T.C.
YILDIZ TEKNİK ÜNİVERSİTESİ
FEN BİLİMLERİ ENSTİTÜSÜ

MAKİNE ÖĞRENMESİNİN SATRANÇ OYUNU ÜZERİNE UYGULAMASI

FATİH DOĞAN CELLAT

YÜKSEK LİSANS TEZİ
İstatistik Anabilim Dalı
İstatistik Programı

Danışman
Prof. Dr. Fatma NOYAN TEKELİ

Temmuz, 2022

T.C.

YILDIZ TEKNİK ÜNİVERSİTESİ

FEN BİLİMLERİ ENSTİTÜSÜ

MAKİNE ÖĞRENMESİNİN SATRANÇ OYUNU ÜZERİNE UYGULAMASI

Fatih Doğan CELLAT tarafından hazırlanan tez çalışması 06.07.2022 tarihinde aşağıdaki jüri tarafından Yıldız Teknik Üniversitesi Fen Bilimleri Enstitüsü İstatistik Anabilim Dalı, İstatistik Programı **YÜKSEK LİSANS TEZİ** olarak kabul edilmiştir.

Prof. Dr. Fatma NOYAN TEKELİ
Yıldız Teknik Üniversitesi
Danışman

Jüri Üyeleri

Prof. Dr. Fatma NOYAN TEKELİ, Danışman
Yıldız Teknik Üniversitesi _____

Prof. Dr. Gülhayat GÖLBAŞI ŞİMŞEK, Üye
Yıldız Teknik Üniversitesi _____

Prof. Dr. Hakan YILDIRIM, Üye
Marmara Üniversitesi _____

Danışmanım Prof. Dr. Fatma NOYAN TEKELİ sorumluluğunda tarafımca hazırlanan Derin Öğrenmenin Satranç Oyunu Üzerine Uygulaması başlıklı çalışmada veri toplama ve veri kullanımında gerekli yasal izinleri aldığımı, diğer kaynaklardan aldığım bilgileri ana metin ve referanslarda eksiksiz gösterdiğimi, araştırma verilerine ve sonuçlarına ilişkin çarpıtma ve/veya sahtecilik yapmadığımı, çalışmam süresince bilimsel araştırma ve etik ilkelerine uygun davrandığımı beyan ederim. Beyanımın aksinin ispatı halinde her türlü yasal sonucu kabul ederim.

Fatih Doğan CELLAT

İmza

Aileme

ve

Zuhal Fatma KARATAŞ'a

TEŞEKKÜR

Çalışma süresince yanımda olan tez hocam, Prof. Dr. Fatma NOYAN TEKELİ başta olmak üzere bana katkıda bulunan tüm hocalarıma teşekkürü borç bilirim. Süreç boyunca beni hiçbir zaman yalnız bırakmayan aileme ve Zuhal Fatma KARATAŞ'a tüm yardımları ve motivasyonu için en içten teşekkürlerimi sunarım.

Fatih Doğan CELLAT

İÇİNDEKİLER

KISALTMA LİSTESİ ..... **vii**
ŞEKİL LİSTESİ ..... **viii**
TABLO LİSTESİ ..... **ix**
ÖZET ..... **x**
ABSTRACT ..... **xii**
1 GİRİŞ **1**
1.1 Literatür Özeti 1
1.2 Tezin Amacı 4
1.3 Hipotez 5
2 KONUYLA İLGİLİ TEMEL KAVRAMLAR **6**
2.1 Makine Öğrenmesi Nedir? 6
2.2 Makine Öğrenimi Nasıl Çalışır? 6
2.3 Makine Öğrenimi Türleri 6
2.3.1 Denetimli Öğrenme 7
2.3.2 Denetimsiz Öğrenme 7
2.3.3 Takviyeli Öğrenme 7
2.4 Yaygın Olarak Kullanılan Makine Öğrenimi Algoritmaları 8
2.4.1 Regresyon 8
2.4.2 Rastgele Orman 9
2.4.3 Karar Ağaçları 9
2.4.4 Nöral Ağlar 10
2.5 Derin Öğrenme Nedir? 11
2.5.1 Yapay Sinir Ağı Nedir? 12
2.5.2 Çok Katmanlı Yapay Sinir Ağları 12
2.5.3 İleri Beslemeli Yapay Sinir Ağları 14
2.5.4 Geri Beslemeli Yapay Sinir Ağları 14
2.5.5 Aktivasyon Fonksiyonları 15
2.5.6 Optimizasyon Yöntemleri 18
2.6 Makine Öğrenmesi ve Derin Öğrenme Kullanım Alanları? 20
3 MATERYAL VE YÖNTEM **23**
3.1 Veri Setinin İncelenmesi 23
3.2 Keras ve Python 25
3.3 Derin Öğrenme Modeli Kurma 26

3.4 En Uygun Modelin Seçimi 26
3.5 Modelin Kurulması 27
4 BULGULAR VE TARTIŞMA 32
4.1 Verinin Görsel Analizleri Sonuçları 32
4.2 Sinir Ağı Modelleri için Sonuçlar 34
5 SONUÇ VE ÖNERİLER 38
KAYNAKÇA 40
TEZDEN ÜRETİLMİŞ YAYINLAR 43

KISALTMA LİSTESİ

AI	Artificial Intelligence (Yapay Zeka)
CNN	Convolutional neural network (Evrişimli Sinir Ağları)
DN	Doğru Negatif
DP	Doğru Pozitif
ECO	Encyclopedia of Chess Openings (Satranç Açılışları Ansiklopedisi)
GPU	Graphics Processing Unit(Grafik İşlem Birimi)
GUI	Graphical User Interface (Grafiksel Kullanıcı Arayüzü)
ML	Machine Learning (Makine Öğrenmesi)
MLP	Multilayer Perceptron(Çok Katmanlı Algılayıcı)
NHG	Nesterov Hızlandırılmış Gradyan
SGİ	Stokastik Gradyan İnişi
TÖ	Takviyeli Öğrenme
YN	Yanlış Negatif
YP	Yanlış Pozitif
YSA	Yapay Sinir Ağları
YZ	Yapay Zeka

ŞEKİL LİSTESİ

Şekil 1.1 Çalışmada kullanılan modeller 2
Şekil 2.1 Pavlovun köpeği koşullanma şeması 8
Şekil 2.2 Kredi vermeye yönelik örnek karar ağacı modeli 10
Şekil 2.3 Örnek derin öğrenme modeli 11
Şekil 2.4 Biyolojik sinir hücresi ve yapay sinir hücresinin karşılaştırılması 12
Şekil 2.5 Çok katmanlı ağ modeli 13
Şekil 2.6 Aktivasyon fonksiyonları çalışma şekli 15
Şekil 2.7 En çok kullanılan aktivasyon fonksiyonları 18
Şekil 2.8 Makine öğrenimi ve derin öğrenme kullanım alanları 20
Şekil 3.1 Veri setinin ilk 10 verisi 25
Şekil 3.2 Dönüşümden geçen değişkenlerin şeması 29
Şekil 3.3 Model 1 için keras kodları 30
Şekil 3.4 Model 1 şeması 30
Şekil 3.5 Model 2 kodları 30
Şekil 3.6 Model 2 şeması 31
Şekil 4.1 Kazanma oranları ve sayıları 32
Şekil 4.2 En çok kullanılan açılışlar 33
Şekil 4.3 En çok oynanan 3 açılışın tahtada gösterimi 33
Şekil 4.4 Model 1 için eğitim ve test verisindeki doğruluk oranları grafiği 34
Şekil 4.5 Model 2 için eğitim ve test verisindeki doğruluk oranları grafiği 34
Şekil 4.6 Örnek maç için modelin öngörüsü 35
Şekil 4.7 Oynanan maçın gerçekte sonucu 1 beyazı temsil etmekte 35
Şekil 4.8 Örnek maçın açılışı tahta üzerinde gösterimi 35
Şekil 4.9 Başka bir örnek maç için modelin tahmini 36
Şekil 4.10 Eşit puanlı iki oyuncu için modelin tahmini 36
Şekil 4.11 Model 2 için karışıklık matrisi 37

TABLO LİSTESİ

Tablo 3.1 Veri seti ve açıklamaları 24
Tablo 3.2 Karışıklık matrisi 27

ÖZET

Makine Öğrenmesinin Satranç Oyunu Üzerine Uygulaması

Fatih Doğan CELLAT

İstatistik Anabilim Dalı

Yüksek Lisans Tezi

Danışman: Prof. Dr. Fatma NOYAN TEKELİ

Satranç çağlar boyunca pek çok kişi tarafından oynanmış bir oyundur. İmparatorların satranç oynarken birçok taktik geliştirdikleri ve savaşlarda kullandıklarına inanılmaktadır. İlk derin öğrenme tekniklerinin satranç üzerinde geliştirilmesi nedeniyle makine öğrenmesi ve derin öğrenme için de önemli bir yere sahiptir.

Makine öğreniminin bir alanı olan derin öğrenme, aynı zamanda insan beyni üzerinde modellenen yapay sinir ağlarının devrim niteliğinde bir gelişimidir. Yapay zeka, grafik modelleme, optimizasyon, örüntü tanıma ve sinyal işlemenin kesiştiği noktada ortaya çıkmıştır.

Bu çalışmada öncelikle satrançtan ve tarihinden bahsedilmiştir. Daha sonra yapay zeka ve makine öğrenmesinden sonrasında makine öğrenmesinin bir alt dalı olan derin öğrenme kavramları tanımlanmıştır.

Uygulama bölümünde Lichess sitesinden alınan verilerle geliştirilen modelde, satranç maçının sonucu derin öğrenme ve nöral ağlar kullanılarak tahminleyen bir uygulama geliştirilmiştir.

Python açık kaynak dilinin kullanıldığı modelde derin öğrenme kütüphaneleri kullanılmış, hazırlık sürecinde beraberlik ile biten oyunlar atılmış, veriler birçok değişken azaltılarak ölçeklendirilmiş ve veriler modele uygun hale getirilmiştir.

Oluşturulan model oyuncuların açılışlarını, puanlarını, maçın süresi vb. verileri giriş olarak kullanarak maç sonucunu tahminlemektedir.

Çalışma üstüne yapılan veri analizleri sonucunda bu veri seti için beyaz renk ile oynayan oyuncuların yaklaşık %52 oranda ve siyah renk ile oynayan oyuncuları %48 oranda kazandığı görülmüştür. Yine aynı veri için en çok oynanılan ilk 3 açılışın sırasıyla Sicilya Defansı, Fransız Defansı ve Vezir'in Piyon Oyunu olduğu görülmüştür.

Çalışmada verinin %15'i test %85'i eğitim verisi olarak kullanılmıştır. Model test verisini %65 oranda doğru eğitim verisini ise %72 oranda doğru tahminlemiştir.

Anahtar Kelimeler: Derin Öğrenme, makine öğrenmesi, satranç

ABSTRACT

Machine Learning Application On Chess

Fatih Doğan CELLAT

Department of Statistics

Master of Science Thesis

Supervisor: Prof. Dr. Fatma NOYAN TEKELİ

Chess is a game which played by many,down the ages.it is believed that emporers strategized many tactics while playing chess and they used them in wars.also has an important place for machine learning and deep learning because the first deep learning techniques were developed with chess.

Deep learning, which is a field of machine learning, also a revolutionary development of artificial neural networks modeled on the human brain. Artificial intelligence emerged at the intersection of graphics modeling, optimization, pattern recognition and signal processing.

In this study, first of all, chess and its history are mentioned. then, after artificial intelligence and machine learning, deep learning concepts, which is a sub-branch of ML, were defined.In the model developed with the data which is taken from the Lichess site, it is aimed to predict winner of the match.

The model created is based on the openings of the players, their ratings, the duration of the match, etc. It predicts the match result using the data as input.

In the model in which the Python open source language was used, deep learning libraries were used, games that ended with a draw were discarded during the preparation process, the data was scaled by reducing many variables and the data was manipulated in accordance with the model.

As a result of the data analysis made on the study, it was seen that for this data set, the players playing with the color white won approximately 51% and the players playing with the black color 49%. Again for the same data, it was seen that the top 3 most played openings were Sicilian Defense, French Defense and Queen's Pawn Game, respectively.

In the study, 15% of the data was used as test data and 85% as training data. While the model predicted the test data 65% correctly, and it predicted the training data 72% correctly.

Keywords: Deep learning, machine learning, chess

1
GİRİŞ

1.1 Literatür Özeti

Satranç uzun ve karmaşık bir tarihe sahiptir. Oyun, Hindistan'daki en eski biçimlerinden biraz değişmiştir. Saat yoktu ve parçalar 19. yüzyıla kadar standardize edilmemişti.

Satranç, bugün bildiğimiz gibi, MS 600'lerden önce Hint oyunu ‘Chaturanga' isimli oyundan doğmuştur. Oyun, sonraki yüzyıllarda Asya ve Avrupa'ya yayılmış ve sonunda 16. yüzyılda satranç olarak bildiğimiz haline dönüşmüştür. Oyunun ilk ustalarından biri, Ruy Lopez adında bir İspanyol rahiptir. Kendi adını taşıyan açılışı 1561'de yayınladığı bir kitapta analiz etmiştir.

Resmi dünya şampiyonası unvanı 19. yüzyılın sonlarında, ilk büyük turnuvaların düzenlenmesinden ve birden fazla oyun stilinin tamamen gelişmeye başlamasından kısa bir süre sonra ortaya çıkmıştır.. Açılışlarla ilgili ilk kitap 1843 gibi erken bir tarihte yayınlanmış olsa da, bildiğimiz kadarıyla teori 20. yüzyılın başlarına/ortalarına kadar gerçekten gelişmedi. Bilgisayar motorları ve veri tabanları 20. yüzyılın sonlarına kadar devreye girmedi.

Yapay Zeka (YZ) ve satrancın ilişkisi 1900lü yılların başına dayansa da en donanımlı makine ‘Deep Blue’ adında IBM’in yapmış olduğu derin öğrenme makinesidir.1996 yılında o dönemin dünya şampiyonu Garry Kasparov’a karşı medyanın önünde maça çıkmış ve kaybetmiştir.

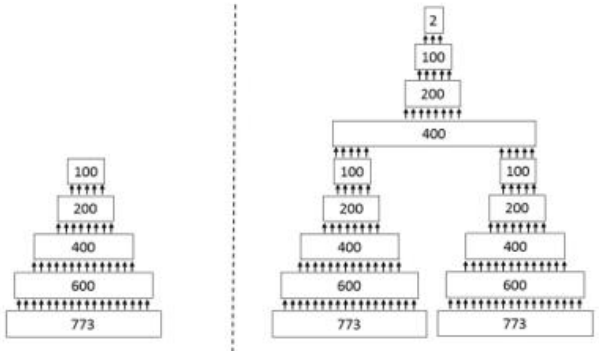
Bir sonraki yıl IBM milyon dolar bütçeler ayırarak ‘Deeper Blue’ adlı yapay zekayı geliştirmiş ve Kasparova karşı 1997 yılında bir dünya şampiyonunu yenen ilk makineyi ortaya koymuştur. Bu, YZ alanında gelişmeler için bir milat oluşturup sonrasında satranç üzerine derin öğrenme uygulamaları birçok kez farklı veri seti ve farklı yöntemler kullanılarak yapılmıştır.

1994 yılında Thrun satranç maç sonucuna göre eğitilen bir yapay zeka modeli geliştirmiş ve adına NeuroChess adını vermiştir. EBNN yöntemiyle modeli geliştirmiştir. Model eğitim sonrası o dönemin başka güçlü yapay zekası olan GNU Chess'e karşı oynadığı maçların %13ünü kazanma sonucunu elde etmiştir [1].

