GENETİK ALGORİTMA TABANLI GİZLİLİĞİ KORUYAN ORTAK FİLTRELEME

Yüksek Lisans Tezi

Mustafa Kemal Birgin

Eskişehir, 2019

GENETİK ALGORİTMA TABANLI GİZLİLİĞİ KORUYAN ORTAK FİLTRELEME

Mustafa Kemal BİRGİN

YÜKSEK LİSANS TEZİ

BİLGİSAYAR MÜHENDİSLİĞİ ANABİLİM DALI

Danışman: Dr. Öğr. Üyesi Alper BİLGE

Eskişehir

Eskişehir Teknik Üniversitesi

Fen Bilimleri Enstitüsü

Mayıs, 2019

JÜRİ VE ENSTİTÜ ONAYI

Mustafa Kemal Birgin'in **"Genetik Algoritma Tabanlı Gizliliği Koruyan Ortak Filtreleme"** başlıklı tezi 29/05/2019 tarihinde aşağıdaki jüri tarafından değerlendirilerek "Eskişehir Teknik Üniversitesi Lisansüstü Eğitim-Öğretim ve Sınav Yönetmeliği'nin ilgili maddeleri uyarınca, Bilgisayar Mühendisliği Anabilim dalında Yüksek Lisans tezi olarak kabul edilmiştir.

Jüri Üyeleri	**Unvanı Adı Soyadı**	**İmza**
Üye (Tez Danışmanı)	: Dr. Öğr. Üyesi Alper BİLGE	.....
Üye	: Doç. Dr. Cihan KALELİ	.....
Üye	: Doç. Dr. Alper Tolga KUMTEPE	.....

Enstitüsü Müdürü

ÖZET

GENETİK ALGORİTMA TABANLI GİZLİLİĞİ
KORUYAN ORTAK FİLTRELEME

Mustafa Kemal BİRGİN

Bilgisayar Mühendisliği Anabilim Dalı

Eskişehir Teknik Üniversitesi, Lisansüstü Eğitim Enstitüsü, Mayıs, 2019

Danışman: Dr. Öğr. Üyesi Alper BİLGE

İnternet kullanımının artması ile insanlar hemen her ihtiyacını çevirim içi hizmet sunan firmalardan karşılamaya başladılar. Ancak internet üzerinden hizmet veren firmaların kullanıcı ve ürün sayısı ile ürün çeşitliliği arttıkça kullanıcıların ürün seçiminde karar verebilmeleri giderek güçleşmiştir. Öneri sistemleri, bu süreçleri otomatikleştirerek kullanıcıların karar verme aşamasında kendilerine en uygun ürünün sunmaktadır. Öneri sistemlerinin doğru öneriyi kabul edilebilir bir süre içerisinde güvenli bir şekilde öneri üretebilmeleri gerekmektedir. Yapılan tez çalışmasında bu kriterleri hepsinin karşılanması amaçlanmıştır. Geleneksel hafıza tabanlı en yakın komşuluk esasına dayalı ortak filtreleme tekniği genetik algoritma ile geliştirilerek kullanıcılara daha doğru önerilerin üretilmesi amaçlanmıştır. Diğer taraftan kullanıcıların ürünlere verdikleri oylama bilgilerinin de gizliliği sağlanarak sistemin veri güvenliği de sağlanmıştır. Yapılan çalışmalarda ortak filtreleme ve gizliliği korunmuş ortak filtreleme yöntemleri ayrı ayrı genetik algoritma ile geliştirilmiş ve sonuçlar analiz edilmiştir. Her iki yöntemde de sistemin geleneksel tekniklere göre daha doğru sonuçlar ürettiği deneylerle gözlemlenmiştir. Gizliliği koruyan ortak filtreleme yönteminin genel doğruluğunun genetik algoritmalar ile iyileştirildikten sonraki artışının gizliliği sağlanmamış ortak filtreleme yönteminde elde edilen iyileştirmeden oransal olarak daha iyi olduğu gözlemlenmiş ve bu durumun sebepleri açıklanmıştır.

