T.C.
BURSA TEKNİK ÜNİVERSİTESİ
LİSANSÜSTÜ EĞİTİM ENSTİTÜSÜ

ÖNERİ SİSTEMLERİNDEKİ VERİ SEYREKLİĞİ PROBLEMİNE DERİN ÖĞRENME YAKLAŞIMI

YÜKSEK LİSANS TEZİ

Ecem BÖLÜK

Bilgisayar Mühendisliği Anabilim Dalı

OCAK 2023

T.C.
BURSA TEKNİK ÜNİVERSİTESİ
LİSANSÜSTÜ EĞİTİM ENSTİTÜSÜ

ÖNERİ SİSTEMLERİNDEKİ VERİ SEYREKLİĞİ PROBLEMİNE DERİN ÖĞRENME YAKLAŞIMI

YÜKSEK LİSANS TEZİ

Öğrenci Ecem BÖLÜK
(19376482014)

Bilgisayar Mühendisliği Anabilim Dalı

Tez Danışmanı: Dr. Öğr Üyesi Mustafa Özgür CİNGİZ

OCAK 2023

BTÜ, Lisansüstü Eğitim Enstitüsü'nün 19376482014 numaralı Yüksek Lisans Öğrencisi Ecem BÖLÜK, ilgili yönetmeliklerin belirlediği gerekli tüm şartları yerine getirdikten sonra hazırladığı "ÖNERİ SİSTEMLERİNDE VERİ SEYREKLİĞİ PROBLEMİNE DERİN ÖĞRENME YAKLAŞIMI" başlıklı tezini aşağıda imzaları olan jüri önünde başarı ile sunmuştur.

Tez Danışmanı : **Dr. Öğr. Üyesi Mustafa Özgür CİNGİZ** .....
Bursa Teknik Üniversitesi

Jüri Üyeleri : **Prof. Dr. Turgay Tugay BİLGİN** .....
Bursa Teknik Üniversitesi

Doç. Dr. Murtaza CİCİOĞLU .....
Bursa Uludağ Üniversitesi

Teslim Tarihi :
Savunma Tarihi : 20 Ocak 2023

20.04.2016 tarihli Resmi Gazete'de yayımlanan Lisansüstü Eğitim ve Öğretim Yönetmeliğinin 9/2 ve 22/2 maddeleri gereğince; Bu Lisansüstü teze, Bursa Teknik Üniversitesi'nin abonesi olduğu intihal yazılım programı kullanılarak Lisansüstü Eğitim Enstitüsü'nün belirlemiş olduğu ölçütlere uygun rapor alınmıştır.

İNTİHAL BEYANI

Bu tezde görsel, işitsel ve yazılı biçimde sunulan tüm bilgi ve sonuçların akademik ve etik kurallara uyularak tarafımdan elde edildiğini, tez içinde yer alan ancak bu çalışmaya özgü olmayan tüm sonuç ve bilgileri tezde kaynak göstererek belgelediğimi, aksinin ortaya çıkması durumunda her türlü yasal sonucu kabul ettiğimi beyan ederim.

Öğrencinin Adı Soyadı: Ecem BÖLÜK

İmzası :

Aileme,

ÖNSÖZ

Başta bu tez çalışmam olmak üzere; yüksek lisans sürecim boyunca bana her konuda bilgisini ve desteğini esirgemeyen, motive eden ve gittiğim yolda beni yönlendiren değerli tez danışmanım Dr. Öğr. Üyesi Mustafa Özgür CİNGİZ hocama sonsuz teşekkürlerimi sunuyorum.

Ocak 2023

Ecem BÖLÜK
(Veri Analisti)

İÇİNDEKİLER

Sayfa

KISALTMALAR

AE : Temel Otomatik Kodlayıcılar
DAE : Gürültü Giderici Otomatik Kodlayıcılar
GAN : Üretici Çekişmeli Ağlar
IRGAN : Bilgi Erişimi Üretici Çekişmeli Ağlar
KL : Kullback Leibler Uzaklığı
MAE : Ortalama Mutlak Hata
RBM : Kısıtlı Boltzmann Makineleri
RMSE : Kök Ortalama Kare Hatası
SAE : Seyrek Otomatik Kodlayıcılar
VAE : Varyasyonel Otomatik Kodlayıcılar

SEMBOLLER

b	**:** Bias değeri
W	**:** Model ağırlıkları
x	**:** Gerçek veri
x'	**:** Model ile oluşturulan veri
z	**:** Gizli katman
Z	**:** Bölme Fonksiyonu
μ	**:** Ortalama
α	**:** Aktivasyon fonksiyonları
σ	**:** Standart sapma
ε	**:** Yeniden parametreleştirme değişkeni
⊗	**:** İç çarpım
⊙	**:** Skaler çarpım

ÇİZELGE LİSTESİ

Sayfa

ŞEKİL LİSTESİ

Sayfa

ÖNERİ SİSTEMLERİNDE VERİ SEYREKLİĞİ PROBLEMİNE DERİN ÖĞRENME YAKLAŞIMI

ÖZET

Öneri sistemleri ile ilgili çalışmalar hem akademide hem de endüstri de oldukça yoğun bir şekilde ilgi görmektedir. Öneri sistemleri kullanıcıların geçmişteki tercihlerinden hareketle gelecekteki tercihlerini tahmin eden sistemlerdir. Bilgi teknolojilerinin gelişmesi ile birlikte şirketler müşterilerden çok fazla veri sağlarlar. Bu verileri müşteri deneyimlerini iyileştirme ve satışlarını arttırmak için öneri sistemlerinde kullanırlar. Fakat kullanıcılar her zaman tercihlerini bu şirketlerle paylaşmayabilir. Bu durum, öneri sistemleri tasarlanırken karşılaşılan en büyük sorunlardan biri olan veri seyrekliğine neden olur.

Veri seyrekliği kullanıcı öğe matrisinde derecelendirme bilgisinin çok az olduğu durumlarda gerçekleşir. Bu durum öneri sistemleri için çok seyrek kullanıcı öğe matrisine neden olur. Veri seyrekliği problemini çözmek için çalışmalarda meta verilerden yararlanılarak seyrek kullanıcı öğe matrisi yeniden yapılandırılmaya çalışılmıştır. Fakat bu çözüm modelleri daha karmaşık hale getirmekte ve veri gizliliği nedeniyle her zaman meta verilere ulaşmak pek mümkün olmamaktadır. Derin öğrenme algoritmalarından otomatik kodlayıcılar, kısıtlı Boltzmann makineleri ve üretici çekişmeli ağlar sadece seyrek kullanıcı öğe matrisini kullanarak seyrek kullanıcı öğe matrisi içerisindeki verilerden öğrendiği iç görülerden hareketle bu seyrek matrisi yeniden doldurarak veri seyrekliği probleminin çözülmesini sağlar.

Bu tezde derin öğrenme algoritmalarından Temel Otomatik Kodlayıcı, Gürültü Giderici Otomatik Kodlayıcı, Seyrek Otomatik Kodlayıcı ve Varyasyonel Otomatik Kodlayıcı olmak üzere dört farklı otomatik kodlayıcı, Kısıtlı Boltzmann Makineleri ve Üretici Çekişmeli Ağlar olmak üzere altı farklı algoritma kullanılarak veri seyrekliğine karşı performansları karşılaştırılmıştır. Veri seti olarak %27 ve %74,5 seyrek veri içeren üç farklı Jester veri setleri ve %93,6 ve %95,4 seyrek veri içeren iki farklı Movielens veri seti olmak üzere beş farklı seyrek kullanıcı öğe matrisi içeren veri setleri kullanılmıştır. Jester veri setleri kullanılarak geliştirilen öğe tabanlı modeller, kullanıcı tabanlı modellere göre daha fazla başarı gösterdiği gözlemlenmiştir. Jester veri seti üzerinde yapılan analizlere göre öğe tabanlı modellerde en başarılı algoritma Seyrek Otomatik Kodlayıcılar olurken, kullanıcı tabanlı modellerde en başarılı algoritmanın Temel Otomatik Kodlayıcılar olduğu gözlemlenmiştir. Movielens veri setlerinde ise kullanıcı tabanlı ve öğe tabanlı modellerin performansları algoritmaya göre değişiklik gösterse de çok büyük bir performans farkı olmadığı gözlemlenmiştir. Movielens veri setinde her iki modelde ve değerlendirme metriğinde en başarılı performans gösteren algoritmanın Temel Otomatik Kodlayıcılar olduğu gözlemlenmiştir.

Anahtar kelimeler: Öneri Sistemleri, Derin Öğrenme, Veri Seyrekliği

A DEEP LEARNING APPROACH TO THE PROBLEM OF DATA SPARSITY IN RECOMMENDATION SYSTEMS

SUMMARY

Studies on recommendation systems are of great interest both in academia and industry. Recommendation systems are systems that predict future preferences of users based on their past preferences. With the development of information technologies, companies provide a lot of data from customers. They use this data in their recommendation systems to improve their customer experience and increase their sales. However, users may not always share their preferences with these companies. This causes data sparsity, which is one of the biggest problems when designing recommender systems.

Data sparsity occurs when there is very little rating information in the user item matrix. This causes a very sparse user item matrix for recommendation systems. In order to solve the data sparsity problem, the sparse user item matrix was tried to be restructured by using metadata in the studies. However, this solution complicates the models and it is not always possible to access metadata due to data privacy. Autoencoders, Restricted Boltzmann Machines and Generative Adversarial Networks, which are deep learning algorithms, use only the sparse user item matrix and re-populate this sparse matrix based on the insights learned from the data in the sparse user item matrix, thereby solving the data sparsity problem.

In this thesis, performances of deep learning algorithms are compared against data sparsity by using four different autoencoders, namely Vanilla Autoencoder, Denoising Autoencoder, Sparse Autoencoder and Variational Autoencoder, six different algorithms, Restricted Boltzmann Machines and Generative Adversarial Networks. Three different Jester datasets containing 27% and 74.5% sparse data, and two different Movielens datasets containing 93.6% and 95.4% sparse data, five different sparse user item matrices were used as datasets. Item-based models developed using jester datasets have been observed to be more successful than user-based models. According to the analyzes made on the Jester dataset, it was observed that the most successful algorithm in item-based models was Sparse Autoencoders, while the most successful algorithm in user-based models was Vanilla Autoencoders. In Movielens datasets, although the performances of user-based and item-based models vary according to the algorithm, it has been observed that there is not a big performance difference. In the Movielens dataset, it was observed that the most successful algorithm in both models and evaluation metric was Vanilla Autoencoders.

Keywords: Recommendation Systems, Deep Learning, Data Sparsity

1. GİRİŞ

Bilgi teknolojilerinin gelişmesiyle birlikte kullanıcılar şirketlere büyük miktarlarda veri sağlamaktadır. Şirketler bu verileri çeşitli istatiksel ve makine öğrenmesi yöntemlerini kullanarak satışları arttırmayı amaçlarlar. Şirketlerin bu alanda en çok kullandığı yöntemlerden biri öneri sistemleridir. Öneri sistemleri kullanıcıların geçmiş tercih ve davranışlarına göre gelecekteki tercihlerini tahmin eden sistemlerdir. Sosyal medya uygulamaları ve e-ticaret şirketleri, kullanıcı deneyimini geliştirmek için öneri sistemlerini kullanır. Örneğin Netflix'te izlenen filmlerin yüzde 80'i ile Youtube'da izlenen videoların %60'ı öneri sistemleri sayesinde kullanıcılar tarafından tercih edilmiştir [1]. Amazon, kârının %35'ini öneri algoritmaları ile temin etmektedir [2].

Öneri sistemleri genellikle üç farklı yaklaşım kullanılarak geliştirilmiştir. Bu yaklaşımlar, işbirlikçi filtreleme, içerik tabanlı yaklaşım ve her iki yaklaşımı birleştiren hibrit öneri sistemleridir. İşbirlikçi filtreleme, kullanıcıların veya öğelerin tercihlerini benzer kullanıcılar veya öğelerle karşılaştırarak tahmin etme yöntemidir. Kullanıcı tabanlı işbirlikçi filtreleme ve öğe tabanlı iş birlikçi filtreleme olarak iki farklı modelde tasarlanır. Kullanıcı tabanlı işbirlikçi filtreleme kullanıcıların birbiri arasındaki benzerlik ilişkisine göre önerileri gerçekleştirir. Öğe tabanlı iş birlikçi filtreleme ise öğelerin birbiri arasındaki benzerlik ilişkisine göre önerileri gerçekleştirir. İçerik temelli yaklaşım, benzer içeriğe sahip bir öğenin, öğe içeriğine göre tavsiye edilmesidir. Bu model tasarlanırken öğenin kategorisi, ismi, öğeye yapılan yorumlar gibi öğe içerikleri kullanılır. Hibrit öneri sistemi ise, işbirlikçi filtreleme ile içerik temelli bir yaklaşımı birleştiren bir yöntemdir.

Öneri sistemleri, aşırı bilgi yükleme sorunuyla başa çıkmak için etkili yollar sağlasa da farklı bir zorlukla da karşı karşıyadır. Bu zorluk, veri seyrekliğidir. Veri seyrekliği sorunu, genellikle sistemdeki yetersiz bilgi nedeniyle ortaya çıkan ve işbirlikçi filtreleme yöntemlerini etkileyen bir sorundur. Bu sorun, veri tabanındaki kullanıcı tarafından derecelendirilen öğelerin toplam sayısının az olması nedeniyle kullanıcı

öğe matrisinin seyrek bir matris olmasıyla sonuçlanır. Aynı zamanda başarılı komşular bulamamaya ve nihayetinde kötü bir tavsiye sürecine yol açar [3].

Öneri sistemi algoritmaları, geleneksel modelleri kullanarak seyrek matrisi yeniden oluşturmaya çalışır [4-6]. Ancak bu modeller aşırı basitleştirilmiştir ve modelleme ifadelerini büyük ölçüde sınırlayacaktır. Ayrıca geleneksel öneri sistemi modelleri karmaşık yapıları yakalayamazlar. Derin öğrenme algoritmaları, seyrek matrisi geleneksel öneri sistemi modellerine göre yeniden yapılandırmada daha başarılıdır [7,8]. Derin öğrenme algoritmaları, aktivasyon fonksiyonlarını (sigmoid, relu vb.) kullanılarak lineer olmayan yapıları kolaylıkla modeller. Bu özellik, karmaşık etkileşim kalıplarıyla başa çıkmayı mümkün kılar ve kullanıcının tercihini tam olarak yansıtır [1].

Derin öğrenme algoritmaları, öneri sistemlerinde meta veri kullanımı ve seyrek kullanıcı öğe matrisi kullanımı şeklinde iki farklı yöntem kullanılarak tasarlanır. Meta veri, öneri sistemlerinde kullanıcıların ya da öğelerin oluşturduğu bilgilerdir. Örnek olarak bir film öneri sistemini düşünürsek, film türleri, yayın tarihi, yönetmeni gibi bilgiler meta veriler olarak tanımlanır. Meta veri kullanımında, derin öğrenme algoritmaları meta veri üzerinden yararlı bilgileri öğrenerek, seyrek kullanıcı-öğe matrisini yeniden oluşturur [9-12]. Meta veriler, öneri sistemlerinin doğruluğunu arttırmak için yaygın olarak kullanılsa da modellerin daha karmaşık olmasına neden olur. Ayrıca veri gizliliği nedeniyle her zaman meta verileri kullanmak mümkün olmaz. Derin öğrenme algoritmaları sadece kullanıcı-öğe matrisini kullanarak da etkili temsilleri öğrenir.

1.1 Öneri Sistemleri

Öneri sistemleri kullanıcıların geçmiş tercih ve davranışlarına göre gelecekteki tercihlerini tahmin eden sistemlerdir. Netflix, Youtube ve Amazon gibi şirketler kullanıcı deneyimi geliştirmek için öneri sistemlerinden yararlanmaktadır [1,2]. Şirketler, öneri sistemlerini geliştirmek için kullanıcıların geçmiş aktivitelerinden faydalanırlar. Öneri sistemlerinin amacı, kullanıcılara daha önce tecrübe etmedikleri yeni şeyleri önermektir.

Öneri sistemlerinin hem kullanıcılara hem de şirketlere önemli faydaları bulunmaktadır. Bunlardan ilki öneri sistemleri kullanıcıların geçmiş aktivitelerine

göre öneriler sunması sebebiyle, kullanıcılara kişiselleştirilmiş bir deneyim sunar. İkinci bir fayda olarak öneri sistemleri ilgili uygulamaya iyi bir trafik ve gelir kaynağı oluşturur. Böylelikle şirketlere daha az pazarlama maliyeti ile tanınabilirliği arttırır. Üçüncü fayda ise öneri sistemlerinin sunduğu doğru öneriler ile kullanıcının hiç keşfedemeyeceği ürün veya hizmeti fark etmesini sağlar. Kullanıcıların farklı ürün veya hizmeti uğraşmadan keşfetmesi sağlanarak müşteri memnuniyeti arttırılmış olur.

1.1.1 Öneri sistemi tarihçesi

Öneri sistemlerinin temelleri ilk olarak İçerik tabanlı öneri sistemlerinin temeli 1920'lerde Emanuel Goldberg'in selüloit bantlarda depolanan belgeleri arayarak otomatik olarak bulmaya çalışan bir istatistik makinesi geliştirmesiyle atılmıştır [16].

İçerik tabanlı öneri sistemleri için yapılan başka bir çalışma ise 1960'larda, Salton çevresinde toplanan Cornwall Üniversitesi'ndeki bir araştırma ekibinin, yaklaşık on yıl içinde, bugün bilinen metin madenciliği süreçlerinin temelini oluşturan, metinlerin otomatik olarak indekslenmesi için geliştirdikleri modeldir. Model belgeler ve indeksler olarak bir vektörde toplanan önceden belirlenmiş belirli özelliklere göre sınıflandırılır. İki belge birbirine ne kadar benzerse, onları tanımlayan vektörlerin açısı o kadar küçük olur [17].

Öneri sistemlerinin temelleri ilk olarak 1970'lerin ikinci yarısında Duke Üniversitesi'nin geliştirdiği Usernet isimli sistemdir [13]. Bu araç metinsel içeriklerin kullanıcıların birbiri ile paylaşmasını sağlamaktadır. Bu sistemde içerikler kolay arama için haber grupları ve alt gruplar şeklinde kategorize edilmiştir. Usernet ile öneri sistemlerinin ilk temelleri atılırken aynı zamanda içerik tabanlı öneri sistemlerinin de temelleri gerçekleştirilmiştir. Daha sonra 1992'de GroupLens Usernet üzerindeki değerlendirmiş makaleleri kullanarak kullanıcılara Usernet makaleleri öneren bir öneri sistemi geliştirmiştir [13].

Öneri sistemleri için bilinen ilk çözüm ise 1979 yılında bilgisayar kütüphanecisi Grundy tarafından gelmiştir. Grundy, kullanıcılar ile röportajlar yaparak, kullanıcıların verilerini toplamış ve kullanıcıları kalıplaşmış gruplara ayırarak aynı gruptaki herkese aynı kitabı önermiştir [14].

1990'ların başında Xerox Parc tarafından geliştirilen Tapestry sistemi ilk iş birlikçi filtreleme sistemlerine örnek gösterilebilir. Sistem öncelikle kullanıcıların okudukları

belgelerine beğeni yapmalarını sağlamaktadır. Öneri sistemleri bu beğeni bilgilerini toplayarak kullanıcılara ilgilenebilecekleri belgeleri önerir. Daha sonra sistem geliştirilerek belgeleri yorumlama özelliği getirilmiştir. Böylelikle sistem beğeni ve kullanıcıların yorumları ile beslenerek belgeleri önermede oldukça başarılı sonuçlar elde etmiştir [15].