Günümüze doğru gelindiğinde satranç ve yapay zekayı insan davranışları için inceleyen, satrançta kişiselleştirilmiş insan davranışlarını konu edindiği çalışmada McIlroy-Young, Wang, Sen, Klenberg ve Andersonun 2008 yılında 400 farklı oyuncunun oyunlarından eğittikleri model ile oyuncuların davranışlarını ve hamlelerini tahminlemişlerdir. Oluşturulan model ilk hamleleri %90'a yakın bir oranla tahminlerken ilk 10 hamle sonrası %40 ve 30 hamle sonrası için %10lara düşen bir doğruluk orana sahiptir [2].

Başka bir çalışmadı Oshri ve Khandwala'nın 2015 yılında yayınladıkları uygulamada satranç veri seti kullanılarak CNN metodu ile bir sonraki hamlenin hangi taş ile yapılacağı tahminlenmiştir. Çalışma sonucunda yapılan en iyi model %38.3'lük doğruluk oranına sahip olup taş bazında bakıldığında piyon, kale, at, fil, şah veya vezirden birinin bir sonraki hamlede kullanılmasının seçilmesi adına kurulan modelin doğruluk oranları %52.20, %29.25, %56.15, %40,54, %26,52 ve %47,29 performans göstermiştir [3].

David Netanyahu ve Wolf'un çalışmasında ise önce denetimli sonra denetimsiz öğrenme yöntemi ile modellenen DeepChess, satranç ustaları kadar iyi oynayacak seviyeye gelmiştir.



Şekil 1.1 Çalışmada kullanılan modeller

Yaptıkları çalışmada Şekil 1.1'de görülen ilk modeli öncelikle kurmuş daha sonrasında katman sayısı arttırılarak ikinci model kurulmuştur [4].

Şekercioğlu çalışmasında minimaks algoritmasını kullanarak bir satranç oyunu yazmıştır. Oluşturulan programda kullanıcı beyaz taş ile bilgisayar ise siyah taşlarla oynamakta olup taşların türüne göre ve konumlarına göre değerlendirme sistemi oluşturmuştur. Oyun teorisinin kullanıldığı model Java ile yazılmıştır kurallar ve puanlar çerçevesinde bilgisayar oynanacak en iyi hamleyi seçmektedir [5].

Erilli ise bulanık kümeleme analizi ile satranç oyuncularının sınıflandırdığı bir çalışma hayata geçirmiştir. Oyuncular maç skorlarına göre veya ECO (Encyclopedia of Chess Openings) kodlarına göre sınıflandırılmıştır. Usta oyuncular Pozisyonel[1], Defansif veya Saldırgan olarak sınıflandırılmıştır. Oyuncuların oyun tarzları konusunda bilgi vermektedir [6].

Satranç oynayan bir başka çalışma ise Lai tarafından 2015 yılında yapılmıştır. Uygulamada takviyeli derin öğrenme sistemi kullanan Lainin modelinin adı "Giraffe" olup, modelin tahmin ettiği ilk 3 hamle içerisinde %70 oranla en iyi hamle bulunmuştur. Giraffe oyuncuların %2.2sinin erişebildiği 'FIDE International Master' seviyesine gelebilmiştir [7].

Ülkemizde satranç alanında yapılan çok sayıda çalışma olmayıp bunlardan biri Şişecinin çalışması olup oyun içi desenleri optimizasyon yöntemleri kullanarak çözebilen bir model tasarlamıştır. Sezgisel algoritmaların gücünden yararlanarak sonsuza yakın sayıda olasılık içeren tüm uzayı taramak yerine umut vaat eden bölgelere odaklanıp en iyi sonucu aramalarıyla sağlamaktadır. Model hem insanlara hem de satranç oynayabilen diğer makinelere karşı çalışabilmektedir [8].

Satranç oynayan makinelerin dışında satranç müsabakalarında destek olma amaçlı bilgisayarla görü tabanlı satranç oyunları izleme sistemi oluşturan Koray hamle algılamak için anlık görüntüleri tek tek işleyerek satranç taşlarının konumunun değişikliğini ve renk değişikliğini kullanarak oyuncuların yaptığı hamleyi tespit etmiştir yapılan hamlenin kurallara uygun olup olmadığı kontrol edilip kullanıcıya gerekli uyarının verildiği bir çalışma literatüre katmıştır. Sistem

[1] Pozisyonel: Saldırgan veya defansif olmayan, yerini korumaya yönelik davranışta bulunan kişi veya kişiler için söylenmiştir.

farklı ışık koşulları altında test edilmiş ve başarılı bir sonuca ulaştığı fark edilmiştir [9].

Türk ise satranç oyununda bu çalışmada da görüldüğü üzere beyaz oyuncunun kazanma olasılığının daha fazla olduğu sonucundan yola çıkarak eş zamanlı oynanan kural setinin değiştiği bir satranç oyun motoru ortaya koymuştur. Çalışma GUI tabanlı web uygulamasına dönüştürülmüş olup eş zamanlı oynanan yeni bir satranç oyunu prototipi literatüre dahil edilmiştir [10].

Satranç ve yapay sinir ağlarının birleştiği bir diğer çalışmada Sabatelli,Bidoia,Codreanu ve Wiering 3 milyon satranç pozisyonundan desen ile öğrenen bir derin öğrenme satranç uygulaması ortaya koymuşlardır. Çalışma sonuçlarına bakıldığında kendi modelleri için MLP sistemlerinin CNN sistemlerine oranla tüm yönlerden daha başarılı olduğu sonucuna varmışlardır [11].

Jankuloski ve Božinovski'nin çalışmasında ise satrancın yapay zeka ile ilişkisini satranç oynayan algoritmaların nasıl çalıştığını anlatmışlardır. Derin öğrenme mimarisini ve sinir ağlarının uygulanma tarzını açıklayan çalışmalarını hem gündelik hem de rekabetçi ortamda satranç oynayan makinelerin geleceği hakkında bilgi vermişlerdir [12].

Maharaj Polson ve Turk ise oyun sonu bulmacalarını çözmek üzere Stockfish'i ve Leela Chess Zero'yu karşılaştırmışlardır. Derin öğrenme modelleri ile eğitilen modellerden, çalışma sonucu olarak Stockfish'in daha başarılı olduğunu gözlemlemişlerdir [13].

1.2 Tezin Amacı

Satranç, insanlar tarafından bulunan basit kuralları olan 64 karelik bir tahta oyunudur. Bir satranç oyunu için olası hamle olasılıklarının sayısı, evrendeki yıldızların sayısından ve hatta evrendeki toplam atom sayısından daha fazladır. (Shannon sayısı 10^{120}).

Satrançta sadece ilk dört hamle 318.879.564.000 potansiyel olasılığa sahiptir, bu da herhangi bir kişinin tüm bu olasılıkları hesaplamasını neredeyse imkansız hale getirir. Makine öğrenimi, analitik modellerin oluşturulmasını otomatikleştiren bir tür veri analizidir. Bilgisayarların verilerden öğrenebileceği, kalıpları

tanıyabileceği ve çok az insan etkileşimi ile yargılarda bulunabileceği öncülüne dayanan bir yapay zeka alt kümesidir [14].

Makine öğrenimi iki kategoriye ayrılır: denetimli öğrenme ve denetimsiz öğrenme. Denetimli öğrenme, makine öğreniminin çoğunu oluşturur. Denetimli öğrenmede girdi değişkenleri ve hedef değişkenleri vardır. Eşleştirme işlevi bir algoritma kullanılarak öğrenilir. Veriler arasındaki ilişki öğrenilmeye çalışılır. Denetimsiz öğrenmede yalnızca girdi verileri kullanılır hedef değişkenleri kullanılmaz [14].

Derin öğrenmenin yaygın olarak benimsenmesinin en temel nedeni, sorunları yüksek doğruluk oranlarıyla çözmesidir. Derin öğrenme ilk olarak 2012 yılında bilim adamları tarafından kullanılmıştır. Derin öğrenme tekniklerinde yapay sinir ağları ve enerji tabanlı modeller kullanılmaktadır. Çok sayıda seviye ve gizli değişkenler içeren karmaşık bir mimariye sahiptir [15].

Bu çalışmada amaç makine öğrenmesi alt dalı olan derin ağlar kullanılarak, bir satranç maçının başında maç sonucunu tahmin etmektir. Çalışmada 20 bin veri seti ile eğitilen model, oyuncuların puanı ve açılışları gibi değişkenleri girdi olarak kullanarak maçın kazananının kim olduğunu ve hangi oranla kazanacağını çıktı olarak verecektir.

1.3 Hipotez

Satranç üzerine yapılan çalışmaların birçoğu satranç oynayabilen ve farklı algoritmalar ile beslenen derin öğrenme modelleri veya makine öğrenmesi modellerinden oluşmaktadır. Bir satranç maçı için kazananın tahmin edildiği çalışmalar literatürde sınırlıdır. Oyunun açılış hamlesi ve oyuncuların önceki maçlardan kazandığı puanlar oyunun süresi ve açılış şekilleri göz önünde bulundurularak bir maçın kazanan oyuncusunu tahmin edilip edilemeyeceği hipotezi ile çalışma yapılmıştır.

2
KONUYLA İLGİLİ TEMEL KAVRAMLAR

2.1 Makine Öğrenmesi Nedir?

Makine öğrenimi, insanların nasıl öğrendiklerini simüle etmek ve zaman içinde doğruluklarını geliştirmek için veri ve algoritmaları kullanan yapay zeka (AI) ve bilgisayar biliminin bir parçasıdır. [16].

2.2 Makine Öğrenimi Nasıl Çalışır?

Bir Karar Süreci: Makine öğrenimi algoritmaları, verileri tahmin etmek veya sınıflandırmak için yaygın olarak kullanılır. Algoritmanız, etiketlenmiş veya etiketlenmemiş belirli girdi verilerine dayalı olarak verilerdeki bir kalıp hakkında bir tahmin yapacaktır.

Bir Hata Fonksiyonu: Modelin tahminini değerlendirmek için bir hata fonksiyonu kullanılır. Bilinen örnekler sağlanırsa, modelin doğruluğunu belirlemek için bir hata işlevi bunları karşılaştırabilir.

Model Optimizasyon Süreci: Model, eğitim setinde daha iyi sonuçlar veriyorsa ağırlıklar, bilinen örnek ile model tahmini arasındaki farkı azaltmak için değiştirilir. Bu değerlendirme ve optimizasyon prosedürü, bir doğruluk eşiğine ulaşılana kadar ağırlıkları bağımsız olarak güncelleyecek olan algoritma tarafından tekrarlanacaktır [17].

2.3 Makine Öğrenimi Türleri

Makine öğrenimi öğrenim tipine göre veya hedef değişkenin varlığına göre farklı kategorilere ayrılır.

- Denetimli Öğrenme
- Denetimsiz Öğrenme
- Takviyeli Öğrenme

2.3.1 Denetimli Öğrenme

Denetimli öğrenme hedef değişkenin olduğu ve verinin bu şekilde beslenip tahminleme yaptığı öğrenme sistemleri tekniğidir. Eğitim verisinde kullanılan ve sonuçlanan olayların geleceğe yönelik tahminlerinde daha çok kullanılır. Birçok sektörde farklı amaçlarla kullanılmaktadır, daha çok dolandırıcılık tahminleri, tanı teşhis koyma, görüntü sınıflandırma sistemlerinde kullanılır.

2.3.2 Denetimsiz Öğrenme

Denetimsiz öğrenme hedef değişkenin olmadığı öğrenme sistemleri algoritmalarına verilen genel addır. Bu sistemin eğitilmediği anlamına gelir.

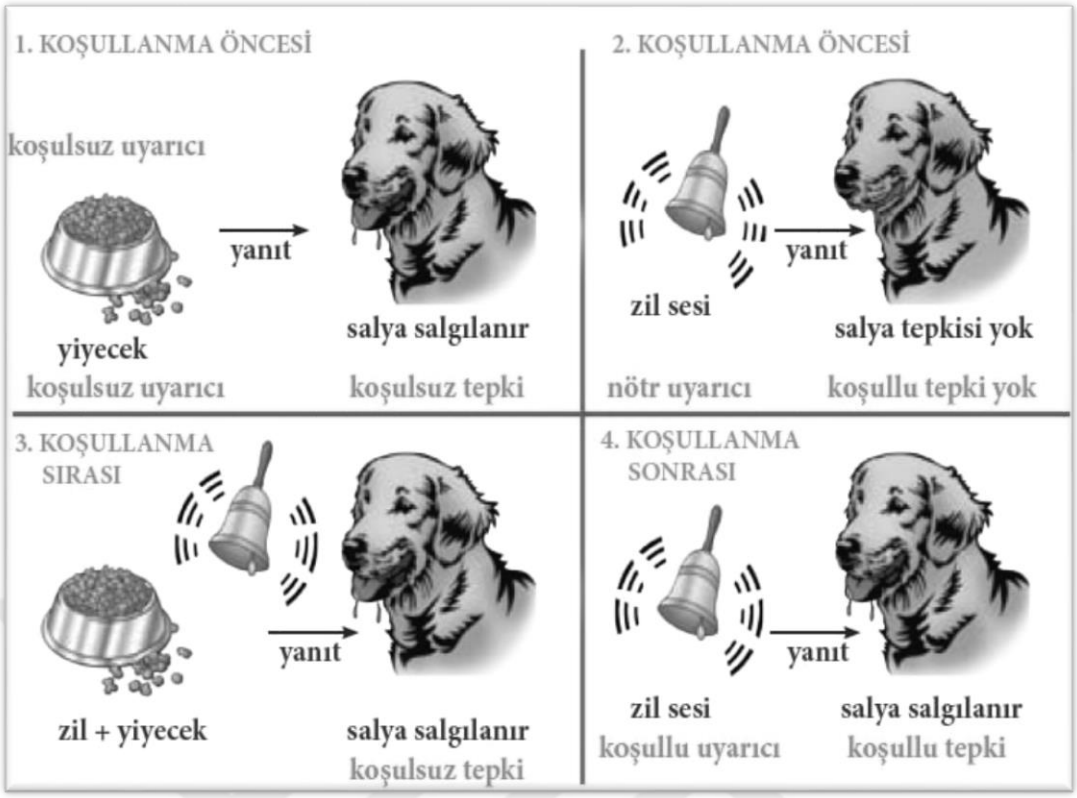
Denetimsiz öğrenme, denetimli öğrenmeye kıyasla daha karmaşık algoritmalar üzerinde çalışır çünkü veriler hakkında çok az bilgi vardır veya hiç yoktur.

Sonuç üretmeyi amaçlayan bir makine veya model üzerinde değişiklik yapma konusunda daha az alan bırakır. Kümeleme algoritmaları denetimsiz öğrenmenin temelini oluşturur. Segmentasyon çalışmalarının büyük çoğunluğu denetimsiz öğrenme tipi çalışmalardır.

2.3.3 Takviyeli Öğrenme

Takviyeli öğrenme (TÖ), deneme yanılma üzerine kurulu bir makine öğrenmesi tekniğidir. TÖ, Markov Karar Süreçleri stratejini kullanır. Markov Karar Süreçleri aksiyonların sadece bir sonraki ödülü değil gelecek durumları da etkilediği sıralı karar verme sürecini modelleyen klasik bir formüldür. Takviyeli öğrenmede ajan olarak adlandırılan eylemleri yerine getiren birim karşılaştığı durumlara göre bir aksiyon alır. Alınan her bir aksiyonun karşılığında da sayısal bir ödül alır. Ajan farklı farklı aksiyonları deneyerek aldığı bu ödül puanını maksimuma çıkartmak için çalışır [18].