Anahtar Kelimeler: Öneri Sistemleri, Genetik Algoritma, Ortak Filtreleme, Gizliliği Koruyan Ortak Filtreleme

ABSTRACT

GENETIC ALGORITHM BASED PRIVACY PRESERVING

COLLABORATIVE FILTERING

Mustafa Kemal BİRGİN

Department of Computer Engineering

Eskişehir Technical University, Graduate School of Eskişehir Technical University, May, 2019

Supervisor: Assist. Prof. Dr. Alper BİLGE

With the increase of internet usage, people started to meet almost every need from online service companies. However, as the number of users and products and the variety of products offered by the companies providing services over the internet increased, it became increasingly difficult for users to decide on product selection. Recommendation systems, by automating these processes, offers users the most suitable product at the decision-making stage. Recommendation systems should be able to produce correct recommendations safely within an acceptable period of time. In this thesis, it is aimed to meet all of these criteria. It is intended to produce more accurate suggestions to the users by improving the closest neighborhood based collaborative filtering technique which is a memory based traditional technique. On the other hand, the data security of the system has been ensured by preserving the privacy of the rating information given by the users to the products. Collaborative filtering and privacy-preserved collaborative filtering methods were developed with separate genetic algorithm and the results were analyzed. In both methods, it was observed that the system produced more accurate results than traditional techniques. It is observed that the increase in the overall accuracy of the privacy-preserved collaborative filtering method after it has been improved by genetic algorithms is proportionally better than the improvement obtained in the collaborative filtering method that is not private and the reasons for this manner are explained.

Keywords: Recommendation Systems, Genetic Algorithm, Collaborative Filtering, Privacy-Preserving Collaborative Filtering

TEŞEKKÜR

Bu çalışmanın gerçekleştirilmesinde her türlü desteğini aldığım sayın Dr. Öğr. Üyesi Alper BİLGE hocama katkılarından dolayı teşekkür ederim. Ayrıca çalışmalarım sırasında bana her türlü desteği verip moral sağladığı için sevgili eşim Pelin ve kızım Duru'ya teşekkür ederim.

Mustafa Kemal BİRGİN

Mayıs, 2019

ETİK İLKE VE KURALLARA UYGUNLUK BEYANNAMESİ

Bu tezin bana ait, özgün bir çalışma olduğunu; çalışmamın hazırlık, veri toplama, analiz ve bilgilerin sunumu olmak üzere tüm aşamalardan bilimsel etik ilke ve kurallara uygun davrandığımı; bu çalışma kapsamında elde edilemeyen tüm veri ve bilgiler için kaynak gösterdiğimi ve bu kaynaklara kaynakçada yer verdiğimi; bu çalışmanın Anadolu Üniversitesi tarafından kullanılan "bilimsel intihal tespit programıyla tarandığını ve hiçbir şekilde "intihal içermediğini" beyan ederim. Herhangi bir zamanda, çalışmamla ilgili yaptığım bu beyana aykırı bir durumun saptanması durumunda, ortaya çıkacak tüm ahlaki ve hukuki sonuçlara razı olduğumu bildiririm.

