YENİ NESİL YAPAY ZEKÂ YAKLAŞIMLARI

YAZILIM, KARİYER VE SİNİRSEL MİMARİLER ÜZERİNE

Editör
EYYÜP GÜLBANDILAR

BİDGE

BİDGE Yayınları

YENİ NESİL YAPAY ZEKÂ YAKLAŞIMLARI: YAZILIM, KARİYER VE SİNİRSEL MİMARİLER ÜZERİNE

Editör: EYYÜP GÜLBANDILAR

ISBN: -

1. Baskı
Sayfa Düzeni: Gözde YÜCEL
Yayınlama Tarihi: 2025-06-25
BİDGE Yayınları

Sertifika No: 71374
Yayın hakları © BİDGE Yayınları
www.bidgeyayinlari.com.tr - bidgeyayinlari@gmail.com
Krc Bilişim Ticaret ve Organizasyon Ltd. Şti.
Güzeltepe Mahallesi Abidin Daver Sokak Sefer Apartmanı No: 7/9 Çankaya / Ankara

ÖNSÖZ

Yapay zekâ (YZ), yalnızca teknik bir devrim değil; aynı zamanda iş yapış biçimlerinden bireysel kariyer planlamasına, eğitimden mühendislik yaklaşımlarına kadar pek çok alanda köklü bir dönüşümün öncüsüdür. Bu kitap, YZ'nin farklı boyutlarına odaklanarak okura hem teorik hem de uygulamalı bir bakış açısı kazandırmayı amaçlamaktadır.

Yapay zekânın yazılım geliştirme süreçlerine sağladığı katkılar, özellikle test güdümlü geliştirme, kod kalitesi, hata tespiti ve otomasyon alanlarında verimlilik artışı sağlamaktadır. Diğer yandan üretken yapay zekâ teknolojileri, dijital dönüşüm sürecini hızlandırmakta ve kariyer yapılarında köklü değişimlere yol açmaktadır. Yeni mesleklerin doğuşu, klasik rollerin evrimi ve bireylerin bu dönüşüme uyum sağlama biçimleri, bu değişimin insani ve toplumsal yönlerine işaret etmektedir.

Ayrıca yapay sinir ağlarının yapı taşlarını anlamak, hem akademik araştırmalar hem de uygulamalı projeler açısından büyük önem taşımaktadır. Bu mimarilerin işleyiş prensipleri, YZ'nin daha derin ve etkili şekilde kullanılabilmesinin önünü açmaktadır.

Bu kitap; yazılım geliştiricilerden mühendislik öğrencilerine, kariyerine yön vermek isteyen bireylerden araştırmacılara kadar geniş bir okuyucu kitlesine hitap etmektedir. Amacımız, yapay zekânın yalnızca teknik yönlerini değil, aynı zamanda birey ve toplum üzerindeki etkilerini de çok boyutlu bir yaklaşımla ortaya koymaktır.

Dijital geleceğe sağlam adımlarla yürümek isteyen herkese ilham vermesi dileğiyle…

Dr.Eyyüp GÜLBANDILAR

Editör

İÇİNDEKİLER

CHAPTER 0

Yapay Zeka ile Kombin Öneri Sistemi

1. Hazim İŞCAN[1]
2. Beyza Elif YENER[2]

1. Giriş

Yapay zeka (YZ), son yıllarda çok katmanlı öğrenme sistemlerinin gelişmesiyle birlikte, insan davranışlarını modelleme, görsel veri üzerinden analiz gerçekleştirme ve kişiselleştirilmiş önerilerde bulunma gibi görevlerde çarpıcı ilerlemeler göstermektedir. Özellikle derin öğrenme (deep learning), çok boyutlu ve karmaşık veri yapılarını otomatik olarak öğrenme kapasitesi sayesinde; görsel tanıma, doğal dil işleme, sağlık, otomotiv, güvenlik ve moda gibi çeşitli alanlarda uygulama imkânı bulmuştur (LeCun, Bengio & Hinton, 2015). Bu gelişmelerin etkisiyle, kullanıcı merkezli estetik tercihlere dayanan alanlarda da YZ tabanlı çözümler giderek yaygınlaşmaktadır. Moda endüstrisi bu bağlamda, hem büyük hacimli görsel veriye sahip olması hem de bireysel beğeni ve tarz unsurlarını taşıması nedeniyle YZ'nin önemli potansiyel taşıdığı disiplinlerden biri haline gelmiştir.

[1] Dr. Öğr. Üyesi, Konya Teknik Üniversitesi, Bilgisayar Mühendisliği, Orcid: 0000-0002-3698-3745
[2] Konya Teknik Üniversitesi, Bilgisayar Mühendisliği, Orcid: 0009-0001-2509-4547

Modern çevrim içi alışveriş sistemleri, genellikle kullanıcı geçmişi veya popüler ürün metriklerine dayalı öneri algoritmaları kullanmaktadır. Ancak bu sistemler, kullanıcıların görsel tercihlerini, mevcut gardıroplarını ya da bireysel giyim tarzlarını analiz etmekten uzaktır. Oysa giyim tercihi sadece estetik değil, aynı zamanda sosyal kimlik, kültürel etki ve bireysel ifade biçimidir. Dolayısıyla, kullanıcıya yalnızca popüler ürünleri değil, kişisel tarzına uygun alternatifleri sunabilecek öneri sistemleri, hem kullanıcı deneyimini artıracak hem de sektörel verimliliği yükseltecektir.

Bu gereksinim doğrultusunda geliştirilen YZ tabanlı giyim öneri sistemleri, özellikle derin öğrenmenin sunduğu görsel anlama yetenekleri sayesinde dikkat çekici bir araştırma alanı haline gelmiştir. Bu alandaki ilerlemelerin temelini, büyük ölçekli ve detaylı etiketlenmiş veri kümeleri oluşturmaktadır. Liu ve ark. (2016) tarafından sunulan DeepFashion veri kümesi, 800.000'den fazla görsel ile giyim kategorisi, detay, kilit nokta (landmark) ve ürün eşleştirme konularında geniş kapsamlı etiketli veri sağlayarak birçok araştırmaya zemin hazırlamıştır. Bu veri setinin daha da geliştirilmiş bir versiyonu olan DeepFashion2 (Ge ve ark., 2019) ise, giysi tespiti, poz tahmini, segmentasyon ve yeniden tanımlama gibi daha karmaşık görevler için çok daha detaylı etiketlemeler sunarak, bu çalışmada kullanılan veri setinin de temelini oluşturan önemli bir kaynak haline gelmiştir. Bu tür veri kümeleri üzerinde geliştirilen modeller, genellikle Evrişimli Sinir Ağları (CNN) gibi, görsel hiyerarşileri etkin bir şekilde öğrenebilen mimarilere dayanmaktadır. Özellikle He ve ark. (2016a) tarafından tanıtılan ResNet (Residual Network) mimarisi, artık bağlantılar (skip connections) sayesinde çok derin ağların başarılı bir şekilde eğitilmesini mümkün kılmış ve görüntü sınıflandırma, özellik çıkarımı gibi temel görevlerde bir standart oluşturarak moda analizi çalışmalarına da büyük katkı sağlamıştır.

Chen ve ark. (2012), kısıtlanmamış görsellerdeki giysiler için otomatik olarak semantik öznitelikler (renk, desen, yaka tipi vb.) üreten bir sistem sunmuştur. Kendi oluşturdukları 1856 görsellik ve 26 öznitelikli veri setini kullanarak, insan duruşu tahminine dayalı adaptif bir özellikle (SIFT, doku, renk) her öznitelik için SVM sınıflandırıcıları eğitmişlerdir. Çalışmalarının önemli bir yönü, öznitelikler arası ilişkileri modellemek ve tahminleri iyileştirmek için Koşullu Rastgele Alan (CRF) kullanmalarıdır. Sonuçlar, duruşa duyarlı özelliklerin ve CRF'nin performansı artırdığını göstermiş; sistemin, giyinme stili analizi ve giysilerden cinsiyet tahmini gibi yeni uygulamalara olanak tanıdığını ortaya koymuştur. Bu çalışma, giysilerin detaylı semantik analizi için bir temel oluşturmuştur.

Simo-Serra ve ark. (2015), bir kişinin fotoğrafında ne kadar "moda" veya "şık" (fashionable) göründüğünü tahmin etmeyi ve kullanıcının görünümünü iyileştirmek için öneriler sunmayı amaçlamıştır. Bu amaçla, moda odaklı sosyal web sitesi chictopia.com'dan topladıkları, 144.169 kullanıcı gönderisi içeren ve görsellerin yanı sıra etiketler, yorumlar ve gönderi oyları gibi zengin meta veriler barındıran Fashion144k adını verdikleri yeni bir veri seti oluşturmuşlardır. Metodolojileri, bir gönderinin moda düzeyini etkileyen çeşitli faktörleri (kıyafet türleri, kullanıcı tipi, fotoğrafın çekildiği ortam/sahne ve moda skoru) ortaklaşa değerlendiren bir Koşullu Rastgele Alan (Conditional Random Field - CRF) modeline dayanmaktadır. Bu CRF modeli, kullanıcının hayran sayısı, yüz öznitelikleri, sahne sınıflandırıcı çıktıları ve gönderi meta verileri gibi çeşitli özellikleri girdi olarak kullanır. Çalışmanın sonuçları, geliştirdikleri CRF modelinin, çeşitli temel yaklaşımlara kıyasla moda skorunu tahmin etmede daha iyi performans gösterdiğini ve modelin sadece skor tahmin etmekle kalmayıp, kullanıcıya hangi giysileri veya fotoğrafın arka planını değiştirmesi durumunda moda algısının nasıl iyileşeceğine dair eyleme geçirilebilir geri bildirimler sunabildiğini göstermiştir. Ayrıca, Fashion144k veri seti üzerinde

yaptıkları analizlerle farklı coğrafi bölgelerdeki ve zaman içindeki moda trendlerini ortaya koymuşlardır. Temel katkıları; moda skorunu tahmin etme ve iyileştirme önerileri sunma görevini tanımlamaları, bu görev için büyük ölçekli Fashion144k veri setini toplamaları ve çeşitli faktörleri ortaklaşa modelleyen bir CRF yaklaşımı geliştirmeleridir.

Simo-Serra ve ark. (2016), "Learning Features from Weakly-Supervised Data by Joint Ranking and Classification" başlıklı çalışmalarında, özellikle moda gibi alanlarda, internetten elde edilen ve genellikle gürültülü etiketlere sahip büyük miktardaki görsel veriden ("zayıf etiketli veri") etkili ve kompakt görsel özellikler öğrenmek amacıyla yeni bir yöntem sunmuştur. Temel yaklaşımları, bir özellik çıkarım ağını (görüntü üçlüleri ve sıralama kaybı kullanarak) ve bir sınıflandırma ağını (zayıf etiketlerle çapraz entropi kaybı kullanarak) ortaklaşa eğitmektir. Fashion144k veri setindeki kullanıcı etiketlerini zayıf denetim olarak kullanarak eğittikleri ve Hipster Wars veri setinde stil sınıflandırması için değerlendirdikleri bu yöntemle, elde ettikleri 128 boyutlu özelliklerin, ImageNet üzerinde önceden eğitilmiş çok daha büyük ve karmaşık CNN özelliklerinden ve mevcut en iyi stil tanımlayıcılarından belirgin şekilde daha iyi performans gösterdiğini kanıtlamışlardır. Çalışmalarının ana katkısı, zayıf denetimli verilerden yüksek kaliteli özellikler öğrenmek için bu ortak sıralama ve sınıflandırma çerçevesini önererek, özel alanlarda manuel etiketleme ihtiyacını azaltan, verimli ve üstün performanslı bir özellik öğrenme tekniği sunmaktır.

Zhang ve ark. (2020), giysi sınıflandırması ve öznitelik tanıma görevlerinde hem doku hem de şekil özelliklerinin etkin kullanımındaki zorluklara değinerek, bu iki özellik türünü hedefe yönelik ve ayrı ayrı güçlendirmek amacıyla "Doku ve Şekil Odaklı Moda Ağları" (TS-FashionNet) adını verdikleri iki akışlı bir derin öğrenme mimarisi sunmuşlardır. DeepFashion-C veri setini kullanan

çalışmalarında, bir akışın kilit noktalarla ortak öğrenme yoluyla şekil özelliklerine odaklandığını (şekil odaklı akış), diğer akışın ise ImageNet üzerinde önceden eğitilmiş bir modelin doku özelliklerini çıkarma yeteneğinden faydalandığını (doku odaklı akış) göstermişlerdir. Bu iki akıştan elde edilen özelliklerin birleştirilmesiyle oluşturulan TS-FashionNet, önceki en son teknoloji modellere kıyasla hem ilk-3 giysi kategorisi sınıflandırma doğruluğunu (%0.83 artış) hem de ilk-3 öznitelik tanıma geri çağırma oranını (%1.39 artış) önemli ölçüde iyileştirmiştir. Çalışma, kilit noktaların şekil özelliklerini, ImageNet ön eğitiminin ise doku özelliklerini öğrenmede etkili olduğunu ve bu iki özelliğin ayrı ayrı güçlendirilmesinin, her ikisini tek bir akışta birleştirmeye çalışmaktan daha üstün sonuçlar verdiğini ortaya koymuştur.

Veit ve ark. (2015), farklı kategorilerdeki giysi ve aksesuarlar arasında görsel bir uyumluluk kavramını öğrenerek, "Bu ayakkabıyla hangi kıyafet iyi gider?" gibi sorulara yanıt verebilen bir öğrenme çerçevesi sunmuştur. Bu amaçla, Amazon.com'dan elde edilen ve ürünlerin birlikte satın alınma bilgilerini uyumluluk ölçütü olarak kullanan büyük ölçekli bir veri seti üzerinde, birbiriyle uyumlu veya uyumsuz ürün çiftleriyle eğitilen bir Siyam Evrişimli Sinir Ağı (Siamese CNN) mimarisi kullanmışlardır. Özellikle, farklı üst düzey kategorilere ait ürünlerden oluşan "heterojen ikililer" (heterogeneous dyads) üzerinde stratejik bir örnekleme yöntemi uygulayarak kategoriler arası stil uyumunu öğrenmeyi hedeflemişlerdir. Çalışmalarının sonuçları, önerdikleri modelin görsel stil hakkında semantik bilgiler öğrenebildiğini, farklı kategorilerden ürünleri içeren uyumlu kıyafet kombinasyonları üretebildiğini ve bu konuda mevcut yaklaşımlardan daha iyi performans gösterdiğini ortaya koymuştur. Kullanıcı çalışmaları da modelin etkinliğini doğrulamış ve öğrenilen stil özelliklerinin daha önce görülmemiş giysi kategorilerine de aktarılabilir olduğunu göstermiştir.

Saranya ve Geetha (2022), Fashion-MNIST veri seti üzerinde moda ürünlerini sınıflandırmak amacıyla optimal bir Evrişimli Sinir Ağı (CNN) mimarisi geliştirmeyi ve bu süreçte CNN katmanlarını değiştirerek Adam ile RMSProp optimizasyon algoritmalarının performansını karşılaştırmayı hedeflemiştir. Araştırmacılar, üç evrişim katmanı, maksimum havuzlama, iki tam bağlı katman ve dropout içeren modifiye edilmiş bir CNN mimarisi önermişlerdir. Fashion-MNIST veri seti üzerinde yapılan deneylerde, önerdikleri bu mimarinin Adam optimizer ile kullanıldığında %92.68'lik bir test doğruluğuna ulaştığı ve bu sonucun, softmax fonksiyonlu standart CNN (%91.86) ve batch normalization kullanan CNN (%92.22) gibi temel modellere göre daha iyi olduğu gösterilmiştir.

Zhou ve ark. (2021), mevcut bir moda ürünü ve hedeflenen diğer ürünlerin referans maskeleri verildiğinde, bu ürünle uyumlu ve tamamlayıcı parçalardan oluşan bütün bir kıyafet kombinini sentezlemeyi amaçlayan OutfitGAN adında yeni bir üretken çerçeve sunmuştur. Bu amaçla, Polyvore.com'dan 20.000 kıyafet kombininden oluşan ve her kombinin dört temel parçasını (üst giyim, çanta, alt giyim, ayakkabı) ve maskelerini içeren OutfitSet adlı kendi büyük ölçekli veri setlerini oluşturmuşlardır. Önerdikleri OutfitGAN mimarisi, bir kombin üretici, bir kombin ayırıcı ve uyumluluğu denetleyen bir Çift Yönlü Uzun Kısa Süreli Bellek (Bi-LSTM) tabanlı Uyumluluk Sınıflandırma Modülü (CCM) içerir; üretici ise, verilen ürün ile hedeflenen ürünler arasındaki mekansal eşleşmeyi yakalamak için yenilikçi bir Semantik Uyum Modülü (SAM) kullanır. Kapsamlı deneyler, OutfitGAN'in, mevcut en son teknoloji görüntüden görüntüye çeviri yöntemlerine kıyasla, hem üretilen görüntülerin benzerlik ve gerçekliği hem de kombin uyumluluğu açısından üstün performans sergilediğini göstermiştir. Bu çalışma, mevcut bir ürüne dayanarak bütün ve uyumlu bir kıyafet kombinini sentezleme görevini ele alan ilk çalışmalardan biri olması ve bu

amaçla SAM ve CCM gibi yenilikçi modüller içeren OutfitGAN çerçevesini sunmasıyla alana önemli katkılar sağlamaktadır.

Hsiao ve Grauman (2018), bir giysi envanterinden maksimum sayıda birbiriyle uyumlu kombin oluşturabilecek minimum sayıda parçayı seçerek otomatik olarak "kapsül gardıroplar" oluşturmayı amaçlamıştır. Bu görevi, görsel uyumluluk, çok yönlülük ve kullanıcı tercihlerini içeren altmodüler amaç fonksiyonlarına sahip bir alt küme seçimi problemi olarak formüle etmişler ve bu problemi çözmek için yinelemeli bir optimizasyon algoritması geliştirmişlerdir. Çalışmalarının önemli bir katkısı, "vahşi doğada" çekilmiş tam vücut kombin fotoğraflarından (chictopia.com verisi) İlişkili Konu Modelleri (CTM) kullanarak denetimsiz bir şekilde görsel uyumluluk öğrenen ve bu bilgiyi katalog fotoğraflarına (Polyvore.com verisi) aktarabilen yeni bir model sunmalarıdır. Sonuçları, geliştirdikleri otomatik kapsül oluşturma yönteminin, moda uzmanlarının manuel yaklaşımlarını taklit edebildiğini, daha ölçeklenebilir olduğunu ve denetimsiz uyumluluk modellerinin mevcut denetimli yöntemlerden daha iyi performans gösterdiğini ortaya koymuştur.

He ve McAuley (2016), özellikle giyim gibi alanlarda, ürünler arasındaki sadece benzerliği değil, aynı zamanda tamamlayıcılık gibi karmaşık ve heterojen ilişkileri öğrenerek, kullanıcılara daha zengin öneriler sunmayı amaçlayan Monomer adlı yeni bir yöntem önermiştir. Geleneksel metrik öğrenme yaklaşımlarının sınırlamalarını aşmak için, Monomer, bir sorgu ürününü bir "referans uzayına" yansıtırken potansiyel eşleşen ürünleri birden fazla "sözde uzaya" yansıtarak ve bu uzayları bir uzmanlar karışımı (mixtures-of-experts) çerçevesiyle ağırlıklandırarak, metriklik varsayımını esnetir ve çoklu "ilişkililik" kavramlarını modelleyebilmektedir. Amazon.com'dan elde edilen büyük ölçekli birlikte satın alma ve birlikte görüntülenme verileri ile ImageNet üzerinde önceden eğitilmiş bir CNN'den çıkarılan görsel

özellikleri kullanarak yaptıkları deneylerde, Monomer'in, mevcut en son teknoloji metrik öğrenme tabanlı yöntemlere kıyasla, özellikle heterojen ürünler arasındaki uyumluluk tahmininde daha doğru sonuçlar verdiğini ve sadece benzerliğin ötesine geçen, çeşitli ve semantik olarak zengin öneriler üretebildiğini göstermişlerdir. Çalışmanın temel katkısı, heterojen ürünler arası öneriler için ölçeklenebilir, metriklik varsayımlarını esneten ve çoklu ilişkililik kavramlarını öğrenebilen Monomer yöntemini sunarak, daha karmaşık ve gerçek dünya senaryolarına uygun öneri sistemlerinin geliştirilmesine olanak tanımasıdır.

Kiapour ve ark. (2015), "Exact Street to Shop" adını verdikleri yeni bir görevi tanımlayarak, sokakta çekilmiş bir giysi fotoğrafındaki ürünün çevrimiçi bir mağazada bulunan birebir aynı ürünle eşleştirilmesini hedeflemiştir. Bu zorlu görev için, 400.000'den fazla mağaza fotoğrafı ve 20.000'den fazla sokak fotoğrafı içeren, 39.000'in üzerinde birebir eşleşme barındıran Exact Street2Shop Dataset adlı yeni bir veri seti oluşturmuşlardır. Metodolojileri, standart CNN özelliklerini kullanan temel geri getirme yaklaşımlarının yanı sıra, sokak ve mağaza görsellerinden elde edilen CNN özellikleri arasındaki benzerliği öğrenmek için üç katmanlı bir tam bağlı sinir ağına dayanan bir benzerlik öğrenme yaklaşımını içermektedir. Sonuçlar, öğrendikleri kategoriye özel ince ayarlanmış benzerlik metriğinin, özellikle ilk-20 geri getirme doğruluğunda, diğer temel yöntemlerden daha iyi performans gösterdiğini ortaya koymuştur.

Shirkhani ve ark. (2023), çalışmalarında moda alanının kendine özgü zorluklarına dikkat çeker. Bunların başında, ürünler arasında basit bir benzerlikten ziyade, estetik ve stilistik bir bütünlük ifade eden uyumluluğun (compatibility) çok daha kritik bir faktör olması gelir. Ayrıca, modanın doğası gereği öznel, kültürel ve sürekli değişen bir kavram olması, standart öneri algoritmalarının uygulanmasını zorlaştırır. Moda ürünlerinin genellikle yüksek

boyutlu ve karmaşık görsel özelliklere sahip olması da bir diğer önemli zorluktur. Yazarlar, bu bağlamda YZ'nin, özellikle derin öğrenmenin, bu zorlukların üstesinden gelmek için nasıl kullanıldığını inceler. Moda öneri sistemlerinde sıkça ele alınan temel kavramlar arasında ise stil (bir kıyafetin veya kişinin genel görünümünü tanımlayan estetik prensipler), estetik (görsel çekicilik ve güzellik algısı) ve kişiselleştirme (önerilerin bireysel kullanıcı zevklerine ve tercihlerine göre uyarlanması) bulunmaktadır. Derleme, YZ tekniklerinin bu karmaşık ve incelikli kavramları nasıl modellemeye çalıştığını ve bu sayede daha etkili moda önerileri sunmayı nasıl hedeflediğini ortaya koymaktadır.

Sun ve ark. (2021), Nijerya moda endüstrisinde yapay zeka (YZ) güdümlü kişiselleştirmenin (ürün önerileri, sanal deneme ve kişiselleştirilmiş stil danışmanlığı gibi) tüketici satın alma kararları ve müşteri memnuniyeti üzerindeki etkisini incelemiştir. Nijerya'daki dört önde gelen çevrimiçi moda perakende platformunun 200 müşterisiyle yapılan anket çalışması ve regresyon analizi sonucunda, YZ güdümlü kişiselleştirmenin hem satın alma kararlarını hem de müşteri memnuniyetini önemli ölçüde etkilediği bulunmuştur. Özellikle, ürün önerilerinin satın alma kararlarında kritik bir rol oynadığı, kişiselleştirilmiş stil danışmanlığının ise müşteri memnuniyetini en çok artıran faktör olduğu tespit edilmiştir; sanal deneme özelliklerinin de her iki değişkene pozitif katkı sağladığı, ancak etkisinin diğerlerine göre daha az olduğu görülmüştür. Çalışmanın temel katkısı, Nijerya gibi gelişmekte olan bir pazarda YZ güdümlü kişiselleştirmenin etkisine dair ampirik kanıtlar sunarak, yerel moda perakendecilerine ve politika yapıcılara önemli içgörüler sağlamasıdır.

Bu kapsamda yapılan çalışmalara rağmen, gerçek kullanıcı görsellerine dayalı, dinamik çalışan, kategori ve renk analizini birlikte kullanarak kombin öneren sistemler literatürde sınırlıdır. Birçok çalışma, yalnızca kıyafet türünü sınıflandırmakla yetinmekte

ya da hazır veri kümelerine dayalı statik analizler yapmaktadır. Oysa ki, kullanıcıların kendi kıyafetlerini sisteme yüklediği ve bu görsellerden çıkarılan görsel özniteliklerin gerçek zamanlı olarak analiz edildiği sistemler, kişiselleştirme düzeyinde niteliksel sıçrama yaratmaktadır.

Sonuç olarak, bu çalışma; YZ destekli kişiselleştirilmiş kombin öneri sistemleri bağlamında literatürdeki önemli bir boşluğu doldurmakta; görsel sınıflandırma, estetik analiz ve öneri üretimini bir araya getiren bütünsel bir yaklaşım sunmaktadır. Geliştirilen model, hem teorik düzeyde YZ ve moda entegrasyonuna katkı sağlamakta hem de mobil uygulamalara entegre edilebilirliğiyle pratik değer taşımaktadır.

2. Materyal ve Metot

2.1. Dataset

Moda odaklı yapay zeka araştırmalarında kullanılan veri setlerinin niteliği ve içeriği, geliştirilen modellerin performansı ve yetenekleri üzerinde doğrudan bir etkiye sahiptir. Bu bağlamda, Thushara Nair tarafından Kaggle platformunda "DeepFashion2 Original with DataFrames" adıyla sunulan ve orijinal DeepFashion2 (Ge et al., 2019) veri setini temel alan koleksiyonun bir parçası olan incelenen bu yapılandırılmış öznitelik tablosu, giyim ürünlerine ilişkin zengin ve yapılandırılmış bilgiler sunarak önemli bir kaynak teşkil etmektedir. Bu veri yapısı, 44.424 adet veri noktasından (ürün veya giysi örneği) oluşmakta ve her bir veri noktası, on farklı sütunda tanımlanan özelliklerle detaylandırılmaktadır. Söz konusu yapılandırılmış veri, özellikle giysi özniteliklerini anlama, stil analizi ve moda tavsiye sistemleri gibi uygulamalar için değerli bir temel oluşturur.

İncelenen bu veri setinin içeriği, genellikle her bir giysi ürünü veya örneği için bir satır ve bu ürüne ait farklı özellikleri temsil eden sütunlardan meydana gelir. Bu veri tablosundaki id sütunu, her bir

kaydı benzersiz bir şekilde tanımlar ve muhtemelen koleksiyonun diğer bileşenleriyle (örneğin, ürün görselleri veya daha detaylı görsel etiketlemeler) bağlantı kurmak için birincil anahtar işlevi görür. Ürünlerin demografik hedeflemesini belirten gender sütunu (örneğin, 'Men', 'Women' gibi değerlerle), kullanıcıya özel uygulamalarda önemli bir segmentasyon kriteridir. masterCategory (örneğin, 'Apparel', 'Accessories'), subCategory (örneğin, 'Topwear', 'Bottomwear') ve articleType (örneğin, 'Shirts', 'Jeans', 'Watches') sütunları ise ürünün türünü hiyerarşik bir yapıda tanımlayarak detaylı bir sınıflandırma imkanı sunar. Bu kategorik bilgiler, giysi tanıma ve sınıflandırma modellerinin eğitimi için temel girdiler olarak değerlendirilebilir.

Görsel estetiği ve stil uyumunu doğrudan etkileyen baseColour sütunu (örneğin, 'Blue', 'Silver'), ürünün ana rengini belirtirken, season (örneğin, 'Summer', 'Winter') ve year (örneğin, 2011.0 gibi bir değerle üretim veya satış yılı) sütunları, ürünlerin zamansal bağlamını ve trendlerle ilişkisini analiz etmek için kritik veriler sağlar. Özellikle dikkat çekici olan usage sütunu (örneğin, 'Casual', 'Party' gibi değerlerle kullanım amacı veya stili), giysilerin fonksiyonel ve stilistik bağlamını tanımlayarak, stil tahmini ve duruma uygun kombin önerileri geliştirmek için güçlü bir zemin sunar. Son olarak, productDisplayName sütunu, ürünün katalog adını metin formatında içerir; bu da doğal dil işleme teknikleriyle zenginleştirilebilecek veya ürünlerin daha spesifik olarak tanımlanmasına yardımcı olabilecek bir özelliktir.

Bu öznitelik koleksiyonu, giyim ürünlerinin tanımlayıcılarını içeren zengin ve çok boyutlu bir yapıya sahip olmasıyla dikkat çeker. Özellikle usage sütunu, stil tahmini ve analizi için önemli ipuçları sunarken; articleType, baseColour ve season gibi alanlar, bir modelin aynı anda birden fazla görevi (örneğin, hem ürün tipini hem de rengini ve mevsimini tahmin etme) öğrenmesini hedefleyen çok görevli öğrenme (multi-task learning) yaklaşımları için ideal bir

kombinasyon oluşturur. Bu tür yapılandırılmış öznitelik verileri, özellikle "yapay zeka destekli kombin uygulaması" geliştirme bağlamında, giysiler arası uyumluluğun modellenmesi, stil kurallarının öğrenilmesi ve kişiselleştirilmiş öneri mekanizmalarının oluşturulması için temel bir rol oynar. Bu tablodaki stil ve öznitelik bilgileri, muhtemelen Kaggle koleksiyonundaki diğer dosyalarda bulunan görsel verilerle (görüntüler, sınırlayıcı kutular, kilit noktalar vb.) birleştirildiğinde, moda ürünlerinin hem görsel hem de semantik özelliklerini derinlemesine anlayabilen kapsamlı yapay zeka çözümlerinin geliştirilmesine olanak tanır.

Bahsi geçen bu veri yapısı, Thushara Nair tarafından hazırlanan Kaggle koleksiyonunun yalnızca bir parçasıdır. Bu koleksiyonun tamamının, orijinal DeepFashion2'den (Ge et al., 2019) türetilen görüntü dosyalarını ve muhtemelen sınırlayıcı kutular, kilit noktalar ve segmentasyon bilgileri gibi daha detaylı görsel etiketlemeleri içeren başka veri çerçevelerini de barındırdığı düşünülmektedir. Bu yapıdaki stil ve öznitelik bilgileri, bu görsel etiketlemelerle birleştirildiğinde, giysilerin hem görsel hem de semantik özelliklerini kapsayan çok zengin bir veri kaynağı meydana getirir. Bu durum, sadece giysileri tanımakla kalmayıp, aynı zamanda onların stilini anlayan, birbiriyle nasıl eşleşeceğini tahmin edebilen ve hatta yeni moda trendlerini analiz edebilen daha sofistike yapay zeka modellerinin geliştirilmesine zemin hazırlar.

Sonuç olarak, incelenen bu öznitelik veri seti, kullanılan Kaggle koleksiyonunun önemli bir bileşeni olarak, giysi ürünlerine ait zengin stil ve öznitelik bilgilerini yapılandırılmış bir formatta sunmaktadır. Bu bilgiler, orijinal DeepFashion2'nin görsel zenginliğini tamamlayarak, özellikle moda anlayışı ve kombin önerisi gibi uygulamalar için derinlemesine analizler ve etkili modellemeler yapılmasına imkan tanır.

2.2. CNN

Evrişimli Sinir Ağları (CNN'ler), özellikle görüntü, video ve ses gibi grid benzeri topolojiye sahip verilerin işlenmesi ve analizi için tasarlanmış, derin öğrenme alanının temel taşlarından biri olan özelleşmiş sinir ağı mimarileridir (Goodfellow et al., 2016; O'Shea & Nash, 2015). Geleneksel tam bağlı sinir ağlarının aksine, CNN'ler girdinin mekansal ve zamansal hiyerarşilerini etkin bir şekilde yakalayarak, özellikle bilgisayarlı görü alanında, görüntü sınıflandırma (Krizhevsky et al., 2012), nesne tespiti, semantik segmentasyon (Long et al., 2015) ve daha birçok karmaşık görevde çığır açan başarılara imza atmıştır. CNN'lerin tasarımındaki temel ilham, memeli beynindeki görsel korteksin, görsel bilgiyi hiyerarşik bir şekilde işleyen ve yerel alıcı alanlara duyarlı nöronlardan oluşan yapısından gelmektedir (LeCun et al., 1998).

CNN'lerin başarısının ve etkinliğinin ardında yatan temel prensipler; seyrek etkileşimler (sparse interactions) yoluyla yerel alıcı alanların (local receptive fields) kullanılması, parametre paylaşımı (parameter sharing) sayesinde paylaşılan ağırlıkların (shared weights) etkinliği ve bu özellikler sonucunda ortaya çıkan eşdeğişken temsillerdir (equivariant representations) (Goodfellow et al., 2016). LeCun ve ark. (1998) tarafından LeNet-5 mimarisi ile somutlaştırılan bu prensipler, ağın öğrenmesi gereken parametre sayısını çarpıcı bir şekilde azaltırken, modelin öznitelikleri konumdan bağımsız olarak öğrenmesine ve böylece genelleme kabiliyetinin artmasına olanak tanır. Ağın katmanları boyunca ilerledikçe, basit yerel özelliklerden (kenarlar, köşeler gibi) başlayarak giderek daha karmaşık ve soyut özelliklerin (nesne parçaları, nesnelerin kendisi gibi) hiyerarşik bir şekilde öğrenildiği gözlemlenir (LeCun et al., 1998). Bu hiyerarşik özellik öğrenme süreci, Zeiler ve Fergus (2014) tarafından "Deconvolutional Network" gibi görselleştirme teknikleriyle daha da aydınlatılmış,

ağın farklı katmanlarında ne tür temsillerin öğrenildiği somut bir şekilde ortaya konmuştur.

Tipik bir CNN mimarisi, ardışık olarak düzenlenmiş temel yapı taşlarından oluşur. Bu yapılar;

Evrişim Katmanı (Convolutional Layer): CNN'lerin kalbi olan bu katman, girdi verisi üzerinde bir veya daha fazla öğrenilebilir filtreyi (kernel) kaydırarak evrişim işlemini gerçekleştirir (LeCun et al., 1998). Her filtre, belirli bir görsel örüntüyü (örneğin, dikey bir çizgi, belirli bir doku) tespit etmek üzere tasarlanmıştır ve bu filtrenin tüm girdi üzerinde uygulanmasıyla bir özellik haritası (feature map) oluşturulur (Goodfellow et al., 2016). Simonyan ve Zisserman (2014), VGGNet mimarisinde gösterdikleri gibi, büyük filtreler yerine ardışık olarak kullanılan çok sayıda küçük (örn: 3x3) filtrenin, daha az parametre ile daha derin ve etkili ağlar oluşturulabileceğini ortaya koymuştur. Ayrıca, Szegedy ve ark. (2015) tarafından GoogLeNet (Inception) mimarisinde sunulan 1x1 evrişimler, özellik haritalarının derinliğini azaltmak (boyut indirgeme) veya farklı özellik haritalarını birleştirmek için etkin bir araç olarak mimarilere dahil edilmiştir.

Aktivasyon Fonksiyonu: Evrişim katmanının lineer çıktısı, genellikle Rectified Linear Unit (ReLU) gibi doğrusal olmayan bir aktivasyon fonksiyonundan geçirilir (Krizhevsky et al., 2012). ReLU, gradyanların daha etkin yayılmasına yardımcı olarak derin ağların eğitimini kolaylaştırır ve ağa daha karmaşık ilişkileri modelleme yeteneği kazandırır.

Havuzlama Katmanı (Pooling Layer): Genellikle evrişim ve aktivasyon katmanlarını takiben kullanılan havuzlama katmanları (örn: maksimum havuzlama, ortalama havuzlama), özellik haritalarının boyutsalını azaltarak (alt örnekleme) hesaplama yükünü hafifletir ve özelliklerin konumundaki küçük değişikliklere karşı bir

miktar değişmezlik (invariance) sağlar (LeCun et al., 1998; Goodfellow et al., 2016). Bu, modelin genelleme kapasitesini artırır.

Tam Bağlı Katman (Fully Connected Layer): CNN mimarilerinin sonlarına doğru, evrişim ve havuzlama katmanları tarafından çıkarılan üst düzey özellikler, genellikle bir veya daha fazla tam bağlı katmana beslenir. Bu katmanlar, öğrenilen bu özellikler üzerinden nihai sınıflandırma veya regresyon görevini gerçekleştirir (LeCun et al., 1998; O'Shea & Nash, 2015). Ancak, Long ve ark. (2015) tarafından geliştirilen Tam Evrişimli Ağlar (Fully Convolutional Networks - FCN), semantik segmentasyon gibi yoğun tahmin (dense prediction) görevleri için tam bağlı katmanlara ihtiyaç duyulmadığını göstermiş, bunun yerine bu katmanları da evrişimsel hale getirerek uçtan uca (end-to-end) piksel düzeyinde tahminler yapılabilmesini sağlamıştır.

CNN mimarilerinin evrimi, LeNet-5 (LeCun et al., 1998) gibi öncü çalışmalarla başlamış, AlexNet'in (Krizhevsky et al., 2012) ImageNet yarışmasındaki çarpıcı başarısıyla derin öğrenme ve CNN'ler ana akım haline gelmiştir. AlexNet, daha derin ağların, ReLU aktivasyonlarının, dropout gibi düzenlileştirme tekniklerinin ve GPU hızlandırmasının gücünü göstermiştir. Bunu takiben, VGGNet (Simonyan & Zisserman, 2014) daha da derin ve basit tasarımların etkinliğini kanıtlarken, GoogLeNet (Szegedy et al., 2015) "Inception modülü" ile ağ içinde farklı ölçeklerdeki özellikleri paralel olarak işleyebilen ve hesaplama açısından daha verimli mimariler sunmuştur. Bu gelişmeler, ağların sadece daha derin değil, aynı zamanda daha akıllıca tasarlanması yönünde bir eğilimi de beraberinde getirmiştir.

Sonuç olarak, Evrişimli Sinir Ağları, yerel alıcı alanlar, parametre paylaşımı ve hiyerarşik özellik öğrenme gibi biyolojik esinlenmelere dayanan güçlü prensipler üzerine inşa edilmiştir. Zaman içinde LeNet'ten AlexNet'e, VGG'den GoogLeNet'e ve FCN gibi daha görev özelleşmiş mimarilere doğru evrilen CNN'ler, sadece

görüntü sınıflandırmasında değil, aynı zamanda nesne tespiti, semantik segmentasyon, doğal dil işleme ve hatta oyun oynama gibi çok çeşitli alanlarda temel bir teknoloji haline gelmiştir. Sürekli gelişen mimarileri, artan anlama ve yorumlama çabaları ve genişleyen uygulama yelpazesi ile CNN'ler, yapay zeka alanındaki ilerlemenin itici güçlerinden biri olmaya devam etmektedir.