Genel olarak takviyeli öğrenme kümülatif ödülü en yüksek ve cezayı minimum yapmaya çalışan makine öğrenimi türüdür. Pavlov'un köpeği verilebilecek en iyi örneklerden biridir.



Şekil 2.1 Pavlovun köpeği koşullanma şeması

Şekil 2.1'de görüldüğü üzere Pavlov köpeğine zilsiz verdiği besin ile salya miktarını ölçmüş, daha sonra zil ile birlikte besin vermiştir. Koşullanan köpeğin besinsiz çalınan zile salya akıttığını görmüştür [19].

2.4 Yaygın Olarak Kullanılan Makine Öğrenimi Algoritmaları

Verileri analiz etmek için makine öğrenimi algoritmalarını kullanmak, makine öğreniminin amacıdır. Bir geliştirici, manuel insan girdisine ihtiyaç duymadan makine öğrenimini kullanarak çok fazla veri içeren bir görevin etkinliğini artırabilir.

Veri bilimi, bilgisayar bilimi ve diğer birçok alandaki uzmanların çıktısını artırmak için küresel olarak güçlü makine öğrenimi teknikleri uygulanabilir [17].

2.4.1 Regresyon

Girdi değişkeni ile çıktı değişkeni arasında bir ilişki varsa, regresyon algoritmaları kullanılır. Sürekli değişkenlerin analizinde kullanılabilir hava tahmini, piyasa Trendleri vb. gibi için kullanılır. Bazı regresyon algoritmaları aşağıdaki gibidir.

Çalışmamızda bu regresyon algoritmaları kullanılmadığı için algoritmalar hakkında detaylı bilgi verilmeyecektir.

- Doğrusal Regresyon
- Regresyon Ağaçları
- Doğrusal Olmayan Regresyon
- Bayes Doğrusal Regresyon
- Polinom Regresyon [20]

2.4.2 Rastgele Orman

Denetimli bir makine öğrenme algoritması olan Rastgele Orman, hem regresyon hem de sınıflandırma için uygulanır. Yine de esas olarak sınıflandırma problemleri için kullanılır. Rastgele Orman algoritması, veri seri üzerinde karar ağaçları oluşturur ve daha sonra her biri için tahmin alır ve sonunda en iyi çözümü seçer.

Sonuçların ortalamasını alarak fazla uydurmayı azalttığı için tek bir karar ağacından daha iyi olan bir topluluk yöntemidir.

Rastgele Orman algoritmasının çalışmasını aşağıdaki adımlardaki gibi olmaktadır.

Adım 1 – Bir veri kümesinden rassal örnekler seçilir.

Adım 2 – Algoritma, her örnek için bir karar ağacı oluşturacaktır. Daha sonra her karar ağacından tahmin sonucunu alacaktır.

Adım 3 – Tahmin edilen her sonuç için puanlama yapılacaktır.

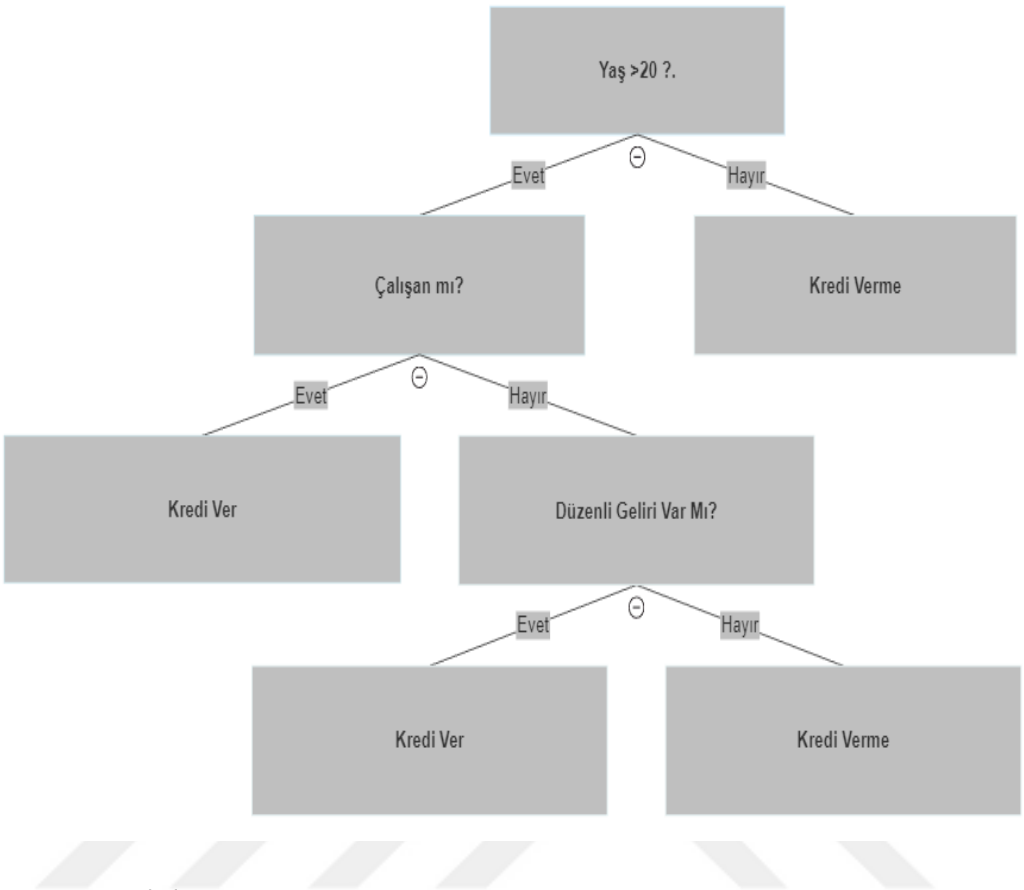
Adım 4 – En çok puanlanan tahmin, nihai tahmin sonucu olarak seçilir [20].

2.4.3 Karar Ağaçları

Genel olarak, Karar ağacı analizi, birçok alanda uygulanabilen tahmine dayalı bir modelleme aracıdır. Karar ağaçları, veri setini farklı koşullara göre farklı şekillerde bölebilen algoritmik bir yaklaşımla oluşturulabilir.

Kararlar, denetimli algoritmalar kategorisine giren en güçlü algoritmalardır hem sınıflandırma hem de regresyon görevleri için kullanılabilirler [21].

Şekil 2.2'de kredi vermek üzerine hazırlanan bir karar ağacı şeması görülmektedir.



Şekil 2.2 Kredi vermeye yönelik örnek karar ağacı modeli

2.4.4 Nöral Ağlar

Sinir ağları olarak bilinen yapay zeka sistemleri, insan beyninin verileri nasıl yorumladığını ve akıllıca kategorize ettiğini taklit etmeyi amaçlar. Binlerce, hatta milyonlarca birbirine bağlı işlem düğümü aracılığıyla, bu sinir ağı öğrenme algoritmaları, veri ve sesteki kalıpları belirlemek, dilleri çevirmek, finansal tahminler yapmak ve çok daha fazlası için kullanılır. Bir sonraki düğüm katmanına aktarılmadan önce veriler, ağırlıkları analiz eden ve uygulayan katmanlar aracılığıyla "ileriye beslenir".

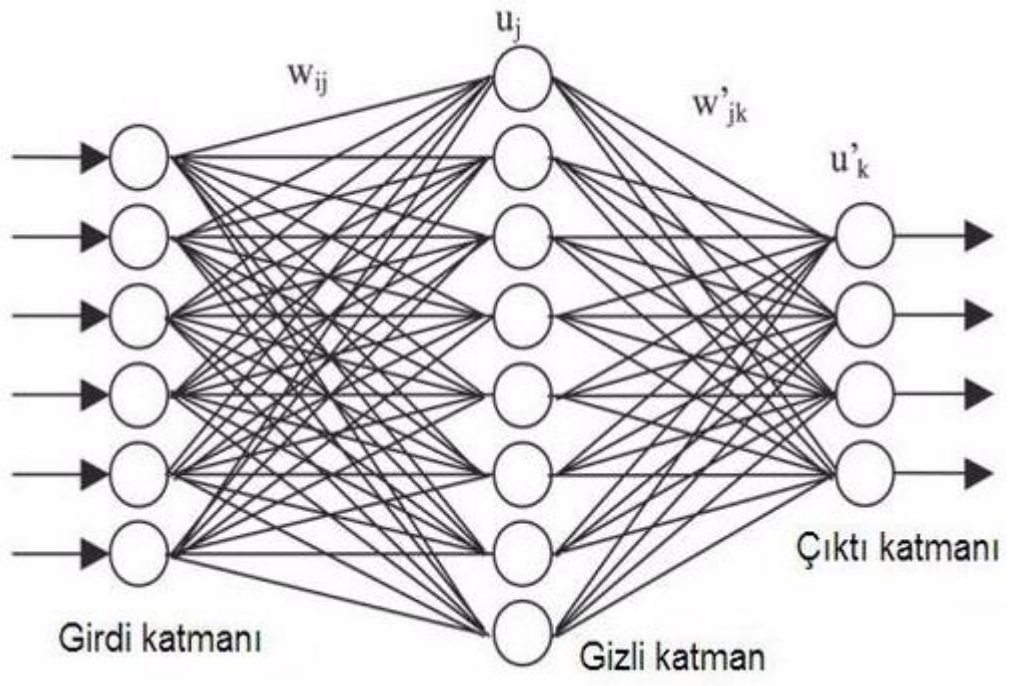
Ağ algoritmalarının doğruluğunu ve etkinliğini artırmak için, girdi eğitim verilerinden hızlı bir şekilde öğrenmek için sinir ağı algoritmaları kritik bir şekilde oluşturulur. Bu nedenle sinir ağları, makine öğrenimi modellerinin gücünün ve vaadinin önemli örnekleridir [17].

2.5 Derin Öğrenme Nedir?

2010'lu yıllarda kullanılmaya başlanan derin öğrenme, makine öğrenmesinde kullanılan hesaplamaları büyük veri ile tek bir katman yerine birden çok katmanda tek seferde yapan, makine öğrenmesinde tanımlanması gereken parametreleri keşfeden, daha iyi parametrelerle değerlendirmeler yapabilen algoritmalar bütünüdür [14].

Derin öğrenme aslında birçok makine öğrenmesi algoritmasını yapay hücrelerinde bulundurarak işler ve çıktılar üretir. Bu yönüyle derin öğrenmenin, makine öğrenmesi algoritmasının bir alt dalı olduğu söylenmektedir.

Yapay sinir ağlarında yeni sayılabilecek gelişmelerden biri de derin öğrenmedir Bir yapay sinir ağı iki veya daha fazla katmandan oluşuyorsa derindir. [22]. Şekil 2.3'te derin öğrenme mimarisi gösterilmiştir [15].

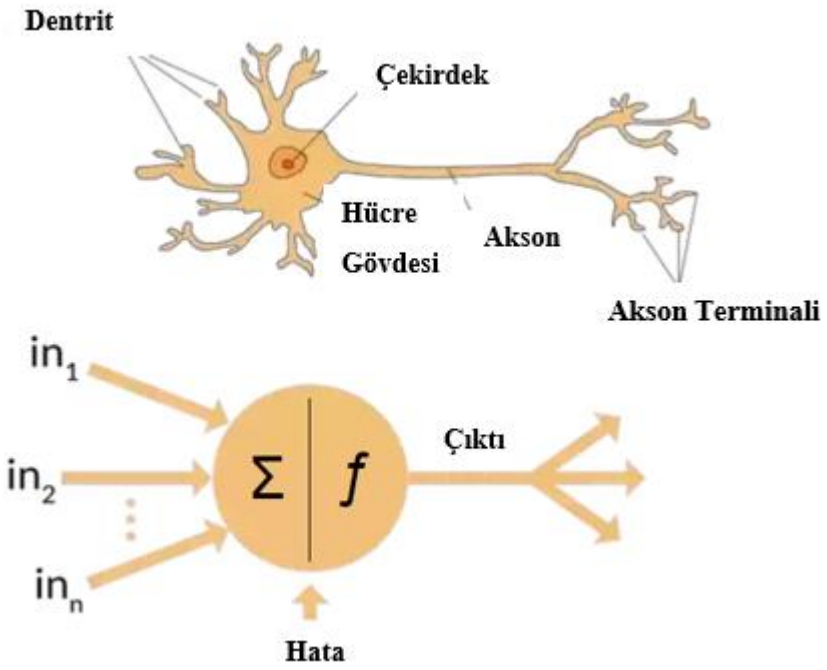


Şekil 2.3 Örnek derin öğrenme modeli

Şekil 2.3'te derin öğrenme modelinin temelde sinir ağı katmanlarından oluştuğu görülmektedir, yapay sinir ağına birçok gizli katman eklenmiş ve bu katmanlar yardımıyla çıkış katmanında sonucun bulunması sağlanmıştır [14]. Derin öğrenme çok sayıda parametreye sahiptir çünkü çok fazla gizli katman vardır. Her katman, arkasına yerleştirilmiş birden çok öğretilebilir katmandan oluşur. [15].

2.5.1 Yapay Sinir Ağı Nedir?

Yapay Sinir Ağı (YSA), bilgi işleme, analiz etme ve sonuç çıkarma gibi insan beyninin özelliklerini taklit etmek için tasarlanmış bir yapay zeka uygulamasıdır. YSA, otomatik olarak ve yardım almadan öğrenme sonucunda yeni bilgiler oluşturma, türetme ve keşfetme gibi özellikleri gerçekleştirmek için kullanılan bir yapay zeka uygulamasıdır. YSA, sistemdeki girdileri kullanarak karşılaştığı durumlar arasındaki farklılıklara veya benzerliklere göre karar verir. Bu nedenle YSA, modellenmesi çok zor veya imkansız olan problemler için geliştirilmiş bir yöntemdir. Aynı zamanda YSA; İnsan beyninin işlevleri esas alınarak geliştirilmiş, birbiriyle paralel çalışan ve aralarında bilgi alışverişinde bulunan kuruluşlardır [23]. Sinir ağı, bilgiyi depolamak, işlemek ve kullanıma sunmak için temel işlem bölümlerinden oluşan ve paralel olarak çalışabilen devasa bir işlemcidir [24].

