden daha hızlı olduğu, ikinci sırada Faktör Analizinin olduğu ve benzer MSE değeri gösteren yöntemler arasında en yavaş olanın Huang tek katmanlı yöntem olduğu tespit edilmiştir.

Bu çalışma, derin öğrenme modellerinin uygulanmasında genetik algoritma, Heaton tarafından önerilen kurallar, Faktör Analizi ve Huang'un önerdiği tek ve çift katmanlı ağlar arasında karşılaştırmalı bir analiz sunarak, bu yöntemlerin teorik ve gerçek verilere göre performanslarını değerlendirmiştir. Teorik verilere dayalı sonuçlar, genetik algoritmanın en düşük MSE ile birinci sırada yer aldığını

gösterirken, gerçek verilere dayalı sonuçlar benzer bir sıralamayı ortaya koymuştur. Bu çalışma, genetik algoritmanın öne çıktığı ve çeşitli durumlarda diğer yöntemlerden daha etkili olduğu bir perspektif sunmaktadır. Ayrıca, çalışmada Heaton tarafından önerilen kuralların ve Faktör Analizinin benzer sonuçlar gösterdiği tespit edilmiştir. Ancak, Huang'un önerdiği tek katmanlı ağın gerçek verilerde daha iyi performans gösterdiği, teorik verilere göre ise çift katmanlı ağın daha iyi sonuçlar verdiği ilginç bir bulgu olarak öne çıkmıştır.

Sınırlamalar açısından, bu çalışmanın belirli bir veri seti üzerinde gerçekleştirildiği ve sonuçların genelleme yeteneğinin bu veri setiyle sınırlı olduğu belirtilmelidir. Ayrıca, derin öğrenme modellerinin hiper parametrelerinin belirlenmesindeki yöntemlerin etkisi üzerine daha fazla detay verilebilirdi. Bununla birlikte, bu çalışma, derin öğrenme modellerinin önerilen yöntemlerle karşılaştırılmasında geniş bir perspektif sunmakta ve gelecekteki çalışmalar için bir temel oluşturmaktadır.

Gelecekte yapılacak araştırmalarda, öncelikle farklı derin öğrenme modellerinin veya optimizasyon yöntemlerinin bu spesifik uygulama bağlamında daha ayrıntılı bir şekilde incelenmesi uygun olacaktır. Yapılan çalışmada genetik algoritma, Heaton tarafından önerilen kurallar, Faktör Analizi ve Huang tarafından önerilen çift ve tek katmanlı ağlar karşılaştırılmıştır, ancak literatürde daha yeni geliştirilen yöntemler ve modellerin performansı da değerlendirilebilir. Ayrıca, farklı boyutlardaki veri setleri üzerinde yapılan deneyler, model performansının genellemesi ve güvenilirliği açısından önemlidir. Farklı boyutlardaki veri setleri üzerinde yapılan deneyler, özellikle e-ticaret ve perakende sektörlerindeki gerçek dünya uygulamalarını daha iyi anlamak için değerli bilgiler sunabilir. Ayrıca, model optimizasyonu ve hiper parametre ayarında daha ileri düzeyde araştırmalar, hem hesaplama kaynaklarını daha etkili kullanma hem de daha iyi performans elde etme açısından değerli olacaktır. Bu öneriler, derin öğrenme tabanlı yöntemlerin bu özel uygulama bağlamında daha da geliştirilmesine ve iyileştirilmesine yönelik gelecek araştırmalara rehberlik edebilir.

KAYNAKÇA

Agostinelli, F., Hoffman, M., Sadowski, P. ve Baldi, P. (2014). Learning Activation Functions to Improve Deep Neural Networks. doi:10.48550/ARXIV.1412.6830

Bessaou, M. ve Siarry, P. (2001). A genetic algorithm with real-value coding to optimize multimodal continuous functions. *Structural and Multidisciplinary Optimization*, *23*(1), 63-74. doi:10.1007/s00158-001-0166-y

Cantú-Paz, E. (2000). A Survey of Parallel Genetic Algorithms. https://api.semanticscholar.org/CorpusID:14264381 adresinden erişildi.

Cebeci, Z. (2021). *R ile Genetik Algoritmalar ve Optimizasyon Uygulamaları* (C. 1). Nobel Yayıncılık.

Chahar, V. ve Kumar, D. (2017). An astrophysics-inspired Grey wolf algorithm for numerical optimization and its application to engineering design problems. *Advances in Engineering Software*, *112*, 231-254. doi:10.1016/j.advengsoft.2017.05.008

Chang-Yong Lee. (2003). Entropy-Boltzmann selection in the genetic algorithms. *IEEE Transactions on Systems, Man and Cybernetics, Part B (Cybernetics)*, *33*(1), 138-149. doi:10.1109/TSMCB.2003.808184

Chollet, F. (2018). *Deep learning with Python*. Shelter Island, NY: Manning.

Deb, K. (2001). *Multi-objective optimization using evolutionary algorithms*. Wiley-Interscience series in systems and optimization (1st ed.). Chichester ; New York: John Wiley & Sons.

Di, W., Bhardwaj, A. ve Wei, J. (2018). *Deep learning essentials: Your hands-on guide to the fundamentals of deep learning and neural network modeling*. Birmingham, UK: Packt Publishing.

Elmas, Ç. (2018). *Yapay Zeka Uygulamaları* (4. bs.). Seçkin Yayıncılık.

El-Mihoub, T. A., Hopgood, A. A., Nolle, L. ve Battersby, A. (2006). Hybrid Genetic Algorithms: A Review. *Eng. Lett.* Citeseer. https://citeseerx.ist.psu.edu/document?repid=rep1\&type=pdf\&doi=340b165fd2108f29e9b03629c1e1373cd4163ecf adresinden erişildi.

Fox, B. R. ve McMahon, M. B. (1991). Genetic Operators for Sequencing Problems. *Foundations of Genetic Algorithms* içinde (C. 1, ss. 284-300). Elsevier. doi:10.1016/B978-0-08-050684-5.50021-5

Goldberg, D. E. (1990). Real-coded genetic algorithms, virtual alphabets and blocking. Citeseer. https://citeseerx.ist.psu.edu/document?repid=rep1\&type=pdf\&doi=9b01cbd96b9fe420ebd89c98aff01ffe9f5b809e adresinden erişildi.