Mustafa Kemal BİRGİN

29/05/2019

İÇİNDEKİLER

Sayfa

TABLOLAR/ÇİZELGELER DİZİNİ

Sayfa

ŞEKİLLER DİZİNİ

Sayfa

SİMGELER VE KISALTMALAR DİZİNİ

AK **:** Aktif Kullanıcı

BP **:** Başlangıç Popülasyonu

GA **:** Genetik Algoritma

GKOF: Gizliliği Koruyan Ortak Filtreleme

kNN : k-En Yakın Komşu

OF **:** Ortak Filtreleme

OMH **:** Ortalama Mutlak Hata

ÖS : Öneri Sistemleri

RBT **:** Rastgele Bozma Tekniği

UD **:** Uygunluk Değeri

UF **:** Uygunluk Fonksiyonu

1. GİRİŞ

Bilişim teknolojilerinin gelişimi, insanlık tarihinde çok önemli bir role sahiptir. Bilginin sayısal veri olarak bilgisayar disklerinde kompakt bir şekilde tutulabilmesi ile milyonlarca kitaptan oluşan bir kütüphane tek bir disk içerisine sığacak hale gelmiştir. İnternet ve internete bağlı teknolojilerin gelişimi ile ise sayısallaşan bu bilgilerin erişimi daha önce hiç olmadığı kadar kolaylaştı. Bilişim şirketlerinin devasa boyutlara ulaşması, bilgisayar donanımlarında ve bilgisayar ağlarındaki devrimsel gelişmeler ile insanlar dünyanın herhangi bir yerinde üretilen bir bilgiye anında erişebilir oldu. Son yıllarda mobil cihazların yaygınlaşması ile de bilgi sadece anında erişilir değil, aynı zamanda her yerden erişilebilir oldu. Bunun bir sonucu olarak insanlar artık birçok ihtiyacını internet üzerinden hizmet veren şirketler üzerinden çözebiliyorlar.

İnternet üzerinden artık alışveriş, film kiralama, tatil rezervasyonu gibi işlemler yapılabiliyor, çevirim içi kurs alınabiliyor hatta üniversite okuyup sınavlara yine çevirim içi sistemler kullanılarak girilebiliyor. Kullanıcılar istedikleri ürün veya hizmeti satın alabilmek için çok büyük bir ürün ve hizmet yelpazesinden kendi ihtiyaçlarını karşılayacak olanları bulması için ciddi çaba ve zaman gerektiren bir iştir. Bu ihtiyaçların giderilmesi esnasında kişiye özel mantıklı öneriler sunulması ihtiyacı doğmuştur. Ancak sisteme giren kullanıcıların sayısındaki artış beraberinde büyük verilerle nasıl baş edileceği problemini de getirmiştir. İnsanlar alışveriş yaparken kendilerine öneri üretmek adına daha önce yaptığı alışverişler ve kendilerine benzer zevkleri ve ihtiyaçları olan diğer kullanıcıların yaptığı işlemler temel alınarak geliştirilen öneri sistemlerinin (*ÖS*) kullanımı kaçınılmaz bir hale gelmiştir [3, 4]. *ÖS*, kullanıcıların kendi ihtiyaçlarını karşılayabilecekleri ürün ve hizmetleri onlara en doğru ve en hızlı bir şekilde sunulması için tasarlanmış sistemlerdir [5].

En başarılı bilgi filtreleme sistemlerinden biri olan *ÖS,* büyük veriler ile baş edebilmek için kullanılan etkili bir yöntemdir. *ÖS*'nin amacı, kullanıcılara film, kitap, haber, müzik, kompakt disk (CD), sayısal çok yönlü disk (DVD), web sayfası gibi öğeleri, kullanıcıların tarihsel tercihleri ve çevirim içi arama sürelerine göre faydalı verileri otomatik olarak çıkarmaktır [1].

ÖS temelde üç farklı şekilde tasarlanabilir: Ortak filtreleme (*OF*), içerik temelli filtreleme ve karma sistemler. *OF* geçmişte benzer seçimler yapmış kullanıcıların

gelecekte de benzer seçimler yapacağını ön görerek geliştirilmiş bir yöntemdir. Bu şekilde bakıldığında belirli bir sayıda ürün tercihinde bulunmuş olan bir aktif kullanıcı (*AK*) yeni bir ürün isteği yaptığında daha önce kendisi ile benzer seçimler yapmış olan kullanıcılar tespit edilir ve bu kullanıcıların tercihleri *AK*'ya öneri olarak sunulur. İçerik temelli filtreleme tekniğinde yine daha önce belirli sayıda ürün tercihinde bulunmuş olan *AK*'nın tercihte bulunduğu ürünlere benzer ürünler kendisine önerilir. Karma sistemlerde ise her iki yöntem birlikte kullanılır [6]. Buradaki motivasyon *AK*'nın tercih belirtirken bir ürünü satın alması veya kendisine sunulan ürünler içerisinden oylama yapmak suretiyle beğenisini göstermesi şeklinde olabilir.