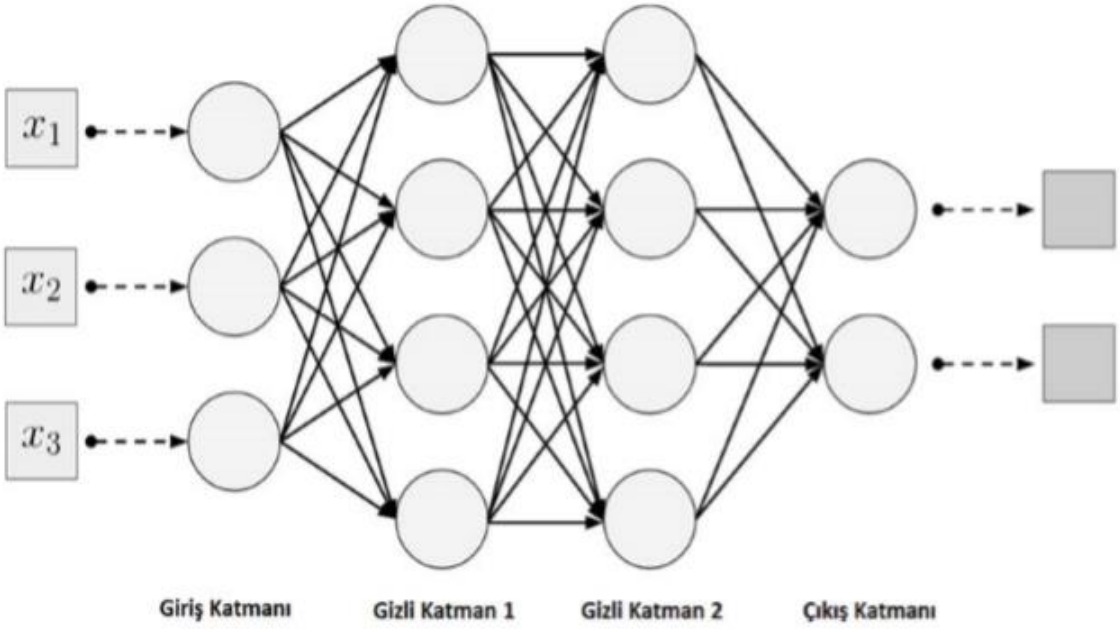


Şekil 2.4 Biyolojik sinir hücresi ve yapay sinir hücresinin karşılaştırılması

2.5.2 Çok Katmanlı Yapay Sinir Ağları

Çok katmanlı bir sinir ağı, birden fazla yapay nöron veya düğüm katmanı içerir. Tasarımda büyük farklılıklar gösterirler. Tek katmanlı sinir ağlarının AI'nın evriminin başlarında faydalı olmasına rağmen, günümüzde kullanılan ağların büyük çoğunluğunun çok katmanlı bir modele sahip olduğunu belirtmek

önemlidir. Çok katmanlı sinir ağları çeşitli şekillerde kurulabilir. Tipik olarak, bir dizi gizli katmana ağırlıklı girdiler gönderen en az bir girdi katmanına ve sonunda bir çıktı katmanına sahiptirler. Bu daha karmaşık kurulumlar, yapay nöronların ateşlenmesini veya aktivasyonunu yönlendirmek için sigmoidleri ve diğer işlevleri kullanan doğrusal olmayan yapılarla da ilişkilidir. Bu sistemlerden bazıları fiziksel olarak fiziksel malzemelerle oluşturulabilirken, çoğu sinirsel aktiviteyi modelleyen yazılım işlevleriyle oluşturulur [25].



Şekil 2.5 Çok katmanlı ağ modeli [22]

Ağ parametreleri başlatıldıktan sonra YSA öğrenme süreci başlamalıdır. Giriş verileri düzensiz ve rastgele şekilde belirlenir; ancak, sezgisel yaklaşım ile muhtemel değerler girilirse, iterasyonların hızı artacaktır. Eğitim veri seti üzerinde öğrenme olur. Bu, her girdi için öğrenme verisinde üretilen çıktıların hata oranlarına göre başarı yüzdelerinin olduğu bir süreçtir, daha sonrasında sonuçların iyileştirilmesi için tekrarlanır. Ağ, eğitim verileri üzerinde yüksek bir doğruluk değerine ulaşınca, ağ eğitilmiş olarak kabul edilebilmektedir. Aşırı uyum, öğrenme sürecine bağlıdır çok uzun süre öğrenme yapılan ağlarda daha sık rastlanır [22] .Başka bir yönden eğitim seti, çok küçük olduğunda oluşan bir sorundur bu sorunu çözmek adına veri setini büyütmek adına işlemler yapılmalıdır veri seti kendi verisi ile arttırılarak modele dahil edilebilir veya başka bir kaynaktan veri bulunabilir.

2.5.3 İleri Beslemeli Yapay Sinir Ağları

İleri beslemeli sinir ağlarında bir katmandaki tüm düğümler, bir sonraki katmandaki düğümlere bağlanır ve katmanlar, girdiden çıktıya ileri yönde birbirini besler. Çoğu durumda, türevlenebilen aktivasyon fonksiyonu kullanılır. Gizli katmanlarda birkaç katman kullanırken, işlem süreleri açısından hücrelerin miktarına dikkat etmek çok önemlidir; "Kısa" ağlar yerine "Uzun" ağlar seçilmelidir. [26], [27], [28].

2.5.4 Geri Beslemeli Yapay Sinir Ağları

Geri bildirim ileten yapay sinir ağları, ağın ağırlıklarını değiştirmek ve mevcut hatayı azaltmak için ağın gerçek çıktısı ile amaçlanan çıktı arasındaki farkı hesaplar.

Buradaki fikir, hatayı minimumda tutmaktır. Her birim bu hata hesaplamasına ve düzeltmesine tabi tutulur. Üç katman, çok katmanlı sensörleri oluşturur: giriş, gizli ve çıkış. Giriş katmanı, verileri bir vektör biçiminde alır. Gizli katman, ağırlık girdisini ve ayrıca girdi katmanından gelen verileri işlemek için bir etkinleştirme işlevi kullanır. Çıktı katmanı, gizli katmanda yapılan işlemlerin sonucunu ağırlık girdisi ile birlikte alır ve işler. Sinir ağının çıktıları, çıktı katmanı nöronlarının çıktılarıdır. Çok az gizli nörona sahip bir ağ, kalıpları tanıyamayabilir, ancak bir ağ çok fazla gizli katmana sahip ise, eğitim seti kalıplarını hatırlayabilir. Geri beslemeli bir sinir ağında, ağı güncellemek için aşağıdaki işlemler kullanılır.

Adım 1: Ağın giriş katmanına bir vektör iletilir. Bu katmandaki tüm nöronların çıktısı daha sonra algoritma tarafından hesaplanır. Sonuç, bir sonraki katmana iletilir ve önceki katmanın çıktısı olan çıktı katmanına ulaşılana kadar bir sonraki katmana devam eder. Bu adımda aslında her katman tahmin çıktısı üretmiştir.

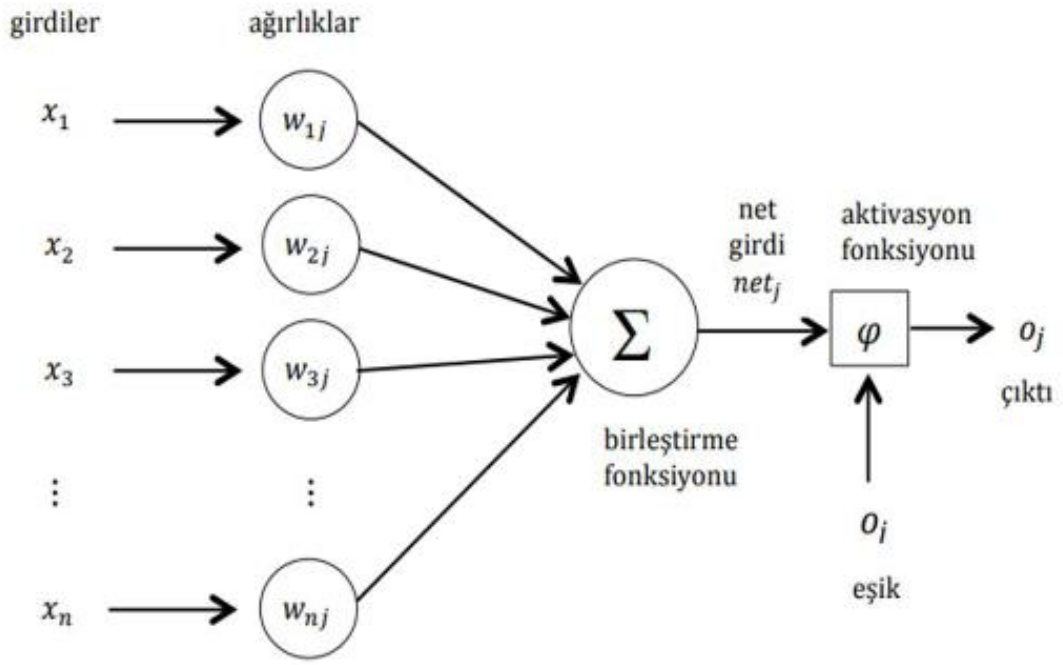
Adım 2: Daha sonra algoritma ağın çıktılarının hata değerini ölçer (Bir hata fonksiyonu kullanarak gerçek ile tahmin değeri arasındaki farkın en düşük olmasını sağlamaya çalışır.)

Adım 3: Ölçülen hatalar geri yönlü tekrar tahmin edilerek giriş katmanına kadar devam eder.

Adım 4: Algoritma "Eğim-iniş" adımını gerçekleştirdiği son adımda hesapladığı hata eğimini kullanır. [27], [29], [28], [30].

2.5.5 Aktivasyon Fonksiyonları

Aktivasyon işlevi, ağın verilerdeki karmaşık kalıpları öğrenmesine yardımcı olmak için yapay sinir ağına eklenen bir işlevdir. Beynimizde bulunan nöron temelli bir modelle karşılaştırıldığında, aktivasyon işlevi sonunda bir sonraki nörona neyin ateşleneceğine karar verir. Aktivasyon fonksiyonları matematiksel türevlenebilen fonksiyonlardan tercih edilir, türevleme işlemi en iyileme yöntemlerinin temelini oluşturduğundan aslında yapılan işlem hata minimumlama yönlü olduğu söylenebilir.



Şekil 2.6 Aktivasyon fonksiyonları çalışma şekli

2.5.5.1 Doğrusal Aktivasyon Fonksiyonları

Bazen düz çizgi işlevi olarak adlandırılan doğrusal bir işlevin aktivasyonu, girdiyle veya nöronlardan gelen girdilerin ağırlıklı toplamıyla orantılıdır. Denklemi aşağıdaki gibidir:

$$f(x) = ax + b \tag{2.5.1}$$

Belirli bir aralıkla sınırlandırılamaması bu aktivasyonun dezavantajıdır. Aktivasyon işlevi, bu işlev her birleşme noktasına uygulandığında doğrusal regresyon gibi davranır. Sinir ağının son katmanı, ilk katmanın doğrusal bir işlevi olarak çalışacaktır.

Başka bir problem, gradyan iniş yoluyla farklılaşma gerçekleştirildiğinde, çıktının sabit olmasıdır, bu istenmeyen bir durumdur, çünkü geri yayılımın çıktısı ve mantığı, sabit hata değişimi oranı tarafından bozulabilir.

2.5.5.2 Doğrusal Olmayan Aktivasyon Fonksiyonları

En popüler aktivasyon fonksiyonlarının lineer olmayan fonksiyonlar olduğu bilinmektedir. Bir sinir ağı modelinin sonuçlar arasında ayrım yapmasını ve bir dizi veriye uyum sağlamasını kolaylaştırır. Doğrusal olmayan aktivasyon fonksiyonları aşağıdaki gibi ayrılabilir.

Sigmoid Aktivasyon Fonksiyonları:

Sigmoid fonksiyonu girilen herhangi bir reel sayıyı (0,1) aralığındaki bir değere indirgeyecek şekilde çalışır.

Tanh Aktivasyon Fonksiyonları:

Bir sinir ağının katmanları arasında doğrusal olmayan bir etkinleştirme işlevi olarak kullanılabilecek başka bir potansiyel işlev, Tanh işlevidir. Bununla sigmoid aktivasyon fonksiyonu arasında bazı benzerlikler vardır. Tanh işlevi, 0 ile 1 arasındaki girdi değerlerini eşleyen bir sigmoid işlevinden farklı olarak -1 ile 1 arasındaki değerleri eşler. fonksiyonun kendisi cinsinden ifade edilir.

ReLU Aktivasyon Fonksiyonları:

Maksimum (0, z), aralığındadır, lineer değildir ve kendisine çok benzeyen sigmoidden daha iyi sonuçlar verir.

Maksimum çıkış:

ReLU ve sızdıran ReLU işlevleri, Maxout aktivasyonu ile genelleştirilir. Bırakma düzenlileştirme yaklaşımıyla birlikte kullanılması amaçlanan, maksimum girdiyi döndüren parçalı doğrusal bir işlevdir. ReLU ve sızdıran ReLU, Maxout'un

benzersiz örnekleridir. Bu nedenle, Maxout nöronu, ölmekte olan bir ReLU'nun dezavantajları olmaksızın bir ReLU biriminin tüm avantajlarına sahiptir. Her nöron için toplam parametre sayısını iki katına çıkardığı için daha fazla toplam parametre öğretilmelidir.

ELÜ:

Genellikle daha hızlı yakınsayan ve daha kesin sonuçlar veren bir işlev, üstel doğrusal birim veya ELU'dur. ELU, diğer etkinleştirme işlevlerinin aksine, pozitif bir nicelik olması gereken ek bir alfa sabitine sahiptir. Negatif girdiler dışında, ELU ve ReLU oldukça benzerdir. Negatif olmayan girdiler için her ikisi de kimlik işlevi biçimindedir. Keskin bir şekilde yumuşatan ReLU'nun aksine, ELU çıktısı -'ye eşit olana kadar kademeli olarak düzgün hale gelir.

Softmax Aktivasyon Fonksiyonları:

Bir olayın 'n' farklı olaylar arasındaki olasılık dağılımı, Softmax işlevi kullanılarak hesaplanır. Bu fonksiyon, genel olarak, her hedef sınıfın tüm potansiyel hedef sınıflar üzerindeki olasılığını belirleyecektir. Sağlanan girdiler için hedef sınıf daha sonra hesaplanan olasılık yardımıyla belirlenecektir. [31].

Yaygın olarak kullanılan aktivasyon fonksiyonlarından bazıları Şekil 2.7'de verilmiştir.

Linear	$f(x) = ax + c$	$f'(x) = a$
Sigmoid	$f(x) = \frac{1}{1+e^{-x}}$	$f'(x) = f(x)\,(1 - f(x))$
TanH	$f(x) = tanh(x) = \frac{2}{1+e^{-2x}} - 1$	$f'(x) = 1 - f(x)^2$
ReLU	$f(x) = \begin{cases} 0 \text{ for } x<0 \\ x \text{ for } x \geq 0 \end{cases}$	$f'(x) = \begin{cases} 0 \text{ for } x<0 \\ 1 \text{ for } x \geq 0 \end{cases}$
Parametric ReLU	$f(x) = \begin{cases} \alpha x \text{ for } x<0 \\ x \text{ for } x \geq 0 \end{cases}$	$f'(x) = \begin{cases} \alpha \text{ for } x<0 \\ 1 \text{ for } x \geq 0 \end{cases}$
ELU	$f(x) = \begin{cases} \alpha(e^x - 1) \text{ for } x<0 \\ x \text{ for } x \geq 0 \end{cases}$	$f'(x) = \begin{cases} f(x) + \alpha \text{ for } x<0 \\ 1 \text{ for } x \geq 0 \end{cases}$

Şekil 2.7 En çok kullanılan aktivasyon fonksiyonları **[31]**

2.5.6 Optimizasyon Yöntemleri

Derin ağlarla, birkaç optimizasyon yaklaşımı iyi sonuçlar verir, bu nedenle bu teknikler yaygın olarak kullanılmaktadır. Bu stratejilerden bazıları bu bölümde gösterilecektir. Yöntemler, basit stokastik gradyan inişine dayanmaktadır. [22].