2.3. Resnet

Derin Evrişimli Sinir Ağları (CNN'ler), yapay zeka alanında, özellikle bilgisayarlı görü görevlerinde, öğrenilebilir özellik hiyerarşileri aracılığıyla karmaşık örüntüleri modelleme yetenekleriyle bir devrim yaratmıştır. Ağ derinliğinin, modelin temsil kapasitesini artırmada kritik bir faktör olduğu genel kabul görse de, geleneksel derin ağların eğitimi, belirli bir derinliğin ötesinde "bozulma" (degradation) olarak bilinen bir sorunla gölgelenmiştir: Daha derin ağlar, hem eğitim hem de test aşamalarında daha sığ benzerlerinden daha kötü performans göstermeye başlamıştır (He et al., 2016a). Bu zorluğun üstesinden gelmek amacıyla Kaiming He ve ark. (2016a), ResNet (Residual Network) mimarisini ve temelindeki "derin artık öğrenme" (deep residual learning) çerçevesini sunarak, yüzlerce, hatta binlerce katmanlı ağların etkili bir şekilde eğitilmesinin önünü açmış ve derin öğrenme mimarilerinin tasarımında yeni bir çağ başlatmıştır. ResNet'in sunduğu bu çığır açan yaklaşım, sonrasında mimarinin kendisinin ve altında yatan prensiplerin daha iyi anlaşılmasına, iyileştirilmesine ve çeşitlendirilmesine yönelik yoğun bir araştırma faaliyetini tetiklemiştir.

Ağ derinliği arttıkça, teorik olarak modelin daha karmaşık fonksiyonları öğrenebilmesi beklenirken, pratikte çok derin "düz" (plain) ağların optimizasyonu zorlaşmakta ve eğitim hatası belirli bir derinlikten sonra artmaya başlamaktadır. Modelin kapasitesinin yetersizliğinden veya aşırı öğrenmeden ziyade, derin yapıların etkin

bir şekilde optimize edilememesinden kaynaklanan bu "bozulma" problemi (He et al., 2016a), derinliğin faydalarından tam olarak yararlanmanın önünde ciddi bir engel teşkil etmektedir. Bu önemli zorluğa bir çözüm olarak ResNet'in getirdiği temel fikir, katmanların doğrudan bir hedef eşlemesi öğrenmek yerine, bir "artık eşlemesi" (residual mapping) öğrenmesinin daha kolay olduğu varsayımına dayanır. Bu durumda, istenen asıl eşleme H(x)=F(x)+x şeklinde ifade edilir. Bu formülasyon, "atlama bağlantıları" (skip/shortcut connections) aracılığıyla mimariye entegre edilir: Bir veya daha fazla katmanın girdisi (x), bu katmanların çıktısına (F(x)) doğrudan eklenir. He ve ark. (2016a), eğer birim eşleme (identity mapping) optimal ise, artık fonksiyonu F(x)'i sıfıra yakınsatmanın, bir dizi doğrusal olmayan katmanla doğrudan birim eşlemesini öğrenmekten çok daha kolay olduğunu öne sürmüştür.

Bu artık öğrenme fikrinin başarısı, orijinal ResNet makalesindeki (He et al., 2016a) temel artık biriminin daha da geliştirilmesine yönelik çalışmaları beraberinde getirmiştir. Aynı yazarlar, "Identity Mappings in Deep Residual Networks" (He et al., 2016b) başlıklı çalışmalarında bu birimleri daha da analiz etmiş ve iyileştirmişlerdir. Artık birim içindeki sinyal yayılımı ve gradyan akışını detaylı bir şekilde inceleyerek "ön-aktivasyon" (pre-activation) adı verilen bir tasarım önermişlerdir. Bu yeni düzende, Batch Normalization (BN) ve ReLU aktivasyonu, evrişim katmanlarından önce uygulanır. Bu değişikliğin, bilgi akışını daha pürüzsüz hale getirerek ve gradyanların engellenmeden yayılmasını sağlayarak optimizasyonu kolaylaştırdığı ve genelleme performansını artırdığı gösterilmiştir.

ResNet'lerin ve iyileştirilmiş artık bloklarının bu olağanüstü başarısı, doğal olarak altında yatan mekanizmaların daha derinlemesine incelenmesini teşvik etmiştir. Bu mekanizmalardan biri, Veit ve ark. (2016) tarafından öne sürülen topluluk (ensemble) davranışıdır; bu çalışmaya göre ResNet'ler, atlama bağlantıları

sayesinde ağ içinde eksponansiyel sayıda potansiyel yol oluşturarak örtük olarak çok sayıda daha sığ ağın bir topluluğu gibi davranır, bu da modelin sağlamlığını ve genelleme yeteneğini artırır. Bir diğer önemli faktör, atlama bağlantılarının gradyanların eğitim sırasında daha derin katmanlardan geriye doğru daha kolay akmasını sağlayarak kaybolan gradyan sorununu hafifletmesi ve optimizasyon sürecini basitleştirmesidir (He et al., 2016a; He et al., 2016b). Ayrıca, Huang ve ark. (2016) tarafından önerilen Stokastik Derinlik (Stochastic Depth) tekniği –eğitim sırasında ResNet bloklarının rastgele olarak atlanması– hem bir tür örtük topluluk eğitimi etkisi yaratır hem de güçlü bir düzenlileştirme sağlayarak daha derin ağların eğitilmesine olanak tanır ve ResNet'lerdeki yolların birçoğunun yedekli olabileceği fikrini destekler.

ResNet'in sunduğu esneklik ve başarı, sadece derinlik boyutuyla sınırlı kalmamış, mimari tasarımında farklı boyutların da keşfedilmesine yol açmıştır. Örneğin, Zagoruyko ve Komodakis (2016) tarafından sunulan Geniş ResNet'ler (Wide Residual Networks), artık blokların derinliğini artırmak yerine genişliğini (evrişim katmanlarındaki filtre sayısı) artırmanın, benzer veya daha iyi performansla daha verimli modeller elde etmeyi sağlayabileceğini göstermiş ve aşırı derinliğe bir alternatif olarak genişliğin önemini vurgulamıştır. Benzer şekilde, Xie ve ark. (2017) tarafından geliştirilen ResNeXt (Aggregated Residual Transformations) mimarisi, ResNet bloklarını "kardinalite" adı verilen yeni bir boyutla zenginleştirmiştir. Bu yaklaşımda, artık blok içindeki dönüşümler, paralel ve özdeş topolojiye sahip birden fazla küçük gruba ("yol") ayrılır ve bu yolların çıktıları toplanır; bu "ayrık-dönüştür-birleştir" stratejisi, model kapasitesini derinlik veya genişlik artışına kıyasla daha etkin bir şekilde artırmanın bir yolunu sunmuştur.

ResNet'in temelindeki artık öğrenme prensibinin gücü ve genelliği, onun diğer başarılı derin öğrenme mimarileriyle de

başarıyla entegre edilebilmesini sağlamıştır. Szegedy ve ark. (2017), Inception mimarisini artık bağlantılarla birleştirerek Inception-ResNet hibrit modellerini oluşturmuş ve artık bağlantıların Inception modüllerinin eğitimini önemli ölçüde hızlandırdığını, basitleştirdiğini ve genel performansını artırdığını göstermiştir. Bu durum, artık bağlantıların farklı mimari paradigmalarla sinerji oluşturabileceğini ve genel bir iyileştirme tekniği olarak kullanılabileceğini kanıtlamıştır.

Tüm bu gelişmeler ve entegrasyonlar, ResNet ve temelini oluşturan artık öğrenme ilkesinin derin sinir ağları alanındaki derin etkisini açıkça ortaya koymaktadır. Atlama bağlantıları, çok derin ağların optimizasyonunu mümkün kılarak, daha önce ulaşılamaz kabul edilen performans seviyelerine ulaşılmasını sağlamıştır. Bu başarı, sadece görüntü sınıflandırmasında değil, nesne tespiti, semantik segmentasyon, yüz tanıma ve hatta doğal dil işleme gibi birçok farklı alanda sayısız uygulamaya ve yeni mimari geliştirmesine ilham kaynağı olmuştur. Sonuç olarak ResNet, derin öğrenme tarihinde bir kilometre taşıdır. Başlangıçta çok derin ağların eğitimindeki "bozulma" sorununu çözmek için artık öğrenme ve atlama bağlantıları gibi zarif çözümler sunmuş, sonrasında ise ön-aktivasyonlu tasarımlar, genişlik ve kardinalite gibi yeni boyutların keşfi ve stokastik derinlik gibi eğitim stratejileriyle daha da geliştirilmiştir. ResNet'lerin örtük topluluk davranışı gibi teorik analizler, başarısının altında yatan nedenlere ışık tutarken, Inception-ResNet gibi hibrit modeller, artık bağlantıların evrensel bir iyileştirme prensibi olarak potansiyelini göstermiştir. Derin öğrenmenin sınırlarını zorlamaya devam eden ResNet ve türevleri, yapay zeka alanındaki ilerlemenin temelini oluşturan ve etkisini uzun yıllar sürdürecek olan bir mimari paradigma olarak yerini sağlamlaştırmıştır.

2.4. Resnet50

Derin Evrişimli Sinir Ağları (CNN'ler), bilgisayarlı görü alanında çığır açan gelişmelere öncülük etmiş, ancak ağ derinliği arttıkça karşılaşılan optimizasyon zorlukları ve "bozulma" (degradation) problemi, daha derin ve dolayısıyla daha yetenekli modellerin geliştirilmesinin önünde bir engel teşkil etmiştir. He ve ark. (2016a) tarafından sunulan ResNet (Residual Network) mimarisi ve temelindeki "derin artık öğrenme" (deep residual learning) çerçevesi, bu engeli aşmak için devrim niteliğinde bir çözüm sunmuştur. Bu çerçevenin en popüler ve yaygın olarak kullanılan uygulamalarından biri olan ResNet-50, elli katmanlı yapısıyla hem yüksek performans sergilemiş hem de çok derin ağların eğitilebilirliğini kanıtlayarak alanında bir standart haline gelmiştir. Bu metin, ResNet-50'nin mimari yapısını, temelini oluşturan artık öğrenme prensiplerini ve bu prensiplerin daha geniş bir araştırma bağlamındaki etkilerini inceleyecektir.

ResNet-50, adından da anlaşılacağı gibi, 50 ağırlıklı katmandan oluşan bir derin evrişimli sinir ağıdır (He et al., 2016a). Temel yapısı, bir başlangıç evrişim ve havuzlama katmanını takiben yığınlanmış artık blokları, ardından global ortalama havuzlama (global average pooling) ve sınıflandırma için bir tam bağlı katmandan oluşur. ResNet-50 ve diğer derin ResNet varyantları (ResNet-101, ResNet-152 gibi) için kritik bir mimari tercih, hesaplama verimliliğini artıran "darboğaz" (bottleneck) tasarımına sahip artık blokların kullanılmasıdır. Bir darboğaz artık bloğu tipik olarak üç evrişim katmanından oluşur: önce kanal sayısını azaltan bir 1x1 evrişim katmanı, ardından daha az sayıda kanalda çalışan bir 3x3 evrişim katmanı ve son olarak kanal sayısını tekrar orijinaline artıran bir 1x1 evrişim katmanı. Bu üç katmanın çıktısına, bloğun girdisi bir "atlama bağlantısı" (skip connection) aracılığıyla eklenir (F(x)+x). Bu yapı, daha az parametre ve hesaplama ile benzer bir temsil gücü sağlayarak çok derin ağların oluşturulmasını pratik hale

getirir (He et al., 2016a). ResNet-50'de bu darboğaz blokları, özellik haritası boyutlarının küçültüldüğü ve kanal sayılarının artırıldığı farklı "aşamalar" (stages) halinde gruplandırılmıştır. Boyut küçültme genellikle bir aşamanın ilk bloğundaki evrişim katmanlarından birinde adım (stride) değerinin 2 olarak ayarlanmasıyla gerçekleştirilir.

Bu karmaşık ve derin mimarinin etkin bir şekilde çalışmasını sağlayan temel ilke ise ResNet-50'nin kalbinde yatan, He ve ark. (2016a) tarafından önerilen artık öğrenme prensibidir. Katmanların doğrudan bir hedef eşlemesi öğrenmek yerine bir artık eşlemesi öğrenmesi, özellikle birim eşlemenin (identity mapping) optimal olduğu durumlarda optimizasyonu büyük ölçüde kolaylaştırır. Atlama bağlantıları, gradyanların eğitim sırasında daha derin katmanlara etkili bir şekilde yayılmasına olanak tanır. Bu temel konsept, He ve ark. (2016b) tarafından "Identity Mappings in Deep Residual Networks" başlıklı çalışmada daha da incelenmiş ve iyileştirilmiştir. Bu çalışmada önerilen "ön-aktivasyon" (pre-activation) düzeni –Batch Normalization ve ReLU'nun evrişim katmanlarından önce uygulanması– sinyal yayılımını ve gradyan akışını daha da pürüzsüz hale getirerek ResNet-50 gibi mimarilerin eğitimini kolaylaştırır ve genelleme performansını artırır.

ResNet-50'nin bu etkileyici performansı ve genel olarak ResNet mimarilerinin başarısı, sadece derinliğin etkin bir şekilde yönetilmesinden kaynaklanmaz; altında yatan çeşitli mekanizmalar bu başarıya katkıda bulunur. Veit ve ark. (2015), ResNet'lerin atlama bağlantıları sayesinde örtük olarak çok sayıda farklı derinlikteki daha sığ ağın bir topluluğu (ensemble) gibi davrandığını öne sürmüştür; bu "topluluk etkisi", modelin sağlamlığını ve genelleme yeteneğini artırır. Benzer şekilde, Huang ve ark. (2016) tarafından önerilen Stokastik Derinlik (Stochastic Depth) tekniği, eğitim sırasında ResNet bloklarının rastgele atlanmasıyla bu topluluk etkisini güçlendirir, güçlü bir düzenlileştirme sağlar ve çok derin

ağların eğitimini daha da kolaylaştırır. ResNet-50'nin başarısı, mimari tasarımında derinliğin ötesinde başka boyutların da keşfedilmesine ilham vermiştir. Zagoruyko ve Komodakis (2016), "Geniş ResNet'ler" (Wide Residual Networks) ile artık blokların genişliğini artırmanın, aşırı derinliğe kıyasla daha iyi veya benzer performans sağlayabileceğini göstermiştir. Öte yandan, Xie ve ark. (2017) tarafından sunulan ResNeXt mimarisi, artık bloklara "kardinalite" (paralel dönüşüm yollarının sayısı) adı verilen yeni bir boyut ekleyerek, model kapasitesini artırmada derinlik ve genişliğe göre daha etkin bir yol sunmuştur. Bu çalışmalar, ResNet-50'nin temelini oluşturan artık öğrenme prensibinin farklı tasarım tercihleriyle nasıl daha da geliştirilebileceğini göstermiştir. Ayrıca, ResNet'in artık öğrenme konseptinin evrenselliği, Szegedy ve ark. (2017) tarafından Inception mimarisiyle birleştirilerek Inception-ResNet gibi hibrit modellerin oluşturulmasıyla da kanıtlanmıştır. Bu entegrasyon, artık bağlantıların farklı ve güçlü mimarilerin eğitimini ve performansını iyileştirebileceğini ortaya koymuştur.

Sonuç olarak, ResNet-50, derin artık öğrenme çerçevesinin somut ve son derece başarılı bir uygulaması olarak, yapay zeka ve özellikle bilgisayarlı görü alanında bir dönüm noktası temsil eder. Darboğaz tasarımı gibi verimli mimari seçimleriyle elli katmanlı bir derinliğe ulaşarak, çok derin ağların eğitilebilirliğini ve üstün performansını kanıtlamıştır. Orijinal ResNet makalesinde (He et al., 2016a) detaylandırılan yapısı, sonrasında gelen iyileştirmeler (He et al., 2016b), teorik analizler (Veit et al., 2015) ve mimari çeşitlenmeler (Zagoruyko & Komodakis, 2016; Xie et al., 2017; Szegedy et al., 2017) ile birlikte düşünüldüğünde, ResNet-50 sadece bir mimari olmanın ötesinde, derin öğrenmede yeni bir tasarım felsefesinin öncüsü olmuştur. Günümüzde birçok farklı görev ve veri seti için standart bir omurga mimarisi olarak yaygın şekilde kullanılmaya devam etmekte ve derin öğrenme alanındaki ilerlemelere ilham vermektedir.

2.5. Resnet18

Derin Evrişimli Sinir Ağları (CNN'ler) alanındaki en etkili yeniliklerden biri olan Derin Artık Öğrenme (Deep Residual Learning) çerçevesi, Kaiming He ve ark. (2016a) tarafından sunulan ResNet (Residual Network) mimarileriyle somutlaşmıştır. Bu mimari ailesi, çok derin ağların eğitiminde karşılaşılan "bozulma" (degradation) problemini, artık bloklar (residual blocks) ve atlama bağlantıları (skip connections) kullanarak başarılı bir şekilde çözmüştür. ResNet ailesi içinde, farklı derinlik ve karmaşıklık seviyelerine sahip çeşitli modeller bulunmaktadır; bunlar arasında ResNet-18, daha sığ ancak birçok görev için oldukça etkili ve verimli bir seçenek olarak öne çıkar. ResNet-18, toplamda 18 ağırlıklı katmana (evrişim ve tam bağlı katmanlar) sahip olup, ResNet-50 gibi daha derin varyantlarla aynı temel artık öğrenme prensibini paylaşır, ancak yapı taşı olan artık blokların tasarımında farklılık gösterir.

ResNet-18'in temel mimari yapı taşı, "temel artık blok" (basic residual block) olarak adlandırılır. Bu temel blok, ResNet-50 ve daha derin modellerde kullanılan "darboğaz" (bottleneck) tasarımından daha basittir. Bir temel artık bloğu, tipik olarak her biri 3x3 boyutunda filtrelere sahip iki ardışık evrişim katmanından oluşur. Bu iki evrişim katmanının çıktısına (F(x)), bloğun girdisi (x) bir atlama bağlantısı aracılığıyla doğrudan eklenir (F(x)+x). Eğer girdi ve çıktı boyutları arasında bir uyuşmazlık varsa (örneğin, özellik haritası boyutlarının küçültülmesi veya kanal sayısının artırılması gerektiğinde), atlama bağlantısı bu boyutları eşleştirmek için uygun bir projeksiyon (genellikle 1x1 evrişim) kullanabilir; ancak boyutlar aynı olduğunda, kimlik eşlemeli (identity shortcut) atlama bağlantıları tercih edilir ve bu, parametre eklemeden ve ek hesaplama yükü getirmeden bilgi akışını sağlar (He et al., 2016a). Her evrişim katmanından sonra genellikle bir Batch Normalization katmanı ve bir ReLU aktivasyon fonksiyonu uygulanır; ancak He ve

ark. (2016b) tarafından önerilen "ön-aktivasyonlu" (pre-activation) ResNet varyantlarında bu sıralama değişebilir.

ResNet-18'in genel mimarisi, bir başlangıç evrişim katmanı ve bir maksimum havuzlama (max pooling) katmanıyla başlar. Bunu, farklı sayılarda temel artık blok içeren dört ana aşama (stage) takip eder. Her bir aşamada, özellik haritalarının mekansal boyutu genellikle yarıya indirilirken (örneğin, aşamanın ilk bloğundaki ilk evrişim katmanında adım (stride) değeri 2 olarak ayarlanarak) kanal sayısı iki katına çıkarılır. Bu yapı, ağın hiyerarşik olarak giderek daha karmaşık özellikleri öğrenmesine olanak tanır. Dört aşamanın ardından, bir global ortalama havuzlama (global average pooling) katmanı ve son olarak sınıflandırma görevi için bir tam bağlı katman (softmax aktivasyonlu) gelir. Toplamda, 17 evrişim katmanı ve 1 tam bağlı katman olmak üzere 18 ağırlıklı katman içerir.

ResNet-18, daha derin kardeşleri olan ResNet-34 (yine temel blokları kullanan), ResNet-50, ResNet-101 ve ResNet-152 (darboğaz bloklarını kullanan) ile karşılaştırıldığında daha az parametreye ve daha düşük hesaplama karmaşıklığına sahiptir. Bu özellikler, onu özellikle hesaplama kaynaklarının kısıtlı olduğu durumlar, daha küçük veri setleri veya daha hızlı çıkarım (inference) süresi gerektiren uygulamalar için cazip bir seçenek haline getirir. Temel blokların kullanımı, darboğaz bloklarına kıyasla daha az karmaşık bir yapı sunarken, artık öğrenme prensibi sayesinde hala etkili bir şekilde derinlikten faydalanabilir ve tatmin edici performanslar elde edebilir. He ve ark. (2016a), ImageNet gibi büyük ölçekli veri setlerinde ResNet-18 ve ResNet-34'ün, benzer derinlikteki "düz" (plain) ağlara kıyasla önemli ölçüde daha iyi sonuçlar verdiğini ve daha kolay optimize edildiğini göstermiştir.

Sonuç olarak, ResNet-18, derin artık öğrenme çerçevesinin temel prensiplerini (artık eşlemeler ve atlama bağlantıları) daha basit "temel artık bloklar" kullanarak uygulayan, verimli ve etkili bir evrişimli sinir ağı mimarisidir. Daha derin ResNet varyantlarına göre

daha az karmaşık olmasına rağmen, birçok bilgisayarlı görü görevinde güçlü bir performans sergileme potansiyeline sahiptir ve transfer öğrenme için popüler bir omurga (backbone) mimarisi olarak yaygın şekilde kullanılmaktadır. ResNet-18'in başarısı, artık öğrenmenin sadece aşırı derin ağlar için değil, orta derinlikteki ağlar için de optimizasyonu kolaylaştırdığını ve performansı artırdığını göstermektedir.

2.6. ADAM Algoritması

Derin sinir ağlarının etkin bir şekilde eğitilmesi, modern yapay zeka uygulamalarının temelini oluşturmakta ve bu süreçte optimizasyon algoritmaları merkezi bir rol oynamaktadır. Bu algoritmaların amacı, genellikle yüksek boyutlu ve konveks olmayan kayıp fonksiyonlarını minimize ederek model parametrelerini optimal değerlere ulaştırmaktır. Stokastik Gradyan İnişi (SGD) ve onun momentum gibi uzantıları uzun yıllar temel yaklaşımlar olmuşsa da, özellikle derin ve karmaşık ağların eğitiminde karşılaşılan zorluklar öğrenme oranının hassasiyeti, farklı parametreler için farklı güncelleme ihtiyaçları ve seyrek gradyanlarla başa çıkma gibi daha gelişmiş ve adaptif yöntemlere olan ihtiyacı ortaya koymuştur. Bu ihtiyaca yönelik olarak Diederik P. Kingma ve Jimmy Ba (2014), Adam (Adaptive Moment Estimation) adını verdikleri, birinci dereceden gradyanlara dayalı, verimli ve yaygın olarak kullanılan bir stokastik optimizasyon yöntemi geliştirmişlerdir. Adam, özellikle büyük ölçekli veri ve parametre içeren derin öğrenme modellerinin eğitiminde standart bir tercih haline gelmiştir.

Adam'ın geliştirilmesi, kendisinden önceki adaptif öğrenme oranı yöntemlerinin evrimsel bir devamı niteliğindedir. SGD'nin yavaş yakınsaması ve salınımlarını azaltmak için Momentum gibi teknikler geliştirilmiş, ardından AdaGrad gibi her bir parametre için öğrenme oranını ayrı ayrı adapte eden yöntemler ortaya çıkmıştır.

AdaGrad, özellikle seyrek özelliklerle çalışırken etkili olsa da, zamanla öğrenme oranının aşırı derecede küçülerek öğrenmeyi durdurması gibi bir dezavantaja sahipti. RMSProp, AdaGrad'in bu sorununu, gradyanların karesinin üstel olarak azalan bir hareketli ortalamasını kullanarak hafifletmiştir (Ruder, 2016; Goodfellow et al., 2016). Adam, bu iki güçlü fikri –yani momentumun hareketli ortalamasını ve RMSProp'un adaptif öğrenme oranı ölçeklemesini bir araya getirerek daha da ileri götürmüştür. Temelde, gradyanların hem birinci momentini (ortalama, momentum terimi gibi düşünülebilir) hem de ikinci momentini (merkezlenmemiş varyans) üstel olarak azalan hareketli ortalamalarla tahmin eder (Kingma & Ba, 2014).

Kingma ve Ba (2014) tarafından tanımlanan Adam algoritmasında, t zaman adımındaki parametreler θ_t ve gradyan $g_t=\nabla_\theta L_t$ olmak üzere, birinci moment vektörü m_t (gradyanların ortalamasının tahmini) ve ikinci moment vektörü v_t (gradyanların karesinin ortalamasının tahmini) Denklem 1 ve Denklem 2 ile güncellenir:

$$m_t = \beta_1 \, m_{(t-1)} + (1 - \beta_1) \, g_t \tag{1}$$

$$v_t = \beta_2 \, v_{(t-1)} + (1 - \beta_2) \, g_t^2 \tag{2}$$

Burada β_1 ve β_2, sırasıyla birinci ve ikinci momentler için unutma faktörleri olan üstel azalma oranlarıdır (tipik değerler sırasıyla 0.9 ve 0.999'dur). m_t ve v_t başlangıçta sıfır olarak ilklendirildiğinden, özellikle eğitimin ilk aşamalarında sıfıra doğru yanlı (biased) olma eğilimindedirler. Bu başlangıç yanlılığını düzeltmek için Adam, moment tahminlerini Denklem 3 ve Denklem 4 ile düzeltir:

$$\hat{m}_t = m_t/(1 - \beta_1^t) \tag{3}$$

$$\hat{v}_t = v_t/(1 - \beta_2^t) \tag{4}$$

Bu düzeltilmiş, yanlılıktan arındırılmış moment tahminleri kullanılarak nihai parametre güncelleme kuralı Denklem 5 ile verilir:

$$\theta_{(t+1)} = \theta_t - \alpha/(\sqrt{\hat{v}_t} + \epsilon)\hat{m}_t \quad (5)$$

Bu kuralda α öğrenme oranını (adım büyüklüğü), ϵ ise paydada sıfıra bölünmeyi engellemek ve sayısal kararlılığı sağlamak için eklenen çok küçük bir sabiti (genellikle 10−8) temsil eder.

Adam optimizasyon algoritmasının birçok avantajı bulunmaktadır. Hesaplama açısından verimlidir, her bir iterasyonda az miktarda ek hesaplama gerektirir ve bellek kullanımı düşüktür, bu da onu büyük modeller ve veri setleri için uygun kılar. Gradyanların köşegen yeniden ölçeklenmesine karşı değişmez olması, özelliklerin farklı büyüklüklerde olduğu durumlarda bile iyi çalışmasını sağlar. Ayrıca, gürültülü veya seyrek gradyanlarla başa çıkmada etkilidir ve genellikle hiperparametrelerinin ayarlanması SGD'ye göre daha kolaydır; varsayılan ayarları birçok farklı problem için iyi bir başlangıç noktası sunar (Kingma & Ba, 2014). Bu özellikler, Adam'ı derin öğrenme uygulayıcıları için popüler bir seçenek haline getirmiştir.

Bununla birlikte, Adam'ın performansı ve genelleme yetenekleri üzerine akademik literatürde tartışmalar da mevcuttur. Örneğin, Wilson ve ark. (2017), bazı durumlarda dikkatlice ayarlanmış momentumlu SGD'nin Adam gibi adaptif yöntemlerden daha iyi genelleme yapabileceğini öne sürmüştür. Ayrıca, Reddi ve ark. (2018), Adam'ın bazı teorik durumlarda yakınsamayabileceğini göstermiş ve bu sorunu çözmek için AMSGrad gibi varyantlar önermiştir. Adam'ın bir diğer eleştirilen yönü, ağırlık düşüşü (weight decay) düzenlileştirmesinin L2 düzenlileştirmesi ile tam olarak eşdeğer olmamasıdır; bu sorunu gidermek için Loshchilov ve Hutter (2017), ağırlık düşüşünü öğrenme oranından ayrıştıran AdamW algoritmasını geliştirmiştir. Bu eleştirilere ve önerilen varyantlara rağmen, Adam'ın temel versiyonu hala birçok uygulamada güçlü bir

performans sergilemekte ve derin öğrenme optimizasyonunda önemli bir araç olarak kabul edilmektedir. Sonuç olarak, Adam optimizasyon algoritması, gradyanların birinci ve ikinci momentlerini adaptif olarak tahmin ederek her parametre için bireysel öğrenme oranları belirleyen sofistike ve etkili bir yöntemdir. Derin öğrenme modellerinin eğitimini önemli ölçüde kolaylaştırmış, alandaki hızlı ilerlemeye katkıda bulunmuş ve optimizasyon algoritmaları üzerine yapılan araştırmalar için bir referans noktası oluşturmuştur.

3. Uygulama

Bu çalışmada geliştirilen yapay zeka destekli kombin öneri sisteminin deneysel altyapısı, iki farklı derin öğrenme modelinin tasarlanması, eğitilmesi ve bu modellerin çıktılarının birleştirilmesi üzerine kurulmuştur. Sistemin temel amacı, giysi görsellerinden hem temel ürün özelliklerini (kategori ve renk) hem de daha soyut olan stil bilgisini çıkararak, kullanıcıya bütüncül ve stilistik olarak tutarlı kombinler sunmaktır.

Uygulamanın temelini oluşturan veri seti, Thushara Nair tarafından Kaggle platformunda sunulan ve orijinal DeepFashion2 (Ge et al., 2019) veri setini temel alan "DeepFashion2 Original with DataFrames" koleksiyonundan türetilmiştir.

Veri hazırlık sürecinde, hem stil tahmini hem de kategori-renk tahmini modellerinde kullanılmak üzere, tüm görseller için ortak bir ön işleme (preprocessing) süreci uygulanmıştır. Öncelikle veri bütünlüğünü sağlamak amacıyla, referans verilen ancak fiziksel olarak mevcut olmayan görsel dosyalarına ait kayıtlar veri kümesinden ayıklanmıştır. Ayrıca stil sınıflandırması görevinde, veri kümesinde çok az sayıda bulunan Party, Smart Casual ve Travel gibi bazı kullanım amaçları (usage) veri bütünlüğünü olumsuz etkileyebileceği gerekçesiyle çalışma kapsamı dışında bırakılmıştır. Bu sayede, eğitim sürecinde dengeli ve temsil gücü yüksek bir etiket

dağılımı sağlanması hedeflenmiştir. Tüm görseller, evrişimli sinir ağı (CNN) mimarisinin gerektirdiği sabit giriş boyutuna uyacak şekilde 224x224 piksele yeniden boyutlandırılmıştır. Görseller, PyTorch tarafından işlenebilir hale getirilmek için tensör formatına dönüştürülmüştür. Bu adım ayrıca, piksel değerlerini [0, 255] aralığından [0.0, 1.0] aralığına normalize etmektedir. Görseller, daha önce ImageNet veri kümesi üzerinde hesaplanmış ortalama (mean: [0.485, 0.456, 0.406]) ve standart sapma (std: [0.229, 0.224, 0.225]) değerleri kullanılarak normalize edilmiştir. Bu işlem, modeli önceden eğitilmiş (pretrained) ağlarla uyumlu hale getirmek için önemlidir.

Veri kümesi her iki model için de, eğitim (%70), doğrulama (%15) ve test (%15) olmak üzere üç alt kümeye orantılı (stratified) şekilde ayrılmış ve her alt kümede etiket dağılımının dengeli kalmasına dikkat edilmiştir.

Modelin öğrenmesi hedeflenen kategorik çıktılar olan ürün tipi (articleType), ana renk (baseColour) ve stil (usage) etiketleri için, veri kümesindeki tüm benzersiz değerler tespit edilerek sayısal etiket haritaları oluşturulmuş ve bu haritalar, modelin eğitimi ve sonraki çıkarım süreçlerinde tutarlılık sağlamak üzere kaydedilmiştir.

Model geliştirme sürecinde, hem kategori ve renk tahmini hem de stil sınıflandırması görevleri için transfer öğrenme yaklaşımı benimsenmiş ve ImageNet üzerinde önceden eğitilmiş ResNet tabanlı mimariler tercih edilmiştir. Kategori ve renk tahmini için geliştirilen model, CategoryColorModel adı verilen çok görevli bir sinir ağı mimarisine sahiptir. Bu model, özellik çıkarıcı olarak derin yapısıyla bilinen ResNet-50 mimarisini kullanmaktadır. ResNet-50, toplamda elli ağırlıklı katmandan oluşan ve her biri "darboğaz" (bottleneck) yapısına sahip artık blokları barındıran bir mimaridir. Bu darboğaz blokları, kanal boyutlarını düzenlemek ve hesaplama verimliliğini artırmak amacıyla sırasıyla 1x1, 3x3 ve tekrar 1x1

evrişim katmanları kullanarak daha derin bir öğrenme süreci sağlar. CategoryColorModel içerisinde, ResNet-50'nin orijinal tam bağlı katmanı çıkarılmış ve yerine özellik vektörünü doğrudan çıkış katmanlarına ileten bir kimlik eşlemesi (nn.Identity()) yerleştirilmiştir. Böylece elde edilen 2048 boyutundaki özellik vektörü, iki ayrı sınıflandırma başlığına (head) yönlendirilmiştir. Kategori başlığı, ReLU aktivasyonlu ve %50 dropout oranına sahip 1024 nöronlu bir katman ile ürün tipi tahminini yaparken; benzer yapıya sahip renk başlığı da aynı vektörü kullanarak ana renk sınıflandırmasını gerçekleştirmektedir.

Stil tahmini için ise daha hafif bir yapı olan ResNet-18 mimarisi tercih edilmiştir. StyleClassifier adı verilen bu modelde, temel olarak yine ImageNet üzerinde önceden eğitilmiş bir ağdan yararlanılmış; ancak ResNet-50'nin aksine, daha az sayıda ağırlıklı katmana (18) sahip olan bu mimari, daha düşük hesaplama maliyetiyle çalışmaktadır. ResNet-18'in temel yapı taşı, "temel blok" (basic block) olarak adlandırılan ve her biri iki adet 3x3 evrişim katmanı içeren daha sade yapılardır. Bu bloklar, ResNet-50'deki gibi kanal boyutu değiştiren 1x1 evrişimler içermez. StyleClassifier modelinde, bu temel bloklardan elde edilen özellikler, yeniden yapılandırılmış bir çıkış katmanına iletilmiş; ResNet-18'in orijinal tam bağlı katmanı çıkarılarak yerine stil sınıfı sayısına (beş sınıf) göre özelleştirilmiş bir tam bağlı katman yerleştirilmiştir. Böylece model, görsellerden elde edilen özellikleri doğrudan stil sınıflandırması için kullanabilir hale gelmiştir.

Her iki model de PyTorch kütüphanesi kullanılarak GPU ve CPU uyumlu şekilde geliştirilmiş; eğitim sürecinde ise her iki ağ için de Adam optimizasyon algoritması tercih edilmiştir. Bu tercih, adaptif öğrenme oranı özelliği sayesinde derin öğrenme modellerinin daha kararlı ve hızlı bir şekilde eğitilmesine olanak tanımaktadır. Modellerin her biri için öğrenme oranı 0.0005 olarak belirlenmiş ve sınıflandırma görevlerinin doğasına uygun biçimde

çapraz entropi (CrossEntropyLoss) kayıp fonksiyonu kullanılmıştır. CategoryColorModel'da, toplam kayıp değeri, kategori ve renk tahminlerinden ayrı ayrı elde edilen kayıpların toplamı olarak hesaplanmış ve böylece her iki görevin eş zamanlı öğrenilmesi sağlanmıştır. Eğitim süreci boyunca CategoryColorModel ve StyleClassifier 20 epoch boyunca eğitilmiştir. Her bir epoch sonunda modellerin doğruluk performansları doğrulama ve test kümeleri üzerinde değerlendirilmiştir.

Geliştirilen bu iki model –kategori ve renk tahmini yapan CategoryColorModel ile stil sınıflandırması yapan StyleClassifier– yapay zeka destekli kombin öneri uygulamasının temelini oluşturmaktadır. Uygulamanın öngörülen işleyişinde, kullanıcıdan alınan bir giysi görseli, tanımlanan ön işleme adımlarından geçirilerek her iki modele de girdi olarak sunulur. CategoryColorModel görselden giysinin kategorisini ve ana rengini çıkarırken, StyleClassifier giysinin genel stilini belirler. Bu üç temel özellik (kategori, renk ve stil), daha sonra geliştirilecek olan bir kombinasyon mantığı modülüne aktarılacaktır. Bu modülün, elde edilen bu çıkarımlarından gelen ek bilgileri kullanarak, stilistik olarak uyumlu ve bağlamsal olarak anlamlı kombinasyon önerileri üretmesi hedeflenmektedir. Dolayısıyla, bu çalışmada detaylandırılan iki model, kombin uygulamasının giysileri çok boyutlu bir perspektiften anlamlandırması ve bu anlayışı kullanıcıya faydalı önerilere dönüştürmesi sürecinde birbirini tamamlayan ve temel teşkil eden bileşenler olarak kritik bir rol oynamaktadır.

4. Test Sonuçları

Modellerin eğitim süreci tamamlandıktan sonra, her biri ilgili test kümesi üzerinde değerlendirilmiş ve sınıflandırma performansları çeşitli metrikler kullanılarak analiz edilmiştir. Bu kapsamda, doğruluk, hassasiyet, duyarlılık, F1 skoru ve test kaybı gibi ölçütler Tablo 1 ve Tablo 2'de sunulmuştur.

Tablo 1. Kategori ve Renk Modeli Test Sonuçları

Metrik	**Kategori Tahmini**	**Renk Tahmini**
Doğrulama Doğruluğu (Val Acc)	86.76%	65.96%
Test Doğruluğu (Accuracy)	87.99%	66.37%
Test Kaybı (Loss)	2.3061	2.3061
Hassasiyet (Precision)	63.69%	39.22%
Duyarlılık (Recall)	60.08%	35.97%
F1 Skoru	60.32%	36.26%

Bu sonuçlar, kategori ve renk sınıflandırmasını aynı anda gerçekleştiren modelin test veri setinde tatmin edici bir performans gösterdiğini ortaya koymaktadır. Kategori tahmini görevinde %87.99 doğruluk oranına ulaşılırken, renk tahmini görevi %66.37 doğruluk ile sonuçlanmıştır. Ortak kayıp değeri olan 2.3061, modelin her iki görevi aynı anda optimize ederken istikrarlı bir şekilde öğrenme sağladığını göstermektedir.