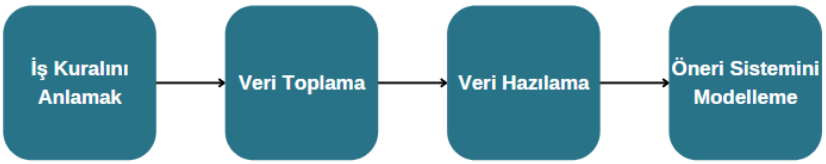
İşbirlikçi ve içerik tabanlı filtreleme çözümlerini birleştiren ilk çözüm, Stanford öğrencileri tarafından geliştirilen ve 1994 yılında sunulan Fab projesidir. Fab önce belirli konular için içerik toplar, ardından her bir kullanıcı için belirli konulardan toplanmış ve özellikle ilgilerini çekmesi muhtemel öğeleri seçer ve son olarak bu içerikler kullanıcıya ulaştırır. Fab'ta iki öneri sisteminin bir arada kullanılmasının sebebi, her bir öneri sisteminin oluşturacağı dezavantajları ortadan kaldırmaktır [19].

İçerik tabanlı öneri sistemi üzerine yapılmış başarılı araştırmalardan bir diğeri 1999 tarihli Music Genome Projesidir. Genome Projesi kapsamında bir şarkı için 450'den fazla özellik ortaya çıkarılmış ve bunların ilişkileri bir algoritma kullanılarak tanımlanmıştır. Modelin temeli, kullanıcı bir şarkıyı beğendiğinde, şarkının belirli özelliklerine (tarz, dönem, sanatçı, orkestrasyon, vuruş vb.) pozitif değerler atanmasıdır. Benzer özelliklere sahip şarkılar daha sonra tercih listesinde öne çıkarılarak ve kullanıcılara sunulmuştur. Bugün 250 milyon kullanıcısı olan Pandora İnternet Radyosu bu proje üzerine kuruludur [18].

Hibrit sistemler ilk başta iki farklı yöntemin birbiri içine yerleştirerek oluşturulsa da 2000'li yılların başında farklı stilde kullanımlar da denenmiştir. Bunlardan en iyi örnek Netflix'tir. Netflix'in algoritması olan CineMatch, 2000'li yılların başında çevrimiçi film satışları için en başarılı öneri sistemidir. Bu algoritma hem içerik hem de işbirlikçi filtreleme yöntemlerini kullanarak ortak bir öneride bulunmaktadır. Dönemin en başarılı algoritmasını %10 daha iyileştirmek isteyen Netflix 2006 yılında ödüllü yarışmasını duyurarak bugün öneri sistemlerinin hem literatürde hem de endüstride yoğun ilgi oluşmasını sağlayan ilk adımı atmıştır [20].

1.1.2 Öneri sistemi iş akışı

Bir öneri sistemini doğru tasarlayabilmek için öneri sistemi iş akışına hakim olmak gerekir. Kullanıcılara en iyi tahminler sunmak için iş akışı sürecinde dikkat edilmesi gereken noktalar vardır. Şekil 1.1'de öneri sistemi tasarlanması aşamasında gerçekleştirilen iş akışı sunulmuştur.



Şekil 1.1 : Öneri sistemi iş akışı.

Doğru bir öneri sisteminin tasarlanabilmesi için ilk adım iş kuralını anlamaktır. Öneri sağlanacak kullanıcı kitlesinin verileri, ürün hizmet verileri, kullanıcı segmentasyonlarının belirlenmesi, öneriden beklenen bağlam gibi öneri sisteminin başarıyla gerçekleşmesini sağlayacak durumların iyi anlaşılması ve bu doğrultuda sistemin tasarlanması gerekir.

İkinci adım olarak veri toplama işlemi gerçekleştirilir. Bir öneri sistemi ne kadar çok veri ile beslenir ise kullanıcılara daha doğru öneriler sunacaktır. Kullanıcı tercihlerinde iki tür geri bildirim mevcuttur: Açık ve örtük geri bildirim.

Açık geri bildirim bir kullanıcının bir ürünü veya hizmeti direk değerlendirdiği bildirimdir. Açık geri bildirimler çok net olabileceği gibi ön yargılı da olabilir. Örneği bir kullanıcı yüksek bir geri bildirim verdiği şeyi hiç denememiş olabilir ya da düşük geri bildirim verdiği şeyi beğenmiş de olabilir.

Örtük geri bildirim ise kullanıcının herhangi bir değerlendirme yapmadan uygulama veya web sitesi üzerine bıraktığı verileri tanımlar. Örtük bildirime arama ve satın alma geçmişi, belirli teklifleri veya içerikleri inceleme, sayfalarda durduğu süreler ve sosyal ağlar üzerinden alınan veriler örnek olarak gösterilebilir. Örtük geri bildirimlerinin yorumlanması açık geri bildirimlere göre daha karmaşıktır. İyi bir öneri sistemi tasarlamak için genellikle bu iki geri bildirimin kombinasyonunda yararlanılır.

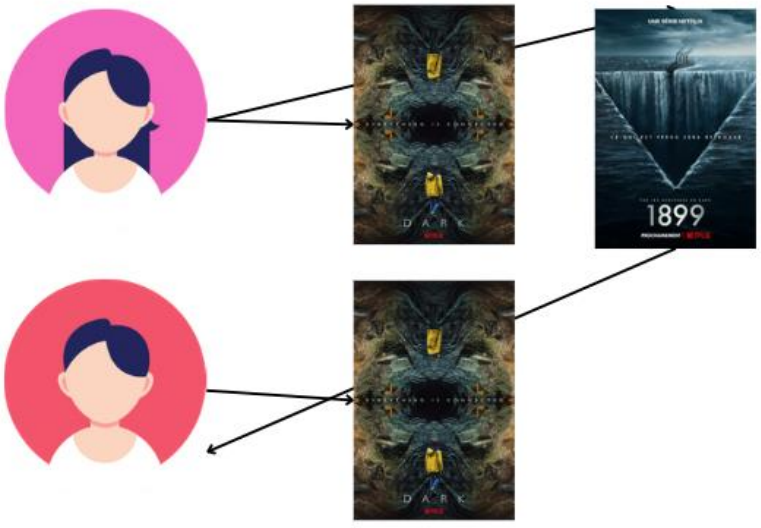
Öneri sistemi tasarımı aşamasında üçüncü adım verilerin öneri sistemi modeli için hazırlanmasıdır. Veriler hazırlanırken dikkat edilmesi gereken en önemli şeylerden biri değişen kullanıcı zevkleridir. Yalnızca kullanıcının mevcut zevklerini temsil etme olasılığı daha yüksek olan özelliklere öneri sistemine dahil etmek, modelin başarının arttırır.

Öneri sistemindeki son adım olan öneri sistemi modellemede ihtiyaca göre doğru algoritmanın seçilmesi oldukça önemlidir. Öneri sistemleri üç farklı şekilde tasarlanabilir: İşbirlikçi Filtreleme, İçerik Tabanlı, Demografik Tabanlı, Popülarite Tabanlı ve Hibrit Öneri Sistemleri.

1.1.3 Öneri sistemi yöntemleri

Beş farklı öneri sistemlerinden ilki olan işbirlikçi filtreleme, kullanıcıların veya öğelerin tercihlerini benzer kullanıcılar veya öğelerle karşılaştırarak tahmin etme yöntemidir. Bu yöntem, geçmişte benzer tercihe sahip olan bireylerin gelecekte aynı tercihleri yapacaklarını varsaymaktadır. İşbirlikçi filtrelemenin en önemli avantajlarından biri analiz için içeriğe ihtiyaç duymadan karmaşık öğeleri doğru bir şekilde önerebilmesidir. İşbirlikçi filtreleme kendi içerisinde iki farklı alt yönteme ayrılmaktadır: Kullanıcı tabanlı ve öğe tabanlı.

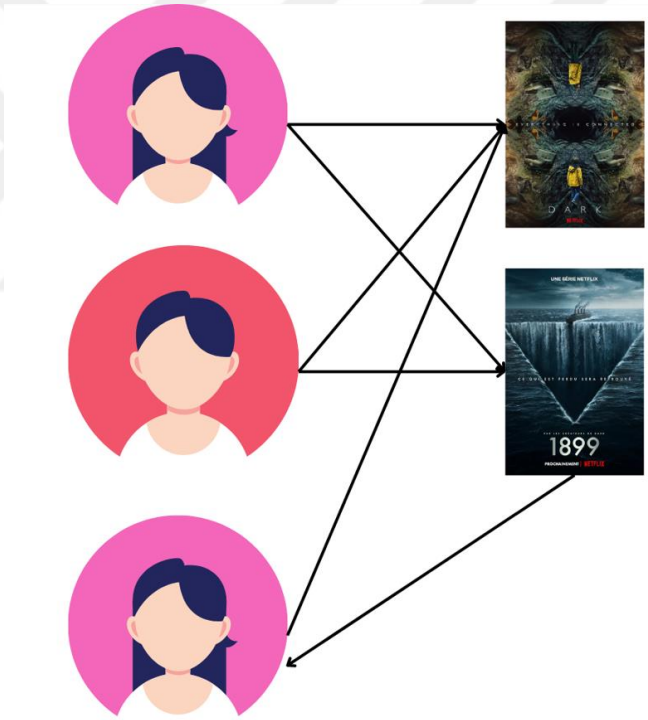
Kullanıcı tabanlı işbirlikçi filtreleme kullanıcıların birbiri arasındaki benzerlik ilişkisine göre önerileri gerçekleştirir. Kullanıcı tabanlı işbirlikçi filtreleme her iki müşterinin bilgisini analiz etmeyi gerektirir. Bu, modelin çok fazla hesaplama gücünün harcanmasına neden olur. Ayrıca kullanıcıların alışkanlıkları zamanla değişmektedir. Bu durum doğru ve yararlı tavsiyeler bulmayı zaman içinde zorlaştırabilir [21]. Şekil 1.2'de kullanıcı tabanlı işbirlikçi filtreleme yönteminin yapısı sunulmuştur.



Şekil 1.2 : Kullanıcı tabanlı işbirlikçi filtreleme yapısı.

Şekil 1.2'de görüldüğü gibi kullanıcı tabanlı sistemlerde, öneri modeli kullanıcıları karşılaştırır. Şekil 1.2'deki kullanıcılar aynı film hakkında sisteme derecelendirme bilgisi bıraktıkları görülmektedir. Öneri sistemi bu bilgiyi kullanarak bu iki kullanıcı için benzer kullanıcılar olabileceği bilgisini ortaya çıkarır ve birinci kullanıcının derecelendirdiği filmi ikinci kullanıcıya önerir.

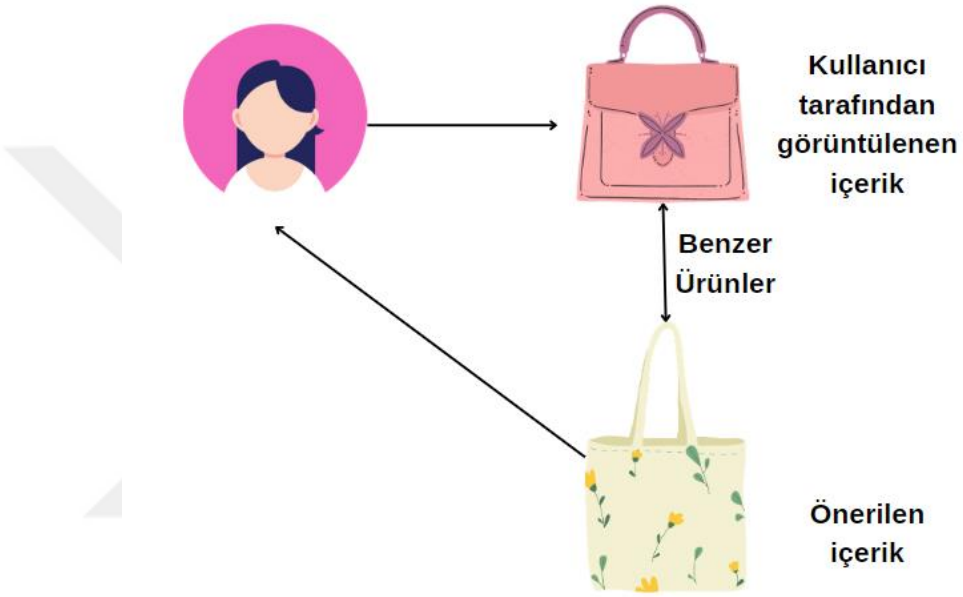
Öğe tabanlı işbirlikçi filtreleme ise, öğelerin birbiri arasındaki benzerlik ilişkisine göre önerileri gerçekleştirir. Öğe tabanlı işbirlikçi filtreleme yöntemi öğe benzerliğine odaklanır. Kullanıcı tabanlı işbirlikçi filtrelemeye göre modellemek için daha az kaynak ve zaman gerektirir. Öğe tabanlı işbirlikçi filtreleme kullanıcıların zamana bağlı değişen alışkanlıklarından kullanıcı tabanlı işbirlikçi filtrelemeye göre daha az etkilenerek, kullanıcılara daha doğru öneriler sunmayı sağlar [21]. Şekil 1.3'te öğe tabanlı işbirlikçi filtreleme yönteminin yapısı sunulmuştur.



Şekil 1.3 : Öğe tabanlı işbirlikçi filtreleme yapısı.

Şekil 1.3'te görüldüğü gibi öğe tabanlı işbirlikçi filtreleme sistemlerinde, öneri modeli öğeleri karşılaştırır. Şekil 1.3'te üç kullanıcının da aynı film hakkında sisteme derecelendirme bıraktığı görülmektedir. Birinci ve ikinci kullanıcının da iki aynı filme derecelendirme bilgisi vermiştir. Bu noktada öneri sistemi bu iki filmi karşılaştırarak benzerlik ilişkisini bulmaya çalışır. Yakaladığı bu benzerlik ilişkisi bilgisine göre birinci filmi izlemiş üçüncü kullanıcıya, ikinci filmi de önerir.

İçerik tabanlı öneri sistemleri, benzer içeriğe sahip bir öğenin, öğe içeriğine göre tavsiye edilmesidir. Bu sistem, geçmişte tercih edilen bir ürünün içeriği alarak benzer içerikteki başka ürünleri önermeye dayanır. İçerik tabanlı öneri sistemlerinde anahtar kelimeler önerilecek ürünü veya hizmeti tanımlamak için kullanılır. İçerik tabanlı öneri sistemleri ile ilgili en büyük sorun, ürünlerin içerik kaynağından kullanıcı tercihlerinin öğrenilememesidir. Bu da kullanıcı tercihlerini anlamayı zorlaştırmaktadır [21]. Şekil 1.4'te içerik tabanlı öneri sistemlerinin yapısı gösterilmiştir.



Şekil 1.4 : İçerik tabanlı öneri sistemi yapısı.

Şekil 1.4'te kullanıcı bir öğeyi görüntüleyerek ya da bir derecelendirme bilgisi bırakarak sisteme bir veri bırakabilir. İçerik tabanlı öneri sistemlerinde kullanıcının ilgilendiği öğenin içerik bilgileri alınır ve benzer içerikler bulunmaya çalışılarak kullanıcıya en benzer öğe önerilir. İçerik tabanlı öneri sistemlerinde içerik benzerliklerini bulma konusunda genellikle doğal dil işleme yöntemlerinden yararlanılır.

Demografik tabanlı öneri sistemleri kullanıcıların demografik yapılarına göre öneri sunan sistemlerdir [22]. Örneğin üniversite sınavlarına hazırlanan yaş grubu öğrencilere test kitaplarının önerilmesi bu yönteme örnek gösterilebilir. Burada kullanıcının sisteme girdiği herhangi bir bilgi kullanılmaz. Sadece kullanıcıya sorulan demografik soruların yanıtlarına göre öneriler gerçekleştirilir. Uygulaması kolay ve pratiktir.

Popülarite tabanlı öneri sistemleri, kullanıcı tercihlerine dayanmayan ilgili uygulamada veya web sitesinde popular olan öğenin önerildiği sistemlerdir. Genel olarak bu sistemler kullanıcıların ilgi alanlarını genişletmek, uygulama tercihlerini yaygınlaştırmak amacıyla kullanılabilir [22].

Hibrit öneri sistemi ise, işbirlikçi filtreleme ile içerik temelli bir yaklaşımı birleştiren bir yöntemdir. Hibrit öneri sistemleri iki ayrı yöntemi ayrı ayrı geliştirdikten sonra birleştirilip uygulanabileceği gibi işbirlikçi yaklaşıma içerik tabanlı özellikler eklenerek ya da içerik tabanlı yaklaşıma işbirlikçi yöntemler eklenerek tek bir modelde de geliştirilebilirler [21].

1.1.4 Öneri sistemi problemleri

Öneri sistemleri, başarılı öneriler gerçekleştirmek içine etkin çözümler sağlasalar da birçok farklı problemle de karşılaşırlar. Bu problemler doğruluk, soğuk başlama, veri seyrekliği, ölçeklenebilirlik ve güvenilirliktir.

Doğruluk problemi, bir öneri sisteminin kullanıcının gereksinimlerini karşılayacak şekilde doğru öneriler sunmasıdır. Bir öneri sisteminin kritik gereksinimlerinden biri, kullanıcılara heyecan verici ve alakalı öğeler getirmektir. Kullanıcıların sisteme olan güven düzeyi, tavsiyelerin kalitesiyle doğrudan ilgilidir. Kullanıcılara uygun ürün ve hizmetler sunulmazsa, öneri sistemi müşteri memnuniyeti açısından yetersiz olarak değerlendirilebilir ve bu da kullanıcıların alternatif sistemler aramasını açık hale getirir. Bu nedenle, bir öneri sistemi, tercih edilebilirliği ve etkinliği iyileştirmek için uygun bir tahmin doğruluğu düzeyini karşılamalıdır.

Soğuk başlama problemi, sisteme yeni katılan kullanıcının bilgi eksikliği nedeniyle öneri gerçekleştirilmemesi sonucu oluşan bir problemdir. Soğuk başlatma problemi öneri sisteminin performansını düşürür ve kullanıcılara doğru önerilerin sunulmasını engeller [3].

Ölçeklenebilirlik problemi, öneri sisteminde kullanıcı verisine göre ölçeklenebilmesi problemini tanımlar. Öneri sistemleri tasarlanırken sınırlı veriler kullanılarak geliştirilmiştir. Fakat veri akışı sürekli artmaktadır. Bu da öneri sistemlerinde ölçeklenebilirlik sorununa yol açmaktadır. Bu nedenle, bir veri tabanında veri kümesinin hacmi arttıkça başarılı bir şekilde ölçeklendirilebilen öneri tekniklerinin kullanılması çok önemlidir [3].

Güvenilirlik problemi, kullanıcıların açıklama eksikliği veya yanlış öneri sonuçları nedeniyle öneri sistemlerine güvenememesidir. Örneğin, bir kullanıcı benzer geçmiş kayıtlarına sahip olsa bile bir yabancının zevkine güvenmeyebilir; dahası, sistem kötü niyetli kullanıcılar tarafından kasıtlı olarak yüksek puan alan bir öğe önerebilir. Bütün bunlar, güvenilir bir öneri sistemini acil ve önemli kılar.