Stokastik Gradyan İnişi (SGİ) Optimizasyon Yöntemi: Gradyan sinir ağının katsayılarını bulmak için tekrar tekrar bulunur. Genel olarak, eğitim seti büyük olduğunda tüm veri seti için gradyanları hesaplamak çok kullanışlı değildir. Hem yavaş hem de çok fazla bellek kaplıyor. Bunu akılda tutarak, birkaç örnek üzerinden hesaplanan gradyan olan stokastik gradyan inişi kullanılır (tüm set yerine). Stokastik gradyan inişi kullanmanın bir başka avantajı da yerel minimumların genişleme yeteneğidir. Küçük partiler, öğrenme varyansını azalttığı ve dolayısıyla daha kararlı yakınsama olduğu için yaygın olarak uygulanır. Diğer

bir neden ise GPU'ların yüksek bilgi işlem gücüne sahip olmasıdır., işleme kolayca paralelleştirildiğinden küçük partilerin çok hızlı bir şekilde işlenebilmesidir. Aşağıdaki denklem stokastik gradyan inişinin iterasyon adımını göstermektedir.

$$\theta = \theta - \alpha * \sum_{k=1}^{i+m} \nabla_\theta J\left(\theta; x^{(k)}; y^{(k)}\right) \tag{2.5.2}$$

Burada θ güncellenecek parametredir ve α öğrenme oranıdır.

Nesterov Hızlandırılmış Gradyan (NHG) Optimizasyon Yöntemi: Gradyan inişinin bir sınırlaması, düz alanlarda takılıp kalabilmesi veya amaç fonksiyonu gürültülü gradyanlar döndürmesi durumunda zıplayabilmesidir. Momentum, düz alanlar arasında gezinmek ve zıplama eğimlerini yumuşatmak için aramanın ilerlemesini hızlandıran bir yaklaşımdır.

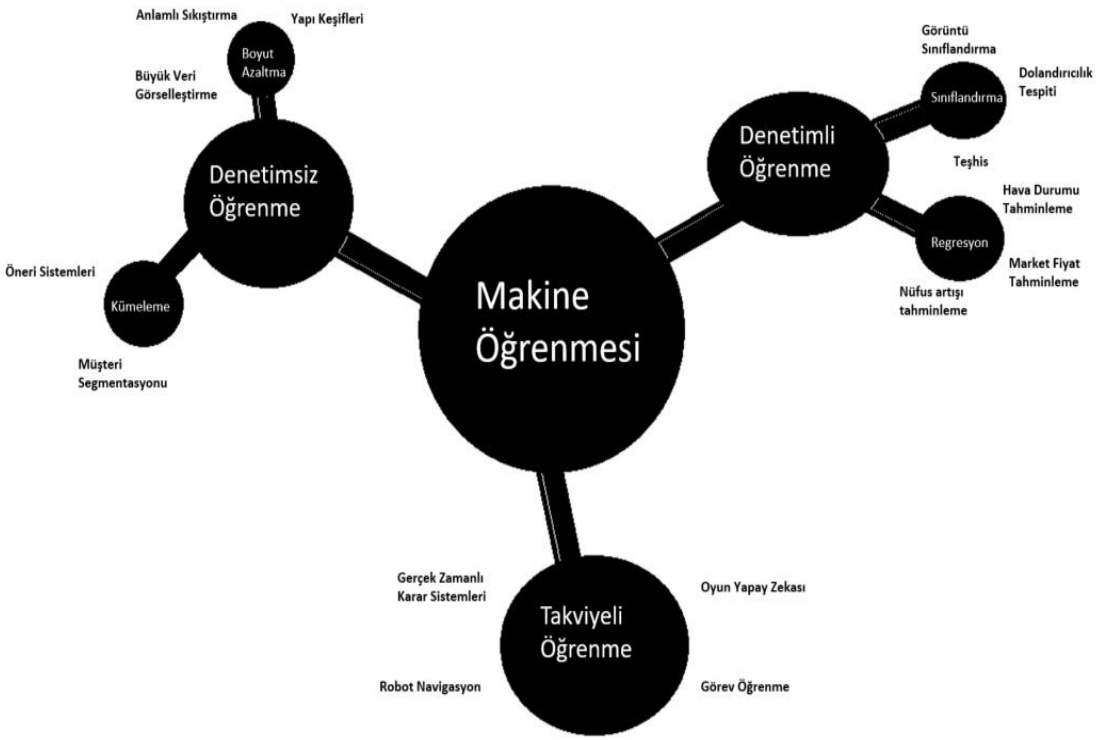
$$\text{r}.v_t = \mu_{t-1} v_{t-1} - \epsilon_{t-1} \nabla f(\theta_{t-1} + \mu_{t-1} v_{t-1})$$

$$\theta_t = \theta_{t-1} + \mu_t \tag{2.5.3}$$

Yukarıda denklemi verilen NHG optimizasyonu derin sinir ağlarında sıkça kullanılabilir denklemdeki parametreler uygulayan kişi tarafından seçilebilir. [32]

ADAM Optimizasyon Yöntemi: Adam optimizer, gelecek yıllarda bilgisayarla görme ve doğal dil işleme gibi çeşitli derin öğrenme uygulamalarında uygulanabilecek stokastik gradyan inişinin genişletilmiş versiyonudur.

Adam ilk olarak 2014 yılında tanıtıldı. İlk olarak ICLR 2015 adlı derin öğrenme araştırmacıları için ünlü bir konferansta sunuldu. Stokastik gradyan iniş sürecine alternatif olabilecek bir optimizasyon algoritmasıdır. Ad, uyarlanabilir moment tahmininden türetilmiştir. Optimize ediciye Adam denir, çünkü öğrenme hızını sinir ağının her ağırlığına uyarlamak için gradyanın birinci ve ikinci anlarının tahminlerini kullanır. Sıkça kullanılmakta olan ADAM oldukça hızlıdır.



Şekil 2.8 Makine öğrenimi ve derin öğrenme kullanım alanları

2.6 Makine Öğrenmesi ve Derin Öğrenme Kullanım Alanları?

Şekil 2.8'de makine öğrenmesinin kategorilerini ve kullanım alanlarını görmekteyiz. Ayrıntılı olarak bakıldığında günümüzde birçok alanda ve sistemde kullanılmaktadır. Kullanıldığı alanlar aşağıdaki gibi örneklendirilebilir.

Görüntü Tanıma:

Makine öğreniminde en çok kullanılan uygulamalardan biridir. Nesneleri, kişileri, yerleri, dijital görüntüleri vb. tanımlamak için kullanılır.

Görüntü işleme arkadaş etiketleme önerisinde de kullanılır. Örneğin; Facebook bize otomatik arkadaş etiketleme önerisi özelliği sunuyor, arkadaşlarımızın olduğu fotoğrafları yüklediğimizde otomatik olarak bir etiket önerisi veriliyor. Yüz tanıma ve görüntü algılama algoritması makine öğrenmesinin önemli bir parçasıdır.

Konuşma Tanıma:

Google'ı kullanırken "Sesle ara " seçeneğiyle karşılaşıyoruz, bu durum konuşma tanımaya giriyor ve popüler bir makine öğrenimi uygulaması konuşma tanıma, sesli talimatları metne dönüştürme işlemidir ve " Konuşmadan metne " veya " Bilgisayar konuşma tanıma " olarak da bilinir.

Birçok ses tanıma uygulaması şu anda makine öğrenimi yöntemlerini kullanmaktadır. Konuşma tanıma teknolojisi, Google Asistan, Siri, Cortana ve Alexa gibi sesli komut asistanları tarafından kullanılır.

Trafik tahmini:

Trafiğin temiz mi, yavaş mı, yoksa çok sıkışık mı bilgilerini verir. Aracın Gerçek Zamanlı konumu Google Harita uygulamasından ve sensörlerden tespit edilir. Google Harita'yı kullanan herkes bu uygulamanın daha iyi hale gelmesine yardımcı oluyor. Performansı artırmak için kullanıcıdan bilgi alır ve veri tabanına geri gönderir.

Ürün önerileri:

Makine öğrenimi, kullanıcıya ürün tavsiyesi için Amazon, Netflix vb. gibi çeşitli e-ticaret ve eğlence şirketleri tarafından yaygın olarak kullanılmaktadır.

İnternette yaptığımız aramalar sonrası sitelerin bize önerdiği ürünlerin sebebi öneri sistemi kullanılan makine öğrenmesi algoritmalarıdır.

Google, çeşitli makine öğrenimi algoritmaları kullanarak kullanıcının ilgisini anlar ve müşteri ilgisine göre ürünü önerir.

Benzer şekilde, Netflix kullandığımızda eğlence dizileri, filmler vb. için bazı öneriler alıyoruz ve bu da makine öğrenmesi yardımıyla yapılmaktadır.

Sürücüsüz araçlar:

Makine öğreniminin en heyecan verici uygulamalarından biri kendi kendini süren arabalardır. Kendi kendini süren arabalarda makine öğrenimi önemli bir rol oynar. En popüler otomobil üretim şirketi Tesla, sürücüsüz otomobil üzerinde çalışıyor.

Araba modellerini sürüş sırasında insanları ve nesneleri algılamak için eğitmek için denetimsiz öğrenme yöntemini kullanıyor.

E-posta İstenmeyen Posta ve Kötü Amaçlı Yazılım Filtreleme:

Yeni bir e-posta aldığımızda, önemli, normal ve spam olarak otomatik olarak filtrelenir. Gelen kutumuza her zaman önemli sembolü olan önemli bir posta ve spam kutumuzdaki spam e-postaları alırız ve bunun arkasındaki teknoloji Makine öğrenimidir.

Çevrimiçi Dolandırıcılık Tespiti:

Makine öğrenimi, dolandırıcılık işlemini tespit ederek çevrimiçi işlemimizi güvenli ve güvenli hale getiriyor. Ne zaman bir çevrimiçi işlem gerçekleştirsek, sahte hesaplar, sahte kimlikler ve ve para çalma işlemin gerçekleşmesinin çeşitli yolları olabilir. Bunu tespit etmek için, Feed Forward Sinir ağı, bunun gerçek bir işlem mi yoksa bir sahtekarlık işlemi mi olduğunu kontrol ederek bize yardımcı olur.

Borsa ticareti:

Makine öğrenimi, borsa ticaretinde yaygın olarak kullanılmaktadır. Borsada, hisselerde her zaman iniş ve çıkış riski vardır, bu nedenle bu makine öğreniminin uzun kısa süreli bellek sinir ağı borsa trendlerinin tahmini için kullanılır.

Tıbbi Teşhis:

Akciğer hastalıklarının erken teşhisinde hastaların filmleri incelenerek geleceğe yönelik hastalık tahmininde makine öğrenmesi algoritmaları kullanılır.

Genel olarak, görüntü işleme kullanılarak ileride oluşabilecek hastalıkların tahminlenmesinde yine derin öğrenme kullanılır.

Otomatik Dil Çevirisi:

Günümüzde, yeni bir yeri ziyaret edersek ve dilin farkında değilsek, bu hiç sorun değil, çünkü burada da makine öğrenmesi metni bilinen dillerimize çevirerek bize yardımcı oluyor. Google'ın GNMT'si (Google Neural Machine Translation), metni tanıdık dilimize çeviren bir Neural Machine Learning olan ve otomatik çeviri olarak adlandırılan bu özelliği sağlar.

3
MATERYAL VE YÖNTEM

Bu çalışmada yukarıda anlatıldığı üzere yapay zeka ve makine öğrenmesinin bir alt dalı olan derin öğrenme mimarisi kullanılarak tahmin yapan bir sistem oluşturma amacıyla dünyada en çok satranç oynanan ve veri setini açık olarak paylaşan Lichess sitesinin verisi kullanıldı. Veri seti oyuncu ID'leri, puanları, açılış türleri, vb. birçok değişken içeriyor gerçek hayat verisi olduğundan ayrık değer sayısı yok denecek kadar az olan veri seti kategorik ve sayısal değişkenler içerdiğinden belli dönüşümlerden geçerek algoritmaya dahil edilmesi gerekmektedir. Çalışma diğer çalışmalara kıyasla satranç oynayan bir makine tasarlamak yerine satranç maçı sonucu tahmin eden bir model oluşturmak amacıyla yapılmıştır.

Günümüzde hemen hemen çoğu kişinin bildiği ve oynadığı bu oyunda oyun başında beyaz taşlar ile oynamayı seçersek kazanma olasılığımız ne olur? Gerçekten avantajlı bir seçim midir? Yaptığımız açılışın kazanma olasılığımızı nasıl etkiler gibi soruların analizini görmek ve cevaplamak adına çalışmada görsel teknikler kullanılarak veri seti incelendi.

Çıktılar ile sonucun oynanılan açılış ile ve renk ile ilişkili olduğu tespit edilip tahminleme yapan modelin bundan beslenerek kazananı belli bir doğruluk oranında doğru tahminlemesi beklentisi ile model kuruldu.

Kurulan model donanımın ve ağın gücüyle orantılı olduğu farklı yöntemler denenerek görüldü. Uygulama Python programı ile yazılmış olup, veri düzenleme için Numpy ve Pandas, görselleştirmeler için Seaborn ve Matplotlib, makine öğrenmesi modelinde ise Tensorflow ve Keras kütüphanesi kullanılmıştır.

3.1 Veri Setinin İncelenmesi

Bu çalışmada dünyada en çok satranç oyunu oynanan online satranç sitesi olan Lichess sitesinden alınmış yaklaşık 20 bin satranç oyunu verisi kullanılarak yapılan

bir derin öğrenme uygulaması anlatılacak olup oluşturulan yapay zeka girilen veriye göre oyunu kazanan tarafı tahmin etmektedir.

Uygulama için Lichess sitesinden alınmış 20 bin satranç verisinin 8 değişkeni incelenerek model oluşturulmuştur. Bunun için Python programı kullanılmış olup oluşturulan yapay zekanın, kullanıcı kendi bilgilerini girdiği takdirde maçı kazanıp kazanmayacağı hakkında sonuç vermesi sağlanmıştır.

Veri setinde başlangıçta oynayan kişilerin kullanıcı adlarını, oyun kodlarını vb. içeren değişkenlerin uygulama için gereksiz görülmesi sonucu veri seti o değişkenlerden temizlenmiştir. Bazı değişkenler farklı varyasyonları ile modele dahil edilmiştir. Kalan değişkenler için ise literatüre uygun şekilde kodlama yapılarak veri içinde dönüşümler yapılmıştır. Veri Tablo 3.1'de gösterilmektedir.

Tablo 3.1 Veri seti ve açıklamaları

ID	Oyun ID	Benzersiz Oyun Adı
Rated	Oyun Puanlı mı?	1-0
Turns	Tur Sayısı	Sayısal
Victory_Status	Oyunun Bitiş Şekli	Kategorik
Winner	Oyunu Kazanan Renk	1-0
Game Category	Oyunun Türü Zamana Göre	Kategorik
White Rating	Beyaz Oyuncu Puanı	Sayısal
Black Rating	Siyah Oyuncu Puanı	Sayısal
Opening Eco	Evrensel Olarak Kullanılan Açılış Kodu	Kategorik
Opening Name	Açılış Adı	Yazı
Opening Ply	Açılışın Kaç Hamle Sürdüğü Bilgisi	Sayısal

Veri seti incelendiğinde herhangi bir maç için boş değerin olmadığı görüldü görsel analizleri ve modeli etkilememesi adına nötr olmasından dolayı berabere biten oyunlar veri setinden atıldı. Görsel analizlerde kullanılmak üzere ilk hamleye göre oynanan tüm oyunlara varyasyon bilgisi içermeyecek şekilde genel bir açılış adı verildi e4 ile başlayan maçlar için “King’s Pawn’ , d4 ile başlayan maçlar için Queen’s Pawn, c4 ile başlayan maçlar için “English ve kalanları için “Other” adlı yeni değişkenler veri setine değişken olarak eklendi.Görsel analiz ve YSA modeline dahil edilirken süre değişkeni 3 dakikadan uzun maçlar için ‘bullet’, 3-10 dakika arası maçlar için “blitz” ve 10 dakikadan uzun süren maçlar için “rapid” değişkenine çevrilerek oyun sistemine uygun hale getirildi.