Goldberg, David Elliott. (1989). *Genetic algorithms in search, optimization, and machine learning* (30. print.). Boston: Addison-Wesley.

Goodfellow, I., Bengio, Y. ve Courville, A. (2016). *Deep learning*. Adaptive computation and machine learning. Cambridge, Massachusetts: The MIT Press.

Guang-Bin Huang. (2003). Learning capability and storage capacity of two-hidden-layer feedforward networks. *IEEE Transactions on Neural Networks*, *14*(2), 274-281. doi:10.1109/TNN.2003.809401

Guang-Bin Huang ve Babri, H. A. (1997). General approximation theorem on feedforward networks. *Proceedings of ICICS, 1997 International Conference on Information, Communications and Signal Processing. Theme: Trends in Information Systems Engineering and Wireless Multimedia Communications (Cat. No.97TH8237)* içinde (ss. 698-702). 1st International Conference on Information Communications and Signal Processing, sunulmuş bildiri, Singapore: IEEE. doi:10.1109/ICICS.1997.652067

Heaton, Jeff. (2013). *Deep learning and neural networks*. Artificial Intelligence for Humans Vol. 3. St. Louis, MO: Heaton Research, Inc.

Heaton, Jeffrey. (2008). The number of hidden layers. *Heaton Research Inc*.

Hillier, F. S. ve Lieberman, G. J. (2021). *Introduction to operations research* (Eleventh edition.). New York, NY: McGraw-Hill Education.

Holland, J. H. (1975). *Adaptation in natural and artificial systems: An introductory analysis with applications to biology, control, and artificial intelligence*. Ann Arbor: Univ. of Michigan Press.

Holland, J. H. (1992). *Adaptation in natural and artificial systems: An introductory analysis with applications to biology, control, and artificial intelligence*. Complex adaptive systems (Nachdr.). Cambridge, Mass.: MIT Press.

Ibnu Choldun R., Muh., Santoso, J. ve Surendro, K. (2020). Determining the Number of Hidden Layers in Neural Network by Using Principal Component Analysis. Y. Bi, R. Bhatia ve S. Kapoor (Ed.), *Intelligent Systems and Applications* içinde , Advances in Intelligent Systems and Computing (C. 1038, ss. 490-500). Cham: Springer International Publishing. doi:10.1007/978-3-030-29513-4_36

Jebari, K. (2013). Selection Methods for Genetic Algorithms. *International Journal of Emerging Sciences*, *3*, 333-344.

Katoch, S., Chauhan, S. S. ve Kumar, V. (2021). A review on genetic algorithm: Past, present, and future. *Multimedia Tools and Applications*, *80*(5), 8091-8126. doi:10.1007/s11042-020-10139-6

Kora, P. ve Yadlapalli, P. (2017). Crossover Operators in Genetic Algorithms: A Review. *International Journal of Computer Applications*, *162*(10), 34-36. doi:10.5120/ijca2017913370

Kumar, R., Memoria, M., Thapliyal, M., Kirola, M., Ahmad, I., Gupta, A., … Ansari, N. (2022). Analyzing The Performance Of Crossover Operators (OX, OBX, PBX, MPX) To Solve Combinatorial Problems. *2022 International Conference on Machine Learning, Big Data, Cloud and Parallel Computing (COM-IT-CON)* içinde (ss. 817-821). 2022 International Conference on Machine Learning, Big Data, Cloud and Parallel Computing (COM-IT-CON), sunulmuş bildiri, Faridabad, India: IEEE. doi:10.1109/COM-IT-CON54601.2022.9850689

Kumar, V., Chhabra, J. K. ve Kumar, D. (2014). Parameter adaptive harmony search algorithm for unimodal and multimodal optimization problems. *Journal of Computational Science*, *5*(2), 144-155. doi:10.1016/j.jocs.2013.12.001

LeCun, Y., Bengio, Y. ve Hinton, G. (2015). Deep learning. *Nature*, *521*(7553), 436-444. doi:10.1038/nature14539

Li-Jiang, Y. ve Tian-Lun, C. (2002). Application of Chaos in Genetic Algorithms. *Communications in Theoretical Physics*, *38*(2), 168-172. doi:10.1088/0253-6102/38/2/168

Lim, S. M., Sultan, A. B. Md., Sulaiman, Md. N., Mustapha, A. ve Leong, K. Y. (2017). Crossover and Mutation Operators of Genetic Algorithms. *International Journal of Machine Learning and Computing*, *7*(1), 9-12. doi:10.18178/ijmlc.2017.7.1.611

Liu, B., T. Haftka, R., A. Akgün, M. ve Todoroki, A. (2000). Permutation genetic algorithm for stacking sequence design of composite laminates. *Computer Methods in Applied Mechanics and Engineering*, *186*(2-4), 357-372. doi:10.1016/S0045-7825(99)90391-2

Liu, D. (2019). Mathematical modeling analysis of genetic algorithms under schema theorem. *Journal of Computational Methods in Sciences and Engineering*, *19*, 131-137. doi:10.3233/JCM-191019

M. Almufti, S., Ahmad Shaban, A., Arif Ali, Z., Ismael Ali, R. ve A. Dela Fuente, J. (2023). Overview of Metaheuristic Algorithms. *Polaris Global Journal of Scholarly Research and Trends*, *2*(2), 10-32. doi:10.58429/pgjsrt.v2n2a144

Malik, S. K. ve Rizvi, S. (2011). Information Extraction Using Web Usage Mining, Web Scrapping and Semantic Annotation. *2011 International Conference on Computational Intelligence and Communication Networks* içinde (ss. 465-469). 2011 International Conference on

Computational Intelligence and Communication Networks (CICN), sunulmuş bildiri, Gwalior, India: IEEE. doi:10.1109/CICN.2011.97

Mann, H. B. ve Whitney, D. R. (1947). On a Test of Whether one of Two Random Variables is Stochastically Larger than the Other. *The Annals of Mathematical Statistics*, *18*(1), 50-60. doi:10.1214/aoms/1177730491

