T.C.

YILDIZ TEKNİK ÜNİVERSİTESİ

FEN BİLİMLERİ ENSTİTÜSÜ

ÖNERİ SİSTEMLERİNDE OTOKODLAYICI TABANLI DERİN ÖĞRENME YAKLAŞIMI

Uğuray DURDU

YÜKSEK LİSANS TEZİ

İstatistik Anabilim Dalı

İstatistik Programı

Danışman

Doç. Dr. Gülder KEMALBAY

Şubat, 2022

T.C.

YILDIZ TEKNİK ÜNİVERSİTESİ

FEN BİLİMLERİ ENSTİTÜSÜ

ÖNERİ SİSTEMLERİNDE OTOKODLAYICI TABANLI DERİN ÖĞRENME YAKLAŞIMI

Uğuray DURDU tarafından hazırlanan tez çalışması 14.02.2022 tarihinde aşağıdaki jüri tarafından Yıldız Teknik Üniversitesi Fen Bilimleri Enstitüsü İstatistik Anabilim Dalı, İstatistik Programı **YÜKSEK LİSANS TEZİ** olarak kabul edilmiştir.

Doç. Dr. Gülder KEMALBAY
Yıldız Teknik Üniversitesi
Danışman

Jüri Üyeleri

Doç. Dr. Gülder KEMALBAY, Danışman
Yıldız Teknik Üniversitesi _____

Doç. Dr. Atıf Ahmet EVREN, Üye
Yıldız Teknik Üniversitesi _____

Dr. Öğretim Üyesi İlker ÜSTOĞLU, Üye
İstanbul Teknik Üniversitesi _____

Danışmanım Doç. Dr. Gülder KEMALBAY sorumluluğunda tarafımca hazırlanan Öneri Sistemlerinde Otokodlayıcı Tabanlı Derin Öğrenme Yaklaşımı başlıklı çalışmada veri toplama ve veri kullanımında gerekli yasal izinleri aldığımı, diğer kaynaklardan aldığım bilgileri ana metin ve referanslarda eksiksiz gösterdiğimi, araştırma verilerine ve sonuçlarına ilişkin çarpıtma ve/veya sahtecilik yapmadığımı, çalışmam süresince bilimsel araştırma ve etik ilkelerine uygun davrandığımı beyan ederim. Beyanımın aksinin ispatı halinde her türlü yasal sonucu kabul ederim.

Uğuray DURDU

İmza

Bu çalışma, Yıldız Teknik Üniversitesi Bilimsel Araştırma Proje Koordinatörlüğü'nün FYL-2021-4281 numaralı projesi ile desteklenmiştir.

Aileme

ve

dostlarıma

TEŞEKKÜR

Lisans öğrenimim ve yüksek lisans tez çalışmam boyunca gösterdiği her türlü destek, hoşgörü ve yardımlarından dolayı sadece tez sürecimde değil kendisini tanıdığım ilk günden beri her zaman yanımda olup beni destekleyip yardımlarını esirgemeyen bilgi ve tecrübelerinden faydalandığım çok değerli, kıymetli danışman hocam sayın Doç. Dr. Gülder KEMALBAY'a; derin bilgi birikimi ve çok yönlü yaklaşımı ile bizi her daim aydınlatan ve şaşırtan sayın hocam Doç. Dr. Atıf Ahmet EVREN'e; İTÜ Kontrol ve Otomasyon Mühendisliği Bölümü'nün kıymetli hocası olan ve tez konusundaki görüşlerini bize aktaran sayın Dr. Öğretim Üyesi İlker ÜSTOĞLU'na ve Yıldız Teknik Üniversitesi İstatistik Bölümü'nden çok kıymetli sayın hocalarım Arş. Gör. Dr. Coşkun PARİM'e, Arş. Gör. Mehmet Şamil GÜNEŞ'e ve Arş. Gör. Batuhan ÖZKAN'a beni daima desteklediği, bana yol gösterdiği ve her konuda destekçim olduğu için teşekkürü borç bilirim.

Hayatım boyunca bana yol gösteren, destek olan, beni bugünlere getiren ve varlığını arkamda hissettirip bana daima güç veren haklarını ödeyemeyeceğim sevgili annem ve babama; muhteşem bir ağabey olan ve bilgisinden, yol göstericiliğinden, fikirlerinden daima yararlandığım sevgili ağabeyim Anıl DURDU'ya ve tez sürecinde desteklerini esirgemeyip tanıdığım günden beri varlığını daima hissettiren sevgili Meltem BOZ'a minnettarım.

Ayrıca, tezimi okuma nezaketinde bulunan can dostum Hasan ÖRGEV'e tez ile ilgili geri dönüşleri için teşekkür ediyorum. Aynı zamanda, sevgili Ali AKGÜN ve Onur ULUÇINAR gibi dostlarım olduğu için de çok şanslıyım. Ek olarak tez sürecimde beni destekleyen Yıldız Teknik Üniversitesi Bilimsel Araştırma Proje Koordinatörlüğü'ne katkılarından dolayı teşekkür ediyorum.

Beni seven ve benim sevdiğim, hayatımı güzelleştiren güzel, iyi yürekli ve bu çalışmanın faydasının dokunacağı insanlara çalışmamı atfediyorum.

Uğuray Durdu

İÇİNDEKİLER

SİMGE LİSTESİ viii
KISALTMA LİSTESİ ix
ŞEKİL LİSTESİ x
TABLO LİSTESİ xi
ÖZET xii
ABSTRACT xiv
1 GİRİŞ 1
1.1 Literatür Özeti……..1
1.2 Tezin Amacı……7
1.3 Hipotez……8
2 ÖNERİ SİSTEMLERİ 9
2.1 Öneri Sistemlerindeki Yaklaşımlar……10
2.1.1 İşbirliğine Dayalı Filtreleme……10
2.1.1.1 Bellek Tabanlı Yöntemler……11
2.1.1.2 Model Tabanlı Yöntemler…… 12
2.1.2 İçeriğe Dayalı Filtreleme……12
2.1.3 Hibrit Sistemler……13
2.1.4 Bilgiye Dayalı Sistemler……14
2.2 Öneri Sistemlerinde Karşılaşılan Problemler……14
2.2.1 Veri Seyrekliği……14
2.2.2 Soğuk Başlangıç Problemi……15
2.2.3 Ölçeklenebilirlik……16
2.3 Geri Bildirimler……16
2.3.1 Geri Bildirim Yaklaşımları……16
2.3.1.1 Açık Geri Bildirim……17
2.3.1.2 Örtük Geri Bildirim……18
2.3.2 Geri Bildirim Yaklaşımlarının Problemleri……19
3 OTOKODLAYICILAR 21
3.1 Basit Otokodlayıcılar……21
3.2 Derin Otokodlayıcılar……23
3.3 Basit Yarı-Otokodlar……26

3.4 Derin Yarı-Otokodlayıcılar……28
3.5 Otokodlayıcıların Kullanım Alanları……30
3.6 Yoğun Yeniden Besleme……31
3.7 Kayıp Fonksiyonu……31
3.8 Düzenlileştirme……32
3.9 Bilgi Kriterleri……34
3.10 MinMax Ölçekleyici……34
3.11 Aktivasyon Fonksiyonları……34
3.12 Adam Algoritması……35
3.13 Performans Metrikleri……36
4 UYGULAMA **38**
4.1 Geliştirme Araç ve Gereçleri……38
4.2 Veri Seti ve Ön İşleme……38
4.2.1 Kullanıcı Verisi……39
4.2.2 Öğe Verisi……41
4.2.3 Kullanıcı Verisi……43
4.3 Veri Setini Parçalama……45
4.4 Nihai Veri Setinin Elde Edilmesi……46
4.5 Performans Metriklerinin Hesaplanması……46
4.6 Deneysel Sonuçlar……49
4.6.1 Düğüm Sayısına Karar Verilmesi……50
4.6.2 Katman Sayısına Karar Verilmesi……52
4.6.3 Model Mimarisinin Oluşturulması……52
4.6.4 Hiperparametre Optimizasyonu……54
4.6.5 Temel ve Yeniden Beslemeli Modelin Karşılaştırılması……57
4.6.6 Performans Sonuçları……59
4.7 Modelden Öğe Tahminlerinin Alınması……62
5 SONUÇ VE ÖNERİLER **65**
KAYNAKÇA **66**
TEZDEN ÜRETİLMİŞ YAYINLAR **70**

SİMGE LİSTESİ

x	Giriş Verisi
$sub(x)$	Giriş Verisinin Bir Alt Kümesi
Θ^i	i.Katmanın parametre kümesi
W_e^i	i.Kodlayıcı Katmanın Ağırlık Matrisi
b_e^i	i.Kodlayıcı Katmanın Yan Vektörü
W_d^i	i.Kod Çözücü Katmanın Ağırlık Matrisi
b_d^i	i.Kod Çözücü Katmanın Yan Vektörü
$\mathcal{L}$	Kayıp Fonksiyonu
f_d	Kod Çözücü Fonksiyonu
d	Kod Çözücü Katmanı
g_e	Kod Çözücü Katmanı Aktivasyon Fonksiyonu
D	Kod Çözücü Katmanı Boyutu
h	Kod Katmanı
H	Kod Katmanı Boyutu
f_e	Kodlayıcı Fonksiyonu
E	Kodlayıcı Katman Boyutu
e	Kodlayıcı Katmanı
g_d	Kodlayıcı Katmanı Aktivasyon Fonksiyonu
l	Kodlayıcı ve Kod Çözücü Katman Sayısı
L	Toplam Katman Sayısı
x'	Yeniden Yapılandırılmış Veri

KISALTMA LİSTESİ

BCE	İkili Çapraz Entropi (Binary Cross Entropy)
CE	Çapraz Entropi (Cross Entropy)
KL	Kullback–Leibler
L-BFGS	Limited-memory Broyden Fletcher Goldfarb Shanno
MAP	Ortalama Kesinlik (Mean Average Precision)
MF	Matris Çarpanlarına Ayırma (Matrix Factorization)
MMSE	Maskelenmiş Hata Kareler Ortalaması (Masked Mean Squared Error)
MNAR	Rastgele Olmayan Kayıp (Missing not at Random)
MRR	Ortalama Karşılıklı Sıralama (Mean Reciprocal Rank)
MSE	Hata Kareler Ortalaması (Mean Square Error)
NDCG	Normalize İndirgenmiş Kümülatif Kazanç (Normalized Discounted Cumulative Gain)
RMSE	Kök Ortalama Kare Hatası (Root Mean Square Error)
Rprop	Esnek Yayılım (Resilient Propagation)
SVD	Tekil Değer Ayrışımı (Singular Value Decomposition)

ŞEKİL LİSTESİ

Şekil 2.1 Öneri sistemlerinin sınıflandırma şeması……10
Şekil 3.1 Basit otokodlayıcı……22
Şekil 3.2 Derin otokodlayıcı……24
Şekil 3.3 4 katmanlı derin otokodlayıcı akış şeması……25
Şekil 3.4 Çıkış katman boyutu girdi boyutundan büyük olan basit otokodlayıcı…27
Şekil 3.5 Basit yarı-otokodlayıcı ……27
Şekil 3.6 Derin yarı-otokodlayıcı……29
Şekil 4.1 Kullanıcı verisi……39
Şekil 4.2 Ön işlemeden sonraki kullanıcı verisi……40
Şekil 4.3 Öğe verisi……41
Şekil 4.4 Ön işlemeden sonraki kullanıcı bazlı öğe verisi……42
Şekil 4.5 Etkileşim verisi……43
Şekil 4.6 Ana akış şeması……48
Şekil 4.7 Basit yarı-otokodlayıcı örnek mimarisi……51
Şekil 4.8 4 katmanlı derin yarı-otokodlayıcı model mimarisi……53
Şekil 4.9 Yeniden beslemeli derin yarı-otokodlayıcı model mimarisi……57
Şekil 4.10 Temel ve yeniden beslemeli modelin doğrulama kaybı üzerindeki etkisi……58
Şekil 4.11 Doğrulama setinin temel model üzerindeki performansı……61
Şekil 4.12 Doğrulama setinin yeniden beslemeli model üzerindeki performansı.62
Şekil 4.13 Model tahmin sonuçları……64

TABLO LİSTESİ

Tablo 4.1 Veri seti……..……..……38
Tablo 4.2 Kullanıcı verisinin içeriği……40
Tablo 4.3 Öğe verisinin içeriği……..41
Tablo 4.4 Etkileşim verisinin içeriği…… ……..…43
Tablo 4.5 Filtreleme sonrası veri miktarları ve seyreklik oranı…….……...44
Tablo 4.6 Parçalanmış veri setlerinin içerdiği kullanıcı sayıları……..45
Tablo 4.7 Deneysel sonuçlarda kullanılan parametreler……....49
Tablo 4.8 Bazı değişken ve açıklamaları…….……..……50
Tablo 4.9 Gizli düğüm sayısına göre 2 katmanlı modelin performans sonuçları..51
Tablo 4.10 Katman sayısına göre modelin performans sonuçları…….……52
Tablo 4.11 Parametre arama aralıkları……..55
Tablo 4.12 Bazı parametre tercihleri……..……55
Tablo 4.13 Hiperparametre optimizasyonuyla belirlenen parametreler…..……56
Tablo 4.14 Temel ve yeniden beslemeli modelin etkisi……..…58
Tablo 4.15 Performans sonuçları……..……60

ÖZET

Öneri Sistemlerinde Otokodlayıcı Tabanlı Derin Öğrenme Yaklaşımı

Uğuray DURDU

İstatistik Anabilim Dalı

Yüksek Lisans Tezi

Danışman: Doç. Dr. Gülder KEMALBAY

Günümüzde artan internet kullanımı ile kullanıcıların öğeler ile etkileşime geçtikleri sosyal medya ve e-ticaret siteleri gibi birçok platform, öneri sistemleri sayesinde işlemesi zor ve büyük boyutlu veri seti içerisinden kullanıcılara kişiselleştirilmiş öneriler sunmaktadır. Kişiselleştirilmiş öneriler için sıklıkla kullanılan işbirlikçi öneri sistemlerinde, kullanıcı-öğe etkileşiminin az olduğu durumlarda öneri performansı düşerken etkileşimin hiç olmadığı durumlarda ise öneri yapılamamaktadır.

Bu tez çalışmasında, işbirlikçi öneri sistemlerinde ortaya çıkan veri seyrekliği ve soğuk başlangıç problemlerini azaltmak için otokodlayıcı tabanlı derin öğrenme yaklaşımı kullanılmıştır. Kullanılan yaklaşım, Zhang ve arkadaşları tarafından 2017 yılında ortaya konulan basit yarı-otokodlayıcı yapısının derin yarı-otokodlayıcı olarak geliştirilmiş halidir. Geliştirilen derin ağ mimarisi, Kuchaiev ve Ginsburg tarafından 2017 yılında önerilen yeniden beslemeli algoritma ile eğitilmiştir. Tez çalışmasının yöntemi, literatürde mevcut olan Movielens 1M veri

seti üzerinde uygulanmıştır. Tez çalışmasının uygulama kısmı, Python programlama dilinde TensorFlow kütüphanesi aracılığı ile gerçekleştirilmiştir.

Elde edilen sonuçlara göre, geleneksel öneri sistemleri yerine otokodlayıcı tabanlı derin öğrenme yaklaşımı ile oluşturulan öneri sisteminin veri seyrekliği problemini azalttığı ve modele dâhil edilen yan bilgiler sayesinde herhangi bir öğe ile etkileşimi olmayan yeni bir kullanıcı için soğuk başlangıç problemini azaltarak bir öneri yapılabileceği gösterilmiştir.

Anahtar kelimeler: Öneri sistemleri, otokodlayıcılar, işbirliğine dayalı filtreleme, soğuk başlangıç problemi, veri seyrekliği problemi

ABSTRACT

Autoencoder Based Deep Learning Approach in Recommender Systems

Uğuray DURDU

Department of Statistics

Master of Science Thesis

Supervisor: Assoc. Prof. Dr. Gülder KEMALBAY

Today, with the increasing use of the internet, many platforms such as social media and e-commerce sites, where users interact with items, offer personalized recommendations to users from a data set that is difficult to process and large. In collaborative suggestion systems, which are frequently used for personalized suggestions, suggestion performance decreases in cases where user-item interaction is low, and suggestions cannot be made in cases where there is no interaction.

In this thesis, an autoencoder based deep learning approach is used to reduce data sparsity and cold start problems in collaborative recommendation systems. The approach used is the deep semi-autoencoder which is the development of the basic semi-autoencoder structure introduced by Zhang et al in 2017. The developed deep network architecture was trained with the dense refeeding algorithm proposed by Kuchaiev and Ginsburg in 2017. The method of the thesis was applied on the Movielens 1M dataset available in the literature. The application part of

the thesis was carried out in Python programming language through the TensorFlow library.

According to the results obtained, it has been shown that the recommendation system developed with an autoencoder based deep learning approach instead of traditional recommendation systems reduces the data sparsity problem, and thanks to the side information in the model, a recommendation can be made by reducing the cold start problem for a new user who does not interact with any item.

Keywords: Recommendation systems, autoencoders, collaborative filtering, cold start problem, data sparsity problem

1
GİRİŞ

1.1 Literatür Özeti

Otokodlayıcılar, öneri sistemlerinde sıkça kullanılan derin öğrenme yaklaşımlarıdır. Geleneksel öneri sistemlerinde; lineer olmayan özelliklerin yakalanamaması, soğuk başlangıç problemi ve derecelendirme matrisinin seyreklik problemleri mevcuttur. Literatürde yapılan çalışmalarda, veri seyrekliği ve soğuk başlangıç problemini çözmek için farklı yaklaşımlar geliştirilmiştir. Bu yaklaşımlar; kullanıcı ve öğe yan bilgilerini *(side information, user metadata, item metadata, demographic information, item information)* eklemek, kullanıcıların açıklanabilir skorlarını eklemek ve otokodlayıcılarda *yeniden besleme (re-feeding)* yaklaşımı uygulamaktır. Literatürde yer alan çalışmalar incelendiğinde;

Wilson ve arkadaşları, seyreklik sorununun kullanıcıya öğe öneri sınırlarını zorladığını, önerilerin kapsam ve doğruluğunu tehlikeye attığını söyler. Geleneksel öneri sistemlerinin doğrudan öğeler arasındaki benzerliklere dayandığından bahseder. Çalışmalarında, bu benzerlik bilgisini keşfetmek için öneri sırasında benzerliğe dayalı geri alma tekniği uygular. Bu yan bilgiyi de modele ekleyerek öneri yaklaşımlarının gücünü artırır [1].

Strub ve Mary, kullanıcı yan bilgilerini otokodlayıcı girişlerine eklemenin giriş derecelendirmelerinin seyrekliğini azaltmak ve soğuk başlangıç problemini çözmek için etkili olacağını belirtir [2].

Ouyang ve arkadaşları, kullanıcıya özgü bir otokodlayıcı sistemi geliştirir. Girdi olarak kullanıcılardan gelen tamsayı derecelendirmelerini olduğu gibi kullanmazlar. Önce her bir kullanıcının derecelendirmelerini 0 ve 1 ile temsil edilen bir vektöre dönüştürür. Örneğin, bir kullanıcı 1 kimlik numarasına sahip bir öğeye 4 ile geri bildirimde bulunduysa vektör [0, 0, 0, 1, 0] şeklinde olur. Elde edilen bu vektörlerden her biri kullanıcının derecelendirdiği bir öğeyi temsil eder. Böylece *(derecelendirilen toplam öğelerin sayısı x derecelendirme aralığı)*

boyutunda bir matris elde edilip otokodlayıcının bu derecelendirmeleri yeniden yapılandırması beklenir. Veri seti olarak *ML-100K/1M* kullanılmıştır [3].

Sedhain ve arkadaşları, [3] çalışmasından farklı olarak kullanıcı ya da öğe derecelendirme vektörlerini doğrudan alır ve çıktı katmanında yeniden yapılandırılmış derecelendirmeleri elde eder. Bu yaklaşımın *matris çarpanlarına ayırma (MF)* tekniğinden ve benzer tekniklerden daha iyi olduğunu öne sürer. *AutoRec* adını verdikleri bu model, girdi türüne bağlı olarak kullanıcı ya da öğe tabanlı olabilir. Modele yanlılık parametresi de eklenir. *Limited-memory Broyden Fletcher Goldfarb Shanno (L-BFGS)* optimizasyon fonksiyonu ile karşılaştırılabilir bir performans sağlanması için optimizasyon fonksiyonu olarak *Esnek yayılım (Rprop)* tercih edilir. Sonuç olarak bazı çıkarımlarda bulunulur:

- *Rprop* fonksiyonu*, L-BFGS* fonksiyonu ile karşılaştırılabilir performans sağlar ve çok daha hızlıdır.
- Öğe tabanlı otokodlayıcı genellikle, kullanıcı tabanlı otokodlayıcıdan daha iyi performans gösterir. Bunun nedeni her bir öğe için ortalama derecelendirme sayısının her kullanıcının derecelendirme sayısından çok daha fazla olmasıdır.
- Farklı optimizasyon fonksiyonları, modelin performansını etkiler.
- Nöron sayılarını artırmak ya da yoğun katman sayılarını artırmak sonuçları iyileştirir.

Bu modelin, işbirliğine dayalı filtreleme yaklaşımlarına göre temsil ve hesaplama avantajlarına sahip olduğunu, ayrıca mevcut en son yöntemlerden daha iyi performans sağladığını gösterirler*. ML-1M/10M* ve *Netflix* veri setleri üzerinde çalışıp açık geri bildirim derecelendirmelerini kullanmışlardır [4].

Ferreira ve arkadaşları, işbirlikçi filtrelemenin bu tür sistemlerde yaygın olarak kullanıldığını ancak yüksek boyutlar ve veri seyrekliğinin her zaman ana problem olduğundan bahseder. Bu modelin yine, öneri sistemlerinde sıkça kullanılan *Tekil Değer Ayrışımı* (*SVD)* ile bir karşılaştırması yapılır [5]. Veri seti, eğitim ve test olarak %80-%20 oranında rastgele bölünür. Sonuçların performansını ölçmek için *kesinlik (precision)* ve *duyarlılık (recall)* metrikleri kullanılır. Daha fazla yoğun katman eklemenin ise kaybı iyileştirmediği ortaya koyulur. Farklı olarak

kullanıcılardan alınan açık geri bildirimler, (0.2 – 1) arasına alınarak modele girdi olarak verilir. Bu şekilde yapılan bir ön işlemenin kaybı iyileştirdiği görülür. *ML-1M/10M* veri setleri üzerinde çalışılmıştır [6].

İşbirliğine dayalı filtreleme doğrultusunda Hasan ve arkadaşları, otokodlayıcıdan yararlanarak kullanıcılar arasında benzerlik hesaplanmasının yapıldığı bir strateji önerir. Veri seti seyrek olduğunda veya sisteme yeni bir kullanıcı girdiğinde kullanılabilecek hiçbir benzerlik ölçütü mevcut değildir. Bunun üstesinden gelebilmek için kullanıcıların derecelendirmelerini verimli bir şekilde hesaplayabilen bir otomatik kodlayıcı kullanılarak kullanıcıların kod katmanından alınan özelliklerine göre benzerlik skorları hesaplanır. Veri seti, %80-%20 oranında rastgele bölünür. Performans metriği olarak *doğruluk (accuracy),* duyarlılık, F1 skoru ve *Ortalama Karşılıklı Sıralama (MRR)* kullanılır*. ML-1M* veri seti üzerinde çalışılmıştır [7].

Yine bu doğrultuda Haghighi ve arkadaşları, komşuluğa dayalı açıklama yaklaşımından esinlenerek açıklanabilir bir öneri sistemi tasarlar. En doğru öneri modellerinin, öğelerin bir kullanıcıya neden önerildiğini gösteremeyen kara kutular olma eğiliminde olduğunu vurgular. İçerik gibi yan bilgiler kullanılmadan açıklanabilen ve ilk n doğru önerileri bulan bir otokodlayıcı modeli önerir. Cevaplanmaya çalışılan araştırma soruları ise şunlardır:

- Otokodlayıcı mimarisini kullanarak işbirliğine dayalı açıklanabilir bir öneri sistemi oluşturabilir mi?
- Açıklanabilir otokodlayıcı, klasik otokodlayıcı ile karşılaştırılabilir bir doğruluk sağlayabilir mi?

Veri seti, %90-%10 oranında rastgele bölünür. Açık geri bildirimler, (0-1) arasında normalleştirilir. Açıklama puanları, işbirliğine dayalı filtreleme komşu mantığına dayalı olarak çevrimdışı hesaplanır. En yakın komşular, kosinüs benzerliğine göre belirlenir. Sonuç olarak kullanıcı derecelendirmelerini hem yeniden yapılandırılacak girdilere hem de kullanıcı-komşu işbirliğine dayalı filtrelemeyi yan bilgi olarak gerekçelendirilmiş açıklanabilirlik puanlarına dâhil eden bir otokodlayıcı yaklaşımı önerir. Yapılan deneyler, bu yaklaşımın çeşitli performans ölçüm parametreleri için *Kök Ortalama Hata Karesi* (*RMSE),* açıklanabilirlik skoru

ve *Ortalama Kesinlik* (*MAP)* gibi ölçümler açısından temelden daha iyi performans gösterdiğini kanıtlar. Bu modelde ek bilgiler, benzer şekilde derecelendirmelerin seyrekliğini telafi etmeye yardımcı olabilecek açıklanabilirlik puanlarından oluşur. Bir öğenin bir kullanıcıya açıklanabilirliği, doğrudan o öğe üzerindeki en yakın komşu kullanıcıların derecelendirmelerini toplayarak işbirliğine dayalı filtreleme mantığından türetilir. Sonuçlar, derecelendirmelerle aynı boyutta olan ve önceden hesaplanmış açıklanabilirlik puanlarının girdi katmanına beslenen bir ek bilgi olarak kullanılmasının otokodlayıcının aynı anda daha doğru ve açıklanabilir tahminler yapmasına yardımcı olduğunu gösterir. *ML-100K* veri seti üzerinde çalışılmıştır [8].