Index	id	rated	ated	mov	turns	ictory statu	winner	ement c	white id	hite ratin	black id	lack ratin	nove	pening e	ening nai	pening p
0	TZJHLljE	False	1.5…	1.5…	13	outoftime	white	15+2	bourgr…	1500	a-00	1191	d4 …	D10	Slav D…	5
1	l1NXvwaE	True	1.5…	1.5…	16	resign	black	5+10	a-00	1322	skinnerua	1261	d4 …	B00	Nimzow…	4
2	mIICvQHh	True	1.5…	1.5…	61	mate	white	5+10	ischia	1496	a-00	1500	e4 …	C20	King's…	3
3	kWKvrqYL	True	1.5…	1.5…	61	mate	white	20+0	daniam…	1439	adivanov2…	1454	d4 …	D02	Queen'…	3
4	9tXo1AUZ	True	1.5…	1.5…	95	mate	white	30+3	nik221…	1523	adivanov2…	1469	e4 …	C41	Philidor Defense	5
5	MsoDV9wj	False	1.5…	1.5…	5	draw	draw	10+0	trelyn…	1250	franklin1…	1002	e4 …	B27	Sicili…	4
6	qwU9rasv	True	1.5…	1.5…	33	resign	white	10+0	capa_jr	1520	daniel_li…	1423	d4 …	D00	Blackm…	10
7	RVN0N3VK	False	1.5…	1.5…	9	resign	black	15+30	daniel…	1413	soultego	2108	e4 …	B00	Nimzow…	5
8	dwF3DJHO	True	1.5…	1.5…	66	resign	black	15+0	ehabfa…	1439	daniel_li…	1392	e4 …	C50	Italia…	6
9	afoMwnLg	True	1.5…	1.5…	119	mate	white	10+0	daniel…	1381	mirco25	1209	e4 …	B01	Scandi…	4
10	HgKLWPsz	False	1.5…	1.5…	39	mate	white	20+60	daniel…	1381	anaissac	1272	e3 …	A00	Van't …	1

Şekil 3.1 Veri setinin ilk 10 verisi

3.2 Keras ve Python

Keras, Python ile yazılmış, makine öğrenimi platformu TensorFlow'un üzerinde çalışan bir derin öğrenme API'sidir. Hızlı denemeyi mümkün kılmaya odaklanılarak geliştirilmiştir. Fikirden sonuca mümkün olduğunca hızlı gidebilmek, iyi araştırma yapmanın anahtarıdır.

Keras’ın özellikleri 3 ana grupta açıklanabilir;

- Basit, Keras, sizi problemin gerçekten önemli olan kısımlarına odaklanmaktan kurtarmak için geliştiricinin bilişsel yükünü azaltır.

- Esnek, Keras, karmaşıklığın aşamalı olarak ifşa edilmesi ilkesini benimser: basit iş akışları hızlı ve kolay olmalı, keyfi olarak gelişmiş iş akışları ise halihazırda öğrendiklerinizin üzerine inşa edilen net bir yol aracılığıyla mümkün olmalıdır.
- Güçlü, Keras, sektör gücünde performans ve ölçeklenebilirlik sağlar: NASA, YouTube veya Waymo gibi kuruluşlar ve şirketler tarafından kullanılır [33]. Keras bu özellikleri ile öne çıkmış olup, model için tercih edilmiştir.

3.3 Derin Öğrenme Modeli Kurma

Derin öğrenme algoritmasında model aşağıdaki şekilde kurgulanır:

- Veriyi tanıma
- Ön işleme
- Veriyi bölme
- Eğitim veri kümesi ve test veri kümesinin belirlenmesi
- Modelin eğitilmesi
- Modelin testi ve doğrulanması

  Bir makine öğrenimi modelinde öncelikle verileri tanımak ve eldeki verilerle ne yapılacağına karar vermek gerekir. Bir sonraki adımda, mevcut verilerin makine öğrenmesi sürecine uygulanabilmesi için bazı ön işleme yöntemleri uygulanmalı ve veriler öğrenmeye hazır hale getirilmelidir. Doğru ve başarılı sonuçlar elde etmek için verilerin modele doğru öğretilmesi önemlidir. Ön işleme tamamlandıktan sonra veriler belirli bir oranda eğitim ve test verileri olarak ayrılmalıdır.

- Eğitim verisi
- Test verisi [14]

3.4 En Uygun Modelin Seçimi

Genel olarak sınıflandırma algoritmalarında performans değerlendirmesi için sınıfların gerçek ve tahmin değerleri karışıklık matrisi ile karşılaştırılır [15]. Matrisin her satırı, gerçek değeri gösterecek şekilde ve sütun ise tahmin edilen (test sonucu) değeri gösterecek şekilde tanımlanır. Ana köşegen üzerindeki sayılar

oldukça büyük ve köşegen dışındaki sayılar küçük olduğunda iyi sonuçlar elde edilir [34]. Burada hedef değişkenin gerçekte olan değeri ile tahmin neticesinde tespit edilen değeri Tablo 3.2'deki gibi karşılaştırılır. Bu model için tahmin ettiği değerleri ve gerçek değerleri sayısal olarak görebildiğimiz bir matris olup tahmin modellerinde çokça kullanılmaktadır.

Tablo 3.2 Karışıklık matrisi [35]

	Tahmin Pozitif	**Tahmin Negatif**
Gerçek Pozitif	Doğru Pozitif (DP)	Yanlış Negatif (YN)
Gerçek Negatif	Yanlış Pozitif (YP)	Doğru Negatif (DN)

Karışıklık matrisinde DP, gerçekte pozitif olan ve pozitif olarak sınıflandırılan; YP, gerçekte negatif olan ve pozitif olarak sınıflandırılan; DN, gerçekte negatif olan ve negatif olarak sınıflandırılan; YN, gerçekte pozitif olan ve negatif olarak sınıflandırılan olarak tanımlanır [35].

Sınıflandırma yöntemleri ile üretilen ve modelin başarısını ölçmek için kullanılan Doğruluk, Kesinlik, Duyarlılık, F1 Puanı ve Kappa Puanı, karışıklık matrisi kullanılarak elde edilebilir. Kesinlik, pozitif bir gerçek değere sahip ve pozitif olarak sınıflandırılan gözlemlerin sayısının, pozitif bir tahmin değerine sahip tüm gözlemlere oranıdır; Duyarlılık, pozitif gerçek değere ve pozitif tahmine sahip gözlemlerin sayısı ile pozitif gerçek değere ve negatif tahmin değerine sahip gözlemlerin toplamının toplam gözlem sayısına oranıdır. F1 Skoru ise kesinlik ve duyarlılık değerlerinin harmonik ortalamasını; Kappa Skoru, iki ya da daha fazla değerlendiriciye ait sonuçlar arasındaki uyumun güvenilirliğini ifade eder [35].

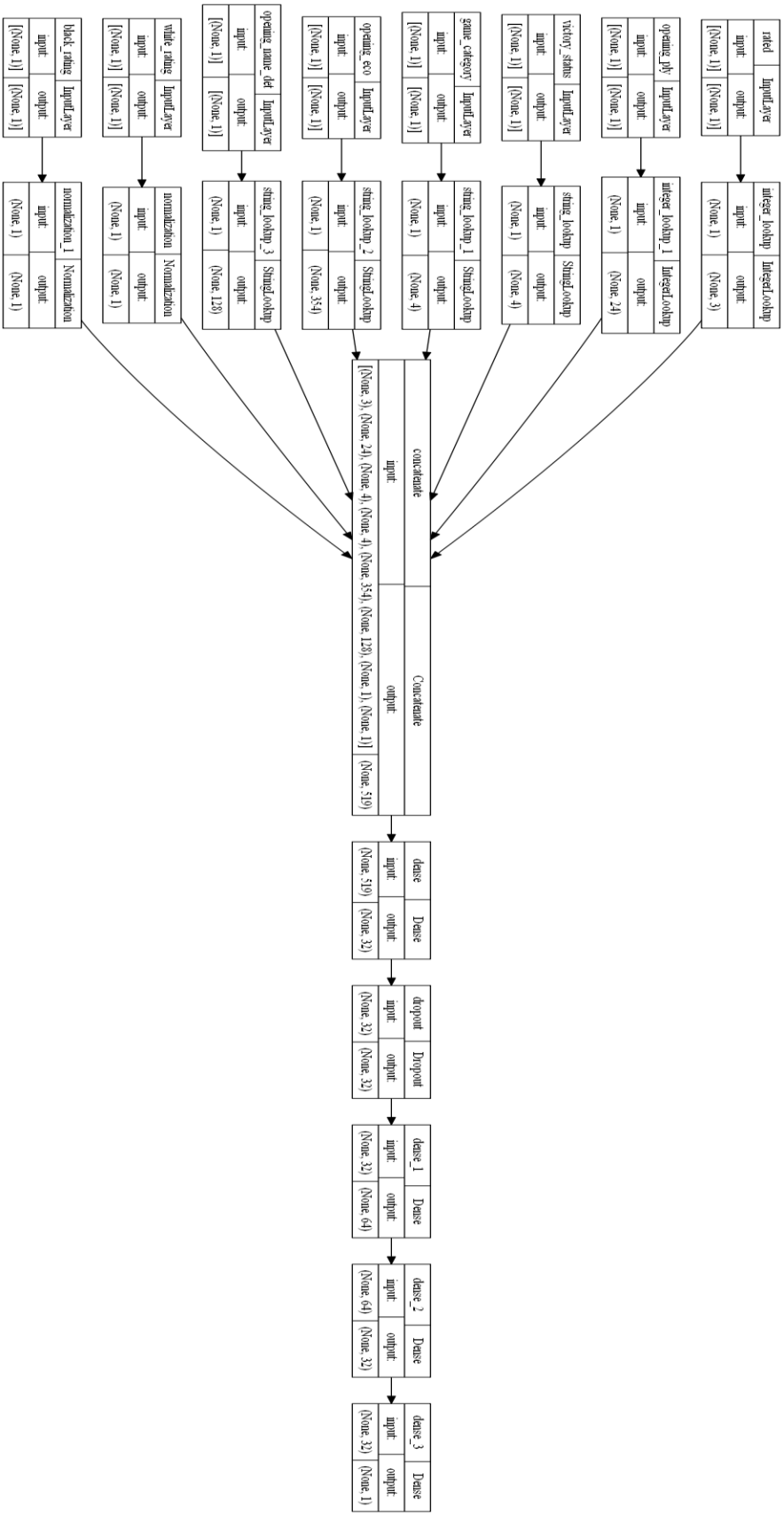
3.5 Modelin Kurulması

Veri %15'i test %85'i eğitim datası olarak ayrılıp modellenmiştir.Öncelikle 16 hücresi olan tek ara katman ile kurulan "Model 1" 10 "Epoch" iterasyondan geçerken 32,64,32 hücreden oluşan 3 ara katmanlı "Model 2" 100 "Epoch'dan geçerek incelenmiş olup ara katmanın ve epoch sayısının modele etkisini ve modelin gelişimi tasarlanmıştır.

Çalışmada değişkenlerden numerik olanlar normalize edilerek veriye girerken metin ve kategorik değişkenler de numaralandırılarak modele dahil edilmiştir.

Modellerde farklı aktivasyon fonksiyonları kullanılarak doğruluk değerleri kontrol edilip en yüksek doğruluk oranlı sonucu veren ‘ReLU’ ve ‘Sigmoid’ aktivasyon fonksiyonları ile yapılan modeller tercih edilmiştir.

Şekil 3.2’de modele giren değişkenler ve modelin şemasına yer verilmiştir.

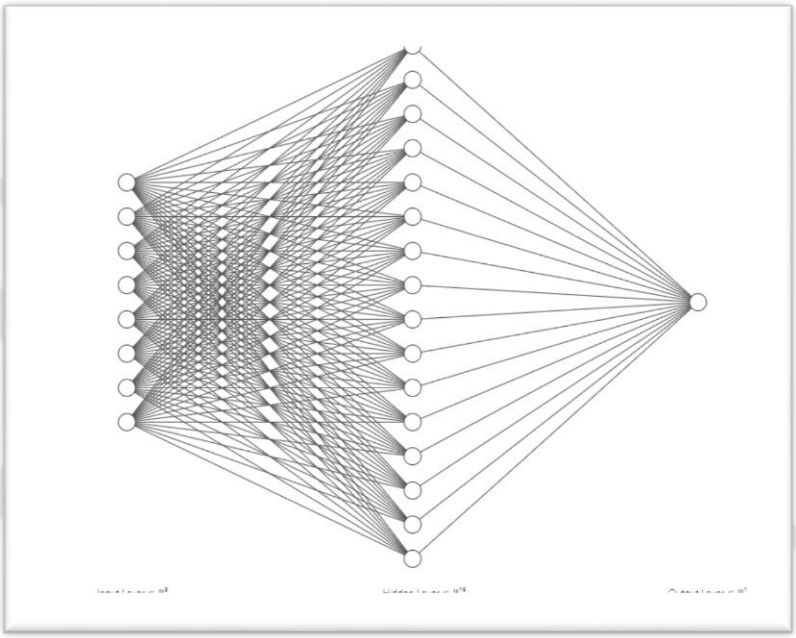


Şekil 3.2 Dönüşümden geçen değişkenlerin şeması

```
x = layers.Dense(16, activation="relu")(all_features)
x = layers.Dropout(0.5)(x)
output = layers.Dense(1, activation="sigmoid")(x)
```

Şekil 3.3 Model 1 için keras kodları

RELU aktivasyon fonksiyonu 16 hücresinde kullanılan sigmoid aktivasyon fonksiyonu çıkış katmanında olan Model 1 için ağın gösterimi Şekil 3.4'deki gibidir.