Veri seyrekliği problemi sistemdeki yetersiz bilgi nedeniyle ortaya çıkan ve işbirlikçi filtreleme yöntemlerini etkileyen bir problemdir. Bu problem, veri tabanındaki kullanıcı tarafından derecelendirilen öğelerin toplam sayısının az olması nedeniyle kullanıcı öğe matrisinin seyrek bir matris olmasına neden olur. Aynı zamanda seyrek kullanıcı öğe matrisinde başarılı komşular bulamamaya ve nihayetinde kötü bir tavsiye sürecine yol açar [3].

1.2 Tezin Amacı

Bu tez kapsamında yapılan incelemeler sonucunda öneri sistemlerinin günlük hayatta çok yaygın kullanıldığı gözlemlenmiştir. Öneri sistemleri kullanıcıların kişisel tercihlerini tahmin etmekte ve hatta onları etkilemektedir. Öneri sistemlerinin iyi performans göstermesi için kullanıcıların öğe hakkındaki derecelendirmelerine ihtiyaç duyar. Fakat kullanıcılar öğe hakkındaki bilgileri kendinde saklı tutmak isteyip, öneri sistemleri ile paylaşmak istemeyebilir. Bu durum öneri sistemi için veri seyrekliğine neden olur. Öneri sistemlerinde veri seyrekliği problemi iki farklı yöntemle ele alınır. İlk yöntem meta veri kullanımı, ikinci yöntem ise seyrek kullanıcı öğe matrisini kullanıcı-öğe derecelendirmeleri kullanarak yeniden yapılandırmaktır.

Öneri sistemlerinin başarılı tahminler yapabilmesinin arkasında iyi tasarlanmış yapay zeka algoritmalarının yanında, bu algoritmaların eğitim sürecinde kullandıkları devasa boyutlarda verilerde yatmaktadır. Kullanıcılar her saniye sistemlere veriler bırakmaktadır. Bu verilerin birçoğu kullanıcıların kişisel verileri veya ilgili öğelerin çeşitli bilgilerini içeren meta verilerdir. Meta veriler algoritmaların başarılarını büyük oranda arttıracak olsa da bu veriler kullanıcıların kişisel bilgilerini de içerebildiğinden veri ihlali sorunlarına yol açarak kullanımına izin verilmeyebilir. Ayrıca kullanıcı-öğe derecelendirme matrisinin yanına meta bilgilerin de dahil edilmesi öneri sistemi algoritmasının karmaşıklığı arttıracaktır. Bu tezin amacı, öneri sistemlerindeki veri seyrekliği problemine karşı sadece kullanıcı-öğe matrisi

kullanıldığında derin öğrenme algoritmalarının veri seyrekliğine karşı gösterdiği performansı analiz etmek üzerinedir. Bu tez çalışması kapsamında derin öğrenme algoritmalarının sadece kullanıcı öğe matrisindeki verileri kullanarak veri seyrekliğine karşı matrisi ne kadar başarılı doldurabildiği incelenecektir. Böylelikle herhangi bir meta veri kullanmadan, karmaşık yapay zeka modelleri kurgulamadan derin öğrenme algoritmaları ile doğru öneriler yapan öneri sistemlerinin tasarlanması amaçlanmaktadır.

Bu tez çalışmasında derin öğrenme algoritmaları üzerinde düşük, orta ve yüksek veri seyrekliği içeren veri setleri kullanılacaktır. Bu veri setleri ile algoritmaların veri seyrekliği arttıkça derin öğrenme algoritmalarını bu seyreklik karşısında ne kadar etkilediğini gözlemlenmesi hedeflenmektedir.

1.3 Literatür Araştırması

Derin öğrenme algoritmaları kullanılarak öneri sistemleri geliştirilmesi konusunda literatürde birçok çalışma gerçekleştirilmiştir. Bu bölümde bu çalışmalar özetlenecektir.

Sedhain ve diğ., seyrek kullanıcı-öğe matrisini girdi olarak alan ve bunları çıktı katmanında yeniden yapılandırmayı amaçlayan bir temel otomatik kodlayıcı modeli önermiştir. Bizim çalışmamızla benzer bir bakış açısına sahip olan bu çalışmada bizim çalışmadan farklı yönü temel otomatik kodlayıcı algoritması üzerinde hiper parametre optimizasyonu gerçekleştirmiş olmasıdır. Veri seti olarak Movielens-1M, Movielens-10M ve Netflix veri setleri kullanılmıştır. Değerlendirme sonuçları RMSE ile alınan bu çalışmada Movielens-1M için 0.831, Movielens-10M için 0.782 ve Netflix için ise 0.832 sonuçları elde edilmiştir [23].

Ferreira ve diğ., bir gizli katman ve bir bırakma(dropout) katmanı kullanılarak bir derin otomatik kodlayıcı modeli önermiştir. Bu çalışma bizim çalışmamıza benzer bir düşünceyle oluşturulmuş yeni bir model önerisidir. Çalışmada, Movielens-100K ve Movielens-1M veri setleri kullanılmıştır. Değerlendirme sonuçları RMSE ile alınan bu çalışmada Movielens-100K için 0.029 Movielens-1M için 0.015 sonuçları elde edilmiştir [7].

Noshad ve diğ., derin otomatik kodlayıcıyı kullanarak karşılıklı bilgi tabanlı bir öneri sistemi önermiştir. Model ilk olarak derin otomatik kodlayıcıda eksik verileri

yeniden yapılandırır. Ardından tahmin edilen kullanıcılara benzer kullanıcıları bulmak için tahmin edilen seyrek verilere karşılıklı bilgiyi uygular. Veri seti olarak Netflix veri seti kullanılmıştır. Değerlendirme sonuçları MAE metriği ile alınan bu çalışmada 0.75 sonucu elde edilmiştir [24].

Jiang ve diğ., sadece eksik olmayan derecelendirmelerin bulunduğu nöronları etkileşime sokan bir derin otomatik kodlayıcı kullanan hızlı öneri sistemi modeli önermiştir. Çalışmada veri seti olarak Movielens-1M, Movielens-10M ve The EachMovie veri setleri kullanılmıştır. Değerlendirme metriği olarak RMSE ve MAE metrikleri kullanılmıştır. Değerlendirme sonuçlarında Movielens-1M için RMSE 0.8347, MAE 0.6537, Movielens-10M için RMSE 0.7843, MAE 0.5990 ve TheEachMovie'de ise RMSE 0.2301, MAE 0.1749 sonuçları elde edilmiştir [25].

Chen ve Wu dikkat birimlerini kullanarak bir gürültü giderici otomatik kodlayıcı modeli önermiştir. Bu çalışma da bizim çalışmadaki hedeflediğimiz seyrek veri problemini gürültü giderici otomatik kodlayıcı modeline çözdürmeyi amaçlamıştır. Bu bağlamda algoritma üzerinde farklı tasarımlara giderek modelleme gerçekleştirmiştir. Çalışmada veri seti olarak Movielens-1M ve Movielens-10M veri setleri kullanılmıştır. Değerlendirme sonuçları RMSE ile alınan bu çalışmada Movielens-1M için 0.736, Movielens-10M için ise 0.756 sonuçları elde edilmiştir [26].

Khan ve diğ., gürültü giderici otomatik kodlayıcı özelliklerini temel alarak, bir kullanıcının bir dizi öğeye yönelik derecelendirme eğilimini değerlendirerek kullanıcı-öğe korelasyonlarını belirleyen, yeni bir kullanıcı derecelendirme-eğilim tabanlı işbirlikçi gürültü giderme otomatik kodlayıcı modeli önermiştir. Çalışmada Movielens-100K ve Movielens-1M veri setleri kullanılmıştır. Değerlendirme sonuçlarında Movielens-100K veri seti için Kesinlik, metriğinde 0.2654, Anma, metriğinde 0.1531, Ortalama Kesinlik Değerlerinin Ortalaması metriğinde 0.2182, Ortalama Karşılıklı Sıralama metriğinde 0.5193 ve Normalleştirilmiş İndirimli Kümülatif Kazanç metriğinde ise 0.5263 sonuçları elde edilirken, Movielens-1M veri setinde için ise Kesinlik metriğinde 0.2712, Anma metriğinde 0.1079, Ortalama Kesinlik Değerlerinin Ortalaması metriğinde 0.1806, Ortalama Karşılıklı Sıralama metriğinde 0.5074 ve Normalleştirilmiş İndirimli Kümülatif Kazanç metriğinde ise 0.5183 sonuçları elde edilmiştir [27].

Wu ve diğ., gürültü giderici otomatik kodlayıcıları kullanarak, bir top-n öneri sistemi modeli önermiştir. Gürültü giderici otomatik kodlayıcı yapısını kullanan model, kullanıcı öğesi geri bildirim verilerini formüle ederek kullanıcıların ve öğelerin dağıtılmış temsillerini öğrenir. Çalışmada Movielens-10M, Netflix ve Yelp veri setleri ile birlikte değerlendirme metriği olarak Ortalama Kesinlik Değerlerinin Ortalaması metriği seçilmiştir. Değerlendirme sonucunda Movielens-10M 0.3494, Netflix 0.2608 ve Yelp veri setinde ise 0.0528 sonuçları elde edilmiştir [28].

Tian ve diğ., gürültü giderici otomatik kodlayıcının kodlayıcı ve kod çözme işlemlerine bir denge matrisi ekleyerek geliştirilen bir öneri sistemi modeli önermiştir. Veri seti olarak Movielens-10M veri seti kullanılmıştır. Değerlendirme metriği olarak RMSE, Kesinlik ve Anma metrikleri kullanılmıştır. Değerlendirme sonuçlarında RMSE'de 0.83, Kesinlik'de 0.08 ve Anma metriğinde ise 0.75 sonuçları elde edilmiştir [29].

Zhang ve diğ., derin seyrek otomatik kodlayıcı ve tekil değer ayrışımı (Singular Value Decomposition - SVD++) algoritmalarını kullanarak hibrit bir öneri sistemi modeli önermiştir. Model, derin seyrek otomatik kodlayıcı kullanarak kullanıcı-öğe matrisindeki etkili temsilleri öğrenir. Öğrenilen özellik temsilleri, SVD++ algoritmasında öğe gizli vektörünü değiştirmek için kullanır. Çalışmada veri seti olarak Movielens-100K, Movielens-1M, Ciao veri setleri ve değerlendirme metriği olarak RMSE ve MAE metrikleri kullanılmıştır. Değerlendirme sonuçlarına göre Movielens-100K RMSE 0.9123, MAE 0.723, Movielens-1M'de RMSE 0.8478, MAE 0.672 ve Ciao ise RMSE 0.9534, MAE 0.702 sonuçları elde edilmiştir [30].

Liang ve diğ., gizli katmandaki parametreleri tahmin etmek için ilkeli bir Bayes çıkarımı kullanan bir varyasyonel otomatik kodlayıcı modeli önermiştir. Çalışmada Movielens-20M, Million Song Dataset ve Netflix veri seti ile değerlendirme metriği olarak Anma ve Normalleştirilmiş İndirimli Kümülatif Kazanç metrikleri kullanılmıştır. Değerlendirme sonuçlarına göre Movielens-20M Anma 0.537, Normalleştirilmiş İndirimli Kümülatif Kazanç 0.426, Netflix'te Anma 0.444, Normalleştirilmiş İndirimli Kümülatif Kazanç 0.386 ve Million Song Dataset ise Anma 0.364 ve Normalleştirilmiş İndirimli Kümülatif Kazanç 0.316 sonuçların elde edilmiştir. Ayrıca sadece Movielens-20M kullanılarak Gauss ve lojistik dağılımları içeren 2 farklı varyasyonel otomatik kodlayıcı yazarların gerçekleştiği çalışma ile karşılaştırılmıştır. Değerlendirme sonuçlarında önerilen modelin parametre

tahmininde Gauss ya da lojistik dağılımlarına göre daha iyi performans gösterdiği gözlemlenmiştir [31].

Sachdeva ve diğ., Liang ve diğ., çalışmalarının bir uzantısı olarak bu çalışmanın tekrarlayan bir versiyonunu önermiştir. Çalışma zamansal bağımlılıklardan bağımsız olarak tüm geçmişin bir alt kümesini geçmek yerine, tüketim dizisi alt kümesini tekrarlayan bir sinir ağından geçirirler. Böylelikle varyasyonel otomatik kodlayıcının doğruluğunu artırmayı hedeflerler. Çalışmada Movielens-1M ve Netflix veri setleri ve değerlendirme metriği olarak Kesinlik, Anma ve Normalleştirilmiş İndirimli Kümülatif Kazanç metrikleri kullanılmıştır. Değerlendirme sonuçlarına göre Movielens-1M Kesinlik 0.1440, Anma 0.1248, Normalleştirilmiş İndirimli Kümülatif Kazanç 0.1781 ve Netflix'te ise Kesinlik 0.2193, Anma 0.893 ve Normalleştirilmiş İndirimli Kümülatif Kazanç 0.2464 sonuçları elde edilmiştir [32].

Pang ve diğ., varyasyonel otomatik kodlayıcı modeline aynı koşul etiketine sahip kullanıcı veya öğelerin kendilerine özgü oluşturulan özel bir gizli ortalama faktörünü ekleyen bir model önermiştir. Veri seti olarak Movielens-1M ve Movielens-2K veri seti kullanılmıştır. Değerlendirme metriği olarak Anma ve Normalleştirilmiş İndirimli Kümülatif Kazanç metrikleri kullanılmıştır. Değerlendirme sonuçlarında Movielens 1-M Anma'da 0.45, Normalleştirilmiş İndirimli Kümülatif Kazanç'ta 0.3478 ve Movielens-2K'da Anma'da 0.4808, Normalleştirilmiş İndirimli Kümülatif Kazanç'ta 0.2211 sonuçları elde edilmiştir [33].

Askari ve diğ., hem kullanıcı tabanlı hem de öğe tabanlı varyasyonel otomatik kodlayıcıyı birleştiren bir model önermişlerdir. Her iki modelin verdiği çıktının ortalaması alınarak nihai yapılandırılmış kullanıcı öğe matrisi elde edilir. Veri seti olarak Movielens-1M, Yelp ve Pinterest kullanılmıştır. Değerlendirme metriği olarak F1 Skor ve Normalleştirilmiş İndirimli Kümülatif Kazanç metriği kullanılmıştır. Değerlendirme sonuçlarında Movielens-1M'de F1 Skor 0.2092, Normalleştirilmiş İndirimli Kümülatif Kazanç 0.2290, Yelp'te F1 Skor 0.0395, Normalleştirilmiş İndirimli Kümülatif Kazanç 0.0553 ve Pinterest'te ise F1 Skor 0.0538, Normalleştirilmiş İndirimli Kümülatif Kazanç 0.0666 sonuçları elde edilmiştir [34].

Chakrabarty ve diğ., Kısıtlı Boltzmann makinelerini ve K-Ortalama algoritmasını kullanarak bir şaka öneri sistemi geliştirmiştir. Bernolli Kısıtlı Boltzmann Makinesi kullanılan modelde, kayıp olan veri yeniden yapılandırılır. Ardında derecelendirilmiş

tüm şakalarla eğitilen K-Ortalama algoritması ile Kısıtlı Boltzmann Makinesi modelinin performansı çizgi grafiği kullanılarak karşılaştırılır. Karşılaştırma sonucunda birbirine yakın şakalarda çizgi hareketleri daha stabil görünürken, benzerlik azaldıkça çizgilerdeki hareketliliğinde arttığı gözlemlenmiştir [35].

Sahoo ve Pradhan Kısıltı Boltzmann Makineleri ile evrişimli sinir ağını bir arada kullanan bir öneri sistemi modeli önermiştir. Veri seti olarak Movielens-100K veri seti ve değerlendirme metriği olarak MAE metriği kullanılmıştır. Değerlendirme sonuçlarına göre MAE değerini 0.1179 elde edilmiştir [36].

Kuo ve Chen, küme tabanlı Kısıtlı Boltzmann Makinesi'ni Diferansiyel Evrim algoritmasıyla birleştiren bir hibrit algoritma önermiştir. Küme tabanlı Kısıtlı Boltzmann Makinesi modeli, Kısıtlı Boltzmann Makinesi modeli için her mini toplu gradyan iniş yönteminin boyutunu ve öğelerini belirlemek üzere bir kümeleme algoritması uygular. Diferansiyel evrim algoritması ise Kısıtlı Boltzmann Makineleri'nin parametrelerini optimize ederek daha iyi sonuçlar vermesi hedeflenmektedir. Çalışmada Movielens-100K, Movielens-1M, Books-Crossing, Restautants veri setleri ve metrik olarak da MAE metriği kullanılmıştır. Değerlendirme sonuçlarında Movielens-100K 0.69629, Movielens-1M 0.699084, Books-Crossing 1.19188 ve Restautants 0.28712 sonuçları elde edilmiştir [37].

Behera ve diğ., kullanıcı derecelendirme bilgisinden yararlanarak, bilinmeyen derecelendirmeleri tahmin etmek için Kısıtlı Boltzmann Makinesi kullanan bir model önermiştir. Çalışmada Movielens-1M veri seti ve MAE metriği kullanılmıştır. Değerlendirmede 0.52 sonucu elde edilmiştir [38].

Liu ve diğ., üretici çekişmeli ağları kullanarak bir işbirlikçi filtreleme modeli önermiştir. Model, ayırıcı sinir ağından gelen pozitif ve negatif etiketli verileri, üretici ağda tekrar modelleyerek sahte verileri gerçek verilere en yakın olacak şekilde modellemeyi hedefler. Çalışmada Movielens-100K, Movielens-1M, UserTag, Netflix ve metrik olarak Kesinlik ve Normalleştirilmiş İndirimli Kümülatif Kazanç kullanılmıştır. Değerlendirme sonuçlarında Movielens-100K Kesinlik 0.4333, Normalleştirilmiş İndirimli Kümülatif Kazanç 0.4425, Movielens-1M Kesinlik 0.4769 Normalleştirilmiş İndirimli Kümülatif Kazanç 0.4856, UserTag Kesinlik 0.3227, Normalleştirilmiş İndirimli Kümülatif Kazanç 0.3272, Netflix'te ise Kesinlik

0.2884, Normalleştirilmiş İndirimli Kümülatif Kazanç'ta 0.2993 sonuçları elde edilmiştir [39].

Wang J. ve diğ., üretici ve çekişmeli ağları kullanarak bilgi çıkarımı yöntemi kullanan bir öneri sistemi modeli (IRGAN) önermiştir. Modelde üretici ağ, öğeler üzerine gerçek alaka dağılımını uydurarak belirli bir kullanıcı için bilgilendirici öğeleri seçer. Ayırıcı ağ ise, ilgili ve seçilmiş öğeler arasında ayrım yapar. Ardından, ayırıcı ağ daha bilgilendirici öğeleri seçmek için sonucu üreticiye besler. Oluşturulan öğeler, onu yanlış yönlendirmek için ayrımcıya girilir. Çalışmada Movielens-100K ve Netflix veri setleri ve Kesinlik, Ortalama Kesinlik Değerlerinin Ortalaması, Normalleştirilmiş İndirimli Kümülatif Kazanç ve Ortalama Karşılıklı Sıralama metrikleri kullanılmıştır. Değerlendirme sonuçlarında Movielens-100K Kesinlik 0.4072, Ortalama Kesinlik Değerlerinin Ortalaması 0.2418, Normalleştirilmiş İndirimli Kümülatif Kazanç 0.4222 ve Ortalama Karşılıklı Sıralama 0.6082, Netflix'te ise Kesinlik 0.4456, Ortalama Kesinlik Değerlerinin Ortalaması 0.1720, Normalleştirilmiş İndirimli Kümülatif Kazanç 0.4498 ve Ortalama Karşılıklı Sıralama 0.6371 sonuçları elde edilmiştir [12].