Tablo 2. Stil Modeli Test Sonuçları

Metrik	Sonuç
Doğrulama Doğruluğu (Validation Accuracy)	91.00%
Test Kaybı (Loss)	0.4240
Doğruluk (Accuracy)	91.22%
Hassasiyet (Precision)	86.75%
Duyarlılık (Recall)	86.04%
F1 Skoru	86.37%

Bu sonuçlar, stil sınıflandırması görevine yönelik geliştirilen modelin test veri seti üzerinde yüksek düzeyde bir başarı sergilediğini göstermektedir. Model, %91.22 doğruluk oranına ulaşarak görsel giyim ögeleri üzerinden stil ayrımı yapma konusunda oldukça etkili olduğunu ortaya koymuştur. Düşük test kaybı (0.424) ile birlikte hassasiyet (%86.75), duyarlılık (%86.04) ve F1 skoru (%86.37) gibi diğer performans metriklerinin de yüksek olması, modelin stil etiketlerini ayırt etme konusundaki genelleme yeteneğini desteklemektedir.

3. Analiz

Bu çalışmada geliştirilen iki ayrı model, farklı görevler üzerinde değerlendirilerek sınıflandırma performansları açısından karşılaştırılmıştır. İlk model olan CategoryColorModel, aynı anda hem ürün tipi (articleType) hem de ana renk (baseColour) tahmini yapabilen çok görevli bir sinir ağı mimarisi olarak tasarlanmıştır. İkinci model olan StyleClassifier ise görsel giysi örneklerinden stil (usage) sınıfını tahmin etmeyi amaçlayan tek görevli bir sınıflandırıcıdır. Her iki model de transfer öğrenme yaklaşımıyla geliştirilmiş ve önceden ImageNet üzerinde eğitilmiş ResNet tabanlı mimariler kullanılmıştır.

CategoryColorModel, kategori tahmini görevinde %87.99 doğruluk, %63.69 hassasiyet, %60.08 duyarlılık ve %60.32 F1 skoru ile tatmin edici bir performans göstermiştir. Renk tahmini görevinde ise modelin doğruluğu %66.37'ye düşerken, diğer metrikler daha sınırlı düzeyde kalmıştır. Bu durum, renk sınıflarının görsel olarak birbiriyle daha fazla benzerlik göstermesi, verideki dağılım dengesizlikleri veya sınıflandırma güçlüğü gibi faktörlerle açıklanabilir. Her iki görev için ortak hesaplanan test kaybı değeri 2.3061 olarak ölçülmüş olup, modelin her iki görevi eş zamanlı olarak öğrenme sürecinde makul bir öğrenme başarısı elde ettiği söylenebilir.

StyleClassifier modeli ise daha yalın bir mimari olan ResNet-18 tabanlı yapısıyla, stil sınıflandırmasında %91.22 doğruluk, %86.75 hassasiyet, %86.04 duyarlılık ve %86.37 F1 skoru ile oldukça yüksek bir genel başarı ortaya koymuştur. Bu sonuçlar, stil etiketlerinin daha belirgin görsel ayrımlara sahip olduğunu ve modelin bu farkları etkili şekilde öğrenebildiğini göstermektedir. Modelin test kaybı değeri 0.4240 olarak ölçülmüş, bu da eğitim sürecinde kararlı bir öğrenme gerçekleştiğini ve modelin aşırı öğrenmeden kaçındığını göstermektedir.

Genel olarak değerlendirildiğinde, çok görevli CategoryColorModel karmaşık yapısı nedeniyle farklı görevler arasında denge kurmak zorunda kalmış, bu durum özellikle renk tahmini performansına yansımıştır. Buna karşılık, StyleClassifier'ın tek hedefe odaklanmış ve sadeleştirilmiş mimarisi, stil sınıflandırmasında daha yüksek başarıya ulaşmasını sağlamıştır. Bu durum, tek görevli modellerin belirli alanlarda daha güçlü sonuçlar verebileceğini; ancak çok görevli modellerin genişletilebilirlik açısından daha esnek ve bütüncül bir çözüm sunduğunu ortaya koymaktadır.

3. Tartışma ve Sonuç

Bu çalışmada, görsel giysi verilerini kullanarak giysi türü, ana renk ve stil sınıflarını tahmin edebilen derin öğrenme tabanlı sınıflandırma modelleri geliştirilmiştir. Kategori ve renk tahmini görevleri için çok görevli bir yapı olan CategoryColorModel, stil sınıflandırması için ise daha sade bir mimariye sahip olan StyleClassifier modeli tasarlanmış ve değerlendirilmiştir. Her iki model de transfer öğrenme yaklaşımından yararlanarak, ImageNet üzerinde önceden eğitilmiş ResNet mimarileri üzerine inşa edilmiştir.

Test sonuçlarına göre, StyleClassifier modeli stil sınıflandırmasında yüksek doğruluk ve genel başarı metrikleriyle

öne çıkmıştır. %91.22 doğruluk oranı ve 0.424 test kaybı ile modelin stil etiketlerini etkili bir şekilde ayırt edebildiği görülmüştür. CategoryColorModel ise ürün tipi sınıflandırmasında güçlü bir performans sergilemiş, ancak renk tahmininde daha düşük metriklerle sınırlı kalmıştır. Yine de çok görevli öğrenmenin sağladığı birlikte öğrenme yetisi sayesinde modelin her iki görevi de aynı anda yürütebildiği ve anlamlı çıktılar üretebildiği görülmüştür.

Genel olarak, bu çalışma hem çok görevli hem de tek görevli derin öğrenme yaklaşımlarının giysi verileri üzerinde uygulanabilirliğini ortaya koymakta ve her iki yaklaşımın güçlü yönlerini göstermektedir. Elde edilen sonuçlar, moda teknolojileri alanında giysi analizi, akıllı etiketleme, stil öneri sistemleri ve görsel moda arama gibi uygulamalara katkı sunabilecek niteliktedir. Gelecek çalışmalarda veri çeşitliliğinin artırılması, stil sınıflarının daha dengeli temsil edilmesi ve ileri düzey görsel temsil öğrenme yöntemlerinin entegrasyonu ile model performanslarının daha da geliştirilebileceği öngörülmektedir.

Referanslar

Chen, M., Xu, Z., Weinberger, K. Q., & Sha, F. (2012). Marginalized denoising autoencoders for domain adaptation. In F. Pereira, C. J. C. Burges, L. Bottou, & K. Q. Weinberger (Eds.), Proceedings of the 29th International Conference on Machine Learning (ICML 2012) (pp. 767–774).

Ge, Y., Zhang, R., Wang, X., Tang, X., & Luo, P. (2019). DeepFashion2: A versatile benchmark for detection, pose estimation, segmentation and re-identification of clothing images. In Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR) (pp. 5337-5346).

Goodfellow, I., Bengio, Y., & Courville, A. (2016). Deep learning. MIT Press.

He, K., Zhang, X., Ren, S., & Sun, J. (2016a). Deep residual learning for image recognition. In Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR) (pp. 770–778).

He, K., Zhang, X., Ren, S., & Sun, J. (2016b). Identity mappings in deep residual networks. In Proceedings of the European Conference on Computer Vision (ECCV) (pp. 630–645). Springer.

He, R., & McAuley, J. (2016). Ups and downs: Modeling the visual evolution of fashion trends with one-class collaborative filtering. In Proceedings of the 25th International Conference on World Wide Web (WWW '16) (pp. 507–517). ACM.

Hsiao, W.-L., & Grauman, K. (2018). Creating capsule wardrobes from fashion images. In Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR) (pp. 7161–7170). IEEE.

Huang, G., Sun, Y., Liu, Z., Sedra, D., & Weinberger, K. Q. (2016). Deep Networks with Stochastic Depth. In European conference on computer vision (ECCV) (pp. 646-661). Springer, Cham.

Huang, X., Kwiatkowska, M., Wang, S., & Wu, M. (2016). Safety verification of deep neural networks. arXiv preprint arXiv:1610.06940.

Kiapour, M. H., Han, X., Lazebnik, S., Berg, A. C., & Berg, T. L. (2015). Where to buy it: Matching street clothing photos in online shops. In Proceedings of the IEEE International Conference on Computer Vision (ICCV) (pp. 3343–3351). IEEE.

Kingma, D. P., & Ba, J. (2014). Adam: A Method for Stochastic Optimization. arXiv preprint arXiv:1412.6980.

Krizhevsky, A., Sutskever, I., & Hinton, G. E. (2012). ImageNet classification with deep convolutional neural networks. In

F. Pereira, C. J. C. Burges, L. Bottou, & K. Q. Weinberger (Eds.), Advances in Neural Information Processing Systems (Vol. 25, pp. 1097–1105). Curran Associates, Inc.

LeCun, Y., Bengio, Y., & Hinton, G. (2015). Deep learning. Nature, 521(7553), 436–444.

LeCun, Y., Bottou, L., Bengio, Y., & Haffner, P. (1998). Gradient-based learning applied to document recognition. Proceedings of the IEEE, 86(11), 2278–2323.

Liu, Z., Luo, P., Qiu, S., Wang, X., & Tang, X. (2016). DeepFashion: Powering Robust Clothes Recognition and Retrieval with Rich Annotations. In Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR) (pp. 1096–1104).

Long, J., Shelhamer, E., & Darrell, T. (2015). Fully convolutional networks for semantic segmentation. In Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR) (pp. 3431–3440). IEEE.

Loshchilov, I., & Hutter, F. (2017). Decoupled Weight Decay Regularization. arXiv preprint arXiv:1711.05101.

O'Shea, K., & Nash, R. (2015). An introduction to convolutional neural networks. arXiv preprint arXiv:1511.08458.

Reddi, S. J., Kale, S., & Kumar, S. (2018). On the Convergence of Adam and Beyond. In Proceedings of the 6th International Conference on Learning Representations (ICLR).

Ruder, S. (2016). An overview of gradient descent optimization algorithms. arXiv preprint arXiv:1609.04747.

Saranya, M. S., & Geetha, P. (2022). Fashion image classification using deep convolution neural network. In IFIP Advances in Information and Communication Technology.

Simonyan, K., & Zisserman, A. (2014). Very deep convolutional networks for large-scale image recognition. arXiv preprint arXiv:1409.1556.

Shirkhani, S., Mokayed, H., Saini, R., & Chai, H. Y. (2023). Study of AI-driven fashion recommender systems. SN Computer Science, 4(5), 514.

Simo-Serra, E., Fidler, S., Moreno-Noguer, F., & Urtasun, R. (2015). Neuroaesthetics in fashion: Modeling the perception of fashionability. In Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR) (pp. 869–877). IEEE.

Simo-Serra, E., & Ishikawa, H. (2016). Fashion style in 128 floats: Joint ranking and classification using weak data for feature extraction. In Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR) (pp. 298–307). IEEE.

Sun, Y., Zhang, J., Wang, X., & Tang, X. (2021). Deep learning for fashion compatibility modeling: A survey. IEEE Transactions on Pattern Analysis and Machine Intelligence, 43(3), 791–809.

Szegedy, C., Ioffe, S., Vanhoucke, V., & Alemi, A. A. (2017). Inception-v4, Inception-ResNet and the Impact of Residual Connections on Learning. In Proceedings of the AAAI Conference on Artificial Intelligence, 31(1).

Szegedy, C., Liu, W., Jia, Y., Sermanet, P., Reed, S., Anguelov, D., Erhan, D., Vanhoucke, V., & Rabinovich, A. (2015). Going deeper with convolutions. In Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR) (pp. 1–9). IEEE.

Veit, A., Kovacs, B., Bell, S., McAuley, J., Bala, K., & Belongie, S. (2015). Learning visual clothing style with heterogeneous dyadic co-occurrences. In Proceedings of the IEEE

International Conference on Computer Vision (ICCV) (pp. 4642–4650). IEEE.

Zagoruyko, S., & Komodakis, N. (2016). Wide Residual Networks. In Proceedings of the British Machine Vision Conference (BMVC) (pp. 87.1-87.12). BMVA Press.

Zeiler, M. D., & Fergus, R. (2014). Visualizing and understanding convolutional networks. In D. J. Fleet, T. Pajdla, B. Schiele, & T. Tuytelaars (Eds.), Computer Vision – ECCV 2014. Lecture Notes in Computer Science (Vol. 8689, pp. 818–833). Springer.

Zhang, Y., Zhang, P., Yuan, C., & Wang, Z. (2020). Texture and shape biased two-stream networks for clothing classification and attribute recognition. In Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR) (pp. 13538–13547). IEEE.

Zhou, S., Ma, Y., Jia, J., & Wang, Y. (2021). Towards better understanding the clothing fashion styles: A multimodal deep learning approach. In Proceedings of the 35th AAAI Conference on Artificial Intelligence (AAAI-21) (pp. 38–44). AAAI Press.

Wilson, A. C., Roelofs, R., Stern, M., Srebro, N., & Recht, B. (2017). The Marginal Value of Adaptive Gradient Methods in Machine Learning. In Advances in Neural Information Processing Systems (NIPS), 30 (pp. 4148-4158).

Xie, S., Girshick, R., Dollár, P., Tu, Z., & He, K. (2017). Aggregated Residual Transformations for Deep Neural Networks. In Proceedings of the IEEE conference on computer vision and pattern recognition (CVPR) (pp. 1492-1500).

CHAPTER 1

TEST GÜDÜMLÜ GELİŞTİRME SÜREÇLERİNDE YAPAY ZEKÂ UYGULAMALARI

MERVE DAYAPOĞLU[1]
NURSENA BAYĞIN[2]
IŞIL KARABEY AKSAKALLI[3]

Giriş

Test Güdümlü Geliştirme (TDD), yazılım geliştirme süreçlerinde kaliteyi ve sürdürülebilirliği artırarak daha verimli bir geliştirme süreci sağlamak için kullanılan döngüsel bir geliştirme tekniğidir. Bu yöntemde, yazılım kodlamasına başlamadan önce test senaryoları yazılır ve bu senaryolar, yazılımın gereksinimlerini karşıladığını doğrulamak amacıyla geliştirme sürecine rehberlik eder (Moe & Oo, 2020:435; Tosun & ark., 2021:2438; Fucci & ark., 2017:597). Geleneksel test yöntemlerinde ise yapılacak olan testler genellikle kodlama tamamlandıktan sonra oluşturulur ve bu durum hataların daha geç tespit edilmesine yol açar. Bu durum da yazılım

[1]Yüksek Lisans Öğrencisi, Erzurum Teknik Üniversitesi, Bilgisayar Mühendisliği Bölümü, Orcid: 0009-0002-7448-3503
[2] Doktor Öğretim Üyesi, Erzurum Teknik Üniversitesi, Bilgisayar Mühendisliği Bölümü, Orcid: 0000-0003-4457-5503
[3] Doktor Öğretim Üyesi, Erzurum Teknik Üniversitesi, Bilgisayar Mühendisliği Bölümü, Orcid: 0000-0002-4156-9098

geliştirme sürecinde zaman ve maliyet yükünü artırabilir. Yazılım geliştirme sektöründe yazılımın kalitesi, sistem güvenilirliği ve kullanıcı memnuniyeti açısından kritik öneme sahiptir. Geleneksel yöntemlerin aksine TDD yöntemi yazılımdaki kusurları erken tespit etmesi, sorunları düzeltmedeki maliyet ve çabanın azalmasını sağlar. Ayrıca TDD yöntemi sürdürülebilir kod yapılarının geliştirilmesini teşvik ederek yazılımın kalitesini ve sürdürülebilirliğini artırır (Rahman & ark., 2024:51).

Yapılan araştırmalarda TDD'nin kullanıcı memnuniyeti, ürünün kalitesi ve kod sürdürülebilirliği üzerinde olumlu etkilerinin olduğu gözlemlenmiştir. Ancak, test yazımına ayrılan başlangıç maliyeti ve değişen gereksinimlere uyum sağlamada zorlukların olmasından dolayı TDD yönteminden faydalanmak istendiğinde yaşanabilecek bu zorlukları aşmak için gerekli bazı stratejiler belirlenmeli ve uygulanmalıdır. Böylece, TDD yöntemi ile yazılım geliştirme, yazılım ekipleri için daha etkili bir kalite güvence çerçevesi sunar (Rahman & ark., 2024:51).

Test güdümlü geliştirmenin bir diğer faydasına baktığımızda geleneksel geliştirme ile kıyaslandığı çalışmalarda, TDD'nin uygulandığı projelerde programcı üretkenliği azaldığı ve buna bağlı olarak harici kod kalitesinin arttığı gözlemlenirken geleneksel yaklaşımla gerçekleştirilen projelerde bu durumun tam tersi gözlemlenmiştir. Bu, yazılımcının daha az kod yazmaya vakit ayırdığında yazdığı kod üzerinde daha fazla düşünebilmesini, test edebilmesini sağladığından harici kod kalitesinin arttığı anlamına gelmektedir. Ayrıca bu durum yazılımın daha az hata içermesini sağlarken daha iyi tasarlanmış modüllerin, iyi belgelenmiş kodların ve testlerle desteklenmiş bir yapının oluşmasını da sağlamaktadır. Ancak programcı üretkenliğinin arttığı zaman harici kod kalitesinin azaldığı bunun da daha fazla kod yazmaya odaklanıldığı zaman, detaylara dikkat etme süresinin azalmasından dolayı olduğu düşünülmektedir. Ayrıca bu durumda kod kalitesinde düşüşler

olabilirken, hataların artmasına, bakımın zorlaşmasına veya testlerin eksik kalmasına da yol açabilir (Moe & Oo, 2020:435).

TDD kullanarak daha kaliteli, sürdürülebilir kodlar yazabilmek için üç temel adım uygulanmaktadır ve bu adımlar "Kırmızı", "Yeşil" ve "Refaktör" olarak adlandırılmaktadır. Bu adımlar, yazılımın geliştirilmesinde belirli bir iş akışını takip etmekte ve her biri yazılımın doğruluğunu sağlamaya yönelik önemli bir aşamayı temsil etmektedir (Abushama, Alassam & Elhaj, 2021:1; Pančur & Ciglarič, 2011:557; Romano & ark., 2017:64). Şekil 1'de TDD'nin iş akışını görsel olarak temsil eden bu üç adım gösterilmektedir ve bu adımlara ilişkin açıklamalar aşağıda maddeler halinde sunulmaktadır.

Şekil 1. Test Güdümlü Geliştirme

Kırmızı (Red) – Başarısız Olacak Bir Test Yazın:

***Amaç*:** Geliştirilmesi gereken bir özelliği veya çözülmesi gereken bir problemi tanımlayan birim testini yazmaktır. Burada yazılan test henüz üretim kodu yazılmamış olduğu için başarısız olacaktır. Bu, sistemin şu anda beklenen işlevselliğe sahip olmadığını doğrular. Burada sırasıyla gerçekleşecek adımlar:

- Yazılımın sahip olması gereken işlevselliği net bir şekilde tanımlayın.
- Bu işlevselliği test edecek minimum test senaryosunu yazın.
- Testi çalıştırın ve "Başarısız" sonucunu bekleyin (Abushama, Alassam & Elhaj, 2021:1; Karac, Turhan & Juristo, 2021:1315; Pančur & Ciglarič, 2011:557).

Yeşil (Green) – Kodu Çalıştırın:

***Amaç*:** Yazılan testin başarıyla geçmesi için gerekli olan en az kodu yazmaktır. Burada en az kodu yazmaktaki amaç, mümkün olan en basit çözümü geliştirmek ve yazılımın testten geçmesini sağlayarak gereksiz veya karmaşık kod yazmaktan kaçınmaktır. Burada sırasıyla gerçekleşecek adımlar:

- Testin başarısız olmasına neden olan eksik kodu yazmaya başlayın. (Sadece testi geçmek için gereken kadar kod yazın.)
- Testi tekrar çalıştırın ve testin "Başarılı" olduğunu doğrulayın (Abushama, Alassam & Elhaj, 2021:1; Karac, Turhan & Juristo, 2021:1315; Pančur & Ciglarič, 2011:557).

Refaktör (Refactor) – Tekrarları Ortadan Kaldırın:

***Amaç*:** Yazılan kodu optimize etmek, temiz ve modüler bir hale getirmektir. Bu adımda kodun işlevselliği değişmez, ancak yapısı iyileştirilir. Bu adım kodun uzun vadeli sürdürülebilirliğini artırmak ve gelecekteki değişikliklere karşı esnekliğini sağlamak için gereklidir. Burada sırasıyla gerçekleşecek adımlar:

- Yazdığınız kodu ve testi gözden geçirin.
- Gereksiz kod tekrarlarını veya karmaşıklıkları ortadan kaldırın.

Refaktör işlemi sırasında, mevcut testlerin hala çalıştığından emin olun (Abushama, Alassam & Elhaj, 2021:1; Karac, Turhan & Juristo, 2021:1315; Pančur & Ciglarič, 2011:557).

TDD, yazılım geliştirme süreçlerinde kaliteyi artırmaya yönelik proaktif bir yaklaşım sunar. Bu yöntem, kod yazılmadan önce testlerin oluşturulmasını teşvik eder ve erken hata tespiti sayesinde daha kapsamlı test süreci sağlar. Yazılım geliştirme sürecinin ilerleyen aşamalarında ortaya çıkabilecek hataların giderilmesi için harcanan çaba ve maliyetler önemli ölçüde azalır. Geleneksel yazılım geliştirme yaklaşımlarında ise hatalar genellikle sürecin sonunda tespit edilip düzeltilir, bu da maliyet ve zaman açısından verimsizlik yaratabilir. TDD ise bu durumu tersine çevirdiğinden yazılım kalitesini artırmak ve süreç verimliliğini iyileştirmek için modern bir yöntem olarak öne çıkar. TDD, yazılımların modüler ve sürdürülebilir kod yapılarıyla geliştirilmesini teşvik ederek uzun vadeli bakımını kolaylaştırır. Sürekli test odaklı yeniden düzenleme ile geliştiriciler, kodlarını güvenli bir şekilde iyileştirirken, daha temiz ve genişletilebilir yazılımlar ortaya koyabilirler. Bu yaklaşım, geliştiricilerin üretkenliğini artırırken, yazılımın kalitesini de yükseltir. Günümüzün rekabetçi yazılım dünyasında, söz edilen avantajlar TDD'yi yazılım mühendisliği için değerli bir yaklaşım haline getirmektedir (Fucci & ark., 2017:597; Rahman & ark., 2024:51; Santos & ark., 2021:42).

Son yıllarda, TDD yalnızca endüstride değil, eğitim alanında da dikkat çekmektedir. Bilgisayar bilimi ve yazılım mühendisliği gibi disiplinlerde TDD'nin, kaliteli kod geliştirme, disiplinli bir yazılım geliştirme pratiği oluşturma ve profesyonel hayata hazırlık açısından faydaları vurgulanmaktadır. Ancak, TDD'yi öğrencilere benimsetmek ve bu yöntemde ısrarcı olmalarını sağlamak öğrenme sürecinde öğrenme eğrisinin dik olması ve anlık faydaların net görülmemesi gibi nedenlerden dolayı öğrenciler için bir engel teşkil

edebilir. Yine de bu yöntemin yazılım mühendisliği bağlamındaki önemi giderek artmaktadır (Coffey & Bitner, 2022:2023).

Yapay zekâ (AI)'nın TDD sürecine dâhil edilmesi, yazılım geliştiricilerin kodlama, test etme ve dağıtma süreçlerini köklü bir şekilde değiştirmektedir. Testler, mühendislikte doğruluk, hassasiyet ve güvenilirlik açısından kritik rol oynamaktadır. Geleneksel test süreçleri zaman alıcı ve manuel iş yükü gerektiren görevler içerdiğinden, yapay zekâ destekli araçların kullanılması test süreçlerinin iyileştirilmesi ve hızlandırılmasında daha etkili, güvenilir ve esnek çözümler sunabilmektedir. AI Destekli Yazılım Geliştirme Araçları ve diğer AI teknikleri kullanıldığında tekrarlayan faaliyetler basitleştirilir, üretkenlik artar ve geliştiriciler üzerindeki bilişsel yük azalır. Ayrıca, AI hata ayıklamayı ve test vakası oluşturmayı otomatikleştirerek yazılım sistemlerinin ölçeklenebilirliğini ve güvenilirliğini de sağlamaktadır (Pangavhane & ark., 2024:1). Son yıllarda yapılan çalışmalara baktığımızda da AI destekli kod ve test üretme araçları, hata tahmini modelleri ve otomatik hata düzeltme mekanizmaları, TDD süreçlerini daha verimli hale getirmeye başlamıştır (Cassieri, Romano & Scanniello, 2024:902; Hayashi & ark., 2021:406; Roundy, Gilbert & Wilkinson, 2024:1).

Bu bölümde test güdümlü geliştirme süreçlerinin yazılım kalitesi, sürdürülebilirliği ve geliştirme maliyetleri üzerinde AI'nin rolünü ele alarak, AI destekli test yazımı üzerine odaklanmaktadır. Öncelikle, TDD yönteminin temel prensipleri ve geleneksel yaklaşımlarla kıyaslaması ele alınıp ardından yapay zekânın bu sürece nasıl entegre edilebileceği incelenmektedir. AI tabanlı test araçları ve çerçeveler (framework) üzerinde durularak, yazılım mühendisliği alanında bu teknolojilerin sağladığı avantajlar ve olası sınırlamalar tartışılmaktadır.

Test Güdümlü Geliştirme ve Yapay Zekâ Destekli Test Süreçleri

Yazılım mühendisliğinde Test güdümlü geliştirme, kaliteyi artırmak ve yazılım hatalarını erken tespit etmek amacıyla yaygın olarak kullanılan bir yöntemdir. Bu bölümde, öncelikle TDD'nin geleneksel yöntem ile kıyaslanarak yazılım geliştirme üzerindeki etkileri ele alınacaktır. Daha sonra, yapay zekâ destekli test süreçlerinin TDD'ye entegrasyonu ve bu yeni yaklaşımların sağladığı avantajlar incelenecektir.

Test Güdümlü Geliştirme ve Yazılım Test Süreçleri

Üniversitelerde yazılım testinin müfredata eklenmesi, olumlu sonuçlar doğursa da çeşitli zorluklarla karşılaşılmaktadır. Coffey ve Bitner'in (2022:2023) çalışmasında bahsedilen üniversite düzeyinde TDD'nin etkinliği inceleyen bir araştırmada, öğrencilerin geliştirdikleri yazılımda TDD'nin kullanılmasının kalitenin artmasına ve daha verimli sonuçlar elde etmesine olumlu katkılar sağladığı gözlemlenmiştir. Ancak TDD'nin uygulanması sırasında öğrencilerin yöntemi benimsemekte zorlanması, ek sürenin gerekliliği ve programların kendilerini ve test vakalarını kontrol etmede ek notlandırma ihtiyacının olması gibi de dezavantajları olduğu tespit edilmiştir. Ayrıca araştırmalar öğrencilerin TDD yaklaşımını benimsemesindeki oranlarının yaş, programlama deneyimi ve özgüvenle ilişkili olarak değiştiğini göstermektedir. TDD yazılım kalitesini artıran, öğrencilerin üretkenliğini olumlu yönde etkileyen ve öğrencilerin yazılım testi becerilerini geliştirmek, test süreçlerinin avantajlarını daha etkin bir şekilde göstermek için önerilen bir yöntem olsa da karşılaşılan zorlukları aşmak için uygun stratejiler geliştirilmelidir (Coffey & Bitner, 2022:2023).

Papis ve arkadaşları (2022:1644), yazılım dünyasında TDD ve TLD (Test Son Geliştirme) yöntemlerinin karşılaştırılması amacıyla yaptıkları çalışmada, stajyer yazılımcılarla gerçek bir

endüstriyel projede deneysel olarak değerlendirme gerçekleştirmiştir. Çalışmanın ana amacı, bu iki yöntemin yazılım kalitesi üzerindeki etkisini değerlendirmek ve geliştirici ekiplerin yetkinlik düzeylerine göre kod kalitesindeki farklılıkları incelemektir. Çalışmada, stajyerlerin yazılım geliştirme süreçlerini TDD kullanarak gerçekleştirmesi sonucunda kod kalitesi, geliştirme süresi ve hata oranları gibi metrikler analiz edilmiş ve sonuç olarak TDD'nin yazılım hatalarını azaltarak özellikle yeni başlayan geliştiricilerin yazılım süreçlerini iyileştirebileceğini göstermiştir (Papis & ark., 2022:1644).

Fucci ve arkadaşlarının (2017:597) yaptığı çalışmada yazılım geliştirme süreçlerinde TDD yönteminin farklı yaklaşımlarını (önce test veya sonra test) analiz etmişlerdir ve bu yaklaşımların yazılım kalitesi ve geliştirici performansı üzerindeki etkilerini deneysel bir perspektiften ele almışlardır. Sonuçlar, her iki yöntemin de bazı bağlamlarda avantajlar sunduğunu, ancak projenin niteliğine göre farklı etkiler gösterebileceğini ortaya koymuştur (Fucci & ark., 2017:597).

Romano ve arkadaşlarının (2017:64) yaptığı çalışmada ise, TDD yöntemini hem nitel hem de nicel yaklaşımlarla inceleyerek TDD'nin geliştiricilerin davranışlarına, süreçlere ve yazılım geliştirme sonuçlarına olan etkisini anlamayı amaçlamıştır. Araştırmada, 14 bilgisayar bilimi yüksek lisans öğrencisi ve 6 profesyonel yazılım geliştirici (1-10 yıl deneyimli) yer almıştır. Yapılan bu araştırmada katılımcılar, TDD'nin temel aşamalarından biri olan kodun yeniden düzenlenmesi (refactoring) aşamasını genellikle göz ardı etmiş ve bu aşamayı daha az önemli olarak görmüşlerdir. Test yazımında çoğunlukla testler güncel tutulmamış ve hızlıca geçilmek adına testler için sadece üretim kodu yazıldığı gözlemlenmiştir. Katılımcıların TDD'nin teorik faydalarının (örneğin, yüksek kod kalitesi vb.) uygulama sırasında doğru uygulanmadığında yeterince gerçekleştirilmediğini göstermektedir.

Bu nedenle çalışma, özellikle TDD'nin uygulama adımlarına tam bağlılık sağlanmadığında yazılım kalitesine zarar verebileceğini vurgulamıştır (Romano & ark., 2017:64).

Baldassarre ve arkadaşlarının (2021:110937) yaptığı çalışmada TDD yönteminin yazılım geliştiriciler üzerindeki uzun vadeli etkilerini araştırmak amacıyla tasarlanmış uzunlamasına bir çalışmadır. TDD'nin uygulanabilirliğini ve etkilerini, altı aylık bir süre boyunca nasıl değiştiğini anlamaya çalışılmıştır. Çalışma, 30 bilgisayar bilimi lisans öğrencisiyle gerçekleştirilmiş ve TDD'nin uzun süreli olarak uygulanıp uygulanmadığını değerlendirilmiştir. Katılımcılar, başlangıçta TDD eğitimi aldıktan sonra, altı aylık süre zarfında belirli aralıklarla yazılım geliştirme görevlerini yerine getirmiştir. Çalışmada, TDD'nin hem yazılım kalitesi hem de geliştirici performansı üzerindeki etkileri değerlendirilmiş ve TDD'yi benimsemiş bir grubun sonuçları ile TDD'yi kullanmayan bir grupla karşılaştırılmıştır. Çalışmanın sonucunda katılımcılar, altı ay boyunca TDD'yi başarıyla kullanmayı sürdürebilmişler ve TDD kullanan katılımcılar, TDD kullanmayanlara kıyasla daha fazla sayıda test üretmiş ve bu testler sayesinde hataları tespit etme konusunda daha yüksek bir kapasiteye sahip olmuştur. Ancak TDD'nin yazılım ürünlerinin dış kalitesi üzerinde anlamlı bir iyileşme sağlamadığını ortaya koymuş ve geliştirici verimliliği üzerinde kayda değer bir etkisi olmadığı gözlemlenmiştir (Baldassarre & ark., 2021:110937).

Santos ve arkadaşlarının (2021:42) yaptığı çalışmada TDD sürecinin yazılım kalitesi üzerindeki etkilerini hem akademik hem de endüstriyel bağlamlarda değerlendirmek amacıyla tasarlanmış bir dizi deneyden oluşmaktadır. Farklı bağlamlar ve katılımcı türleri arasında TDD'nin performansını karşılaştırarak sonuçların genelleştirilebilirliğini artırmayı hedeflemişlerdir. Çalışma kapsamında 12 farklı deney yürütülmüştür. TDD ile yazılım kalitesinin, özellikle yazılım hatalarını tespit etme kapasitesinde

önemli iyileşmeler sağladığı görülmüştür. Ancak TDD'nin sağladığı kalite iyileşmesi, büyük ölçüde görevlerin niteliğine ve katılımcıların deneyimine bağlıdır. TDD'yi uygulayan profesyonellerin performansı, öğrencilere göre daha düşük bulunmuştur. Bunun nedeni, profesyonellerin yeni bir yöntemi benimseme konusunda daha dirençli olmaları veya motivasyon eksikliği olabileceği düşünülmüştür. (Santos & ark., 2021:42).

Karac ve arkadaşları (2021:1315) tarafından gerçekleştirilen çalışmada TDD sürecinde görev tanımlarının ayrıntı düzeyinin yazılım kalitesi üzerindeki etkisini incelemektedir. Bu kontrollü deneyde, yeni başlayan geliştiricilerin daha ayrıntılı görev tanımlarıyla çalıştıklarında, yazılım kalitesinde önemli ölçüde iyileşme sağladıkları gözlemlenmiştir. Sonuçlar, TDD uygulamalarında görev tanımlarının ayrıntı düzeyinin, özellikle deneyimsiz geliştiriciler için, yazılım kalitesini artırmada kritik bir rol oynadığını göstermektedir (Karac & ark., 2021:1315).

Test Güdümlü Geliştirme ve Yapay Zekâ Destekli Test Süreçleri

Yazılımın kalitesi ve kapsamlı test süreçleri, yazılımın güvenilirliği ve kararlılığı açısından büyük önem taşır. Ancak, yazılım geliştirirken özellikle C gibi dillerin kullanıldığı zamanlarda, karmaşıklık ve koşullu dallanmalar nedeniyle %100 kod kapsamına ulaşmak zorlu bir süreçtir. Uç durumların belirlenmesi geleneksel test yöntemlerinin etkinliğini sınırlayabilir. Bu noktada, yapay zekâ destekli test süreçleri, otomatik test senaryoları oluşturma, hata tespiti ve kod analizini geliştirerek test kapsamını artırmak ve yazılım güvenilirliğini sağlamak için önemli bir çözüm sunabilir. TDD süreçlerinde de yapay zekâ araçlarının kullanımı üzerine yapılan çalışmalara baktığımızda, bu araçların test kapsamını genişletmek, hata oranlarını azaltmak ve süreci daha verimli hale getirmek için önemli katkılar sağladığı gözlemlenmektedir. Örneğin

Cassieri ve arkadaşları (2024:902) tarafından gerçekleştirilen çalışmada, TDD sürecine üretken yapay zekânın entegrasyonu ele alınarak geliştirilen GAI4-TDD adlı PyCharm eklentisinin, TDD'nin yeşil (green) adımında geliştiricilere nasıl destek sağladığı incelenmiştir. Sistem, kırmızı (red) adımda yazılan testleri temel alarak otomatik olarak üretim kodu üretebilmekte ve böylece geliştirme sürecini hızlandırmaktadır. Çalışmada yapılan deneyler, gömülü sistemler bağlamında GAI4-TDD'nin önemli ölçüde başarılı olduğunu göstermiştir. Sonuçlar, yapay zekâ destekli TDD yaklaşımlarının hem geliştirme süresini kısaltmada hem de kod kalitesini artırmada etkili olabileceğini ortaya koymaktadır (Cassieri, Romano & Scanniello, 2024:902).

Hossain ve arkadaşları (2025:1) tarafından yapılan çalışmada yapay zekânın test ve ölçüm süreçlerindeki rolü incelenmiştir. Çalışma, AI tabanlı test sistemlerinin anomali tespiti, öngörücü bakım ve otomatik kalibrasyon gibi alanlarda nasıl kullanılabileceğini ortaya koymaktadır. Elektrikli araç şarj istasyonları üzerinde gerçekleştirilen deneylerde, yapay zekâ destekli test sistemlerinin güvenlik açıklarını başarılı bir şekilde tespit ettiği ve test süreçlerini büyük ölçüde hızlandırdığı gözlemlenmiştir. Sonuçlar, yapay zekâ destekli test yaklaşımlarının mühendislik alanında güvenilirliği artırabileceğini ve manuel test süreçlerine kıyasla daha verimli olabileceğini göstermektedir (Hossain & ark., 2025:1).

Yapay zekânın TDD sürecinde kullanımının önemini araştıran bir diğer çalışmada, Hayashi ve arkadaşları (2021:406), nesnelerin interneti (IoT) modüllerinin test edilebilirliğini artırmak amacıyla TDD ve Makine Öğrenimi kombinasyonuna odaklanmıştır. Bu çalışmada, ESP32 IoT modülü temel alınarak gerçekleştirilen deneylerde, bellek kullanımı, sıcaklık ve WiFi sinyal seviyesi gibi ölçümleri kullanarak bir IoT modülünün durumunun otomatik olarak iyi veya kötü olarak sınıflandırabilen bir test

yaklaşımı önerilmektedir. Çalışmada kullanılan K-En Yakın Komşu (KNN) algoritması, modül performansını değerlendirmede başarılı bulunmuştur. Sonuçlar, makine öğrenimi destekli test süreçlerinin IoT sistemlerinin güvenilirliğini artırabileceğini, hata tespit süresini azaltabileceğini ve manuel değerlendirme süreçlerine kıyasla daha etkin olabileceğini ortaya koymaktadır (Hayashi & ark., 2021:406).