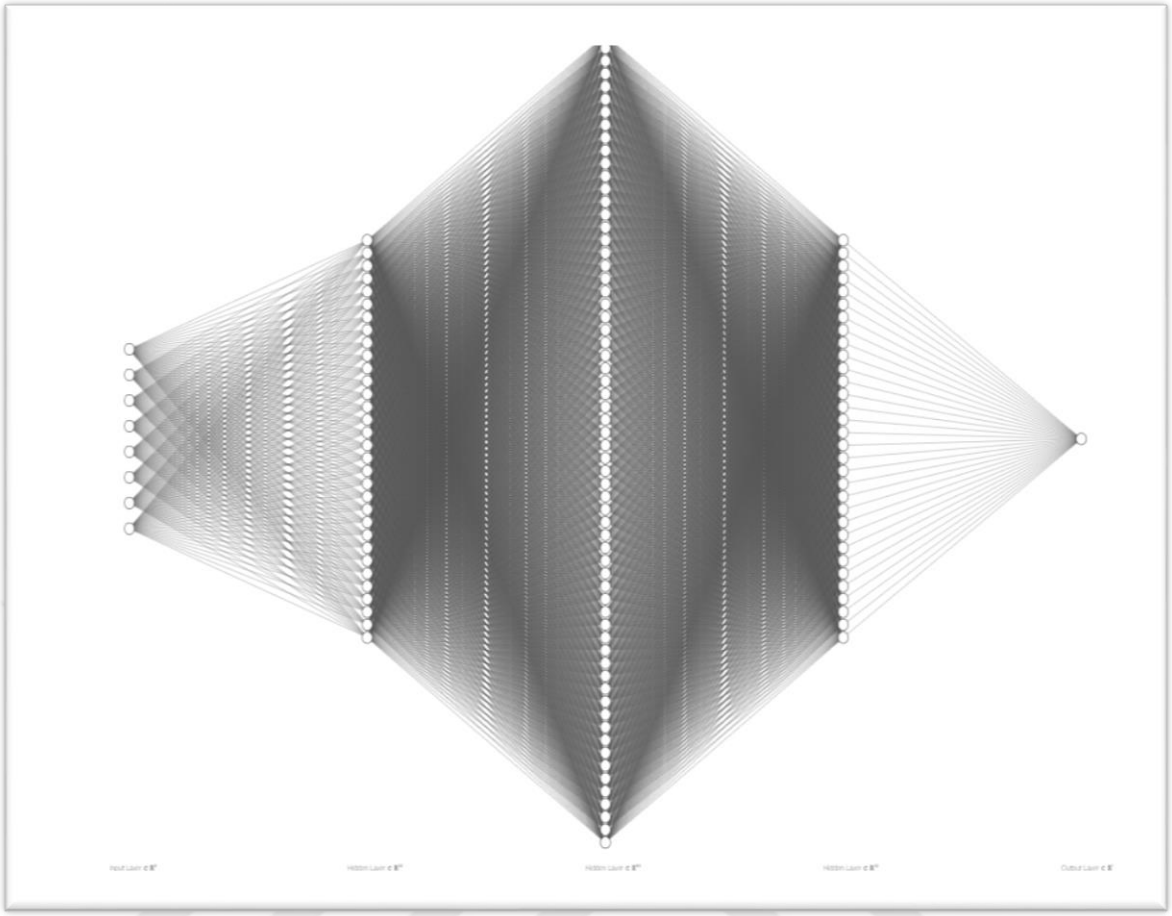


Şekil 3.4 Model 1 şeması

```
x = layers.Dense(32, activation="relu")(all_features)
x = layers.Dropout(0.5)(x)
x = layers.Dense(64, activation="relu")(x)
x = layers.Dense(32, activation="relu")(x)
output = layers.Dense(1, activation="sigmoid")(x)
```

Şekil 3.5 Model 2 kodları

Model 2'de ise giriş yapılan veriler sırasıyla 32,64,32 ve çıkış katmanı olan tek hücreli katmandan geçerek tahmin çıktısı verir. Her katmanda bir önceki katmandan çıkış yapan ve tahminlenen veriler giriş verisi olarak hücrelere iletilir.



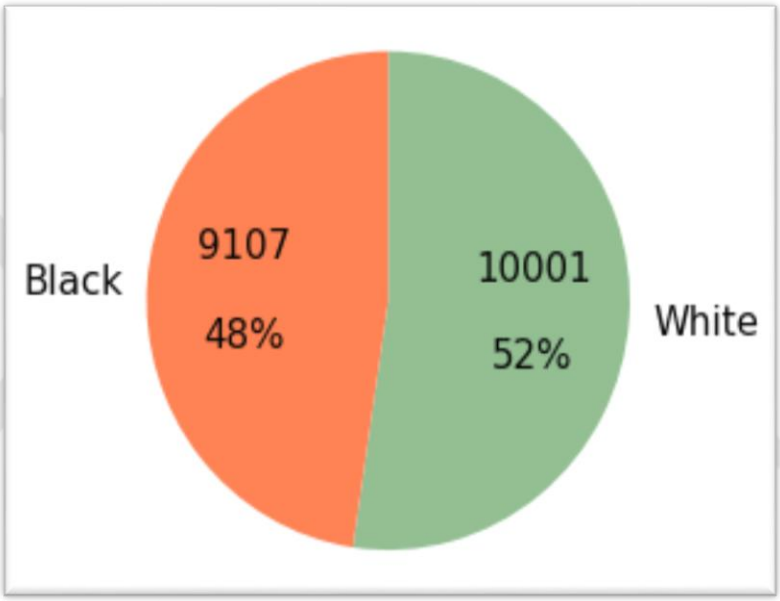
Şekil 3.6 Model 2 şeması

Şekil 3.5'te kodu verilen daha karmaşık şekilde hazırlanan Model 2'nin şematize edilmiş hali Şekil 3.6'daki gibidir. Şekilde de görüldüğü üzere daha karmaşık bir yapıda olan bu modelin daha fazla tahminleme yaparak daha iyi sonuçlar vereceği öngörülmektedir.

4
BULGULAR VE TARTIŞMA

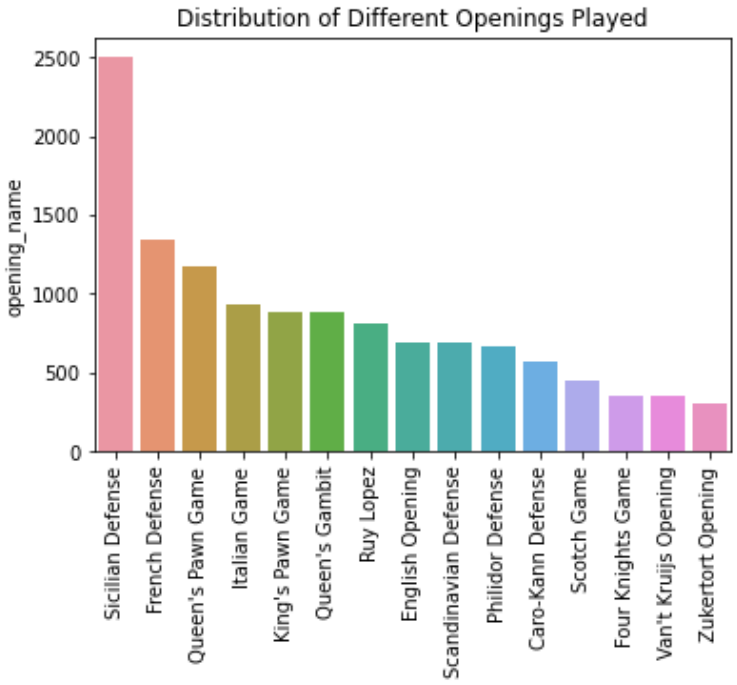
4.1 Verinin Görsel Analizleri Sonuçları

Verinin hem anlamlılığını görmek hem de analizden önce fikir edinmek adına görselleştirme teknikleri kullanılarak incelendi.



Şekil 4.1 Kazanma oranları ve sayıları

Şekil 4.1'deki grafikte ise siyah ve beyaz oyuncuların kazanma sayısı yer almakta, 1910 oyundan 10001 oyunu beyaz oyuncular kazanırken 9107 oyunu ise siyah oyuncu kazanmış olarak görülmekte bu sayılara göre %52 oranda beyaz %48 oranda siyah kazanmakta. Sadece bu veri özelinde önce başlayan oyuncuların yani beyazların kazanma oranını daha yüksek olduğunu görülmektedir. Açılış ve puanlardan bağımsız şekilde bir oyuncunun beyazı tercih etmesi takdirinde kazanma oranının artacağını bu grafik özelinde görebilmekteyiz kurulacak olan yapay sinir ağı modeli için oyuncunun renk tercihi dahi doğruluk oranını arttıracağı öngörülmektedir.



Şekil 4.2 En çok kullanılan açılışlar

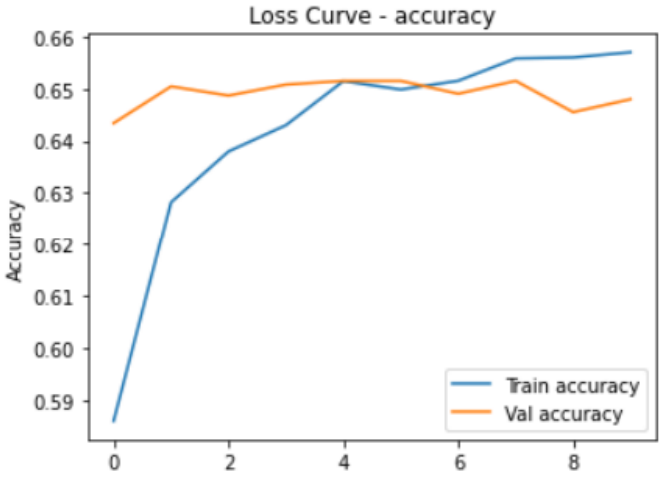
Karşılaşmaların Şekil 4.2'deki açılış sayısını veren grafiği incelendiğinde siyahın en güçlü açılışlarından Sicilya Defansı'nın ilk sırada olduğu görülmekte yine bir siyah açılışı olan Fransız Defansı'nın ikinci sırada olduğu görülürken vezir önünü iki açan beyaz oyuncuların Vezir Gambiti'nin üçüncü sırada olduğu sonucuna varılmıştır. Şekil 4.3'de açılışların tahtada gösterimi görülmektedir. Satranç oyununda merkez üzerinde taş bulundurmak oyunu kazanmaya yönelik yapılacak önemli stratejilerden biridir oyuncuların da genel olarak merkezi kontrol etmeye yönelik açılışlar tercih ettiği görülmektedir



Şekil 4.3 En çok oynanan 3 açılışın tahtada gösterimi

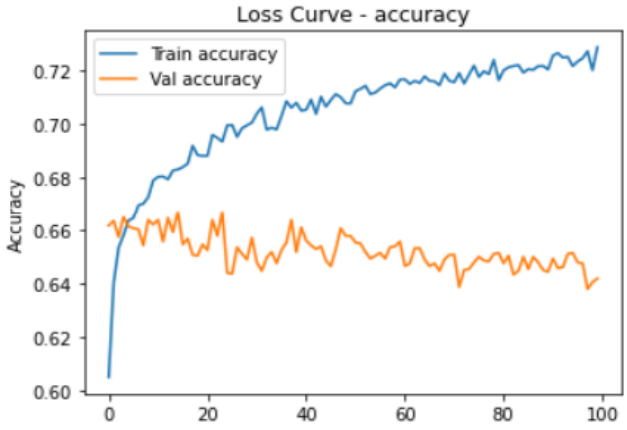
4.2 Sinir Ağı Modelleri için Sonuçlar

Görsel analizi yapılan veride kazanma oranlarının farklılığı ve farklı açılışların kullanımı göze çarpmakta olup çalışmaya uygun ve tahmin edilebilir olduğu görülmüştür. Model 1 ve Model 2'nin makine öğrenimi eğitimi yapılmış ve sonuçları incelenmiştir.



Şekil 4.4 Model 1 için eğitim ve test verisindeki doğruluk oranları grafiği

Model 1 için eğitim ve test verisi için tahminin doğruluk grafiği şekildeki gibi olup eğitim verisi için %66 test verisi için ise %65'e yakın olduğu görülmektedir.



Şekil 4.5 Model 2 için eğitim ve test verisindeki doğruluk oranları grafiği

Model 2 için 100 tekrarlı eğitim sonrasında ise modelin maç sonucunu eğitim verisi için %72 oranda doğru tahmin ettiği test verisi için ise aynı şekilde %65 civarında olduğu sonucuna varılmıştır.

```
In [22]: sample = {
    ...:     "rated": 0,
    ...:     "opening_ply": 5,
    ...:     "victory_status": "outoftime",
    ...:     "game_category" : "rapid",
    ...:     "opening_eco"  : "D10",
    ...:     "opening_name_det" : "Slav Defense",
    ...:     "white_rating" : 1500,
    ...:     "black_rating" : 1191,
    ...: }
    ...:
    ...: input_dict = {name: tf.convert_to_tensor([value]) for name, value in sample.items()}
    ...: predictions = model.predict(input_dict)
    ...:
    ...: print(
    ...:     "Bu Oyunda Beyaz Yüzde %.1f Olasılık İle Kazanacaktır." % (100 * predictions[0][0],)
    ...: )
1/1 [==============================] - 0s 23ms/step
Bu Oyunda Beyaz Yüzde 75.6 Olasılık İle Kazanacaktır.
```

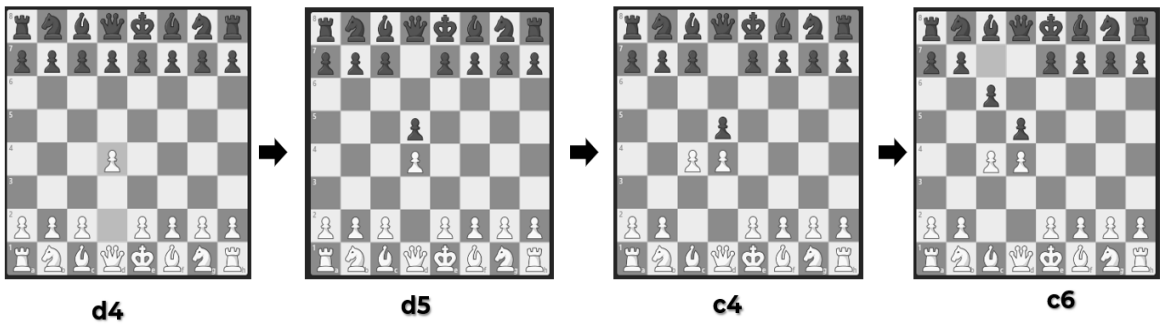
Şekil 4.6 Örnek maç için modelin öngörüsü

Kod çalıştırıldığında maç bilgilerini giren kullanıcı için maç sonucu tahmini yapılmaktadır.

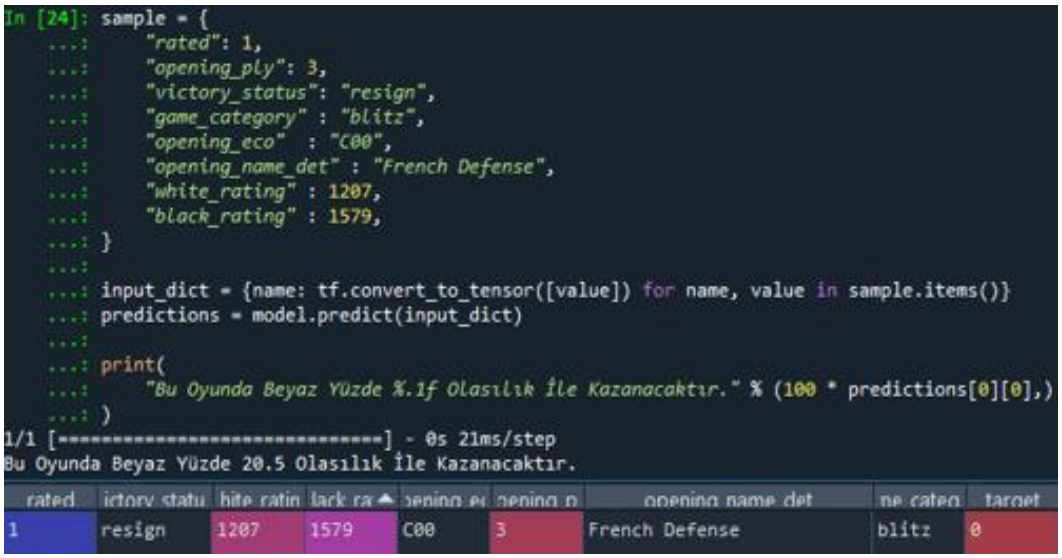
rated	victory statu	hite ratin	lack ratin	pening ec	pening p	opening name det	ne categ	target
0	outoftime	1500	1191	D10	5	Slav Defense	rapid	1

Şekil 4.7 Oynanan maçın gerçekte sonucu 1 beyazı temsil etmekte

Şekil 4.6'daki kodda tahmin fonksiyonuna kullanıcı Puansız, açılış hamle sayısının 5 olduğu süre sorunu ile biten bir maç için 1500 puanda olduğunu rakibin 1191 puanda olduğunu giriyor D10 açılış tipinde oynayacağını Slav Defansı açılışının olduğu bir maç için ve 10 dakika ve üzeri bir maç olduğu inputlarını giriyor, tahmin yapan model beyazın %75.6 olasılıkla kazanacağını öngörmektedir.Şekil 4.7'de oynanan maçın gerçek sonucu görülmektedir.