Strub ve arkadaşları, [8]'deki mimariyi eksik değerlere sahip verileri girmek için uyarlanmış bir kayıp işlevi kullanarak ve yan bilgileri de dâhil ederek geliştirir. Deneyler, yan bilgilerin tüm kullanıcılar/öğeler üzerinde ortalama test hatasını biraz iyileştirdiğini gösterirken soğuk kullanıcılar/öğeler üzerinde daha fazla etkiye sahip olduğunu gösterir. İçerik tabanlı ve işbirliğine dayalı filtrelemenin gücünden yararlanılır. İçerik tabanlı filtreleme, yeni bir derecelendirmeyi tahmin etmek için kullanıcı/öğe bilgisini kullanır. Örneğin kullanıcı bilgisi; kullanıcının yaşı, cinsiyeti, mesleği ve posta kodu olabilir. Öğe bilgisi ise öğenin türü, açıklaması veya etiketleri olabilir. Buna karşın işbirliğine dayalı filtreleme, kullanıcı ve öğelerin derecelendirme geçmişini kullanır. İçerik tabanlı filtrelemenin bir sınırlaması olan soğuk başlangıç sorununun üstesinden gelmek için yan bilginin eğitim sürecine dâhil edildiği bir hibrit model oluşturulur. Amaç, yan bilgiler sayesinde derecelendirme eksikliğini ortadan kaldırıp -[4]'deki yaklaşımın tahmin edilemeyen derecelendirmeleri rastgele seçilen bir puanla değiştirdiğinin bilinmesi üzerine- soğuk başlangıç probleminin etkisini azaltmak için yan bilgileri entegre etmektir. Diğer uygulamaların aksine yan bilgiler, bazı sınırlamalar yapılarak her katmana beslenir. Veri seti, eğitim ve test olarak %90-%10 oranında rastgele bölünür. Sonuç olarak yan bilgi eklemenin kaybı artırdığı kanıtlanır. *ML-1/10/20* veri setleri üzerinde çalışılmıştır [9].

Zhang ve arkadaşları, yeni bir otokodlayıcı yapısına dayalı hibrit filtreleme modeli önerir. Kişiselleştirilmiş öneriler üretmek için hem içerik bilgisini hem de doğrusal olmayan özellikleri kullanır. Girdilerden farklı alt kümeler örneklenerek farklı

temsiller ve yeniden yapılandırmalar esnek bir şekilde yakalanır. Genel olarak otokodlayıcıların giriş ve çıkış boyutlarının aynı olduğundan bahsedilir, ancak bunu uygulamanın zorunlu olmadığı söylenir. Bu sayede, giriş ve çıkış boyutlarının sınırlaması kırılarak giriş katmanına yan bilgi eklemek için farklı otokodlayıcı çeşitlerinin kurulabileceği belirtilir. Çıktı katmanının giriş katmanından daha kısa veya daha uzun olacağı iki durum oluşur. İlk durumda, çıktı katmanı girdi katmanından daha büyük bir boyuta sahip olup bazı yeni özellikler oluşturur, fakat çıkarılan bu yeni özellikler hakkında yorum yapmak zordur. İkinci durumda ise çıktı katmanı girdi katmanından daha küçük bir boyuta sahip olup girdilerin sadece istenilen bölümünü yapılandırmak için kullanılır. Böylece, yan bilgilerin gücünden yararlanılarak kod katmanında ortaya çıkarılan gizli özelliklerin çıktı katmanında sadece istenilen boyuttaki özellikleri tahmin etmesi sağlanır. Yan bilgilerden kullanıcı ya da öğe özellikleri aynı anda kullanılmaz. Kullanıcı özellikleri eklenirse bir sıralama tahmini; öğe özellikleri eklenirse bir derecelendirme tahmini yapılır. Sıralama tahmini yapılırken 4'ten büyük derecelendirmeler 1, 4'ten küçük derecelendirmeler ise 0 olarak üstü kapalı geri bildirimlere dönüştürülür, ancak derecelendirme tahmini yapılırken doğrudan geri bildirimler kullanılır. Çoğu mevcut çalışmaların bu sorunlardan birini çözmeyi amaçlamadığı söylenir. Bu yarı-otokodlayıcıya dayalı hibrit modelin her ikisini de çözebildiği gösterilir. Kullanıcının en çok sevdiği ilk n öğeyi tahmin edip test verilerine göre değerlendirme amaçlanır. Sıralama tahmininde duyarlılık metriği, derecelendirme tahminlerinde ise *RMSE* kullanılmıştır. *ML-100K/1M* veri seti üzerinde çalışılmıştır [10].

Zhang ve arkadaşlarına göre işbirlikçi filtreleme, doğrudan geri bildirimleri yaygın olarak benimseyen öneri tekniklerinden biridir, ancak tıklama verileri gibi örtük geri bildirimlerin kullanıcının gizli tercihlerini keşfedebileceğine inanılır. Son zamanlarda, kullanıcı tıklama ve göz atma verilerini içeren örtük geri bildirimlerin kullanıcının potansiyel tercihlerini içerdiği düşünülür. Kaybı en aza indirmek amacıyla gizli değişkenlerin, öğrenilmiş ve temelde yatan gerçek veri dağılımı arasındaki *Kullback–Leibler (KL)* sapmasını hesaplamak için uygun mesafe ölçütleri belirtilir. Gizli değişkenlerin dağılımının örtük geri bildirimlerden tahmin edilebileceği varsayılır. Gizli değişkenler tahmini dağılımdan örneklenir ve

yeniden yapılandırılacak veriler, çok katmanlı sinir ağı aracılığıyla gizli değişkenlerden çözülür. Oluşturulan gizli değişkenlerin seyrek bir temsil formuna izin vermek için *L1 düzenlileştirme (regularization)* terimi getirilerek yeni kayıp fonksiyonu önerilir. Bildikleri kadarıyla bu yaklaşım *Wasserstein* uzaklığını otokodlayıcılar üzerinden işbirlikçi filtreleme problemine uyarlamak için yapılan ilk girişimdir. Birden çok girdiyi aynı çıktıya yeniden yapılandırmanın yapılandırma hatasını artırdığından Wasserstein otokodlayıcı yaklaşımı önerilir. Performans metriği olarak duyarlılık ve *Normalize İndirgenmiş Kümülatif Kazanç (NDCG)* kullanılır. *ML-20M*, *Netflix Prize* ve *LastFM* veri setleri üzerinde çalışılmıştır [11].

Kuchaiev ve Ginsburg, işbirliğine dayalı filtrelemeden kaynaklanan doğal seyreklik probleminin üstesinden gelmek için çıktıyı modele yeniden beslemeye dayalı yeni bir otokodlayıcı yaklaşımı önerir. Altı katmanlı derin otokodlayıcı kullanılan bu modelin daha iyi genelleştirme sağladığı ve diğer otokodlayıcılara göre daha iyi performans gösterdiği iddia edilir. Kullanılan lineer olmayan aktivasyon fonksiyonlarıyla birlikte, yüksek oranda düğüm seyreltme ve düzenlileştirme teknikleri de uygulanır. Sadece yeniden besleme uygulamanın model performansını önemli derecede etkilemediği, ancak yüksek bir düğüm seyreltme oranıyla bu performans farkının daha net bir şekilde ortaya çıktığı gösterilir. Düğüm seyreltme oranının az olması ya da sıfır olması, giriş katmanının çıkış katmanına bir nevi kopyalanması anlamı taşıdığından modele olumlu bir etki sağlamaz. Aynı zamanda aşırı uyum/aşırı öğrenme/ezberleme sağladığından düğüm seyreltme gibi tekniklerinin yoğun kullanılması gerektiği söylenir. Farklı aktivasyon fonksiyonlarının ve yoğun katmanların kullanılması durumunda elde edilen sonuçlar da grafiklerle gösterilir. Yeniden beslemeli bu modelin eğitimi hızlandırdığı ve model performansını iyileştirdiği görülür. Açık geri bildirimlerin bulunduğu *Netflix Prize* veri seti üzerinde çalışılıp *Maskelenmiş Hata Kareler Ortalaması (MMSE)* performans metriği olarak kullanılmıştır [12].

Örtük geri bildirimlerle yapılan çalışmalarda, geri bildirimlerin kullanım biçimlerine yer vermek gerekirse;

Adomavicius ve Kwon, 1 ile 5 arasındaki tamsayı derecelendirmelerinden 3.5'ten büyük olanları yüksek, düşük olanları ise düşük dereceli olarak tanımlar. Gerçek dünya problemlerinde tavsiye sistemlerinin genelde, en yüksek dereceli öğeleri önerdiğinden bahseder, çünkü kullanıcıların genellikle en alakalı birkaç öneriyle ilgilendiği gerçeği vardır. Performans ölçümünde de en yüksek dereceli *N* önerinin yüzdesine bakarak değerlendirme yapar [13].

Wang ve arkadaşları, öneri sistemlerinde açık geri bildirimlere kıyasla örtük geri bildirimlerin daha yaygın olarak kullanıldığını ve çok sayıda *Rastgele Olmayan Kayıp (MNAR)* içerdiğini söyler. MNAR, Y değişkenindeki kayıp (eksik) gözlem ihtimâlinin Y ile ilgili olduğu hâlde diğer gözlemlerle alakalı olmaması durumudur. Örneğin, bir sınav esnasında bazı soruları zor bulan öğrenciler o soruları cevaplamayabilir. Eksik değerler, bilinmeyen ve olumsuz geri bildirimlerin bir karışımı olup kullanıcının öğeyi beğenmediğini veya öğeyle etkileşime bile geçmediğini gösterir. Bu da kullanıcıların olumsuz tercihlerini öğrenmeyi zorlaştırır. Yalnızca olumlu geri bildirimler söz konusu olduğundan tüm olumsuz geri bildirimler eksiktir. Eksik geri bildirimlerin olumsuz geri bildirimler olarak kabul edilmesi olumlu geri bildirimleri bastırdığından yanlı önerilere yol açar. 1 ile 5 arasındaki tamsayı derecelendirmelerinden 3 ve 3'ten büyük olanlar 1, küçük olanlar ise 0 olarak örtük geri bildirimlere dönüştürülür. *ML-100K/1M* ve *LastFM* veri setleri üzerinde çalışılmıştır [14].

1.2 Tezin Amacı

Bu tez çalışmasının amacı, işbirliğine dayalı filtreleme yönteminde ortaya çıkan *veri seyrekliği (data sparsity)* ve *soğuk başlangıç (cold start)* problemini azaltmaktır. İşbirliğine dayalı filtreleme, bir kullanıcının bir öğe üzerindeki tercihlerini tahmin ederken kullanıcı ile benzer davranış eğilimi gösteren diğer kullanıcıların öğeler üzerindeki etkileşim verilerini göz önüne alır ancak kullanıcı-öğe etkileşim verileri genellikle seyrektir. Veri seyrekliği problemi, yöntemin öğrenme performansında düşüklüğe sebep olur. Karşılaşılan diğer bir problem ise kullanıcı-öğe etkileşimine dayalı öneri sistemlerinde herhangi bir öğe ile etkileşimi olmayan bir kullanıcı veya herhangi bir kullanıcı ile etkileşimi olmayan bir öğe sisteme dâhil olduğunda yeni kullanıcı veya yeni öğe için öneri tahmin

edilememesidir. Soğuk başlangıç problemi olarak bilinen bu durum, öneri sisteminin tahmin görevini zorlaştırır. Tezin amacına ulaşmak için kullanıcı-öğe etkileşimine dayalı olan işbirlikçi öneri sistemlerinde, kullanıcı-öğe etkileşim verisine ilaveten kullanıcı-öğe yan bilgilerinin de dâhil edilmesi ile oluşan bir otokodlayıcı yaklaşımı geliştirilmesi hedeflenmektedir. Böylece veri seyrekliği ve soğuk başlangıç problemleri azalırken öneri performansında iyileştirme gerçekleşecektir.

1.3 Hipotez

Kuchaiev ve Ginsburg [12] tarafından önerilen yeniden beslemeli derin otokodlayıcı yaklaşımının veri seyrekliği problemine çözüm olduğu bilinmektedir ancak bu problemi çözmek önerilerin doğruluğunu artırmaya ve soğuk başlangıç problemini çözmeye yetmez. Öneri tahmin performansını arttırmak için bu problemlerin birlikte ele alınması gerekir. Bu tez çalışmasının hipotezi, yeniden beslemeli derin otokodlayıcı modeline kullanıcı-öğe etkileşim verileri yanı sıra kullanıcı ve öğe yan bilgilerini de dâhil ederek oluşturulacak modelin işbirlikçi öneri sisteminde ortaya çıkan veri seyrekliği ve soğuk başlangıç problemlerini azaltabileceğini savunmaktadır. Oluşturulacak model, Zhang ve arkadaşları [10] tarafından önerilen basit yarı-otokodlayıcı yapısının derin yarı-otokodlayıcı olarak geliştirilmiş halidir. Hipotezin doğruluğunu sınamak için model aracılığı ile öneriler tahmin edilirken kullanıcının sadece istenen öğelerle etkileşime geçtiği varsayılıp ayrıca nasıl bir kullanıcı olduğunu bilgisi de modele verilip (örneğin yaşı, mesleği, cinsiyeti, etkileşime geçtiği varsayılan öğelerin türü gibi) böyle bir kullanıcıya önereceği öğeler yüksek olasılıktan düşük olasılığa doğru sıralanacaktır. Etkileşime geçtiği varsayılan öğeler ise kullanıcıya gösterilecek öğelerden düşürülecektir. Bununla birlikte, herhangi bir öğe ile etkileşimi veya kendisi hakkında bilgi bulunmayan bir kullanıcıya da birtakım öğeler önerilebilecektir.

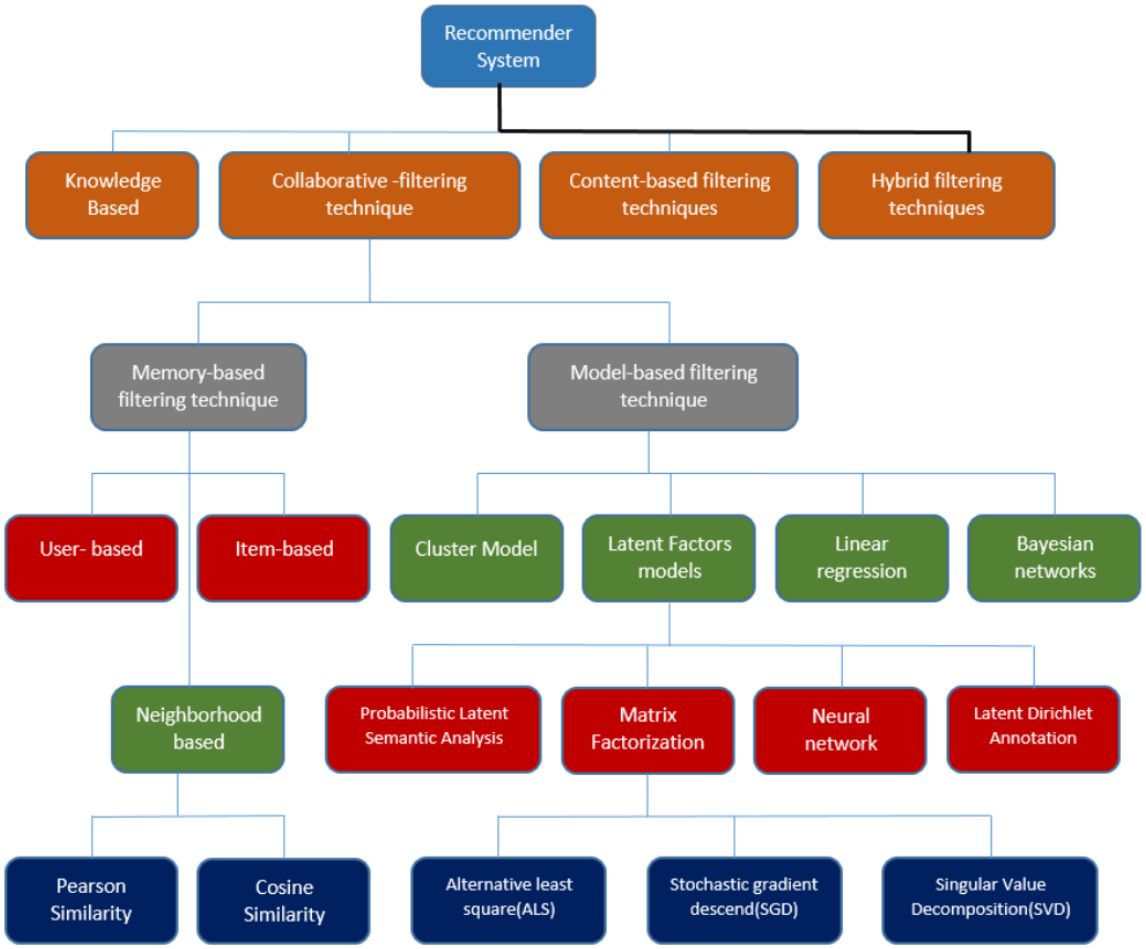
2
ÖNERİ SİSTEMLERİ

Öneri sistemleri, amaçlara bağlı olarak çalışan bireyselleştirme yaklaşımlarıdır. Öneri sistemlerini bireyselleştiren süreç, ürünlere tıklamayla başlayıp kullanıcılar hakkında elde edilen birçok bilginin bu sürece dâhil edilmesi ile devam eder. Bu süreçte, etkileşim her zaman açık bir şekilde ifade edilmeyebilir, örneğin doğrudan derecelendirmeler yerine örtük bir etkileşim verilmiş olabilir. Bu yüzden başarılı öneri yaklaşımları detayda gizlenmiş şekilde çalışır.

Öneri sistemleri, birçok farklı kaynaktan gelen birbiriyle ilişkili ya da ilişkisi ortaya çıkarılacak verileri analiz sürecine dâhil ederek kullanıcıların gideceği noktayı önceden tahmin edip bu bilgilerden hareketle kullanıcının ilgileneceği öğeleri tahmin ederek çeşitli yaklaşımlar üretir. Bu süreçte, farklı kaynaklardan elde edilen karmaşık veriden işe yarayacak bilgilerin çıkarılıp çeşitli makine öğrenmesi teknikleriyle işlenmesi gerekir. Günümüzde, e-ticaret sitelerinden birçok farklı çevrimiçi platforma kadar kazancın büyük kısmını karşıladığı bilinmektedir. Bu sistemden elde edilebilecek kazançlar doğrultusunda, kullanıcının bir adım önünde yer alarak onun doğal tercihinin getirdiği noktaymış gibi bu hedeflere yönlendirmeyi amaçlar. Bir şekilde, kullanıcıya dokunmayı başaran bu sistemleri geliştirebilmek için, yapay zekânın pek çok uygulama alanı efektif bir şekilde kullanılır [15].

Örneğin ürün arama motorları, çevrimiçi platformlarda öneri üretme, sağlık uygulamaları, sosyal ağlar ve tedarik zinciri ürün yönetiminde öneri sistemlerinden sıkça yararlanılır. Öneri sistemleri karar verme sürecini hızlandırmak ve kolaylaştırmak için kritik bir rol oynar. Bu sistemler, tercihlere ve satın alma davranışlarına bağlı olarak ürün ve hizmeti sunma konusunda yardımcı olur. Amazon ve eBay gibi e-ticaret sitelerinde ürün önerme, Netflix gibi film sitelerinde film sunma veya Spotify, YouTube gibi dijital müzik kanallarında müzik listesi oluşturma imkânı sunar. Öneriler; kullanıcı-öğe etkileşimlerine,

kullanıcı profillerine ve bazı kullanıcı/öğe yan bilgilerine bağlı olarak oluşturulur. Şekil 2.1'de öneri sistemlerinin sınıflandırma şeması yer almaktadır.



Şekil 2.1 Öneri sistemlerinin sınıflandırma şeması [16]

2.1 Öneri Sistemlerindeki Yaklaşımlar

2.1.1 İşbirliğine Dayalı Filtreleme

İşbirliğine dayalı filtreleme (collaborative filtering), Breese ve arkadaşlarının çalışmasına dayanır [17]. Öneri sistemleri arasında en popüler ve yaygın olarak kullanılan bir yaklaşım olarak kabul edilir [18]. Birden çok kullanıcı tarafından sağlanan derecelendirmelerin işbirlikçi gücünden yararlanır. Çoğu kullanıcı mevcut öğelerin yalnızca bir bölümüyle etkileşime geçtiğinden derecelendirmelerin büyük bir bölümü bilinmez. Bunu çözmek için farklı kullanıcılar tarafından sağlanan derecelendirmelere dayanarak bilinmeyen derecelendirmeleri tahmin eder. Benzer kullanıcıların benzer öğeleri tercih etmesi

veya bir kullanıcının benzer öğeler için benzer tercihleri ifade etmesi gerçeğine dayanır [19].

Gözlemlenen derecelendirmeler genellikle çeşitli kullanıcı ve öğelerle yüksek oranda ilişki içerisindedir. Örneğin, zevkleri çok benzer olan iki kullanıcı olsun. İki kullanıcının da genel anlamda derecelendirmeleri çok benzerse yalnızca bir kullanıcının derecelendirdiği bir öğeyi diğer kullanıcının da aynı şekilde değerlendirmesi yüksek bir ihtimâldir. Bu ihtimâlden yararlanarak eksik gözlemler hakkında çıkarım yapmak mümkündür. Bu çıkarımı da öğeler ya da kullanıcılar arasındaki ilişkiden yararlanarak yapar. *Bellek tabanlı* ve *model tabanlı* yöntemler yaygın olarak kullanılır.

2.1.1.1 Bellek Tabanlı Yöntemler

Bellek tabanlı (memory-based) yöntemler, *sezgisel (heuristic) veya komşuluk tabanlı (neighborhood-based) yaklaşım* olarak da adlandırılır. Sade, doğru, verimli ve kişiselleştirilmiş öneriler sunması sebebiyle büyük ilgi görür. Benzer kullanıcıların benzer öğeleri tercih etmesi, benzer öğelerin de benzer kullanıcılar tarafından tercih edilmesi ilkesine dayanır [20]. Bu sebeple, öğeler ve kullanıcılar arasındaki ilişkilere odaklanır. İki ana eksende değerlendirilir:

- **Kullanıcı tabanlı *(user-based):*** Bir öğe için kullanıcının puanı, benzer kullanıcıların puanlarına göre tahmin edilir. Bir kullanıcıya öneride bulunmak isteniyorsa bu kullanıcıya benzer tüm kullanıcılar belirlenir ve benzer kullanıcı gruplarının ağırlıklı ortalaması veya basit bir ağırlıklı ortalama hesaplanarak öğenin kullanıcı tarafından ne kadar beğenileceği tahmin edilir. En yüksek puana sahip öğeler kullanıcıya önerilir [21].
- **Öğe tabanlı *(item-based):*** Bir öğe için kullanıcının puanı, sistemdeki benzer öğelere verdiği puanlara göre tahmin edilir. En yüksek puana sahip öğeler kullanıcıya önerilir. Derecelendirme matris boyutundan bağımsız, öğe tabanlı metodun kullanıcı tabanlı metoda göre daha hızlı, karşılaştırılabilir, gerçek zamanlı öneri verebilen ve daha iyi kalitede öneriler sağladığı saptanmıştır [21].

Bellek tabanlı yöntemlerin uygulanması kolay ve basittir ancak sonraki bölümlerde bahsedilecek olan seyrek derecelendirme matrisleriyle iyi bir performans göstermezler.

2.1.1.2 Model Tabanlı Yöntemler

Model tabanlı (model-based) yöntemler*,* kullanıcı ve öğelerin düşük boyutlu gizli uzayda temsil edilebileceği varsayımına dayanır. Bu varsayımı sağlayan bir model oluşturmak için kullanıcıların tüm derecelendirmelerini kullanır. Bu derecelendirmelerle eğitilen model kullanılarak gözlemlenmeyen öğelerin derecelendirmelerini tahmin edip hedef kullanıcıya öneri sağlamak için olasılıksal bir yaklaşımdan yararlanır. En yakın komşuları bulmak için tüm kullanıcı ve öğelerin gizli ilişkilerini öğrenir. Bu sebeple, bellek tabanlı yaklaşımlar kadar performans gerektirmez.

Model süreçleri zor ve maliyetli olmasına rağmen yanıt süreleri hızlıdır. Modele karar verilse bile kullanıcıların yeni derecelendirmeleri gelmeye ya da ürün veya kullanıcılar güncellenmeye devam eder, bu da kullanıcı-öğe matrisinin güncellenmesi demektir. Bu sebeple, güncel veriler üzerinden belirli aralıklarla yeni bir model eğitilerek modelin güncel kalması sağlanır. Modelin çalışabildiği optimal parametreler elde edildiği sürece açık ya da örtük geri bildirim verilerini kullanabilir. Karar ağaçları, SVD, MF, Bayes modelleri, kural tabanlı modeller, kümeleme modelleri ve gizli faktör modelleri gibi bazı türleri vardır [22, 23].

2.1.2 İçeriğe Dayalı Filtreleme

İçeriğe dayalı filtreleme (content-based filtering), içeriklerin açıklamasını ve öğelerin tanımlayıcı özelliklerini kullanan sistemlerdir. Kullanıcılar tarafından derecelendirilen öğelere benzer öğeleri bulduklarından ve sisteme yeni öğeler eklenmesi durumunda bu öğelerin herhangi bir derecelendirme geçmişi bulunmadığından diğer yaklaşımlara göre avantajlıdır ancak öneriler, öğelerin içeriğine ve anahtar kelimelere bağlı olduğundan belirli türdeki öğeleri tahmin etme konusunda modelleri kısıtlar [19].