Mas, J. F. ve Flores, J. J. (2008). The application of artificial neural networks to the analysis of remotely sensed data. *International Journal of Remote Sensing*, *29*(3), 617-663. doi:10.1080/01431160701352154

McKay, B. (1995). Using a tree structured genetic algorithm to perform symbolic regression. *1st International Conference on Genetic Algorithms in Engineering Systems: Innovations and Applications (GALESIA)* içinde (C. 1995, ss. 487-492). 1st International Conference on Genetic Algorithms in Engineering Systems: Innovations and Applications (GALESIA), sunulmuş bildiri, Sheffield, UK: IEE. doi:10.1049/cp:19951096

McKnight, P. E. ve Najab, J. (2010). Mann-Whitney U Test. I. B. Weiner ve W. E. Craighead (Ed.), *The Corsini Encyclopedia of Psychology* içinde (1. bs., ss. 1-1). Wiley. doi:10.1002/9780470479216.corpsy0524

Michalewicz, Z. (1996). *Genetic Algorithms + Data Structures = Evolution Programs*. Berlin, Heidelberg: Springer Berlin Heidelberg. doi:10.1007/978-3-662-03315-9

Mitchell, T. M. (1997). *Machine Learning*. McGraw-Hill series in computer science. New York: McGraw-Hill.

Montgomery, D. C. ve Runger, G. C. (2014). *Applied statistics and probability for engineers* (Sixth edition.). Hoboken, NJ: John Wiley and Sons, Inc.

Mühlenbein, H., Schomisch, M. ve Born, J. (1991). The parallel genetic algorithm as function optimizer. *Parallel Computing*, *17*(6-7), 619-632. doi:10.1016/S0167-8191(05)80052-3

Nandal, V. ve Nandal, D. (2013). GENERIC ALGORITHMS: EVOLUTION, ENCODING AND THEIR APPLICATIONS. *International Journal of Computer Science & Management Studies*, *13*(10).

Nearchou, A. C. (2004). The effect of various operators on the genetic search for large scheduling problems. *International Journal of Production Economics*, *88*(2), 191-203. doi:10.1016/S0925-5273(03)00184-1

Ostertagová, E. ve Ostertag, O. (2013). Methodology and application of oneway ANOVA. *American Journal of Mechanical Engineering*, *1*(7), 256-261.

Ostertagová, E., Ostertag, O. ve Kováč, J. (2014). Methodology and Application of the Kruskal-Wallis Test. *Applied Mechanics and Materials*, *611*, 115-120. doi:10.4028/www.scientific.net/AMM.611.115

Öztemel, E. (2012). *Yapay sinir ağları* (3.basım.). İstanbul: Papatya.

Patel, R. ve Raghuwanshi, M. M. (2010). Review on Real Coded Genetic Algorithms Used in Multiobjective Optimization. *2010 3rd International Conference on Emerging Trends in Engineering and Technology* içinde (ss. 610-613). Third International Conference on Emerging Trends in Engineering and Technology (ICETET 2010), sunulmuş bildiri, Goa: IEEE. doi:10.1109/ICETET.2010.112

Prasad, T. D. ve Park, N.-S. (2004). Multiobjective Genetic Algorithms for Design of Water Distribution Networks. *Journal of Water Resources Planning and Management*, *130*(1), 73-82. doi:10.1061/(ASCE)0733-9496(2004)130:1(73)

Satman, M. H. (2016). *Genetik Algoritmalar* (Paperback.). Türkmen Kitabevi. https://lead.to/amazon/com/?op=bt&la=en&cu=usd&key=6054749668 adresinden erişildi.

Sejnowski, T. J. (2018). *The deep learning revolution*. Cambridge, Massachusetts: The MIT Press.

Sivanandam, S. N. ve Deepa, S. N. (2007). *Introduction to Genetic Algorithms*. Guildford Boulder: Springer London NetLibrary, Inc. [distributor].

Stathakis, D. (2009). How many hidden layers and nodes? *International Journal of Remote Sensing*, *30*(8), 2133-2147. doi:10.1080/01431160802549278

Syaharuddin, S., Fatmawati, F. ve Suprajitno, H. (2022). The Formula Study in Determining the Best Number of Neurons in Neural Network Backpropagation Architecture with Three Hidden Layers. *Jurnal RESTI (Rekayasa Sistem dan Teknologi Informasi)*, *6*(3), 397-402. doi:10.29207/resti.v6i3.4049

Umbarkar, A. J. ve Sheth, P. D. (2015). CROSSOVER OPERATORS IN GENETIC ALGORITHMS: A REVIEW. *ICTACT Journal on Soft Computing*, *06*(01), 1083-1092. doi:10.21917/ijsc.2015.0150

Vujicic, T., Matijevic, T., Ljucovic, J., Balota, A. ve Sevarac, Z. (2016). Comparative analysis of methods for determining number of hidden neurons in artificial neural network. https://api.semanticscholar.org/CorpusID:63991667 adresinden erişildi.

Wright, A. H. (1991). Genetic Algorithms for Real Parameter Optimization. *Foundations of Genetic Algorithms* içinde (C. 1, ss. 205-218). Elsevier. doi:10.1016/B978-0-08-050684-5.50016-1

Yu, F., Fu, X., Li, H. ve Dong, G. (2016). Improved Roulette Wheel Selection-Based Genetic Algorithm for TSP. *2016 International Conference on Network and Information Systems for Computers*

(ICNISC) içinde (ss. 151-154). 2016 International Conference on Network and Information Systems for Computers (ICNISC), sunulmuş bildiri, Wuhan: IEEE. doi:10.1109/ICNISC.2016.041