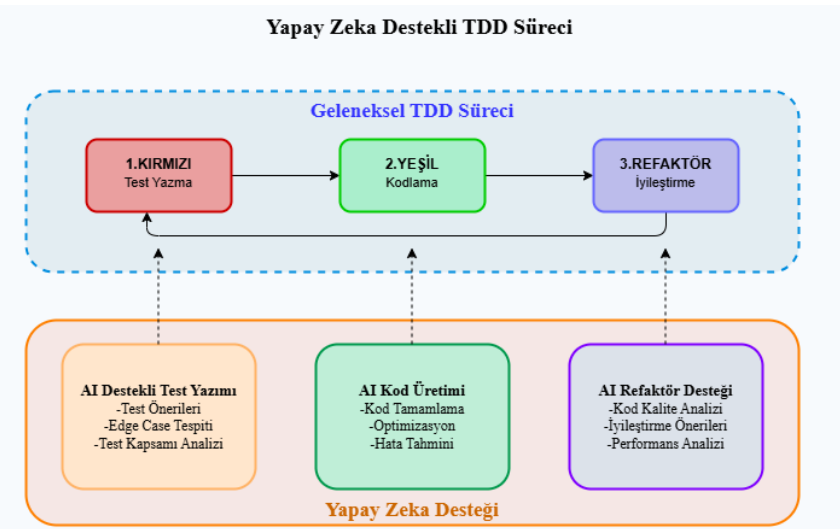
Pangavhane ve arkadaşları (2024:1) tarafından gerçekleştirilen çalışmada yapay zekâ destekli yazılım geliştirme araçlarının yazılım süreçlerini nasıl dönüştürdüğü incelenmiştir. Çalışmada, makine öğrenimi ve doğal dil işleme teknikleriyle desteklenen AI araçlarının, kodlama, test etme, hata ayıklama ve DevOps süreçlerinde nasıl kullanılabileceğini ortaya koymaktadır. Özellikle GitHub Copilot gibi AI destekli kod önerme sistemlerinin geliştirici verimliliğini artırdığı, test otomasyonunun hata tespit süreçlerini iyileştirdiği ve AI tabanlı güvenlik analizlerinin kod güvenliğini artırdığı tespit edilmiştir. Çalışmanın sonuçları, AI'nin yazılım mühendisliğini daha verimli, güvenilir ve ölçeklenebilir hale getirdiğini göstermektedir (Pangavhane & ark., 2024:1).

Helmy ve arkadaşları (2024:1) yaptıkları çalışmada ise, yazılım geliştirme sürecinde güvenilirlik sağlamak için kapsamlı kod kapsamına ulaşılması için Büyük Dil Modelleri'nden (LLMS) ve GCOV aracından yararlanılmış ve Büyük Dil Modellerinin (LLMs) kullanımıyla yazılım test süreçlerini otonom bir şekilde nasıl oluşturulacağı araştırılmıştır. GPT-3.5-turbo, Code Llama ve Llama-2 gibi üretken yapay zekâ modellerinin C programları için test senaryoları üretmek amacıyla nasıl kullanılabileceğini göstermektedir. Çalışmanın deneysel sonuçları, AI tabanlı test senaryolarının %80 ila %100 kod kapsamı sağladığını ve manuel testlere kıyasla daha geniş kapsamlı ve güvenilir test süreçleri sunduğunu bulunmuştur. Sonuç olarak, LLM destekli test otomasyonu, yazılım geliştirme süreçlerinde verimliliği artırırken

hata oranlarını azaltarak yazılım güvenilirliğini önemli ölçüde iyileştirebilir (Helmy, Sobhy & ElHusseiny, 2024:1).

Yapılan birçok çalışma, geleneksel TDD sürecine yapay zekânın entegre edilmesiyle sürecin daha hızlı, verimli ve hatalara karşı daha dayanıklı hale geldiğini göstermektedir.

Şekil 2'de görüldüğü gibi, yapay zekâ desteği TDD'nin her üç aşamasında farklı avantajlar sağlamaktadır. İlk aşama olan kırmızı aşamada, yapay zekâ destekli test yazımı sayesinde geliştiricilere rehberlik edilir; test önerileri sunularak sınır durumları (edge case) belirlenir ve test kapsamı analiz edilmektedir. Yeşil aşamada, yapay zekâ destekli kod üretimi, kod tamamlama, optimizasyon ve hata tahmini gibi işlevlerle daha kaliteli ve verimli kod geliştirilmesini sağlar. Son aşama olan iyileştirme aşamasında ise yapay zekâ, kod kalitesini değerlendirir, iyileştirme önerileri sunar ve performans analizleri gerçekleştirir. Bu entegrasyon sayesinde, test odaklı geliştirme yaklaşımı daha sistematik, güvenilir ve sürdürülebilir hale gelmektedir.



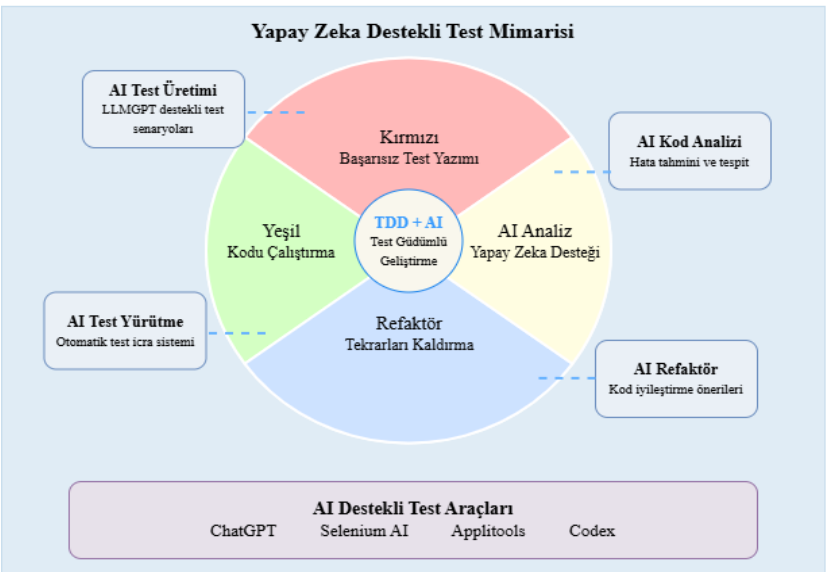
Şekil 2. Yapay Zekâ Destekli TDD Sürecinin Entegrasyonu

Yapay zekâ sistemlerinin TDD sürecinde kullanılmasında birçok avantajı olduğu görünse de karmaşıklıkları nedeniyle geleneksel yazılım test süreçlerinden farklı gereksinimlere sahip olmasından dolayı AI modellerinin doğru, güvenilir ve etik şekilde çalışmasını sağlamak için özel test stratejileri geliştirilmelidir. Fontanesi ve arkadaşları (2023) tarafından gerçekleştirilen çalışmada, farklı yapay zekâ modellerinin test edilmesi için kullanılan stratejiler, detaylı bir vaka çalışması ile ele alınmıştır. Çalışmada, makine öğrenimi ve derin öğrenme modellerinin test edilmesi için kullanılan yöntemler karşılaştırmalı olarak ele alınmakta ve test stratejileri; veri kalitesi ve güvenliği, model doğruluğu, güvenilirlik testleri, etik ve önyargı analizleri ile dayanıklılık testleri gibi aşamalardan oluşmaktadır. Çalışmanın sonuçlarında, AI sistemlerinin güvenilirliğini sağlamak için hem siyah kutu hem de beyaz kutu test tekniklerinin kullanılmasının önemi vurgulanmaktadır. Ayrıca, etik ve önyargı testlerinin, AI sistemlerinin adil ve tarafsız çalışmasını sağlamak için kritik bir rol oynadığı ortaya konmuştur. Sonuç olarak, AI test süreçlerinin sürekli güncellenmesi ve modelin şeffaflığını artırmaya yönelik yeni yöntemlerin geliştirilmesi gerektiği belirtilmektedir (Fontanesi & ark., 2023).

Test Otomasyonu İçin AI Destekli Test Araçları ve Çerçeveleri

Yazılım geliştirme sürecinde hata ayıklama işlemleri, geliştiriciler açısından zaman alıcı, maliyetli ve karmaşık bir süreçtir. Bu nedenle yapay zekâ destekli test otomasyon araçları kullanıldığı zaman test süreçlerinde daha hızlı ve hata oranı daha düşük çözümler sunmaktadır. Makine öğrenimi, derin öğrenme ve doğal dil işleme tekniklerinden yararlanarak geliştirilen AI destekli test sistemleri kullanılarak, test süreçleri optimize edilerek yazılım kalitesini artırmaktadır.

Şekil 3'te gösterildiği gibi, TDD süreci yapay zekâ destekli araçlarla optimize edilebilmektedir. Büyük dil modelleri ve diğer AI destekli sistemler TDD sürecinde test yazımı, kod üretimi, test yürütme ve refaktörizasyon gibi adımlarda geliştiricilere destek sağlayarak sürecin verimliliğini artırmaktadır. Bu adımlarda yapay zekâ destekli araçlar kullanılarak AI Test Üretimi sayesinde LLM tabanlı sistemler test senaryoları oluşturup geliştiricilere rehberlik eder ve test kapsamını genişletir. AI Test Yürütme sistemleri kod çalıştırma sürecinde testlerin geçmesi sağlayarak test süreçlerini otomatikleştirir. Son olarak, AI Refaktör bileşeni, kod kalitesini artırmaya yönelik iyileştirme önerileri sunarak tekrarları ortadan kaldırarak kodun iyileştirilmesine yardımcı olmaktadır (Mathews & Nagappan, 2024:1583). Ayrıca, AI destekli analiz araçları hata tahminleri yaparak yazılım güvenilirliğini artırmaktadır. Bu süreçte ChatGPT, Selenium AI, Applitools ve Codex gibi AI araçları etkin bir şekilde kullanılarak yazılımın test süreçleri optimize edilmektedir.



Şekil 3. Yapay Zekâ Destekli Test Mimarisi ve Süreç Bileşenleri

Özellikle bu konuda yapılan araştırmalarda son yıllarda geliştirilen AI destekli test sistemlerinden GPT-3, Codex ve PaLM gibi büyük ölçekli dil modelleri, kod kapsamında test vakaları oluşturulmasını iyileştirmek üzere tasarlanmıştır. Transformer mimarisi tarafından desteklendiği için doğal dil anlama ve kod üretme yetenekleriyle test senaryoları oluşturma, hata analizi yapma gibi görevlerde önemli bir rol oynamaktadırlar. Büyük parametre boyutları ve geniş veri kümeleri üzerinde eğitilmesi, karmaşık kod yapılarını anlamlandırabilmesini, yazılımın gereksinimlerine uygun test senaryoları üretebilmesini sağlamaktadır (Helmy, Sobhy & ElHusseiny, 2024:1).

Ancak Prenner ve arkadaşları (2022:69) tarafından gerçekleştirilen çalışmada ise, OpenAI'nin Codex modelinin yazılım hatalarını otomatik olarak tespit edip düzeltebilme yeteneği, QuixBugs hata kümesi kullanılarak değerlendirilmiş ve Codex vb. AI sistemlerinin bazı önemli kısıtlamalara sahip olduğu ortaya koyulmuştur. Örneğin, Codex'in değerlendirilmesi sırasında yalnızca tek bir uzman tarafından doğrulama yapılması değerlendirme sürecinin öznelliği açısından sorun oluşturmuş ve hata payını artırabilecek bir durum yaratmıştır. Ayrıca, çalışma yalnızca QuixBugs benchmark[4]'ı ve iki programlama diliyle sınırlı kalmasından dolayı da farklı kod tabanları ve diller üzerindeki performansı tam olarak değerlendirilmemiştir. Bunun yanı sıra, Codex'in büyük veri kümeleri üzerinde eğitildiği için olası veri sızıntıları ve modelin daha önce gördüğü kodlardan hataları öğrenmiş olabileceği riski de bulunmaktadır (Prenner, Babii & Robbes, 2022:69).

Sonuç olarak, Codex gibi AI destekli test otomasyon araçları hata ayıklama süreçlerini önemli ölçüde iyileştirse de değerlendirme

[4]benchmark: Belirli bir sistemin, yazılımın veya modelin performansını değerlendirmek için kullanılan standart bir test kümesi veya ölçüm yöntemidir.

sürecindeki eksiklikler, potansiyel veri sızıntıları ve sınırlı benchmark kapsamı gibi dezavantajların dikkate alınması gerekmektedir.

En Popüler AI Destekli Test Araçları ve Çerçeveleri

Günümüzde birçok AI destekli test aracı, geleneksel test süreçlerine kıyasla daha akıllı ve verimli çözümler sunmaktadır. Öne çıkan bazı araçlar şunlardır:

ChatGPT – Test senaryoları oluşturma, test verisi üretme, hata analizi yapma

Yi ve arkadaşları (2023:72) tarafından yapılan bir çalışmaya göre, ChatGPT'nin test senaryoları oluşturma ve hata analizi süreçlerine nasıl entegre edilebileceği detaylı olarak incelenmiştir. Çalışmada, ChatGPT'nin yazılım testi alanında manuel test süreçlerine yardımcı olabilecek önemli bir araç olduğu belirtilmektedir. Çalışmanın sonucuna göre ChatGPT, farklı programlama dillerindeki kodları anlayabilir ve çeşitli test çerçeveleri ile uyumlu test senaryoları oluşturabilir. Hataları kolayca tespit etmede ve test kapsamını artırmada özellikle test mühendislerine yardımcı bir araç olarak kullanılabilir. Ancak tam otomatik test oluşturma konusunda bazı eksikliklerin bulunduğu ve insan doğrulamasına ihtiyaç duyduğu belirtilmektedir (Yi & ark., 2023:72).

Selenium AI – Web uygulamalarında AI destekli test otomasyonu.

Selenium AI, Selenium WebDriver'ın yapay zekâ ile geliştirilmiş bir versiyonu olarak, web uygulamalarında AI destekli test otomasyonu sağlamak için geliştirilmiştir. Test süreçlerini daha akıllı ve esnek hale getirmeyi amaçlamaktadır. Bu araç, dinamik olarak değişen web öğelerinin tanınması, otomatik test senaryosu

oluşturma ve test bakım maliyetlerinin azaltılması gibi avantajlar sunmaktadır (Bahad, Tadse & Chandankhede, 2024:1).

Zimmermann ve Koziolek(2023:171) tarafından yapılan bir çalışmada, Selenium WebDriver'ın GPT-4 ile entegre edilmesinin, geleneksel test yaklaşımlarına kıyasla daha yüksek bir kod kapsama oranı ve daha verimli hata tespiti sağladığı gösterilmektedir. GPT-4'ün önceden eğitilmiş veri setleri sayesinde kod kapsama oranını artırarak manuel testlere duyulan ihtiyacı önemli ölçüde azalttığını, test süreçlerini hızlandırdığını ve hataları daha hızlı tespit ettiğini göstermektedir (Zimmermann & Koziolek, 2023:171).

GitHub Copilot – Test senaryoları üretimi ve otomatikleştirilmiş test süreçleri

GitHub Copilot, LLM kullanarak test senaryolarını otomatikleştiren ve manuel test yazım sürecine göre daha hızlı ve verimli test senaryoları oluşturabilen bir AI destekli test aracıdır. Pytest ve unittest gibi popüler test çerçeveleri ile uyumlu test senaryoları oluşturabilmektedir (Li & ark., 2024:216).

Mehmood, ve arkadaşları (2023:13) tarafından ve Shuang Li ve arkadaşları (2024:216) tarafından yapılan farklı iki çalışmada, GitHub Copilot'un test üretimindeki sınırlamalar ile otomatik test senaryosu oluşturma yetenekleri manuel yazılan testlerle karşılaştırılarak araştırılmıştır. Copilot tarafından üretilen test senaryolarının, manuel olarak yazılan testler kadar doğru ve geçerli olduğu bulunmuştur. Özellikle, doğru ve açıkça tanımlanmış istemler (prompts) sağlandığında, Copilot'un farklı test senaryoları üretebildiği ve yazılım testi sürecini önemli ölçüde iyileştirdiği tespit edilmiştir. Ancak araştırmada, GitHub Copilot'un test kapsamı konusunda bazı sınırlamaları olduğu ve bazen yinelenen, benzer testler oluşturabildiği belirtilmektedir. Ayrıca, Copilot'un oluşturduğu testin kalitesi, geliştiricinin yazdığı istemlerin (prompt) doğruluğuna ve detayına bağımlı olup, yanlış veya eksik istemler

verildiğinde düşük kaliteli test senaryoları üretebilmektedir. Bu nedenle Copilot ile test yazıldığı zaman test senaryoları eksik veya hatalı olabileceğinden manuel doğrulama ve incelemeye ihtiyaç duyulmaktadır (Li & ark, 2024:216; Mehmood & ark., 2023:13).

Applitools – Görsel UI testlerinde yapay zekâ destekli analiz.

Applitools, görsel GUI testine odaklanan bir araç olup çoğunlukla algılanamayan insan varyasyonlarını filtreleyebilen çeşitli bilgisayarlı görü (CV) algoritmaları kullanmaktadır. Araç, otomatik görsel doğrulama sağlamak için programatik[5] bir API sunar ve Selenium gibi yaygın test çerçeveleriyle entegre çalışabilir. Ayrıca, Applitools, çapraz tarayıcı testlerini destekleyerek web uygulamalarının farklı tarayıcılarda tutarlı bir şekilde çalışmasını sağlar (Awad & ark, 2024:1; Ricca, Marchetto & Stocco, 2021:263).

Ragel ve Balahadia (2023:1) tarafından yapılan bir çalışmaya göre Visual AI, görsel test süreçlerinde manuel müdahaleyi büyük ölçüde azaltarak test doğruluğunu artırmaktadır. Çalışmada, Applitools gibi AI destekli görsel test araçlarının, ara yüz değişikliklerini otomatik olarak analiz ederek test süreçlerini hızlandırdığını ve hata oranlarını azalttığını göstermektedir (Ragel & Balahadia, 2023:1).

Functionize – NLP destekli test senaryosu üretimi, self-healing (kendini düzelten testler), akıllı görsel farklılık analizi

Functionize, test komut dosyalarını oluşturmak ve dinamik bir şekilde güncellemek için NLP tabanlı test komut dosyası oluşturmakta ve yapay zekâ destekli bakım gerçekleştirmektedir. Özellikle akıllı eleman seçimi ile kendi kendini iyileştiren bakım stratejileri önermektedir. Ayrıca, akıllı görsel farklılık tanıma özelliği ile birlikte bozuk testlerin temel nedenini otomatik olarak

[5] programatik: Bir işlemin manuel yerine kod aracılığıyla otomatik olarak yapılması anlamına gelir.

belirleyebilen akıllı öneriler de içermektedir (Ricca, Marchetto & Stocco, 2021:263).

Testim – Test senaryosu optimizasyonu, AI destekli test bakımı

Testim, makine öğrenimi destekli test senaryosu oluşturma ve optimizasyon özellikleriyle, modern test otomasyon süreçlerini hızlandıran bir araçtır. Test komut dosyalarının oluşturulmasında "akıllı yakala-yeniden oynat" (smart capture-replay) yaklaşımını benimsemektedir. Bu sayede, test senaryoları değişen kullanıcı arayüz (UI) bileşenlerine karşı daha dayanıklı hale gelmekte ve zaman içinde güncellenmesi daha kolay olmaktadır (Lal & Kumar, 2021:1; Ricca, Marchetto & Stocco, 2021:263). Yapay zekâ destekli hata tespiti ve test bakımı özellikleri, test senaryolarının adaptasyonunu otomatikleştirerek gereksiz manuel müdahaleyi en aza indirmektedir.

Lal ve Kumar (2021:1) tarafından gerçekleştirilen çalışmada Testim'in kullanıcı dostu arayüzü ve güçlü otomasyon özellikleri sayesinde, özellikle büyük ölçekli yazılım projelerinde test süreçlerini optimize edebildiğini ortaya koymaktadır. Ayrıca, Testim'in entegrasyon kolaylığı sayesinde farklı yazılım geliştirme çerçeveleri ile uyumlu çalışabilmektedir (Lal & Kumar, 2021:1).

Mabl – Test senaryosu üretimi, akıllı locator'lar[6], otomatik hata tespiti.

Mabl, CI (Continuous Integration-Sürekli Entegrasyon) / CD (Continuous Deployment/Delivery- Sürekli Dağıtım/Yayınlama) süreçleri için geliştirilmiş kapsamlı bir test otomasyon platformudur.

---

[6] Akıllı Locator**:** Test otomasyon araçlarının, bir web sayfasındaki öğeleri (butonlar, metin kutuları, bağlantılar vb.) daha sağlam ve güvenilir bir şekilde bulmasını sağlayan gelişmiş bir yöntemdir.

Bulut tabanlı bir SaaS[7] çözümü olarak, testlerin oluşturulmasını, yürütülmesini ve sürdürülmesini kolaylaştırır.

Mabl, web uygulamalarını otomatik olarak analiz eden ve test senaryoları oluşturan bir bağlantı tarayıcısına sahiptir. Mabl'in bu bağlantı tarayıcısı, web uygulamalarını analiz ederek otomatik test senaryoları oluşturması ve test kapsamını genişleterek hata tespit sürecini iyileştirmesi, yazılım mühendisleri ve test uzmanları açısından, manuel test süreçleriyle uğraşmak yerine stratejik hata analizine odaklanmasını sağlamaktadır. Ayrıca, otomatik hata tespiti ve self-healing (kendi kendini iyileştirme) mekanizması, yazılım güncellemeleri sırasında testlerin bozulmasını önleyerek manuel bakım ihtiyacını en aza indirmekte ve test süreçlerini daha verimli ve sürdürülebilir bir hale getirmektedir. AI destekli akıllı locator'lar sayesinde de test senaryolarının dinamik UI değişikliklerine adapte olmasını sağlayarak test süreçlerinin sürdürülebilirliğini artırır (Awad & ark, 2024:1; Ricca, Marchetto & Stocco, 2021:263).

Katalon Studio – Low-code test otomasyonu, geniş entegrasyon desteği, stabil test yürütme.

Katalon Studio, kapsamlı test otomasyonu yetenekleri ve kullanıcı dostu arayüzü ile mobil ve web uygulamalarının test süreçlerini kolaylaştıran bir araçtır. Test mühendisleri ve geliştiriciler bu aracın low-code (düşük kod) yaklaşımı sayesinde, karmaşık test senaryolarını minimal kodlama ile oluşturabilir ve yönetebilir.

Sinaga ve Sitanggang (2023:196) tarafından yapılan bir çalışmada, Katalon Studio'nun test yürütme hızı ve stabilite açısından başka bir yapay zekâ test aracı olan Appium'a ile

[7] SaaS (Software as a Service- Hizmet Olarak Yazılım): Yazılımların internet üzerinden sunulduğu ve kullanıcıların bu yazılımları indirmeden, doğrudan tarayıcı üzerinden kullanabildiği bir hizmet modelidir.

kıyaslanmış ve daha iyi performans gösterdiği bulunmuştur. Çalışmada, Katalon Studio'nun, grafik kullanıcı arayüzü (GUI) testlerinde daha optimize bir test motoruna sahip olduğu ve uzun süreli test yürütmelerinde daha stabil bir şekilde çalıştığı gösterilmiştir (Sinaga & Sitanggang, 2023:196).

Bu kısımda, AI destekli test otomasyon araçlarının sunduğu farklı avantajlar ve bu araçlara dayalı sınırlamalar ele alınmıştır. Her aracın, test süreçlerini hızlandırma, hata tespitini iyileştirme ve test senaryolarının sürdürülebilirliğini artırma konusunda kendine özgü güçlü yönleri bulunmaktadır. Şekil 4'te de bazı AI destekli test araçlarının karşılaştırmalı analizi bulunmaktadır. Bu analiz, otomatik test üretimi, kod analizi, hata tahmini ve UI testi gibi çeşitli kategorilerdeki performanslarını değerlendirmektedir. Puanlama ise kategorilere dayalı olarak her aracın performansını yansıtmaktadır.

AI Destekli Test Araçlarının Karşılaştırılması

Test Aracı	Otomatik Test Üretimi	Kod Analizi	Hata Tahmini	UI Testi	Puan
Testim AI AI ile Test Senaryoları	★★★★★	★★★★☆	★★★★★	★★★★☆	4.5
Mabl Akıllı Test Otomasyon	★★★★☆	★★★★☆	★★★★☆	★★★★★	4.2
Functionize NLP ile Test Otomasyonu	★★★★☆	★★★★☆	★★★★★	★★★★☆	4.1
Github Copilot LLM Destekli Kod Üretim	★★★★☆	★★★★★	★★★☆☆	★★★☆☆	3.8
Applitools Görsel IU Testi	★★★☆☆	★★★☆☆	★★★★☆	★★★★★	3.7
Selenium AI Web UI Testi	★★★☆☆	★★★☆☆	★★★★☆	★★★★★	3.7
Katalon AI Düşük Kodlu Test Ortamı	★★★☆☆	★★★☆☆	★★☆☆☆	★★★☆☆	2.8

Şekil 4. AI Destekli Test Araçlarının Karşılaştırılmalı Analizi

Bu analize göre, en yüksek puana sahip araç Testim AI olarak öne çıkmaktadır. Bu araç, özellikle otomatik test üretimi ve hata tahmini kategorilerinde maksimum puanla değerlendirilmiştir.

İkinci sırada yer alan Mabl, UI testi konusunda üstün performans göstermektedir. Akıllı test otomasyonu ve self-healing

mekanizmalarıyla güçlü bir hata tespiti sunmaktadır. Functionize ise NLP tabanlı test otomasyonu sağlayan önemli bir test aracı olarak üçüncü sırada bulunmaktadır.

LLM tabanlı kod üretimiyle test süreçlerinde yardımcı olan GitHub Copilot, kod analizi kategorisinde en yüksek puanı almasına rağmen, genel performansıyla dördüncü sırada yer almaktadır. Applitools ve Selenium AI ise özellikle UI testi kategorisinde tam puan alan test araçları, görsel AI testi konusunda uzmanlaşan Applitools ve web arayüz testlerinde güçlü olan Selenium AI, spesifik test ihtiyaçları için önemli alternatifler sunmaktadır.

En düşük puanı alan Katalon AI ise düşük kodlu test otomasyonu sunmasına rağmen, kod analizi ve hata tahmini açısından diğer araçlara kıyasla daha sınırlı bir performans göstermektedir. Bu karşılaştırma, farklı yapay zekâ yaklaşımlarını içeren test araçlarının güçlü ve zayıf yönleri göstermesi açısından önemlidir. Yazılım geliştirme ekipleri, test süreçlerinde en uygun aracın seçilmesinde bu tablodan yararlanarak proje gereksinimlerine ve özellikle odaklanmak istedikleri test kategorilerine göre en uygun aracı seçebilir.

Sonuçlar

Yapılan çalışmalar sonucunda TDD'nin yazılım mühendisliği eğitiminde benimsenmesi, öğrencilerin yazılım testi becerilerini geliştirmekte etkili olduğu ancak öğrenciler üzerinde başarıyla uygulanması için, öğrencilerin temel becerilerinin güçlendirilmesi ve zaman kısıtlamalarını yönetebilmeleri adına uygun eğitim stratejilerinin geliştirilmesi gerekmektedir. Yazılımcılar üzerinde yapılan çalışmalar sonucunda ise TDD yöntemini uygulamada bazı yazılımcıların başarıyla uygulayabildiği ve yazılım geliştirme süreçlerini iyileştirdiği gözlemlenirken bazı yazılımcılar için ise yeni bir yöntemi benimsemenin daha zor olduğu gözlemlenmiştir. Ancak, TDD yöntemi başarıyla uygulandığında

hata oranlarını düşürdüğü, kod kalitesini artırdığı ve yazılımcıların gerçek bir endüstriyel proje bağlamında daha disiplinli bir yaklaşım geliştirmelerine katkı sağladığı belirtilmiştir ve TDD'yi etkili bir şekilde uygulamak için yazılımcılara süreç boyunca rehberlik edilmesi gerektiği vurgulanmıştır. Bu nedenle yazılım mühendisliği eğitiminde TDD'yi benimseme oranlarının, rehberlik ve destekle artırılabileceğini göstermiştir. Özellikle, TDD'nin sistematik bir süreç gerektirmesi nedeniyle, yazılımcıların başta zorluk yaşasa da süreç boyunca uygulamalı öğrenme ile bu yöntemi daha verimli kullanabildikleri belirtilmiştir. Ancak TDD'nin yaygın şekilde benimsenmesinin önünde önemli bazı zorluklar bulunmaktadır: test yazma sürecinin başlangıçtaki zaman maliyeti, yöntemi öğrenme eğrisinin dikliği veya uygulama sırasında yeniden düzenleme aşamasına gereken önemin verilmemesi gibi faktörler. Bu zorluklarla başa çıkmak için uygun stratejiler geliştirilmelidir. Test yazma sürecindeki zaman maliyetini azaltmak için otomatikleştirilmiş test araçlarının kullanımı veya TDD yöntemini öğrenme sürecini kolaylaştırmak için takım içi eğitimler ile mentörlük desteğinin artırılması gibi stratejilerle yazılım geliştirme süreçlerinin sürdürülebilirliği sağlanabilir ve yazılım kalitesi önemli ölçüde iyileştirilebilir.

TDD incelediğimizde yazılım mühendisliği açısından birçok avantajı bulunmaktadır. Öncelikle, hata oranlarını düşürerek yazılımın daha güvenilir olmasını ve kod kalitesini artırarak daha modüler ve sürdürülebilir bir yazılımın oluşmasını sağlar. Ayrıca daha disiplinli bir yazılım geliştirme süreci sunması yazılım geliştiricilerin sistematik ve planlı bir şekilde çalışmasını teşvik ederken yazılım mühendisliği eğitiminde de öğrencilerin test becerilerini geliştirmesine yardımcı olur. TDD kullanımı başlangıçta ekstra zaman gerektirse de uzun vadede hata ayıklama süresini azalttığından geliştirme süresinde tasarruf sağlamasına olanak tanımaktadır. TDD'nin birçok avantajı olsa da bazı dezavantajları

da göz ardı edilmemelidir. Özellikle bu yöntemi bilmeyen geliştiriciler için öğrenme eğrisinin dik olması TDD'ye adapte olmayı zorlaştıran önemli bir faktördür. Ayrıca, test yazma sürecinin geliştirme hızını başlangıçta yavaşlatması bu nedenle ek zaman ve çaba gerektirmesi zaman açısından bazı sorunların yaşanmasına neden olabilir. Uygulama sürecinde de yeniden düzenleme aşamasına yeterince önem verilmemesi yöntemin sağladığı faydaların tam anlamıyla elde edilememesine yol açabilir. Bu gibi nedenlerle, TDD'nin projelerde etkin bir şekilde uygulanabilmesi için yöntemin temel prensiplerinin iyi anlaşılması ve geliştirme sürecine disiplinli bir şekilde entegre edilmesi gerekmektedir.

TDD sürecine yapay zekanın entegre edilmesi, test yazma ve hata tespiti süreçlerini otomatikleştirerek süreci hızlandırılabileceğini göstermektedir. AI tabanlı analizler ile test süreçleri daha hızlı tamamlanabilir, manuel test süreçlerini en aza indirerek test maliyetlerini düşürebilir. AI tabanlı kod tamamlama ve öneri sistemleri (örneğin GitHub Copilot), geliştiricilere kod yazarken otomatik öneriler sunarak verimliliği artırabilir. AI sistemleri, kod içerisindeki hataları ve verimsizlikleri belirleyerek geliştiricilere daha kaliteli kod yazma konusunda rehberlik eder ve hata payını düşürerek daha hassas ölçümler yapabilir ve AI sistemler farklı test ortamlarına kolayca adapte edilerek kullanılabilir. Ancak yapay zekâ destekli yazılım geliştirme araçları geliştirme sürecini hızlandırsa da gizlilik endişeleri ve aşırı otomasyon bağımlılığı gibi bazı dezavantajları da içermektedir. AI destekli araçların daha akıllı ve daha iyi kararlar alabilmesi için model geliştirme süreçlerinin devam etmesi gerekmektedir.

Yapay zekâ destekli test süreçleri geleneksel test yaklaşımlarına kıyasla birçok avantajı bulunmaktadır. Yapay zekâ destekli test sistemlerinin sunduğu otomasyon ve analiz yetenekleri sayesinde test döngüleri hızlanmakta ve genel verimlilik artmaktadır. Bunun yanı sıra, manuel iş yükünün azalması, test

süreçlerini daha verimli hale getirirken, maliyetleri de düşürmektedir. Ayrıca, AI destekli test sistemleri erken hata tespiti yapması yazılımın kalitesini artmakta ve test süreçlerindeki doğruluk ile hassasiyet değerlerini yükseltmektedir. Uyarlanabilirlik ve esneklik özellikleri sayesinde, farklı projelere kolayca adapte olabilir ve sürekli değişen yazılım ortamlarına uygun çözümler sunabilir. Ancak yapay zekâ destekli test süreçlerinin bazı dezavantajları da bulunmaktadır. AI modellerinin kullanılması etik, güvenilirlik ve tarafsızlık sorunlarının yaşanmasına neden olabilir. Örneğin AI destekli test araçları yanlış veya eksik test senaryoları üreterek yanlış veya eksik hata raporlarına yol açabilir ya da test edilen yazılımdan toplanan verilerin nasıl işlendiği ve depolandığı konusunda şeffaflık sağlanmazsa, kullanıcı verileri tehlikeye girebilir. Bu gibi nedenlerden dolayı AI destekli test araçları test süreçlerini hızlandırsa da etik, güvenilirlik ve tarafsızlık sorunları barındırmaktadır. Ayrıca, AI tabanlı test sistemlerinin sürekli olarak güncellenmesi gerekmekte, bu da ek maliyet ve zaman gereksinimi doğurmaktadır. Veri kalitesi ve güvenliği de AI'nin etkinliği açısından kritik bir unsur olup, hatalı veya eksik veriler test süreçlerinin güvenilirliğini olumsuz etkileyebilir. Öğrenme ve entegrasyon sürecinin karmaşık olması ek teknik bilgi ve kaynak gerektirmekte ve geliştiricilerin AI destekli test sistemlerine adapte olmasını zorlaştırmaktadır. Son olarak, aşırı otomasyon bağımlılığı, insan faktörünü devre dışı bırakma riski taşıyabilir ve beklenmedik senaryolara karşı esneklik sağlayamayan test süreçleri oluşturabilir.

Kaynakça

Abushama, H. M., Alassam, H. A., & Elhaj, F. A. (2021). The effect of Test-Driven Development and Behavior-Driven Development on Project Success Factors: A Systematic Literature Review Based Study. 2020 International Conference on Computer, Control, Electrical, and Electronics Engineering (ICCCEEE), 1–9. https://doi.org/10.1109/ICCCEEE49695.2021.9429593

Bahad, S. A., Tadse, S., & Chandankhede, P. (2024). Optimizing Test Efficiency in Web Development with Selenium and Java. 2024 IEEE 9th International Conference for Convergence in Technology (I2CT), 1–5. https://doi.org/10.1109/I2CT61223.2024.10543407

Baldassarre, M. T., Caivano, D., Fucci, D., Juristo, N., Romano, S., Scanniello, G., & Turhan, B. (2021). Studying test-driven development and its retainment over a six-month time span. Journal of Systems and Software, 176, 110937. https://doi.org/10.1016/j.jss.2021.110937

Cassieri, P., Romano, S., & Scanniello, G. (2024). Generative Artificial Intelligence for Test-Driven Development: GAI4- TDD. 2024 IEEE International Conference on Software Analysis, Evolution and Reengineering (SANER), 902–906. https://doi.org/10.1109/SANER60148.2024.00098

Coffey, J. W., & Bitner, S. (2022). Student Persistence in the Use of Test Driven Development. 2022 International Conference on Computational Science and Computational Intelligence (CSCI), 2023–2027. https://doi.org/10.1109/CSCI58124.2022.00363

Fontanesi, G., Ortíz, F., Lagunas, E., Baeza, V. M., Vázquez, M. Á., Vásquez-Peralvo, J. A., ... & Chatzinotas, S. (2023). Artificial intelligence for satellite communication and non-terrestrial networks: A survey. *arXiv preprint arXiv:2304.13008*.

Fucci, D., Erdogmus, H., Turhan, B., Oivo, M., & Juristo, N. (2017). A Dissection of the Test-Driven Development Process: Does It Really Matter to Test-First or to Test-Last? IEEE Transactions on Software Engineering, 43(7), 597–614. https://doi.org/10.1109/TSE.2016.2616877

Hossain, M. S., Mohaimin, M. R., Alam, S., Rahman, M. A., Islam, M. R., Anonna, F. R., & Akter, R. (2025). AI-Powered Fault Prediction and Optimization in New Energy Vehicles (NEVs) for the US Market. *Journal of Computer Science and Technology Studies*, *7*(1), 01-16.