Bir veri kümesindeki öğelerle ilişkili içerik veya özellikler arasındaki benzerliği hesaplar. Örneğin, bir kullanıcı aksiyon türünde bir film ile etkileşime geçtiyse bu yaklaşımı kullanan sistem aksiyon türündeki diğer filmleri önerir. Ayrıca,

kullanıcının geçmişte etkileşime geçtiği filmlerin tür, oyuncu ve yönetmenleri birbiriyle karşılaştırılarak aralarındaki ortak noktalar yakalanabilir. Son olarak geçmişteki etkileşimlerle benzerliği yüksek olan öğeler kullanıcıya önerilir [24].

Yine bu sistemlerde, derecelendirmelerle etiketlenen anahtar sözcükler kullanıcıya özel regresyon veya sınıflandırma problemi oluşturmak amacıyla karşımıza çıkar. Bağımsız değişkenler öğe açıklamalarına, bağımlı değişken ise derecelendirmelere veya etkileşim davranışlarına karşılık gelir. Bu sayede ilgili kullanıcının derecelendirmesinin bilinmediği bir öğeyle etkileşime geçip geçmeyeceği belirlenmeye çalışılır.

Bu sistemler de her sistemde olduğu gibi bazı dezavantajlara sahiptir. Örneğin bir kullanıcı belirli anahtar kelimelere sahip öğeler ile etkileşime asla geçmediyse böyle bir öğenin kullanıcıya önerilmesi mümkün değildir. Bu da kullanıcıya önerilen öğelerin çeşitliliğinin azalmasına yol açar. Sebebi, modelin kullanıcıya özel olması ve işbirlikçi yaklaşımların aksine diğer kullanıcıların topluluk bilgilerinden yararlanmamasıdır. Yeni öğeler için öneriler sağlayabilse de soğuk başlangıç problemini çözme konusunda avantajlı bir konumda olmaması ise diğer bir problemdir. Kullanıcıların derecelendirme geçmişini kullanmadığından kaynaklanır. Genel olarak çok sayıda derecelendirme verisine sahip olmak öneri sistemlerinin performansını etkileyen önemli bir durumdur [25].

2.1.3 Hibrit Sistemler

Hibrit sistemler (hybrid systems), farklı yaklaşımların ortak gücünden yararlanmak için birden fazla tekniği bir araya getiren sistemlerdir. *Sağlam (robust)* modeller oluşturmak, yüksek performans elde etmek ve bireysel yaklaşımların dezavantajlarını azaltmak için genellikle işbirliğine ve içeriğe dayalı yöntemleri birleştirir. Örneğin, işbirliğine dayalı yöntemler yeni öğe sorunlarından muzdariptir, yani derecelendirmesi olmayan öğeleri öneremezler. Bu sorun içeriğe dayalı yaklaşımlarını sınırlamaz çünkü öneriler öğelerin açıklamalarına, isimlerine vb. dayanır. İşbirliğine dayalı yöntemler, çok fazla geri bildirime ihtiyaç duyarken içerik tabanlı filtreleme öğelerin iyi tanımlarına ihtiyaç duyar. Derin bir hibrit sistem örneğinde; kullanıcıların demografik özellikleri (yaş, cinsiyet, meslek, posta kodu vb.) veya öğelerin özelliklerinden yararlanılır. MF ve derin sinir

ağlarının pek çok kullanım alanı olmasına rağmen daha ileri bir teknik olan *gömme (embedding)* katmanlarını kullanır [26]. Kullanıcı ve öğeler kimlikleri üzerinden birleştirilerek MF tekniğinin doğrusal; derin sinir ağlarının ise doğrusal olmama gücünü ortak bir katmanda birleştirir. Bu tarz hibrit yöntemlerin, saf yaklaşımlardan daha isabetli öneriler sunabileceği bilinmektedir [27]. Burke [28], hibrit yöntemleri; *weighted*, *switching, mixed, feature combination, cascade, feature augmentation* ve *meta-level* olmak üzere yedi farklı tipte sınıflandırmıştır.

2.1.4 Bilgiye Dayalı Sistemler

Bilgiye dayalı sistemler (knowledge-based systems), sıkça satın alınmayan öğeler için kullanılan sistemlerdir. Geçmişteki etkileşimlere bakılarak bu ürünleri önermek neredeyse imkânsızdır, çünkü yok denecek kadar az sayıda değerlendirme mevcuttur. Örneğin, bilgisayar ya da gayrimenkul genellikle sıkça alınmayan öğelerdir.

Mevcut kullanıcılara öneriler sunmak için bu öğelerin etkileşim geçmişi de mevcut değildir. Bu bilgileri kullanıcıdan almak da her zaman mümkün değildir. Bu tip durumlarda ürünün bazı özelliklerini kullanıcıdan alarak bu koşullar altında öneriler sunulur. Bir ev örneğinde ise kullanıcıya; evin konumu, oda sayısı, kat sayısı, garaj tipi gibi bilgiler sorulup öneriler sunulabilir. Tıpkı, veri tabanından bazı öğeleri filtrelemek gibi düşünülebilir.

2.2 Öneri Sistemlerinde Karşılaşılan Problemler

Veri seyrekliği, soğuk başlangıç ve ölçeklenebilirlik (scalability) gibi sistemin ana performansıyla ilgili ana zorluklar dışında; *synonymy, gray ship, black ship, shilling attacks, the early rater ve privacy protection* gibi bazı problemler de mevcuttur [30]. Burada; soğuk başlangıç, veri seyrekliği ve ölçeklenebilirlik problemleri üzerinde durulacaktır.

2.2.1 Veri Seyrekliği

Seyrek matris, çoğunlukla sıfır değerlerine sahip matristir. Bu sebeple, katsayıları da sıfıra eşit olur. Bir matrisin ne kadar seyrek bir matris olduğu hesaplanabilir. Bir matrisin seyrekliği basit bir mantıkla, matristeki sıfır olan öğe sayısının toplam öğe sayısına bölünmesi ya da derecelendirilen öğelerin sayısının matristeki toplam

öğe sayısına bölünüp 1 sayısından çıkarılmasıdır. Bunun tam tersi bir matrisin yoğunluğu ise seyreklik oranının 1 sayısından çıkarılmasıdır. Seyreklik ve yoğunluk formülleri sırası ile eşitlik (2.1) ve (2.2)'de verilmiştir:

$$seyreklik = 1 - \frac{(derecelendirilen\ öğelerin\ sayısı)}{(kullanıcı\ sayısı\ x\ öğe\ sayısı)} \tag{2.1}$$

$$yoğunluk = 1 - seyreklik \tag{2.2}$$

Derin öğrenme modellerindeki yoğun temsiller oldukça bağımlı ve karışıktır. Bu yüzden, girdilerde meydana gelen değişiklikler temsil vektörlerinde büyük oranda değişikliğe sebep olurlar. Seyrek gösterim; ağın sıfır olmayan gizli düğümleriyle, girdide meydana gelen değişiklikleri koruyarak daha sağlam yapılar oluşturulmasını sağlar. Böylece, ağın değişken boyutlu aktif düğümleri kullanmasını sağlar [31]. Sonuç olarak daha fazla düğüm kullanılmasını sağlayan seyrek bir temsilin otokodlayıcı için elverişli bir özellik olduğu ve bu sebeple, farklı özelliklerin çıkarılmasına ve öğrenilmesine katkı sağladığı anlaşılır. Tüm bunların yanında seyrekliği çok fazla artırmanın (eşit sayıda düğüm için modelin sınırlarını zorlayıp etkin kapasitesini azalttığından) tahminlerin performansını olumsuz etkileyeceğinden bahsedilir [32]. Çok büyük matrisler bellekte çok fazla yer kaplar ve pratikte, bu matrisler seyrektir. Sıfır değerleri herhangi bir bilgiye de sahip olmadığından belleğin boş yere israf edilmesine yol açar. Buna ek olarak seyrek matrisler üzerinde çalışmak uzun bir zaman alabilir ve matrislerin boyutu büyüdükçe işlemlerin süresi artabilir [33].

2.2.2 Soğuk Başlangıç Problemi

Öneri sistemlerinde karşılaşılan büyük problemlerden biridir. Hakkında herhangi bir bilgi bulunmayan kullanıcı kişiselleştirilemez. Bu da yeni kullanıcılara sunulan öneri isabetinin optimum olmamasına ve kullanıcı memnuniyetsizliğine yol açar. Büyük ölçüde kullanıcılardan gelen girdilere bağlı olduklarından işbirlikçi filtreleme önerilerinde sıkça karşılaşılır. Bu problem, yeni kullanıcılara öneriler sunmanın yanı sıra sisteme yeni öğeler eklendiğinde de ortaya çıkar. Yeni eklenen öğelerin etkileşim sayısı fazla olmadığından sistem bunları diğer öğelerle nasıl ilişkilendireceğini bilemez. Bu sebeple, yeni öğeler önerilerde görünmez.

Bazı kullanıcıların etkileşime girdiği öğelerle etkileşime giren kullanıcı sayısı yok denecek kadar az olduğunda da soğuk başlangıç problemine benzer olan *erken değerlendirme (early rater)* sorunu ortaya çıkar. Kullanıcılara e-posta göndermek, yeni öğeleri gösteren kısımları sisteme eklemek ya da yeni ürünleri *öne çıkarmak (boost)* bu sorunun çözümlerinden bazılarıdır.

Bir kullanıcıya öneri yapılabilmesi için kullanıcının yaygın olarak en az 20 ile 50 öğeyle etkileşime girmesi beklenir fakat pek çok kullanıcı önerilerin en kısa zamanda yapılmasını ister. İstisnai bazı durumlar (örneğin bir kullanıcı, herhangi bir öğeyi sistemde aratırsa) o kullanıcının tercihini anlamanın isabetli yolu olabilir. Böylece, kullanıcının ilgilendiği bir şeyler olduğu fikri anlam kazanır [34].

2.2.3 Ölçeklenebilirlik

Birçok sistemde, yaygın olarak kullanılan işbirliğine dayalı algoritmalar benzer komşular bulmak için zamana fazlasıyla ihtiyaç duyar. Kullanıcı ve öğe sayısı artmaya başladığında da ciddi bir ölçeklenebilirlik problemi yaşar. Örneğin, milyonlarca kullanıcı ve öğeye sahip sistemlerin çevrimiçi etkileşim ve gereksinimlere gerçek zamanlı dönüt vermesi, kullanıcılar için önerilerde bulunması gerekir. Bu da sistemin yüksek kapasitede ölçeklenebilir olmasını gerektirir [30].

Boyut azaltma teknikleri ölçeklenebilirlik sorununu hızlı bir şekilde çözer ancak maliyeti yüksek olan MF gibi bazı hesaplamalara ihtiyaç duyar. Mevcut kullanıcıları kullanarak bazı hesaplamalar yapar. Derecelendirilecek bir durum söz konusu olduğunda ise yeniden maliyetli bir hesaplama yapmaz. Önceki hesaplamalara dayanarak çıktıları hızlı bir şekilde üretir. Böylece öneri sistemini son derece ölçeklenebilir bir hâle getirir [35].

2.3 Geri Bildirimler

2.3.1 Geri Bildirim Yaklaşımları

Kullanıcı tercihlerine karşılık gelen kişiselleştirilmiş önerilerde bulunmak için öneri sistemleri, kullanıcılar tarafından derecelendiren öğelerin verilerini toplar ve verilerin nasıl toplandığına bağlı olarak kullanıcı profilleri oluşturur [16].

Uygulamalardaki öneri sistemlerinin görevi genellikle, kullanıcıları geçmiş tercihlerine ve mevcut bağlamsal durumlarına göre sıralayarak veya filtreleyerek kullanıcıları ek ilgi alanlarına yönlendirmektir [36].

Geri bildirimler üç yaklaşımda incelenebilir:

- Kullanıcı geri bildirimine dayalı açık yaklaşım *(explicit approaches)*,
- Kullanıcı davranışını kaydetmeye dayalı *örtük yaklaşım (implicit approaches)*,
- Örtük ve açık yaklaşımın birleşimi *karıştırma yaklaşımı (mixing approaches)*,

Burada, açık ve örtük yaklaşımlar üzerinde durulacaktır. Karıştırma yaklaşımında, açık ve örtük yaklaşımların birleştirilmesi söz konusudur [16].

2.3.1.1 Açık Geri Bildirim

Literatürdeki en belirgin *açık geri bildirim (explicit feedback)* biçimi genellikle *yıldız* olarak belirtilen 1'den 5'e kadar bir ölçekte, her bir öğeye kullanıcı tarafından sağlanan derecelendirmedir. Genellikle, bir öğeye verilen düşük yıldız derecelendirmesi öğenin düşük bir tahminine karşılık gelirken yüksek yıldız derecelendirmesi öğeye yüksek bir ilgi gösterilmesine karşılık gelir. Bu yöntem kullanıcı için ek çaba gerektirir, bu nedenle kullanıcılar, öğelere not vermekten genellikle kaçınır [16]. Diğer yaygın biçimler, tekli *beğenme* ifadelerinin yanı sıra ikili *beğen/beğenme* seçimleridir. Belirli uygulamalarda, bir etkileşimi *gizlemek* gibi açık olumsuz kullanıcı eylemleri de vardır. Bu tür sinyaller tekli veya ikili olmasına rağmen örtük geri bildirim değildir.

Doğrudan işlenebilen kullanıcı tercihlerinin yanı sıra, açık geri bildirimin başka biçimleri de vardır ancak bunlar daha fazla analiz gerektirir veya uygulamaya özeldir. Örneğin, olumlu veya olumsuz çağrışımları olan etiketlerle öğelere açıklama eklenerek görüşler ifade edilebilir. Ayrıca, farklı geri bildirimler arasındaki açık ifadelerden daha fazla örtük bilgi çıkarmak mümkündür. Örneğin, herhangi bir web sayfasıyla ilgili herhangi bir yer imi eylemi, öğe için kalite değerlendirmesi olmadığından hiçbir zaman açık bir geri bildirim sinyali değildir.

Ek olarak açık derecelendirme bilgileri seyrek olabilir, çünkü bu tür derecelendirmeler, kullanıcılar tarafından fazladan çalışma gerektirir. Ayrıca, kullanıcılardan açık bir derecelendirme yapmasını beklemek önemli miktarda çaba gerektirdiğinden kullanıcılar örtülü geri bildirim sağlamayı tercih eder. Örneğin, çevrimiçi bir mağazaya göz atarken kullanıcının odak noktası öğeler arasında dolaşmaktır, bu sebeple, öğelerin tümü kullanıcı tarafından tek tek derecelendirilmez [36].

2.3.1.2 Örtük Geri Bildirim

Örtük geri bildirim (implicit feedback) açık geri bildirimden farklı olarak kullanıcıların etkileşime geçtiği öğelerin sadece tek bir geri bildirimle ifade edilmesidir. Oluşturulan derecelendirme matrisi bu yüzden sadece olumlu geri bildirim verilen öğelerle oluşturulur, eksik olan gözlemler ise boş değer olarak atanır [19]. Geri bildirim yapan aktif bir kullanıcı katılımı gerekmediğinden kullanıcı davranışları kaydedilir ve öğelere verdikleri tepkilere odaklanır. Bu yaklaşımın amacı, kullanıcı tepkisine göre bir öğenin uygunluğunu öğrenmektir. Ayrıca kullanıcının göz atma etkinliğini incelemeye ek olarak bir web sayfasında geçirilen süre, tıklanan bağlantılar ve kaydırma etkinliği de gözlenerek kullanıcı eylemleri izlenebilir [16].

Müzik dinleyen veya bir haber makalesini paylaşan bir kullanıcı düşünün. Bu kullanıcının eylemleri, örtük geri bildirim olarak yorumlanabilir, yani kullanıcı müzik veya makale içeriğine yönelik bir tercihe sahip olabilir. Kullanıcının müziğin sanatçısıyla veya haberin konusuyla ilgilendiği, kullanıcı eylemlerinden de çıkarılabilir. Bununla birlikte kullanıcı müziğe açıkça puan verdiyse veya dolaylı olarak haber makalesini beğendiyse kullanıcının sanatçıya veya konuya ilgili olabileceği de çıkarılabilir. Bu nedenle örtük geri bildirim terimi, kullanıcı tercihlerinin dolaylı olarak çıkarılabileceği her türlü etkileşimi içerir [36].

Örneğin, M ve N sırasıyla kullanıcı ve öğe sayısını temsil etsin. Kullanıcı ve öğeler arasındaki etkileşim matrisi $Y \in R^{MXN}$ olarak gösterilsin.

$$y_{ui} = \begin{cases} 1, & kullanıcı\ ve\ \text{öğe}\ arasında\ etkileşim\ var\ ise \\ 0, & diğer\ durumlar \end{cases}$$

y_{ui}'nin değeri 1 ise kullanıcı ile öğe arasında bir etkileşim mevcuttur. Aksi takdirde, kullanıcı ve öğe arasında herhangi bir etkileşim yoktur. Bu etkileşimler kullanıcının öğeyi gerçekten beğenip beğenmediğini belirtmediğinden *gürültü sinyalleri (noise signals)* olasılığı vardır.

Ydeki derecelendirilmeyen girişlerin puanları tahmin edilerek örtük verilerden tahmin önerileri formüle edilir. Model tabanlı teknikler, öğrenmeyi $y_{ui} = f(\mathrm{u}, \mathrm{i}|\Theta)$ şeklinde özetler; burada $\hat{y}_{ui}$, y_{ui} için tahmin edilen puanı temsil eder, Θ model parametrelerini temsil eder ve f, model parametrelerini hesaplanan puana eşleyen bir etkileşim fonksiyonunu temsil eder [19].

2.3.2 Geri Bildirim Yaklaşımlarının Problemleri

Gerçekte, örtük tercih sinyallerinin miktarı çok büyük olabilir. Bir kullanıcının faresinin her hareketi çevrimiçi bir uygulamada izlenebilir. Gelecekte, *Nesnelerin İnterneti (Internet of Things)*'nin sürekli gelişimi ve kullanıcıların mobil veya giyilebilir cihazlar aracılığıyla *her zaman açık* olması, kullanıcıların davranışları ve etkileşimde bulundukları nesneler hakkında daha da büyük miktarda bilgiyi oluşturacaktır. Farklı sinyal türleri, bir tercihin farklı bir gücünü gösterebilir, yani bir satın alma, bir çevrimiçi mağazada bir öğe görüntüleme işleminden daha fazla sayılabilir. Son olarak örtük geri bildirimler genellikle yalnızca olumludur bu yüzden, doğru yorumlamak zordur. Genel olarak, gerçek dünya uygulamalarında örtük geri bildirime dayalı öneri daha çok yaygındır.

Açık derecelendirmelerin genel bir sorunu; kullanıcı derecelendirmelerini, muhtemelen bir öneri senaryosunda daha alakalı olan zevklerini ifade etmekten ziyade bir öğenin kalitesini değerlendirmek için bir araç olarak görmesidir. Bu nedenle; gerçek kullanıcı davranışına dayalı öneriler, gerçekte açık derecelendirmelere dayanan tahminlerden daha güvenilir olabilir.

Açık geri bildirim, bir kullanıcı tarafından kasıtlı ve net bir kalite değerlendirmesi olarak görülebilir. Değerlendirmenin yorumlanması alana ve verildiği bağlama bağlıdır. Öneri sistemleri için, açık geribildirim genellikle bir kullanıcı tarafından belirli bir öğe için atanan sayısal bir derecelendirmedir. Öte yandan örtük geri bildirim, sistemin kasıtlı bir değerlendirmesini amaçlamaz ancak yine de kullanıcının olumlu veya olumsuz görüşünü çıkarmak için kullanılabilen farklı

türde kullanıcı etkileşimlerini içerir. Örneğin, bir kullanıcının bir öğeye veya birden fazla benzer öğeye karşı olumlu veya olumsuz bir tutumu olup olmadığını belirlemek için kullanılabilir. Bununla birlikte, kaydedilen etkileşim sinyallerinin pozitif mi, negatif mi veya ikisi arasında bir yerde mi olduğu evrensel olarak net olmayabileceğinden genellikle yalnızca bir dereceye kadar belirsizlikle yorumlanabilir. Ek olarak örtük bir geri besleme sinyali doğru bir şekilde yorumlanabilse bile nasıl nicelendirileceği, diğer örtülü ve açık geri bildirim türleri ile nasıl birleştirileceği sorusu hâlâ vardır. Örtük geri bildirimlerde; sinyallerin gücünün yorumlanması, şeffaflık, negatif sinyallerin kaybedilmesi, veri bolluğu ve uzun kuyruk problemi gibi problemler mevcuttur [36].

3
OTOKODLAYICILAR

3.1 Basit Otokodlayıcılar

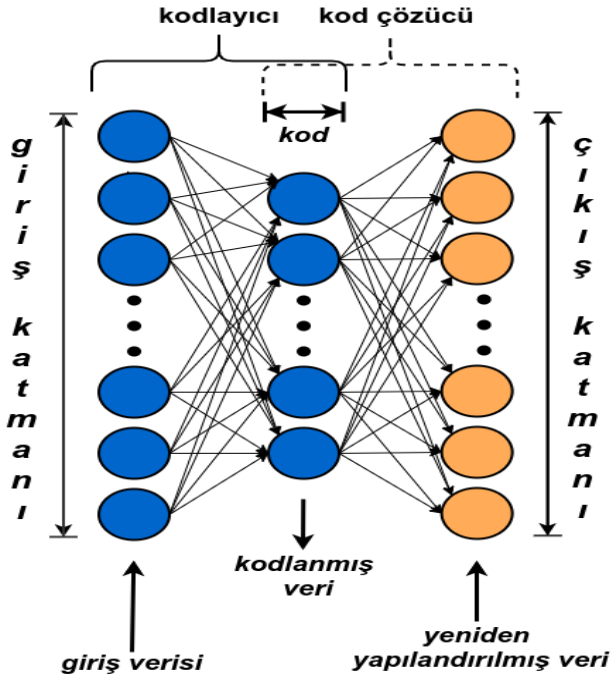
Otokodlayıcılar (autoencoders), çok boyutlu veri setini sıkıştıran sonra da sıkıştırılmış kısmına *yeniden yapılandırma (reconstruction)* uygulayan *denetimsiz (unsupervised)* ve doğrusal olmayan bir yapay sinir ağıdır. Veri setini, en verimli şekilde nasıl sıkıştıracağını ve yeniden yapılandıracağını öğrenerek orijinal girdiyi yeniden oluşturmaya çalışır. Standart otokodlayıcıda, giriş ile çıkışların boyutları birbirine eşittir [37]. Eğitim sırasında, etiketlerini kendi ürettiğinden *öz-denetimli (self-supervised)* bir mimari olarak da bahsedilir. 1986'da yayınlanan bir makalede ortaya atılmıştır [38]. Bugüne kadar araştırma alanları genişlemiş ve bu mimari üzerinde birçok uygulamalar yapılmıştır. Sonuç olarak çıktıların ağırlıklarının hesaplanabildiği bir yapıya dönüşmüştür.

Temel amacı, veri setini dar bir alanda sıkıştırıp boyutunu azaltarak veri setini temsil edebilecek özellikleri çıkarmaktır. Diğer ağlardan farklı olarak sıkıştırılmış katmandaki özellikleri kullanarak girdi verisine olabildiğince yakın bir çıkış verisini yeniden oluşturmaya çalışır. Böylece, en içteki katmanda elde edilen gizli özelliklerle girdi verisinin yeniden yapılandırılabileceği düşüncesi anlam kazanmıştır.

Otokodlayıcı, *kodlayıcı (encoder)* ve *kod çözücü (decoder)* katmanlarından oluşur. Ayrıca, dâhili olarak girdiyi temsil eden bir veya birden fazla *kod (code, bottleneck, latent space)* katmanına ya da *gizli (hidden)* katmana sahiptir [39]. Kodlayıcı kısmında, girişteki her bir öznitelikten alınan veri kullanılarak kod katmanına kadar olan kısımda yeniden kodlanır. Böylece kod katmanı elde edilir. Kod katmanında bulunan *düğüm (nöron)* sayısının kodlayıcı kısımda bulunan gizli katmanlardan genellikle daha az sayıda veya eşit sayıda nörona sahiptir. Kod katmanında, girişten gelen veriler sıkıştırılır. Kod çözücü kısmında ise kod

katmanında sıkıştırılmış verinin çözülerek yeniden yapılandırılmış verinin yani çıkış verisinin elde edilmesi sağlanır.

Gizli katman sayısının bir olduğu otokodlayıcı, *basit otokodlayıcı* olarak adlandırılır. Basit otokodlayıcıda, bir gizli katman ve bir çıkış katmanı olmak üzere toplam 2 katman bulunur. Bu nedenle, basit otokodlayıcı literatürde tek-gizli katmanlı otokodlayıcı ya da 2-katmanlı otokodlayıcı olarak da isimlendirilir. Giriş ve çıkıştaki nöron sayısı eşitken gizli katmanda giriş ve çıkıştan daha az sayıda nörona sahip olması beklenir. Şekil 3.1'de basit otokodlayıcı gösterilmiştir.