EKLER

Ek – 1 Teorik Verilere Ait Sonuçlar

Huang Tek Katman (775)		
Tekrar	MSE	Süre
1	1,013844609	28,15901136
2	1,013064265	39,85384393
3	1,013002872	45,00000954
4	1,011913896	49,05450726
5	1,012030125	41,47610569
6	1,012843013	36,08983731
7	1,012714744	42,79776502
8	1,011833787	36,66634464
9	1,016439676	40,32833838
10	1,014290094	63,69728351
11	1,015906334	37,40296102
12	1,012494087	49,42411399
13	1,013247013	54,71202016
14	1,013450503	38,86287713
15	1,014252782	89,44826841
16	1,013105512	50,28330231
17	1,013496637	46,54898739
18	1,019517064	45,04677796
19	1,013294578	41,60271931
20	1,011867523	55,60937142
21	1,011613846	51,94862843
22	1,01243937	51,37375832
23	1,016383648	40,8248992
24	1,016265273	56,04377246
25	1,013356805	84,90818
26	1,012531996	47,47027922
27	1,013359427	43,67609692
28	1,013000727	51,35836959
29	1,012502432	50,40485191
30	1,01299119	49,00421953
Ort	**1,013568461**	**48,63591671**

Huang İlk Gizli Katman (574) İkinci Katman (387)		
Tekrar	MSE	Süre
1	1,010131359	32,7699976
2	1,009847164	26,50722599
3	1,018698692	18,34480095
4	1,016165376	17,38955879
5	1,009878516	26,60469031
6	1,01066339	37,2921741
7	1,011171937	25,72393632
8	1,011617064	22,83298492
9	1,013601184	30,56828618
10	1,010354877	22,2369628
11	1,012595057	24,46628857
12	1,013413906	29,48654127
13	1,009836674	20,35889292
14	1,010117412	16,28184819
15	1,01005733	21,73026466
16	1,009788871	20,85891533
17	1,015014648	27,96304917
18	1,010355711	30,15784955
19	1,014847517	19,51499319
20	1,013429403	22,57060218
21	1,012855411	17,4539423
22	1,010594964	20,91700101
23	1,014886856	18,8785212
24	1,016156435	27,87702179
25	1,011018038	17,42941332
26	1,00990963	30,66114807
27	1,012626886	17,06591225
28	1,011714697	20,29355216
29	1,01001811	17,35607004
30	1,016661286	16,95811725
Ort	**1,012267613**	**23,28501875**

Heaton Tek Katman (10)		
Tekrar	MSE	Süre
1	1,010445476	18,98092794
2	1,010310173	17,01889682
3	1,011348367	16,93442965
4	1,009651899	15,10293579
5	1,009994984	15,59633136
6	1,011129856	17,95880842
7	1,010069966	26,97633362
8	1,010623932	20,95149732
9	1,010011673	19,15042973
10	1,010166526	21,1950984
11	1,009842634	25,78789163
12	1,010593534	16,39607334
13	1,010159373	20,81034231
14	1,010824561	15,22358251
15	1,009816527	22,41037655
16	1,010303855	15,50511479
17	1,010077715	17,23275733
18	1,010789871	25,41018963
19	1,009956598	15,62959075
20	1,010457397	21,62544727
21	1,010167599	16,17108941
22	1,010556817	14,51322293
23	1,010940075	14,35816145
24	1,010246038	15,41420078
25	1,011836648	14,3213222
26	1,010007024	18,90092802
27	1,010561466	17,67754245
28	1,010066509	14,59001493
29	1,010266185	15,25883293
30	1,010564923	15,12587452
Ort	**1,01039294**	**18,07427483**

Faktör Analizi (8)		
Tekrar	MSE	Süre
1	1,011238694	17,74292731
2	1,009751439	19,88122606
3	1,010422111	23,78185153
4	1,010049939	18,32704735
5	1,01075995	16,75764632
6	1,010896921	19,6120379
7	1,010029435	23,0721395
8	1,010080814	15,35238409
9	1,009866357	21,03412104
10	1,009968162	22,32251692
11	1,011902452	15,20686507
12	1,010078073	14,084795
13	1,01066339	22,7542789
14	1,012140632	19,98534036
15	1,009818196	21,47955108
16	1,011112452	15,70108151
17	1,009996891	16,36264205
18	1,010510683	14,81447029
19	1,009780884	16,9663806
20	1,010989666	17,59120965
21	1,010130286	18,66983604
22	1,010694981	20,62041068
23	1,010118008	17,29742289
24	1,01085031	15,37522221
25	1,010011673	16,6949141
26	1,010227442	13,42302537
27	1,010499716	14,79084301
28	1,010210276	13,25774169
29	1,011722803	18,28726578
30	1,009929061	13,24690008
Ort	**1,010481723**	**17,81646981**

Genetik Algoritma (64)		
Tekrar	MSE	Süre
1	1,0096	4893,6000
2	1,0094	4507,8000
3	1,0095	4323,0000
4	1,0093	6784,8000
5	1,0095	5394,6000
6	1,0096	4830,0000
7	1,0094	3223,8000
8	1,0093	3912,6000
9	1,0093	6904,8000
10	1,0094	4143,0000
11	1,0096	4822,8000
12	1,0096	5521,8000
13	1,0096	4669,8000
14	1,0096	4461,6000
15	1,0096	4488,0000
16	1,0095	4705,8000
17	1,0095	3604,8000
18	1,0091	4467,0000
19	1,0096	5788,8000
20	1,0096	5775,0000
21	1,0095	3607,8000
22	1,0094	6673,8000
23	1,0096	10410,6000
24	1,0095	4185,0000
25	1,0095	3709,8000
26	1,0096	4437,0000
27	1,0095	5313,0000
28	1,0096	4222,8000
29	1,0095	7489,8000
30	1,0087	10404,0000
Ort	**1,0095**	**5255,9000**

Ek – 2 Gerçek Verilere Ait Sonuçlar

Huang Tek Katman (775)		
Tekrar	MSE	Süre
1	3,182452679	898,2016656
2	3,181602955	908,8281803
3	3,226835728	883,6889303
4	3,191046238	875,3050482
5	3,195027351	868,2659526
6	3,183797359	921,3718088
7	3,181941509	969,6646097
8	3,181970835	919,8885188
9	4,008738041	923,6674728
10	3,183134794	906,2193182
11	3,182017326	936,1636071
12	3,179840088	950,4032245
13	3,225164652	920,9180341
14	3,187620163	934,0962374
15	3,179370403	910,1065862
16	3,180964231	937,6170852
17	3,181392908	870,8812714
18	3,183238745	926,7669413
19	3,222468853	853,8811758
20	3,252556324	858,3025982
21	3,185450077	851,3846815
22	3,18073678	851,0632248
23	3,196378469	861,9611015
24	3,17903018	893,1382303
25	3,182270527	911,0850148
26	3,182179928	905,9062679
27	3,197505236	905,7195003
28	3,180079699	912,4914525
29	3,185247898	936,4141717
30	3,182009935	877,8659499
Ort	**3,218068997**	**902,7089287**