Hayashi, V. T., Ribeiro, C. M. N., Filho, A. Q., Pita, M. A. B., Trazzi, B. M., Estrella, J. C., & Ruggiero, W. V. (2021). Improving IoT Module Testability with Test-Driven Development and Machine Learning. 2021 8th International Conference on Future Internet of Things and Cloud (FiCloud), 406–412. https://doi.org/10.1109/FiCloud49777.2021.00066

Helmy, M., Sobhy, O., & ElHusseiny, F. (2024). AI-Driven Testing: Unleashing Autonomous Systems for Superior Software Quality Using Generative AI. 2024 International Telecommunications Conference (ITC-Egypt), 1–6. https://doi.org/10.1109/ITC-Egypt61547.2024.10620598

Karac, I., Turhan, B., & Juristo, N. (2021). A Controlled Experiment with Novice Developers on the Impact of Task Description Granularity on Software Quality in Test-Driven Development. IEEE Transactions on Software Engineering, 47(7), 1315–1330. https://doi.org/10.1109/TSE.2019.2920377

Mathews, N. S., & Nagappan, M. (2024). Test-Driven Development and LLM-based Code Generation. Proceedings of the 39th IEEE/ACM International Conference on Automated Software Engineering, 1583–1594. https://doi.org/10.1145/3691620.3695527

Moe, M. M., & Oo, K. K. (2020). Evaluation of Quality, Productivity, and Defect by applying Test-Driven Development to perform Unit Tests. 2020 IEEE 9th Global Conference on Consumer Electronics (GCCE), 435–436. https://doi.org/10.1109/GCCE50665.2020.9291950

Pančur, M., & Ciglarič, M. (2011). Impact of test-driven development on productivity, code and tests: A controlled experiment. Information and Software Technology, 53(6), 557–573. https://doi.org/10.1016/j.infsof.2011.02.002

Pangavhane, S., Raktate, G., Pariane, P., Shelar, K., Wakchaure, R., & Kale, J. N. (2024). AI-Augmented Software Development: Boosting Efficiency and Quality. 2024 International Conference on Decision Aid Sciences and Applications (DASA), 1–5. https://doi.org/10.1109/DASA63652.2024.10836523

Papis, B., Grochowski, K., Subzda, K., & Sijko, K. (2022). Experimental Evaluation of Test-Driven Development With Interns Working on a Real Industrial Project. IEEE Transactions on Software Engineering, 48(5), 1644–1664. https://doi.org/10.1109/TSE.2020.3027522

Prenner, J. A., Babii, H., & Robbes, R. (2022). Can OpenAI's codex fix bugs? Proceedings of the Third International Workshop on Automated Program Repair, 69–75. https://doi.org/10.1145/3524459.3527351

Ragel, R. K. C., & Balahadia, F. F. (2023). Visual Test Framework: Enhancing Software Test Automation with Visual Artificial Intelligence and Behavioral Driven Development. 2023 IEEE 15th International Conference on Humanoid, Nanotechnology, Information Technology, Communication and Control, Environment, and Management (HNICEM), 1–5. https://doi.org/10.1109/HNICEM60674.2023.10589222

Rahman, M. S., Kumar Saha, A., Chakraborty, U., Sujana, H. T., & Shafi, S. M. A. (2024). Evaluating the impact of Test-Driven Development on Software Quality Enhancement. International Journal of Mathematical Sciences and Computing, 10(3), 51–76. https://doi.org/10.5815/ijmsc.2024.03.05

Ricca, F., Marchetto, A., & Stocco, A. (2021). AI-based Test Automation: A Grey Literature Analysis. 2021 IEEE International Conference on Software Testing, Verification and Validation Workshops (ICSTW), 263–270. https://doi.org/10.1109/ICSTW52544.2021.00051

Romano, S., Fucci, D., Scanniello, G., Turhan, B., & Juristo, N. (2017). Findings from a multi-method study on test-driven development. Information and Software Technology, 89, 64–77. https://doi.org/10.1016/j.infsof.2017.03.010

Roundy, S., Gilbert, N., & Wilkinson, J. (2024). The Future of AI in Test and Measurement. 2024 IEEE AUTOTESTCON, 1–6. https://doi.org/10.1109/AUTOTESTCON47465.2024.10697428

Santos, A., Vegas, S., Dieste, O., Uyaguari, F., Tosun, A., Fucci, D., Turhan, B., Scanniello, G., Romano, S., Karac, I., Kuhrmann, M., Mandić, V., Ramač, R., Pfahl, D., Engblom, C., Kyykka, J., Rungi, K., Palomeque, C., Spisak, J., … Juristo, N. (2021). A family of experiments on test-driven development. Empirical Software Engineering, 26(3), 42. https://doi.org/10.1007/s10664-020-09895-8

Sinaga, A. M., & Sitanggang, M. (2023). Performance Comparison of Katalon Studio and Appium on Investree Application in Funding Module. 2023 IEEE International Conference on Data and Software Engineering (ICoDSE), 196–201. https://doi.org/10.1109/ICoDSE59534.2023.10291590

Tosun, A., Dieste, O., Vegas, S., Pfahl, D., Rungi, K., & Juristo, N. (2021). Investigating the Impact of Development Task on External Quality in Test-Driven Development: An Industry Experiment. IEEE Transactions on Software Engineering, 47(11), 2438–2456. https://doi.org/10.1109/TSE.2019.2949811

Yi, G., Chen, Z., Chen, Z., Wong, W. E., & Chau, N. (2023). Exploring the Capability of ChatGPT in Test Generation. 2023 IEEE 23rd International Conference on Software Quality, Reliability, and Security Companion (QRS-C), 72–80. https://doi.org/10.1109/QRS-C60940.2023.00013

Zimmermann, D., & Koziolek, A. (2023). GUI-Based Software Testing: An Automated Approach Using GPT-4 and Selenium WebDriver. 2023 38th IEEE/ACM International Conference on Automated Software Engineering Workshops (ASEW), 171–174. https://doi.org/10.1109/ASEW60602.2023.00028

CHAPTER 2

ÜRETKEN YAPAY ZEKA İLE DİJİTAL DÖNÜŞÜM: GELECEĞİN KARİYER ALANLARI

ALİ ÇETİNKAYA[1]
ERCAN AYKUT[2]

Giriş

Yapay Zeka (YZ), yenilikçi teknolojilerle bir araya geldiğinde, sunduğu inovatif çözümlerle insan hayatını köklü bir şekilde dönüştürmeye devam etmektedir. Algoritmalar ve programlama yapılarıyla bütünleşmiş çalışan YZ, yalnızca mevcut problemleri çözmekle kalmayıp aynı zamanda elde ettiği verilerden öğrenerek kendini geliştirme yeteneği sayesinde teknolojik dönüşümün öncüsü haline gelmiştir. Bu güçlü birleşim hem bireysel hem de endüstriyel uygulamalarda inovasyonun sınırlarını yeniden tanımlayarak, geleceğin şekillenmesinde kilit bir rol oynamaktadır.

Yeni nesil dijital dönüşüm ve yapay zeka teknolojileri, günümüz eğitim sisteminde eleştirel düşünme ve problem çözme

[1] Öğr. Gör. Ali ÇETİNKAYA, İstanbul Gelişim Üniversitesi, İstanbul Gelişim Meslek Yüksekokulu, Elektronik Teknolojisi Programı, 34310, İstanbul, Türkiye, alcetinkaya@gelisim.edu.tr, ORCID ID: 0000-0003-4535-3953.
[2] Dr. Ercan AYKUT, İstanbul Gelişim Üniversitesi, Mühendislik ve Mimarlık Fakültesi, Elektrik - Elektronik Mühendisliği Bölümü, 34310, İstanbul, Türkiye, eaykut@gelisim.edu.tr, ORCID: 0000-0001-8639-8408.

becerilerinin gelişiminde önemli bir rol oynamaktadır. Dijital eğitim platformları, proje tabanlı öğrenme, işbirlikçi öğrenme, çevrimiçi tartışmalar, simülasyonlar ve senaryo bazlı öğrenme gibi yöntemlerle bireylerin bilgiyi analiz etme, sentezleme ve uygulama yetkinliklerini artırırken, aynı zamanda teknolojinin sunduğu yenilikçi imkanlarla eğitimi daha etkileşimli hale getirmektedir. Özellikle yapay zeka, nesnelerin interneti, sanal gerçeklik ve blockchain gibi teknolojiler, sanat, tasarım, mühendislik ve eğitim gibi alanlarda dönüşüm yaratmakta ve bireylerin yaratıcılıklarını teknolojiyle bütünleştirmelerine olanak sağlamaktadır. Bu bağlamda, eğitimcilerin öğrenme materyallerini dijital araçlarla daha dinamik ve eleştirel düşünmeyi teşvik edecek şekilde tasarlamaları büyük önem taşımaktadır. Yeni nesil öğrenme yaklaşımlarıyla desteklenen dijital eğitim platformları, öğrencilerin teknolojiyi sadece tüketen bireyler olarak değil, üreten ve dönüştüren bireyler olarak yetişmelerine rehberlik ederek hem akademik hem de profesyonel hayatlarında başarılı olmalarını sağlayacaktır.

Literatür Özeti

Yapay zekâ (YZ), insan benzeri bilişsel yetenekleri taklit eden sistemler geliştirerek veri analizi, karar verme ve otomasyon süreçlerini dönüştüren bir teknolojidir. Makine Öğrenimi (ML) ve Derin Öğrenme (DL) gibi alt alanlar sayesinde yapay zekâ, farklı sektörlerde inovasyonu destekleyen güçlü bir araç haline gelmiştir. Günümüzde YZ; sağlık, eğitim, finans, üretim, güvenlik, oyun ve medya gibi birçok alanda kullanılmakta, iş süreçlerini hızlandırmakta ve karar alma mekanizmalarını geliştirmektedir.

YZ'nin tarihsel temellerinden birini oluşturmaktadır. Turing'in algoritmik yaklaşımı, problem çözme teknikleri ve makine zekası kavramları, modern YZ teknolojilerinin gelişiminde ilham kaynağı olmuştur. Bu nedenle, Turing’in bu eseri sadece kriptografi için değil, aynı zamanda yapay zekanın temel prensiplerinin

şekillenmesi için de kritik bir öneme sahiptir (Turing, 1940). Yapay zekanın bilimsel bir disiplin olarak ortaya çıkmasında kilit bir rol oynamış ve Turing'in makine zekasına dair fikirleri, günümüzdeki YZ teknolojilerinin temellerini oluşturmuştur (Turing, 1948).

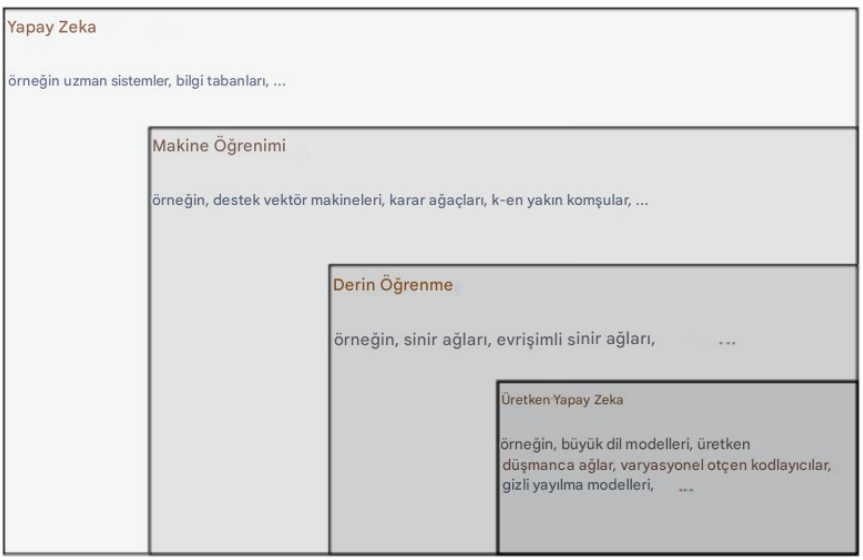
YZ araştırmalarında karşılaşılan zorlukları ve insan seviyesinde zekâya ulaşmanın karmaşıklığını vurgulayan önemli bir eserdir. Bu çalışma, YZ alanında daha derinlemesine araştırmalar yapılmasının ve insan zekâsının farklı yönlerinin incelenmesinin önemini ortaya koymuştur (McCarthy, 1955). YZ araştırmalarında tek katmanlı perceptronların doğrusal olarak ayrılabilir problemleri çözebildiği, ancak daha karmaşık, doğrusal olmayan problemleri çözmede yetersiz kaldığı anlaşılmıştır. Bu sınırlamalar, daha sonra çok katmanlı yapay sinir ağlarının geliştirilmesine ve derin öğrenme alanının doğmasına yol açmıştır (Rosenblatt, 1957). YZ ve bilişsel bilimler alanında Arf'ın makinelerin düşünme kapasitesi üzerine yaptığı değerlendirmeler, YZ'nin felsefi temellerine dair önemli bir katkı sunmaktadır. Arf, makinelerin dil kullanabilme, hesap yapabilme, benzerlik kurabilme ve eleme yapabilme gibi zihinsel yetilere sahip olacak şekilde tasarlanabileceğini belirtmiştir. Ancak, insan ve makine arasındaki temel farkın, insanın sahip olduğu estetik bilincin makinelere kazandırılmasının zorluğunda yattığını vurgulamıştır. (Arf, 1959).

Lotfi A. Zadeh'in 1965 yılında yayımladığı "Fuzzy Sets" başlıklı makalesi, bulanık mantık ve bulanık kümeler teorisinin temelini atan bir çalışmadır. Bu çalışmada Zadeh, geleneksel küme teorisindeki kesin üyelik kavramını genişleterek, bir elemanın bir kümeye ait olma derecesini [0, 1] aralığında bir değerle ifade eden bulanık kümeler kavramını tanıtmıştır. Bu yaklaşım, özellikle belirsizlik ve kesin olmayan bilgilerin işlendiği mühendislik, yapay zeka ve karar verme sistemleri gibi alanlarda büyük yankı uyandırmıştır (Zadeh, 1965). Derin öğrenme ve özellikle evrişimli sinir ağlarının belge tanıma ve genel olarak desen tanıma

alanlarındaki potansiyelini ortaya koyan temel bir referans niteliğindedir. Ayrıca, gradyan tabanlı öğrenme yöntemlerinin karmaşık tanıma sistemlerinin geliştirilmesindeki önemini vurgulamaktadır (LeCun ve diğerleri, 1988).

Üretken Yapay Zeka (GAI), yapay zeka (YZ) alanındaki alt kavramlardan ayrıştırılarak ele alınması gereken, yenilikçi ve dönüşüm sağlayan bir teknolojidir. Makine Öğrenimi (ML) ve Derin Öğrenme (DL) gibi alt disiplinlerden yararlanarak, GAI, mevcut verilerden öğrenerek yeni ve özgün içerikler üretebilme yeteneğine sahiptir. Bu kavramlar Şekil 1 üzerinde üretken yapay zekanın ve diğer yz kavramlarının yapısı gösterilmiştir (Banh ve Strobel, 2023).

Şekil 1: Üretken Yapay Zeka ve diğer Yapay Zeka kavramları
(Banh ve Strobel, 2023)



Üretken Yapay Zeka teknolojisi, sanat ve tasarım, sağlık, eğitim, sanal gerçeklik ve içerik üretimi gibi farklı alanlarda büyük bir potansiyel sunmaktadır. Yaratıcı endüstrilerde, sanat eserleri, dijital tasarımlar ve medya içeriklerinin otomatik olarak üretilmesi sağlanırken, sağlık sektöründe ise hastalık teşhisi, tedavi planlaması

ve tıbbi veri analizi gibi alanlarda yapay zekâ destekli çözümler geliştirilmektedir (Jovanovic ve Campbell, 2022). GAI'nin sunduğu bu yenilikçi yaklaşımlar, farklı sektörlerdeki iş akışlarını optimize etmekte ve yeni kariyer fırsatları yaratmaktadır. Üretken Yapay Zeka (GAI), yapay zekâ alanında, mevcut verilerden yeni ve orijinal içerik oluşturabilen modelleri ifade eder. Bu modeller, metin, görüntü, ses ve video gibi çeşitli veri türlerinde yaratıcı çıktılar üretebilir. GAI'nin temelinde, geniş veri kümeleri üzerinde eğitilmiş derin öğrenme modelleri yer alır. Bu modeller, karmaşık desenleri ve yapıları öğrenerek, benzer özelliklere sahip yeni içerikler oluşturabilir. Örneğin, dil modelleri, doğal dil işleme tekniklerini kullanarak anlamlı ve akıcı metinler üretebilir (Lv, 2023).

Sağlık hizmetlerinde dijital dönüşümün bir parçası olarak, sesli asistanlar (Apple Siri, Amazon Alexa, Google Assistant, ve Microsoft Cortana) potansiyel olarak önemli bir rol oynayabilir. Yapay zeka ve dijital sağlık teknolojilerinin geliştirilmesi ve sağlık profesyonelleri ile entegrasyonunun önemini bir kez daha ortaya koymaktadır (Yang ve diğerleri, 2021). Sağlık sektöründe dijital dönüşümün bir parçası olarak, bu tür makine öğrenimi modelleri, erken teşhis ve tedavi süreçlerini iyileştirmede önemli bir rol oynayabilir. Üretken yapay zeka teknikleriyle birleştiğinde, bu modellerin daha da geliştirilerek, hastalık teşhisi ve tedavisinde yenilikçi çözümler sunması mümkündür (Ozcan ve diğerleri, 2022). Nesne tanıma teknolojileri, endüstri 4.0 ve dijital dönüşüm süreçlerinde önemli bir rol oynamaktadır. Bu tür çalışmalar, üretken yapay zekanın endüstriyel uygulamalarda nasıl kullanılabileceğine dair önemli örnekler sunmaktadır (Yılmaz ve diğerleri, 2020).

Göç ve nüfus hareketliliği gibi dinamik süreçlerin tahmin edilmesi, dijital dönüşümün önemli bir parçasıdır. Bu tür makine öğrenimi modelleri, politika yapıcıların ve planlamacıların daha bilinçli kararlar almasına yardımcı olabilir. Üretken yapay zekâ teknikleriyle birleştiğinde, bu modellerin daha da geliştirilerek,

göçmen entegrasyonu ve kaynak planlaması gibi konularda yenilikçi çözümler sunması mümkündür (Aydemir ve diğerleri, 2022). Araç tespiti ve takibi, akıllı ulaşım sistemlerinin önemli bir parçasıdır. Bu tür çalışmalar, dijital dönüşüm sürecinde, trafik yönetimi, güvenlik ve otonom sürüş teknolojilerinin geliştirilmesine katkı sağlamaktadır. Üretken yapay zeka teknikleriyle birleştiğinde, bu modellerin daha da geliştirilerek, trafik akışı optimizasyonu ve kazaların önlenmesi gibi alanlarda yenilikçi çözümler sunması mümkündür (Elassy ve diğerleri, 2024). k-NN algoritması, sınıflandırma problemlerinde sıkça kullanılan temel bir yöntemdir. Bu tür kaynaklar, dijital dönüşüm sürecinde veri analizi ve modelleme becerilerinin geliştirilmesine katkı sağlar. Ayrıca, üretken yapay zeka uygulamalarında da temel oluşturabilecek algoritmaların anlaşılmasına yardımcı olur (Çetinkaya ve Gök, 2024).

Sağlık araştırmalarında dijital dönüşümün önemli bir unsuru olarak, yapay zekâ (YZ) ve doğal dil işleme (NLP) modellerinin kullanımı, bilgiye erişimi kolaylaştırmakta ve bilimsel iletişimi geliştirmektedir. Üretken yapay zekâ, araştırmacılara daha hızlı ve doğru içerik üretme imkânı sunarak akademik dünyada büyük bir dönüşüm sağlamaktadır. Özellikle ortopedi ve travmatoloji gibi spesifik alanlarda, bu teknolojilerin entegrasyonu bilimsel yazım süreçlerini dönüştürmekte ve araştırmacılara yeni fırsatlar sunmaktadır. Ancak, bu yeniliklerin etik boyutları göz önünde bulundurulmalı ve yapay zekâ destekli sistemlerin güvenilirliği titizlikle değerlendirilmelidir (Köse ve diğerleri, 2024).

Üretken Yapay Zekâ (GAI), eğitimde kişiselleştirilmiş öğrenme materyalleri oluşturma, otomatik içerik üretimi ve öğretim süreçlerini daha etkileşimli hale getirme gibi önemli fırsatlar sunmaktadır. Örneğin, GAI tabanlı sistemler, öğrencilerin bireysel özelliklerine uygun, adım adım ilerleyen ve kişiselleştirilmiş öğrenme ortamları sağlayan Akıllı Öğretici Sistemler (Intelligent

Tutoring Systems) geliştirebilir. Ayrıca, belirli bir alanda uzmanlık bilgisi sunan ve öğrencilere rehberlik eden Uzman Sistemler (Expert Systems) ile öğrencilerin sorularına yanıt veren ve onlara yardımcı olan yapay zekâ destekli sohbet robotları (Chatbotlar) da GAI'nin eğitimdeki uygulamaları arasındadır. Bu teknolojiler, eğitimde yapay zekâ uygulamaları hakkında farkındalık oluşturulmasına katkı sağlamaktadır (İncemen ve Öztürk, 2024).

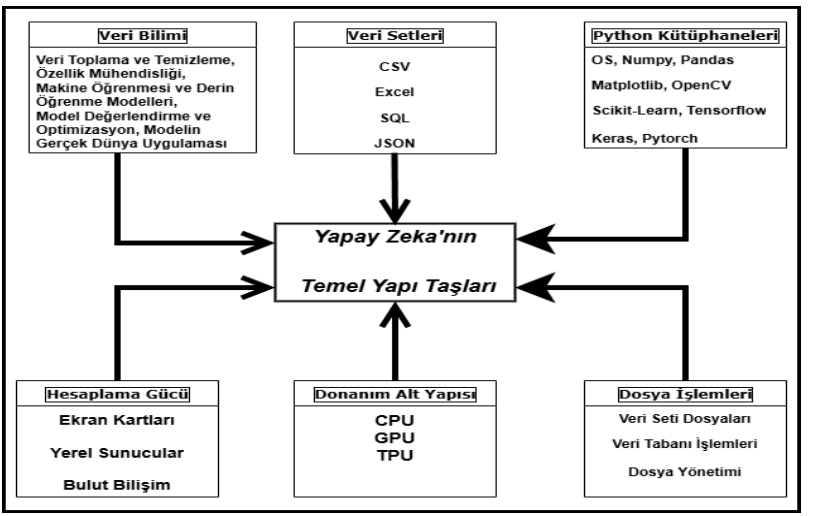
Üretken Yapay Zeka (GAI), yaratıcı endüstrilerden sağlık, ulaşım, eğitim ve göç politikalarına kadar geniş bir yelpazede yenilikçi çözümler sunarak dijital dönüşüm süreçlerine katkıda bulunmaktadır. Bu bağlamda, üretken yapay zekâ yalnızca teknolojik dönüşümü hızlandırmakla kalmayıp, aynı zamanda yapay zekâ alanında yeni kariyer fırsatları da yaratmaktadır. Sağlık, eğitim, endüstri ve ulaşım gibi alanlarda yapay zekâ çözümlerine duyulan ihtiyaç arttıkça, AI mühendisleri, veri bilimcileri, etik uzmanları ve yazılım geliştiriciler için geniş bir iş sahası oluşmaktadır. Özellikle üretken yapay zekâ modellerinin geliştirilmesi, eğitimi ve uygulanması konularında uzmanlaşan profesyoneller, gelecekte bu dönüşümün en önemli aktörleri arasında yer alacaktır. Bu nedenle, yapay zekâ alanında kariyer yapmak isteyen bireyler için teknik becerilerin yanı sıra etik, güvenilirlik ve disiplinler arası yaklaşımlar da kritik öneme sahiptir.

Üretken Yapay Zekaya Genel Bir Bakış

Yapay zeka uygulamalarında, algoritmaların etkili bir şekilde çalışabilmesi için güçlü ve esnek yazılım araçlarına ihtiyaç duyulur. Bu noktada, Python programlama dili ve onun zengin kütüphane ekosistemi, YZ projelerinde önemli bir rol oynamaktadır. Python, geniş kullanım alanı ve kullanıcı dostu yapısı ile YZ geliştirmeleri için tercih edilen bir dildir. Özellikle veri işleme, modelleme ve görselleştirme süreçlerinde kullanılan Python kütüphaneleri, YZ projelerinin başarısını artırmaktadır. Bu bağlamda, Pandas ve

Numpy, veri manipülasyonu ve matematiksel hesaplamalar için sıklıkla tercih edilirken; Matplotlib görselleştirme araçları ile veri analizi sonuçlarını grafiksel olarak sunmak için kullanılır. Scikit-Learn, makine öğrenmesi modellerinin hızlı bir şekilde geliştirilmesi için geniş bir algoritma yelpazesi sunar. Derin öğrenme uygulamaları için ise TensorFlow ve PyTorch gibi kütüphaneler, yüksek performanslı hesaplamalar ve model eğitimi sağlar. BeautifulSoup, web veri kazıma uygulamaları için kullanılarak, verinin internetten toplanmasını ve analiz edilmesini mümkün kılar. Bu kavramlar Şekil 2 üzerinde YZ'nin temel yapı taşları olarak gösterilmiştir.

Şekil 2: YZ'nin Temel Yapı Taşları



Tablo 1'de yer alan bu Python kütüphaneleri, YZ projelerinde veri işleme, model oluşturma, sonuçları görselleştirme ve dış veri kaynaklarından veri toplama gibi çeşitli görevleri yerine getirirken, her biri kendine özgü güçlü özellikleriyle uygulamaların etkinliğini arttırmaktadır. YZ uygulamaları, bu kütüphanelerin sağladığı işlevsellik sayesinde daha verimli ve hızlı hale gelir, böylece problemlerin çözümü daha etkili bir şekilde gerçekleştirilir.

Tablo 1: Python Kütüphaneleri ve Kullanım Alanları.

Python Kütüphaneleri ve Kullanımları	
Kütüphane İsmi	**Kütüphane Kullanım Alanı ve Örnek Kütüphane Tanımlaması**
Pandas	Veri yapıları ve veri işlemleri "import pandas as pd"
Numpy	Sayısal hesaplamalar, büyük boyutlu dizi ve matris işlemleri. "import numpy as np"
Matplotlib	Grafikler, histogramlar, çizgi grafiklerinin oluşturulması "import matplotlib.pyplot as plt"
Scikit-Learn	Sınıflandırma, regresyon algoritmaları ve makine öğrenimi modellerinin oluşturulması "from sklearn.ensemble import RandomForestClassifier"
Tensorflow	Derin öğrenme ve yapay sinir ağı modellerinin oluşturulması "import tensorflow as tf"
PyTorch	Derin öğrenme ve yapay sinir ağı modellerinin oluşturulması "import torch"
BeautifulSoup	Web'ten HTML ve XML dosyalarını ayrıştırmak ve webten veri toplama – kazıma işlemleri "from bs4 import BeautifulSoup"

Açık kaynak veriler, Yapay Zekâ (YZ) çalışmalarının temel yapı taşlarından biri olarak, model geliştirme sürecinde kritik bir rol oynamaktadır. Yapay zekâ sistemlerinin başarılı bir şekilde eğitilebilmesi için büyük ve çeşitli veri kümelerine ihtiyaç duyulmaktadır. Açık kaynak veri setleri, araştırmacılara ve geliştiricilere serbestçe erişim sağlayarak projelerin hızla gelişmesine katkıda bulunur. Ayrıca, bu veri setleri YZ algoritmalarının doğruluğunu test etmek ve farklı modelleri karşılaştırmak için standart bir temel sunar. Sağlık, finans, ulaşım ve çevre gibi çeşitli alanlarda model geliştirmeye olanak tanıyan bu

veriler, yapay zekânın şeffaf, adil ve erişilebilir bir ekosistem oluşturmasına yardımcı olmaktadır.

YZ alanındaki algoritmalar, makine öğrenmesi (ML) ve derin öğrenme (DL) teknikleri sayesinde veriler üzerinden tahminler ve kararlar üretmektedir. Her algoritmanın belirli adımları ve süreçleri içermesi, tutarlı ve güvenilir sonuçlar elde edilmesini sağlar. Başarılı bir YZ modeli oluşturabilmek için algoritmaların kesin, optimize edilmiş ve anlaşılır bir yapıda olması büyük önem taşımaktadır.

Bu süreçte, veri analizi ve model geliştirme için çeşitli YZ kütüphaneleri ve araçları kullanılmaktadır. Pandas ve NumPy, veri manipülasyonu ve matematiksel hesaplamalar için en çok tercih edilen kütüphanelerdir. Matplotlib, veri analizi sonuçlarının görselleştirilmesi için kullanılırken, Scikit-Learn geniş bir makine öğrenmesi algoritma yelpazesi sunarak model geliştirme sürecini hızlandırmaktadır. Derin öğrenme uygulamalarında TensorFlow ve PyTorch, büyük veri kümeleri üzerinde yüksek performanslı eğitim ve tahmin yapma yetenekleriyle öne çıkmaktadır. Ayrıca, BeautifulSoup, web veri kazıma süreçlerinde veri toplamayı ve analiz etmeyi mümkün kılarak açık kaynak verinin daha geniş bir çerçevede kullanılmasını sağlamaktadır.

Bu kapsamda, açık kaynak verilerin erişilebilirliği, YZ modellerinin gelişimini hızlandırmakta, işbirliklerini teşvik etmekte ve inovatif çözümlerin daha geniş bir kitleye ulaşmasını sağlamaktadır. Özellikle üretken yapay zekâ ile birleştiğinde, bu açık veri kaynakları; içerik üretimi, eğitim, güvenlik ve sanal gerçeklik gibi alanlarda yeni fırsatlar doğurarak geleceğin yapay zekâ odaklı kariyerlerini şekillendirmektedir.

Yapay Zekada Kariyer Fırsatları

Yapay zekâ (YZ), dijital dönüşümün merkezinde yer alarak farklı sektörlerde geniş iş fırsatları yaratmaktadır. Özellikle üretken yapay zekâ, içerik üretiminden güvenlik ve etik alanlarına kadar

birçok yeni meslek dalını beraberinde getirmektedir. Bu teknolojiler, metin, görsel, ses ve video üretimi gibi yaratıcı süreçleri otomatikleştirirken, aynı zamanda eğitim, oyun ve sanal gerçeklik gibi alanlarda kişiselleştirilmiş deneyimler sunarak profesyoneller için yeni kariyer yolları açmaktadır.

Şekil 3 üzerinde, Yapay Zekâ ve Üretken Yapay Zekâ birleştiğinde, dört ana iş alanının hızla geliştiği gözlemlenmiştir. Üretken Yapay Zekâ ve İçerik Üretimi, medya, pazarlama ve yaratıcı endüstrilerde otomatik içerik üretimi, görsel ve metin oluşturma gibi süreçleri dönüştürmüştür. Eğitim ve Kişiselleştirilmiş Öğrenme, akıllı öğretici sistemler, sanal eğitim asistanları ve özelleştirilmiş öğrenme materyalleri ile bireyselleştirilmiş eğitim deneyimlerini mümkün kılmıştır. Güvenlik, Etik ve Sorumlu Yapay Zekâ, yapay zekâ sistemlerinin adil ve güvenilir olmasını sağlamak için etik denetim, veri mahremiyeti ve sahte içerik tespiti gibi kritik alanlara odaklanmaktadır. Son olarak, Oyun, Sanal ve Artırılmış Gerçeklik, yapay zekâ destekli interaktif oyunlar, sanal karakterler ve gerçekçi simülasyonlarla yeni nesil dijital deneyimler sunarak oyun ve eğlence sektöründe devrim yaratmaktadır.

Şekil 3: Yapay Zeka ile Kariyer Yolları

Yapay zekâ, farklı sektörlerde devrim yaratmaya devam ederken, üretken yapay zekâ tabanlı çözümler yeni kariyer yolları açmaktadır. İçerik üretimi, eğitim, güvenlik, oyun ve sanal gerçeklik gibi alanlarda yapay zekâ mühendisleri, veri bilimcileri, etik uzmanları ve AR/VR geliştiricileri geleceğin mesleklerini şekillendiren temel aktörler olacaktır. Bu nedenle, yapay zekâ alanında kariyer yapmak isteyen bireyler için teknik becerilerin yanı sıra etik farkındalık, veri analitiği ve disiplinler arası çalışma yetenekleri büyük önem taşımaktadır.

SONUÇ

Yeni nesil dijital dönüşüm ve yapay zeka teknolojileri, üretken zeka ile birleşerek geleceğin kariyer alanlarını kökten değiştirmektedir. Yapay zeka (YZ), insan zekasına özgü bilgi edinme, görme, düşünme, karar verme ve strateji geliştirme gibi yetenekleri yazılımlara kazandırarak, veri işleme süreçlerinden yaratıcı içerik üretimine kadar geniş bir yelpazede etkili çözümler sunmaktadır. Bu bağlamda, yapay sinir ağları, makine öğrenmesi, derin öğrenme ve bulanık mantık gibi YZ alt dalları, dijital dönüşüm sürecinde kritik bir rol oynamaktadır.

Özellikle üretken zeka (Generative AI), sanat, tasarım, mühendislik ve eğitim gibi disiplinlerde yeni fırsatlar yaratmakta, nesnelerin interneti, blockchain, metaverse ve sanal etkileyiciler gibi yenilikçi teknolojilerle entegre olarak dijital dönüşüm sürecini hızlandırmaktadır. Python programlama dili, geniş kütüphane desteği ve kullanıcı dostu yapısı sayesinde, YZ projelerinde veri işleme, modelleme ve görselleştirme gibi temel süreçleri kolaylaştırarak inovasyonu desteklemektedir. Google Colab gibi çevrimiçi platformlar, bu teknolojilerin deneyimlenmesini ve geliştirilmesini hızlandırarak daha kapsayıcı ve erişilebilir çözümler sunmaktadır.

Sonuç olarak, dijital dönüşüm ve yapay zeka teknolojilerinin eğitimden sanayiye, sanattan ticarete kadar pek çok alana entegrasyonu, bireylerin ve kurumların üretkenliği artırarak geleceğin kariyer fırsatlarını yeniden şekillendirmektedir. Açık veri politikalarının benimsenmesi ve güçlü yazılım araçlarının kullanımı, yapay zekanın inovasyon süreçlerindeki etkisini daha da artarak, sürdürülebilir ve kapsayıcı bir dijital ekosistemin oluşmasına katkı sağlayacaktır.

Kaynakça

Arf, C. (1959). Makine Düşünebilir mi ve Nasıl Düşünebilir? Atatürk Üniversitesi 1958-1959 Öğretim Yılı Halk Konferansları, (1), 91-103.

Zadeh, L. A. (1965). Fuzzy sets. Information and control, 8(3), 338-353.

LeCun, Y., Bottou, L., Bengio, Y., & Haffner, P. (1998). Gradient-based learning applied to document recognition. Proceedings of the IEEE, 86(11), 2278-2324.

Çetinkaya, A., & Gök, M., (2024). k-NN Sınıflandırma Yöntemi. Makine Öğrenmesi Algoritmaları (pp.186-198), Ankara: Nobel Yayınevi.

Banh, L., & Strobel, G. (2023). Generative artificial intelligence. Electronic Markets, 33(1), 63.

İncemen, S., & Öztürk, G. (2024). Farklı eğitim alanlarında yapay zekâ: uygulama örnekleri. International Journal of Computers in Education, 7(1), 27-49.

Turing, A. M. (1948). Intelligent machinery. report for national physical laboratory. reprinted in ince, dc (editor). 1992. mechanical intelligence: Collected works of am turing.

Aydemir, B., Aydın, H., Çetinkaya, A., & Polat, D. Ş. (2022). Predicting the Income Groups and Number of Immigrants by Using Machine Learning (ML). International Journal of Multidisciplinary Studies and Innovative Technologies, 6(2), 162-168.

Jovanovic, M., & Campbell, M. (2022). Generative artificial intelligence: Trends and prospects. Computer, 55(10), 107-112.

Köse, Ö., Yapar, D., & Boz, M. (2024). Ortopedi ve travmatoloji araştırmalarında yapay zekâ uygulamaları: Doğal dil işleme potansiyeli. TOTBİD Dergisi, 23, 79-90.

Lv, Z. (2023). Generative artificial intelligence in the metaverse era. Cognitive Robotics, 3, 208-217.

McCarthy, J. (1955). Human-Level Ai Is Harder Than It Seemed.

Ozcan, I., Aydin, H., & Cetinkaya, A. (2022). Comparison of classification success rates of different machine learning algorithms in the diagnosis of breast cancer. Asian Pacific journal of cancer prevention: APJCP, 23(10), 3287.

Rosenblatt, F. (1957). The perceptron, a perceiving and recognizing automaton Project Para. Cornell Aeronautical Laboratory.

Turing, A. M. (1940). Mathematical theory of enigma machine. Public Record Office, London, 3, 150.

Yang, S., Lee, J., Sezgin, E., Bridge, J., & Lin, S. (2021). Clinical advice by voice assistants on postpartum depression: cross-sectional investigation using Apple Siri, Amazon Alexa, Google Assistant, and Microsoft Cortana. JMIR mHealth and uHealth, 9(1), e24045.

Elassy, M., Al-Hattab, M., Takruri, M., & Badawi, S. (2024). Intelligent transportation systems for sustainable smart cities. Transportation Engineering, 100252.

Yılmaz, O., Aydın, H., & Çetinkaya, A. (2020). Faster R-CNN Evrişimsel sinir ağı üzerinde geliştirilen modelin derin öğrenme yöntemleri ile doğruluk tahmini ve analizi: Nesne Tespiti Uygulaması. Avrupa Bilim ve Teknoloji Dergisi, (20), 783-795.

CHAPTER 3

YAPAY SİNİR AĞI MİMARİLERİ: TEK KATMANLI, ÇOK KATMANLI VE GERİ BESLEMELİ YAPILAR

ZEYNEP ILKILIÇ[1]
İSMAİL İŞERİ[2]
BEŞİR DANDIL[3]

Giriş

Yapay sinir ağları (YSA), insan beynindeki sinaptik bağlantılardan esinlenerek geliştirilen, öğrenme ve karar verme süreçlerini taklit edebilen güçlü bilgi işleme modelleridir. Bu yapılar, doğrusal olmayan ilişkileri öğrenebilme kapasiteleri ve yüksek genelleme başarıları sayesinde özellikle büyük ve karmaşık veri kümelerinde oldukça başarılı sonuçlar vermektedir. Sınıflandırma, regresyon, örüntü tanıma, zaman serisi analizi gibi görevlerde yaygın biçimde kullanılan bu modeller, günümüzde yapay zekânın birçok alanında temel yapı taşı hâline gelmiştir.

[1] Dr, Ondokuzmayıs Üniversitesi Yeşilyurt Meslek Yüksekokulu Mekatronik Bölümü , Orcid: 0000-0003-1828-1181
[2] Dr. Öğretim Görevlisi, Ondokuzmayıs Üniversitesi, Mühendislik Fakültesi , Orcid: 0000-0002-0442-1406
[3] Prof.Dr.,Mustafa Kemal Üniversitesi, Mühendislik Fakültesi, Orcid: 0000-0002-3625-

Son yıllarda yapay sinir ağlarına yönelik araştırmalar, mimari yapıların öğrenme performansına etkilerini daha derinlemesine incelemeye başlamıştır. Özellikle katman sayısı, bağlantı türü, aktivasyon fonksiyonu ve veri akış yönü gibi parametrelerin öğrenme kapasitesi üzerindeki etkisi büyük önem taşımaktadır. 2023 yılında Bingham ve Miikkulainen [1], 2913 farklı aktivasyon fonksiyonunu karşılaştırmalı olarak analiz ederek mimari performansı optimize eden yeni yaklaşımlar geliştirmiştir. Benzer şekilde, Lee [2], GELU fonksiyonunun CNN ve MLP tabanlı yapılarda klasik fonksiyonlara kıyasla belirgin performans artışları sağladığını göstermiştir. Sinir ağı mimarilerinin uygulama başarısı, yalnızca matematiksel yapılarına değil; aynı zamanda kullanım bağlamına göre şekillenmektedir. Schneider ve Vlachos [3], ileri ve geri beslemeli yapılardan transformer tabanlı modellere kadar geniş bir mimari yelpazeyi değerlendirerek, mimari kararların uygulama başarısı üzerindeki belirleyici rolünü vurgulamıştır. Özellikle sağlık sektöründe, zaman serisi analizine dayalı risk tahmini ve hasta takibi gibi uygulamalarda geri beslemeli yapılar (RNN, LSTM) oldukça etkili bulunmuştur [4]. Beyin görüntüleme, biyobelirteç analizi ve hastalık sınıflandırmasında kullanılan CNN temelli mimariler ise, görsel verinin anlamlandırılmasında yüksek doğruluk sağlamaktadır [4]. Yapay sinir ağı mimarilerinin tasarımı yalnızca araştırmacıların elinde kalmamış; Neural Architecture Search (NAS) gibi algoritmalar aracılığıyla otomatikleştirilmeye de başlanmıştır. White ve arkadaşları [5], NAS alanında yayımlanmış 1000'den fazla çalışmayı tarayarak sinir ağı tasarım süreçlerinde yükselen eğilimleri ve mimari karmaşıklığın nasıl yönetildiğini tartışmıştır. Yeni yayınlanan Hyena [6] ve Mamba [7] gibi mimariler ise, uzun dizili verileri daha az hesaplama yüküyle işleyebilmek için geleneksel yaklaşımlara alternatifler sunmuştur.