Şekil 3.1 Basit otokodlayıcı

Şekil 3.1'de gösterilen basit otokodlayıcıda, 2 temel katman olduğundan bahsedilmişti. $x \in R^D$, D boyutlu giriş verisidir. İlk katman olan kod katmanı $h \in R^H$ ile gösterilir ve genellikle giriş verisinden daha düşük boyuta sahip olması istenir $(H < D)$. İkinci katman olan çıkış katmanı ise $x' \in R^D$ ile gösterilir ve D boyutlu çıktı verisinden oluşur. $f_e: R^D \rightarrow R^H$ kodlayıcı fonksiyonu, x girdi verisini h değişkenlerine eşleştirirken; $f_d: R^H \rightarrow R^D$ kod çözücü fonksiyonu ise h değişkenlerini x' çıktı verisine eşleştirir. Matematiksel olarak bir x girdi verisini yeniden yapılandırılmış x' çıktı verisine eşleştiren basit otokodlayıcının ifadesi izleyen şekildedir:

$$h = f_e(x) = g_e(V_e x + b_e) \tag{3.1}$$

$$x' = f_d(h) = g_d(V_d h + b_d) \tag{3.2}$$

(3.1) ve (3.2) eşitliğine göre, $V_e \in R^{H \times D}$ ve $V_d \in R^{D \times H}$ *ağırlık (weight)* matrisleri; $b_e \in R^H$ ve $b_d \in R^D$ *yan (bias)* terimleri; f_e kodlayıcı, f_d kod çözücü; g_e ve g_d ise aktivasyon fonksiyonudur.

Otokodlayıcının amacı, kodlayıcı ve kod çözücü fonksiyonları yardımı ile

$$f_d\big(f_e(x)\big) = g_d(V_d g_e(V_e x + b_e) + b_d) \tag{3.3}$$

dönüşümlerini uygulayarak x ile x'arasındaki yeniden yapılandırma hatasını minimize edecek şekilde verilerin D boyutlu gösterimini elde etmektir [40].

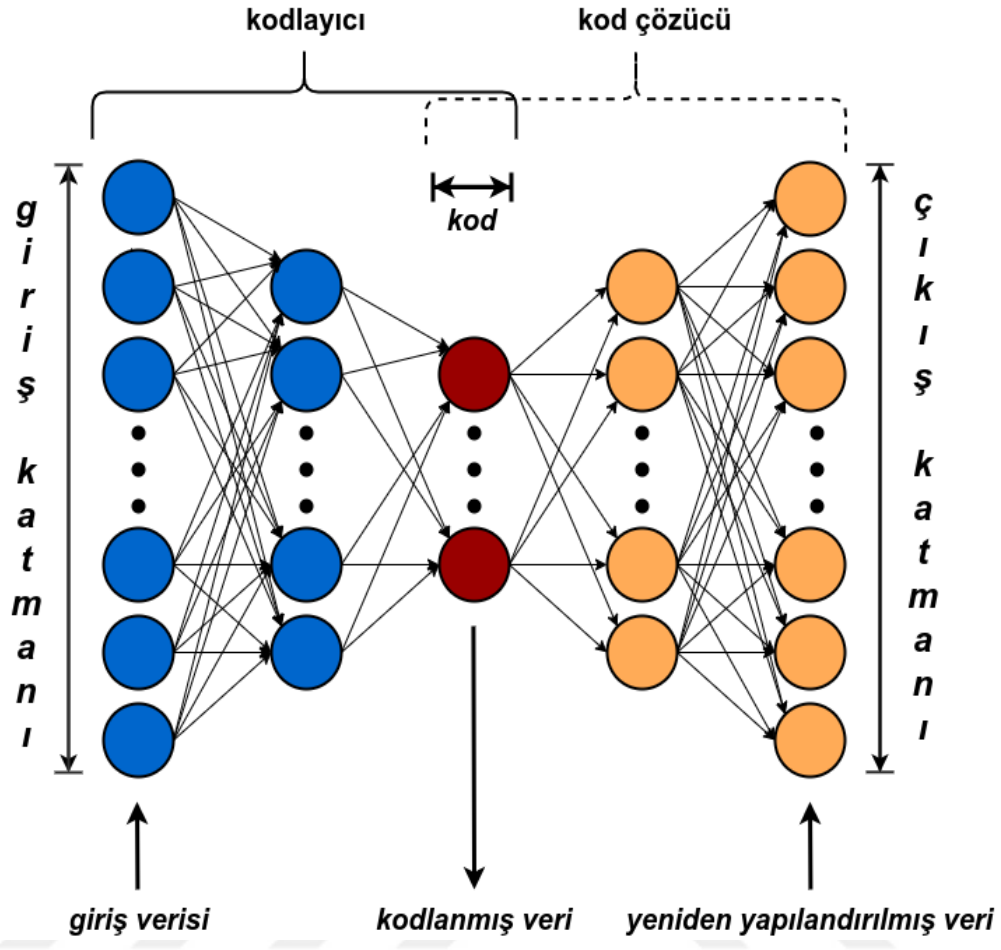
Kayıp (loss) fonksiyonu,

$$L(x, x') = \|x - x'\|^2 = \big\|x - f_d\big(f_e(x)\big)\big\|^2 \tag{3.4}$$

olarak ifade edilir. $\mathscr{L}$, girdinin tipine göre örneğin *hata kareler ortalaması (MSE)* ya da *çapraz entropi (CE)* olabilir.

3.2 Derin Otokodlayıcılar

Basit otokodlayıcılardan farklı olarak gizli katman sayısı birden fazla olan çok katmanlı sinir ağları, *derin otokodlayıcı (deep autoencoder)* olarak adlandırılır. Derin otokodlayıcı, bir dizi simetrik kodlayıcı ve kod çözücü katmanlarından oluştuğundan toplam L katmana sahiptir. Alt katman eğitildikten sonra elde edilen kod, girdi olarak bu katmanın kodlamasını kullanan başka bir katmana verilir. Kod çözücü katmanları girdi verisini değil, kod katmanlarının akışındaki simetrik sırasını koruyarak önceki katmandan aldığı kodu boyutları da simetrik olacak şekilde yeniden oluşturmaya çalışır. Bu sebeple son katman olan çıkış katmanı, kod çözücü katmanlarının bir *yığını (stack)* olarak nitelendirilir. Şekil 3.2'de derin otokodlayıcı yapısı gösterilmiştir.



Şekil 3.2 Derin otokodlayıcı

Derin otokodlayıcıda, l tane kodlayıcı ve l tane kod çözücü olmak üzere toplam $2l = L$ katman vardır. Genel olarak L katmana sahip bir derin otokodlayıcı, $\Theta^i = \{V_e^i, V_d^i, b_e^i, b_d^i\}$ olmak üzere, $\Theta = \{\Theta^i | i \in \{1, \dots, l\}\}$ parametrelerine sahiptir. Bir x girdi verisini yeniden yapılandırılmış x' çıktı verisine eşleştiren derin otokodlayıcının matematiksel ifadesi izleyen şekildedir:

$$e_0 = x \tag{3.5}$$

$$e_i = f_e^i(e_{i-1}) = g_e\left(V_e^i e_{i-1} + b_e^i\right) \tag{3.6}$$

$$d_{l+1} = e_l = h \tag{3.7}$$

$$d_i = f_d^i(d_{i+1}) = g_d\left(V_d^i d_{i+1} + b_d^i\right) \tag{3.8}$$

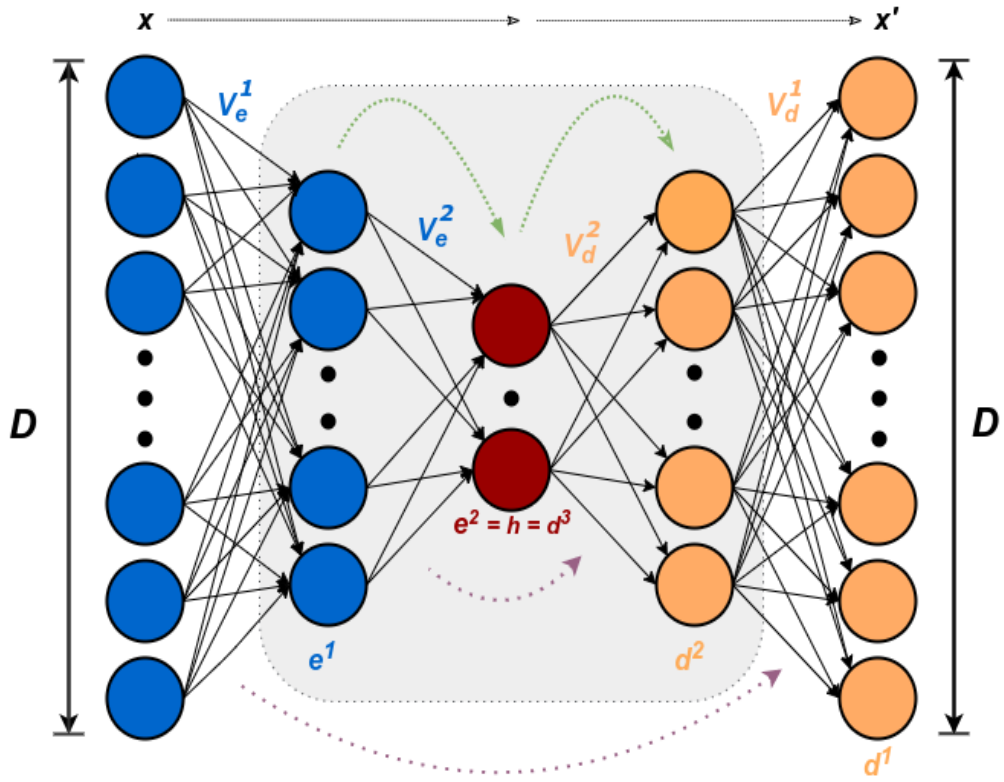
$$x' = d_1 \tag{3.9}$$

(3.5) – (3.9) eşitliğine göre, V_e ve V_d ağırlık matrisleri; $b_e$ve b_d yan terimleri; e_i kodlayıcı, h kod, d_i kod çözücü katmanlar; f_e kodlayıcı, f_d kod çözücü fonksiyonlar; g_e ve g_d ise aktivasyon fonksiyonudur. Kayıp fonksiyonu,

$$L(x, x') = \|x - x'\|^2 = \left\|x - f_d^1(f_d^2 \dots (f_d^l(h))\right\|^2 \tag{3.10}$$

olarak ifade edilir. $\mathscr{L}$, girdinin tipine göre örneğin *MSE* ya da *CE* olabilir [41].

Örneğin; 4 katmanlı derin otokodlayıcı modeli ele alınsın. e_1 kodlayıcı, e_2 kod, d_2 kod çözücü, d_1 çıkış katmanını oluşturur ve akış şeması Şekil 3.3'te gösterilmektedir.



Şekil 3.3 4 katmanlı derin otokodlayıcı akış şeması

Şekil 3.3'te verilen modelin matematiksel yapısı izleyen şekildedir:

$$e_0 = x \tag{3.11}$$

$$e_1 = f_e^1(e_0) = g_e(V_e^1 x + b_e^1) \tag{3.12}$$

$$e_2 = f_e^1(e_1) = g_e(V_e^2 e_1 + b_e^2) \tag{3.13}$$

$$d_3 = e_2 \tag{3.14}$$

$$d_2 = f_d^2(d_3) = g_d(V_d^2 e_2 + b_d^2) \tag{3.15}$$

$$d_1 = f_d^1(d_2) = g_d(V_d^1 d_2 + b_d^1) \tag{3.16}$$

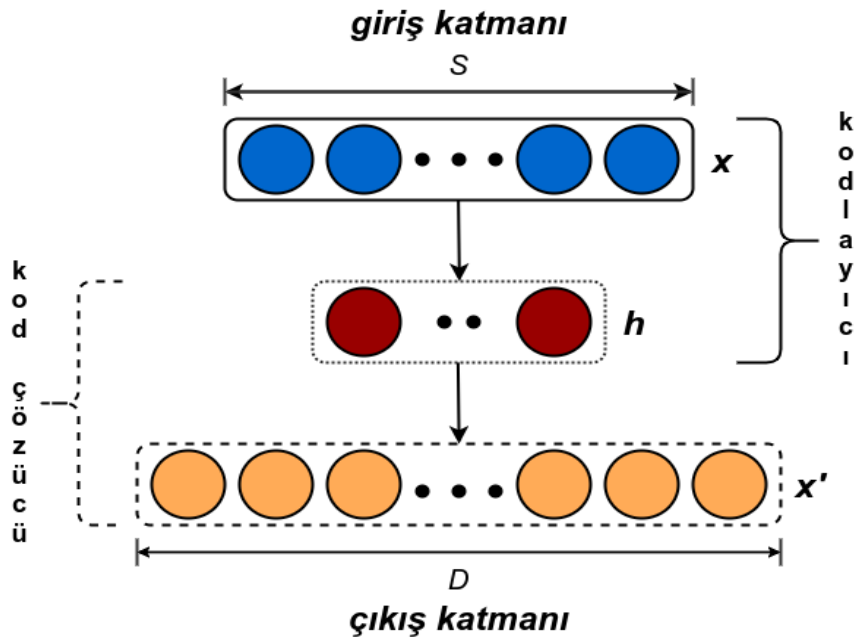
$$x' = d_1 \tag{3.17}$$

Gizli katman sayısı birden fazla olan ve lineer olmayan otokodlayıcılarda, ağırlıkları optimize etmek zordur. Eğer kod çözücü katmanı kodlayıcı katmanının aynası görevini üstlenecek şekilde mimariye sahip ise herhangi bir l katmanındaki kod çözücü ağırlık matrisi, l katmanındaki kodlayıcı ağırlık matrisinin transpozesine eşit alınabilir: $V_d^l = (v_e^l)^T$. Bu türdeki derin otokodlayıcılar, *bağlı (tied)* ya da *kısıtlamalı* (*constrained*) *otokodlayıcı* olarak adlandırılır ve kısıtlamasız türdeki otokodlayıcı modeline göre 2 kat daha az serbest parametre içerir [12, 42].

3.3 Basit Yarı-Otokodlayıcılar

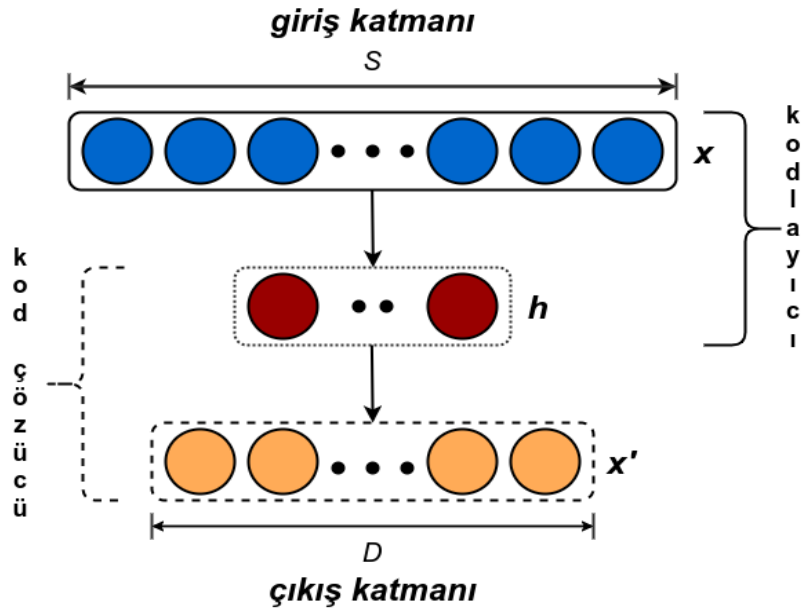
Zhang ve arkadaşları [10] tarafından yapılan çalışmada; otokodlayıcıda giriş ve çıkış boyutlarının aynı olmasının zorunlu olmadığı, bu katmanların boyutlarının farklı olabileceği ve bu durumun geleneksel yöntemlere göre bazı avantajlar getireceği gözlenmiştir. Girdilerden farklı alt kümeleri örnekleyerek farklı temsilleri esnek bir şekilde yakalayabilme ve giriş katmanına ek bilgiler eklemeye uygun olması sahip olduğu avantajlardır. Zhang ve arkadaşları, girdi ve çıktıların boyut sınırlamasını kırarak iki çeşit otokodlayıcı tasarlamıştır: Çıkış katmanının boyutu girdi verisinin boyutundan büyük olduğu ilk durumda, gizli katmanın bazı yeni öğeler de üreterek girdiyi yeniden yapılandırması sağlanır. Ağ iyi bir şekilde eğitilse de üretilen yeni öğeler için yorum yapmak zordur. Çıkış katmanı boyutunun girdi verisinin boyutundan küçük olduğu ikinci durumda ise çıktı katmanı girdilerin sadece bir kısmını yeniden yapılandırır. Girdinin yeniden yapılandırılmayan kısmı, daha iyi temsiller yaratmayı sağlar, kod katmanını güçlendirir veya öğrenmeyi kolaylaştırır. Böylece girdiden daha fazla bilgi sentezlenerek çıktıyı daha düzgün yapılandırması beklenir. Bu yapı, *yarı-otokodlayıcı (semi-autoencoder)* terimiyle ifade edilmiştir.

Gizli katman sayısı 1 olan yarı-otokodlayıcı, *basit yarı-otokodlayıcı* olarak adlandırılır. Şekil 3.4'te çıkış katmanı boyutu girdi verisinin boyutundan büyük olan basit otokodlayıcı gösterilmiştir.



Şekil 3.4 Çıkış katman boyutu girdi boyutundan büyük olan basit otokodlayıcı

Şekil 3.5'te ise çıkış katmanı boyutu girdi verisinin boyutundan küçük olan yani kısaca basit yarı-otokodlayıcı ile adlandırılan yapı gösterilmiştir.



Şekil 3.5 Basit yarı-otokodlayıcı

Tezin bu bölümünde, Şekil 3.5'te verilen basit yarı-otokodlayıcı yapısı ile ilgilenilecektir.

Basit yarı-otokodlayıcıda 2 katman mevcuttur. $x \in R^S$, S boyutlu girdi verisidir. İlk katman H boyutlu kod katmanı olup $h \in R^H$ ve ikinci katman D boyutlu çıkış katmanı olup $x' \in R^D$ ile gösterilir. Aralarında boyut olarak $H < D < S$ ilişkisinin mevcut olması istenir. $f_e: R^S \rightarrow R^H$ kodlayıcı fonksiyonu, x girdi verisini h değişkenlerine eşleştirirken; $f_d: R^H \rightarrow R^D$ kod çözücü fonksiyonu ise h değişkenlerini x' çıktı verisine eşleştirir. Basit yarı-otokodlayıcının matematiksel yapısı izleyen şekildedir:

$$h = f_e(x) = g_e(W_e x + b_e) \tag{3.18}$$

$$x' = f_d(h) = g_d(W_d h + b_d) \tag{3.19}$$

(3.18) ve (3.19) eşitliğine göre, $W_e \in R^{H \times S}$ ve $W_d \in R^{DXH}$ ağırlık matrisleri; $b_e \in R^H$ ve $b_d \in R^D$ yan terimleri; f_e kodlayıcı, f_d kod çözücü; g_e ve g_d ise aktivasyon fonksiyonudur.

Basit yarı-otokodlayıcıyı eğitmek için x girdisinden, çıkış boyutunda bir x' alt kümesi elde edilip $sub(x)$ ile gösterilir. Basit yarı-otokodlayıcı geleneksel basit otokodlayıcıdan farklı olmak üzere, x girdisiyle x' çıktısının hatasını hesaplamak yerine x girdisinden elde ettiği $sub(x)$ ile x'çıktısının hatasını hesaplar:

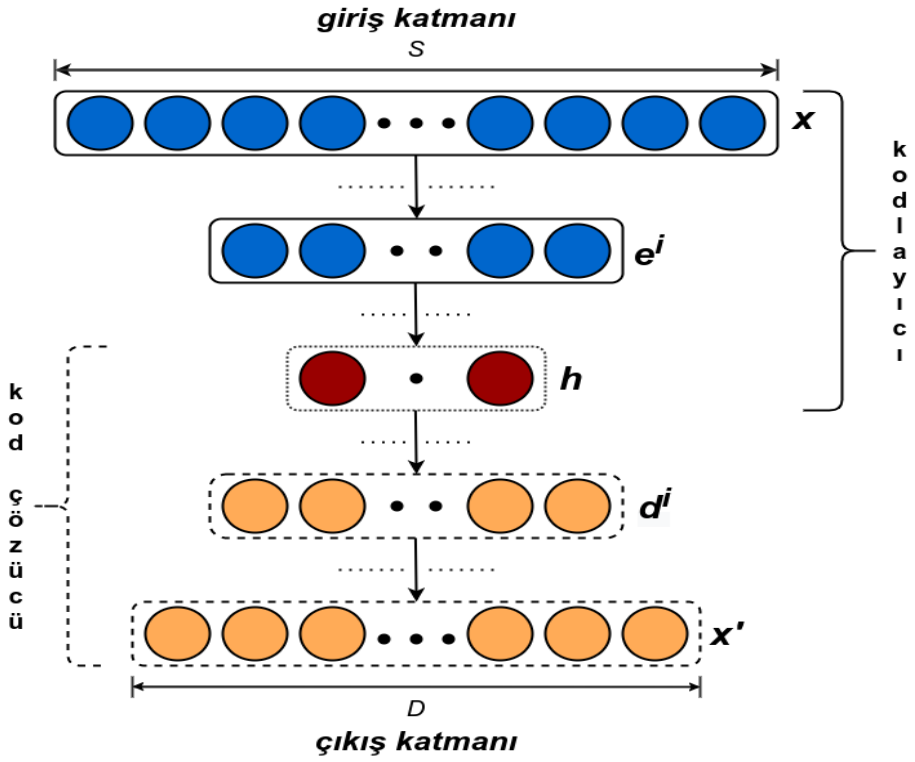
$$L(x, x') = \| sub(x) - x' \|^2 = \left\| sub(x) - f_d\left(f_e(x)\right) \right\|^2 \tag{3.20}$$

(3.20) eşitliğinde verilen kayıp fonksiyonu hesaplanırken basit yarı-otokodlayıcı tüm girdiyi yeniden yapılandırmayı öğrenmek yerine $sub(x)$ ile gösterilen alt kümeyi yeniden yapılandırmayı öğrenir [10].

3.4 Derin Yarı-Otokodlayıcılar

Tezin bu bölümünde, Zhang ve arkadaşları [10] tarafından önerilen basit yarı-otokodlayıcı yapısı geliştirilerek derin yarı-otokodlayıcılar tanıtılacaktır. Basit yarı-otokodlayıcılardan farklı olarak gizli katman sayısı birden fazla olan çok katmanlı sinir ağları *derin yarı-otokodlayıcı (deep semi-autoencoder)* olarak adlandırılır. Derin yarı-otokodlayıcı, çıkış katmanı boyutunun girdi verisinin boyutundan küçük olduğu ve bir dizi simetrik kodlayıcı ve kod çözücü katmanlarından oluşan

otokodlayıcıdır. Derin yarı-otokodlayıcı girdinin hepsini yapılandırmak yerine çıktı katmanı ile aynı boyutta olacak şekilde girdiden bir alt küme elde ederek alt kümenin yeniden yapılandırılmasını öğrenir. Girdinin yapılandırılmayan kısmı ise kodlayıcı katmanlar arasında kullanılarak kod katmanına iletilirken ileri beslemeli ağın daha kolay ve hızlı öğrenmesini sağlar. Şekil 3.6'da derin yarı-otokodlayıcı verilmiştir.



Şekil 3.6 Derin yarı-otokodlayıcı

x girdi verisini yeniden yapılandırılmış x' çıktı verisine eşleştiren derin yarı-otokodlayıcının matematiksel ifadesi izleyen şekildedir:

$$e_0 = x \tag{3.21}$$

$$e_i = f_e^i(e_{i-1}) = g_e\left(W_e^i e_{i-1} + b_e^i\right) \tag{3.22}$$

$$d_{l+1} = e_l = h \tag{3.23}$$

$$d_i = f_d^i(d_{i+1}) = g_d\left(W_d^i d_{i+1} + b_d^i\right) \tag{3.24}$$

$$x' = d_1 \tag{3.25}$$

(3.21) – (3.25) eşitliğine göre, W_e ve W_d ağırlık matrisleri; b_e ve b_d yan terimleri; e_i kodlayıcı, h kod, d_i kod çözücü katmanlar; f_e kodlayıcı, f_d kod çözücü fonksiyonlar; g_e ve g_d ise aktivasyon fonksiyonudur.

Derin yarı-otokodlayıcı, çıktı verisi ile aynı boyutta olacak şekilde girdinin bir alt kümesi olan ve $sub(x)$ ile gösterilen alt kümeyi yeniden yapılandırmayı öğrenir. Kayıp fonksiyonu,

$$\mathcal{L}(x, x') = \|sub(x) - x'\|^2 = \left\|sub(x) - f_d^1(f_d^2 \dots (f_d^l(h))\right\|^2 \tag{3.26}$$

olarak ifade edilir.

3.5 Otokodlayıcıların Kullanım Alanları

Makine öğrenmesinde model aşamasına geçmeden önce en çok başvurulan yöntem olan *temel bileşenler analizi (PCA)* yöntemiyle, öznitelikler arasında doğrusal dönüşümler yapılarak özniteliklerin sayısı azaltılır. Bu boyut azaltma işlemi doğrusal olmayan dönüşümler yapılarak otokodlayıcı ile de uygulanabilir.