Huang İlk Gizli Katman (574) İkinci Katman (387)		
Tekrar	MSE	Süre
1	3,211458921	905,7949605
2	3,192945957	879,34026
3	3,183351278	883,574609
4	3,213840008	882,3265834
5	3,262513161	892,3826849
6	3,278428793	887,0942926
7	3,197621584	876,3372033
8	3,266379833	886,7560284
9	3,20540309	879,8880935
10	3,228445768	874,95961
11	3,209302187	885,9029484
12	3,275287867	887,9726353
13	3,187841892	851,7468808
14	3,182407379	832,1428034
15	3,245693207	845,7768943
16	3,230013132	842,0269339
17	3,266971588	844,6221371
18	3,211651802	840,7776816
19	3,193544626	838,1665826
20	3,326534748	841,5995259
21	3,207489729	837,7332933
22	3,204936504	841,4970818
23	3,200445414	838,2351978
24	3,207854986	837,5093057
25	3,239323139	844,0859041
26	3,186469316	838,9526908
27	3,217571497	841,6510971
28	3,220357895	842,0283766
29	3,221914053	848,4864867
30	3,253671885	845,9518669
Ort	**3,224322375**	**859,177355**

Heaton Tek Katman (604)		
Tekrar	MSE	Süre
1	3,203653336	856,3116629
2	3,179166079	866,1162772
3	3,181607962	845,7483916
4	3,182210445	845,1138785
5	3,181900978	841,2505665
6	3,184716463	846,0373771
7	3,182892561	840,74982
8	3,182575464	853,724757
9	3,256515265	866,3793075
10	3,18022418	848,113971
11	3,180253267	855,3002756
12	3,182031631	848,5912707
13	3,181668997	862,2928467
14	3,183376074	831,8266981
15	3,181297302	843,1038401
16	3,185084105	828,7903986
17	3,181453228	845,938477
18	3,18341589	859,0324302
19	3,18249011	871,9474897
20	3,181447983	835,8788235
21	3,179930687	890,6757653
22	3,180692673	898,7503901
23	3,187274694	918,7640417
24	3,183269024	872,0734124
25	3,180510044	843,4160752
26	3,191590548	835,1035964
27	3,185188293	851,5556271
28	3,180712461	828,5940821
29	3,18201828	839,9306409
30	3,181734562	858,8151267
Ort	**3,185696753**	**854,3309106**

Faktör Analizi (751)		
Tekrar	MSE	Süre
1	3,181943655	896,3887644
2	3,195803165	869,0999019
3	3,377578974	844,3406923
4	3,18125701	836,9307222
5	3,196771383	921,5056121
6	3,184253216	888,1269517
7	3,218111515	858,8600984
8	3,182342768	855,3602815
9	3,18195343	845,6821086
10	3,180901766	859,7712133
11	3,180480719	856,457062
12	3,182122946	864,0690238
13	3,18013072	860,5825739
14	3,261842489	848,5706487
15	3,1815207	874,8885529
16	3,182808876	855,9104762
17	3,183099508	882,9458663
18	3,181652069	848,3069727
19	3,184412718	882,6642699
20	3,199419975	865,6764307
21	3,18101716	880,039907
22	3,304522753	895,5670121
23	3,17923665	903,2369463
24	3,181632996	884,5667193
25	3,181634188	884,9738379
26	3,18138957	871,8581777
27	3,179522753	894,4119022
28	3,186053514	859,5959768
29	3,18117857	877,1786067
30	3,181583166	857,3247032
Ort	**3,197872631**	**870,8297338**

Genetik Algoritma (128)		
Tekrar	MSE	Süre
1	3,1779	79554,00
2	3,1778	84796,80
3	3,1784	87280,80
4	3,1775	77530,80
5	3,1785	78924,60
6	3,1771	82836,60
7	3,1787	86892,60
8	3,178	102232,80
9	3,1785	94227,00
10	3,1776	85393,80
11	3,1783	95041,80
12	3,1784	90006,00
13	3,1785	87153,00
14	3,1784	85794,00
15	3,1785	83809,80
16	3,1783	94366,80
17	3,1787	75417,00
18	3,1784	92811,60
19	3,1776	93391,80
20	3,1779	87076,80
21	3,178	89152,80
22	3,1779	73755,60
23	3,1783	88084,80
24	3,1782	86550,60
25	3,1785	74900,40
26	3,1781	87415,80
27	3,1782	82021,80
28	3,178	85819,80
29	3,1782	78330,00
30	3,1785	93267,60
Ort	**3,178163333**	**86127,92**