Derin öğrenmenin uygulamaya dönük yönü de hızla gelişmektedir. Onyekpe ve arkadaşları [8], görüntü işleme, finansal

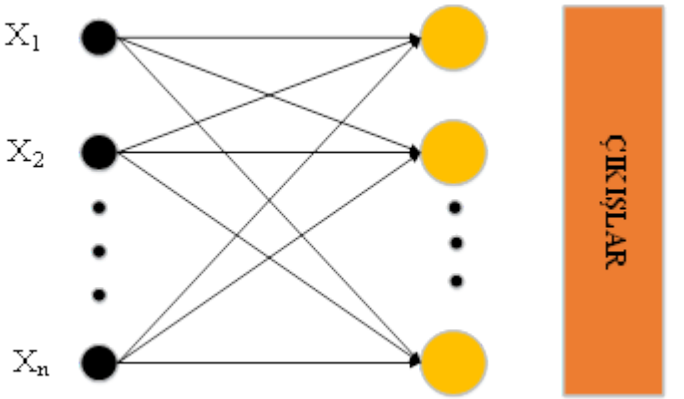
tahmin, robotik ve doğal dil işleme gibi alanlarda sinir ağı mimarilerinin pratik kullanımına dair yeni teknikleri ve örnek uygulamaları paylaşmışlardır. Bu gelişmeler, yapay sinir ağlarının yalnızca teorik bir yapı değil, gerçek dünya problemlerinin çözümünde temel bir teknoloji olduğunu ortaya koymaktadır.

Bu çalışmanın amacı, yapay sinir ağı mimarilerini teorik temelleriyle birlikte ele alarak; tek katmanlı, çok katmanlı, ileri beslemeli ve geri beslemeli yapıları kuramsal ve uygulamalı yönleriyle tanıtmaktır. Mimari yapıların avantajları, sınırlılıkları ve kullanım senaryoları karşılaştırmalı olarak incelenmekte ve okuyucunun problem türüne uygun yapıyı seçebilmesi hedeflenmektedir.

Tek Katmanlı Yapay Sinir Ağı Mimarileri

Yapay sinir ağlarının en temel biçimini oluşturan tek katmanlı mimariler, öğrenme sürecine dair temel prensiplerin kavranabilmesi açısından kritik öneme sahiptir [9]. Özellikle sınıflandırma ve regresyon gibi temel görevlerde kullanılmak üzere tasarlanan bu mimariler, veri ile model arasındaki doğrudan ilişkiyi incelemek için başlangıç düzeyinde güçlü bir araç sunar. Aynı seviyede sinir ağlarının bulunduğu en basit yapay sinir ağı mimarisi tek katmanlı yapay sinir ağı mimarileridir. En temel özelliklerin öğrenildiği bu ağ yapısında bir giriş katmanı bir de çıkış katmanı bulunmaktadır. Şekil 1'de tek katmanlı bir yapay sinir ağı mimarisi verilmiştir. Bu mimari yapısında öğrenmenin gerçekleşmesi için giriş katmanına gelen girdi verilerinin her bir özelliği bir giriş düğümüne bağlanır ve ağırlıklarla çarpılarak çıkış düğümlerine gönderilir. Çıkış düğümlerine gelen bilgiler girişlerin ağırlıklı toplamıdır. Çıkışlar modele uygun bir şekilde seçilecek aktivasyon fonksiyonu ile aktive edilerek ağın tahminleri elde edilir [10].

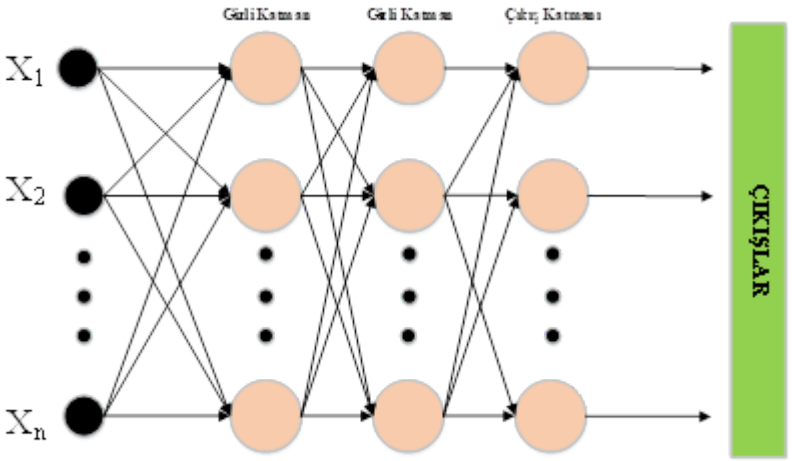
***Şekil 1**. Tek katmanlı sinir ağı mimarisi*



Çok Katmanlı Yapay Sinir Ağı Mimarileri

Temel özelliklerin öğrenildiği tek katmanlı mimarilere, daha karmaşık bilgilerin öğrenilmesi için gizli katmanların eklendiği yapılar çok katmanlı mimarilerdir. Bu yapılarda yüksek düzeyde özelliklerin öğrenilebilmesi için birçok gizli katman bulunur. Şekil 2'de verilen çok katmanlı sinir ağı mimarisinde görüldüğü gibi çok katmanlı mimariler en az bir gizli katman içerir ve bu katmanlar sayesinde verilerdeki karmaşık özellikler öğrenilir. Giriş katmanında bulunan düğümler gizli katman düğümlerine ve gizli katman düğümleri de çıkış katman düğümlerine bağlanır. Her gizli katman düğümü ile çıkış düğümü arasında bias terimi bulunmaktadır. Bu yapılar çoklu işlem katmanlarından oluştuğu için doğrusal olmayan verilerden öğrenme özelliğine de sahiptir [11].

Şekil 2. *Çok katmanlı sinir ağı mimarisi*



İleri Beslemeli Ağlar

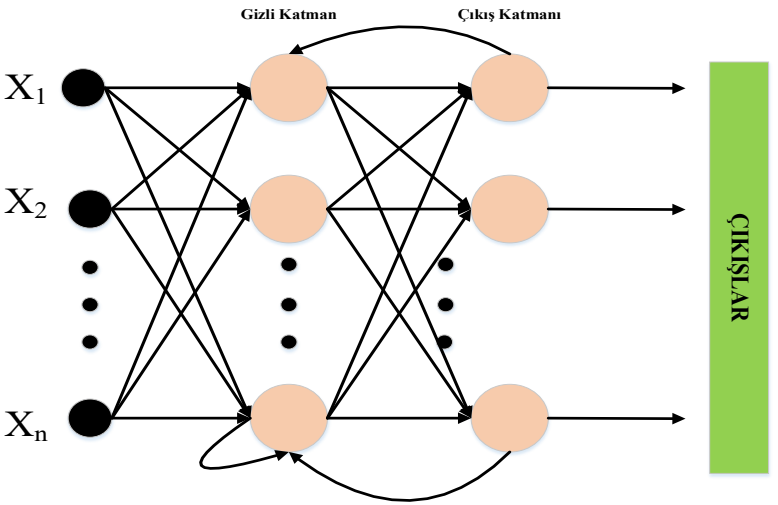
İleri beslemeli yapay sinir ağları, sinir ağı mimarilerinin en yaygın kullanılan türlerinden biridir. Bu yapılarda bilgi akışı yalnızca bir yöndedir: giriş katmanından başlayarak gizli katman(lar) aracılığıyla çıkış katmanına doğru ilerler. Geri dönüş bağlantılarının olmaması, bu ağları özellikle sabit boyutlu veri setleri ve bağımsız örnekler üzerinde yapılan sınıflandırma ve regresyon görevleri için ideal hâle getirir. Bu bölümde, ileri beslemeli ağların yapısal özellikleri, uygulama alanları ve sınırlılıkları ele alınacaktır.

Teknik olarak, ileri beslemeli yapılarda girdi verileri, giriş katmanına uygulandıktan sonra herhangi bir geri döngü olmaksızın doğrudan bir sonraki katmana iletilir. Bu doğrusal akış yapısı sayesinde eğitim süreci nispeten daha stabildir ve hesaplama açısından daha verimlidir. Literatürde, bu tür ağların sınıflandırma ve regresyon problemlerinde yüksek başarı gösterdiği bildirilmiştir [10,11].

Geri Beslemeli Ağlar

Zaman bağımlı verilerin modellenmesinde ileri beslemeli yapılar yetersiz kaldığında, daha dinamik ve belleğe sahip mimariler olan geri beslemeli yapay sinir ağları (Recurrent Neural Networks-RNN) devreye girer. Bu mimariler, önceki zaman adımlarındaki bilgileri iç döngüler aracılığıyla ağ içinde tutarak, sıralı veriler arasındaki bağımlılıkları öğrenebilme yeteneği sunar. Özellikle doğal dil işleme, zaman serisi tahmini ve konuşma tanıma gibi alanlarda yaygın olarak kullanılan bu yapılar, modelin geçmiş girdilere dair hafızasını koruyarak öğrenme sürecini derinleştirir. Şekil 3'de geri beslemeli ağ yapısı görülmektedir [10,11]

Şekil 3. *Çok katmanlı sinir ağı mimarisi*



Yapay Sinir Ağı Mimarilerinin Karşılaştırılması

Yapay sinir ağı mimarilerinin uygulama başarısı, doğrudan kullanılan yapının veri türü, problem tipi ve öğrenme hedefi ile olan uyumuna bağlıdır. Her bir mimari, kendine özgü avantajlar sunarken aynı zamanda belirli sınırlamalara da sahiptir. Örneğin, basit doğrusal problemler için tek katmanlı yapılar yeterli olabilirken;

karmaşık örüntülerin tanınması veya zaman bağımlı verilerin modellenmesi gibi durumlarda çok katmanlı ya da geri beslemeli mimariler tercih edilmelidir. Mimari seçim sürecinde, sadece doğruluk oranları değil; aynı zamanda veri boyutu, eğitim süresi, hesaplama maliyeti, overfitting riski, yorumlanabilirlik düzeyi ve genelleme yeteneği gibi faktörler de dikkate alınmalıdır. Özellikle uygulama alanı sağlık, finans veya otonom sistemler gibi yüksek hassasiyet gerektiren alanlarsa, mimari yapının özelliklerine dair bilinçli tercihler yapılması büyük önem taşır.

Aşağıda, literatürde en sık karşılaşılan yapay sinir ağı mimarileri; katman yapısı, veri akış yönü, avantajları ve sınırlılıkları temelinde karşılaştırmalı olarak sunulmuştur [12].

Tablo 1. *Yaygın Yapay Sinir Ağı Mimarilerinin Yapısal Özellikler, Avantajlar ve Sınırlılıklar Açısından Karşılaştırılması*

Mimari Türü	**Katman Yapısı**	**Veri Akışı**	**Avantajlar**	**Sınırlılıklar**
Tek Katmanlı YSA	Giriş - Çıkış	İleri yönlü	Basit yapısı, hızlı hesaplama	Sadece lineer ayrılabilir verilerde başarılı
Çok Katmanlı YSA	Giriş - Gizli - Çıkış	İleri yönlü	Karmaşık örüntüleri öğrenebilme	Fazla sayıda parametre, overfitting riski
İleri Beslemeli YSA	Giriş → Çıkış	Tek yönlü	Regresyon ve sınıflandırmada yaygın kullanım	Zaman bağımlı verilerde etkisiz
Geri Beslemeli YSA	Giriş ↔ Hafıza ↔ Çıkış	Zaman geri beslemeli	Bellek mekanizması, zaman serisi ve sıralı veri analizi	Karmaşık yapı, eğitim sürecinde kararsızlıklar yaşanabilir

Bu tablo, hangi mimarinin hangi veri türü ve problem yapısı için uygun olduğunu özetlemektedir. Örneğin, statik veri kümelerinde ileri beslemeli ağlar tercih edilirken; zaman bağımlı veri setlerinde geri beslemeli ağlar (RNN, LSTM gibi) daha etkilidir. Tek katmanlı yapılar, düşük karmaşıklıklı görevlerde yeterli

olabilirken; karmaşık yapay zekâ görevleri için çok katmanlı mimariler daha uygundur. Mimari seçiminde yalnızca görev tipi değil; aynı zamanda hesaplama maliyeti, eğitim süresi, genelleme kapasitesi ve yorumlanabilirlik gibi faktörler de dikkate alınmalıdır. Bu nedenle, farklı mimarilerin avantaj ve sınırlılıklarına hâkim olmak, uygulayıcılar açısından etkili model seçimi ve performans optimizasyonu açısından kritik bir karar destek mekanizması sunar.

Yapay Sinir Ağı Mimarilerinin Uygulama Alanları

Yapay sinir ağları, günümüzde sağlık, finans, savunma, ulaşım, tarım ve eğitim gibi birçok sektörde etkin biçimde kullanılmakta; farklı mimari yapıların sunduğu avantajlar doğrultusunda çeşitli problem türlerine adapte edilebilmektedir. Her bir mimari, belirli veri yapıları ve görev türleri için daha etkili sonuçlar üretme potansiyeline sahiptir. Bu bölümde, en sık kullanılan YSA mimarilerinin öne çıktığı uygulama alanları örneklerle açıklanacaktır.

- Görüntü İşleme ve Bilgisayarla Görü

Özellikle Evrişimli Sinir Ağları (CNN), medikal görüntü analizi, yüz tanıma, nesne tespiti ve uydu görüntüsü yorumlama gibi görsel veri gerektiren görevlerde yaygın olarak kullanılmaktadır [13]. Katmanlı yapıları sayesinde düşük seviyeli özniteliklerden yüksek seviyeli kavramsal temsillere ulaşabilmektedirler.

- Doğal Dil İşleme (NLP)

Geri beslemeli ağ yapıları (RNN, LSTM), metin sınıflandırma, makine çevirisi, duygu analizi ve konuşma tanıma gibi zaman bağımlı görevlerde yüksek performans göstermektedir. Bu mimariler, girdiler arasındaki ardışık bağımlılıkları öğrenebilme kabiliyetine sahiptir.

- Finansal Tahmin ve Piyasa Analizi

Döviz kuru, hisse senedi fiyatları ve ekonomik göstergelerin tahmininde, zaman serisi verilerinin modellenmesi amacıyla hem ileri beslemeli hem de geri beslemeli yapılar (özellikle LSTM) kullanılmaktadır.

- Sağlık ve Tıbbi Tanı Sistemleri

YSA modelleri; medikal görüntü analizi, hastalık sınıflandırması, biyobelirteç analizi ve tedavi süreci optimizasyonu gibi birçok alanda kullanılmaktadır. CNN mimarileri röntgen, MR ve biyopsi gibi görüntülerin analizinde, RNN tabanlı yapılar ise hasta geçmişine dayalı tahmin ve risk analizlerinde öne çıkmaktadır.

- Endüstriyel Otomasyon ve Kalite Kontrol

Makine arızası tespiti, üretim hattı optimizasyonu ve görüntü tabanlı kalite denetimi gibi süreçlerde ileri beslemeli ağlar, hızlı hesaplama ve düşük gecikme avantajları nedeniyle tercih edilmektedir. Özellikle anomali tespiti ve sınıflandırma görevlerinde başarıyla kullanılmaktadır.

- Otonom Sistemler ve Robotik

Çevresel algı, yol planlama ve hareket kontrolü gibi görevlerde CNN ve RNN mimarileri, görsel verilerin işlenmesi ve sıralı kontrol komutlarının öğrenilmesinde önemli roller üstlenmektedir.

Bu uygulama alanları, yapay sinir ağı mimarilerinin disiplinler arası doğasını ve farklı görev yapılarına gösterdiği yüksek adaptasyon kabiliyetini açıkça ortaya koymaktadır. Uygulamanın doğası gereği, mimari seçimi yapılırken probleme özgü veri yapısı, öğrenme hedefi ve sistem gereksinimleri mutlaka dikkate alınmalıdır.

SONUÇ

Yapay sinir ağı mimarileri, farklı veri türlerine ve problem yapısına uyarlanabilir esnek yapıları sayesinde günümüz yapay zekâ uygulamalarının temelini oluşturmaktadır. Bu bölümde ele alınan tek katmanlı, çok katmanlı, ileri beslemeli ve geri beslemeli yapılar; hem mimari düzeyde yapısal farklılıkları hem de uygulama alanlarındaki etkinlikleri açısından karşılaştırmalı olarak incelenmiştir. Elde edilen bulgular, sinir ağı mimarilerinin başarısının yalnızca katman sayısı gibi yapısal özelliklerle değil; aynı zamanda problem türüne, veri yapısına ve sistem gereksinimlerine göre yapılan mimari tercihle doğrudan ilişkili olduğunu göstermektedir. Basit sınıflandırma problemlerinde tek katmanlı yapılar düşük hesaplama maliyetiyle yeterli sonuçlar verirken, karmaşık örüntülerin öğrenilmesinde çok katmanlı yapılar belirgin avantaj sunmaktadır. Zaman bağımlı veri analizlerinde ise geri beslemeli ağlar –özellikle LSTM gibi gelişmiş türevleri– belleğe dayalı yapı sayesinde güçlü bir öğrenme kapasitesi ortaya koymaktadır. Farklı sektörlerdeki uygulamalar incelendiğinde, sağlık, finans, doğal dil işleme, görüntü işleme ve otomasyon gibi alanlarda YSA mimarilerinin yüksek düzeyde etkililik sağladığı görülmektedir.

Gelecekte, YSA mimarilerinin daha açıklanabilir, ölçeklenebilir ve veri etkin hâle getirilmesi, derin öğrenme sistemlerinin güvenli ve yorumlanabilir biçimde yaygınlaşabilmesi için kritik bir araştırma gündemidir. Mimari seçimin sadece doğruluk değil, enerji verimliliği, işlem süresi ve etik uygunluk gibi çok boyutlu kriterlerle birlikte değerlendirilmesi gereklidir.

Sonuç olarak, yapay sinir ağı mimarilerinin teorik temellerinin derinlemesine kavranması, bu yapıların başarılı şekilde modellenmesi ve uygun görevlerle eşleştirilmesi, hem akademik araştırmalar hem de uygulama odaklı projeler açısından vazgeçilmezdir. Bu doğrultuda, YSA mimarilerine dair kuramsal bir

çerçeve ve uygulamalı farkındalık kazandırmak, bu çalışmanın temel hedefi olmuştur.

KAYNAKLAR

[1] Bingham, G., & Miikkulainen, R. (2023). Efficient activation function optimization through surrogate modeling. arXiv. https://arxiv.org/abs/2301.05785

[2] Lee, M. (2023). GELU activation function in deep learning: A comprehensive mathematical analysis and performance. arXiv. https://arxiv.org/abs/2305.12073

[3] Schneider, J., & Vlachos, M. (2023). A survey of deep learning: From activations to transformers. arXiv. https://arxiv.org/abs/2302.00722

[4] Preti, L. M., Ardito, V., Compagni, A., Petracca, F., & Cappellaro, G. (2024). Implementation of machine learning applications in health care organizations: A scoping review. Journal of Medical Internet Research, 26, e55897. https://www.jmir.org/2024/1/e55897/

[5] White, C., Safari, M., Sukthanker, R., Ru, B., Elsken, T., Zela, A., Dey, D., & Hutter, F. (2023). Neural architecture search: Insights from 1000 papers. arXiv. https://arxiv.org/abs/2301.08727

[6] Poli, M., Massaroli, S., Nguyen, E., Fu, D. Y., & Dao, T. (2023). Hyena hierarchy: Towards larger convolutional language models. arXiv. https://arxiv.org/abs/2304.01952

[7] Gu, A., Goel, K., & Re, C. (2023). Mamba: Linear-time sequence modeling with selective state spaces. arXiv. https://arxiv.org/abs/2302.06696

[8] Onyekpe, U., Palade, V., & Wani, A. (Eds.). (2024). Recent advances in deep learning applications: New techniques and practical examples. Routledge. https://www.routledge.com/Recent-Advances-in-Deep-Learning-Applications-New-Techniques-and-Practical-Examples/Onyekpe-Palade-Wani/p/book/9781032944623.

[9] Z. İlkılıç Aytaç, *Tiroit İnce İğne Aspirasyon Biyopsilerinde Papiller Tiroit Karsinomların Metasezgisel Temelli Evrişimsel Sinir Ağı ile Tespit Edilmesi*, Doktora Tezi, Fırat Üniversitesi Fen Bilimleri Enstitüsü, Mekatronik Mühendisliği Anabilim Dalı, Elâzığ, 2024.

[10] Hatipoğlu P.U.(2016). Zaman Serilerinin Derin Öğrenme ile Sınıflandırılması, Yüksek Lisans Tezi, Ortadoğu Teknik Üniversitesi, Fen Bilimleri Enstitüsü.

[11] Learning and Soft Computing: (2021) Support Vector Machines, Neural Networks, and Fuzzy Logic Models. MIT Press eBooks,IEEE Xplore, (n.d.). https://ieeexplore.ieee.org/book/6267294.

[12] Schmidhuber, J. (2015). Deep learning in neural networks: An overview. Neural Networks, 61, 85–117. https://doi.org/10.1016/j.neunet.2014.09.003.

[13] Alzubaidi, A.H., Zhang, J., Humaidi, A.J. et al. (2021). "Review of deep learning: Concepts, CNN architectures, challenges, applications, future directions". Expert Systems with Applications, 115860. https://doi.org/10.1016/j.eswa.2021.115860

CHAPTER 4

Yazılım Proje Yönetiminde Başarı ve Başarısızlık Faktörlerinin Karşılaştırmalı Değerlendirmesi

İsmail İSTAY[1]
Esra ÇALIK BAYAZIT[2]
Kıvanç KIŞLAL[3]
Buket DOĞAN[4]

Giriş

Proje yönetimi, belirli hedeflere ulaşmak amacıyla zaman, maliyet ve kalite gibi kısıtlar altında yürütülen faaliyetlerin planlanması, yürütülmesi ve kontrol edilmesi süreci olarak tanımlanmaktadır. Projeler yapısı gereği belirli bir başlangıç ve bitiş tarihine, belirli hedeflere, sınırlı bir bütçeye sahip olma, birbirine bağlı karmaşık görevler veya aktivitelerden oluşma gibi özelliklere

[1] Lisansüstü Öğrenci, Marmara Üniversitesi, Bilgisayar Mühendisliği, Orcid: 0009-0009-0290-9799
[2] Dr.Öğr. Üyesi, Fatih Sultan Mehmet Vakıf Üniversitesi, Bilgisayar Mühendisliği, Orcid:0000-0002-6813-1037
[3] Mühendis, DIP Bilgisayar Yazılım Ticaret Anonim Şirketi, Orcid: 0009-0008-9015-5773
[4] Doçent, Marmara Üniversitesi, Bilgisayar Mühendisliği, Orcid: 000-0003-1062-2439

sahiptir (Sudhakar, 2012). Özellikle teknolojik dönüşümün hızlandığı dijital çağda, artan karmaşıklık ve belirsizlik koşulları altında yazılım projeleri hem kapsamları hem de dinamik yapıları gereği proje yönetimi literatüründe özel bir yer edinmiştir. Geleneksel projelere kıyasla, yazılım projeleri daha yüksek belirsizlik, değişken gereksinimler ve kullanıcı odaklı teslimat beklentileriyle karakterize edilmekte ve yüksek rekabet koşullarında riskler barındırmaktadır. Bu nedenle, projelerin başarılı bir şekilde yönetilmesi organizasyonlar için çok önemlidir. Yazılım projeleri belli bir faydayı sağlayabilmek amacıyla başlatılmaktadır fakat başarısızlıklar ve tahminlerin aşılması yazılımın bir ürün haline getirilememesiyle sonuçlanabilmektedir. Araştırmalar ve endüstri raporları, yazılım projelerinin yüksek başarısızlık oranlarının genellikle proje zamanlaması, bütçe, kalite gibi unsurların yönetilmesindeki zorluklardan kaynaklanmakta olduğunu göstermektedir.

Yazılım proje yönetiminde başarı, projenin nasıl tanımlandığına göre de farklılık göstermektedir. Standish Group (2015) raporuna göre; geleneksel başarı tanımı esas alındığında, yani projelerin zamanında, belirlenen bütçe içinde ve planlanan özelliklerle tamamlanması durumunda, başarı oranı 2011-2015 yılları arasında incelenen projelerde ortalama yüzde 37,8 olarak gerçekleşmiştir. Modern başarı tanımına göre ise, yani projelerin zamanında, bütçeye uygun ve kullanıcıyı tatmin edecek şekilde tamamlanması durumunda, bu oran yüzde 28,8'e düşmektedir. Bu veriler, yazılım projelerinde hem teknik hedeflere ulaşmanın hem de kullanıcı beklentilerini karşılamanın önemli bir zorluk olduğunu ortaya koymaktadır (Standish Group Rapor, 2015).

Farklı projelerin, başarı kriterleri ve faktörlerinin çeşitli kombinasyonlarını belirleyen, hedefler, paydaşlar, çevre ve riskler gibi proje bağlamı açısından dikkate alınması gereken belirli özellikleri vardır. Proje başarısının dinamik yapısını anlamak için

araştırmacılar, projelerin nasıl gerçekleştirildiği, hangi bağlamlarda uygulandığı ve sonuçlarının nasıl değerlendirilmesi gerektiği gibi çeşitli proje yönetimi alanlarını analiz etmektedirler. Son dönemdeki literatürün çoğu, önceki çalışmaların bulgularını özetlemekte ve projelerin "gerçek" başarı faktörlerini anlamaya çalışmakta ve aynı zamanda proje başarısının stratejik ve bütünsel bakış açısını vurgulamaktadır. Literatürde bahsedilen birçok proje başarı kriteri arasında, "Demir Üçgen" (iron triangle) başarı yaklaşımlarının temelini oluşturur çünkü bir projeyi zamanında, bütçe dahilinde ve performans spesifikasyonlarına uygun olarak tamamlamak için gerekli kriterlerin değerlendirilmesi kolaydır. Ancak, bu anlayış, proje başarısını derinlemesine değerlendirmek için yeterli değildir, çünkü bu parametreleri incelemek yalnızca kâra olan doğrudan katkıları gösterir, ancak projenin doğru bir şekilde uygulanıp uygulanmadığını dikkate almaz (Cserháti & Szabó, 2014). Yazılım proje yönetimi süreçleri temelinde başarıları ve başarısızlıkları etkileyen pek çok kavram ve uygulama bulunmaktadır. Gerçekleştirilen çalışmada bu kavramlar Standish Grup'un (1995), (2015) raporları ve literatürdeki önemli çalışmalar ile derlenerek ortaya konmaktadır.

1. Yazılım Proje Yönetiminde Başarı Kavramı

Yazılım projelerinin başarılı olmasına yol açan birçok faktör vardır. Bu faktörler, Tsun Chow ve Dac-Buu Cao tarafından genel başarı düzeyi kalite, kapsam, zaman ve maliyet kavramlarıyla ifade edilmiştir (Chow & Cao, 2008). Bu kavramlara bağlı olarak başarı ve başarısızlıkların kategorilendirilmesi yapılarak, organizasyonel, insan, süreç, teknik ve proje olmak üzere beş kategoriye ayrılmış ve Tablo 1'de gösterilmiştir. Proje misyonunun netliği, üst yönetim desteği ve teknik kaynakların mevcudiyetinin yüksek proje başarı oranlarına yol açtığı belirlenmiştir (Chow & Cao, 2008; Sudhakar, 2012).

Tablo 1: YPY Genel Başarı Kavramları

Organizasyonel	Güçlü yönetici desteği
	Kararlı sponsor veya yönetici
	Hiyerarşik yerine işbirlikçi organizasyonel kültür
	Yüz yüze iletişime yüksek değer veren sözlü kültür
	Çevik metodolojinin evrensel olarak kabul edildiği organizasyonlar
	Tüm ekibin aynı ortamda çalışması
	Çevik tarzda uygun çalışma ortamı
	Çevik için uygun ödül sistemi
İnsan	Yüksek yeterlilik ve uzmanlığa sahip takım üyeleri
	Büyük motivasyona sahip takım üyeleri
	Çevik süreç konusunda bilgili yöneticiler
	Hafif dokunuşlu veya uyumlu yönetim tarzına sahip yöneticiler
	Tutarlı, kendi kendini organize eden takım çalışması
	İyi müşteri ilişkisi
Süreç	Çevik odaklı gereksinim yönetimi sürecini takip
	Çevik odaklı proje yönetimi sürecini takip
	Çevik odaklı konfigürasyon yönetimi sürecini takip
	Güçlü iletişim odağı ile günlük yüz yüze toplantılar
	Düzenli çalışma programına saygı – fazla mesai yok
	Güçlü müşteri kararlılığı ve varlığı
	Müşterinin tam yetkiye sahip olması
Teknik	Önceden tanımlanmış kodlama standartları
	Basit tasarım izleme
	Sıkı refaktoring (yeniden yapılandırma) faaliyetleri
	Doğru miktarda dokümantasyon
	Yazılımın düzenli teslimatı
	En önemli özelliklerin önce teslim edilmesi
	Doğru entegrasyon testi
	Takıma uygun teknik eğitim verilmesi
Proje	Projenin hayati öneme sahip olmaması
	Proje türünün ortaya çıkan gereksinimlere karşı değişken kapsamlı olması
	Dinamik, hızlandırılmış takvime sahip projeler
	Küçük ekiplere sahip projeler
	Birden fazla bağımsız ekibin bulunmadığı projeler
	Önceden yapılan maliyet değerlendirmesi olan projeler

	Önceden yapılan risk analizine sahip projeler

Campanelli ve arkadaşları (2017) tarafından çevik dönüşüm için başarı faktörleri gruplar halinde bir araya getirilerek değerlendirmesi yapılmıştır. Bu değerlendirmeyi, ABD'nin Chicago şehrinde bulunan bir şirkette gerçekleştirilen bir vaka çalışmasıyla doğrulanmıştır. Değerlendirme, organizasyonun bağlamını (kültür, mevcut durum, hedefler, hiyerarşi) göz önünde bulundurarak, başarı faktörlerinin uygulanma zorluğuna göre bir zorluk sıralaması ürettiği vurgulanmıştır. Şirket A için yapılan sıralama, ölçüm modeli, eğitim, çevik liderler, yeni zihniyet/roller ve yönetim tarzı değişiklikleri ile merkezi olmayan karar alma gibi faktörlerin uygulanmasının daha zor olduğunu gösterdi. Buna karşılık, müşteri katılımı, kendi kendine organize olan ekipler ve proje yöneticilerinin zihniyet değişikliği gibi faktörlerin daha kolay uygulandığı tespit edildi (Campanelli, Neto, & Parreiras, 2017).

Campanelli ve arkadaşları (2017) çevik dönüşüm için başarı faktörleri gruplandırması organizasyonel, takım, süreç, yönetim, müşteri ve araçlar olmak üzere altı kategoriye ayrılmış ve Tablo 2'de gösterilmiştir (Campanelli, Neto, & Parreiras, 2017).

Tablo 2: Çevik Dönüşüm İçin Başarı Faktörleri

Organizasyonel	Eğitim
	Çevik liderler
	Yeni zihniyet/roller
	Bilgi paylaşımı
	İş hedefleri
	Çevik metodları benimseme motivasyonu
	Organizasyonda iletişim akışı
	Koçluk/mentörlük
	Kültürel değişim
Takım	Dağıtık ekipler
	Teknik beceriler/aktiviteler
	Güvenilir ilişkiler kurabilme
	Ekip katılımı
	İşbirliği
	Kendi kendine organize ekipler

Süreç	Ölçüm modeli
	Hafif dokümantasyon
	Organizasyonel bağlamla uyumlu süreç
Yönetim	Yönetim tarzı değişikliği ve merkezi olmayan karar alma
	Yönetim desteği
	Proje yöneticilerinin zihniyet değişimi
Müşteri	Müşteri katılımı
Araçlar	Araç seti

Gabriella Cserháti ve Lajos Szabó'e göre başarı kriterleri ve başarı faktörleri ulaşılan sonuç ve süreci kapsamaktadır. Başarı kriterleri ortaya çıkarılacak ürünün sonucunda paydaş memnuniyeti, proje performans ve projenin belirtilen hedeflerinin karşılanmasına, zamanında teslimat, başta belirlenen bütçe ile işin bitirilmesi, sözleşmesel yükümlülüklerin yerine getirilmesi, iş başarısının elde edilmesi, yönetim süreçleri, proje kaynakları, proje ekibi, örgütsel kültür, iletişim ve işbirliği detaylarını içermektedir. Başarı faktörleri projenin başarılı bir sonuca ulaşmasını sağlayan süreçlerin bütünüdür. Projenin başarı şansını garanti altına almak için başarı kriterleri özel ve sürekli bir dikkat gerekmektedir; aksi takdirde, bu faktörler ciddiye alınmazsa, bir projenin başarısız olmasına katkıda bulunabilir. Bir proje başarılı bir şekilde tamamlanırken, diğer bir proje benzer proje yönetimi yöntemlerine sahip olmasına rağmen başarısız olabilir. Buna göre, başarılı bir projede, uygulama esasen proje yönetim teknikleri, yapıları ve sistemleriyle değil, proje katılımcıları ve paydaşlarına yönelik bir yaklaşım ile belirlenir. Gabriella Cserháti ve Lajos Szabó tarafından sunulan başarı faktörleri ve başarı kriterleri şartları **Tablo 3' te** gösterilmiştir (Cserháti & Szabó, 2014).

Tablo 3: Başarı Faktörleri ve Başarı Kriterleri

	Proje Performans Hedeflerinin Karşılanması	Zamanında Teslimat
		Bütçe İçerisinde Teslimat
		Sözleşmesel Yükümlülüklerin Yerine Getirilmesi
		İş Başarısının Elde Edilmesi

Başarı Kriterleri	Paydaşların Memnuniyeti	Proje Sahibi, Kullanıcılar, Proje Ekibi, Yükleniciler, Sponsorlar, Yerel ve Ulusal Paydaşların Memnuniyeti
		Sponsorlar, Yerel ve Ulusal Paydaşlarla Uzun Vadeli Ortaklıklar
Başarı faktörleri	Yönetim Süreçleri	Hedef Yapısının Oluşturulması
		Görev Yapısının Oluşturulması
		Proje Planlarının İyileştirilmesi
		Kapsam ve Sorumlulukların Tanımlanması
	Proje Kaynakları	Yüklenicilerin Seçimi
		Yüklenicilerin Kontrolü
		Alt Yüklenicilerle Sorumluluk Paylaşımı
		Alt Yüklenicilerde Finansal Koşullar
	Proje Ekibi	Proje Liderinin Yeterlilik ve Becerileri
		Ekip Üyelerinin Yeterlilik ve Becerileri
		Proje Ekibinin Bağlılığı
	Örgütsel Kültür	Proje Ekibi İçindeki İletişim
		Proje Ekibi İçinde Bilgi Paylaşımı
		Takım Çalışmasının Desteklenmesi
		Bireysel Çabaların Desteklenmesi
		Örgütsel Öğrenme
	İletişim ve İşbirliği	Proje Sahibi, Kullanıcılar, Yükleniciler ve Sponsorlarla İletişim
		Yerel ve Ulusal Paydaşlarla Ortaklıklar

Standish Grup,ncak bu üç unsur sağlandığında başarı şansı çok daha yüksektir. Bu unsurlar olmadan, başarısızlık ihtimali yüksek bir şekilde artar. Standish Grup'a göre proje başarısı, birçok faktörün bir araya gelmesiyle sağlanır ve bu faktörlerin her biri, projenin başarıya ulaşmasında farklı bir rol oynar. Bu faktörlerden en önemlileri sırasıyla %15,9 ile kullanıcı katılımı, proje başarısını en çok etkileyen faktördür. Kullanıcıların projeye aktif olarak dahil edilmesi, projenin ihtiyaçlarının daha iyi anlaşılmasını ve sonuçların daha tatmin edici olmasını sağlar. Yönetici desteği %13,9 olarak, projenin kaynaklara erişimini kolaylaştırır ve ekibin motivasyonunu artırır. Üst düzey yöneticilerin projeye sağladığı destek, projenin önceliklendirilmesi ve engellerin aşılması açısından kritik öneme

sahiptir. Net gereksinim açıklanması %13 ile projenin başında gereksinimlerin net bir şekilde tanımlanmasını ifade eder. Standish Grup'a göre projelerde zorluk yaşanmasına neden olan faktörler, proje yönetimi süreçlerinde sıkça karşılaşılan sorunlardır ve bu faktörlerin her biri, projenin başarısını olumsuz yönde etkilemektedir. Bu faktörlerden en önemlileri sırasıyla %12,8 ile kullanıcı geri bildiriminin eksikliği, projede yaşanan en büyük zorluklardan biridir. Kullanıcıların projeye yeterince dahil edilmemesi veya geri bildirimlerinin alınmaması, projenin ihtiyaçlarının tam olarak karşılanmamasına ve sonuçların tatmin edici olmamasına neden olur. Eksik gereksinimler ve spesifikasyonlar %12,3 ile projenin başında gereksinimlerin tam olarak tanımlanmaması, projenin yanlış yönde ilerlemesine ve gereksiz revizyonlara yol açar. Değişen gereksinimler ve spesifikasyonlar %11,8 ile proje sürecinde sıkça karşılaşılan bir diğer sorundur. Gereksinimlerin sürekli değişmesi, proje planlarının sık sık revize edilmesine ve ekstra maliyetlere neden olur. Standish Grup'a göre projelerde zayıflık (kusur) yaşanmasına neden olan faktörler, proje yönetimi süreçlerinde sıkça karşılaşılan sorunlardır ve bu faktörlerin her biri, projenin başarısını olumsuz yönde etkilemektedir. Bu faktörlerden en önemlileri sırasıyla %13,1 ile eksik gereksinimler, projede yaşanan en büyük zorluklardan biridir. Projenin başında gereksinimlerin tam olarak tanımlanmaması, projenin yanlış yönde ilerlemesine ve gereksiz revizyonlara neden olur. Kullanıcı katılımının eksikliği %12,4 ile kullanıcıların projeye yeterince dahil edilmemesi, projenin ihtiyaçlarının tam olarak anlaşılmamasına ve sonuçların tatmin edici olmamasına yol açar. Kaynak eksiklikleri %10,6 ile proje için gerekli olan insan kaynağı, bütçe veya ekipman eksikliği, projenin zamanında tamamlanmasını engeller (Standish, 1995). Ayrıca, Standish grup (2015) raporuna göre projelerin başarısında birçok faktör farklı ağırlıklarda rol oynamaktadır. En yüksek etkiye sahip faktörler arasında %15 oranlarıyla üst düzey yönetici desteği, duygusal olgunluk, kullanıcı

katılımı ve optimizasyon yer almaktadır. Bu unsurlar, stratejik yönlendirme, ekip içi uyum, kullanıcı beklentilerine uygunluk ve süreç iyileştirmesi açısından kritik önemdedir. Yeterli kaynaklar %10'luk bir etkiye sahipken, standart mimarinin katkısı %8 olarak değerlendirilmiştir. Çevik süreçlerin esnekliği %7'lik bir payla ölçülmüş, yalın uygulama yaklaşımları %6 oranında etkili bulunmuştur. Proje yönetimi uzmanlığı %5, açık iş hedeflerinin netliği ise proje başarısına %4'lük katkı sağlamaktadır (Standish, 2015). Bu veriler ışığında, 1995 ve 2015 tarihli Standish Group raporları karşılaştırıldığında bazı temel benzerliklerin yanı sıra dikkat çekici farklılıklar da göze çarpmaktadır. Her iki raporda da kullanıcı katılımı, yönetici desteği ve net gereksinimlerin tanımlanması proje başarısının temel unsurları olarak öne çıkmaktadır. Bu durum, proje yönetiminde insan faktörünün ve iletişimin yıllar içinde değişmeyen kritik öneme sahip olduğunu göstermektedir.