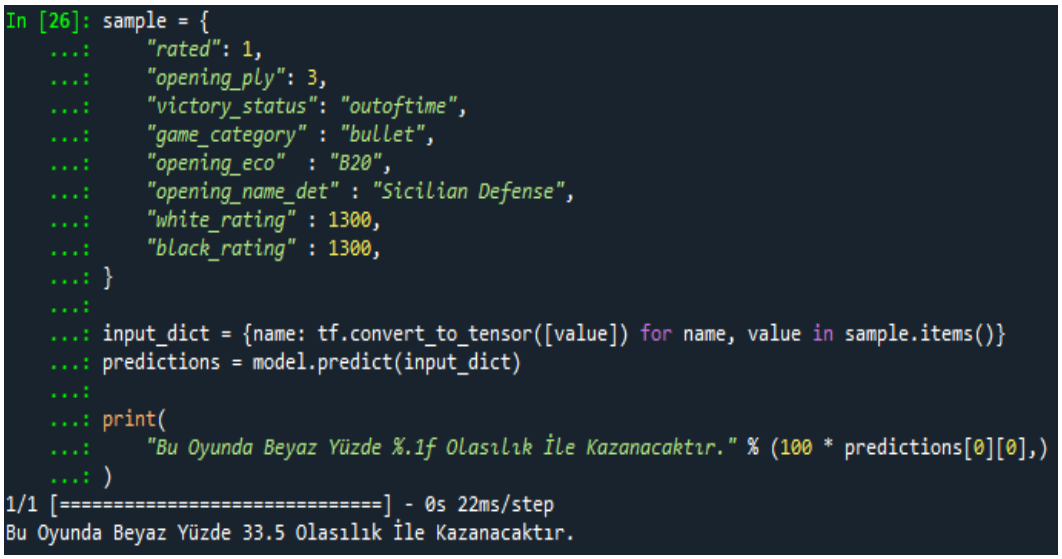


Şekil 4.8 Örnek maçın açılışı tahta üzerinde gösterimi



Şekil 4.9 Başka bir örnek maç için modelin tahmini

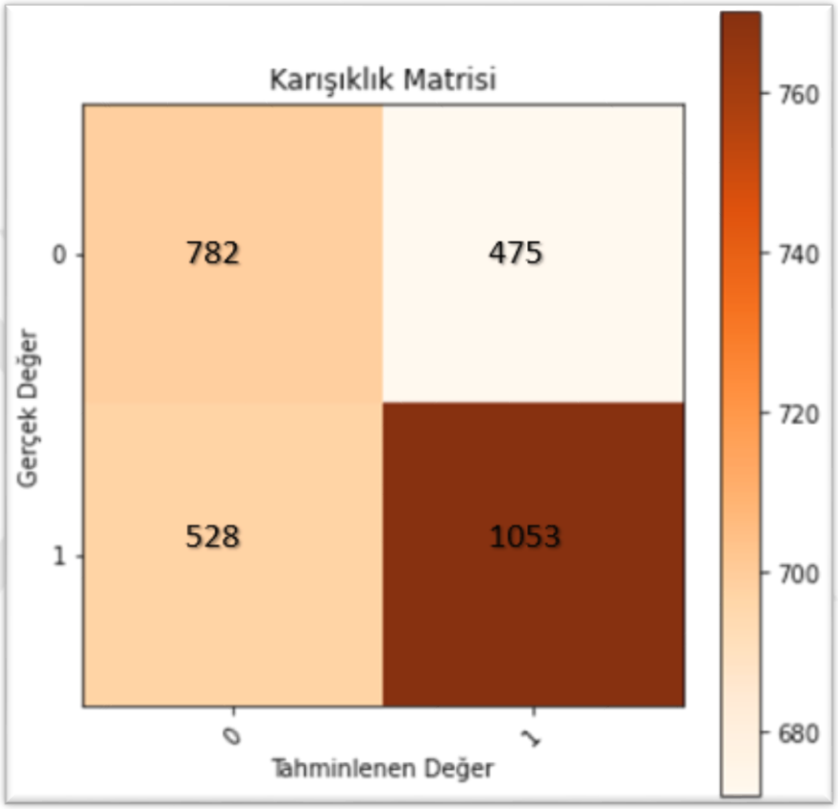
Siyahın puanının daha yüksek olduğu ve güçlü bir defans oyunu oynadığı başka bir maç için modelin performansına bakıldığında puanlı oynanan bu oyun için modelin tahmini %20 olasılıkla beyazın kazanacağı yönünde modelden berabere biten oyunlar çıkarıldığından %80 ihtimal ile siyahın kazanacağını tahmin etmektedir. Oyunun gerçekte sonucuna bakıldığında encode edilmiş haliyle 0 yani siyahın kazandığını görmekteyiz. Modelin tahminleri yaparken oyuncu puanını büyük oranda kullandığı fark edilmekte bu nedenle aynı puandaki oyuncuların maçları karşılaştırılarak açılışların sonuca etkisi daha net görülebilmektedir.



Şekil 4.10 Eşit puanlı iki oyuncu için modelin tahmini

Şekil 4.10'da kodu verilen oyunda, veri setinde olmayan fakat sezgisel olarak anlayabileceğimiz bir maç örneği için sonuca bakıldığında 3 dakikanın altında

oynanan eşit puanlı iki oyuncu için sonucu değiştirecek şey siyah oyuncunun kapalı bir defans oyunu olan Sicilya defansı oynadığını ve maçın süreyle biteceğini öngördüğümüzden zamandan kaybedilen bir oyun olarak seçildiğinde model siyahın %66.5 oranla kazanacağını öngörmektedir. Model kazananı sezgisel olarak tahmin edilen şekilde süreyle biten bir maç için siyahın kazacağını tahminlemkte bu açıdan uygulamanın başarılı bir performans verdiği söylenebilir.



Şekil 4.11 Model 2 için karışıklık matrisi

Şekil 4.11'de karışıklık matrisine bakıldığında modelin test verisindeki siyahın kazandığı 1257 maçtan 782'sini doğru bildiğini, Beyaz oyuncunun kazandığı 1581 maçtan 1053'ünü doğru tahmin ettiği sonucuna varılmaktadır.

Model her çalıştırıldığında tekrar tahminleme yapacağından bu sonuçlar değişecektir fakat oran olarak bakıldığında her zaman yaklaşık olarak %65 oranda doğru tahminlemektedir.

5
SONUÇ VE ÖNERİLER

M.S 600'lü yıllarda temelleri atılan satranç oyunu basit kuralları olan herkesin oynayabildiği bir oyundur. Oyun yüzyıllarca oynansa da üzerine yapılan çalışmalar 1900'lü yılların başında başlamıştır. Oyun oynayabilen makineler kurallardan oluştuğundan yapay zekanın ilgilendiği ilk alanlardandır. 1997 yılında IBM'in makinesinin o dönemin dünya şampiyonu Kasparov'u yenmesi ile yapay zekanın insanlığın önüne geçeceği hakkında birçok tartışma meydana gelmiştir. Matematiksel zemine oturtulan her durum için tahmin yapabilen makineler gelişen dünyanın en popüler konularından biridir.

Verilen kararların sonuçlarını doğru tahmin ettiği bir dünyada herkes kararlarını yapay zeka yardımıyla alabilirdi. Bu düşünce yapısını satranç oyunu ile birleştirerek çalışmaya başlanmıştır.

Çalışmanın amacı, derin öğrenme modeli ile oynanan verilerin girişi sonucunda kazanan tahmini yapmasıdır. Yapılan uygulama sonrası model, kazananı test verisi için %65 eğitim verisi için %72 oranla doğru tahmin etmektedir. Modelin farklı aktivasyon fonksiyonları ve daha büyük veri setleri ile çalışarak daha yüksek oranlı tahminler yapabileceği öngörülmektedir.

Satranç oyunu oynayan farklı yapay zekalar ve makine öğrenimi tipleri olsa da bu çalışma bir oyun için henüz oyun başında kazananı tahmin edebilen bir model olduğundan bu alanda özgünlük sağlıyor.

Özellikle ülkemizde satranç üzerine çalışma sayısı oldukça az olup genel olarak satranç oynayabilen modeller geliştirilmiştir. Bu alanda yapılan diğer çalışmalar incelendiğinde ise satranç oynayan derin öğrenme uygulamalarının sayısının genel anlamda literatürü oluşturduğu söylenebilmektedir. Son yıllarda satranç üzerine çalışma sayısının artış yaşadığı görülmektedir fakat oynayan sayısının yüz yıllardır bu seviyede olan bir spor ve oyun için çalışma sayısı nispeten az sayılabilmektedir.

Model için Lichess sitesinden 20 bin oyun üzerinden analiz yapılmış olup, veri seti makine öğrenimi ve derin ağlar için yüksek sayılabilecek bir sayı değildir. Bir başka siteden veya aynı site üzerinden indirilen ve düzenlenen daha büyük veri setleri ile yeni modeller kurulabilir. Büyük veri setleri ile beslenen yapay sinir ağlarının daha iyi sonuç vereceği tahminlenmektedir. Bu nedenle benzeri bir modelin 100 bin – 1 milyon aralığında veri seti ile kurulması önerilebilir.

Çalışmada farklı gizli katman ve Epoch ile denenen farklı modellerin farklı yüzdelikler ile doğruluk oranı verdiği görülmektedir. Bu nedenle optimal ara katman sayısı ve optimal aktivasyon fonksiyonu, optimizasyon yöntemi, hata fonksiyonu bulunması adına çeşitli yöntemler denenerek en doğru modelin kurulması üzerine çalışmalar yapılabilir.

Çalışmada kullanılan işlemci Intel(R) Core(TM) i7-4790 CPU @ 3.60GHz olup GPU için ise NVIDIA GeForce GT 1030 ekran kartı kullanılmıştır. Çalışma süresince makineye öğretim amaçlı 1 gün bırakılan ve bu süreçte doğruluk oranı %50 den %72'lere çıkan modelin daha uzun süreli ve güçlü bir donanım ile daha yüksek oranlarda tahminler yapacağı öngörülmektedir.

KAYNAKÇA

[1] S. Thrun, Learning to play the game of chess, 1994.

[2] R. McIlroy-Young, R. Wang, S. Sen , J. Kleinberg ve A. Anderson, Learning Personalized Models of Human Behavior in Chess, 2008.

[3] B. Oshri ve N. Khandwala, Predicting Moves in Chess using Convolutional Neural Networks, Stanford University, 2015.

[4] E. David, N. S. Netanyahu ve L. Wolf, DeepChess: End-to-End Deep Neural Network, 2017.

[5] F. Şekercioğlu, Minimaks algoritmasına dayalı bir satranç oyun yazılımı, 2012.

[6] N. A. Erilli, Bulanık sayıların bulanık kümeleme analizinde kullanımı ve satranç oyuncularının sınıflandırılması, 2012.

[7] M. Lai, Giraffe: Using Deep Reinforcement Learning to Play Chess, 2015.

[8] M. Şişeci Çeşmeli, Satranç oyunu durum uzayının sezgisel modellenmesi ve global optimizasyon yöntemleri ile çözümü, 2016.

[9] C. Koray, Bilgisayarla görü tabanlı satranç oyunu izleme sistemi, 2016.

[10] M. Türk, Eş zamanlı hamle tabanlı bir satranç oyun motoru geliştirmesi, 2016.

[11] M. Sabatelli, F. Bidoia, V. Codreanu ve M. Wiering, Learning to Evaluate Chess Positions with Deep Neural Networks and Limited Lookahead, 2018.

[12] F. Jankuloski ve A. Božinovski, Chess as Played by Artificial Intelligence, 2020.

[13] S. Maharaj, N. Polson ve A. Turk, Chess AI: Competing Paradigms for Machine Intelligence, Entropy, 2022.

[14] H. Bedük, Derin Öğrenme ile Histopatolojik Olarak Tanı Konmuş 65 Yaş Üstü Metastatik Meme Kanseri Hastalarında Kemoterapi Yan Etkisini Öngörme, 2019.

[15] A. Ö. Türkçetin, Akciğer Kanserinin Tespit Edilmesinde Derin Öğrenme Algoritmalarının Kullanılması, Isparta: Yüksek Lisans Tezi, 2019.

[16] I. C. Education, https://www.ibm.com/cloud/learn/machine-learning, 2020.

[17] UC Berkeley, https://ischoolonline.berkeley.edu/blog/what-is-machine-learning/, 2020.

[18] A. G. Barto ve R. S. Sutton, Reinforcement Learning: An Introduction, 2018.

[19] I. P. Pavlov, The work of the digestive glands, 1902.

[20] N. Aktaş, Makine öğrenmesi algoritmaları ile mali tablolarda hile analizi, 2022.

[21] Classification Algorithms - Decision Tree, https://www.tutorialspoint.com/machine_learning_with_python/classification_algorithms_decision_tree.htm, 2021.

[22] A. M. O. Anwer, Derin Öğrenme Yöntemleri İle Göğüs Kanseri Teşhisi, 2017.

[23] Ç. Elmas, Yapay Sinir Ağları, Seçkin Yayıncılık San ve Ticaret A.Ş, 2003.

[24] S. Haykin, Neural Networks: A Comprehensive Foundation, Second Edition, New Jersey, 1999.

[25] Multi-Layer Neural Network , www.techopedia.com/definition/33268/multi-layer-neural-network, 2018.

[26] C. C. Aggarwal, Neural Networks and Deep Learning A Textbook, 2018.

[27] E. Alpaydın, Yapay Öğrenme, İstanbul: Boğaziçi Üniversitesi Yayınevi, 2017.

[28] E. Öztemel, Yapay Sinir Ağları, İstanbul: Papatya Yayıncılık Eğitim, 2016.

[29] A. Geron, Hands-On Machine Learning with Scikit-Learn, Keras, and TensorFlow: Concepts, Tools, and Techniques to Build Intelligent Systems, 2019.

[30] M. C. Purucker, Neural network quarterbacking, 1996.

[31] L. Panneerselvam, https://www.analyticsvidhya.com/blog/2021/04/activation-functions-and-their-derivatives-a-quick-complete-guide/, Activation Functions and their Derivatives – A Quick & Complete Guide, 2021.

[32] J. Brownlee, https://machinelearningmastery.com/gradient-descent-with-nesterov-momentum-from-scratch/, Gradient Descent With Nesterov Momentum From Scratch, 2021.

[33] About Keras, https://keras.io/about/, 2020.

[34] D. Aksu, Veri Madenciliğinde Kullanılan Öğrenme Yöntemlerinin Farklı Koşullar Altında Karşılaştırılması, 2018.

[35] G. Yangın, XGBoost ve Karar Ağacı Tabanlı Algoritmaların Diyabet Veri Setleri Üzerine Uygulaması, 2019.

TEZDEN ÜRETİLMİŞ YAYINLAR

Konferans Bildirileri

F.D. Cellat, İkinci Uluslararası Dicle Bilimsel Araştırmalar ve İnovasyon Kongresi, Derin Öğrenmenin Satranç Oyunu Üzerine Uygulanması, 297-306, 2022.