Ek – 3 Web Scrapping Kodları

```
import time
from bs4 import BeautifulSoup
import undetected_chromedriver as uc
from nordvpn_switcher import initialize_VPN, rotate_VPN
from csv import writer


def main():
    initialize_VPN(stored_settings=1)
    my_file = open("websites.txt", "r")
    data = my_file.read()
    websites = data.split("\n")

    for hata in range(3):
        try:
            my_file = open("websites.txt", "r")
            data = my_file.read()
            websites = data.split("\n")
            for website in websites:
                index = websites.index(website)
                if index % 10 == 0:
                    rotate_VPN()
                    driver = uc.Chrome()
                    driver.get(website)
                    time.sleep(3)
                    driver.refresh()
                else:
                    driver.get(website)
                try:
                    soup = BeautifulSoup(driver.page_source,
"html.parser")
                    links = soup.find_all(
                        "td", {"class": "searchResultsLargeThumbnail"}
                    )
                    for link in links:
                        site = link.find("a").get("href")
                        with open("links.txt", "a") as file:
                            content = site + "\n"
                            file.write(content)
                            file.close()
                except:
                    pass
                time.sleep(2)
                if index % 10 == 9:
                    driver.close()

                try:
                    with open("websites.txt", "r") as fr:
                        lines = fr.readlines()

                        with open("websites.txt", "w") as fw:
                            for line in lines:
                                if line.strip("\n") != website:
                                    fw.write(line)
                    print(
                        "Deleted "
```

```
                        + website
                        + " ("
                        + str(index + 1)
                        + "/"
                        + str(len(websites) + 1)
                        + ")"
                    )
                except:
                    print("Oops! Bir hata meydana geldi")

        except Exception as e:
            print(e)

    while True:
        try:
            my_file = open("links.txt", "r")
            data = my_file.read()
            links = data.split("\n")
            for link in links:
                index = links.index(link)
                house_link = "https://www.sahibinden.com" + link
                if index % 10 == 0:
                    rotate_VPN()
                    options = uc.ChromeOptions()
                    options.headless = True
                    driver = uc.Chrome(options=options)
                    driver.get(house_link)
                    time.sleep(3)
                    driver.refresh()
                else:
                    driver.get(house_link)
                try:
                    soup = BeautifulSoup(driver.page_source,
"html.parser")
                    try:
                        bot_check = soup.find("a", {"class": "error-page-
logo"})
                        if bot_check:
                            driver.close()
                            rotate_VPN()
                            options2 = uc.ChromeOptions()
                            options2.headless = True
                            driver = uc.Chrome(options=options2)
                            driver.get(house_link)
                            time.sleep(3)
                            driver.refresh()
                            soup = BeautifulSoup(driver.page_source,
"html.parser")
                        else:
                            pass
                    except:
                        pass

                    price = (
                        soup.find("div", {"class": "classifiedInfo"})
                        .find("h3")
                        .get_text()
                        .split("\n")[1]
                        .replace(" ", "")
                        .replace(".", "")
```

```
                        .replace("TL", "")
                    )
                    ilce = (
                        soup.find("div", {"class": "classifiedInfo"})
                        .find("h2")
                        .find_all("a")[1]
                        .get_text()
                        .replace(" ", "")
                        .replace("\n", "")
                    )
                    mahalle = (
                        soup.find("div", {"class": "classifiedInfo"})
                        .find("h2")
                        .find_all("a")[2]
                        .get_text()
                        .replace("  ", "")
                        .replace("\n", "")
                    )
                    brut = (
                        soup.find("ul", {"class": "classifiedInfoList"})
                        .find_all("span")[3]
                        .get_text()
                        .replace(" ", "")
                        .replace("\t", "")
                        .replace("\n", "")
                    )
                    net = (
                        soup.find("ul", {"class": "classifiedInfoList"})
                        .find_all("span")[4]
                        .get_text()
                        .replace(" ", "")
                        .replace("\t", "")
                        .replace("\n", "")
                    )
                    oda_sayisi = (
                        soup.find("ul", {"class": "classifiedInfoList"})
                        .find_all("span")[5]
                        .get_text()
                        .replace(" ", "")
                        .replace("\t", "")
                        .replace("\n", "")
                    )
                    bina_yasi = (
                        soup.find("ul", {"class": "classifiedInfoList"})
                        .find_all("span")[6]
                        .get_text()
                        .replace(" ", "")
                        .replace("\t", "")
                        .replace("\n", "")
                    )
                    bulundugu_kat = (
                        soup.find("ul", {"class": "classifiedInfoList"})
                        .find_all("span")[7]
                        .get_text()
                        .replace(" ", "")
                        .replace("\t", "")
                        .replace("\n", "")
                    )
                    kat_sayisi = (
                        soup.find("ul", {"class": "classifiedInfoList"})
```

```
                        .find_all("span")[8]
                        .get_text()
                        .replace("  ", "")
                        .replace("\t", "")
                        .replace("\n", "")
                    )
                    isitma = (
                        soup.find("ul", {"class": "classifiedInfoList"})
                        .find_all("span")[9]
                        .get_text()
                        .replace("\t", "")
                        .replace("  ", "")
                        .replace("\n", "")
                    )
                    banyo = (
                        soup.find("ul", {"class": "classifiedInfoList"})
                        .find_all("span")[10]
                        .get_text()
                        .replace("\t", "")
                        .replace("  ", "")
                        .replace("\n", "")
                    )
                    balkon = (
                        soup.find("ul", {"class": "classifiedInfoList"})
                        .find_all("span")[11]
                        .get_text()
                        .replace("\t", "")
                        .replace("  ", "")
                        .replace("\n", "")
                    )
                    esyali = (
                        soup.find("ul", {"class": "classifiedInfoList"})
                        .find_all("span")[12]
                        .get_text()
                        .replace("\t", "")
                        .replace("  ", "")
                        .replace("\n", "")
                    )
                    kullanim_durumu = (
                        soup.find("ul", {"class": "classifiedInfoList"})
                        .find_all("span")[13]
                        .get_text()
                        .replace("\t", "")
                        .replace("  ", "")
                        .replace("\n", "")
                    )
                    site = (
                        soup.find("ul", {"class": "classifiedInfoList"})
                        .find_all("span")[14]
                        .get_text()
                        .replace("\t", "")
                        .replace("  ", "")
                        .replace("\n", "")
                    )
                    aidat = (
                        soup.find("ul", {"class": "classifiedInfoList"})
                        .find_all("span")[16]
                        .get_text()
                        .replace("\t", "")
                        .replace("  ", "")
```

```
                        .replace("\n", "")
                    )
                    kredi = (
                        soup.find("ul", {"class": "classifiedInfoList"})
                        .find_all("span")[17]
                        .get_text()
                        .replace("\t", "")
                        .replace("  ", "")
                        .replace("\n", "")
                    )
                    tapu = (
                        soup.find("ul", {"class": "classifiedInfoList"})
                        .find_all("span")[18]
                        .get_text()
                        .replace("\t", "")
                        .replace("  ", "")
                        .replace("\n", "")
                    )
                    ev = [
                        price,
                        ilce,
                        mahalle,
                        brut,
                        net,
                        oda_sayisi,
                        bina_yasi,
                        bulundugu_kat,
                        kat_sayisi,
                        isitma,
                        banyo,
                        balkon,
                        esyali,
                        kullanim_durumu,
                        site,
                        aidat,
                        kredi,
                        tapu,
                    ]

                    with open(
                        "houses.csv", "a", newline="", encoding="utf-8"
                    ) as f_object:
                        writer_object = writer(f_object, delimiter=",")
                        writer_object.writerow(ev)
                        f_object.close()

                except:
                    pass
                time.sleep(2)
                if index % 10 == 9:
                    driver.close()

                try:
                    with open("links.txt", "r") as fr:
                        lines = fr.readlines()

                        with open("links.txt", "w") as fw:
                            for line in lines:
                                if line.strip("\n") != link:
                                    fw.write(line)
```

```
                    print(
                        "Deleted "
                        + link
                        + " ("
                        + str(index + 1)
                        + "/"
                        + str(len(links) + 1)
                        + ")"
                    )
                except:
                    print("Oops! Bir hata meydana geldi")

        except Exception as e:
            print(e)


if __name__ == "__main__":
    main()
```