Tong ve diğ., üretici bölümünde varyasyonel otomatik kodlayıcı kullanan bir üretici çekişmeli ağ modeli önermiştir. Modelde üretici kısmında bulunan varyasyonel otomatik kodlayıcı eğitim verisinin dağılımını öğrenerek sahte verileri üretir. Ayırıcı ağ ise, üretilen örnekler ile gerçek örnekleri ayırt etmek olasılığını maksimize etmeye odaklanır. Çalışmada Movielens-1M ve Netflix veri setleri ve ve Kesinlik, Ortalama Kesinlik Değerlerinin Ortalaması, Normalleştirilmiş İndirimli Kümülatif Kazanç ve Ortalama Karşılıklı Sıralama metrikleri kullanılmıştır. Değerlendirme sonuçlarında Movielens-1M Kesinlik 0.449, Ortalama Kesinlik Değerlerinin Ortalaması 0.287, Normalleştirilmiş İndirimli Kümülatif Kazanç 0.458 ve Ortalama Karşılıklı Sıralama 0.597, Netflix'te ise Kesinlik 0.472, Ortalama Kesinlik Değerlerinin Ortalaması 0.207, Normalleştirilmiş İndirimli Kümülatif Kazanç 0.478 ve Ortalama Karşılıklı Sıralama 0.662 sonuçları elde edilmiştir [40].

Wang H. ve diğ., hızlı öneri sistemi konseptini benimseyen IRGAN çalışmasının başka ilgili uzantısı olan bir rakip ikili işbirlikçi filtreleme modeli önermiştir. Model hem çevrimiçi öneri hızında hem de depolama hızında büyük avantajlara sahip olmayı hedeflemektedir. Çalışmada Movielens-100K, Movielens-10M, Book, Yelp veri setleri ile Kesinlik ve Normalleştirilmiş İndirimli Kümülatif Kazanç metrikleri

kullanılmıştır. Değerlendirme sonuçlarına göre Movielens-100K Kesinlik 0.2742, Normalleştirilmiş İndirimli Kümülatif Kazanç 0.3244, Movielens-10M Kesinlik 0.2582, Normalleştirilmiş İndirimli Kümülatif Kazanç 0.2648, Book'ta Kesinlik 0.0352, Normalleştirilmiş İndirimli Kümülatif Kazanç 0.0457 ve Yelp'te ise Kesinlik 0.0510 Normalleştirilmiş İndirimli Kümülatif Kazanç 0.0520 sonuçları elde edilmiştir [41].

Fan ve Chow otomatik kodlayıcılar tabanlı matris tamamlama yöntemi önermiştir. Burada otomatik kodlayıcı doğrusal olmayan bir gizli değişken modeli öğrenmek ve oluşturmak için kısmen gözlemlenen verileri kullanır. Çalışmada Jester, Movielens-100K ve Movielens-1M veri setleri ile değerlendirme metriği olarak Normalleştirilmiş Ortalama Mutlak Hata kullanılmıştır. Değerlendirme sonuçlarına göre Jester'da 16.17, Movielens-100K'da 18,87 ve Movielens-1M'de 18,17 sonuçları elde edilmiştir [42].

Mehrdad ve Kahaei veri matrisi için doğrusal ve doğrusal olmayan modellerin bir kombinasyonu olan bir derin sinir ağı tabanlı matris tamamlama algoritması önermiştir. Bu algoritma ile kullanıcı öğe matrisini yeniden yapılandırarak, daha doğru sonuçlara ulaşmayı hedeflemişlerdir. Çalışmada Jester ve Movielens-100K veri setleri ile değerlendirme metriği olarak Normalleştirilmiş Ortalama Mutlak Hata kullanılmıştır. Değerlendirme sonuçlarına göre Jester'da 15.61, Movielens-100K'da 18.02 sonuçları elde edilmiştir [43].

2. DERİN ÖĞRENME ALGORİTMALARI

Derin öğrenme, herhangi bir sinir hücrelerinden oluşan farklılaştırılabilir mimariyi, stokastik gradyan iniş fonksiyonunu kullanarak farklılaştırılabilir bir objektif işlevi optimize eden algoritmalardır [1]. Makine öğrenmesinin bir alt kümesi olan derin öğrenme algoritmaları, yapay sinir ağlarından esinlenerek ortaya çıkmıştır. Derin öğrenme, verilerin özellik temsillerini keşfetmek için üst düzey temsil öznitelik kategorilerini daha düşük seviyeli özelliklere dönüştürür [42]. Görüntü sınıflandırma ve nesne tanıma problemleriyle öne çıkan derin öğrenme algoritmaları bu uygulamaların dışında doğal dil işleme, zaman serisi tahmini ve öneri sistemleri gibi

alanlarda da sıklıkla kullanılmaktadır. Derin öğrenme aktivasyon fonksiyonlarını (relu, tanh, sigmoid vb.) kullanarak verilerde doğrusal olmayan yapıları da öğrenme yeteneğine sahiptir. Bununla doğrusal öneri modellerinin yakalayamadığı karmaşık kullanıcı-öğe etkileşim kalıpları derin öğrenme teknikleri ile yakalanabilir [44].

Derin öğrenme algoritmaları öneri sistemlerinde üç farklı amaç için kullanılabilir: Temsili öğrenme, tahmine dayalı öğrenme ve üretken öğrenme [45]. Temsil öğrenme, içerik, etiket veya görüntüler gibi yardımcı verilerden önemli temsilleri öğrenmek için derin öğrenme algoritmalarını kullanır ve daha sonra tahmin için bu temsilleri geleneksel matris ile birleştirilir.

Örneğin, otomatik kodlayıcının darboğaz katmanı, öğeler ve kullanıcılar için önemli temsilleri öğrenmek amacıyla kullanılabilir [9, 46]. Tahmine dayalı öğrenme kullanıcının tercihlerini tahmin etmek için derin öğrenme algoritmalarını kullanır. Otomatik kodlayıcının girdileri yeniden yapılandırmaya çalışan kodlayıcı kısmı da derecelendirmeler için bir tahmin edici olarak kabul edilebilir [9].

Üretken öğrenme ise potansiyel derecelendirmelerin var olan derecelendirmeler kullanarak üretilmesini sağlayan yapılardır. Örneğin, bir üretici ve bir ayrımcı içeren üretici çekişmeli sinir ağı, bir minimax oyun çerçevesinde aynı anda iki sinir ağını eğiterek gerçek örneklere benzer dağılıma sahip örnekler üretebilir [47].

Bu özellik, öneri problemlerini karşılarken derecelendirmeleri oluşturmak veya eksik değerlerle ilgilenmek için kullanılabilir [12, 48]. Bu bölümde tez çalışmasında kullanılan altı farklı derin öğrenme algoritmalarından bahsedilecektir.

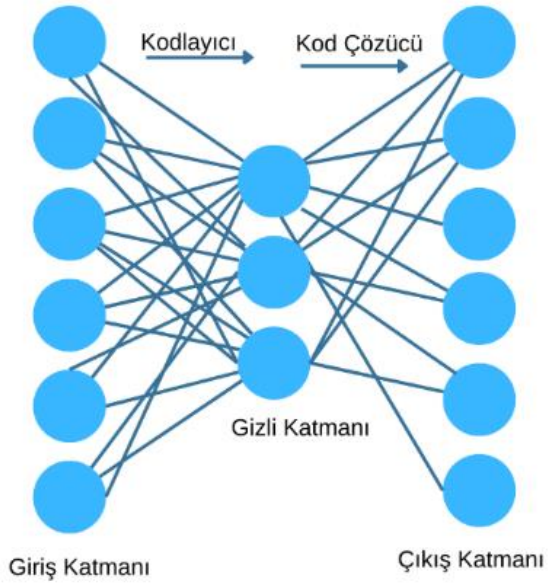
2.1 Otomatik Kodlayıcılar

Otomatik kodlayıcı, gizli katmanda elde edilen temsil gösterimini kullanarak çıktı katmanı üzerinde giriş katmanını yeniden oluşturan ileri beslemeli bir sinir ağıdır. Bir otomatik kodlayıcı giriş katmanı, gizli katman ve çıktı katmanı olmak üzere üç katmandan oluşur. Giriş katmanındaki nöron sayısı, çıkış katmanındaki nöron sayısına eşittir. Öğrenme süreci sırasında otomatik kodlayıcı, kodlayıcı ve kod çözücü olarak adlandırılan iki eşleme kullanır. Kodlayıcı, verileri giriş katmanından gizli katmana eşlerken, kod çözücü, kodlanmış verileri gizli katmandan çıktı katmanına eşler [48]. Öneri sistemlerinde otomatik kodlayıcı iki farklı şekilde uygulanır. Birincisi darboğaz katmanında daha düşük boyutlu özellik temsillerini

öğrenmek için otomatik kodlayıcıyı kullanmak, bir diğeri ise kullanıcı öğe matrisinin boşluklarını doğrudan yeniden yapılandırma katmanına doldurmaktır [43]. Bu çalışmada 4 farklı Otomatik Kodlayıcı kullanılmıştır. Kullanılan modellerin detayları aşağıda sunulmuştur.

2.1.1 Temel otomatik kodlayıcılar

Temel Otomatik Kodlayıcılar, otomatik kodlayıcı modellerin en temel halini oluşturur. Gizli katmandaki nöron sayısı giriş ve çıkış katmanındaki nöron sayısından daima küçük olmalıdır. Buradaki amaç veriyi daha az boyuta düşürerek önemli iç görüleri öğrenmek ve seyrek kullanıcı-öğe matrisini yeniden yapılandırmaktır. Şekil 2.1'de Temel Otomatik Kodlayıcı modelinin yapısı gösterilmiştir.



Şekil 2.1 : Temel Otomatik Kodlayıcı yapısı.

Şekil 2.1'de gördüğümüz Temel Otomatik Kodlayıcı modelinin kod hali Şekil 2.2'deki gibidir. Python kodundaki Input sınıfı giriş katmanını, Dense sınıfları ise gizli ve çıkış katmanını oluşturmaktadır. Dense ve Input katmanlarının içerisinde katmanların nöron sayıları belirtilmiş, aktivasyon fonksiyonu gibi diğer parametreler kütüphanenin default parametresi kullanılarak modellenmiştir.

```
input_movie = Input(shape=(train.shape[1],))
encoded = Dense(100)(input_movie)
output = Dense(train.shape[1])(encoded)

autoencoder = Model(input_movie, output)
autoencoder.compile(optimizer='adam', loss="mse")
```

Şekil 2.2 : Temel Otomatik Kodlayıcı Python kodu.

Şekil 2.1 ve 2.2'de gördüğümüz kodlayıcı yapısı bir aktivasyon işlevinden geçirilen standart sinir ağı işlevi gibi Denklem 2.1 ile temsil edilir.

$$z = \alpha(Wx+b) \tag{2.1}$$

Denklem 2.1'de görüldüğü üzere z gizli katmanı temsil etmektedir. W değeri kodlayıcı ağırlıklarını, x değeri giriş katmanını b değeri kodlayıcı bias değerini α değeri ise kodlayıcı da kullanılan aktivasyon fonksiyonunu temsil eder. Benzer şekilde Şekil 2.1'deki kod çözücü yapısı da kodlayıcı yapısı gibi Denklem 2.2 ile temsil edilir.

$$x' = \alpha'(W'z+b') \tag{2.2}$$

Denklem 2.2'de de görüldüğü üzere z değeri gizli katmanı W' kod çözücünün ağırlık değerlerini, b' kod çözücünün bias değerini ve α' ise kod çözücünün aktivasyon fonksiyonunu temsil eder. Bu formülasyon çıktı olarak x' yani çıkış katmanının değerini verir. Kayıp fonksiyonu ise bu formüller kullanılarak Denklem 2.3 ile ifade edilir.

$$L(x,x') = \|x-x'\|^2 = \|x- \alpha(W'(\alpha(Wx+b))+b')\|^2 \tag{2.3}$$

2.1.2 Gürültü giderici otomatik kodlayıcılar

Gürültü Giderici Otomatik Kodlayıcılar, gürültü içeren verilerden önemli bilgileri öğrenen bir otomatik kodlayıcı türüdür. Gürültü Giderici Otomatik Kodlayıcı'nın amacı gizli katmanı daha sağlam özellikler keşfetmeye zorlamaktır [27]. Yapısı Temel Otomatik Kodlayıcı ile aynıdır. Veri eğitilmeden önce veri üzerine sentetik gürültü eklenir. Eğitim hataları gürültüsüz veri ve otomatik kodlayıcının oluşturduğu veri kullanılarak hesaplanır. Böylelikle otomatik kodlayıcı hem ilgili iç görüleri öğrenirken, aynı zamanda gürültü gidermeyi de öğrenir.

Gürültü oluşturmak için genelde Gauss gürültüsü veya çarpımsal maske çıkış/bırakma gürültüsü kullanılır [27]. Şekil 2.3'te Gürültü Giderici Otomatik Kodlayıcı modelinin Python kodu gösterilmiştir.

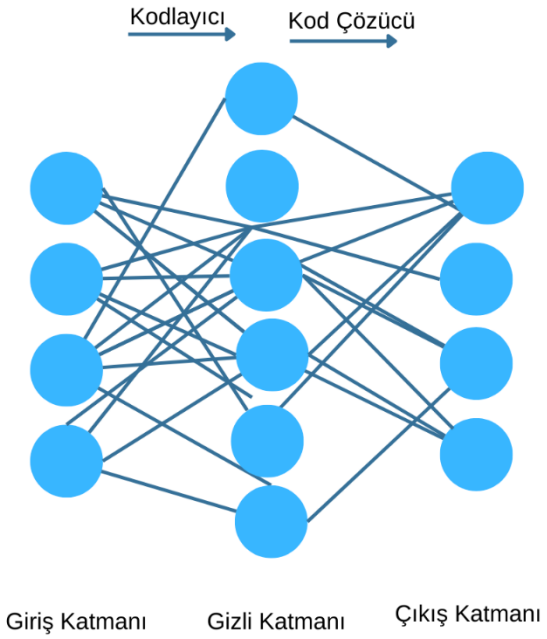
```
train = np.array(train_scaling).astype("float32")
test = np.array(test_scaling).astype("float32")

noise_factor = 0.5
train_noisy = train + noise_factor * np.random.normal(loc=0.0, scale=1.0, size=train.shape)
test_noisy = test + noise_factor * np.random.normal(loc=0.0, scale=1.0, size=test.shape)

train_noisy = np.clip(train_noisy, 0., 1.)
test_noisy = np.clip(test_noisy, 0., 1.)

input_movie = Input(shape=(train_noisy.shape[1],))
encoded = Dense(100)(input_movie)
output = Dense(train_noisy.shape[1])(encoded)

autoencoder = Model(input_movie, output)
autoencoder.compile(optimizer='adam', loss="mse")
```

Şekil 2.3 : Gürültü Giderici Otomatik Kodlayıcı Modeli yapısı.

Şekil 2.3'teki Python kodunda da görüldüğü gibi veriye öncelikle gürültü eklenir. Gürültü eklenen veri otomatik kodlayıcı modelinde eğitilir. Bu eğitim sayesinde model gürültülü verilerden doğru bilgileri öğrenerek doğru çıktıyı elde eder.

2.1.3 Seyrek otomatik kodlayıcılar

Seyrek Otomatik Kodlayıcılar, gizli katmana düğümlerin sadece bir kısmını aktif etmek için ceza terimi (seyreklik kısıtlaması) olarak adlandırılan bir kısıt eklenerek tasarlanan bir otomatik kodlayıcı modelidir. Seyrek Otomatik Kodlayıcılar ceza terimini ekleyerek gizli katmanı giriş katmanından farklı olmaya zorlar [29]. Eklenen ceza terimi, gizli katmanın giriş katmanındaki nöron sayısına eşit ya da daha fazla nöron sayısına sahip olacak şekilde tasarlanmasını ve otomatik kodlayıcının daha yararlı bilgileri öğrenmesini sağlar. Şekil 2.4'te Seyrek Otomatik Kodlayıcı modelinin yapısı gösterilmiştir.



Şekil 2.4 : Seyrek Otomatik Kodlayıcı yapısı.

Seyrek otomatik kodlayıcıların eğitimi, temel otomatik kodlayıcılara benzer şekilde ilerler. Sadece kayıp fonksiyonuna ceza terimi eklenmesi işlemi gerçekleştirilir. Seyrek otomatik kodlayıcı modelinin kayıp fonksiyonunun formülü Denklem 2.4'te verilmiştir.

$$L(x,x') = \|x - x'\|^2 = \|x - \sigma(W'(\sigma(Wx + b)) + b')\|^2 + \text{ceza terimi} \qquad (2.4)$$

Seyrek otomatik kodlayıcı modellerinde üç adet ceza terimi tercih edilir. Bunlar L1, L2 Düzenlemesi ve Kullback Leibler Uzaklığı (KL) dir. L1 Düzenlemesi, ağırlıkların mutlak büyüklük değerini ceza terimi olarak ekleyerek, ceza katsayısını sıfıra düşürme eğilimi gösterir. L2 Düzenlemesi ise ağırlıkların karelerini toplamını ceza terimi olarak ekleyerek ceza katsayısını sıfıra yaklaştırır. KL Uzaklığı ise iki olasılık dağılımını arasındaki farkı ölçer. Bu tez çalışmasında seyrek otomatik kodlayıcılar için L1 Düzenlemesi ceza terimi olarak kullanılmıştır. Şekil 2.5'te Seyrek Otomatik Kodlayıcı Modelinin Python kodu aşağıda verilmiştir.

```
input_movie = Input(shape=(train.shape[1],))
encoded = Dense(6140, activity_regularizer=l1(10e-5))(input_movie)
output = Dense(train.shape[1])(encoded)

autoencoder = Model(input_movie, output)
autoencoder.compile(optimizer='adam', loss="mse")
```

Şekil 2.5 : Seyrek Otomatik Kodlayıcı Modeli Python kodu.

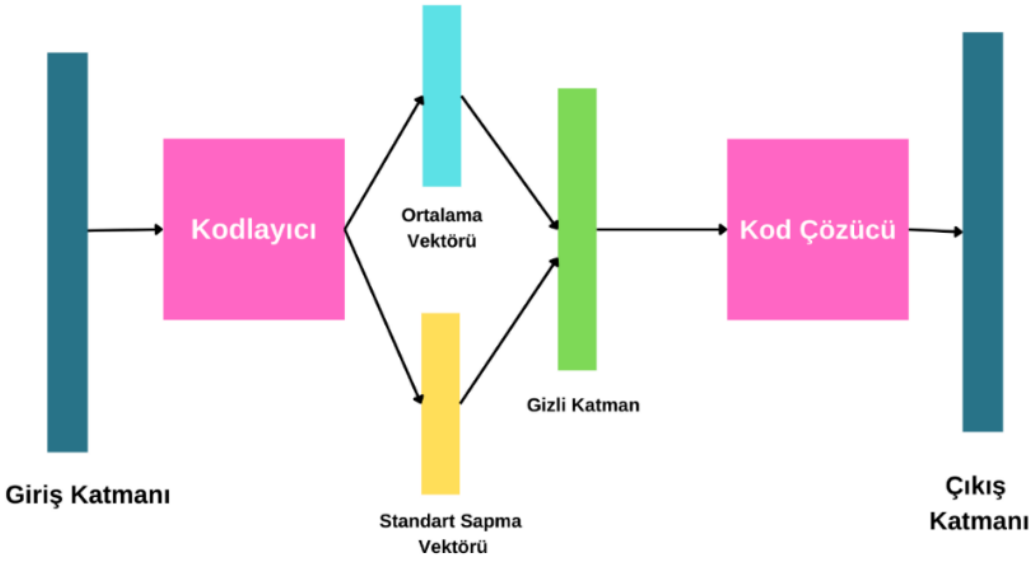
Şekil 2.5'teki Python kodunda da görüldüğü gibi gizli katmana seyrek otomatik kodlayıcı için ceza terimi eklenir. Eklenen ceza terimi otomatik kodlayıcı modelinin gizli katmanı içerisindeki bazı nöronları cezalandırır. Bu cezalandırma işlemi sayesinde model gizli katmanda daha fazla nöron kullanımını sağlar ve daha fazla bilgi öğrenerek doğru çıktıyı elde eder.