1995 raporunda daha çok kullanıcı geri bildirimi, eksik gereksinimler, değişen ihtiyaçlar ve kaynak yetersizlikleri gibi somut ve operasyonel sorunlar ön plana çıkarken; 2015 raporunda stratejik yönetim, ekip uyumu ve süreç iyileştirme gibi daha yapısal ve yönetsel unsurların etkisi vurgulanmıştır.

Yazılım Proje Yönetiminde Başarısızlık Faktörleri

Başarısızlık, yazılım projelerinin önemli bir kısmında karşılaşılan bir sorundur. Yazılım projelerinin başarısız olmasına yol açan birçok faktör vardır. Bu faktörler, Tsun Chow ve Dac-Buu Cao tarafından organizasyonel, insanlar, süreç ve teknik olmak üzere dört kategoriye ayrılmıştır (Chow & Cao, 2008). Bunlar Tablo 4'te belirtildiği gibi; organizasyonel, insan, süreç ve teknik faktörler olarak sınıflandırılmıştır. Organizasyonel faktörler, yöneticilerin projeye yeterli destek vermemesi, yönetim kararlılığının eksikliği, geleneksel veya politik kurumsal kültür, organizasyonun aşırı büyük

olması ve çevik yöntemleri destekleyecek esnek yapının bulunmaması gibi unsurları içermektedir. Bu tür yapısal zayıflıklar, projenin yönünün belirsizleşmesine ve stratejik uyumsuzluklara neden olabilir. İnsan kaynaklı faktörler ise ekibin gerekli teknik bilgi ve beceriye sahip olmaması, proje yönetimi yetkinliklerinin yetersizliği, takım çalışması eksikliği, değişime karşı direnç ve müşteriyle zayıf ilişkiler gibi unsurları kapsamaktadır. Bu tür insani eksiklikler, iletişim kopuklukları, motivasyon düşüklüğü ve iş birliği sorunlarıyla birlikte proje performansını olumsuz etkiler. Süreç faktörleri arasında proje kapsamı, gereksinimleri ve planlamasının belirsiz olması, çevik ilerleme ve izleme mekanizmalarının eksikliği ile müşteri katılımının sınırlı ya da rolünün belirsiz olması yer almaktadır. Bu durumlar, projenin yönünü kaybetmesine ve beklentilerle sonuçlar arasında büyük farkların oluşmasına yol açar. Son olarak, teknik faktörler olarak çevik uygulama setlerinin pratikte yeterince uygulanamaması ve uygun olmayan teknoloji ya da araç kullanımı, yazılım geliştirme sürecinde verimliliği düşürerek kaliteyi olumsuz etkiler. Bu dört kategori birlikte değerlendirildiğinde, yazılım projelerinin başarısızlıkla sonuçlanmasına neden olan faktörlerin sadece teknik değil; organizasyonel yapı, insan unsuru ve süreç yönetimiyle de yakından ilişkili olduğu görülmektedir.

Tablo 4: Yazılım Proje Yönetiminde Genel Başarısızlık Faktörleri

Organizasyonel	Yöneticilerin sponsorluk eksikliği
	Yönetimin kararlılık eksikliği
	Organizasyonel kültürün çok geleneksel olması
	Organizasyonel kültürün çok politik olması
	Organizasyonel boyutun çok büyük olması
	Çevik lojistik düzenlemelerin eksikliği
İnsan	Gerekli beceri setinin eksikliği
	Proje yönetimi yeterliliğinin eksikliği
	Takım çalışması eksikliği
	Gruplardan veya bireylerden direnç
	Kötü müşteri ilişkisi
Süreç	Proje kapsamının belirsizliği

	Proje gereksinimlerinin belirsizliği
	Proje planlamasının belirsizliği
	Çevik ilerleme izleme mekanizmasının eksikliği
	Müşteri varlığının eksikliği
	Müşteri rolünün belirsizliği
Teknik	Çevik uygulama setinin pratik eksikliği
	Teknoloji ve araçların uygunsuzluğu

Paydaş Yönetimi ve Katılımının Proje Yönetimine Etkileri

Proje takım katılımı, özellikle pazarlama, satış, operatörler ve müşteriler, ürün Ar-Ge'sine dahil olmalıdır. Bu, tasarımcıların ürün ihtiyaçlarını anlamalarına yardımcı olur. İş gücü ve müşteriler, ürün geliştirme süreci hakkında bilgilendirilmelidir. Ürünün Ar-Ge planını öğrendikten sonra, satış elemanı işlevlerini pazarlayabilir. Müşteri bu planı anlayarak yorumlar sunabilir ve böylece ürün değerini iyileştirebilir (Cserháti & Szabó, 2014), (Tam, Moura, Oliveira, & Varajão, 2020).

Müşteri katılımı, müşteri temsilcileri ile şirket arasındaki etkileşimleri projelerin süresi boyunca devam eder. Müşteri katılım düzeyinin, yazılım geliştirme projelerinin başarısıyla yakından ilişkili olduğunu ve bu nedenle projelerin daha yüksek müşteri katılımı seviyeleriyle daha başarılı olduğunu bildirmiştir (Cserháti & Szabó, 2014), (Kenny, 2004). Müşterilerin veya kullanıcıların katılımına izin verilmeli ve kullanıcı deneyimini tam olarak anlamaları sağlanmalıdır. Bunun avantajları, ürün geliştirme hızını hızlandırmak, aidiyet duygusunu artırmak ve ürün satışlarını iyileştirmektir (Tam, Moura, Oliveira, & Varajão, 2020). Müşteriler ve istemciler, bilgi sistemini sipariş eden, bunun için ödeme yapacak ve gereksinimlerini ilk olarak belirleyen kişilerdir. Müşteriler aynı zamanda bilgi sisteminin kullanıcıları da olabilir; yani, doğrudan bilgi sistemini kullanan ve onu operasyonel hale getirenlerdir. Ancak kullanıcılar genellikle müşteriler değildir (Kautz, 2011).

Kullanıcı katkısı çevresel faktörlerin içerisinde bulunan müşteri katılımı, tedarikçi ortaklığı, dış çevre olayları, kullanıcı

katılımı ve müşteri kabulü değerlendirildiğinde en çok karşılaşılan etken kullanıcı katılımı olarak bulunmuştur (Sudhakar, 2012). Kullanıcıların kültürel seviyesi görece yüksekse, proje sürecinde ilgili bilgileri edinme becerileri güçlü olacaktır ve bu da onlara ilgili tartışmalara ve toplantılara katılma fırsatı sunabilir, bu da projenin sorunsuz bir şekilde uygulanmasına büyük yardım sağlayabilir (Tam, Moura, Oliveira, & Varajão, 2020).

Paydaşların katkısı kapsamında proje sahibinin önemini kapsayan dört başarı koşulu vurgulanmıştır:

1. Başarı kriterleri, proje başlamadan önce paydaşlarla kararlaştırılmalı ve proje boyunca yapılandırma gözden geçirme noktalarında tekrar gözden geçirilmelidir.

2. Proje sahibi ile proje yöneticisi arasında, her ikisinin de projeyi bir ortaklık olarak görerek, işbirliği yapan bir çalışma ilişkisi sürdürülmelidir.

3. Proje yöneticisine, projeyi nasıl başarıyla tamamlaması gerektiği konusunda rehberlik sağlansada, karşılaşılan öngörülemeyen durumlarla en iyi nasıl başa çıkacağı konusunda esneklik tanınmalıdır.

4. Proje sahibi, projenin performansına ilgi göstermelidir.

Bu dört durumun tamamı başarılı bir proje için gerçekleştirilmelidir (Kenny, 2004).

Proje Planlama ve Yönetimi

Yazılım projelerinde, işler genellikle ana geliştirme aşamalarına göre yapılandırılır. Bunlar gereksinimlerin tanımlanması, mimari ve detaylı tasarım, kod yazma ve hata ayıklama, test etme ve uygulamadır. RUP (Rational Unified Process) metodolojisi, 4 aşama sunar: Başlangıç (Inception), Tasarım (Elaboration), İnşa (Construction) ve Geçiş (Transition). Bu metodoloji, süreçleri çalışanlar, faaliyet, artefaktlar ve iş akışları

açısından tanımlar. Ürün karmaşıksa, artımlara (increments), göreceli olarak bağımsız modüllere bölünerek kendi döngülerinde ayrı ayrı geliştirilir ve uygulanır (Polianskii & Chukalova, 2020).

Her somut yazılım geliştirme için bir proje planı oluşturulmalıdır. Bu plan, proje yöneticisi tarafından hazırlanır ve bir veya birden fazla süreç modeline dayanır (Chow & Cao, 2008). Süreç ilerledikçe, yazılım proje geliştirme planlaması Tablo 5'te belirtilen faaliyetleri benimser (Chow & Cao, 2008).

Tablo 5: Proje Planı Tanımla

Proje Tanımlaması	Değerlendirme	Risk Değerlendirmesi
Hedeflerin ve gereksinimlerin tanımlanması	Ürünün boyutlarının belirlenmesi	Risk analizi
Geliştirme süreçlerinin seçimi	Zaman planlaması (Takvim)	Risk değerlendirmesi
Çalışmanın tanımı	Maliyet tahmini ve bütçe	Risklere karşı reaksiyon planları

Tüm faaliyetlerin planlanmasının temeli, projeyi net bir şekilde tanımlanmış görevler veya faaliyetlere ayıran ve dolayısıyla projeyi tamamlamak için yapılması gereken işleri belirleyen WBS (Work Breakdown Structure) ya da proje hiyerarşi yapısıdır. Proje planının yapısı bir tablo yada grafik şeklinde olabilir (Chow & Cao, 2008).

Plan ve program proje yönetim etkileri içerisinde bulunan proje planlaması, proje kontrol mekanizması, proje takvimi, proje yöneticisinin yeterliliği ve açık proje hedefleri etkenlerinden en çok karşılaşılan etken proje planlamasıdır (Sudhakar, 2012). Proje planlaması ve düzenlemesi, proje uygulamasını ve kontrolünü rehberlik eder, yöneticilerin ve müşterilerin proje durumunu takip etmelerine olanak tanır. Proje geliştikçe, genel takvimin içeriği daha ayrıntılı hale getirilmelidir (Tam, Moura, Oliveira, & Varajão, 2020).

Proje kapsamını belirlemek, projenin sorunsuz ilerlemesinin temelidir; bu, proje kapsamını belirlemeyi ve proje ekibini oluşturmayı içerir. Projenin işlevsel ve zaman kapsamı ile proje ekibi üyeleri belirlenmelidir (Tam, Moura, Oliveira, & Varajão, 2020).

Proje hedef planlaması açık bir proje hedef planlaması, proje başlatılmasının birincil görevidir ve bir analiz aşamasından geçmeyi gerektirir. Projenin beklenen sonuçlarını ve bu sonuçlara ulaşmak için tamamlanması gereken çeşitli göstergeleri ve standartları netleştirmek gereklidir (Tam, Moura, Oliveira, & Varajão, 2020).

Süreç modeli bir yazılım süreç modelinin ana işlevleri, yazılım geliştirme ve evriminde yer alan aşamaların sırasını belirlemek ve bir aşamadan diğerine geçiş için geçiş kriterlerini oluşturmaktır. Bunlar, mevcut aşamanın tamamlanma kriterlerini ve bir sonraki aşama için seçim ve giriş kriterlerini içerir. Böylece, bir süreç modeli yazılım projesinin sonraki adımında ne yapılacağı ve bu faaliyetlerin ne kadar sürede yapılacağı ile ilgili sorulara yanıt arar. Karmaşık yazılım projeleri için metodoloji seçmek oldukça zor ve önemli bir karardır, çünkü proje için uygun tek bir yöntem olmayabilir, farklı yaklaşımların birlikte kullanımı da gerekebilmektedir. (Krasniqi, Ismajli, & Krasniqi, 2024). Yazılım süreç modelleri öncelikle, bir projenin ana görevlerini hangi sırayla (aşamalar, artırımlar, prototipler, doğrulama görevleri vb.) yerine getirmesi gerektiği konusunda rehberlik sağlarlar. Birçok yazılım projesi, geliştirme ve çevrim aşamalarını yanlış sırayla izledikleri için başarısız olabilmektedir (Zhang, Abdullah, & Rosli, 2024).

Projede kullanılan yöntemin başarılı olabilmesi için düzenli çalışma takviminin kullanılması da çok önemlidir. Standish group (2015) raporu ise 2011-2015 yılları arasında raporda incelenen projelerden %56'sının zaman aşımı yaşadığını ortaya koymaktadır (Standish, 2015). Proje yönetiminde zaman aşımı yaşanması önemli bir sorun olduğundan dolayı doğru planlama yapılması ve proje uygulaması sırasında proje ekibinin ilerlemelerinin izlenmesi son

derece önemlidir. Bu izlemelerde; düzenli iş ilerlemesi sağlamalı, ekip üyelerinin tüm görevlerin ilerlemesini bilmesini sağlamalı ve bir görev ilerleme yakma grafiği oluşturmalıdır. Bu, projenin sorunsuz bir şekilde tamamlanmasına yardımcı olur (Tam, Moura, Oliveira, & Varajão, 2020).

Zaman yönetimi yazılım projesinin geliştirilmesine harcanan saatlere bağlı olarak proje boyutu küçük, orta ve büyük olarak sınıflandırılabilmektedir. Bu sınıflandırma projelerin büyüklüğünü belirlemek için kullanılır. Küçük projeler 1-250 saat arası, orta projeler 251-5000 saat arası ve büyük projeler 5001 saat ve üzeri olan projeler olarak sınıflandırılabilmektedir (Chow & Cao, 2008).

Proje değişiklik yönetimi, yazılım geliştirme yaşam döngüsünün tamamında meydana gelir. Onarım içeriği, program modüllerinin yeniden tasarımından, bir kaynak dosyasının değişken değişimine kadar genişleyebilir, bu da yönetim değişikliğinin gerçek gelişim içindeki öncülüğünü gösterir (Tam, Moura, Oliveira, & Varajão, 2020). Bir yeniliğin uygulanmasıyla ilişkili belirsizlik düzeyini, bunun benimsenmesi için personelin gereken bilgi miktarı ile bağdaştırılmıştır. Değişim beklentisi arttıkça belirsizlik de artmaktadır (Boehm, 1988).

Risk yönetimi, proje geliştirme sürecinde sermaye aşımı, ürünü ortaya çıkarma süresinin uzaması gibi çeşitli riskler bulunmaktadır. Bu nedenle, iyi bir risk yönetimi, geliştirme risklerinin azaltılmasını kolaylaştıracak ve geliştirme kalitesini sağlayacaktır (Tam, Moura, Oliveira, & Varajão, 2020). Risk yönetimi sürecinin ilk adımı, sürecin kapsamını net bir şekilde tanımlamaktır. Bu, hangi risklerin ele alınacağı, hangi yöntemlerin kullanılacağı ve sürecin hangi aşamalarda uygulanacağı gibi konuları içerir. Proje ilerledikçe, risklerin durumu değişebilir. Bu adım, risklerin yeniden tanımlanmasını, uygulanmasını ve değerlendirilmesini içerir. Bu sayede, proje ekibi risklerin güncel durumunu takip edebilir. Risk tedavisi, risklerin projeye olumsuz

etkilerini azaltmak veya ortadan kaldırmak için alınan önlemleri içerir. Bu adım, risklerin düzeltilmesi veya maruziyetin önlenmesi gibi faaliyetleri kapsar (Polianskii & Chukalova, 2020).

Maliyet ve Kaynak Yönetiminin Proje Yönetimine Etkileri

Proje geliştirme bütçesi, önemli ve kapsamlı bir çalışmadır. Maliyeti makul bir bütçe içinde kontrol etmek, bir projenin başarısını garanti altına alabilir. Eğer bütçe problemi yaşanırsa, proje en uygun pazara giriş zamanını kaçırır ve bu durum rakiplerin bu fırsattan yararlanmasına neden olabilir (Tam, Moura, Oliveira, & Varajão, 2020). Maliyet aşımı, zaman aşımı ve uygulamaların beklenen özellikleri sağlayamamasıyla ilgili sonuçlar oldukça dikkat çekicidir. Standish Grup (1995) yılına ait rapora göre tüm şirketler için ortalama maliyet aşımı, orijinal maliyet tahmininin %189'u olmuştur. 1995 yılında yayınlanan Standish Grup raporuna göre, projelerin maliyet aşım yüzdeleri projelerin %15,5'i için %20'nin altında bir maliyet aşımı yaşarken, %31,5'i için %21-50 aralığında, %29,6'sı için %51-100 aralığında, %10,2'si için %101-200 aralığında, %8,8'i için %201-400 aralığında ve %4,4'ü için %400'ün üzerinde bir maliyet aşımı yaşamıştır. Bu veriler, projelerin önemli bir kısmının planlanan bütçelerini aştığını ve özellikle %21-100 aralığındaki maliyet aşımlarının daha yaygın olduğunu göstermektedir (Standish, 1995).

Organizasyonel boyutun çok büyük olması son birkaç on yılda, organizasyonlar küreselleşmeye doğru güçlü bir eğilim göstermiş, bu da daha büyük bir uyum sağlama, yanıt verme ve iş tutarlılığı gerektiren yüksek rekabet derecesine sahip pazarlama ortamları yaratmıştır (França, Dias, & Borges, 2015). Organizasyonlar küresel yazılım geliştirmeye yöneldiklerinde, genellikle küresel olarak dağıtılmış yazılım geliştirme ekiplerinin etkisini küçümsemekte ve kültürel çeşitlilik gösteren ekiplerin işbirliğine ihtiyaç duymaktadırlar (Sudhakar, 2012).

İletişim ve Koordinasyonun Proje Yönetimine Etkileri

İletişim etmenleri içerisinde bulunan proje içi iletişim, liderlik, kullanıcı ve personel arası iletişim, belirsizliği azaltma ve ortaya çıkabilecek risklerin minimize edilmesi etkenlerinden en çok karşılaşılan durum olarak belirlenmiştir (Cserháti & Szabó, 2014, Tam, Moura, Oliveira, & Varajão, 2020).

İletişim faktörleri başarılı ve başarısız bir yazılım proje yönetiminin geleceğinde risk almaya istekli olma, iletişim becerileri, siyasetle başa çıkabilme, insanları iyi yönetebilme, stresle başa çıkabilme, projeye zaman ayırabilme, problem çözme, espri anlayışı, dayanıklılık, doğru duygusal denge, esneklik ve sağlam iş yargısına sahip olması gerektiği belirtilmiştir (Sudhakar, 2012). İletişime uygun bir ofis çalışma ortamı ve yüz yüze iletişim, bilgi iletimi sırasında zaman kaybını önlemeyi amaçlayan hızlı bir iletişim şeklidir. Ekiplerin bir arada oturmasına izin verilmeli veya coğrafi olarak dağılan ekipler için iletişim araçları sağlanarak, gerçek zamanlı iletişim kolaylaştırılmalı ve verimlilik artırılmalıdır. İyi bir müşteri ilişkisi, müşterinin en son ihtiyaçlarını zamanında anlamayı teşvik ederken, yüksek müşteri memnuniyeti sağlayan ürünlerin geliştirilmesi de çok önemlidir (Tam, Moura, Oliveira, & Varajão, 2020; Cserháti & Szabó, 2014). Bir geliştirme takımına bilgi iletmenin en verimli ve etkili yolu, yüz yüze görüşmedir (França, Dias, & Borges, 2015).

Organizasyonel etkiler bir organizasyonun unsurlarının mantıklı bir şekilde gruplandırılması, iç süreçlerinde uyum arayışı ve çevresiyle uyum sağlama çabası olarak tanımlanabilir (Baham, 2016). Proje yöneticilerinin perspektifinden, strateji, kolaylaştırma ve liderlik, yazılım projelerinin başarısını sağlamak için yönetimin odaklanması gereken üç temel rol olarak tanımlanmıştır (Sudhakar, 2012). Hızlı gelişim döneminde, geliştiriciler artan geliştirme baskısı altındadır, ancak çoğu yönetici bu uzmanlığa sahip değildir,

bu da onların durumu anlamalarını zorlaştırır ve bu da ekip liderliğini zedeleyebilir (Tam, Moura, Oliveira, & Varajão, 2020).

Üst yönetim desteği organizasyonel etkilerin içerisinde bulunan üst yönetim desteği, gerçekçi beklentiler, organizasyonel politika, finansal destek ve güç etkenlerinden en çok karşılaşılan etken üst yönetim desteğidir (Sudhakar, 2012). Herhangi bir şirketin projesi, yönetim desteğini içermelidir; bu destek, özellikle proje tıkanma noktasına geldiğinde. Kilit destek sağlamak, proje başarısının olasılığını önemli ölçüde artırabilir (Cserháti & Szabó, 2014), (Tam, Moura, Oliveira, & Varajão, 2020). Projeler, motive olmuş bireyler etrafında inşa edilmelidir. Onlara ihtiyaç duydukları ortam ve destek verilmelidir ve işlerini yapmaları için güvenilmelidir (França, Dias, & Borges, 2015).

Ekip Yönetimi ve Motivasyonun Proje Yönetimine Etkileri

İnsan boyutu bilişsel olmayan bazı nitelikleri vurgular; bu nitelikler arasında iletişim becerileri, empati ve dayanıklılık yer alır. Bu özellikler arasında iletişim ve kişilerarası becerilerin yanı sıra dürüstlük, motivasyon, işbirlikçi tutum, sorumluluk duygusu ve öğrenmeye açıklık da vurgulanmaktadır. Projeler bağlamında, iletişim ve kişilerarası beceriler, dürüstlük, işbirlikçi tutum ve başkalarıyla çalışma, bireylerin bir projede, programda veya portföyde iyi performans sergileyebilmesi için gerekli olan kişisel ve kişilerarası yeterliliklerin bir parçasıdır ve bu da başarıya yol açar (Kenny, 2004). İnsan boyutu, yeterli ve motive olmuş ekip üyeleri, süreç hakkında bilgiye sahip yöneticiler, personel deneyimi, iyi müşteri ilişkileri, müşteri katılımı, eğitim ve öğretim, personel kültürü ve aktif iletişim gibi unsurları içerir. İnsanlar, herhangi bir projede son derece önemlidir çünkü iletişim, yönetim desteği ve liderlik, proje uygulamaları için önemlidir (Tam, Moura, Oliveira, & Varajão, 2020).

Verimli bir personel kültürü, ekibin proje hedeflerinin tutarlı olmasını sağlar. Proje çıkarları, departman çıkarları ve kişisel çıkarların entegrasyonu, ekibin müşteri odaklılık bilincini artırabilir (Tam, Moura, Oliveira, & Varajão, 2020).

Sosyal kültür insanlar tarafından gerçekleştirilen diğer tüm etkinlikler gibi, bölgesel kültür de yazılım geliştirmeyi büyük ölçüde etkileyebilir. Toplumsal kültür, paylaşılan değerler, inançlar ve normlardan oluşan bir sistemdir; bu değerler, inançlar ve normlar nesiller boyu öğrenilir ve sürdürülür, toplumun yasalarına, politikalarına ve eylemlerine yansır. Çalışanlar farklı toplumsal kültürlere sahip olduğunda kültürel sürtüşme riski vardır, çünkü kurulan ilişkiler farklı bakış açılarını içeren bir karışımdan oluşur. Toplumsal kültür, bireylerin genel olarak ne kadar iletişimsel, dinamik ve ilerici olduklarını etkiler (Kenny, 2004). Organizasyonel, yönetsel ve kültürel faktörler, projelerin başarısını teknik faktörlerden daha fazla etkiler (Sudhakar, 2012). Herkes saygı görmeli, şirketin sistemi ve kültürü, ekibin büyümesine katkı sağlamak için kullanılmalıdır. Yakın iş birliği, üyelerin birbirlerinden öğrenmelerine ve birbirlerine güvenmelerine olanak tanır, projelerin uygulanmasını ve tanınmasını ortaklaşa teşvik eder. Ayrıca, her üye kendi değerini fark edecek, takımı tanıyacak, aidiyet duygusunu güçlendirecek ve proje gelişimini daha verimli hale getirecektir (Tam, Moura, Oliveira, & Varajão, 2020).

Ekibin yeterliliği ve motivasyonu, bilgi kullanımını ve ekiplerin görevlerini başarıyla yerine getirmelerini sağlayan koşulları ifade eder. Yüksek kapasiteli bir ekip, müşteri gereksinimlerini karşılayan çalışan yazılımların hızlı bir şekilde teslim edilmesini sağlar. Bağlılık ve teknik uzmanlık gibi unsurlar, ekiplerin risklerle daha iyi başa çıkmalarını sağlayan etmenlerdir ve bu da proje başarısının olasılığını artırır (Cserháti & Szabó, 2014), (Kenny, 2004). Takım etkilerinin içerisinde bulunan ekibin yeteneği, takım çalışması, doğru proje ekibinin seçiminden, proje ekibinin

koordinasyonundan ve görev odaklılığından etkenlerinden en çok karşılaşılan etken ekibin yeteneğidir (Sudhakar, 2012). Ekip üyelerinin yetenekleri ve motivasyonu, projenin ilerlemesini ciddi şekilde etkiler. Toplumsal kültür ve kişisel özellikler, ekip yetkinliğini etkiler (Tam, Moura, Oliveira, & Varajão, 2020). Motivasyon faktörleri, yazılım geliştirmede önemli bir rol oynamaktadır (Ghayyur, Naseem, Razzaq, & Ahmed, 2017).

Proje takım bağlılığı, proje takımının davranış ve sonuçlarını dikkate alır, takım üyeleri arasında daha iyi bir anlayış sağlamak için iletişimi vurgular ve böylece projeyi uygular (Cserháti & Szabó, 2014), (Tam, Moura, Oliveira, & Varajão, 2020).

Takım ödüllendirme mekanizması etkili bir ödüllendirme sistemi, sadece projenin başarısına katkıda bulunmakla kalmaz, aynı zamanda proje yönetimi üzerinde de olumlu bir etki yaratır. Bu, projenin ilerleme yönetimini teşvik edebilir, hedefleri netleştirebilir, proje takımının istikrarını sağlayabilir ve takım üyelerinin motivasyonunu artırabilir (Tam, Moura, Oliveira, & Varajão, 2020).

Ürün Kalitesi ve Teslimat

Yazılımı düzenli aralıklarla teslim etmek için etkili bir strateji geliştirilmelidir, böylece müşteri memnuniyeti sağlanabilir. Düzenli yazılım teslimatı, müşterilerin yazılımı deneyimlemelerini, geri bildirimde bulunmalarını ve zamanında ayarlamalar yapmalarını sağlar; bu, sistemin kararlılığını güçlendirir ve ürün teslimatının kalitesini garanti eder (Tam, Moura, Oliveira, & Varajão, 2020).

Ürünün doğruluğu, çıkarılan ürünün güvenirliliği, çıkarılan ürünün zamanlaması, kalitesinin kontrolü ve sistem ve prosedürlerin belgelenmesi etkenlerinden en çok karşılaşılan etken çıkarılan ürünün doğruluğu olmuştur. Teknik görevler, sorun giderme, teknik belirsizlik, teknik uygulama sorunları ve sistem entegrasyonu

etkenlerinden en çok karşılaşılan etken teknik görevler olarak belirlenmiştir (Sudhakar, 2012).

Eğitim ve Kullanıcı Deneyimi

Müşteri eğitimi, kullanıcı ihtiyaçlarının doğru bir şekilde ifade edilmesini ve bu ihtiyaçlar ile değişikliklerin etkin bir şekilde toplanmasını sağlar. Eğitim ve öğretim, çalışanlara eğitim kursları ve bilgi sağlanarak, beklenen sonuçlara başarıyla ulaşılabilir. Eğitim sonrası, ekip üyeleri tutarlı değerler, düşünme ve becerilere sahip olur. Kullanıcı deneyimi başarısı, kullanıcı deneyimini içermeyi gerektirir. İş ve kullanıcı ortamlarını göz önünde bulunduracak zamanın yetersizliği ve bu bilgilerin tasarım vizyonunu oluşturmak için uygulanmaması, tasarımcıların teslimat sürecini zorlaştırabilir (Tam, Moura, Oliveira, & Varajão, 2020).

Metodolojilerin Başarısızlık ve Başarı Üzerindeki Etkileri

Yazılım projelerinin karmaşıklığı, müşteri ihtiyaçlarının hızla değişmesi ve yukarıda belirtilen başarısızlık ve başarı üzerinde faktörlerin fazla olması, proje yönetim metodolojilerinin seçimini kritik bir hale getirmiştir. Geleneksel Şelale (Waterfall) modeli ve modern Çevik (Agile) yaklaşımı, proje yönetiminde en yaygın kullanılan iki metodolojidir. Her iki yaklaşımın da başarıyı ve başarısızlığı nasıl etkilediği, yukarıda bahsi geçen birçok faktöre göre değişmektedir (Campanelli, Neto, & Parreiras, 2017), (Boehm, 1988), (Choukou, 2018).

Organizasyonlar, yazılım geliştirme sorunlarını aşmak ve iş hedeflerine ulaşmak için çevik metodolojilere yönelmektedir. Çevik dönüşüm sürecinde başarı faktörlerinin anlaşılması, değişimin yönetilmesi ve başarı şansını artırmak için kritik öneme sahip (Campanelli, Neto, & Parreiras, 2017). Çevik yöntemler, düşük dokümantasyon, kısa iterasyonlar, erken test etme ve müşteri işbirliği gibi bir dizi teknik kullanarak kararsız ve değişken gereksinimlerle iyi başa çıkar (Ji & Sedano, 2011).

Şelale modeli ise doğrusal ve aşamalı bir süreçtir. Bu model sistem ve yazılım gereksinimleri analizi, tasarım, kodlama, test, operasyon/bakım süreçlerinden oluşur (Allison & Olszewska, 2018). Şelale modeli pek çok büyük ve karmaşık projede benimsenmiş olsa da, değişen gereksinimler karşısında esneklikten yoksun olma gibi bazı doğal dezavantajlara sahiptir. Gereksinim ve tasarım faaliyetlerine büyük miktarda proje kaynağı yatırılmış ise, sonradan yapılacak değişiklikler çok maliyetli olabilir (Ji & Sedano, 2011).

Sonuç

Yazılım projelerinin başarıya ulaşması, yalnızca teknik yeterlilikle değil; aynı zamanda organizasyonel yapılanma, insan kaynakları yönetimi, süreç planlaması, iletişim stratejileri ve paydaş etkileşimi gibi çok boyutlu faktörlerin bütüncül biçimde yönetilmesine bağlıdır. Bu çalışma, yazılım proje yönetiminde başarı ve başarısızlığa etki eden faktörleri detaylı biçimde ortaya koymaktadır.

Yazılım projelerinde başarı kriterleri şu şekilde sıralanabilir; organizasyonel faktörler, insan faktörleri, süreç yönetimi (net gereksinimler, müşteri işbirliği), teknik faktörler (test odaklı geliştirme, düzenli teslimat) başarıyı doğrudan etkiler. Yazılım projelerinde önemli etkiye sahip başarısızlık nedenleri ise; organizasyonel eksiklikler (sponsorluk yokluğu, bürokrasi), planlama hataları (belirsiz kapsam, kaynak eksikliği), iletişim sorunları (zayıf paydaş katılımı), teknik yetersizlikler (test eksikliği) olarak sıralanabilir.

Yazılım projelerinin başarısı, salt teknik becerilerden ziyade insan odaklı yönetim, stratejik planlama ve esnek süreçlerle mümkündür. Şelale ve Çevik modelleri, projenin bağlamına göre doğru uygulandığında başarıyı artırabilir. Ancak, Standish Grup'un vurguladığı gibi, projelerin yalnızca %28,8'i "modern başarı" kriterlerini karşılamaktadır. Bu oranı yükseltmek için,

organizasyonların sürekli iyileştirme kültürünü benimsemesi ve veriye dayalı karar alma mekanizmaları kurması gereklidir. Gelecekte yapılacak çalışmalar kapsamında yapay zeka tabanlı proje yönetimi araçlarının kullanımı değerlendirilebilir.

Bu çalışma, yazılım proje yönetimi alanında başarı ve başarısızlığa etki eden faktörleri literatürdeki çalışmalar ışığında inceleyerek önemli bir boşluğu doldurmayı hedeflemektedir. Farklı araştırmalardan elde edilen veriler bütüncül bir çerçevede bir araya getirilmiş; özellikle yöntemsel farklar, uygulama bağlamları ve sonuçların sürdürülebilirliği gibi kriterler ön plana çıkarılmıştır. Elde edilen bulgular sayesinde, hangi kriterlerin yüksek başarı sağladığı, hangilerinin ise risk barındırdığı ortaya konmuştur. Bu bağlamda, bu çalışma, yazılım proje yönetimi literatürüne teorik katkı sağlamanın yanı sıra uygulayıcılar için de yol gösterici niteliktedir.

Kaynakça

Allison, I., & Olszewska, J. (2018). ODYSSEY: Software Development Life Cycle Ontology., (s. 303-311). doi:10.5220/0006957703030311

Baham, C. (2016). Understanding Agile Software Development Assimilation Beyond Acceptance. doi:DOI: 10.31390/gradschool_dissertations.19

Boehm, B. (1988). A Spiral Model of Software Development and Enhancement. *IEEE EXPLORE*. doi:10.1109/2.59

Campanelli, A. S., Neto, F. S., & Parreiras, F. S. (2017). Assessing Agile Transformation Success Factors. *arXiv.org*, 6. doi:10.48550/arXiv.1711.04188

Choukou, A. A. (2018). Agile Software Development Methodologies: A Comparative Study.

Chow, T., & Cao, D.-B. (2008). A survey study of critical success factors in agile software projects. *The Journal of Systems and Software 81*, 961-971. doi:10.1016/j.jss.2007.08.020

Cserháti, G., & Szabó, L. (2014). The relationship between success criteria and success factors in organisational event projects. *International Journal of Project Management 32*, 613-624. doi:10.1016/j.ijproman.2013.08.008

França, J., Dias, A., & Borges, M. (2015). Observations on collaboration in Agile software development. *2015 IEEE 19th International Conference on Computer Supported Cooperative Work in Design*, (s. 147-152). doi:10.1109/CSCWD.2015.7230949

Ghayyur, S., Naseem, A., Razzaq, A., & Ahmed, S. (2017). Motivators and Demotivators of Agile Software Development: Elicitation and Analysis. *International Journal of Advanced Computer Science and Applications, Vol. 8, No. 12*. doi:10.14569/IJACSA.2017.081239

Group, T. S. (2015). *Standish Group Rapor, 2015*. The Standish Group.

Ji, F., & Sedano, T. (2011). Comparing Extreme Programming and Waterfall Project Results. *24th IEEE-CS Conference on Software Engineering Education and Training (CSEE&T)*, (s. 482-486). doi:10.1109/CSEET.2011.5876129

Kautz, K. (2011). Investigating the design process: participatory design in agile software development. *www.emeraldinsight.com/0959-3845.htm*, 217-235. doi:10.1108/09593841111158356

Kenny, J. (2004). A study of educational technology project man**a**gement in Australian universities. *Australasian Journal*

of Educational Technology, 388-404. doi:10.14742/ajet.1354

Krasniqi, I., Ismajli, N., & Krasniqi, G. (2024). Software Project Management and Their Economic Evaluation in terms of Enterprise Sustainability. *he Eurasia Proceedings of Science, Technology, Engineering & Mathematics (EPSTEM), 2024 Volume 28*, (s. 533-553). doi:10.55549/epstem.1535078

Polianskii, A., & Chukalova, D. (2020). Software product management: planning tool integration. *MATEC Web of Conferences 311*. doi:https://doi.org/10.1051/matecconf/202031102011

Standish, T. (1995). *Standish Group Rapor 1995*. The Standish Group.

Standish, T. (2015). *Standish Group Rapor*. The Standish Group.

Sudhakar, G. P. (2012). A model of critical success factors for software projects. *www.emeraldinsight.com/1741-0398.htm*. doi:10.1108/17410391211272829

Tam, C., Moura, E., Oliveira, T., & Varajão, J. (2020). The factors influencing the success of on-going agile software development projects. *International Journal of Project Management 38*, 165-176. doi:10.1016/j.ijproman.2020.02.001

Zhang, F., Abdullah, N., & Rosli, M. M. (2024). Critical Success Factors of Agile Software Projects: A Review. *Engineering, Technology & Applied Science Research Vol. 14 No. 5* (s. 16886-16873). içinde doi:https://doi.org/10.48084/etasr.8358

CHAPTER 5

Tablolaştırılmış Sağlık Verilerinin Makine Öğrenmesi Eğitimine Uygun Şekilde Hazırlanması: Temizlik, Bölme ve Klinik Uyum

EMRE ÇANAYAZ[1]

Giriş

Makine öğrenmesi (ML) tabanlı yaklaşımlar, medikal karar destek sistemlerinin geliştirilmesinde giderek daha merkezi bir rol oynamaktadır (Shickel, 2018). Ancak bu sistemlerin güvenilirliği ve klinik geçerliliği, öncelikle kullanılan verinin yapısal kalitesi ve uygun şekilde ön işlenmesine bağlıdır (Rajkomar, 2018). Özellikle tablo yapısındaki medikal veriler, elektronik sağlık kayıtları (EHR), laboratuvar bulguları, vital işaretler ve hasta anketleri gibi çeşitli kaynaklardan gelen çok boyutlu bilgileri barındırır. Bu nedenle, model eğitimi öncesinde veri setinin uygun biçimde düzenlenmesi ve temizlenmesi, öğrenme sürecinin başarısını belirleyen temel adımlardan biridir.