Ek – 4 Derin Öğrenme ve Genetik Algoritma Kodları

```
import logging
from keras.models import Sequential
from keras.layers import Dense, Dropout
from keras.callbacks import EarlyStopping
import random
from functools import reduce
from operator import add
from tqdm import tqdm
from pandas import read_excel
from sklearn.preprocessing import StandardScaler
from sklearn.model_selection import train_test_split
from time import time
import csv

# Helper: Early stopping.
early_stopper = EarlyStopping(patience=5)


def get_data(dataset_path):
    batch_size = 64
    input_shape = (910,)

    data = read_excel(dataset_path)
    scaler = StandardScaler()
    scaled_data = scaler.fit_transform(data)
    res = train_test_split(
        scaled_data, train_size=0.630308380, test_size=0.157577093,
random_state=42
    )
    train_data, test_data = res
    x_train = train_data[:, 1:]
    y_train = train_data[:, 0:1]
    x_test = test_data[:, 1:]
    y_test = test_data[:, 0:1]

    return (batch_size, input_shape, x_train, x_test, y_train, y_test)


def compile_model(network, input_shape):
    """Compile a sequential model.
    Args:
        network (dict): the parameters of the network
    Returns:
        a compiled network.
    """
    # Get our network parameters.
    nb_layers = network["nb_layers"]
    nb_neurons = network["nb_neurons"]
    activation = network["activation"]
    optimizer = network["optimizer"]

    model = Sequential()

    # Add each layer.
    for i in range(nb_layers):
        # Need input shape for first layer.
        if i == 0:
```

```
            model.add(Dense(nb_neurons, activation=activation,
input_shape=input_shape))
        else:
            model.add(Dense(nb_neurons, activation=activation))

        model.add(Dropout(0.2))  # hard-coded dropout

    # Output layer.
    model.add(Dense(1, kernel_initializer="normal"))

    model.compile(loss="MSE", optimizer=optimizer, metrics=["MSE"])

    return model


def train_and_score(network, dataset_path):
    """Train the model, return test loss.
    Args:
        network (dict): the parameters of the network
        dataset (str): Dataset to use for training/evaluating
    """

    (
        batch_size,
        input_shape,
        x_train,
        x_test,
        y_train,
        y_test,
    ) = get_data(dataset_path)

    model = compile_model(network, input_shape)

    model.fit(
        x_train,
        y_train,
        batch_size=batch_size,
        epochs=10000,  # using early stopping, so no real limit
        verbose=0,
        validation_data=(x_test, y_test),
        callbacks=[early_stopper],
    )

    score = model.evaluate(x_test, y_test, verbose=0)

    return score[1]  # 1 is accuracy. 0 is loss.


class Network:
    """Represent a network and let us operate on it.
    Currently only works for an MLP.
    """

    def __init__(self, nn_param_choices=None):
        """Initialize our network.
        Args:
            nn_param_choices (dict): Parameters for the network,
includes:
                nb_neurons (list): [64, 128, 256]
```

```
                nb_layers (list): [1, 2, 3, 4]
                activation (list): ['relu', 'elu']
                optimizer (list): ['rmsprop', 'adam']
        """
        self.accuracy = 0.0
        self.nn_param_choices = nn_param_choices
        self.network = {}  # (dic): represents MLP network parameters

    def create_random(self):
        """Create a random network."""
        for key in self.nn_param_choices:
            self.network[key] = random.choice(self.nn_param_choices[key])

    def create_set(self, network):
        """Set network properties.
        Args:
            network (dict): The network parameters
        """
        self.network = network

    def train(self, dataset_path):
        """Train the network and record the accuracy.
        Args:
            dataset (str): Name of dataset to use.
        """
        if self.accuracy == 0.0:
            self.accuracy = train_and_score(self.network, dataset_path)

    def print_network(self):
        """Print out a network."""
        logging.info(self.network)
        logging.info("Network MSE: %.4f" % (self.accuracy))


class Optimizer:
    """Class that implements genetic algorithm for MLP optimization."""

    def __init__(
        self, nn_param_choices, retain=0.4, random_select=0.1,
mutate_chance=0.2
    ):
        """Create an optimizer.
        Args:
            nn_param_choices (dict): Possible network paremters
            retain (float): Percentage of population to retain after
                each generation
            random_select (float): Probability of a rejected network
                remaining in the population
            mutate_chance (float): Probability a network will be
                randomly mutated
        """
        self.mutate_chance = mutate_chance
        self.random_select = random_select
        self.retain = retain
        self.nn_param_choices = nn_param_choices

    def create_population(self, count):
        """Create a population of random networks.
        Args:
            count (int): Number of networks to generate, aka the
```

```
                size of the population
        Returns:
            (list): Population of network objects
        """
        pop = []
        for _ in range(0, count):
            # Create a random network.
            network = Network(self.nn_param_choices)
            network.create_random()

            # Add the network to our population.
            pop.append(network)

        return pop

    @staticmethod
    def fitness(network):
        """Return the accuracy, which is our fitness function."""
        return network.accuracy

    def grade(self, pop):
        """Find average fitness for a population.
        Args:
            pop (list): The population of networks
        Returns:
            (float): The average accuracy of the population
        """
        summed = reduce(add, (self.fitness(network) for network in pop))
        return summed / float((len(pop)))

    def breed(self, mother, father):
        """Make two children as parts of their parents.
        Args:
            mother (dict): Network parameters
            father (dict): Network parameters
        Returns:
            (list): Two network objects
        """
        children = []
        for _ in range(2):
            child = {}

            # Loop through the parameters and pick params for the kid.
            for param in self.nn_param_choices:
                child[param] = random.choice(