Sınıflandırma modellerinde ise kendi kendine öğrenme amacıyla kullanılır. Etiket sayısının az olduğu durumlarda örneğin kredi kartı sahtekârlıklarının tespit edilmesi konusunda uygulamaları yapılmaktadır. Ayrıca, görüntülerdeki gürültüleri temizleme konusunda otokodlayıcılar başarılı sonuçlar elde etmektedir [43]. Yarı otokodlayıcılar ise resim yazıları veya açıklamalarını modele ekleyerek resimlerin özelliklerinin çıkarılması veya öneri sistemlerinde yan bilgilerin de modele dâhil edilerek daha isabetli öneriler verilmesi amacıyla farklı alanlarda kullanılabilir [10]. Otokodlayıcıların birçok türü mevcuttur [44]:

- Denoising autoencoder
- Sparse autoencoder
- Deep autoencoder
- Contractive autoencoder
- Concrete autoencoder
- Undercomplete autoencoder
- Overcomplete autoencoder

- Convolutional autoencoder
- Multinomial / Standard variational autoencoder
- Regularized autoencoder

3.6 Yoğun Yeniden Besleme

Eğitim ve çıktı verilerinin çok seyrek olma probleminden bahsedilmişti. Buna karşın çıkış katmanı yoğundur. Başarılı bir aktivasyon fonksiyonu ile idealleştirilmiş bir senaryoda; $\forall i: x_i \neq 0$ olduğunda $f(x)_i = x_i$ ve bir $x_i = 0$ olduğunda $f(x)_i$ i. öğeler için tüm kullanıcıların gelecekteki derecelendirmelerini doğru bir şekilde tahmin eder; yani kullanıcı yeni k öğesini derecelendirirse $f(x)_k = x'_k$ ve $f(x) = f(x')$ olduğu anlamına gelir. Bu sebeple $y = f(x)$ iyi eğitilmiş bir otokodlayıcının sabit noktası olmalıdır: $f(y) = y$. Sabit nokta kısıtlamasını uygulamak ve yoğun eğitim güncellemeleri yapabilmek için her optimizasyon yinelemesi yoğun yeniden besleme adımlarıyla aşağıdaki gibi tekrardan gerçekleşir:

1. Seyrek x girdisi verildiğinde, yoğun $f(x)$ çıktısı ve kayıp fonksiyonu hesaplanır. (ileri yayılım)
2. Ardından gradyanlar hesaplanır ve ağırlıklar güncellenir. (geri yayılım)
3. Yoğun $f(x)$ çıktısı yeni bir x girdisi olarak alınıp bu kez $f(f(x))$ çıktısı hesaplanır. (ikinci ileri yayılım)
4. Ardından gradyanlar hesaplanır ve ağırlıklar güncellenir. (ikinci geri yayılım)

Adım (3) ve (4), her yineleme için birden fazla kez de gerçekleştirilebilir [12].

3.7 Kayıp Fonksiyonu

CE, pratikte, daha iyi genelleme performansına ve daha hızlı bir eğitim hızına yol açmaktadır [45]. Mutlak hataya dayanan kare hata hatanın aksine, ağın çıktılarının göreceli hatalarına bağlıdır. Bu sebeple hem küçük hem de büyük sınıf değerleri üzerinde iyi bir performans gösterebilir. İki sınıflı bir sınıflandırma problemi için; $y^{(i)}$ çıktı verisi, $\hat{y}^{(i)}$ ise çıktı tahmini olursa,

$$J(\theta) = -\sum_{i=1}^{m}\{y^{(i)} \ln\hat{y}^{(i)} + (1 - y^{(i)}) \ln (1 - \hat{y}^{(i)}\} \quad (3.36)$$

(3.36) eşitliği, hataları minimize eden *kayıp* fonksiyonudur [46].

3.8 Düzenlileştirme

[47]'ye göre, derin sinir ağları çok büyük parametrelere sahip güçlü makine öğrenmesi sistemleridir. Dolayısıyla, bir yavaşlık problemi de söz konusudur. Ayrıca, bu sistemlerde meydana gelen aşırı öğrenme problemi, modelin genelleme performansında düşüşe yol açmaktadır.

İleri beslemeli ağlarda, aşırı öğrenmeyi önlemenin birçok yolu vardır:

- *Önceden eğitilmiş (pre-trained)* modeller kullanmak,
- l_1 ya da l_2 *düzenlileştirme* terimleri kullanmak,
- Her gizli birimin ağırlık vektörünün maksimum kare uzunluğuna bir kısıtlama koymak,
- Düğüm seyreltme metodu uygulamak gibi.

Bu noktada devreye giren düğüm seyreltme, bu sorunları çözmek için kullanılan tekniklerden biridir. Ana fikri, düğümlerin birbirlerine aktardıkları bilgiler sebebiyle meydana gelen aşırı derecede uyum problemini, sinir ağındaki bazı düğümleri (bağlantılarıyla birlikte) rastgele bir şekilde bırakarak çözmektir. Bu yaklaşım, düğümlere gürültü eklenerek sinir ağlarının düzenlenmesi olarak da düşünülebilir. Düğüm seyreltme kullanılmasının ağların performansını önemli ölçüde geliştirdiği kanıtlanmıştır. Düğüm seyreltme, gizli birimlerine gürültü ekleyerek bir sinir ağını düzenlemenin bir yolu olarak da yorumlanabilir. Bu sayede, önceki düğümlerden herhangi bir bilgi edinmeyen düğümler, diğer düğümlerden bağımsız bir şekilde hatasını düzeltmeyi öğrenir. İlişkileri kesilecek düğümler, her düğüm, diğer düğümlerden bağımsız olacak şekilde, seçilen sabit p olasılığı ile belirlenir.

n adet gizli düğümden, p birim çıkararak 2^n adet kombinasyon örneklenebilir. Eğitim sırasında, bu 2^n boyutundaki katman eğitilerek her düğüm $(1-p)$ olasılığıyla tutulur. Test anında ise, her bir düğümün ağırlığı $(1-p)$ olasılığıyla

çarpılarak küçültülür. Bu sayede ağ, daha genelleyebilir bir yapıya sahip olduğundan aşırı öğrenme problemi önlenmiş olur.

Matematiksel olarak ifade edilecek olursa, $l \in \{1, 2, ..., L\}$ ağın gizli katmanları olsun. z^l, l katmanındaki çıkış düğümlerinin toplam vektörünü; y^l, l katmanındaki çıktıların vektörünü göstersin. y^0 ise, girdi vektörü olsun. W_l, l katmanındaki ağırlıklar; b_l l katmanının biası olsun. İleri beslemeli standart ağlarda,

$$z_i^{(l+1)} = w_i^{(l+1)} * y^l + b_i^{(l+1)} \tag{3.37}$$

$$y_i^{(l+1)} = f\left(z_i^{(l+1)}\right) \tag{3.38}$$

olacaktır. Katmanlara, düğüm seyreltme uygulandığı zaman,

$$r_j^{(l)} \sim Bernoulli(p) \tag{3.39}$$

$$\tilde{y}^{(l)} = r^{(l)} * y^{(l)} \tag{3.40}$$

$$z_i^{(l+1)} = w_i^{(l+1)} * \tilde{y}^{(l)} + b_i^{(l+1)} \tag{3.41}$$

$$y_i^{(l+1)} = f\left(z_i^{(l+1)}\right) \tag{3.42}$$

olacaktır. *, element bazında vektör çarpımını ifade eder. Herhangi bir l katmanındaki j nöronu için, $r_j^{(l)}$, Bernoulli rastgele sayısıdır ve $r^{(l)}$; her biri, p olasılığı 1 olan bağımsız Bernoulli rastgele değişkenlerinin bir vektörüdür. Her bir katmandan $r^{(l)}$ vektörü örneklenerek elde edilen alt ağ çıktıları $y^{(l)}$ ile eleman bazında çarpılarak $\tilde{y}^{(l)}$ oluşturulur ve bir sonraki katmana girdi olarak verilir. Bu akış, her katman için yenilenir. Ardından, elde edilen gradyanlar alt ağlar üzerinden geri yayılır. Test esnasında ise basit bir ortalama alma yöntemiyle, elde edilen alt ağ ağırlıklarının ortalaması alınarak

$$w_{test}^{(l)} = p * w_{test}^{(l)} \tag{3.43}$$

şeklinde ifade edilir.

3.9 Bilgi Kriterleri

Bilgi kriterleri (information criteria), bir modelin seçilebilmesi için model hakkında bilgi sağlayan yaklaşımlardır. Yapay sinir ağı kullanan derin öğrenme modellerinde farklı düğüm sayısına sahip mimariler birbiriyle kıyaslanırken de sıkça yararlanılır. Birçok bilgi kriteri mevcuttur, ancak burada sıkça kullanılan bilgi kriterlerinden bahsedilecektir. *Akaike Bilgi Kriteri* [48],

$$AIC = -2\ \ln(L) + 2p \tag{3.44}$$

Düzeltilmiş Akaike Bilgi Kriteri [49],

$$AICc = \ -2\ \ln(L) + 2p\left(\frac{n}{n-p-1}\right) \tag{3.45}$$

Bayes Bilgi Kriteri [50],

$$\mathrm{BIC} = -2\ln(\mathrm{L}) + \mathrm{p}\ln(\mathrm{n}) \tag{3.46}$$

şeklinde gösterilebilir. Bilgi kriterleri için L, likelihood değeri; p, parametre sayısı; n, eğitim setinin gözlem sayısını gösterir. Her bir bilgi kriterinin küçük olması daha iyi bir model olduğu anlamına gelmektedir.

3.10 MinMax Ölçekleyici

Veriyi ölçeklendirmek için kullanılan yöntemlerden biridir. Belirlenen minimum ve maksimum değerler arasında, verideki her bir değişkeni ölçekler. Genellikle, minimum değer 0, maksimum değer ise 1 olarak alınır. Aşağıdaki şekilde basit bir formül aracılığıyla hesaplanır:

$$x' = \frac{X_i - x_{min}}{x_{max} - x_{min}} \tag{3.47}$$

(3.47) eşitliğinde; X_i gözlem değerini, x_{min} her bir değişken içerisinde yer alan en küçük gözlem değerini, x_{max} ise her bir değişken içinde yer alan en büyük gözlem değerini ifade eder.

3.11 Aktivasyon Fonksiyonları

Bir girdinin ağırlığı (net girdi), aktivasyon fonksiyonuna girmeden önce, belirli yöntemlerle hesaplanır. Ardından bir aktivasyon fonksiyonundan geçirilir. Girdinin tipine göre, birçok farklı aktivasyon fonksiyonu kullanılabilir. Girdiyi,

matematiksel bir formül aracılığıyla işleyerek bir çıktı üretir. Aktivasyon fonksiyonlarınım, genellikle, doğrusal olmayan bir fonksiyon olması istenir. Bu sayede, doğrusal olmayan özelliklerin ortaya çıkarılmasını sağlar.

Sigmoid fonksiyonu, literatüre bakıldığında ikili sınıflandırmada çıktı katmanında en sık kullanılan aktivasyon fonksiyonudur. Gizli katmanlarda, öğrenmeyi zorlaştırdığı için tercih edilmemektedir. Doğrusal ve doğrusal olmayan özellikler için çıktılar üretebilir ve aşağıdaki formül ile çıktı değerini hesaplar [51]:

$$F(Net) = \frac{1}{1 + e^{-Net}} \tag{3.48}$$

Gizli katmanlarda ise *ReLU* fonksiyonu yaygın olarak tercih edilir. Optimizasyonu diğer fonksiyonlara göre kolaydır. Elde edilen sonuç çoğunlukla 0 değerlerinden oluştuğundan türevler daha robust hâle gelir. *Leaky ReLU* ve *parametrik ReLU (PReLU)* gibi versiyonları da mevcuttur. Aşağıdaki şekilde çıktıyı üretir [52]:

$$\varphi(x) = \max(0, x) \tag{3.49}$$

ReLU'nun aksine *SELU* 'da, 0'dan küçük değerler göz ardı edilmez. Bu değerler, doğrusal olmayan ve negatif bir fonksiyondan geçer ve aşağıdaki şekilde çıktıyı üretir [53]:

$$h(x) = \lambda \begin{cases} x, & x > 0\ ise \\ ae^{x} - a, & x \leq \text{ise} \end{cases} \tag{3.50}$$

3.12 Adam Algoritması

Adam (adaptive momentum), diğer algoritmalardan farklı olarak yanlılık değerlerine de düzeltme işlemi uygular. Bu sebeple, momentlerin tahmini için azalma oranlarını ifade eden iki parametreye sahiptir. ρ1, birinci moment; ρ2 ikinci moment olmak üzere, bu parametreler [0, 1) aralığındadır. Ayrıca, ε öğrenme çarpanı ve δ sabit değeri mevcuttur. Ağırlıkların başlangıç değeri oluşturulduktan sonra, moment değişkenleri 0 alınarak eğitim başlatılır. Eğitim verileri model tarafından görüldükçe, moment değerleri güncellenir. Yanlılık değerleri de ayrıca düzeltilir [54].

3.13 Performans Metrikleri

Kesinlik (Precision@k), duyarlılık (Recall@k), normalize indirgenmiş kümülatif kazanç (NDCG@k) ve ortalama kesinlik (MAP@k) gibi metrikler sıralamaya dayalı metriklere örnek verilebilir. Bu metriklerin her biri farklı açılardan yaklaşarak sıralanan öğelerin kalitesine karar verir. Bu metriklerden kesinlik, seçilen bir öğenin ilgili olma olasılığını temsil eder. Duyarlılık ise ilgili bir öğenin seçilme olasılığını temsil eder. Duyarlılıktaki performans artışı kesinlikte bir azalmaya yol açtığından iki metrik ters orantılıdır. Seçilen bir k sayısına ve öğenin sıralamadaki pozisyonuna göre hesaplanırlar:

$$Precision@k = \frac{1}{|U|} \sum_{u \in U} \frac{|Rel_u@k|}{k} \tag{3.50}$$

$$Recall@k = \frac{1}{|U|} \sum_{u \in U} \frac{|Rel_u@k|}{|Rel_u|} \tag{3.51}$$

(3.50) ve (3.51) eşitliğine göre; Rel_u, u kullanıcısı için ilgili öğeler kümesini ve $Rel_u@k$, k sınırına kadar önerilen ilgili öğelerin sayısıdır. Duyarlılık, *isabet oranı (hit rate)* olarak da anılır. İsabet oranı, en az bir doğru öneriye sahip kullanıcıların yüzdesi olarak da tanımlanır. Duyarlılık, k sınırı içindeki tüm öğelerin eşit derecede önemli olduğunu kabul eder.

MAP ise ilgili her yeni öğe ede edildikten sonra elde edilen kesinlik sonuçlarının ortalamasını alarak kullanıcının öğe sıralamasının özetini sağlar:

$$MAP@k = \frac{1}{|U|} \sum_{u} \frac{1}{|Rel_u|} \sum_{i \in Rel_u} Precision@rank(u,i) \tag{3.52}$$

(3.52) eşitliğine göre; $rank(u,i)$, u. kullanıcının listesindeki i. öğenin sıralama konumunu gösterir. Bu nedenle kesinlik, ilgili her bir öğenin önerildiği konumda hesaplanır.

Son olarak NDCG metriği, sıralamanın en üstünden başlayarak toplanan ve daha düşük sıralarda indirilebilen derecelendirilmiş alaka düzeyini kullanır. Daha yüksek sıraların daha düşük sıralara göre önemini vurgulamak için monoton artan bir indirim kullanır:

$$NDCG@k = \frac{1}{|U|}\sum_{u}\frac{1}{IDCG_u^{p_u}}\sum_{p=1}^{p_u} f_{dis}(rel(u, i_p), p) \tag{3.53}$$

(3.53) eşitliğine göre; f_{dis}, indirim fonksiyonu; IDCG farklı kullanıcı ve veri kümelerini karşılaştırmak için bir normalleştirme faktörü görevi gören u kullanıcısı için k sınırına kadar ideal bir sıralama ile elde edilen puanı ifade eder. Ayrıca p_u, her kullanıcı için değerlendirilen maksimum öğe sayısını belirtir ve tüm kullanıcılar için aynı olan bir k sınırı olduğu varsayılır [55].

4
UYGULAMA

4.1 Geliştirme Araç ve Gereçleri

Tüm süreçler, *Python 3.8.12* programlama dilinde gerçekleştirilmiştir. Derin öğrenme tabanlı model, *TensorFlow* kütüphanesiyle mimarileştirilmiştir (tf.keras==2.7.0) [56]. HP model, Intel® Core™ i7-8565U CPU@1.80GHz×8, Intel® UHD Graphics 620 (WHL GT2), 16GB RAM ve 64-bit özelliklere sahip dizüstü bilgisayar kullanılmıştır. Geliştirmeler, *Ubuntu 18.04.5 LTS* üzerinde yapılmıştır. *TensorFlow ranking, keras tuner, pandas, numpy, matplotlib, searborn, scikit-learn, flake8, pep8* ve *scipy* kütüphanelerinden de yararlanılmıştır.

4.2 Veri Seti ve Ön İşleme

Minnesota Üniversitesi, Bilgisayar Bilimleri ve Mühendislik Bölümü, GroupLens Araştırma Grubu tarafından 2000 yılında oluşturulan *halka açık (publicly)* ve öneri sistemlerinde sıklıkla kullanılan *Movielens 1M (ML-1M)* veri seti kullanılmıştır [57]. Veri seti, Tablo 4.1'de gösterilen üç farklı parçadan oluşur:

Tablo 4.1 Veri seti

Veri	**Satır sayısı**	**Sütun sayısı**
Etkileşim verisi *(ratings)*	1000209	3
Öğe verisi *(items)*	3883	3
Kullanıcı verisi *(users)*	6040	4

Kategorik değişkenlerin *sıcak kodlamasında (one-hot encoding)* ortaya çıkan *kukla değişken tuzağı (dummy variable trap),* yüksek düzeyde ilişkili niteliklerin olduğu ve bir değişkenin diğerlerinin değerini tahmin ettiği bir senaryodur. Bu nedenle, bir kukla değişken, diğer kukla değişkenlerle yüksek oranda ilişkilidir. Bu da değişkenlerin gereksiz yere artmasına ve makinelerin yanılmasına sebep olur. Bu problemin önlenmesi için *(kategori sayısı – 1)* olacak şekilde değişkenler

kodlanır. Örneğin, cinsiyet değişkeni iki kategoriden oluşur ve kodlama sonucunda bir sütundan iki sütuna dönüşmesi gerekir, ancak kodlanırken iki ayrı sütun oluşturulmayıp tek bir sütunda kodlanarak bu sorunun üstesinden gelinir. Yine, değişken içerisindeki kategoriler kodlanmış ifade olsa bile bunların *sıralı (sequential)* bir şekilde kodlanmış olması özellikle derin öğrenme ağları için kritik bir önem taşır.

Öğe verisi öğe bazlı, kullanıcı ve etkileşim verisi ise kullanıcı bazlı veridir. Öğe, kullanıcı ve etkileşim verisini tek bir veride birleştirmek için her üç verinin de kullanıcı bazlı olması gerekir. Bu sebeple, öğe verisi kullanıcı bazlı veriye çevrilmelidir. Bu da öğe verisinin kullanıcı ve etkileşim verisiyle birleştirilip öğe verisindeki kullanılacak her bir değişkendeki kategorilerinin kullanıcı bazında toplam sayıları alınarak yapılmıştır. Örneğin bir kullanıcı, 1989-2000 yapımı öğe ile kaç kere etkileşime geçti? İlgili bölümlerde detaylandırılmıştır.

Veri setindeki etkileşim, öğe ve kullanıcı verisinin içeriği ile ilgili bilgi verildikten sonra ön işleme sonrası elde edilen nihai hâllerine değinilecektir.

4.2.1 Kullanıcı verisi

Kullanıcıların demografik bilgilerini içerir. Eşsiz kullanıcı kimlik numarası bazında; kullanıcının cinsiyet, yaş, meslek ve posta kodu mevcuttur (Şekil 4.1).

	userID	gender	age	occupation	zip_code
0	1	F	1	10	48067
1	2	M	56	16	70072
2	3	M	25	15	55117
3	4	M	45	7	02460
4	5	M	25	20	55455
...	...	...	...	...	...
6035	6036	F	25	15	32603
6036	6037	F	45	1	76006
6037	6038	F	56	1	14706
6038	6039	F	45	0	01060
6039	6040	M	25	6	11106

6040 rows × 5 columns

Şekil 4.1 Kullanıcı verisi

Tablo 4.2'de kullanıcı verisinin değişken adı, tipi, kategori sayısı ve ölçeği yer almaktadır.

Tablo 4.2 Kullanıcı verisinin içeriği

Değişken adı	Değişken tipi	Kategori sayısı	Değişken ölçeği
Kullanıcı kimlik numarası *(userID)*	Kategorik	6040	Nominal
Cinsiyet *(gender)*	Kategorik	2	Nominal
Yaş *(age)*	Kategorik	7	Ordinal
Meslek *(occupation)*	Kategorik	21	Nominal
Posta kodu *(zip_code)*	Kategorik	3439	Nominal

Tablo 4.2'de yer alan *Posta kodu* değişkeninin kardinalite yüksekliği sebebiyle matrisin boyutunu artırdığından bunun sonucunda da seyrekliği artırdığından kullanılmamasına ve veriden düşürülmesine karar verilmiştir.

Şekil 4.2'de yapılan ön işleme sonrası kullanıcı verisi ekran görüntüsü yer almaktadır.

	age	gender_M	occupation_1	occupation_10	occupation_11	...	occupation_5	occupation_6	occupation_7	occupation_8	occupation_9
userID											
1	0.0	0.0	0.0	1.0	0.0	...	0.0	0.0	0.0	0.0	0.0
2	6.0	1.0	0.0	0.0	0.0	...	0.0	0.0	0.0	0.0	0.0
3	2.0	1.0	0.0	0.0	0.0	...	0.0	0.0	0.0	0.0	0.0
4	4.0	1.0	0.0	0.0	0.0	...	0.0	0.0	1.0	0.0	0.0
5	2.0	1.0	0.0	0.0	0.0	...	0.0	0.0	0.0	0.0	0.0
...	...	...	...	...	...	...	...	...	...	...	...
6036	2.0	0.0	0.0	0.0	0.0	...	0.0	0.0	0.0	0.0	0.0
6037	4.0	0.0	1.0	0.0	0.0	...	0.0	0.0	0.0	0.0	0.0
6038	6.0	0.0	1.0	0.0	0.0	...	0.0	0.0	0.0	0.0	0.0
6039	4.0	0.0	0.0	0.0	0.0	...	0.0	0.0	0.0	0.0	0.0
6040	2.0	1.0	0.0	0.0	0.0	...	0.0	1.0	0.0	0.0	0.0

6040 rows × 22 columns

Şekil 4.2 Ön işlemeden sonraki kullanıcı verisi

Şekil 4.2'ye göre 2 kategoriden oluşan *cinsiyet* değişkeni, 7 kategoriden oluşan *yaş* değişkeni ve 21 kategoriden oluşan *meslek* değişkeni sıcak kodlanmıştır.

4.2.2 Öğe verisi

Öğelerin özelliklerini içerir. Şekil 4.3'te öğe verisi ekran görüntüsü sunulmuştur.

	itemID	item_name	genres
0	1	Toy Story (1995)	Animation|Children's|Comedy
1	2	Jumanji (1995)	Adventure|Children's|Fantasy
2	3	Grumpier Old Men (1995)	Comedy|Romance
3	4	Waiting to Exhale (1995)	Comedy|Drama
4	5	Father of the Bride Part II (1995)	Comedy
...	...	...	...
3878	3948	Meet the Parents (2000)	Comedy
3879	3949	Requiem for a Dream (2000)	Drama
3880	3950	Tigerland (2000)	Drama
3881	3951	Two Family House (2000)	Drama
3882	3952	Contender, The (2000)	Drama|Thriller

3883 rows × 3 columns

Şekil 4.3 Öğe verisi

Şekil 4.3'e göre öğe verisi, eşsiz öğe kimlik numarası bazında; öğe adı ve öğenin türünü (aksiyon, komedi, dram vb.) içerir. Bir öğenin birden fazla türü olabilir. Ayrıca öğe adı, 81 adet yayımlanma yılını da içermektedir.

Tablo 4.3'te öğe verisinin değişken adı, tipi, kategori sayısı ve ölçeği yer almaktadır.

Tablo 4.3 Öğe verisinin içeriği

Değişken adı	Değişken tipi	Kategori sayısı	Değişken ölçeği
Öğe kimlik numarası *(itemID)*	Kategorik	3883	Nominal
Öğe adı + yayımlanma yılı *(item_name)*	Kategorik	3883	Nominal
Öğe türü ya da türleri *(genres)*	Kategorik	18	Nominal
*Öğe yılı *(item_year)*	Kategorik	81	Ordinal

Orijinal veride bulunmayan ancak ön işleme sonrası veriye eklenen yeni değişken

Tablo 4.3'e göre bir öğenin birden fazla türü (*Animation*|*Comedy*|*Drama* vb. gibi) olabilir. Bu yüzden, karakterlerden oluşan öğe türleri her biri ayrı bir tür olacak şekilde parçalanıp öğe bazında çoklanmıştır. Örneğin, bir öğenin üç türü var ise o öğe veride üç kere tekrar etmiş ve öğenin üç türü ayrı ayrı yer almıştır. Kullanıcı

bazlı olması içinse her kullanıcının etkileşime geçtiği öğenin türüne bakılmış ve hangi türden kaç adet öğeyle etkileşime girdiği bilgisi çıkarılmıştır. Örneğin bir kullanıcı; komedi türündeki öğelerle 50, korku türündeki öğelerle 20 etkileşime geçmiş olabilir. Etkileşime girmediği türler ise 0 olarak doldurulmuştur. Bu işlemler sonucunda *(kullanıcı sayısı x tür sayısı)* boyutunda bir matris elde edilmiştir.

Öğe adının yayımlanma yılını da içermesi sebebiyle *özellik mühendisliği (feature engineering)* ile kullanıcıların etkileşime geçtiği öğelerin yayımlanma yıllarından yeni bir *özellik çıkarma (feature extraction)* işlemi yapılmıştır. Karakterlerden oluşan her bir öğenin yılı *düzenli ifadeler (regular expressions / regex)* ile yakalanıp *öğe yılı* adıyla yeni bir sütun olarak öğe verisine eklenmiştir. 1919-2019 yıllarını içeren *öğe yılı* değişkeni, 81 kategoriye sahiptir. Kategori sayısının fazla olması sebebiyle, yıllar 10 yıllık gruplara ayrılmış ve 8 kategoriye düşürülmüştür. Ardından öğe türlerine uygulanan işlem bu değişkene de uygulanmıştır. Örneğin bir kullanıcı; 1989-2000 yapımı öğelerle 50, 1929-1939 yapımı öğelerle 20 etkileşime geçmiş olabilir. Bu işlemler sonucunda *(kullanıcı sayısı x öğe yılı sayısı)* boyutunda bir matris elde edilmiştir.