2.1.4 Varyasyonel otomatik kodlayıcılar

Varyasyonel Otomatik Kodlayıcılar, gizli katmandaki nöronlardaki verilerin dağılımını normal dağılıma göre dağılmaya zorlayan bir otomatik kodlayıcı modelidir. Varyasyonel Otomatik Kodlayıcılar tanıma modeli ve üretici model olmak üzere iki bölümden oluşur.

Varyasyonel Otomatik Kodlayıcılar, tanıma modelinde giriş katmanındaki veriyi, kodlayıcı yardımıyla ortalama ve standart sapma şeklinde iki ayrı vektöre ayırır ve bu vektörleri kullanarak üretici modelde gizli katman içerisinde giriş katmanındaki verilere benzer rastgele şekilde yeni veriler üretir [30].

Varyasyonel Otomatik Kodlayıcılar her bir gizli katmanda yer alan özellik için bir dağılım tanımlar. Bu özelliği ile Varyasyonel Otomatik Kodlayıcılar diğer otomatik kodlayıcı modellerinden ayrılır. Şekil 2.6'te Varyasyonel Otomatik Kodlayıcı modelinin yapısı gösterilmiştir.



Şekil 2.6 : Varyasyonel Otomatik Kodlayıcı yapısı.

Şekil 2.6'te ifade edilen kodlayıcı ve kod çözücü yapısının formülasyonu Denklem 2.5 ve 2.6'da ifade edilmiş ve Python kodu Şekil 2.7'de gösterilmiştir.

$$P(z|x) = \frac{P(x|z)P(z)}{P(x)} \quad (2.5)$$

$$P(x|z) = \frac{P(z|x)P(x)}{P(z)} \quad (2.6)$$

```
# Create encoder
encoder = keras.Model(inputs, [z_mean, z_log_sigma, z], name='encoder')

# Create decoder
latent_inputs = Input(shape=(latent_dim,), name='z_sampling')
x = Dense(intermediate_dim, activation='relu')(latent_inputs)
outputs = Dense(original_dim, activation='sigmoid')(x)
decoder = Model(latent_inputs, outputs, name='decoder')

# instantiate VAE model
outputs = decoder(encoder(inputs)[2])
vae = Model(inputs, outputs, name='vae_mlp')
```

Şekil 2.7 : Varyasyonel Otomatik Kodlayıcı encoder decoder Python kodu.

Denklem 2.5 ve 2.6'da yer alan x değeri veriyi z değeri ise veriden öğrendiği olasılıksal dağılımı temsil eder. Bu olasılıksal dağılımda örnekleme süreci türevlenebilir değildir. Türevlenebilir hale getirmek için Varyasyonel Otomatik Kodlayıcılar'da yeniden parametrelendirme yöntemi uygulanır. Yeniden

parametrelendirme yöntemi ile z değeri Denklem 2.7'deki gibi hesaplanır. Z değerinin Python kodu ile hesaplanması Şekil 2.8'de gösterilmiştir.

$$z = \mu + \sigma \odot \varepsilon \tag{2.7}$$

```
def sampling(args):
    z_mean, z_log_sigma = args
    epsilon = K.random_normal(shape=(K.shape(z_mean)[0], latent_dim),
                              mean=0., stddev=1)
    return z_mean + K.exp(z_log_sigma/2) * epsilon
```

Şekil 2.8 : z değerinin Python kodu.

Denklem 2.5 ve 2.6'daki P(x) değerinin hesaplanması oldukça zordur ve izlenilebilir bir dağılım olmasını engeller. İzlenilebilir bir dağılım gerçekleştirilebilmek amacıyla p(z|x) ve p(x|z) dağılımlarını birbirine yaklaştırılması hedeflenir. Bu yaklaştırma işleminin gerçekleştirilmesi için kayıp fonksiyonunda KL Uzaklığı eklenir. KL uzaklığı yukarıda belirtildiği gibi iki olasılıksal dağılım arasındaki benzerliği ölçer. Böylelikle Varyasyonel Otomatik Kodlayıcı modelinde kullanılan kayıp fonksiyonu Denklem 2.8'deki gibi temsil edilir ve Python kodu da Şekil 2.9'da gösterilmiştir:

$$\text{Kayıp} = \text{L(x, x')} + \sum_j KL(p_j(z|x)||p(x|z)) \tag{2.8}$$

$$= \text{L(x, x')} + \sum_j P(z|x).log\frac{P(z|x)}{P(x|z)} \tag{2.8a}$$

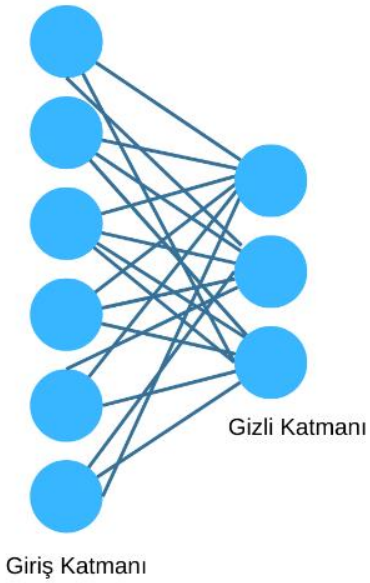
```
reconstruction_loss = K.mean(K.square(inputs - outputs), axis=-1)

kl_loss = 1 + z_log_sigma - K.square(z_mean) - K.exp(z_log_sigma)
kl_loss = K.sum(kl_loss, axis=-1)
kl_loss *= -0.5
vae_loss = K.mean(reconstruction_loss + kl_loss)
```

Şekil 2.9 : Varyasyonel Otomatik Kodlayıcı kayıp fonksiyonu.

2.2 Kısıtlı Boltzmann Makineleri

Kısıtlı Boltzmann Makineleri, katman içerisindeki bağlantıların kaldırıldığı sadece katmanlar arasında bağlantıların bulunduğu bir Boltzmann makinesi türüdür [49]. Kısıtlı Boltzmann Makineleri görünür katman ve gizli katman olarak iki katmandan oluşur. Görünür katmandaki nöron sayısı verinin boyutu ile eşittir.

Gizli katmandaki nöron sayısı daima görünür katmandan daha azdır. Kısıtlı Boltzmann Makinelerinde görünür katman ve gizli katman birbirleri ile bağlantı sağlarlar. Bu bağlantıların oluşturdukları modele simetrik iki parçalı grafik (symmetrical bipartite graph) denir. Şekil 2.10'da Kısıtlı Boltzmann Makineleri modelinin yapısı gösterilmiştir.



Şekil 2.10 : Kısıtlı Boltzmann Makinesi yapısı.

Kısıtlı Boltzmann Makineleri veri seti üzerindeki olasılıksal dağılımları öğrenebilen enerji tabanlı bir modeldir. Zamanın her noktasında Kısıtlı Boltzmann Makineleri belirli bir durumdadır. Bu durum görünür ve gizli katmanlardaki nöronların alacağı değeri ifade eder. Enerji fonksiyonu kullanılarak görünür ve gizli katmanın ortak olasılıkları bulunur. Kısıtlı Boltzmann Makineleri modeli için enerji fonksiyonu ve model içerisinde yer alan görünür ve gizli katmanların ortak olasılığının formülleri Denklem 2.9, 2.10 ve 2.10a'da sunulmuştur.

$$E(v,h) = -\sum_i a_i v_i - \sum_j b_j h_j - \sum_{i,j} v_i h_j w_{ij} \qquad (2.9)$$

$$p(v,h) = \frac{1}{Z} e^{-E(v,h)} \qquad (2.10)$$

$$Z = \sum_{v,h} e^{-E(v,h)} \qquad (2.10a)$$

Denklem 2.9'da tanımlanan enerji modelinde v_i görünür katmanı, h_i gizli katmanı ve w_{ij} ise ağırlıkları temsil eder. a_i ve b_j ise her iki katmanın bias değerlerini temsil eder. Denklem 2.10 ve 2.10a'da hesaplandığı gibi Z değeri tüm olası görünür ve gizli katmanların toplamını temsil eder. Buna bölme fonksiyonu (partition function) ya da normalizasyon sabiti adı da verilir. Görünür ve gizli katmanlardaki nöronların çok sayıdaki olası kombinasyonlarından dolayı her iki yapının ortak olasılığını veren Z değerini hesaplamak zordur. Bu nedenle modeli eğitiminde görünür ve gizli katmanların koşullu olasılıklarının hesaplanması tercih edilir. Denklem 2.11 ve 2.12'da bu koşullu olasılıkların hesaplanma formülleri sunulmuş ve Şekil 2.11'de Python kodu gösterilmiştir.

$$P(h|v) = \prod_i p(h_i|v) \tag{2.11}$$

$$P(v|h) = \prod_i p(v_i|h) \tag{2.12}$$

```
# Phase 1: Input Processing
v0 = tf.placeholder("float", [None, visibleUnits])
_h0 = tf.nn.sigmoid(tf.matmul(v0, W) + hb)  # Visible layer activation
h0 = tf.nn.relu(tf.sign(_h0 - tf.random_uniform(tf.shape(_h0))))  # Gibb's Sampling

# Phase 2: Reconstruction
_v1 = tf.nn.sigmoid(tf.matmul(h0, tf.transpose(W)) + vb)  # Hidden layer activation
v1 = tf.nn.relu(tf.sign(_v1 - tf.random_uniform(tf.shape(_v1))))
h1 = tf.nn.sigmoid(tf.matmul(v1, W) + hb)
```

Şekil 2.11 : Kısıtlı Boltzmann Modeli koşullu olasılık Python kodu.

Denklem 2.11 ve 2.12'de gösterilen formülasyonda belirtilen koşullu olasılıkların hesaplaması klasik sinir ağı formülasyonu ile gerçekleştirilebilir. Fakat tüm olasılıkların ve parametrelerin hesaplanması eğitim sürecini zorlaştırmaktadır. Buna çözüm bulmak amacıyla Zıt Iraksama (Contrastive Divergence) algoritması ve Gibbs Örnekleme yöntemleri geliştirilmiştir.

Gibbs Örnekleme yöntemi, hedef dağılımı bir değişkenin koşullu dağılımından tekrar tekrar örnekleyen bir algoritmadır. Gibbs örneklemesi, diğer tüm değişkenler üzerinde koşullu bir değişkenin koşullu dağılımlarını hesaplanabileceğini ve tam olarak bu dağılımlardan örnek alınabileceğini varsayar. Zıt Iraksama ise, Kısıtlı Boltzmann Makinelerinin ağırlıklarını güncellemek için kullandığı bir algoritmadır. Görünür nöronların değerleri ile görünür nöronların gizli nöronlara olan koşullu olasılığının iç çarpımlarının farkı ile hesaplanır. Yeni hesaplanan ağırlık değeri eski

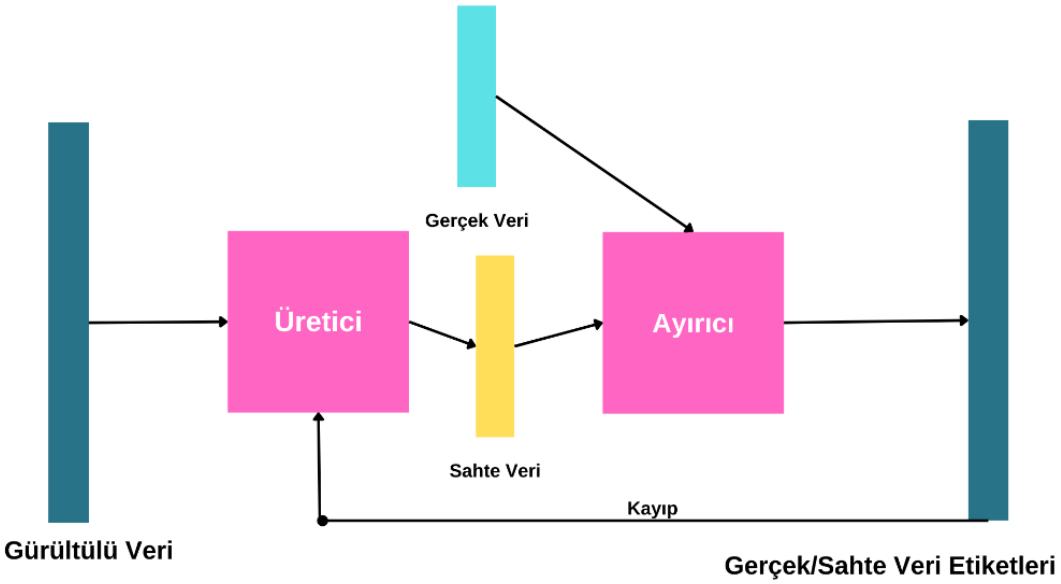
ağırlığın üzerine eklenerek güncelleme işlemi gerçekleştirilir. Zıt Iraksama yönteminin formülü Denklem 2.13 ve 2.14'de sunulmuştur.

$$\Delta W = v_0 \otimes p(h_o|v_0) - v_k \otimes p(h_k|v_k) \quad (2.13)$$

$$W_{new} = W_{old} + \Delta W \quad (2.14)$$

2.3 Üretici Çekişmeli Ağlar

Üretici Çekişmeli Ağlar, Ian Godfellow tarafından geliştirilen bir minimax oyun çerçevesinde aynı anda iki sinir ağını eğiterek gerçek örneklere benzer dağılıma sahip örnekler üretebilen bir yapay sinir ağları modelleri olarak tanımlanır [47]. Üretici Çekişmeli Ağlar, üretici ve ayırıcı olmak üzere iki farklı yapay sinir ağının birleşiminden oluşur. Üretici yapay sinir ağı eğitim verilerine çok benzer veriler üreterek bunu ayırıcı yapay sinir ağına gönderir. Ayırıcı yapay sinir ağı ise üretilen sahte verilerden gerçek verileri ayırt etmeye çalışır. Şekil 2.12'te Üretici Çekişmeli Ağların yapısı ve Şekil 2.13'te Python kodu gösterilmiştir.



Şekil 2.12 : Üretici Çekişmeli Ağlar yapısı.

```
def create_gan(discriminator, generator):
    discriminator.trainable = False
    gan_input = Input(shape=(item_dim,))
    x = generator(gan_input)
    gan_output = discriminator(x)
    gan = Model(inputs=gan_input, outputs=gan_output)
    gan.compile(loss="binary_crossentropy", optimizer="adam")
    return gan
```

Şekil 2.13 : Üretici Çekişmeli Ağlar Python kodu.

Şekil 2.12'deki Üretici Çekişmeli Ağ modeli yapısında ve Şekil 2.13'teki aynı modelin Python kodunda da görüldüğü üzere üretici ağlar gerçek verilere en yakın verileri elde etmek için hata oranını minimize etmeye çalışırken, ayırıcı ağlar gerçek ve sahte verileri daha doğru ayıt ederek üretici ağların hata oranını maksimize etmeye çalışır.

Aynı şekilde ayırıcı ağlar gerçek ve sahte verileri en doğru şekilde ayırt etmek için hata oranını minimize etmeye çalışırken üretici ağlar gerçek verilere en benzer verileri üretmeye çalışarak, ayırıcı ağların hata oranını maksimize etmeye çalışır. Buna min-max oyunu denir ve Üretici Çekişmeli Ağlar'ın kayıp fonksiyonu Denklem 2.15'teki gibi tanımlanır.

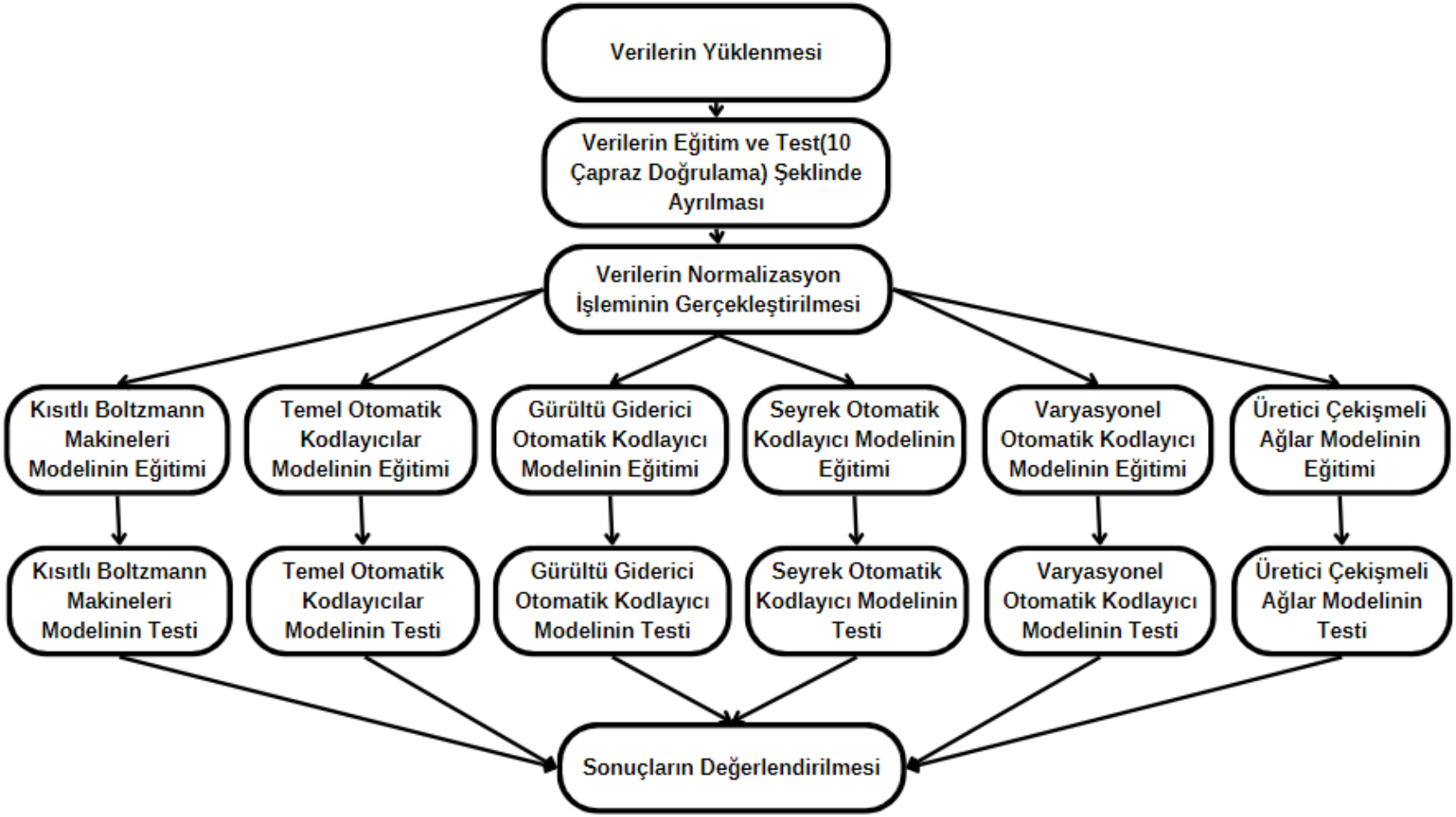
$$\min_{G}\max_{D} V(D,G) = E_{x\sim P_{data}(x)}[logD(x)] + E_{z\sim P_{data}(z)}\left[log\left(1 - D\big(G(z)\big)\right)\right] \quad (2.15)$$

Denklem 2.15'te görüldüğü üzere Üretici ve Çekişmeli Ağlar, kayıp işlevi için ikili çapraz entropiden yararlanırlar. Denklemin birinci parçası gerçek veriyi, ikinci kısmı ise üretici tarafından üretilmiş verinin kayıp işlevini temsil eder. Ayırıcı bu denklemi maksimum değerine çıkarmak isterken, üretici de minimum değerine indirmeye çalışır ve min-max teorisi bu şekilde oluşturulmuş olur.