                    [mother.network[param], father.network[param]]
                )

            # Now create a network object.
            network = Network(self.nn_param_choices)
            network.create_set(child)

            # Randomly mutate some of the children.
            if self.mutate_chance > random.random():
                network = self.mutate(network)

            children.append(network)

        return children
```

```
    def mutate(self, network):
        """Randomly mutate one part of the network.
        Args:
            network (dict): The network parameters to mutate
        Returns:
            (Network): A randomly mutated network object
        """
        # Choose a random key.
        mutation = random.choice(list(self.nn_param_choices.keys()))

        # Mutate one of the params.
        network.network[mutation] =
random.choice(self.nn_param_choices[mutation])

        return network

    def evolve(self, pop):
        """Evolve a population of networks.
        Args:
            pop (list): A list of network parameters
        Returns:
            (list): The evolved population of networks
        """
        # Get scores for each network.
        graded = [(self.fitness(network), network) for network in pop]

        # Sort on the scores.
        graded = [x[1] for x in sorted(graded, key=lambda x: x[0],
reverse=False)]

        # Get the number we want to keep for the next gen.
        retain_length = int(len(graded) * self.retain)

        # The parents are every network we want to keep.
        parents = graded[:retain_length]

        # For those we aren't keeping, randomly keep some anyway.
        for individual in graded[retain_length:]:
            if self.random_select > random.random():
                parents.append(individual)

        # Now find out how many spots we have left to fill.
        parents_length = len(parents)
        desired_length = len(pop) - parents_length
        children = []

        # Add children, which are bred from two remaining networks.
        while len(children) < desired_length:
            # Get a random mom and dad.
            male = random.randint(0, parents_length - 1)
            female = random.randint(0, parents_length - 1)

            # Assuming they aren't the same network...
            if male != female:
                male = parents[male]
                female = parents[female]

                # Breed them.
                babies = self.breed(male, female)
```

```
                # Add the children one at a time.
                for baby in babies:
                    # Don't grow larger than desired length.
                    if len(children) < desired_length:
                        children.append(baby)

        parents.extend(children)

        return parents


# Setup logging.
logging.basicConfig(
    format="%(asctime)s - %(levelname)s - %(message)s",
    datefmt="%m/%d/%Y %I:%M:%S %p",
    level=logging.DEBUG,
    filename="log.txt",
)


def train_networks(networks, dataset_path):
    """Train each network.
    Args:
        networks (list): Current population of networks
        dataset (str): Dataset to use for training/evaluating
    """
    pbar = tqdm(total=len(networks))
    for network in networks:
        network.train(dataset_path)
        pbar.update(1)
    pbar.close()


def get_average_accuracy(networks):
    """Get the average accuracy for a group of networks.
    Args:
        networks (list): List of networks
    Returns:
        float: The average accuracy of a population of networks.
    """
    total_accuracy = 0
    for network in networks:
        total_accuracy += network.accuracy

    return total_accuracy / len(networks)


def generate(generations, population, nn_param_choices, dataset_path):
    """Generate a network with the genetic algorithm.
    Args:
        generations (int): Number of times to evole the population
        population (int): Number of networks in each generation
        nn_param_choices (dict): Parameter choices for networks
        dataset (str): Dataset to use for training/evaluating
    """
    optimizer = Optimizer(nn_param_choices)
    networks = optimizer.create_population(population)

    # Evolve the generation.
```

```
    for i in range(generations):
        logging.info("***Doing generation %d of %d***" % (i + 1,
generations))

        # Train and get accuracy for networks.
        train_networks(networks, dataset_path)

        # Get the average accuracy for this generation.
        average_accuracy = get_average_accuracy(networks)

        # Print out the average accuracy each generation.
        logging.info("Generation Average MSE: %.4f" % (average_accuracy))
        logging.info("-" * 80)

        # Evolve, except on the last iteration.
        if i != generations - 1:
            # Do the evolution.
            networks = optimizer.evolve(networks)

    # Sort our final population.
    networks = sorted(networks, key=lambda x: x.accuracy, reverse=False)

    # Print out the top 5 networks.
    print_networks(networks[:5])

    return networks[0].accuracy


def print_networks(networks):
    """Print a list of networks.
    Args:
        networks (list): The population of networks
    """
    logging.info("-" * 80)
    for network in networks:
        network.print_network()


def main():
    """Evolve a network."""
    generations = 10  # Number of times to evole the population.
    population = 20  # Number of networks in each generation.
    dataset_path = "houses.xlsx"

    nn_param_choices = {
        "nb_neurons": [64, 128, 256, 512, 768, 1024],
        "nb_layers": [1, 2, 3, 4],
        "activation": ["relu", "elu", "tanh", "sigmoid"],
        "optimizer": [
            "rmsprop",
            "adam",
            "sgd",
            "adagrad",
            "adadelta",
            "adamax",
            "nadam",
        ],
    }
```

```
    logging.info(
        "***Evolving %d generations with population %d***" %
(generations, population)
    )

    return generate(generations, population, nn_param_choices,
dataset_path)


if __name__ == "__main__":
    accuracies = []
    times = []
    for i in range(27):
        start_time = time()
        accuracy = main()
        finish_time = time()
        time_cost = finish_time - start_time
        accuracies.append(accuracy)
        times.append(time_cost)

    with open("genetic.csv", "wb") as f:
        writer = csv.writer(f)
        writer.writerows(zip(accuracies, times))
```