Ardından öğe verisinden elde edilen bu iki matris birleştirilip *(kullanıcı sayısı x (tür sayısı + öğe yılı sayısı)* boyutunda bir matrise dönüştürülmüştür. Elde edilen matris *minmax ölçekleyici* işlemine tabii tutularak ölçeklendirilmiştir. Şekil 4.4'te ön işleme sonrası kullanıcı bazlı öğe verisi ekran görüntüsü verilmiştir.

userID	Animation	Children's	Comedy	Musical	Romance	...	1949	1959	1939	1979	1919
1	0.208955	0.135338	0.028947	0.190476	0.016393	...	0.036458	0.029240	0.040000	0.026834	0.000000
2	0.000000	0.000000	0.036842	0.000000	0.071038	...	0.031250	0.070175	0.033333	0.048301	0.000000
3	0.029851	0.015038	0.050000	0.015873	0.021858	...	0.000000	0.029240	0.000000	0.075134	0.000000
4	0.000000	0.007519	0.000000	0.000000	0.005464	...	0.000000	0.035088	0.000000	0.019678	0.000000
5	0.044776	0.030075	0.076316	0.031746	0.049180	...	0.020833	0.000000	0.013333	0.007156	0.000000
...	...	...	...	...	...	...	...	...	...	...	...
6036	0.328358	0.180451	0.250000	0.301587	0.289617	...	0.296875	0.327485	0.220000	0.284436	0.181818
6037	0.014925	0.022556	0.084211	0.047619	0.071038	...	0.166667	0.169591	0.133333	0.103757	0.045455
6038	0.029851	0.000000	0.021053	0.000000	0.027322	...	0.015625	0.000000	0.000000	0.017889	0.000000
6039	0.119403	0.067669	0.110526	0.412698	0.120219	...	0.223958	0.198830	0.153333	0.050089	0.045455
6040	0.029851	0.037594	0.144737	0.111111	0.136612	...	0.161458	0.152047	0.146667	0.164580	0.045455

6034 rows × 26 columns

Şekil 4.4 Ön işlemeden sonraki kullanıcı bazlı öğe verisi

4.2.3 Etkileşim verisi

Kullanıcıların etkileşimlerini içerir. Şekil 4.5'te etkileşim verisi ekran görüntüsü sunulmuştur. Şekil 4.5'e göre öğelerin, belirli zamanlarda, kullanıcılar tarafından 1'den 5'e kadar tamsayı derecelendirmesi mevcuttur. Yani, etkileşim verisi, açık geri bildirimlerden oluşur. Her kullanıcı en az 20 her öğe en az 1 etkileşime sahiptir. Kullanıcı ve öğeler ve eşsiz kimlik numaralarına sahiptir.

	userID	itemID	rating	timestamp
0	1	1193	5	978300760
1	1	661	3	978302109
2	1	914	3	978301968
3	1	3408	4	978300275
4	1	2355	5	978824291
...	...	...	...	...
1000204	6040	1091	1	956716541
1000205	6040	1094	5	956704887
1000206	6040	562	5	956704746
1000207	6040	1096	4	956715648
1000208	6040	1097	4	956715569

1000209 rows × 4 columns

Şekil 4.5 Etkileşim verisi

Tablo 4.4'te etkileşim verisinin değişken adı, tipi, kategori sayısı ve ölçeği yer almaktadır.

Tablo 4.4 Etkileşim verisinin içeriği

Değişken adı	Değişken tipi	Kategori sayısı	Değişken ölçeği
Kullanıcı kimlik numarası *(userID)*	Kategorik	6040	Nominal
Öğe kimlik numarası *(itemID)*	Kategorik	3706	Nominal
Derece *(rating)*	Kategorik	5	Ordinal
Zaman damgası *(timestamp)*	Tarih saat	-	Ordinal

Tablo 4.4'te yer alan *Zaman damgası*, *sıralı (sequential)* bir model mimarisi kullanılmadığı için veriden düşürülmüştür. Ardından, [58] tarafından önerilen üç adım uygulanarak etkileşim verisi filtrelenir:

- Derecelendirmesi 4'ün altında olan etkileşimler performans metriği hesaplama kısmında kullanılmak üzere filtrelenir. Bu filtrelemeyle, 4'ün

altında derecelendirilen öğelerin son *tıklama matrisi (click matrix)* üzerinde yer almayacağından emin olunur. Bu filtreleme uygulanmadığı takdirde, son matris daha da seyrek olacaktır.

- 5'ten az öğeyle etkileşime geçen kullanıcılar filtrelenir.
- Herhangi bir kullanıcı tarafından etkileşime geçilmeyen öğeler filtrelenir.

Filtreleme sonrasında etkileşim verisindeki kalan gözlem, kullanıcı ve öğe sayısıyla birlikte hesaplanan seyreklik oranı Tablo 4.5'te yer almaktadır.

Tablo 4.5 Filtreleme sonrası veri miktarları ve seyreklik oranı

Orijinal satır sayısı	1. adım sonrası satır sayısı	2. ve 3. adım sonrası satır sayısı	Son kullanıcı sayısı	Son öğe sayısı	Seyreklik oranı
1000209	575281	575272	6034	3466	1 – (575272 / (6034 * 3466))

Tablo 4.5'e göre seyreklik oranı, *%97,25* (yoğunluk *%2,75*) olarak hesaplanmıştır.

Etkileşime geçmek kavramının yerine *tıklamak* kavramı; *etkileşim matrisi* yerine de *tıklama matrisi* kavramı sıkça kullanılır. Oluşturulan etkileşim matrislerinde hücreler, kullanıcı bir öğeyle etkileşime geçmediğinde ya da bir öğeyle etkileşime geçip beğenmediğinde (derecelendirmesi 4'ün altında) 0; aksi durumlarda 1 içerecek şekilde oluşturulur. Tıklama matrisi, sadece, 0 ve 1 değerlerinden oluşmalıdır. Derece, açık geri bildirimler olan 1'den 5'e kadar tamsayı değerinden oluşmaktaydı. Açık geri bildirimler örtük geri bildirimlere dönüştürüldüğünden problem, *derecelendirme (rating)* probleminden *sıralama (ranking)* problemine; değişken ölçeği, ordinal ölçekten nominal ölçeğe dönüşmüştür.

Kullanıcı, öğe ve etkileşim verisi ön işleme aşamalarından geçirilmiştir. Etkileşim verisi, veri parçalama işlemlerinden geçirildikten sonra, kullanıcı ve öğe verisi bu aşamada parçalara dahil edilecektir.

4.2, 4.3 ve 4.4 başlıklarının akış grafiği, Şekil 4.6'da gösterilmiştir. Şekil 4.6, kullanıcı ve öğe setinin ön işlemlerden geçirildiğini varsayarak etkileşim verisinin parçalanma kısmından başlar.

4.3 Veri Setini Parçalama

Ön işlemeler gerçekleştirildikten sonraki adım veri setini parçalama adımıdır. Bu süreç, yine, [38] tarafından uygulanan yöntemle, aşağıdaki sırayla gerçekleşir:

- Son kullanıcılar; rastgele bir şekilde *eğitim (train), doğrulama (validation) ve t*est seti olarak ayrılmıştır (Tablo 4.6). Bu aşamada, her bir veri setine düşen eşsiz kullanıcıların rastgele olmaması için belirli bir parametre konulmadıysa sonuçlar her defasında farklı gelecektir. Bu sebeple, belirli bir değer ile sabitlenmesi, sürekliliğin sağlanması için faydalı olacaktır.

Tablo 4.6 Parçalanmış veri setlerinin içerdiği kullanıcı sayıları

Eğitim verisi	**Doğrulama verisi**	**Test verisi**	**Toplam kullanıcı sayısı**
4224 kullanıcı (~70%)	905 kullanıcı (~15%)	905 kullanıcı (~15%)	6034 kullanıcı

- Model, eğitim kullanıcılarının tıklama geçmişi kullanılarak eğitileceğinden doğrulama ve test setlerinde bulunan öğelerin sadece eğitim setinde bulunan öğeleri içerdiğinden emin olunmalıdır. Yani, doğrulama ve test seti, eğitim setinde olmayan öğeleri içermemelidir.
- Eğitim, doğrulama ve test setlerinin elde edilmesinden sonra; kullanıcı tercihlerini içeren ve her satırın bir kullanıcıyı her sütunun da bir öğeyi temsil ettiği eğitim, doğrulama ve test setinin tıklama matrisi oluşturulur.
- Eğitim setinin, tüm eğitim kullanıcılarının tam geçmişini içeren bir tıklama matrisi olmasına karşın doğrulama seti, *doğrulama setinin eğitim seti (val_data_tr)* ve *doğrulama setinin test seti (val_data_te)* olarak ikiye bölünür. Yine, aynı şekilde, test seti de *test kümesinin eğitim seti (test_data_tr)* ve *test kümesinin test seti (test_data_te)* olarak ikiye bölünür.
- Doğrulama setinin eğitim seti, etkileşim matrisinde bir olarak etiketlenen öğeler içerisinden kullanıcı bazında ~%70'ini içerir. Etkileşime geçilen öğelerin geri kalan ~%30'u ise doğrulama setinin test setini oluşturur. Doğrulama setinin eğitim setinde bulunan bir öğe, doğrulama setinin test setinde bulunmaz. Test seti için de aynı bölme işlemi tekrarlanır.

4.4 Nihai Veri Setinin Elde Edilmesi

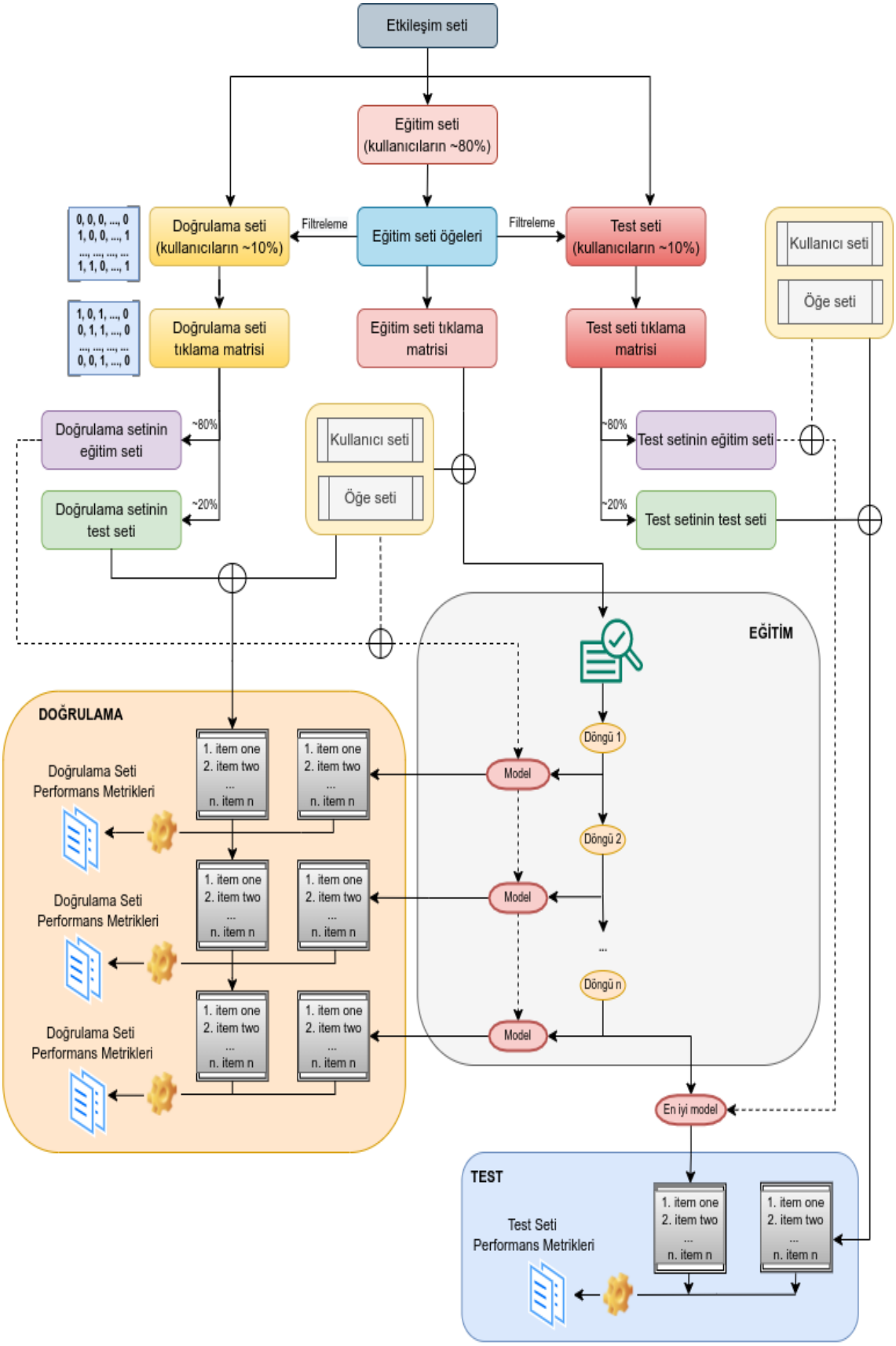
Tıklama matrisinin veri setleri (eğitim, doğrulama, test) elde edildikten sonra, bu veri setlerinin kullanıcı ve öğe veri setiyle kullanıcı bazında birleştirilmesi gerekir. Her bir veri setinin içerdiği eşsiz kullanıcılar mevcuttur. Bu sebeple, her bir veri seti, kullanıcı bazında ayrı ayrı birleştirilir. Toplamda; (*kullanıcı sayısı x (tıklama matrisi + kullanıcı seti + öğe seti))* boyutunda üç matris elde edilir. Ön işleme ve etkileşim verisini parçalama kısımları tamamlandığına göre, model eğitilirken her bir döngü sonunda hesaplanan performans metriklerinin işleyiş metodu anlatılabilir.

4.5 Performans Metriklerinin Hesaplanması

Veri setleri bölündükten sonra, performans metrikleri aşağıdaki adımlarla hesaplanır:

- Her bir kullanıcı etkileşimlerinin ~%70'ini içeren doğrulama kümesinin eğitim seti, her bir döngünün sonunda modelin tahminlemesi için girdi olarak modele verilir. Modelin sonucu yani model tarafından öngörülen öneriler, her kullanıcı için önerilen öğeleri içeren bir *yeniden yapılandırılmış doğrulama setinin eğitim seti (reconstructed_val_data_tr)'* dir. Elde edilen önerilerden, kullanıcının izlediği öneriler düşürülür. Kalan önerilerin olasılıkları büyükten küçüğe doğru sıralanır.
- Kullanıcıların etkileşime geçtiği ve dereceleri kesin olarak bilinen ve dolayısıyla doğrulama setinin test setinde yer alabilen öğelerin kullanıcı bazında açık geri bildirimleri büyükten küçüğe doğru sıralanır. Bir *k* değeri belirlendikten sonra (genellikle k = 5, 10, 20, 50, 100), modelin tahmin ettiği ve kullanıcının etkileşime geçtiği öğeler birbiriyle karşılaştırılıp performans metrikleri hesaplanır. Örneğin k=10 olduğunda, her iki öneri listesinde ilk 10 öneri içerisinde birbiriyle eşleşen öğelerin sayısına göre farklı metrik hesaplamaları yapılır.
- Modelin performansını değerlendirmek için her döngünün sonunda; *MAP@k, NDCG@k, Recall@k, Precision@k* metrikleri kullanılarak *yeniden yapılandırılmış doğrulama setinin eğitim seti* ile *doğrulama setinin test seti* (her kullanıcının gerçek öğe tercihleri) karşılaştırılmış olur.

- Bir *geri çağırma (callback)* işlevi aracılığıyla, her bir döngü sonunda hesaplanan *NDCG@k* skoruna göre en iyi model her defasında kaydedilir. Döngü sayısı arttığında model aşırı ya da eksik öğrense bile en iyi NDCG skorunu veren model kaydedildiğinden öneriler bu eğitilmiş model üzerinden yapılır. Böylece, her bir döngü sonunda performans metriklerinin başarısı takip edilir.
- Modelin nihai değerlendirmesi için test setinin eğitim ve test seti kullanılır. Doğrulama kümesi üzerinden metrik hesaplarken uygulanan adımlar burada da takip edilir. Test setinin eğitim seti modele bir girdi olarak verilir ve model tarafından yapılan önerilerle yeniden yapılandırılmış test eğitim setini döndürülür. Ardından, yeniden yapılandırılmış test setinin eğitim seti farklı metrikler aracılığıyla test setinin test setiyle karşılaştırılır. Bu sayede, modelin daha önce görmediği veri üzerinde bir sınama yapılarak modelin ne kadar genelleştirilebilir olduğuna ne kadar iyi ya da eksik öğrendiğine karar verilir.



Şekil 4.6 Ana akış şeması

4.6 Deneysel Sonuçlar

Yapılan deneysel çalışmalarda, veri ölçeklendirme metodu olarak MinMax ölçekleyici kullanılmıştır. Ağ mimarisi oluşturulurken düğüm ve katman sayısının belirlenmesi ve ağın eğitilmesi için yeniden beslemenin etkisi araştırılmıştır. Önceden eğitilmiş model kullanılmamıştır. Doğrulama veri seti kullanılarak modelin doğrulama kaybına bakılmıştır. @TopK belirlenebilen performans metriklerinin *k* değeri 10 olarak seçilmiştir. Tablo 4.7'de, bu deneysel çalışmalarda kullanılan parametreler ve değerleri yer almaktadır.

Tablo 4.7 Deneysel sonuçlarda kullanılan parametreler

Parametre	**Değeri**
Parti boyutu *(batch size)*	128
Döngü *(epoch)* sayısı	200
Öğrenme oranı *(learning rate)*	1e-5
Optimizasyon algoritması	Uyarlanabilir Momentum *(ADAM)*
Rastgele ağırlık başlatıcı *(kernel initializer)*	Glorot Uniform (Xavier Uniform)
Aktivasyon fonksiyonu *(activation function)*	son katman: *sigmoid*, diğer katmanlar: *ReLU*
Kayıp fonksiyonu	İkili çapraz entropi (BCE)

Model mimarisinin belirlenmesinde hiperparametreler kritik bir önem arz eder. Hiperparametreler belirlenmeden önce düğüm ve katman sayısına karar vermek önemli bir aşamadır. Düğüm ve katman sayısına belirli deneysel karşılaştırmalarla karar verildikten sonra, hiperparametre optimizasyonu yapılarak modelin en iyi çalıştığı parametrelerine karar verilir. Bundan sonraki süreçler de en iyi parametreleri bulmak ve en iyi skorları elde etmektir. Parametrelerin ne kadar iyi seçildiği konusuna ise eğitilen modelin belirli hesaplamalara tabii tutulup çıkan sonuçların karşılaştırılması suretiyle karar verilebilir. Bunun kesin bir yolu yoktur, çünkü birçok parametreyi birbiriyle eş zamanlı olarak modeli eğiterek denemek ve doğru parametrelere karar vermek zaman ve hesaplama açısından maliyetlidir. Bu maliyetleri azaltan sistemler ve yaklaşımlar elbette mevcuttur.

Tablo 4.8'de, sonuçların karşılaştırılması için oluşturulan tablolarda yer alacak bazı değişkenler ve açıklamaları verilmiştir.

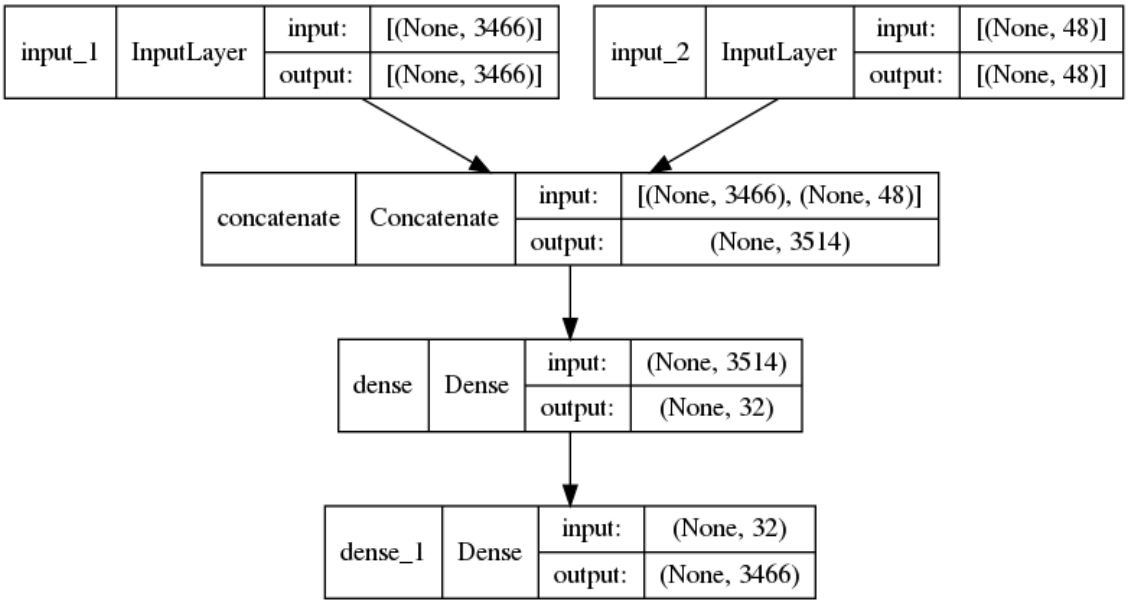
Tablo 4.8 Bazı değişken ve açıklamaları

Değişken	Açıklama
h	gizli düğüm sayısı
w	ağırlık
b	yan
p	parametre sayısı
Recall@k	duyarlılık
Precision@k	kesinlik
BCE	ikili çapraz entropi
L	toplam katman sayısı

4.6.1 Düğüm Sayısına Karar Verilmesi

Parametreler belirlenmeden önce, düğüm sayısına karar vermenin önemli bir aşama olduğundan bahsedilmişti. Düğüm sayısına karar vermek için, öncelikle, iki katmandan oluşan basit yarı-otokodlayıcılar kullanılmıştır. Bir adet kod ve çıkış katmanından oluşan basit yarı-otokodlayıcılarda, çıkış katmanının düğüm sayısı, kullanıcı-öğe matrisindeki öğelerin boyutu kadar olacaktır. Kod katmanının düğüm sayısına ise [32, 64, 128, 256, 512] düğümleri içerisinden her bir düğüm denenerek karar verilecektir.

Şekil 4.7'de iki katmanlı basit yarı-otokodlayıcı örnek mimarisine yer verilmiştir.



Şekil 4.7 Basit yarı-otokodlayıcı örnek mimarisi

Şekil 4.7'de görüldüğü üzere, model iki adet girişe sahiptir. Yarı-otokodlayıcı yaklaşımını uygulayabilmek için bundan sonraki mimarilerde de iki adet giriş verilecektir. Bu girişlerden biri kullanıcı-öğe matrisi, diğeri kullanıcı-yan bilgi matrisidir. Çıkış katmanı ise kullanıcı-öğe matrisinin boyutundadır.

Tablo 4.9'da, farklı düğümlere sahip modelden elde edilen performans sonuçları mevcuttur.

Tablo 4.9 Gizli düğüm sayısına göre 2 katmanlı modelin performans sonuçları

h	*p*	**Bilgi Kriterleri**			**BCE**		**NDCG@10**		**Recall@10**	
		AIC	*AICc*	*BIC*	*Eğit.*	*Doğr.*	*Eğit.*	*Doğr.*	*Eğit.*	*Doğr.*
32	226858	453407	-8603	822215	.0731	.0813	.6043	.5984	.0605	.0578
64	450250	900218	-8522	1632200	.0667	.0758	.6234	.6294	.0647	.0618
128	897034	1794068	-8482	7488922	.0684	.0776	.6651	.6590	.0667	.0638
256	1790602	3581003	-8461	6492027	.0475	.0651	.8879	.8548	.09	.0833
512	3577738	7155361	-8450	1297775	.0272	.0573	.9799	.9628	.094	.095

Tablo 4.9'a göre, 2 katmanlı basit otokodlayıcı modelde gizli katmandaki düğüm sayısı 128, 256 ve 512 için model kayıp fonksiyonu değerleri düşük ve NDGC ile Recall performans metrikleri yüksektir ancak düğüm sayısı 128'den 512'ye arttıkça

AIC, AICc ve BIC kriterlerinde artış görülmektedir. Bu nedenle, gizli katmandaki düğüm sayısı 128 seçilmiştir.

4.6.2 Katman Sayısına Karar Verilmesi

Düğüm sayısına karar verildiğine göre, bir sonraki aşama olan katman sayısına karar verilebilir. Bu aşamada, çıkış katmanı hariç bütün katmanların düğüm sayısı eşit ve 128 olacaktır. Farklı katmana sahip modeller üzerinden yapılan bazı hesaplamalar Tablo 4.10'da gösterilmiştir.