3. MATERYAL VE YÖNTEM

3.1 Tez İş Akışı

Bu çalışmada derin öğrenme algoritmaların veri seyrekliğine karşı gösterdikleri performansın karşılaştırılması amaçlanmıştır. Bu amaçla kullanılan 6 farklı derin öğrenme algoritması Movielens'in 2 farklı versiyonu ile Jester'ın 3 farklı versiyonu kullanılarak 5 farklı veri setinde eğitim gerçekleştirilmiştir. Her bir modelin eğitim sonuçları çeşitli hata metriklerine göre karşılaştırılmış ve analizleri gerçekleştirilmiş ve performans analizlerinin iş akışı diyagramı Şekil 3.1'deki gibi aşağıda sunulmuştur.



Şekil 3.1 : Tez iş akışı.

Şekil 3.1'deki iş akışı diyagramında da görüldüğü üzere öncelikle veri setleri 10 çapraz doğrulama kullanarak eğitim ve test veri seti olarak iki bölüme ayrılması gerçekleştirilmiştir. İkinci adımda verileri derin öğrenme algoritmaları ile eğitebilmek için veri normalizasyonu işlemi gerçekleştirilmiştir. Burada min-max normalizasyon yöntemi kullanıldı. Üçüncü adımda hazırlanan veri seti derin öğrenme algoritmaları ile eğitim işlemi gerçekleştirilmiştir. Dördüncü bölümde ise test işlemleri gerçekleştirilerek, başarı sonuçları değerlendirilmiştir.

3.2 Veri Seti

Bu tez çalışmasında Movielens ve Jester veri setlerinden oluşan beş farklı veri seti kullanılarak çalışma gerçekleştirilmiştir. Bu bölümde bu veri setleri hakkında ayrıntılı bilgiler anlatılacaktır.

3.2.1 Jester veri seti

Jester, UC Berkeley Üniversitesi Automation Lab laboratuvarı tarafından hazırlanan bir anonim şaka derecelendirme veri setidir. Jester veri setleri çeşitli dönemlerde derecelendirilmiş veri içeren dört farklı gruptan oluşmaktadır. Tez çalışması kapsamında Nisan 1999- Mayıs 2003 arasında oluşturulmuş birinci gruptaki veri seti kullanılmıştır. Birinci gruptaki veri seti 73421 kullanıcıya 100 adet şakayı puanlandırılması istenerek elde edilmiştir. Bu veri seti üç farklı derecelendirilmiş veri seti ve şakaların bulunduğu ayrı bir veri seti ile birlikte içerisinde üç ayrı veri setinden oluştur. Puanlama sistemi -10 ile +10 arasındadır. Derecelendirilmeyen veriler 99 ile ifade edilmiştir. Bu çalışma kapsamında sadece kullanıcı-öğe derecelendirmesi içeren üç veri setinden yararlanılmıştır. Jester veri seti ile ilgili bilgiler aşağıdaki Çizelge 3.1'de sunulmuştur.

Çizelge 3.1 : Jester veri seti ile ilgili bilgiler.

Veri Seti Adı	Kullanıcı Sayısı	Şaka Sayısı	Derecelendirilmiş Veri Sayısı	Seyreklik Oranı
Jester-1	24983	100	1835438	%27
Jester-2	23500	100	1732493	%27
Jester-3	24938	100	641850	%74,5

3.2.2 Movielens veri seti

MovieLens, Minnesota Üniversitesi Bilgisayar Bilimi ve Mühendisliği Bölümü'nde bulunan GroupLens araştırma laboratuvarı tarafından hazırlanan bir kullanıcı-film derecelendirmeleri bulunan bir veri setidir. Bu veri seti içerisinde kullanıcı-film derecelendirmeleri (puan ataması) ile film isimleri, film türleri ve kullanıcı yan bilgileri de bulunmaktadır. Bu çalışmada veri olarak sadece kullanıcı-film derecelendirmeleri kullanılmıştır. MovieLens projesi içerisinde birçok boyutta derecelendirilmiş veri setleri bulunmaktadır. Bu çalışmada bu veri setleri arasından MovieLens-100K ve Movielens-1M veri seti kullanılmıştır. Her iki veri seti için de puanlama sistemi 1-5 arasında yapılmıştır. Movielens veri seti ile ilgili bilgiler aşağıdaki Çizelge 3.2'de sunulmuştur.

Çizelge 3.2 : Movielens veri seti ile ilgili bilgiler.

Veri Seti Adı	Kullanıcı Sayısı	Film Sayısı	Derecelendirilmiş Veri Sayısı	Seyreklik Oranı
Movielens-100K	943	1682	100000	%93,6
Movielens-1M	6040	3706	1000000	%95,4

3.3 Değerlendirme Metrikleri

Performansları değerlendirmek için öneri sistemlerini karşılaştırmada en çok faydalanılan metriklerden Kök Ortalama Kare Hatası (Root Mean Square Error-RMSE) ve Ortalama Mutlak Hata (Mean Absolute Error-MAE) metrikleri kullanılmıştır. Öneri sistemlerinde RMSE ve MAE doğruluk metrikleri kategorisi içerisindeki tahmine dayalı doğruluk metrikleri içerisinde yer alır. Tahmine dayalı doğruluk metrikleri kullanıcıların gerçek derecelendirmelerinin doğruluğunu ölçen metriklerdir. Bu metrikler, öneri listesindeki konumlarından bağımsız olarak tüm derecelendirmeleri aynı şekilde ele alır. RMSE ve MAE, genellikle bir öneri sisteminin genel performansını ölçmek için kullanılır. RMSE ve MAE sırasıyla Denklem 3.1 ve Denklem 3.2'de gösterildiği gibi hesaplanır:

$$\text{RMSE} = \sqrt{\frac{1}{n}\sum_{i=1}^{n}(x_i - x'_i)^2} \quad (3.1)$$

$$\text{MAE} = \frac{1}{n}\sum_{i=1}^{n}|x_i - x'_i| \quad (3.2)$$

Denklem 3.1'de x_i gerçek değeri x_i' ise tahmin edilen değeri temsil etmektedir. İlk olarak gerçek değer ve tahmin edilen değer arasındaki farkların ortalaması alınır, ardından ortalamanın karekökü alınır. Denklem 3.2'de ise Denklem 3.1'deki gibi x_i gerçek değeri x_i' ise tahmin edilen değeri temsil etmektedir. MAE'de gerçek değer ve tahmin edilen değerin farklarının mutlak değeri alındıktan sonra ortalaması alınır.

3.4 Çalışma Ortamı

Tez çalışması boyunca performansları karşılaştırılan tüm derin öğrenme modelleri Google Colaboratory ortamında, 2.4 GHz Intel i5-6200U işlemciye ve 12 GB RAM belleğine sahip olan sistem üzerinden elde edilmiştir. Google Colaboratory, Google tarafından hizmete sunulan derin öğrenme uygulamaları geliştirmek için yüksek GPU kullanma imkanı sağlayan ücretsiz bir bulut hizmetidir. İçerisinde birçok makine ve derin öğrenme kütüphanelerini barındırmaktadır. Python ve R programlama dillerini desteklemektedir. Bu tez çalışması kapsamında Python programlama dili kullanılmıştır. Derin öğrenme modellerinin geliştirilmesinde Python içerisinde bulunan Keras kütüphanesi kullanılmıştır.

Keras kütüphanesi Tensorflow üzerinde çalışan ve neredeyse her derin öğrenme modelinin geliştirilebilmesini sağlayan bir kütüphanedir. Keras, Tensorflow yanı sıra Microsoft Cognitive Toolkit, Theano ve PlaidML üzerinde de çalışabilen bir derin öğrenme API'sidir. Keras kütüphanesi Python dilinin yansıra C++, Java, C#, JavaScript ve R gibi birçok dili desteklemektedir. Keras bugün kendi geliştiricisi Google dışında Microsoft, Amazon, Apple, Nvidia, Uber gibi şirketler tarafından da desteklenmektedir. Keras kütüphanesi, derin öğrenme algoritmalarının kolay bir şekilde oluşturularak hızlı deneyler yapılabilmesi için tasarlanmıştır. Keras'ın çalışma akışında öncelikle eğitim verisinin tanımlanması yatar. Ardından katmanlar ve modeller tanımlanır ve hiper parametreler eklenir. Keras içerisinde çok fazla hiper parametre barındırmasının yanı sıra hiper parametreleri ihtiyaca göre kolaylıkla özelleştirebilme imkanı sağlar. Son adım olarak eğitim verisi ile modeli besler ve modelin çıktılarını verir.

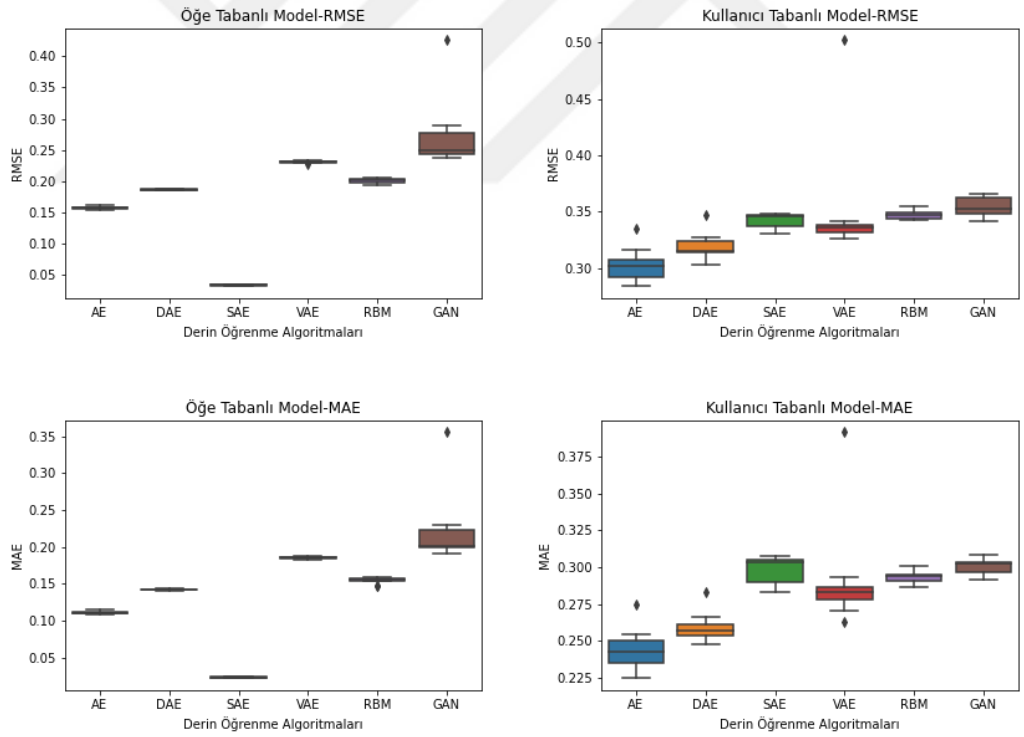
4. BULGULAR VE TARTIŞMA

Bu tez çalışmasında dört farklı otomatik kodlayıcı, Kısıtlı Boltzmann Makineleri ve Üretici Çekişmeli Ağlar olmak üzere altı derin öğrenme algoritmasının öneri sistemlerindeki veri seyrekliği problemine karşı performans sonuçları elde edilmiştir.

Bu bölümde tez çalışmaları sonucunda elde edilen sonuçların karşılaştırmalı analizleri sunulacaktır.

4.1 Jester-1 Veri Setine Ait Sonuçlar

Jester-1 veri seti üzerinde iki farklı değerlendirme metriği kullanılarak altı farklı algoritmanın eğitilerek gerçekleştirilen analiz sonuçları kutu grafiği kullanılarak aşağıda Şekil 4.1'de sunulmuştur. Her bir algoritmanın değerlendirme metrikleri sonunda alınan analizlerin ortalaması ise Çizelge 4.1'de gösterilmiştir.



Şekil 4.1 : Jester-1 veri setinin karşılaştırılması.

Şekil 4.1'teki kutu grafiklerinden hareketle öğe tabanlı modellerin veri dağılımlarının kullanıcı tabanlı modellere göre daha homojen dağıldığını gözlemlenmiştir. Öğe tabanlı modellerde üretici çekişmeli ağlar algoritmasındaki ayrık değerlerin varlığı seyrek verileri yapılandırmada daha fazla hatayla karşılaştığını göstermektedir.

Öğe tabanlı modellerde Gürültü Giderici Otomatik Kodlayıcılar ile Kısıtlı Boltzmann Makineleri benzer sonuçlar göstermiştir. Kullanıcı tabanlı modellerde Temel Otomatik Kodlayıcılar, Gürültü Giderici Otomatik Kodlayıcılar seyrek veriyi yapılandırmada diğer modellere göre daha fazla hatayla karşılaşmıştır.

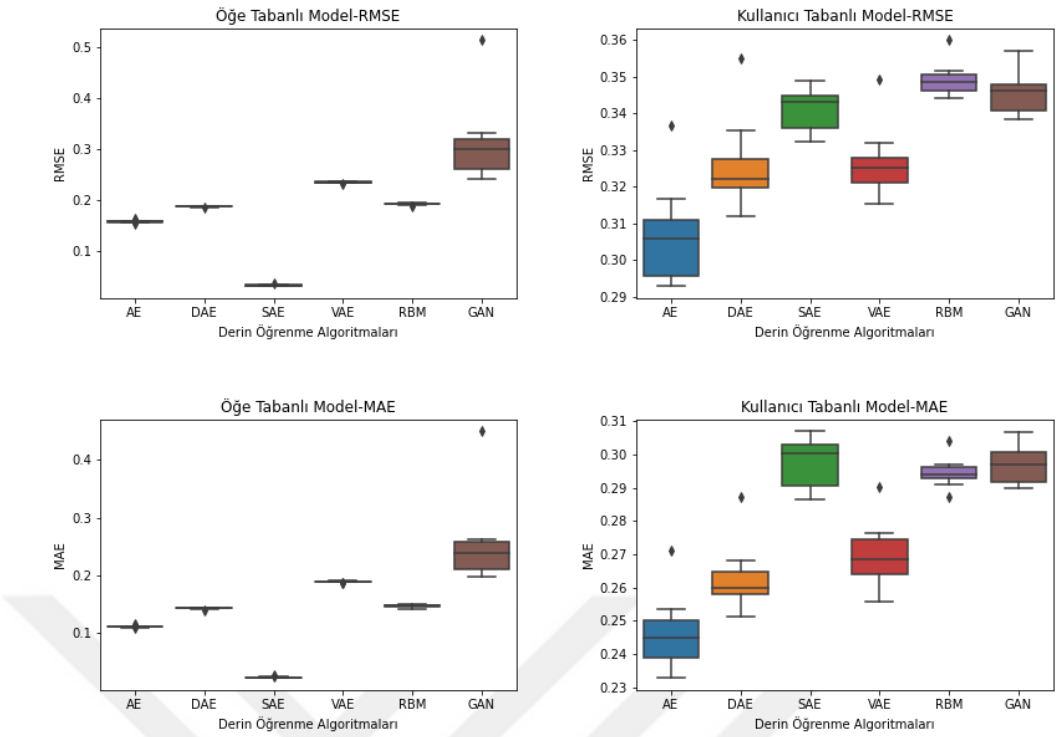
Çizelge 4.1 : Jester-1 veri seti sonuçları.

Model Adı	Öğe Tabanlı		Kullanıcı Tabanlı	
	RMSE	MAE	RMSE	MAE
AE	0.1576	0.1115	0.3027	0.2436
DAE	0.1864	0.1424	0.3185	0.2589
SAE	0.0337	0.0232	0.3421	0.2983
VAE	0.2308	0.1850	0.3514	0.2912
RBM	0.2001	0.1548	0.3467	0.2932
GAN	0.2734	0.2213	0.3541	0.3008

Çizelge 4.1'de de Jester-1'in MAE ve RMSE ortalamalarında gözlemlendiği gibi öğe tabanlı modeller kullanıcı tabanlı modellere göre daha fazla performans göstermiştir. Jester-1 veri seti için öğe tabanlı modellerde en başarılı algoritma her iki değerlendirme metriği için de Seyrek Otomatik Kodlayıcılar olduğu gözlemlenmiştir. Kullanıcı tabanlı modeller için ise Temel Otomatik Kodlayıcılar her iki değerlendirme metriğine göre en başarılı algoritma olmuştur.

4.2 Jester-2 Veri Setine Ait Sonuçlar

Jester-2 veri seti kullanılarak iki farklı değerlendirme metriği kullanılarak altı farklı algoritmanın eğitilerek gerçekleştirilen analiz sonuçları kutu grafiği kullanılarak aşağıda Şekil 4.2'de sunulmuştur. Her bir algoritmanın değerlendirme metrikleri sonunda alınan analizlerin ortalaması ise Çizelge 4.2'de gösterilmiştir.



Şekil 4.2 : Jester-2 veri setinin karşılaştırılması.

Şekil 4.2'deki kutu grafiklerinden hareketle öğe tabanlı modellerde Jester-1 ile Jester-2 veri setlerinin birbirine yakın sonuçlar gözlemlenmiştir. Her iki veri setinin birbirine yakın seyreklikte veriler içermesi bu sonucun gerçekleşmesinde önemli bir rol oynayabilir. Kullanıcı tabanlı modelde Temel Otomatik Kodlayıcılar, Gürültü Giderici Otomatik Kodlayıcılar, Varyasyonel Otomatik Kodlayıcılar ve Kısıtlı Boltzmann Makineleri seyrek veriyi yapılandırmada diğer modellere göre daha fazla hatayla karşılaşmıştır.