Medikal veriler çoğunlukla eksik gözlemler, aykırı değerler, ölçüm sıklığı farklılıkları ve dengesiz sınıf dağılımları gibi yapısal sorunlar içerir (Beaulieu-Jones, 2017). Bu sorunlar, modelin

[1] Dr. Öğr. Üyesi, Marmara Üniversitesi, Teknik Bilimler Meslek Yüksekokulu, Orcid: 0000-0002-3695-3642

öğrenme sürecini yanıltabilir ve sonuçta yanlış teşhislere, hatalı sınıflandırmalara veya klinik açıdan geçersiz karar destek sistemlerine yol açabilir. Literatürde bu tür hatalı çıkarımların genellikle verinin ön işleme aşamasında yapılan hatalardan kaynaklandığı gösterilmiştir (Antoniadi, 2021). Bu nedenle bu bölümde, tabular medikal verilerde kullanılan veri ön işleme teknikleri sistematik biçimde ele alınacaktır. Öncelikle medikal verinin yapısal özellikleri tanıtılacak, ardından sırasıyla eksik verilerle başa çıkma stratejileri, aykırı değerlerin tespiti, kategorik ve sayısal veri dönüşümleri, veri ölçekleme, sınıf dengesizliği problemleri için çözümler gibi kritik adımlar açıklanacaktır. Bölümün sonunda, ön işleme kalitesinin model başarısına etkisi üzerine genel çıkarımlar ve öneriler sunulacaktır.

Medikal Verinin Yapısal Özellikleri

ML modellerinin klinik alanda başarılı bir şekilde uygulanabilmesi, büyük ölçüde kullanılan medikal verinin yapısal özelliklerinin doğru anlaşılmasına bağlıdır. Tablolaştırılmış medikal veriler, genellikle hastane bilgi yönetim sistemlerinden (HBYS), EHR, laboratuvar bilgi sistemlerinden veya hasta tarafından beyan edilen anket verilerinden elde edilir (Shickel, 2018).Bu veriler; demografik bilgiler, vital bulgular (örneğin tansiyon, nabız), biyokimyasal parametreler (kan şekeri, kreatinin vb.), medikal tanılar, reçeteler ve tedavi süreçlerine ilişkin kayıtları kapsar. Bu tür verilerin karakteristik yapısı üç temel zorluk alanı barındırır: eksik gözlemler, heterojenlik ve zaman frekansı ile ölçüm tutarsızlıkları (Lee, 2017).

Eksik Gözlemler ve Ölçüm Tutarsızlıkları

Tıbbi uygulamalarda her hasta için her değişkenin ölçülmesi mümkün olmayabilir. Tanı veya tedaviye özel olarak bazı testlerin yapılmaması, hasta uyumsuzluğu, veri giriş hataları veya teknik aksaklıklar nedeniyle veri setlerinde eksik değerler yaygındır.

Eksikliklerin rastgele (MCAR), değişkene bağlı (MAR) veya sistematik (MNAR) olması, uygulanacak olan tamamlayıcı yöntemlerin de farklılaşmasına yol açar (Little, 2019).

Medikal veri genellikle farklı kaynaklardan toplandığından, birleştirilen veri setlerinde biçimsel (format), anlamsal (anlam kayması), ve zamansal (ölçüm zamanı) heterojenlikler görülebilir (Jensen, 2012). Örneğin bir merkezde glikoz düzeyi "mg/dL" biriminde kaydedilirken, diğerinde "mmol/L" kullanılabilir. Aynı şekilde bazı merkezlerde 24 saatlik ölçümler yapılırken, diğerlerinde haftalık ölçümler tercih edilebilir. Bu durum, özellikle çok merkezli klinik araştırmalarda normalizasyon ihtiyacını doğurur (Antoniadi, 2021). Vital bulgular veya laboratuvar testleri gibi bazı parametreler zamana bağlı olarak değişir. Ancak bu ölçümler her hasta için aynı sıklıkta yapılmadığından veri setlerinde "düzensiz zaman serileri" oluşabilir. Bu, klasik tablolaştırılmış verilere dayanan analizlerde bilgi kaybına veya yanıltıcı eğilimlere neden olabilir. Bu tür yapıların dikkate alınmadığı durumlarda, modelin eğitimi sırasında zamansal önyargılar oluşabilir (Futoma, 2017). Bu sebeplerle, medikal veriyle çalışan araştırmacıların yalnızca istatistiksel analiz değil, aynı zamanda tıbbi bağlam bilgisine de sahip olması gerekir. Klinik mantığın dışına çıkan sayısal değerlerin modele sokulması, yalnızca doğruluk değil, etik riskler de doğurabilir (Wiens, 2019).

Eksik Verilerle Başa Çıkma

Eksik veriler, özellikle tablolaştırılmış medikal veri setlerinde yaygın bir problemdir ve bu eksikliklerin niteliği, uygulanacak ML modelinin doğruluğunu ve genellenebilirliğini doğrudan etkileyebilir (Antoniadi, 2021). Klinik uygulamalarda eksik veri, yalnızca teknik bir sorun değil; aynı zamanda tanı süreçleri, hasta uyumu ve sağlık sistemi altyapısı gibi çok katmanlı nedenlere dayanmaktadır (Little, 2019). Eksik veriler, nedenlerine göre üç temel gruba ayrılır (Tablo 1):

- Rastgele Eksik Veri (MCAR): Verinin eksik olması tamamen tesadüfidir ve diğer gözlemlerle ilişkisizdir.
- Koşullu Rastgele Eksik Veri (MAR): Eksiklik, gözlemlenmiş başka bir değişkene bağlıdır.
- Rastgele Olmayan Eksik Veri (MNAR): Eksiklik, gözlemlenememiş değerin kendisiyle ilişkilidir (örneğin, depresyon seviyesi yüksek bireylerin anket doldurmaması) (Jakobsen, 2017).

Tablo 1 Eksik veri türleri ve uygun tamamlama yöntemleri

Eksik Veri Türü	Tanım	Neden Ortaya Çıkar?	Önerilen Tamamlama Yöntemleri
MCAR	Eksiklikler tamamen rastgeledir; başka değişkenlerle ilişkili değildir.	Sistemsel hata, anketlerde yanlışlıkla atlanma	Ortalama/medyan ile doldurma, KNN imputation
MAR	Eksik değerler başka bir gözlemlenebilir değişkenle ilişkilidir.	Yaşlı bireylerin bazı soruları yanıtsız bırakması gibi sistematik durumlar	Multiple Imputation (MICE), regresyon tabanlı tamamlama
MNAR	Eksikliğin nedeni eksik verinin kendisinden kaynaklanır.	Kilo bilgisi utandığı için girilmemişse	Domain bilgisi ile doldurma, klinik olarak anlamlı default değer atama

Tablo 1'de, tablolaştırılmış medikal veri setlerinde karşılaşılan eksik veri türleri açıklanmakta ve her bir eksik tür için uygun tamamlama yöntemleri özetlenmektedir. Eksik verilerin nedenleri ve istatistiksel doğası, modelleme sürecinde veri kalitesini doğrudan etkileyebileceği için doğru sınıflandırma ve uygun tekniklerin seçimi hayati öneme sahiptir.

Basit Tamamlama Yöntemleri

Medikal veri analizlerinde en yaygın başvuru yöntemi, ilgili sütundaki değişkenlere ait ortalama veya medyan değerle

doldurmadır. Bu yöntemler hızlı ve anlaşılır olmakla birlikte, özellikle MAR ve MNAR durumlarında önyargı yaratabilir ve varyansın olduğundan düşük tahmin edilmesine yol açabilir (Azur, 2011).

K-en yakın komşu (KNN) tamamlama, eksik değeri doldurmak için en yakın komşuların (k adet) benzer gözlemleri referans alır. Özellikle MAR durumlarında etkili bir yöntemdir. Ancak yüksek boyutlu ve seyrek verilerde hesaplama yükü artabilir. Ayrıca komşuluk yapısı medikal anlamda mantıklı değilse klinik geçerliliği azalır (Batista G. E., 2002). Çoklu tamamlama (MICE) yöntemi, eksik verilerin birden fazla olası değeri üzerinden modelleme yaparak belirsizliği hesaba katar. Tablolaştırılmış medikal verilerde genellikle regresyon tabanlı modellerle zincirleme tahmin yapılır. Özellikle epidemiyolojik çalışmalarda tercih edilir (Van Buuren, 2011). Bazı durumlarda, eksikliği anlamlı olan değişkenlerde "klinik sınır değeri" veya "güvenli eşik değeri" atamak, bilimsel olarak tercih edilebilir. Örneğin, hastaneye yatırılmamış bir bireyin C-reaktif protein (CRP) testi yoksa "normal kabul edilen" değer atanabilir. Bu yaklaşımın etik ve metodolojik olarak açıklanması gerekir (Sterne, 2009).

Eksik veri ile mücadele, yalnızca eksik gözlemlerin doldurulması değil, aynı zamanda bu eksikliğin kaynağının anlaşılması sürecidir. Ayrıca eksik veri oranlarının yüksek olduğu durumlarda tamamlamanın ardından model performansının ve önyargıların analiz edilmesi de gereklidir (Cheema, 2014).

Aykırı Değer Tespiti Ve İşlenmesi

Aykırı değerler, değişkenlerin dağılımında beklenen aralığın dışında yer alan uç noktalardır. Bu değerler, ölçüm hatasından, klinik ekstrem durumlara (örneğin hiperkalemi, hipoglisemi) kadar çeşitli nedenlere bağlı olabilir. Aykırı değerlerin etkileri arasında ML

modellerinde eğitimin bozulması, ağırlıkların yanlış öğrenilmesi ve önyargılı sonuçların elde edilmesi sayılabilir (Aggarwal, 2017).

Medikal verilerde aykırı değerler, yanlış ölçümler, cihaz hataları, veri girişindeki tutarsızlıklar veya istisnai klinik durumlar nedeniyle ortaya çıkabilir. Bu nedenle istatistiksel yöntemlerle bu değerlerin tanımlanması hem model doğruluğunu artırmak hem de klinik yorumların güvenilirliğini sağlamak açısından kritik önemdedir (Zimek, 2012). Aşağıda yaygın olarak kullanılan dört istatistiksel aykırılık yöntemi detaylandırılmaktadır (Tablo 2).

Tablo 1 Aykırı değer tespiti için kullanılan yaygın yöntemler

Yöntem	Açıklama	Kullanım Durumu	Avantajları	Sınırlılıkları
Z-Puanı	Gözlemin ortalamadan kaç standart sapma uzaklıkta olduğunu gösterir.	Sürekli ve normal dağılan verilerde	Kolay hesaplanır, hızlı uygulanabilir	Normal dağılıma duyarlıdır, aykırı değerlere karşı hassastır
IQR	1. Çeyrek ve 3. Çeyrek arasındaki fark kullanılarak aşırı düşük/yüksek değerler tanımlanır.	Sağlam dağılıma sahip sayısal verilerde	Medyana dayalıdır, uç değerlere karşı dayanıklıdır	Simetrik olmayan dağılımlarda hatalı sınıflandırma riski olabilir
MAD	Değerlerin medyandan olan mutlak sapmalarının medyanı üzerinden aykırılık belirlenir.	Sağlam istatistik isteyen uygulamalarda	Aykırı değerlerden etkilenmeyen sağlam yöntem	Zayıf sezgisellik ve yorumlama zorluğu
Klinik Sınırlarla Karşılaştırma	Literatürde belirtilen normal referanslar	Biyobelirteçler ve laboratuvar verileri	Klinik geçerlilik sağlar, uzman görüşüyle uyumludur	Her değişken için standart sınır bulunmayabilir

Z-skoru, bir gözlemin kendi dağılımı içerisindeki konumunu standart sapma cinsinden ifade eden bir ölçüttür. Matematiksel olarak şu şekilde tanımlanır (1):

$$Z = \frac{x - \mu}{\sigma} \quad (1)$$

Burada, x: Gözlemin değeri, μ: Değişkenin ortalaması, σ: Değişkenin standart sapmasıdır. Z-skoru, değişkenin dağılımının normal (gauss dağılımı) olduğu varsayımı altında çalışır. Genel uygulamalarda ±3'ün dışında kalan gözlemler aykırı kabul edilir. Yani bir veri noktası ortalamadan 3 standart sapmadan daha uzaksa, bu durum potansiyel aykırılık sinyali verir. Hesaplaması kolaydır. Ancak verinizin normal dağıldığı varsayımına dayanır. Medikal verilerde çoğu zaman bu varsayım sağlanmadığı için yanlış pozitif aykırı değer tespitleri yapılabilir.

Çeyrekler arası aralık yöntemi (IQR), verilerin ortanca değer çevresindeki yayılımını incelemek için kullanılan dayanıklı bir aykırı değer tespit yöntemidir. Veriyi küçükten büyüğe sıraladıktan sonra, ortadaki %50'lik kısmı temsil eden alt ve üst çeyrek değerleri arasındaki fark temel alınır. Bu yöntem, özellikle medyan etrafında toplanan ve uç değerlerin etkili olduğu tıbbi veriler gibi durumlarda tercih edilir. Normal dağılım varsayımı gerektirmez ve uç değerlerden daha az etkilenir. Ancak veri yapısı çok çarpıksa veya örneklem küçükse, sınır belirlemede yanılma riski oluşabilir.

Medyan Mutlak Sapma Yöntemi (MAD), medyanın çevresindeki mutlak sapmaların medyanı olarak tanımlanır (2) (Leys, 2013):

$$\tilde{x}MAD = median(|x_i - \tilde{x}|) \quad (2)$$

Burada x̃, verinin medyanıdır. Aykırı değer tanımı genellikle aşağıdaki eşik ile yapılır (3):

$$\frac{|x_i - \tilde{x}|}{MAD} > k \quad (3)$$

Burada k, genellikle 2.5 veya 3 alınır. Bu yöntem özellikle dağılımın çarpık olduğu ve klasik yöntemlerin güvenilir olmadığı durumlarda etkilidir. Aykırı değerlerden etkilenmeyen sağlam bir istatistiksel ölçüdür. Ancak hesaplaması daha karmaşıktır ve sezgisel yorumlama açısından zorlayıcı olabilir.

Ayrıca medikal analizlerde yalnızca istatistiksel yöntemlere değil, aynı zamanda klinik sınır değerlerine başvurmak önerilir. Örneğin, kalp atım hızı 30 bpm ya da 250 bpm gibi istatistiksel olarak aykırı görünen bir değer, bazı özel hastalarda (örneğin bradikardi, taşikardi) geçerli olabilir (Tracy, 2012). Bu nedenle aykırı değer istatistiksel olarak uçta olabilir, ancak klinik olarak geçerli olabilir ve de klinik sınır bilgisi yoksa yalnızca istatistiksel eşikler kullanılmalıdır

Kategorik Verilerin Dönüştürülmesi

Tablo halindeki medikal veriler, yalnızca sayısal değil; aynı zamanda tanı kodları, tedavi sınıfları, cinsiyet, ilaç türleri, hastane birimleri gibi çok sayıda kategorik değişken içermektedir. ML algoritmalarının çoğu bu tür verilerle doğrudan çalışamaz. Bu nedenle, kategorik verilerin sayısal temsile dönüştürülmesi gerekir. Ancak bu dönüşüm süreci dikkatli yapılmadığında anlam kaybına, model önyargılarına veya veri sızıntısına neden olabilir (Zheng, 2018).

Kategorik verilerin makine öğrenmesi modellerine dâhil edilebilmesi için uygun sayısal temsillere dönüştürülmesi gereklidir. Kategorik değişkenlerin ikili temsil edilmesi yöntemi, özellikle sırasız kategorik değişkenler için tercih edilmekte ve her bir sınıf için ayrı bir ikili sütun oluşturarak bu sınıfları bağımsız biçimde temsil etmektedir (Kuhn, 2013). Ancak sınıf sayısı fazla olduğunda bu yöntem modelin boyutunu gereksiz yere artırabilir ve hesaplama

karmaşıklığına yol açabilir. Öte yandan, doğal bir sıralama içeren değişkenlerde ordinal encoding yaklaşımı tercih edilir; bu yöntemde sınıflar ardışık sayılarla kodlanır. Ancak sırasız kategorilere uygulanması durumunda model, gerçekte var olmayan bir sıralama ilişkisi öğrenebilir ve bu da hatalı sonuçlara neden olabilir. Medikal veri setlerinde sıkça rastlanan ICD-10, ATC veya SNOMED CT gibi kod sistemleri, doğrudan kategorik değişkenlerin ikili temsil edilmesi yöntemine tabi tutulduğunda ciddi boyut artışına yol açabilir. Bu nedenle, bu tür kodların benzer klinik anlam grupları altında birleştirilmesi yöntemleri ile düşük boyutlu temsillere dönüştürülmesi önerilmektedir (Choi, 2016). Ayrıca, dönüşüm süreçlerinde dikkat edilmesi gereken önemli bir unsur da etiket sızıntısı riskidir. Eğer dönüşümler eğitim ve test verileri ayrılmadan önce yapılırsa veya hedef değişkenle doğrudan ilişkili kategoriler modele dâhil edilirse, modelin doğruluk değeri yapay olarak yüksek çıkabilir (Kaufman, 2012). Bu nedenle tüm kategorik kodlamalar eğitim-test ayrımından sonra yapılmalı ve yalnızca klinik olarak anlamlı değişkenler dönüşüme tabi tutulmalıdır.

Sayısal Verilerin Ölçeklenmesi

Sayısal değişkenler, medikal veri setlerinde yaş, kan basıncı, glikoz düzeyi, kolesterol, vücut kitle indeksi (BMI), kreatinin düzeyi gibi çok sayıda biyokimyasal ve fizyolojik ölçümle temsil edilir. Bu değişkenler birbirinden farklı ölçü birimlerine, dağılımlara ve değişkenliklere sahip olabilir. Bu çeşitlilik, özellikle ölçüme duyarlı ML algoritmalarında (örneğin KNN, Destek Vektör Makineleri (SVM), Lojistik Regresyon vb.) model performansını doğrudan etkileyebilir. Bu nedenle ölçekleme işlemi, medikal modellemelerde ön işleme sürecinin vazgeçilmez bir parçası hâline gelmiştir (James, 2013). Farklı ölçekleme yöntemleri mevcuttur (Tablo 3).

Tablo 2 Sayısal verilerde ölçekleme yöntemleri

Yöntem Adı	Açıklama	Avantajlar	Dezavantajlar	Uygun Olduğu Durumlar
Z-Skor Ölçekleme	Ortalama 0, standart sapma 1 olacak şekilde dönüştürme	Normal dağılıma yakın verilerde başarılı	Normal dağılmayan verilerde yanıltıcı olabilir	Glikoz, tansiyon, yaş gibi dağılımı simetrik veriler
Min-Maks Ölçekleme	Veriyi 0–1 aralığına sıkıştırır	Sinir ağı, karar ağacı gibi modellerde eğitim kolaylığı sağlar	Aykırı değerlere duyarlıdır	Normalize edilmiş sinyaller, karar ağaçları
Aykırıya Dayanıklı Ölçekleme	Medyan ve çeyrekler arası aralığa (IQR) göre dönüştürme	Aykırı değerlere karşı dayanıklıdır	Verinin genel dağılımına dair bilgi kaybına yol açabilir	Biyobelirteçler, sağa çarpık dağılımlar, CRP, IL-6 gibi veriler

z-skor ölçekleme, her sayısal değişkenin ortalamasını (μ) sıfır, standart sapmasını (σ) bir olacak şekilde dönüştürür. Özellikle normal dağılıma yakın değişkenlerde etkilidir. Algoritmaların değişkenler üzerindeki ağırlık farklarını minimize eder. Glikoz, tansiyon gibi sürekli ve simetrik dağılıma yakın verilerde tercih edilir. Lojistik regresyon, SVM gibi metrik duyarlılığı yüksek modellerde yaygındır (Shalabi, 2006).

Min-max ölçeklemede ise verileri belirli bir aralığa (genellikle 0–1) sıkıştırır. Şu formülle hesaplanır (4):

$$x_{ölç} = \frac{x - x_{min}}{x_{max} - x_{min}} \quad (4)$$

Tüm veriler belirli bir aralıkta olduğundan sinir ağı modellerinde eğitim sürecini hızlandırır. Özellikle karar ağaçları ile çalışırken kullanışlıdır. Ancak aykırı değerler mevcutsa bu değerler

min–max dengesini bozabilir ve de klinik açıdan anlamlı olan yüksek uç değerler bastırılabilir (Shalabi, 2006).

Aykırıya dayanıklı ölçekleme yöntemleri, özellikle tıbbi verilerde sıkça karşılaşılan uç değerlerin etkisini azaltmak amacıyla geliştirilmiştir. Bu yöntemde ortalamadan ziyade medyan ve çeyrekler arası aralık (IQR) esas alınarak veri dönüştürülür. Bu sayede, CRP ve IL-6 gibi sağa çarpık dağılıma sahip biyobelirteçlerin modellenmesi sırasında uç değerlerin modele olan bozucu etkisi önemli ölçüde azaltılır (Ghosh, 2025). Bununla birlikte, bazı medikal veriler için yalnızca istatistiksel ölçekleme yeterli değildir; verinin klinik bağlamda yorumlanabilirliğini artıracak dönüşümlere de ihtiyaç vardır. Klinik anlam taşıyan bu tür dönüşümler, verilerin sadece teknik olarak modele uyarlanmasını değil, aynı zamanda sağlık profesyonelleri için anlamlı hâle getirilmesini sağlar (Goldstein, 2017). Örneğin, kreatinin düzeyleri ABD'de mg/dL cinsinden ölçülürken Avrupa'da μmol/L birimi kullanılmaktadır; bu durum, farklı veri kaynaklarının karşılaştırılabilirliğini sağlamak için birim dönüşümlerini zorunlu kılar.

Benzer şekilde, yaş verisi sayısal olarak modele girilmek yerine, "60 yaş üstü" gibi klinik olarak risk tanımları içeren kategorilere dönüştürülebilir. HbA1c gibi belirteçler ise yaş ve cinsiyete göre z-skora çevrilerek normal aralıklarla ilişkilendirilebilir. Biyobelirteçlerin bu şekilde normalize edilmesi, modelin hem teknik doğruluğunu artırır hem de klinik karar süreçlerine daha fazla katkı sağlar. Örneğin, eGFR'nin yaşa göre düzeltilmiş değerleri ya da T3/T4 oranı gibi türev ölçümler bu amaca hizmet eder. Bu tür dönüşümler yalnızca modelleme açısından değil, aynı zamanda etik açıdan da daha adil ve bireyselleştirilmiş bir yaklaşım sunar; çünkü aynı biyobelirteç düzeyi, farklı yaş ya da fizyolojik özelliklere sahip bireylerde farklı klinik anlamlar taşıyabilir (Doupe, 2019).

Veri Dengesizliği Ve Sınıf Dağılımı

Medikal veri setlerinde sınıf dengesizliği, özellikle nadir görülen sağlık durumlarının sınıflandırılmasında sık karşılaşılan bir sorundur. Bu durum, modelin baskın sınıfa aşırı odaklanmasına ve azınlık sınıfını göz ardı etmesine yol açarak, genel doğruluk yüksek görünse bile nadir hastalıkların doğru tahmin edilememesi gibi ciddi problemlere neden olabilir (Batista et al., 2004). Dengesizliğin klinik yansımaları oldukça kritiktir; örneğin sepsis gibi acil müdahale gerektiren bir durumu atlayan model, pratikte kullanılmaz hâle gelir. Bu nedenle, performans değerlendirmelerinde doğruluk oranı yerine özellikle duyarlılık gibi sınıf odaklı metrikler öncelikli olarak dikkate alınmalıdır (He, 2008; Chawla, 2002).

Sınıf dengesizliğiyle başa çıkmak için çeşitli stratejiler geliştirilmiştir. Bunlardan biri olan aşırı örnekleme yaklaşımı, azınlık sınıfını yeniden dengelemek amacıyla daha fazla örnek üretmeyi hedefler. Bu kapsamda en yaygın yöntemlerden biri SMOTE'tur (Synthetic Minority Oversampling Technique). SMOTE, azınlık sınıfındaki gözlemler arasındaki benzerliklerden yararlanarak sentetik veriler üretir ve bu sayede veri setinde daha dengeli bir dağılım sağlar; ancak yüksek boyutlu verilerde ezberleme riski taşır (Chawla, 2002).

SMOTE'un gelişmiş bir versiyonu olan ADASYN (Adaptive Synthetic Sampling) ise yanlış sınıflandırılma olasılığı yüksek gözlemlere öncelik vererek sentetik örnek üretir ve bu sayede karar sınırlarını daha etkili biçimde tanımlar (He, 2008). Daha yeni bir yaklaşım olan CTGAN (Conditional Tabular GAN), çekişmeli üretici ağ mimarisine dayanır ve karmaşık, çok boyutlu medikal veri setlerinde azınlık sınıfı için gerçekçi ve istatistiksel olarak geçerli örnekler oluşturabilir; ayrıca veri gizliliğini koruma avantajı da sunar. Diğer taraftan, örneklem azaltma yöntemleri çoğunluk sınıfından bazı örneklerin çıkarılması yoluyla uygulanır. En temel biçimi rastgele örneklem azaltmadır; bu yöntem kolay uygulanabilir

ancak bilgi kaybı riski yüksektir (Batista G. E., 2004). Alternatif olarak, sınıf sınırlarını daha net belirlemeyi amaçlayan Tomek Links yöntemi, sınıflar arası belirsizlikleri ortadan kaldırabilir (Chandra, 2023).

NearMiss yöntemi ise azınlık sınıfına en yakın çoğunluk örneklerini seçerek sınıflar arası karar sınırlarını optimize etmeye çalışır (Mani, 2003). Tüm bu yöntemlerin sınırlılıklarını aşmak amacıyla kombine yaklaşımlar da geliştirilmektedir. Bu yaklaşımlar hem aşırı örnekleme hem de örneklem azaltma yöntemlerini birlikte kullanarak hem veri çeşitliliğini artırır hem de modelin aşırı öğrenme riskini düşürerek daha kararlı tahminler elde edilmesini sağlar (Batista G. E., 2002).

Sonuç Ve Öneriler

Makine öğrenmesine dayalı klinik karar destek sistemleri yüksek doğruluk ve öngörü vadetse de bu potansiyelin hayata geçirilebilmesi büyük ölçüde veri ön işleme sürecinde alınan kararların doğruluğuna bağlıdır. Bu bağlamda, eksik veri yönetimi, aykırı değerlerin işlenmesi, kategorik değişken dönüşümleri, ölçekleme teknikleri, sınıf dengesizliğiyle başa çıkma stratejileri ve veri sızıntısının önlenmesi gibi kritik adımlar, yalnızca modelin istatistiksel başarımını değil, aynı zamanda klinik bağlamda güvenilirliğini de belirlemektedir.

Eksik verilerin gelişigüzel doldurulması ya da yok sayılması, sistematik önyargılara ve öğrenme hatalarına yol açarak özellikle hassas klinik karar mekanizmalarında yanlış sınıflandırma riskini artırır. Benzer şekilde, aykırı değerlerin göz ardı edilmesi, nadir fakat klinik olarak anlamlı gözlemlerin analiz dışında kalmasına neden olur ve bu durum modelin azınlık gruplar üzerindeki genellendirilebilme kabiliyetini zayıflatır. Kategorik değişkenlerin bağlamdan kopuk biçimde dönüştürülmesi de ciddi sorunlara yol açabilir; özellikle tanı kodları veya ilaç sınıflandırmaları gibi

kavramsal açıdan yüklü veriler, anlamsız sayısal ifadelere indirgenirse modelin iç yapısal örüntüleri tanıma becerisi azalır. Ölçeklenmemiş sayısal değişkenler de modelin bazı girdilere aşırı ağırlık vermesine neden olabilir; örneğin kan şekeri gibi geniş dağılıma sahip bir değişken ile yaş gibi dar dağılımlı bir değişken aynı modelde birlikte işlendiğinde öğrenme sürecinde öncelik kaymaları meydana gelir. Bu tür ön işleme hataları bir araya geldiğinde, görünürde yüksek doğruluğa sahip modeller gerçekte düşük klinik geçerlilik gösterebilir.

Öte yandan, tıbbi verilerle yapılan her müdahale yalnızca teknik değil, aynı zamanda etik bir eylem olarak değerlendirilmelidir. Modelin öğrenme sürecinde kullanılan bir verinin nasıl işlendiği, gelecekte gerçek bir hastanın tanı veya tedavi sürecini doğrudan etkileyebilir. Özellikle nadir hastalıkların veya marjinal hasta gruplarının veri setinden çıkarılması ya da temsil edilmemesi, yapay zekâ sistemlerinin bu grupları dışlaması anlamına gelir. Bu nedenle, araştırmacıların ön işleme kararlarını yalnızca istatistiksel optimizasyon açısından değil, hasta hakları ve toplumsal eşitlik açısından da değerlendirmeleri gerekmektedir (Kaufman, 2012).

Uygulamada ön işleme adımlarının açıkça belgelenmesi ve sürdürülebilir, tekrarlanabilir bir veri hazırlama protokolü oluşturulması önerilir. Klinik karar destek sistemlerinde kullanılacak modeller, yalnızca veri bilimciler tarafından değil; klinik uzmanlarla iş birliği içinde geliştirilmelidir. Böylece değişken seçimi ve dönüşüm adımları, gerçek klinik karar mekanizmalarıyla örtüşen bir yapıda planlanabilir. Özellikle zaman serisi içeren veri setlerinde geriye dönük bilgi aktarımını engelleyecek bölme stratejileri benimsenmelidir. Örneğin, 2018–2022 yıllarına ait laboratuvar kayıtlarında geçmiş yıllar eğitim seti olarak, daha güncel veriler ise test seti olarak kullanılmalıdır. Bu, modelin geleceği öngörme becerisinin gerçekçi şekilde test edilmesini sağlar. Ancak düşük

tekrarlı klinik olayların yalnızca test setine düşmesi, modelin bu durumları öğrenememesine yol açabilir. Benzer biçimde, aynı hastaya ait verilerin eğitim ve test setleri arasında bölünmesi, veri sızıntısına neden olur; bu da modelin doğruluğunu yapay biçimde artırır (Schulam, 2017).

Bu tür hataları önlemek için GroupKFold gibi grup bazlı bölme stratejileri önerilmektedir. Ayrıca sınıf dengesizliği olan durumlarda, örneğin sepsis gibi %5 oranında görülen nadir olaylarda hem zaman duyarlı hem de sınıf oranlarını koruyan katmanlı bölme yöntemleri kullanılmalıdır (Kaufman, 2012). Veri seti ne zaman bölünmelidir sorusu da kritik öneme sahiptir: veri temizliği tamamlandıktan, eksik ve aykırı değerler belirlendikten sonra, ancak henüz ölçekleme ya da kodlama işlemleri yapılmadan önce bölme gerçekleştirilmelidir. Bu yaklaşım, ön işleme işlemlerinin yalnızca eğitim verisi üzerinde yapılmasını ve test verisine ilişkin istatistiksel bilginin modele "sızmamasını" garanti eder (Vabalas, 2019).

Genel olarak eğitim–test bölme oranları %70–80 eğitim ve %20–30 test olarak önerilse de bu oranlar sabit değildir ve veri setinin büyüklüğü, olay sıklığı ve sınıf yapısına göre ayarlanmalıdır (Cawley, 2010). Özellikle derin öğrenme gibi modellerde, eğitim–test dışında ayrı bir validasyon seti (%10–15) kullanılması önerilir. En önemli ilke ise test verisinin model geliştirme sürecinden tamamen bağımsız tutulmasıdır. Veri bölme stratejisi yalnızca metodolojik bir detay değil; bilimsel geçerlilik ve klinik güvenlik açısından temel bir etik zorunluluktur.

Kaynakça

Aggarwal, C. C. (2017). *An introduction to outlier analysis.* 1-34: Springer International Publishing.

Antoniadi, A. M. (2021). Current challenges and future opportunities for XAI in machine learning-based clinical decision support systems: a systematic review, 11(11),. *Applied Sciences, 11*(11).

Azur, M. J. (2011). Multiple imputation by chained equations: what is it and how does it work? *International journal of methods in psychiatric research, 20*(1), 40-49.

Batista, G. E. (2002). A study of K-nearest neighbour as an imputation method. *His, 87*, 251-260.

Batista, G. E. (2004). A study of the behavior of several methods for balancing machine learning training data. *ACM SIGKDD Explorations Newsletter, 6*(1), 20–29. doi:https://doi.org/10.1145/1007730.1007735

Beaulieu-Jones, B. K. (2017). Missing data imputation in the electronic health record using deeply learned autoencoders. *Pacific Symposium on Biocomputing*, *22*, s. 207–218.

Cawley, G. C. (2010). On over-fitting in model selection and subsequent selection bias in performance evaluation. *Journal of Machine Learning Research, 11*, 2079–2107.

Chandra, W. S. (2023). Median-KNN Regressor-SMOTE-Tomek links for handling missing and imbalanced data in air quality prediction. *Symmetry, 15*(4), 887.

Chawla, N. V. (2002). SMOTE: Synthetic Minority Over-sampling Technique. *Journal of Artificial Intelligence Research, 16*, 321–357. doi:https://doi.org/10.1613/jair.953

Cheema, J. R. (2014). A review of missing data handling methods in education research. *Review of Educational Research, 84*(4), 487–508. doi:https://doi.org/10.3102/0034654314532697

Choi, E. B. (2016). Doctor AI: Predicting clinical events via recurrent neural networks. . *Proceedings of the Machine Learning for Healthcare Conference*, (s. 301–318).

Doupe, P. F. (2019). Machine learning for health services researchers. *Value in Health, 22*(7), , 808–815. doi:https://doi.org/10.1016/j.jval.2019.02.012

Futoma, J. H. (2017). Learning to detect sepsis with a multitask Gaussian process RNN classifier. *arXiv preprint,, arXiv:1706.04152*.

Ghosh, S. S. (2025). An Overview of Optimization Techniques for Pre-processing of RNA-seq Data . *Swarm Optimization for Biomedical Applications*, 153-171.

Goldstein, B. A. (2017). Opportunities and challenges in developing risk prediction models with electronic health records data: A systematic review. Journal of the American Medical Informatics Associati. *Journal of the American Medical Informatics Association, 24*(1), 198-208. doi:https://doi.org/10.1093/jamia/ocw042

He, H. B. (2008). ADASYN: Adaptive synthetic sampling approach for imbalanced learning. *Proceedings of the IEEE International Joint Conference on Neural Networks* (s. 1322–1328). IEEE. doi:https://doi.org/10.1109/IJCNN.2008.4633969

Jakobsen, J. C. (2017). When and how should multiple imputation be used for handling missing data in randomised clinical trials–a practical guide with flowcharts. *BMC Medical*

Research Methodology, 17(1), 162. doi:https://doi.org/10.1186/s12874-017-0442-1

James, G. W. (2013). *An introduction to statistical learning: With applications in R.* Springer. doi:https://doi.org/10.1007/978-1-4614-7138-7

Jensen, P. B. (2012). Mining electronic health records: towards better research applications and clinical care. *Nature Reviews Genetics, 13*(6), 395–405. doi:https://doi.org/10.1038/nrg3208

Kaufman, S. R. (2012). Leakage in data mining: Formulation, detection, and avoidance. *ACM Transactions on Knowledge Discovery from Data (TKDD), 6*(4), 1-21. doi:https://doi.org/10.1145/2382577.2382579

Kuhn, M. &. (2013). *Applied predictive modeling* (Cilt 26). New York: Springer.

Lee, J. &. (2017). Medical big data: Promise and challenges . *Kidney Research and Clinical Practice, 36*(1), 3-11. doi:https://doi.org/10.23876/j.krcp.2017.36.1.3

Leys, C. L. (2013). Detecting outliers: Do not use standard deviation around the mean, use absolute deviation around the median. *Journal of Experimental Social Psychology,, 49*(4), 764–766.

Little, R. J. (2019). *Statistical analysis with missing data (3rd ed.).* Hoboken, NJ: Wiley.

Mani, I. &. (2003). kNN approach to unbalanced data distributions: a case study involving information extraction. . *Proceedings of workshop on learning from imbalanced datasets, 126*, 1-7.

Rajkomar, A. D. (2018). Machine learning in medicine . *New England Journal of Medicine, 380*(14), 1347–1358. doi:https://doi.org/10.1056/NEJMra1814259

Schulam, P. &. (2017). Reliable decision support using counterfactual models. *Advances in Neural Information Processing System*, *30.* doi:9781510860964

Shalabi, L. A. (2006). Data mining: A preprocessing engine . *Journal of Computer Science, 2*(9), 735–739. doi:https://doi.org/10.3844/jcssp.2006.735.739

Shickel, B. T. (2018). Deep EHR: A survey of recent advances in deep learning techniques for electronic health record (EHR) analysis. *IEEE Journal of Biomedical and Health Informatics, 22*(5), 1589–1604.

Sterne, J. A. (2009). Multiple imputation for missing data in epidemiological and clinical research: potential and pitfalls. Bmj. *Bmj, 338*, 2393. doi: https://doi.org/10.1136/bmj.b2393

Tracy, C. M. (2012). ACCF/AHA/HRS focused update of the 2008 guidelines for device-based therapy of cardiac rhythm abnormalities. *Journal of the American College of Cardiology, 60*(14), 1297–1313.

Vabalas, A. G. (2019). Machine learning algorithm validation with a limited sample size. *PloS One, 14*(11), 1-20. doi:https://doi.org/10.1371/journal.pone.0224365

Van Buuren, S. &.-O. (2011). Mice: Multivariate imputation by chained equations in R. *Journal of Statistical Software, 45*(3), 1-67. doi:https://doi.org/10.18637/jss.v045.i03

Wiens, J. S.-V. (2019). Do no harm: a roadmap for responsible machine learning for health care. *Nature Medicine, 25*(9), 1337–1340. doi: https://doi.org/10.1038/s41591-019-0548-6

Zheng, A. &. (2018). *Feature engineering for machine learning: principles and techniques for data scientists.* . O'Reilly Media, Inc.

Zimek, A. S.-P. (2012). A survey on unsupervised outlier detection in high-dimensional numerical data. *Statistical Analysis and Data Mining: The ASA Data Science Journal, 5*(5), 363–387. doi:https://doi.org/10.1002/sam.11161

YENİ NESİL YAPAY ZEKÂ YAKLAŞIMLARI

YAZILIM, KARİYER VE SİNİRSEL MİMARİLER ÜZERİNE

BİDGE