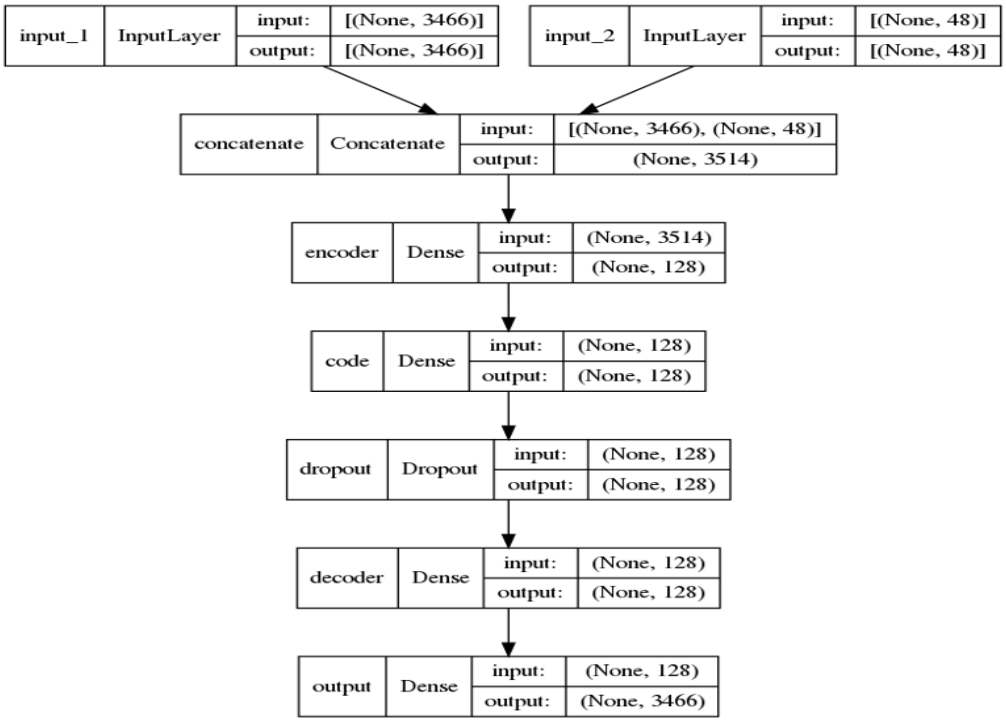
Tablo 4.10 Katman sayısına göre modelin performans sonuçları

L	*p*	**Bilgi Kriterleri**			**BCE**		**NDCG@10**		**Recall@10**	
		AIC	*AICc*	*BIC*	*Eğit.*	*Doğr.*	*Eğit.*	*Doğr.*	*Eğit.*	*Doğr.*
2	897034	**1794068**	**-8487**	**7488922**	,0684	,0776	,6651	,6590	,0667	,0638
4	930058	1860116	-8486	7764624	**,0640**	**,0723**	**,6852**	**,6773**	**,0687**	**,0657**
6	963082	1926164	-8485	8040326	,0655	,0730	,6599	,6581	,0662	,0639
8	996106	1992212	-8483	8316028	,0708	,0773	,5933	,5792	,0594	,0560
10	1029130	2058260	-8482	8591730	,0828	,0881	,4525	,4372	,0448	,0414

Tablo 4.10'a göre; 2 ve 4 katmana sahip modellerin performans sonuçlarının yakın olduğu görülür. 4 katmana sahip model; BCE, NDCG ve Recall performanslarında diğer modellere göre daha başarılıdır. Bilgi kriterlerinde ise 2 katmanlı model genel itibariyle diğer modellere göre daha başarılıdır. Model parametreleri optimize edilmediğinden ve bu sonuçlar ön analiz sonuçları olduğundan 4 katmanlı model ile devam edilebilir. Diğer bir önemli unsur ise katman sayısını daha fazla artırmanın model performansını iyileştirmediğidir. Düğüm sayısına karar verilip bunun sonucunda da katman sayısına karar verildiğine göre, model mimarisi oluşturulup hiperparametre optimizasyonuna geçilebilir.

4.6.3 Model Mimarisinin Oluşturulması

Belirli optimizasyonlar ile parametrelere karar verilmesi sürecine başlarken model mimarisinin oluşturulması gerekmektedir. Model mimarisinden kasıt katman sayısının net bir şekilde oluşturulmasıdır. Katman sayısına karar verildiğine göre, hiperparametre optimizasyonu yapılacak derin mimari oluşturulabilir. Şekil 4.8'de, düğüm sayısı 128 olan 4 katmanlı örnek bir derin yarı-otokodlayıcı model mimarisi yer almaktadır.



Şekil 4.8 4 katmanlı derin yarı-otokodlayıcı model mimarisi

Şekil 4.8'e göre; 3466 uzunluğa sahip bir kullanıcı-öğe matrisiyle, 48 uzunluğa sahip kullanıcı ve öğe yan bilgilerinin birleştirilerek 3514 uzunluğunda yeni bir matris elde edilmektedir. Elde edilen bu matris yoğun katmanlardan geçirilerek sadece kullanıcı-öğe matrisinin boyutu yani 3466 uzunluğa sahip bir matris olarak yeniden inşa edilmektedir. Dikkat edilirse kullanıcı ve öğe yan bilgileri kullanıcı-öğe matrisiyle birleştirilerek kod katmanında bazı gizli özellikleri çıkarılmakta ve elde edilen bu özellikler sayesinde sadece kullanıcı-öğe matrisi yeniden oluşturulmaktadır. Kullanıcı ve öğe yan bilgileri soğuk başlangıç problemine katkı sağlamak ve kod katmanında elde edilen gizli özelliklerin oluşturulmasına büyük bir katkı sağlamaktadır.

Hiperparametre optimizasyonunda, Şekil 4.8'deki mimari kullanılacaktır. Bu mimarinin model parametreleri belirlendikten sonra, mimarinin parametreleri güncellenecektir.

4.6.4 Hiperparametre Optimizasyonu

Bu tez çalışmasında, açık kaynak olan *Keras* kütüphanesinin *KerasTuner* adı verilen ölçeklenebilir bir hiperparametre optimizasyon *çerçevesi (framework)* kullanılmıştır. Bu yapı; *Rastgele Arama ile Arayıcı (RandomSearch Tuner), Bayes Optimizasyonu ile Arayıcı (BayesianOptimization Tuner) ve Hiperbant Arayıcı (Hyperband Tuner)* gibi işlevlere sahiptir [59].

- **Rastgele Arama ile Arayıcı:** Belirli bir arama alanından rastgele bir hiperparametre kombinasyonu dener.
- **Hiperbant Arayıcı:** *Uyarlanabilir kaynak tahsisi (adaptive resource allocation)* ve *erken durdurma (early stopping)* yoluyla *Rastgele Arama* yöntemini optimize eder. Önce, birkaç rastgele hiperparametre yapılandırmaları çalıştırır, ardından hangi yapılandırmaların iyi performans gösterdiğini seçer ve en iyi performans gösterenleri ayarlamaya devam eder.
- **Bayes Optimizasyonu ile Arayıcı:** Olasılıksal bir yaklaşım olan Bayesçi yaklaşımın Gaussian süreci ile hiperparametreler amaç fonksiyonu üzerinde bir olasılık puanına eşlenir. *Rastgele Arama* ve *Hiperbant* yaklaşımlarından farklı olarak geçmiş değerlendirme sonuçlarını takip eder ve bunu olasılık modelini oluşturmak için kullanır. Diğer aramalara göre, maliyeti daha azdır ancak diğer arama türleriyle de minimum çalışılabilirlikte arama yapılması, çıkan sonuçların sağlıklı olup olmadığını anlamak için gerekebilir.

Optimal parametreleri bulma süreci maliyetli bir süreçtir ve yoğun derecede bir işçilik gerektirir. Yardımcı uygulamalar belirli yaklaşımlarla parametrelere karar verse de bu parametrelerin optimal olup olmadığı konusu muammadır. Bu sürecin doğru bir şekilde yönetilmesi ve tasarlanması, minimum hataların oluşmasına sebebiyet verecektir çünkü parametrelerin aranacağı aralıkların doğru belirlenememesi modeli yanlış yerlere sürükleyecek hem de maliyeti fazlasıyla artıracaktır. Bu sebeple, erken durdurma gibi bazı geri çağırma işlevleri, bu süreci

fazlasıyla kolaylaştıracaktır. Bu işlev sayesinde izlenmesi istenen bir metrik, modeli iyileştirmeyi bıraktığında eğitim otomatik bir şekilde durdurulacak ve başka aralıklarda parametre aramaya *(trial)* devam edilecektir. Bu da gereksiz maliyetleri önleyecektir. Parametre arama tercih ve aralıkları Tablo 4.11'de verilmiştir. Kod katmanının düğüm sayısı kodlayıcı katmanının düğüm sayısının yarısı, kod çözücü katmanının düğüm sayısı ise kodlayıcı katmanın düğüm sayısına eşit olacaktır. Düğüm seyreltme katmanı, kod kısmından sonra olacak şekilde bu aşamada ilk defa mimariye eklenmiştir. Ayrıca, çıkış katmanının aktivasyon fonksiyonu *sigmoid* olarak belirlendiğinden bu parametreler aramaya konulmamıştır.

Tablo 4.11 Parametre arama aralıkları

Parametre	**Parametre arama aralığı**
Düğüm sayısı	[32, 64, 128, 256, 512]
Parti boyutu	[32, 64, 128]
Öğrenme oranı	[5e-3, 1e-3, 5e-4, 1e-4, 5e-5, 1e-5]
Düğüm seyreltme miktarı	[min=0.0, max=0.9, artış miktarı=0,05]
Optimizasyon algoritması	[SGD, Adam, RMSprop]
Kodlayıcı kat. aktivasyon fonk.	[relu, selu, elu]
Kod kat. aktivasyon fonk.	[relu, selu, elu]
Kod çözücü kat. aktivasyon fonk.	[relu, selu, elu]

Tablo 4.12 Bazı parametre tercihleri

Parametre	**Parametre tercihi**
En iyileme fonksiyonu	BCE
Erken durdurma	doğrulama kaybının min. olmasına göre
Erken durdurma bekleme süresi	3 adım
Parametre arama türü	hiperbant arama

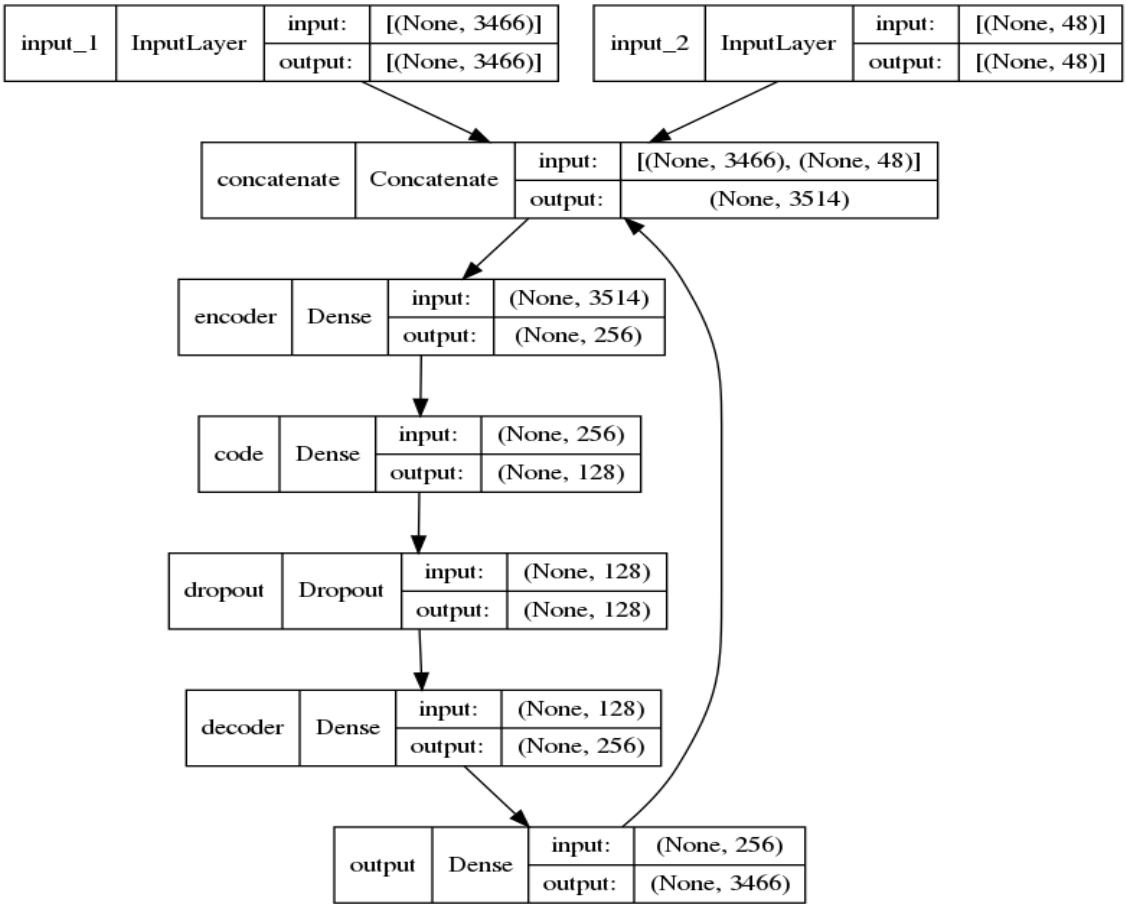
Optimal parametreler belirlendikten sonra, mimarinin parametreleri güncellenerek yeni bir model eğitilir. Model eğitilirken erken durdurma ve yeni bir kavram olarak karşımıza çıkan *model kontrol noktası (model checkpoint)* kullanılarak en iyi skoru veren döngüdeki model ağırlıkları kaydedilir. Normalde, eğitim sonundaki model ağırlıkları kaydedilirken eğitim sonunu bekleyip çıkan sonuçların en iyi ağırlıklar olmaması ihtimali göz önünde bulundurulduğunda, aşırı ya da eksik öğrenme durumlarında eğitimi durdurarak işlevsellik kazandırıp maliyeti önemli ölçüde azaltmaktadır. Model kaydetme aşamasının gerçekleşmesi, optimal parametrelerin seçilip seçilmediği konusundan bağımsızdır. Eğitilen model grafiklerine bakılarak eğitimin nerede durdurulduğuna, eğitim ve doğrulama veri grafiklerine, performans metrik sonuçlarına hep birlikte bakılıp optimal parametrelerin seçilip seçilmediği konusunda bir çıkarım yapılarak bir sonraki aşamaya geçilir. Buna göre, hiperparametre optimizasyonuyla belirlenen en iyi model parametreleri, Tablo 4.13'te gösterilmiştir.

Tablo 4.13 Hiperparametre optimizasyonuyla belirlenen parametreler

Parametre	**Seçilen parametre**
Düğüm sayısı	256
Gizli düğüm sayısı	128
Parti boyutu	32
Öğrenme oranı	5e-5
Düğüm seyreltme miktarı	0.0
Optimizasyon algoritması	adam
Kodlayıcı katmanının aktivasyon fonksiyonu	selu
Kod katmanının aktivasyon fonksiyonu	selu
Kod çözücü katmanının aktivasyon fonksiyonu	selu
En iyi kayıp değeri	,067

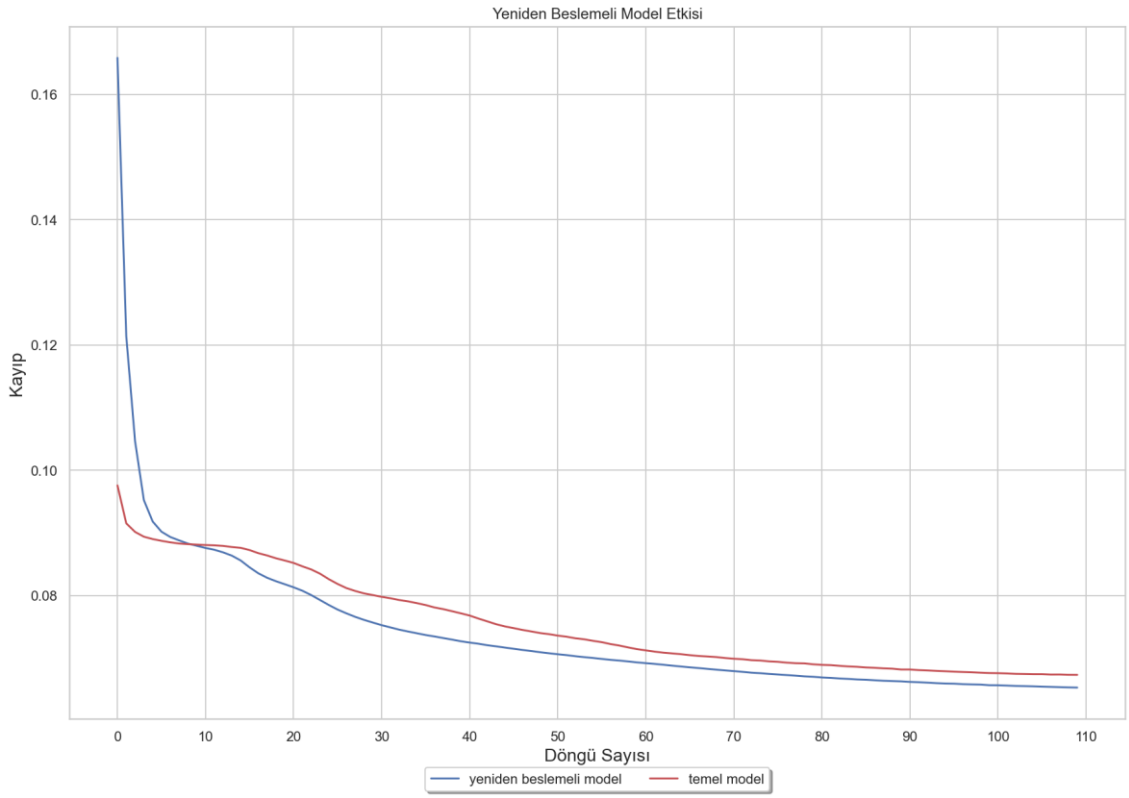
4.6.5 Temel ve Yeniden Beslemeli Modelin Karşılaştırılması

Oluşturulan farklı mimariler birbiriyle karşılaştırılıp yeniden beslemeli algoritma ile eğitilen derin-yarı otokodlayıcı yaklaşımının faydalı olup olmadığı tartışılacaktır. Bu yeni yaklaşımın seyreklik ve soğuk başlangıç problemine katkı sağlayıp sağlamadığına da yer verilecektir. Mimarisi ve parametreleri belirlenen 4 katmanlı model *temel model* olarak adlandırılacaktır. Yeniden beslemeli modelin etkisini görmek için ise temel ve yeniden beslemeli model karşılaştırılacaktır. Karşılaştırmaya başlamadan önce, Şekil 4.9'daki yeniden beslemeli derin yarı-otokodlayıcı mimarisini incelemek faydalı olacaktır.



Şekil 4.9 Yeniden beslemeli derin yarı-otokodlayıcı model mimarisi

Karşılaştırmaya, temel modelin eğitimiyle başlanacaktır. Parametreleri belirlenen model eğitildikten sonra, Şekil 4.10'da görüldüğü üzere bu iki mimarinin performansları karşılaştırılmıştır. Şekil 4.8'deki model mimarisine yeniden besleme adımı da eklenerek Şekil 4.9'daki mimari elde edilmiştir.



Şekil 4.10 Temel ve yeniden beslemeli modelin doğrulama kaybı üzerindeki etkisi

Temel ve yeniden beslemeli modelin performans grafiği, Şekil 4.10'da gösterilmiştir. Ayrıca, elde edilen sonuçlar, birbiriyle karşılaştırılarak Tablo 4.14'de verilmiştir.

Tablo 4.14 Temel ve yeniden beslemeli modelin etkisi

Model/Performans	**BCE**			**AUC**		
	Eğitim	*Doğrulama*	*Test*	*Eğitim*	*Doğrulama*	*Test*
Temel	,0583	,0670	,0628	,9666	,9521	,9519
Yeniden beslemeli	,0568	,0650	,0615	,9684	,9532	,9534

Tablo 4.14'e göre, yeniden beslemeli modelin genel performansı temel modelin genel performansına göre daha başarılıdır. Aralarında büyük bir farklılık gözlenmemesine rağmen model eğitilirken kayıp değerleri oldukça yavaş bir şekilde düşmektedir. Ayrıca, hiperparametre optimizasyonu ile model parametreleri en iyi şekilde belirlenmeye çalışılsa da parametrelerin birbiriyle olan

bütün kombinasyonlarının denenmesi mümkün değildir. Bu sonuçlar gözetilirken eğitim ve doğrulama kaybının grafiklerine aynı anda bakılarak optimal sonuçlar seçilmiştir. Daha farklı parametreler seçildiğinde, çıkan sonuçlar değişkenlik gösterebilir. Belirlenen koşullar ve parametreler altında ortaya konulan sonuçlardır.

Model metriklerinin performans sonuçları gösterildiğine göre, bu metriklerden bağımsız olarak ayrı bir yaklaşımla hesaplanan ve uygulama kısmında da değinilen performans metriklerini hesaplama kısmına geçilebilir.

4.6.6 Performans Sonuçları

Temel ve yeniden beslemeli model eğitilirken her bir döngü sonunda doğrulama verisi üzerinden modele *tahmin (prediction)* yaptırılıp çıkan sonuçlar belirli aşamalardan geçirilerek *MAP@k, NDCG@k, Recall@k ve Precision@k* metrikleri hesaplanmaktaydı. Ayrıca, eğitilmiş olan temel ve yeniden beslemeli temel model üzerinden de test metriği hesaplanarak sonuçlar birbiriyle karşılaştırılacaktır. Karşılaştırılan iki yaklaşımın birbirine göre ne derece iyi performans gösterdiği tartışılacaktır. Elde edilen performans grafikleri üzerinden görselleştirme de yapılacaktır. Bu sayede, her bir döngü sonunda performansların hangi ölçüde azaldığı ya da arttığı tespit edilebilecektir. Bu tespit ile döngü sayısı kaç olduğunda performanslarda bir düşüş ya da artış başladığı hakkında fikir yürütülerek modelin eğitileceği döngü sayısına da karar verilebilir. Doğrulama ve test setleri üzerinden farklı k sayılarına göre hesaplanan performans değerlendirme sonuçları, Tablo 4.15'te gösterilmiştir.

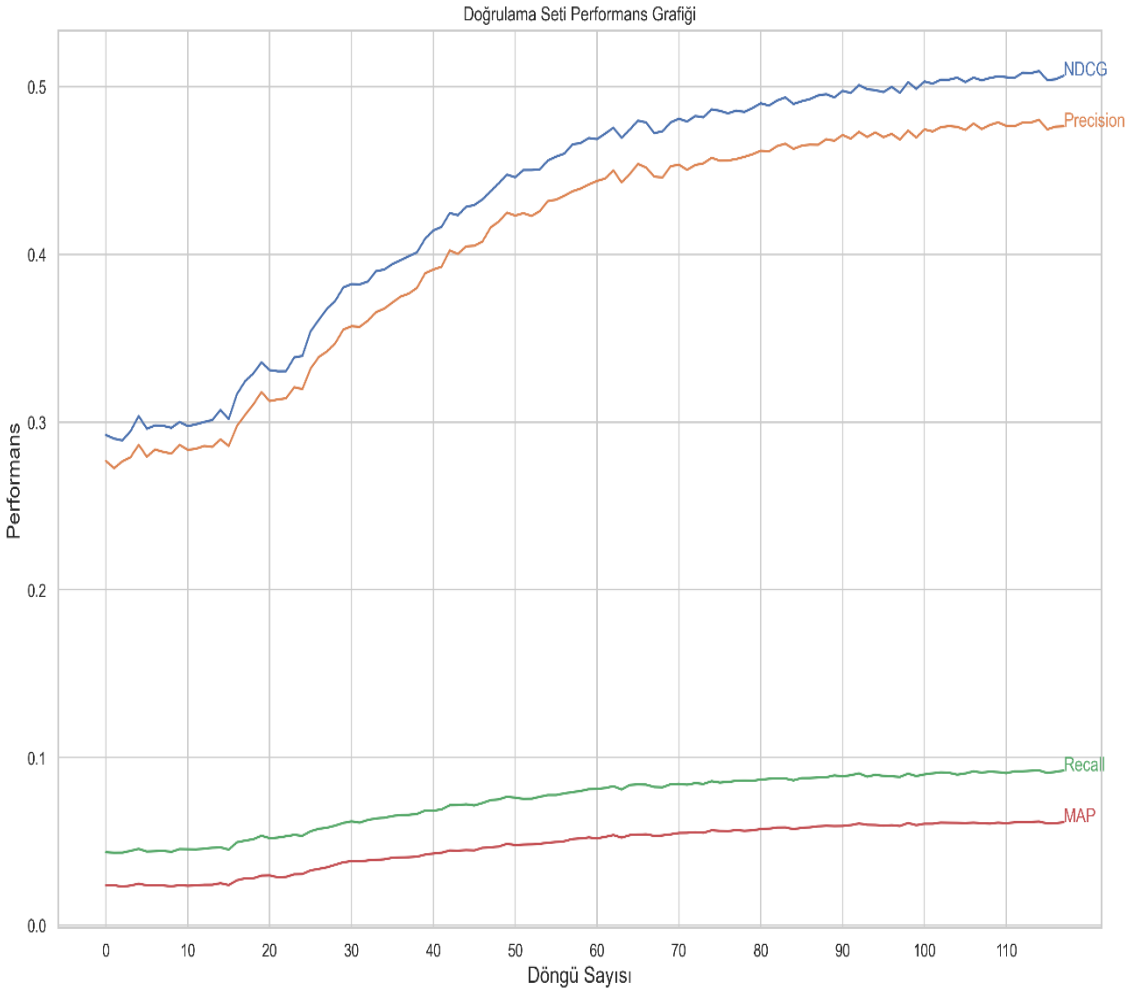
Tablo 4.15 Performans sonuçları

Metrik / Model	Temel		Yeniden Beslemeli	
	Doğrulama	*Test*	*Doğrulama*	*Test*
NDCG@10	,5092	,4887	,5202	,5050
NDCG@20	,4623	,4462	,4724	,4581
NDCG@50	,4330	,4202	,4400	,4263
NDCG@100	,4426	,4350	,4493	,4420
Recall@10	,0923	,0906	,0948	,0935
Recall@20	,1508	,1492	,1543	,1517
Recall@50	,2679	,2711	,2734	,2724
Recall@100	,3874	,3963	,3947	,4009
MAP@10	,0618	,0599	,0637	,0628
MAP@20	,0875	,0851	,0904	,0890
MAP@50	,1292	,1270	,1329	,1315
MAP@100	,1633	,1619	,1681	,1676
Precision@10	,4802	,4614	,4927	,4770
Precision@20	,4158	,4016	,4261	,4116
Precision@50	,3291	,3188	,3339	,3221
Precision@100	,2617	,2531	,2649	,2572

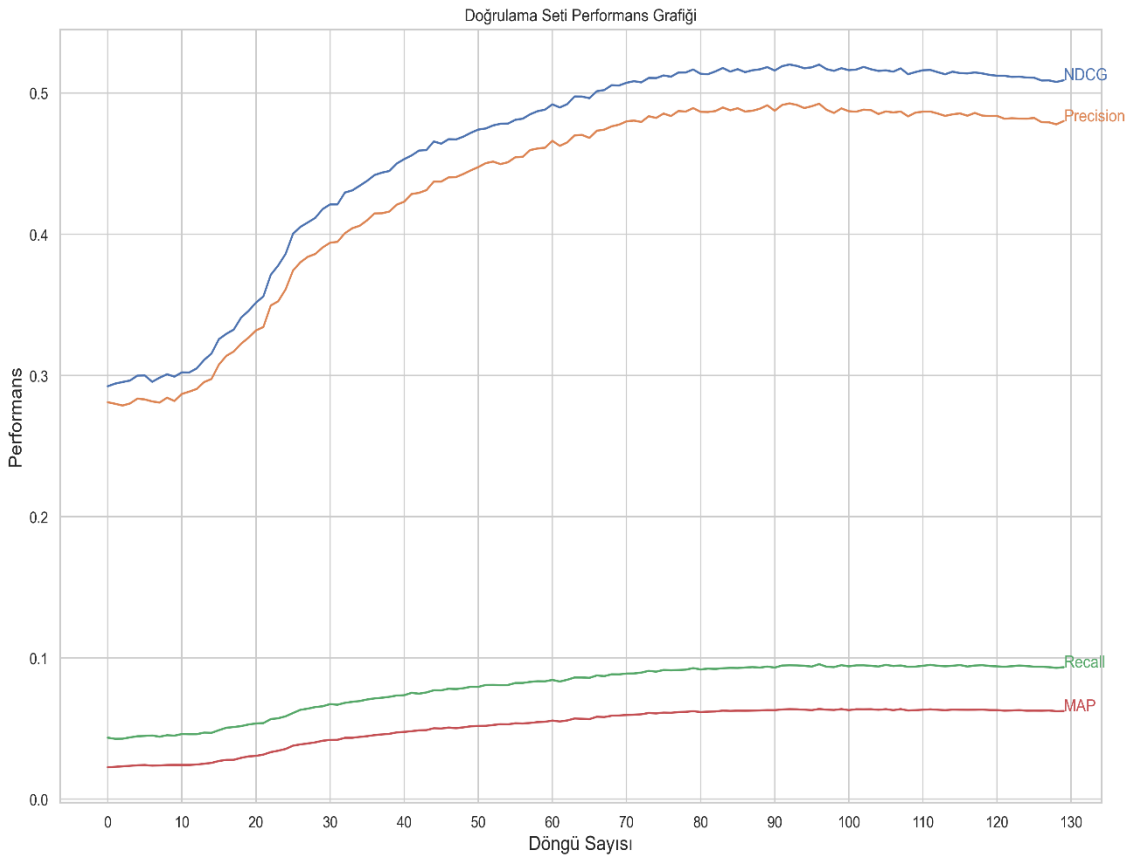
Tablo 4.15'e göre, temel ve yeniden beslemeli modelin performans sonuçlarına göre @k sayısı arttıkça NDCG ve Precision performansı düşmekte Recall ve MAP performansı ise artmaktadır.