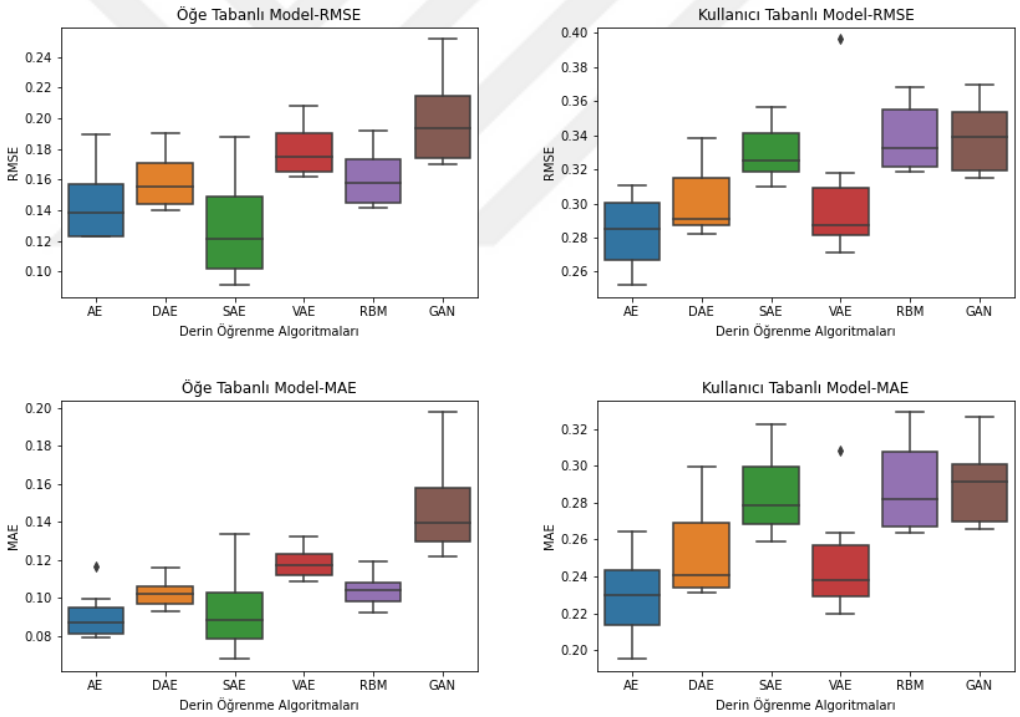
Çizelge 4.2 : Jester-2 veri seti sonuçları.

Model Adı	Öğe Tabanlı		Kullanıcı Tabanlı	
	RMSE	MAE	RMSE	MAE
AE	0.1587	0.1116	0.3067	0.2463
DAE	0.1884	0.1431	0.3260	0.2627
SAE	0.0339	0.0239	0.3410	0.2976
VAE	0.2357	0.1892	0.3265	0.2698
RBM	0.1937	0.1469	0.3490	0.2945
GAN	0.3102	0.2528	0.3453	0.2969

Çizelge 4.2'de MAE ve RMSE ortalamalarında da gözlemlendiği gibi öğe tabanlı modeller kullanıcı tabanlı modellere göre daha fazla performans göstermiştir. Jester-2 veri seti için öğe tabanlı modellerde en başarılı algoritma Jester-1 veri setinde olduğu gibi Seyrek Otomatik Kodlayıcılar olduğu gözlemlenmiştir. Kullanıcı tabanlı modeller için ise Jester-1'de gözlemlendiği gibi Temel Otomatik Kodlayıcılar her iki değerlendirme metriğine göre en başarılı algoritma olmuştur.

4.3 Jester-3 Veri Setine Ait Sonuçlar

Jester-3 veri seti kullanılarak iki farklı değerlendirme metriği kullanılarak altı farklı algoritmanın eğitilerek gerçekleştirilen analiz sonuçları kutu grafiği kullanılarak aşağıda Şekil 4.3'te sunulmuştur. Her bir algoritmanın değerlendirme metrikleri sonunda alınan analizlerin ortalaması ise Çizelge 4.3'te gösterilmiştir.



Şekil 4.3 : Jester-3 veri setinin karşılaştırılması.

Şekil 4.3'teki kutu grafiklerinden hareketle Jester-3 veri setinin öğe tabanlı modellerde daha iyi performans gösterdiği gözlemlenmiştir. Öğe tabanlı modellerde Gürültü Giderici Otomatik Kodlayıcılar ve Kısıtlı Boltzmann Makineleri'nde birbirine yakın değerler gözlemlenmiştir. Temel Otomatik Kodlayıcılar ile Seyrek Otomatik Kodlayıcılar da birbirine daha yakın değerler gözlemlese de Temel Otomatik Kodlayıcılardaki aykırı değerler model performansını etkilemektedir.

Kullanıcı tabanlı modellerde Temel Otomatik Kodlayıcılar ve Varyasyonel Otomatik Kodlayıcılar ile Seyrek Otomatik Kodlayıcılar ve Kısıtlı Boltzmann Makineleri'nde birbirlerine yakın değerler gözlemlenmiştir. Varyasyonel Otomatik Kodlayıcılarda gözlemlenen aykırı değerler, algoritmanın performansını etkilemektedir.

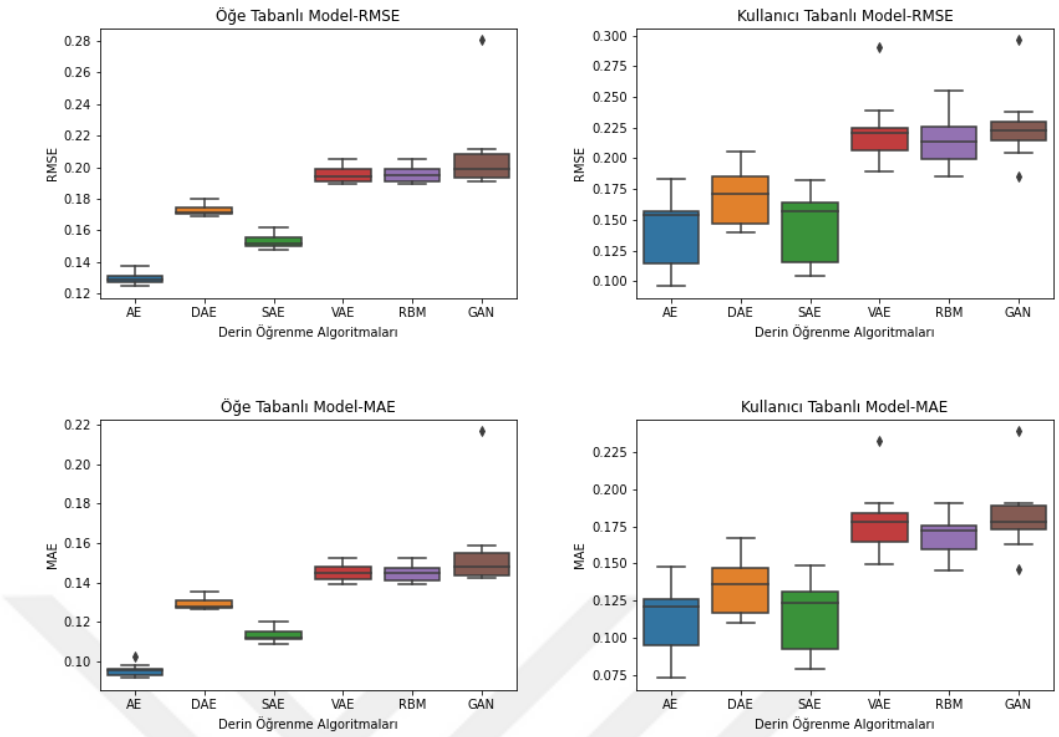
Çizelge 4.3 : Jester-3 veri seti sonuçları.

Model Adı	Öğe Tabanlı		Kullanıcı Tabanlı	
	RMSE	MAE	RMSE	MAE
AE	0.1431	0.0900	0.2837	0.2297
DAE	0.1586	0.1025	0.3012	0.2524
SAE	0.1283	0.0935	0.3308	0.2835
VAE	0.1785	0.1181	0.3013	0.2457
RBM	0.1600	0.1040	0.3387	0.2889
GAN	0.1981	0.1473	0.3383	0.2901

Çizelge 4.3'te MAE ve RMSE ortalamalarında da gözlemlendiği gibi öğe tabanlı modeller kullanıcı tabanlı modellere göre daha fazla performans göstermiştir. Jester-3 veri seti için öğe tabanlı modellerde RMSE metriği için Seyrek Otomatik Kodlayıcılar MAE metriği için ise Temel Otomatik Kodlayıcılar olduğu gözlemlenmiştir. Kullanıcı tabanlı modeller için ise Temel Otomatik Kodlayıcılar her iki değerlendirme metriğine göre en başarılı algoritma olmuştur.

4.4 Movielens-100K Veri Setine Ait Sonuçlar

Movielens-100K veri seti üzerinde iki farklı değerlendirme metriği kullanılarak altı farklı algoritmanın eğitilerek gerçekleştirilen analiz sonuçları kutu grafiği kullanılarak aşağıda Şekil 4.4'te sunulmuştur. Her bir algoritmanın değerlendirme metrikleri sonunda alınan analizlerin ortalaması ise Çizelge 4.4'te gösterilmiştir.



Şekil 4.4 : Movielens-100K veri setinin karşılaştırılması.

Şekil 4.4'deki kutu grafiklerinden hareketle öğe tabanlı modellerin kullanıcı tabanlı modellere göre daha homojen dağılıma sahip olduğunu gözlemlenmektedir. Öğe tabanlı modellerde Varyasyonel Otomatik Kodlayıcılar ve Kısıtlı Boltzmann Makineleri'nde benzer sonuçlar gözlemlenmiştir. Üretici Çekişmeli Ağlar'da aykırı değerler gözlemlenmiştir. Kullanıcı tabanlı modellerde Temel Otomatik Kodlayıcılar ve Seyrek Otomatik Kodlayıcılar'da benzer sonuçlar gözlemlendiği gibi Varyasyonel Otomatik Kodlayıcılar ve Üretici Çekişmeli Ağlar'da da benzer sonuçlar ve aykırı değerler gözlemlenmektedir.

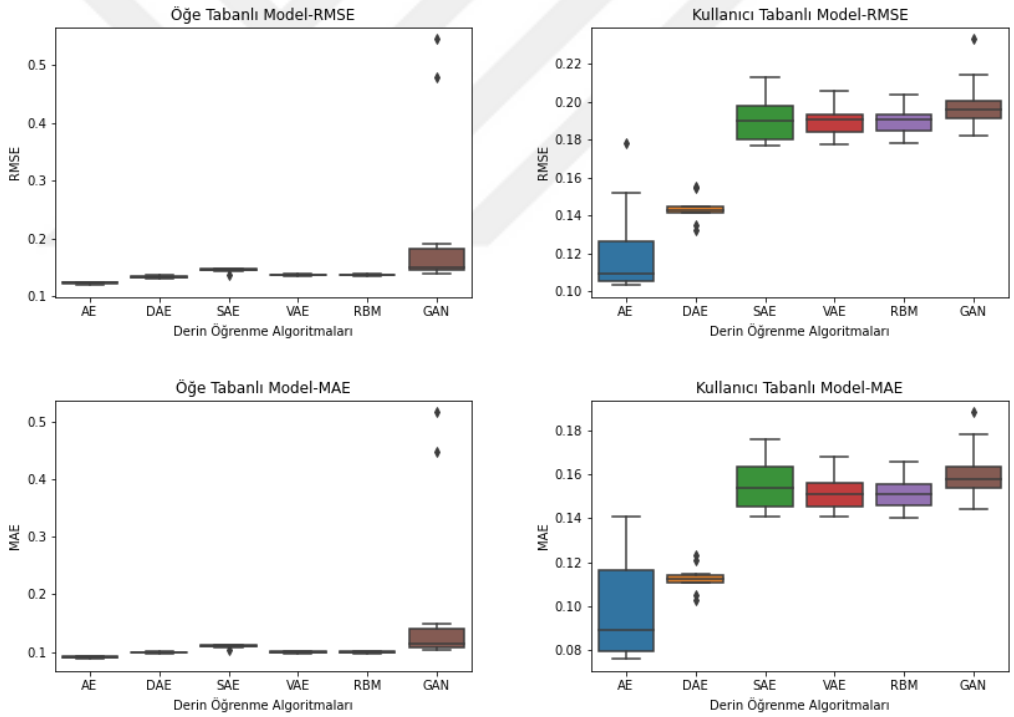
Çizelge 4.4 : Movielens100K veri seti sonuçları.

Model Adı	Öğe Tabanlı		Kullanıcı Tabanlı	
	RMSE	MAE	RMSE	MAE
AE	0,1295	0,0954	0,1398	0,1112
DAE	0,1728	0,1292	0,1679	0,1336
SAE	0,1531	0,1133	0,1455	0,1161
VAE	0,1954	0,1450	0,2218	0,1787
RBM	0.1955	0.1445	0.2147	0.1692
GAN	0.2071	0.1549	0.2256	0.1816

Çizelge 4.4'te MAE ve RMSE ortalamalarında da gözlemlendiği gibi öğe tabanlı modeller ve kullanıcı tabanlı modellerin performansları algoritmaya göre değişmektedir. Movielens-100K veri seti için öğe tabanlı modellerde her iki metrik için en başarılı model Temel Otomatik Kodlayıcılar olduğu gözlemlenmiştir. Kullanıcı tabanlı modeller için ise Temel Otomatik Kodlayıcılar her iki değerlendirme metriğine göre en başarılı algoritma olmuştur.

4.5 Movielens-1M Veri Setine Ait Sonuçlar

Movielens-1M veri seti kullanılarak iki farklı değerlendirme metriği kullanılarak altı farklı algoritmanın eğitilerek gerçekleştirilen analiz sonuçları kutu grafiği kullanılarak aşağıda Şekil 4.5'te sunulmuştur. Her bir algoritmanın değerlendirme metrikleri sonunda alınan analizlerin ortalaması ise Çizelge 4.5'te gösterilmiştir.



Şekil 4.5 : Movielens-1M veri setinin karşılaştırılması.

Şekil 4.5'teki kutu grafiklerinden hareketle öğe tabanlı modellerin kullanıcı tabanlı modellere göre daha homojen dağılım gösterdiği gözlemlenmiştir. Üretici Çekişmeli Ağlar öğe tabanlı modellerde ayrık değerler gözlemlendiğinden hata oranı diğer modellere göre fazladır. Kullanıcı tabanlı modellerde Temel Otomatik Kodlayıcılar, Gürültü Giderici Otomatik Kodlayıcılar ve Üretici Çekişmeli Ağlar'da aykırı

değerler gözlemlenmiştir. Seyrek Otomatik Kodlayıcı, Varyasyonel Otomatik Kodlayıcı ve Kısıtlı Boltzmann Makineleri'nde birbirine yakın sonuçlar vermiştir.

Çizelge 4.5 : Movielens-1M veri seti sonuçları.

Model Adı	Öğe Tabanlı		Kullanıcı Tabanlı	
	RMSE	MAE	RMSE	MAE
AE	0,1226	0,0909	0,1220	0,0980
DAE	0,1332	0,0996	0,1437	0,1128
SAE	0,1454	0,1098	0,1917	0,1557
VAE	0,1366	0,1000	0,1900	0,1519
RBM	0.1367	0.0999	0.1899	0.1514
GAN	0.2239	0.1882	0.1991	0.1606

Çizelge 4.5'te MAE ve RMSE ortalamalarında da gözlemlendiği gibi öğe tabanlı modeller ve kullanıcı tabanlı modellerin performansları algoritmaya göre değişmektedir. Movielens-1M veri seti için öğe tabanlı modellerde her iki metrik için en başarılı model Temel Otomatik Kodlayıcılar olduğu gözlemlenmiştir. Kullanıcı tabanlı modeller için ise Temel Otomatik Kodlayıcılar her iki değerlendirme metriğine göre en başarılı algoritma olmuştur.

5. SONUÇLAR VE ÖNERİLER

Öneri sistemleri kullanıcıların geçmiş tercih ve davranışlarına göre gelecekteki tercihlerini tahmin eden sistemlerdir. Öneri sistemleri işbirlikçi filtreleme, içerik tabanlı demografi tabanlı, popülarite tabanlı ve hibrit öneri sistemleri olmak üzere beş farklı şekilde tasarlanabilir. İşbirlikçi filtreleme, kullanıcıların veya öğelerin tercihlerini benzer kullanıcılar veya öğelerle karşılaştırarak tahmin etme yöntemidir. İşbirlikçi filtreleme kendi içerisinde kullanıcı tabanlı ve öğe tabanlı olmak üzere ikiye ayrılmaktadır. Kullanıcı tabanlı işbirlikçi filtreleme kullanıcılar arasındaki ilişkileri belirleyerek bir öneri sunar. Öğe tabanlı işbirlikçi filtreleme ise öğeler arasındaki ilişkileri belirleyerek bir öneri sunar. İçerik temelli yaklaşım, benzer içeriğe sahip bir öğenin, öğe içeriğine göre tavsiye edilmesidir. Demografik tabanlı öneri sistemleri kullanıcıların demografik yapılarına göre öneri sunan sistemlerdir. Popülarite tabanlı sistemler kullanıcıların tercihlerini kullanmadan sistemdeki popüler öğenin önerilmesiyle oluşturulan sistemlerdir. Hibrit öneri sistemi ise, işbirlikçi filtreleme ile içerik temelli bir yaklaşımı birleştiren bir yöntemdir. Bu tez çalışmasında işbirlikçi filtreleme yöntemleri kullanılmıştır.

Öneri sistemleri verileri modellerken farklı zorluklarla karşılaşırlar. Bunlar doğruluk, soğuk başlama, veri seyrekliği, ölçeklenebilirlik ve güvenilebilirliktir. Bu tez kapsamında veri seyrekliği problemi incelenmiştir. Veri seyrekliği genellikle sistemdeki yetersiz bilgi nedeniyle ortaya çıkan ve işbirlikçi filtreleme yöntemlerini etkileyen bir sorundur. Öneri sistemlerinde veri seyrekliğini engellemek için iki farklı yöntem kullanılır. Bunlardan ilki seyrek kullanıcı matrisini yeniden yapılandırmak, diğeri ise var olan kullanıcı-öğe matrisi içerisindeki derecelendirmeleri kullanarak seyrek verileri yeniden yapılandırmaktır.

Seyrek matrisi yeniden yapılandırmak için kullanıcı yan bilgilerinden faydalanılarak eksik veriler yapılandırılmaya çalışılır. Fakat bu yöntem öneri sistemi modelleri daha karmaşık hale getirir. Ayrıca veri gizliliği nedeniyle her zaman kullanıcı yan bilgilerini kullanmak mümkün olmaz. Derin öğrenme algoritmaları kullanıcı yan bilgilerini kullanmaya ihtiyaç duymadan sadece kullanıcı-öğe matrisinden etkili temsilleri öğrenerek seyrek derecelendirme matrisini yeniden yapılandırır. Derin öğrenme algoritmaları, aktivasyon fonksiyonlarını kullanılarak lineer olmayan

yapıları kolaylıkla modeller. Bu özellik, karmaşık etkileşimleri kolaylıkla çözümlemeyi sağlar ve kullanıcının tercihini tam olarak yansıtır.

Bu tez çalışması kapsamında farklı nöron sayıları ve mimarilerle oluşturulmuş derin öğrenme algoritmalarının seyrek verilerden oluşan öneri sistemleri verisi üzerinde başarılarını değerlendirmek hedeflenmiştir. Bu alanda regresyon tabanlı, kümeleme tabanlı işbirlikçi yaklaşımlarından öte verileri anlayan ve modelleyen mimarilerin seyrek veriler üzerindeki başarıları göz önüne alınarak çalışma gerçekleştirilmiştir.

Çalışma kapsamında hem kullanıcı tabanlı hem de öğe tabanlı işbirlikçi filtreleme yöntemleri kullanılmış ve bu yöntemlerin de veri seyrekliğine karşı derin öğrenme algoritmalarındaki davranışları incelenmiştir.