Her iki yaklaşımın da sonuçları birbirine çok yakındır ancak yeniden besleme yapılan modelin performansı temel modele göre biraz daha yüksek görünmektedir. MAP ve Recall metriklerinde iyi bir başarı elde etmek diğer metriklere kıyasla oldukça zordur. Bu sebeple, k sayısı arttıkça MAP ve Recall da artma eğilimi göstermektedir. Bu performans karşılaştırmaları, modelin daha önce görmediği öğeler üzerinden model başarısını değerlendirdiği için gerçek dünyaya daha yakındır. Ayrıca, kullanıcının daha önceden etkileşime geçtiği fakat modele gösterilmeyen öğelerin bir kısmı bilindiğinden model performansı kullanıcının geçmişine göre ölçülmektedir.

Temel ve yeniden beslemeli modelin doğrulama verisi üzerindeki sonuçları Tablo 4.10'da paylaşılmıştı. Buna ek olarak k=100 olacak şekilde, iki modelin eğitim boyunca doğrulama kümesi üzerindeki performans grafikleri, Şekil 4.11 ve Şekil 4.12'de gösterilmiştir.



Şekil 4.11 Doğrulama setinin temel model üzerindeki performansı



Şekil 4.12 Doğrulama setinin yeniden beslemeli model üzerindeki performansı

Şekil 4.11'de model eğitim süreci ilerledikçe performans metrikleri de yükselişini sürdürmeye devam etmiştir. Şekil 4.12'de, model eğitim süreci ilerledikçe performans başarıları belli bir noktadan sonra geriye doğru bir düşüşe geçmiştir. Aynı koşullar altında temel model, yeniden beslemeli modelden daha önce aşırı öğrenme reaksiyonu gösterdiğinden model erken durdurma yapmış fakat yeniden beslemeli modelin belli bir süre daha eğitimine devam etmesine karşın performansını daha fazla iyileştirememiştir.

4.7 Modelden Öğe Tahminlerinin Alınması

Model aşamasından sonra kritik bir önem arz eden meseledir. Birçok çalışmada, bu kısım genellikle göz ardı edilir. Hâlbuki öneri vermeyi bekleyen bu eğitilmiş model tüm bu süreç içerisinde elde edilen en önemli öğedir.

Zorlu bir eğitim sürecinden sonra elde edilen modellerden tahmin alınması, sanılanın aksine kolay bir süreç olmaz. Bu süreçte, eğitim verisinin geçtiği aşamalardan doğrulama ve test verisini de geçirmek gerekir.

Ağırlıklarını içinde barındıran eğitilmiş model, öneri(ler) verebilmek için bazı kilit noktalara önem verir. Örneğin model eğitilirken kaç adet girdi veriliyorsa model de hiçbir değişiklik gösterilmeden bu sürecin uygulanmasını ister. Aksi takdirde girdi uyumsuzluklarından dolayı tahmin yapılmasını olanaksız kılar. Modele tahminlemesi için verilen girdi, aynı zamanda bir vektördür ve model de bu girdinin bir vektör olarak beslenmesini ister. Makine öğrenmesinde tahmin edilecek veri, farklı veri tiplerinden oluşsa bile standart bile hâle dönüştürülmesi şarttır. Çoğu modelde bunun bir standardı mevcuttur. Hâttâ özellikle derin ağlar; örneğin veri, bir ölçekleme işlemine tabii tutulmamışsa ya da değişken içerisindeki kategoriler sıralı bir yapıda değilse bunu büyük bir problem olarak gördüğünden bir sonraki aşamaya geçilemez.

Bu tez çalışmasında, belirli yaklaşımlarla belirli modeller arasından seçilen eğitilmiş modellerden kullanıcılara bazı öneriler verilerek modellerin çalışma biçimi de tanıtılacaktır. Kullanıcı ya da öğe yan bilgisinin de modele eklenmesinin soğuk başlangıç problemine karşı bir çözüm olduğundan önceki bölümlerde sıkça bahsedilmişti. Bunu uygulamak gerçekten bir çözüm oluyor mu konusunu tasdik etmek gerekirse hakkında hiçbir bilgisi bulunmayan bir kullanıcıya ne şekilde tahmin yapıldığının anlaşılması gerekir. Kullanıcı hakkında bir bilgi mevcut olduğunda bu bilgiler ışığında yapılan önerilerin değişip değişmediği gözlemlenmelidir.

Otokodlayıcılar, işbirliğine dayalı filtrelemenin gücünden sıkça yararlanan yapılardır. Bu yüzden, hakkında bilgi bulunmayan kullanıcılara öneri yapılırken birçok kullanıcının önceki etkileşimleri model tarafından öğrenilir, ağırlıklandırılır ve bunlardan bir çıkarım yapılmaya çalışılır. İlk sırada yapılan önerilerin de bu sebeple, en çok etkileşim alan öğeler olduğu ya da pek çok kullanıcıyla bir şekilde etkileşime giren öğeler olduğu çıkan sonuçlarda anlaşılmaktadır. Bir kullanıcı hakkında bilgi gelse bile en önde gelen klasik öğelerin önerilmesi, öğelerin olasılıkları değişse bile devam etmektedir. Bu, sıradaki diğer öğelerin de olasılığının değişmemesi demek değildir.

Farklı durumlarda model tarafından yapılan öneriler birbiriyle birkaç örnek üzerinden karşılaştırılacaktır. Bu karşılaştırma sonucunda hangi öğelerin belirli

sıralarını koruduğunu hangi öğelerin sıralamada yerinin değiştiği görülmüş olacaktır. Kullanıcı hakkında bir bilgi bulunmuyorsa modele beslenecek girişlerden kullanıcı-öğe vektörü sadece 0'lardan oluşacaktır. Kullanıcının etkileşime geçtiği öğeler sonradan oluşursa bu vektördeki ilgili yerlere kullanıcının o öğelerle etkileşime geçtiğinin modele anlatılması için 1 olarak doldurulacaktır. Yan bilgiler için de hemen hemen aynı durum söz konusudur. O kullanıcı hakkında herhangi bir yan bilgi mevcutsa vektördeki ilgili yer herhangi bir doldurma (ortalama, medyan, mod) yaklaşımıyla vektör üzerinde doldurularak o kullanıcıya özgü bir vektör olacak ve modele tahminlemesi için verilecektir. Unutulmamalıdır ki modele verilen vektör *(1 x model girdi toplamı)* boyutunda olacaktır.

Hakkında bilgi bulunmayan bir kullanıcı için oluşturulan boş vektör modele verildiğinde, Şekil 4.13'teki öğe isimleriyle birleştirilmiş öğeleri belirli olasılıklarla önermektedir.

	itemID	interaction	item_name	genres	year
0	1	0.185044	Toy Story (1995)	Animation	1989
3	58	0.156073	Postino, Il (The Postman) (1994)	Drama	1989
5	347	0.147907	Bitter Moon (1992)	Drama	1989
6	296	0.143503	Pulp Fiction (1994)	Crime	1989
8	50	0.139261	Usual Suspects, The (1995)	Crime	1989
10	78	0.137714	Crossing Guard, The (1995)	Drama	1989
11	66	0.128466	Lawnmower Man 2: Beyond Cyberspace (1996)	Sci-Fi	1989
13	34	0.126071	Babe (1995)	Children's	1989
16	184	0.124571	Nadja (1994)	Drama	1989
17	18	0.124386	Four Rooms (1995)	Thriller	1989
18	254	0.122697	Jefferson in Paris (1995)	Drama	1989
19	131	0.120839	Frankie Starlight (1995)	Drama	1989
21	41	0.116423	Richard III (1995)	Drama	1989
23	743	0.114872	Spy Hard (1996)	Comedy	1989
24	20	0.114302	Money Train (1995)	Action	1989
25	64	0.114214	Two if by Sea (1996)	Comedy	1989
27	235	0.113024	Ed Wood (1994)	Comedy	1989
29	177	0.111791	Lord of Illusions (1995)	Horror	1989
30	37	0.111452	Across the Sea of Time (1995)	Documentary	1989
31	21	0.110720	Get Shorty (1995)	Action	1989

Şekil 4.13 Model tahmin sonuçları

Şekil 4.13'te görüldüğü üzere, modele bir bilgi verilmeden öneri yapması istenildiğinde, işbirliğine dayalı yaklaşımla öğrendiği öğeleri belirli olasılıklarda önermektedir. Önerilen ilk öğe pek çok kullanıcının etkileşime geçtiği bir öğe özelliği taşımaktadır.

5
SONUÇ VE ÖNERİLER

Tez çalışmasının amacı, işbirliğine dayalı filtreleme yönteminde ortaya çıkan veri seyrekliği ve soğuk başlangıç problemini azaltmaktır. Bu problemleri çözmek için derin yarı-otokodlayıcı ağ mimarisi geliştirilmiş ve ağın eğitilmesi için iteratif bir yaklaşım kullanılmıştır. Tez çalışması, literatürdeki araştırmalardan farklı olarak soğuk başlangıç problemi için hem öğe hem de kullanıcı yan bilgilerini aynı anda modele dâhil etmektedir. Eklenen yan bilgiler ile soğuk başlangıç problemi çözülürken, veri seyreklik problemini azaltmak için geliştirilen mimariye yeniden besleme yaklaşımı uygulayarak eğitim esnasında çıktıların modele yeniden beslenmesini sağlanmıştır.

Yapılan deneysel çalışmalarda; deneysel ve hiperparametre yöntemleriyle mimarinin parametreleri belirlenmiş, model mimarisi oluşturulmuştur. Elde edilen 4 katmanlı derin yarı-otokodlayıcı mimarisi temel model olarak adlandırılmış ve yeniden besleme yaklaşımı uygulanan mimariyle performansları karşılaştırılmıştır. Elde edilen nihai model üzerinden NDCG, MAP, Recall ve Precision metriği kullanılarak k=10, 20, 50, 100 üzerinden bir skorlama yapılmıştır. Her bir k değerine ait skor tablo halinde gösterilmiştir. k sayısının artmasının NDCG ve Precision performansını düşürdüğü, Recall ve MAP performansını ise artırdığı görülmüştür. Ayrıca, eğitim esnasında doğrulama verisi üzerinden hesaplanan performans metriklerinin grafiğiyle her bir döngü sonunda elde edilen performans skorlarını da göstermektedir.

Elde edilen model üzerinden öğe tahminlerinin alınması kritik bir önem arz etmektedir. Birçok çalışmada bu kısım doğrudan geçilebilmektedir. Bu sebeple, bu model üzerinden hakkında bir bilgisi bulunmayan kullanıcılar için öğe tahminleri alınarak modelin tahminleme yapması sağlanmıştır. İşbirliğine dayalı yaklaşım olmasından ötürü modelden çıkan önerilerin bütün kullanıcıların ortak bir paydada bir şekil buluştuğu öğeler olması beklenen bir gerçektir ancak belirli bir sıralamadan sonra önerilerin değiştiği de saptanmıştır.

KAYNAKÇA

[1] D.C. Wilson, B. Smyth and D.O'Sullivan, "Sparsity Reduction in Collaborative Recommendation: A Case-Based Approach", International Journal of Pattern Recognition and Artificial Intelligence vol. 17, no. 5, pp. 863–884, 2003.

[2] F. Strub and J. Mary, “Collaborative filtering with stacked denoising autoencoders and sparse inputs”, NIPS Workshop on Machine Learning for eCommerce, Canada, Dec 2015.

[3] Y. Ouyang, W. Liu, W. Rong and Z. Xiong, “Autoencoder-based collaborative filtering”, ICONIP 2014, Part II, LNCS 8836, pp. 284–291, 2014.

[4] S. Sedhain, A. K. Menon, S. Sanner and L. Xie, “AutoRec: Autoencoders meet collaborative filtering”, WWW 2015 Companion, Italy, May 18–22, 2015.

[5] R. J. Mooney and L. Roy, “Content-based book recommending using learning for text categorization”, Available: ArXiv:cs.DL/9902011, 2000.

[6] D. Ferreira, S. Silva, A. Abelha and J. Machado, “Recommendation system using autoencoders”, Appl. Sci. 10(16), 5510, 2020.

[7] M. Hasan, T. Hasan, S. Reza, N. Akonda, S. H. Khan and M. A. Uddin, “Comprehensive collaborative filtering approach using autoencoder in recommender system”, ICCAI’19, Indonesia, 2019.

[8] P. S. Haghighi, O. Seton and O. Nasraoui, “An explainable autoencoder for collaborative filtering recommendation”, Faculty Scholarship, 429, 2019.

[9] F. Strub, R. Gaudel and J. Mary, “Hybrid recommender system based autoencoders”, DLRS ’16, USA, 2016.

[10] S. Zhang, L. Yao, X. Xu, S. Wang and L. Zhu, “Hybrid collaborative recommendation via semi-autoencoder”, International Conference on Neural Information Processing, Springer, Cham, 2017.

[11] X. Zhang, J. Zhong and K. Liu, “Wasserstein autoencoders for collaborative filtering”, Neural Comput & Applic, 2020.

[12] O. Kuchaiev and B. Ginsburg, “Training deep autoencoders for collaborative filtering”, Available: arXiv:1708.01715, 2017.

[13] G. Adomavicius and Y. Kwon, “Toward more diverse recommendations: Item re-ranking methods for recommender system”, WTS, 2009.

[14] M. Wang, M. Gong, X. Zheng and K. Zhang, “Modeling dynamic missingness of implicit feedback for recommendation”, 32nd Conference on Neural Information Processing Systems (NeurIPS), Canada, 2018.

[15] “Öneri Sistemleri.” Available: https://yapayzeka.itu.edu.tr/arastirma/oneri-sistemleri

[16] L. N. Tondji, “Web recommender system for job seeking and recruiting”, Partial Fulfillment of a Masters II at AIMS, Senegal, 2018.

[17] J. S. Breese, D. Heckerman and C. Kadie, “Empirical analysis of predictive algorithms for collaborative Filtering”, In: Proceedings of the fourteenth conference on Uncertainty in Artificial Intelligence, UAI’98, San Francisco, CA, USA, pp. 43–52, 1998.

[18] F. Ricci, L. Rokach and B. Shapira, “Recommender systems handbook”, Springer, pp. 1–34, 2015.

[19] N. R. Pinnapareddy, "Deep Learning based Recommendation Systems”, Master's Projects. 644, 2018.

[20] A. N. Nikolakopoulos, X. Ning, C. Desrosiers and G. Karypis, ”Trust your neighbors: A comprehensive survey of neighborhood-based methods for recommender systems”, Available: arXiv cs.IR/2109.04584, 2021.

[21] M. Deshpande and G. Karypis, “Item-based top-N recommendation algorithms”, ACM Transactions on Information Systems, vol. 22, no. 1, pp. 143-177, 2004.

[22] B. Zhu, “Towards reliability in collaborative filtering recommender systems”, Ph.D. dissertion, Universidad Politecnica de Madrid, Spain, 2018.

[23] P. Koskela, “Comparing ranking-based Collaborative filtering algorihtms to a rating-based alternative in recommender systems context”, M.S. thesis, University of Jyvaskyla, Finland, 2017.

[24] M. Bdiri, “Deep learning based context-aware recommender system”, M.S. thesis, Universite de Tunis Institut Superieur de Gestion, Tunis, 2018.

[25] C. C. Aggarwal, “Recommender Systems The Textbook”, Springer, pp. 14-15, 2016.

[26] P. Covington, J. Adams and E. Sargin,” Deep neural networks for youtube recommendations”, Proceedings of the 10th ACM conference on recommender systems, ACM, pp. 191–198, 2016.

[27] Muhammet Çakır, “Deep Learning Based Hybrid Recommender System. M.S. thesis, Istanbul Technical University Department of Computer Engineering, Istanbul, 2019.

[28] R. Burke, “Hybrid recommender systems: survey and experiments”, User Modeling and User-Adapted Interaction, 2002.

[29] R. Banik, “Hands-on recommendation systems with Python”, Packt, pp.13-14, 2018.

[30] X. Su and T. M. Khoshgoftaar, “A Survey of Collaborative Filtering Techniques”, Hindawi Publishing Corporation Adcances in Artificial Intelligence, vol. 2009, no. 421421, 2009.

[31] M. B. Ali, “Use of dropouts and sparsity for regularization of autoencoders in deep neural networks” M.S. thesis, The Graduate School of Engineering and science of Bilkent University in Electrical and Electronics Engineering, Ankara, 2015.

[32] X. Glorot, A. Bordes, and Y. Bengio, “Deep sparse rectifier networsks”, AISTATS, vol. 15, pp. 315–323, 2011.

[33] G. Strang, “Introduction to Linear Algebra”, 5th Edition, pp. 465, 2016.

[34] F. Ricci, L. Rokach and B. Shapira, “Practical recommender systems handbook”, Springer, 2015.

[35] H. Al-bashiri, M. Abdulgabber, A. Romli, F. Hujainah, “Collaborative Filtering Recommender System: Overview and Challenges”, Journal of Computational and Theoretical Nanoscience, vol. 23, pp. 9045-9049, 2017.

[36] L. Lerche, “Using Implicit Feedback for Recommender Systems: Characteristics, Applications, and Challenges”, Ph.D. dissertion, Dortmund University of Technology Faculty of computer science, 2016.

[37] S. Cao, N. Yang, Z. Liu, “Online news recommender based on stacked auto-encoder”, In Proceedings of the 2017 IEEE/ACIS 16th International Conference on Computer and Information Science (ICIS), Wuhan, China, 24–26 May, 2017, pp. 721–726.

[38] D. Rumelhart and J. McClelland, “Parallel distributed processing: explorations in the microstructure of cognition”. vol. 1 foundations, 1986.

[39] G. Ian and B. Yoshua, “Deep learning”, MIT Press, 2016.

[40] G. Hinton and R. Zemel, “Autoencoders, Minimum Description Length and Helmholtz Free Energy”, In Advances in Neural Information Processing Systems 6, pp. 3–10. 1994.

[41] Y. Zhou, D. Arpit, I. Nwogu and V. Govindaraju, “Is joint training better for deep auto-encoders?”, Available: arXiv:1405.1380, 2014.

[42] G.E. Hinton and R.R. Salakhutdinov, “Reducing the dimensionality of data with neural networks”, S*cience*, *313*(5786), pp. 504-507, 2006.

[43] Y. Özkan, “Uygulamalı Derin Öğrenme”, Papatya Bilim Akademik Yayınevi, 2021.

[44]“Autoencoders.”, pp. 499-523, Available: https://www.deeplearningbook.org/contents/autoencoders.html

[45] P. Golik, P. Doetsch and H. Ney, “Cross-entropy vs. squared error training: a theoretical and experimental comparison” in INTERSPEECH, pp. 1756 1760, 2013.

[46] M. B. Ali, “Use of dropouts and sparsity for regularization of autoencoders in deep neural networks”, M.S. thesis, The Graduate School of Engineering and Science of Bilkent University, 2015.

[47] N. Srivastava, G. Hinton, A. Krizhevsky, I. Sutskever, and R. Salakhutdinov, “Dropout: A simple way to prevent neural networks from overfitting”, The Journal of Machine Learning Research, vol. 15, no. 1, pp. 1929–1958, 2014.

[48] H. Akaike, "Information theory and an extension of the maximum likelihood principle" in Selected papers of hirotugu akaike, Springer, pp. 199-213, 1998.

[49] C. M. Hurvich and C. L. Tsai, "Regression and time series model selection in small samples" Biometrika, vol. 76, no. 2, pp. 297-307, 1989.

[50] G. Schwarz, "The annals of statistics", Estimating the dimension of a model, vol. 6, pp. 461-464, 1978.

[51] Seda T. “Yapay Sinir Ağlari İle Türkiye’nin Uzun Dönem Elektrik Talep Tahmini”, Yüksek Lisans Tezi, Yıldız Teknik Üniversitesi İstatistik ABD, 2019.

[52] X. Glorot, A. Bordes and Y. Bengio, “Deep sparse rectifier neural networks”, 14th International Conference on Artificial Intelligence and Statistics, pp. 315–323, 2011.

[53] A. S. Özbaygın, “Hayvan Seslerini Ayrıştırmak İçin Modifiye Edilmiş Bir Otokodlayıcı Mimarisi Yaklaşımı”, Yüksek Lisans Tezi, KTO Karatay Üniversitesi Elektrik ve Bilgisayar Mühendisliği ABD, 2019.

[54] D. Kingma and j. Ba, “Adam: A method for stochastic optimization”, Available: arXiv:1412.6980, 2014.

[55] A. B. Kouki, “Recommender System Performance Evaluation and Prediction: An Information Retrieval Perspective”, Ph.D. dissertion, Universidad Autónoma de Madrid Escuela Politécnica Superior Departamento de Ingeniería Informática, 2012.

[56] TensorFlow: Large-Scale Machine Learning on Heterogeneous Systems (2015), Available: https://www.tensorflow.org/

[57] GroupLens Research Project. University of Minnesota, 2000, Mevcut: https://grouplens.org/datasets/movielens/1m/

[58] D. Liang, R. G. Krishnan, M. D. Hoffman and T. Jebara. “Variational Autoencoders for Collaborative Filtering”, In Proceedings of The 2018 Web Conference (WWW 2018), ACM, New York, NY, USA, pp. 10, 2018.

[59] Deep Learning with Keras, Packt Publishing Ltd. (2017), Available: https://keras.io/keras_tuner/

TEZDEN ÜRETİLMİŞ YAYINLAR

Konferans Bildirileri

1. U. Durdu ve G. Kemalbay, “Öneri Sistemlerinde Otokodlayıcı Tabanlı Derin Öğrenme Yaklaşımı”, 5.Uluslararası Mühendislik ve Teknoloji Yönetimi Kongresi, Bildiri Kitabı, sayfa 669, 24-24 Nisan 2021, İstanbul, Türkiye.

Projeler

1. U. Durdu (Araştırmacı) ve G. Kemalbay (Yürütücü), “Öneri Sistemlerinde Otokodlayıcı Tabanlı Derin Öğrenme Yaklaşımı”, Yıldız Teknik Üniversitesi, FYL-2021-4281 no’lu Bilimsel Araştırma Projesi.