Tez çalışması kapsamında Movielens ve Jester veri setlerinden oluşan seyreklikleri birbirinden farklı beş farklı veri seti kullanılmıştır. Değerlendirme metriği olarak tahmine dayalı doğruluk metriklerinden olan RMSE ve MAE metrikleri kullanılmıştır. 10 Kat Çapraz Doğrulama ile eğitilen modeller, her kat çapraz doğrulama için RMSE ve MAE sonuçları alınmış, bu sonuçların dağılımları kutu grafikleri kullanılacak sunulmuş, aynı zamanda çapraz doğrulama sonuçlarının da ortalamaları tez çalışmasında sunulmuştur.

Bu çalışma kapsamında RMSE ve MAE metriklerinin tercih edilmesinin nedeni yeniden yapılandırılan derecelendirmelerin gerçek derecelendirmelere ne kadar yakın bir dağılımda olduğunu keşfetmek ve algoritmaların öneri performansını ölçmektir.

Yapılan analizlerde elde edilen sonuçlar her iki veri setinde farklılık göstermektedir. Jester veri setleri kullanılarak geliştirilen öğe tabanlı modeller, kullanıcı tabanlı modellere göre daha fazla başarı gösterdiği gözlemlenmiştir. Jester-1 ve Jester-2 veri setlerinin seyreklikleri birbirine yakın olduğu için benzer sonuçlar göstermiştir. Jester-3 veri seti diğer iki veri setine karşın daha seyrek veri seti olmasına rağmen değerlendirme metriği sonuçlarına göre diğer iki veri setinden daha başarılı sonuçlar vermiştir. Fakat çapraz doğrulama sonuçlarının veri dağılımı karşılaştırıldığında Jester veri setinde de seyreklik arttıkça heterojenlik artmıştır. Bu noktada Jester-3 veri seti kullanılarak geliştirilen model, diğer veri setlerine göre daha hatalı öneriler sunacaktır. Jester veri setlerinde seyreklik konusunda derin öğrenme algoritmaları her ne kadar iyi sonuçlar gösterse de seyreklik miktarı arttıkça algoritmaların doğru sonuçları verme oranı azalmaktadır. Fakat yine de düşük RMSE ve MAE sonuçları

alındığından derin öğrenme algoritmaların Jester verinin veri seyrekliğine karşı iyi performans gösterdiği çalışma sonucunda gözlemlenmiştir.

Jester veri seti üzerinde yapılan analizlere göre öğe tabanlı modellerde en başarılı algoritma Seyrek Otomatik Kodlayıcılar olurken, kullanıcı tabanlı modellerde en başarılı algoritmanın Temel Otomatik Kodlayıcılar olduğu gözlemlenmiştir. Literatürde Jester veri seti kullanılarak yapılmış çok fazla çalışma bulunmamaktadır. Literatüre katkı sağlamak amacıyla Jester veri seti kullanılarak derin öğrenme tabanlı öneri sistemleri üzerine çalışmalar gerçekleştirilebilir.

Movielens veri setlerinde ise kullanıcı tabanlı ve öğe tabanlı modellerin performansları algoritmaya göre değişiklik gösterse de çok büyük bir performans farkı olmadığı gözlemlenmiştir. Fakat her iki öneri sistemi yönteminin Movielens-100K ve Movielens-1M çapraz doğrulama sonuçlarının veri dağılımı incelendiğinde, kullanıcı tabanlı öneri sistemlerinde veri dağılımının daha heterojen olduğu gözlemlenmiştir. Çapraz doğrulama sonuçlarının ortalamaları alındığında her iki yöntemde aralarında az fark gözükse de veri dağılımı bize kullanıcı tabanlı modellerin, öğe tabanlı modellere göre daha fazla hatayla öneri verebileceğini göstermektedir.

Yapılan çalışma öncesinde Movielens-1M veri setinin seyreklik oranının Movielens-100K veri setine göre daha yüksek olmasından dolayı sonuçların daha Movielens-1M'da Movielens-100K'ya göre başarı oranının daha düşük olması beklenmekteydi. Fakat analiz sonuçlarında Movielens1-M veri setinin Üretici Çekişmeli Ağlar dışında beş algoritmada Movielens-100K'ya göre daha iyi performans gösterdiği gözlemlenmiştir. İki veri setini kendi içerisinde incelendiğinde içerdikleri kayıp olmayan veri miktarının Movielens-1M veri setinde Movielens-100K veri setinden daha fazla olduğu görülmektedir. Bu noktada algoritmayı besleyen veri miktarı arttıkça veri seyrekliğinin getirdiği zorluklar derin öğrenme algoritmaları tarafından aşıldığı yapılan analizler sonucu gözlemlenmiştir.

Movielens veri setinde her iki modelde ve değerlendirme metriğinde en başarılı performans gösteren algoritmanın Temel Otomatik Kodlayıcılar olduğu gözlemlenmiştir. Ferreira ve diğ., [7] Movielens-1M için yaptığı çalışmada kullandığı derin otomatik kodlayıcılar, RMSE metriğinde 0.029 değerini elde ederek tez çalışmasında kullanılan Temel Otomatik kullanıcıların sonuçlarına göre daha

başarılı sonuçlar göstermiştir. Ferreira ve ekibi çalışmalarında derin otomatik kodlayıcıları tercih etmiş ve hiper parametre yöntemlerine başvurmuştur. Buradan hareketle Temel Otomatik Kodlayıcılar'ın hiper parametre, katman ve nöron sayısı arttırılarak veri seyrekliğine karşı performansının artışı sağlanabilir.

Tez çalışması kapsamında, Seyrek Otomatik Kodlayıcılar'ın Temel Otomatik Kodlayıcı'ya göre daha iyi performans göstermesi beklenmekteydi. Fakat analiz sonuçları Temel Otomatik Kodlayıcılar'ın performansının Seyrek Otomatik Kodlayıcılar'a göre daha iyi olduğunu göstermektedir. Seyrek Otomatik Kodlayıcılar'a yapılacak farklı düzenleştirme parametresi denemeleri ile algoritmanın sonuçlarının iyileştirilmesi sağlanabilir.

Kısıtlı Boltzmann Makineleri'nin bu çalışmada en az başarı gösteren algoritma olması beklenirken, Üretici Çekişmeli Ağlar'dan daha başarılı sonuçlar verdiği gözlemlenmiştir. Otomatik kodlayıcılarda ise veri setine ve kullanıcı ya da öğe tabanlı öneri sistemi olmasına göre bu durum değişmektedir. Fakat genel olarak otomatik kodlayıcılar, Kısıtlı Boltzmann Makineleri'ne göre daha fazla performans gösterdiği gözlemlenmiştir. Sedhain ve diğ., [23] Temel Otomatik Kodlayıcı kullanarak yaptığı çalışmada aynı zamanda Kısıtlı Boltzmann Makineleri ile sonuçları karşılaştırmıştır. Sonuçlar Temel otomatik kodlayıcıların Kısıtlı Boltzmann Makineleri'ne göre daha iyi başarılı olduğunu göstermektedir. Ayrıca Sahoo ve Pradhan'ın [36] Kısıtlı Boltzmann Makineleri ve evrişimli sinir ağları kullanarak Movielens-100K gerçekleştirdiği çalışmada MAE metriğinde 0.1179 değerini elde ederek tez çalışmasındaki MAE sonuçlarına göre daha başarılı sonuçlar elde etmiştir. Bu noktada Kısıtlı Boltzmann Makineleri algoritmasının farklı algoritmalarla hibrit bir yapı oluşturularak modellenmesi öneri sisteminin daha iyi performans elde etmesini sağlayabilir.

Çalışma boyunca ve yapılan literatür araştırmalarına da göre Üretici Çekişmeli Ağların diğer algoritmalara göre daha başarılı sonuçlar göstermesi beklenmekteydi. Fakat yapılan analiz sonuçlarında Üretici ve Çekişmeli Ağlar'da ayrık veriler ve diğer algoritmalara göre daha yüksek hata oranları gözlemlenmiştir. Çalışmada Temel Üretici Çekişmeli Ağ kullanılarak ve varsayılan parametrelerle eğitimin gerçekleşmiştir. Üretici Çekişmeli Ağlar'ın diğer varyasyonları ile yapılacak çalışmalar bu algoritmanın sonuçlarının iyileştirilmesini sağlayabilir.

Bu tez çalışması kapsamında kullanılan tüm derin öğrenme algoritmaları varsayılan parametreleri kullanılarak eğitim süreci gerçekleştirilmiştir. Algoritmalara uygulanacak hiper parametre ayarı ile daha iyi performanslar elde edilebilir.

Gerçekleştirilen tez çalışmasında hibrit bir sistem kullanılmamıştır. Birden fazla algoritma birleştirilerek oluşturulacak derin hibrit modeller ile algoritmaların başarıları arttırılabilir.

KAYNAKLAR

[1] Zhang, S., Yao, L., Sun, A., & Tay, Y. (2019). Deep learning based recommender system: A survey and new perspectives. ACM Computing Surveys (CSUR), 52(1), 1-38.

[2] Dong, M., Yuan, F., Yao, L., Wang, X., Xu, X., & Zhu, L. (2022). A survey for trust-aware recommender systems: A deep learning perspective. Knowledge-Based Systems, 249, 108954.

[3] Da'u, A., & Salim, N. (2020). Recommendation system based on deep learning methods: a systematic review and new directions. Artificial Intelligence Review, 53(4), 2709-2748.

[4] Anwar, T., Uma, V., & Srivastava, G. (2021). Rec-cfsvd++: Implementing recommendation system using collaborative filtering and singular value decomposition (svd)++. International Journal of Information Technology & Decision Making, 20(04), 1075-1093.

[5] Joorabloo, N., Jalili, M., & Ren, Y. (2020). Improved collaborative filtering recommendation through similarity prediction. IEEE Access, 8, 202122-202132.

[6] Karpus, A., Raczynska, M., & Przybylek, A. (2019). Things You Might Not Know about the k-Nearest Neighbors Algorithm. In KDIR (pp. 539-547).

[7] Ferreira, D., Silva, S., Abelha, A., & Machado, J. (2020). Recommendation system using autoencoders. Applied Sciences, 10(16), 5510.

[8] Yi, B., Shen, X., Liu, H., Zhang, Z., Zhang, W., Liu, S., & Xiong, N. (2019). Deep matrix factorization with implicit feedback embedding for recommendation system. IEEE Transactions on Industrial Informatics, 15(8), 4591-4601.

[9] He, M., Meng, Q., & Zhang, S. (2019). Collaborative additional variational autoencoder for top-N recommender systems. IEEE Access, 7, 5707-5713.

[10] Zhu, Y., Wu, X., Qiang, J., Yuan, Y., & Li, Y. (2021). Representation learning with collaborative autoencoder for personalized recommendation. Expert Systems with Applications, 186, 115825.

[11] Liu, X., Ouyang, Y., Rong, W., & Xiong, Z. (2015, November). Item category aware conditional restricted boltzmann machine based recommendation. In International conference on neural information processing (pp. 609-616). Springer, Cham.

[12] Wang, J., Yu, L., Zhang, W., Gong, Y., Xu, Y., Wang, B., ... & Zhang, D. (2017, August). Irgan: A minimax game for unifying generative and discriminative information retrieval models. In Proceedings of the 40th International ACM SIGIR conference on Research and Development in Information Retrieval (pp. 515-524).

[13] Sanderson, M., & Croft, W. B. (2012)**.** The history of information retrieval research. Proceedings of the IEEE, 100(Special Centennial Issue), 1444-1451.

[14] Salton, G., Wong, A., & Yang, C. S. (1975). A vector space model for automatic indexing. Communications of the ACM, 18(11), 613-620.

[15] Resnick, P. (1994). Anopen architecture for collaborative filterring of netnews. In Proc CSCW'94.

[16] Rich, E. (1979). User modeling via stereotypes. Cognitive science, 3(4), 329-354.

[17] Goldberg, D., Nichols, D., Oki, B. M., & Terry, D. (1992). Using collaborative filtering to weave an information tapestry. Communications of the ACM, 35(12), 61-70.

[18] Balabanović, M., & Shoham, Y. (1997). Fab: content-based, collaborative recommendation. Communications of the ACM, 40(3), 66-72.

[19] Jun, S., & Hwang, E. (2013). SmartRadio: Cloning Internet Radio Broadcasting Stations. International Information Institute (Tokyo). Information, 16(4), 2701.

[20] Bennett, J., & Lanning, S. (2007, August). The netflix prize. In Proceedings of KDD cup and workshop (Vol. 2007, p. 35).

[21] Mu, R. (2018). A survey of recommender systems based on deep learning. Ieee Access, 6, 69009-69022.

[22] Wu, C., Wu, F., Huang, Y., & Xie, X. (2023). Personalized news recommendation: Methods and Challenges. ACM Transactions on Information Systems, 41(1), 1-50.

[23] Sedhain, S., Menon, A. K., Sanner, S., & Xie, L. (2015, May). Autorec: Autoencoders meet collaborative filtering. In Proceedings of the 24th international conference on World Wide Web (pp. 111-112).

[24] Noshad, Z., Bouyer, A., & Noshad, M. (2021). Mutual information-based recommender system using autoencoder. Applied Soft Computing, 109, 107547.

[25] Jiang, J., Li, W., Dong, A., Gou, Q., & Luo, X. (2020). A Fast Deep AutoEncoder for high-dimensional and sparse matrices in recommender systems. Neurocomputing, 412, 381-391.

[26] Chen, S., & Wu, M. (2020). Attention collaborative autoencoder for explicit recommender systems. Electronics, 9(10), 1716.

[27] Khan, Z. A., Zubair, S., Imran, K., Ahmad, R., Butt, S. A., & Chaudhary, N. I. (2019). A new users rating-trend based collaborative denoising auto-encoder for top-N recommender systems. IEEE Access, 7, 141287-141310.

[28] Wu, Y., DuBois, C., Zheng, A. X., & Ester, M. (2016, February). Collaborative denoising auto-encoders for top-n recommender systems. In Proceedings of the ninth ACM international conference on web search and data mining (pp. 153-162).

[29] Tian, Z., & Liu, H. (2020, December). Collaborative filtering recommendation algorithm based on improved denoising auto encoder. In 2020 2nd International Conference on Information Technology and Computer Application (ITCA) (pp. 23-28). IEEE.

[30] Zhang, Y., Zhao, C., Chen, M., & Yuan, M. (2021). Integrating stacked sparse auto-encoder into matrix factorization for rating prediction. IEEE Access, 9, 17641-17648.

[31] Liang, D., Krishnan, R. G., Hoffman, M. D., & Jebara, T. (2018, April). Variational autoencoders for collaborative filtering. In Proceedings of the 2018 world wide web conference (pp. 689-698).

[32] Sachdeva, N., Manco, G., Ritacco, E., & Pudi, V. (2019, January). Sequential variational autoencoders for collaborative filtering. In Proceedings of the twelfth ACM international conference on web search and data mining (pp. 600-608).

[33] Pang, B., Yang, M., & Wang, C. (2019, April). A novel top-n recommendation approach based on conditional variational auto-encoder. In Pacific-Asia conference on knowledge discovery and data mining (pp. 357-368). Springer, Cham.

[34] Askari, B., Szlichta, J., & Salehi-Abari, A. (2021, July). Variational autoencoders for top-k recommendation with implicit feedback. In Proceedings of the 44th International ACM SIGIR Conference on Research and Development in Information Retrieval (pp. 2061-2065).

[35] Chakrabarty, N., Rana, S., Chowdhury, S., & Maitra, R. (2019, December). RBM based joke recommendation system and joke reader segmentation. In International Conference on Pattern Recognition and Machine Intelligence (pp. 229-239). Springer, Cham.

[36] Sahoo, A. K., & Pradhan, C. (2020). Accuracy-Assured Privacy-Preserving Recommender System Using Hybrid-Based Deep Learning Method. Recommender System with Machine Learning and Artificial Intelligence: Practical Tools and Applications in Medical, Agricultural and Other Industries, 101-120.

[37] Kuo, R. J., & Chen, J. T. (2020). An application of differential evolution algorithm-based restricted Boltzmann machine to recommendation systems. Journal of Internet Technology, 21(3), 701-712.

[38] Behera, D. K., Das, M., & Swetanisha, S. (2019). Predicting users' preferences for movie recommender system using restricted Boltzmann machine. In Computational intelligence in data mining (pp. 759-769). Springer, Singapore.

[39] Liu, J., Pan, W., & Ming, Z. (2020). CoFiGAN: Collaborative filtering by generative and discriminative training for one-class recommendation. Knowledge-Based Systems, 191, 105255.

[40] Tong, Y., Luo, Y., Zhang, Z., Sadiq, S., & Cui, P. (2019, April). Collaborative generative adversarial network for recommendation systems. In 2019 IEEE 35th International Conference on Data Engineering Workshops (ICDEW) (pp. 161-168). IEEE.

[41] Wang, H., Shao, N., & Lian, D. (2019, July). Adversarial binary collaborative filtering for implicit feedback. In Proceedings of the AAAI Conference on Artificial Intelligence (Vol. 33, No. 01, pp. 5248-5255).

[42] Fan, J., & Chow, T. (2017). Deep learning based matrix completion. Neurocomputing, 266, 540-549.

[43] Mehrdad, S., & Kahaei, M. H. (2021). Deep learning approach for matrix completion using manifold learning. Signal Processing, 188, 108231.

[44] Khoali, M., Tali, A., & Laaziz, Y. (2020, March). Advanced recommendation systems through deep learning. In Proceedings of the 3rd International Conference on Networking, Information Systems & Security (pp. 1-8).

[45] Dong, M., Yuan, F., Yao, L., Wang, X., Xu, X., & Zhu, L. (2022). A survey for trust-aware recommender systems: A deep learning perspective. Knowledge-Based Systems, 249, 108954.

[46] Strub, F., Gaudel, R., & Mary, J. (2016, September). Hybrid recommender system based on autoencoders. In Proceedings of the 1st workshop on deep learning for recommender systems (pp. 11-16).

[47] Goodfellow, I., Pouget-Abadie, J., Mirza, M., Xu, B., Warde-Farley, D., Ozair, S., ... & Bengio, Y. (2020). Generative adversarial networks. Communications of the ACM, 63(11), 139-144.

[48] He, X., He, Z., Du, X., & Chua, T. S. (2018, June). Adversarial personalized ranking for recommendation. In The 41st International ACM SIGIR Conference on Research & Development in Information Retrieval (pp. 355-364).

[49] Batmaz, Z., Yurekli, A., Bilge, A., & Kaleli, C. (2019). A review on deep learning for recommender systems: challenges and remedies. Artificial Intelligence Review, 52(1), 1-37.

ÖZGEÇMİŞ

Ad-Soyad **:** Ecem Bölük
Doğum Tarihi ve Yeri **:**
E-posta **:**

ÖĞRENİM DURUMU:

- **Lisans** **:** 2018, Sakarya Üniversitesi, Bilgisayar ve Bilişim Bilimleri Fakültesi, Bilgisayar Mühendisliği

MESLEKİ DENEYİM VE ÖDÜLLER:

- Ithinka Bilgi Teknolojileri (Veri Analisti), 23.08.2021 – halen çalışıyor.

TEZDEN TÜRETİLEN ESERLER, SUNUMLAR VE PATENTLER:

- Öneri Sistemlerinde Veri Seyrekliği Problemine Otomatik Kodlayıcı Yaklaşımlarının Karşılaştırmalı Bir Çalışması, IDAP-2023