OF sistemleri kullanıcılara öneri üretebilmek için en sık kullanılan yöntemlerden birisidir. Bu teknik ilk defa Goldberg ve ark. Tarafından 1992 yılında ortaya atılmıştır [15]. *OF* sistemlerinde, kullanıcıya iki farklı şekilde öneri üretilir: İlk yaklaşımda *AK*'ya sistemin kendi veri tabanındaki ürünlerden kullanıcının beğenmesi en muhtemel N tane ürün sunulur. Bu işlem yapılırken *AK*'nın daha önce yaptığı ürün puanlamalarına bakılır, kendisi ile benzer puanlamalar yapmış diğer kullanıcıların geçmişte beğendiği ürünler içerisinden *AK*'nın beğenisine en yakın N tane ürün gösterilir. İkinci yaklaşımda ise *AK* belirli bir ürün için sistemden öneri ister. Bu yöntemde yine *AK* ile benzer puanlamalar yapmış diğer kullanıcılar tespit edilir, daha sonra öneri istenen ürünü daha önce puanlamış olan benzer kullanıcıların bu ürüne verdiği puanlara göre *AK*' ya ürün önerilir veya önerilmez. Burada son aşamada ürün ile ilgili sayısal puanlama bilgisi (5 yıldızlı oylama sistemi gibi) veya ikili sistemde (beğen/beğenme) şeklinde öneri üretilebilir. *OF* sistemleri hafıza tabanlı, model tabanlı veya melez yapıda tasarlanabilir [15].

Dünyanın dört bir yanından birçok kullanıcı, bir takım ihtiyaçlarını karşılamak için bilgisayar ve mobil cihazlar kullanarak internet ortamına dâhil olmaktadır. Sisteme dâhil olan kullanıcılara kişiye özel önerilerin sunulması için kendisinden bir takım kişisel bilgiler talep edilmektedir. Kullanıcıların birçoğu sistemleri tasarlandıkları amaçlar için kullanır; ancak bazı kötü niyetli kullanıcılar bir takım korsanlık teknikleri ile sistemlerin veri tabanlarına erişip kullanıcı bilgilerini çalabilmektedir. Diğer taraftan kullanıcıların satın alma bilgileri veya ürünlere verdikleri puanlar, işlem yaptıkları sistemde kendilerine daha iyi öneri üretebilmek için kullanıldığı gibi, bazı ticari işletmelerce istismar edilebilir. Kullanıcı bilgileri, kullanıcı bazlı fiyatlandırma, kullanıcı profillerinin çıkarılması ve pazarlanması, kullanıcıların hareketlerinin takip edilmesi gibi bir takım kötü amaçlar için

kullanılabilir [7, 8]. Bu risklerin sonucu olarak kullanıcılar ya bu sistemi kullanmaktan kaçınacak, gerçek profil bilgilerini gizleyerek sahte hesaplar kullanacak veya tercihlerini manipüle edecektir. Ancak yüksek doğrulukta öneri üretilebilmesi için eldeki verinin doğru ve tutarlı olması gerekmektedir. Bu aşamada gizliliği koruyan ortak filtreleme (*GKOF*) yöntemleri devreye girmektedir [8, 9, 38].

Genetik algoritmalar (*GA*) temelde doğal seçilim ve genetik mekanizmalarının hâkim olduğu arama algoritmalarıdır. Bir problemin en iyi çözümünün bulunması için, sonsuz olasılığın olduğu çözüm evreninde, belirli olasılıklarla ancak rastgele seçilmiş olası çözümlerden en iyi olanları birbirleri ile çaprazlayarak optimum sonuca ulaşmayı amaçlar. Her ne kadar rastgele alt çözümleri harmanlayıp kullansa da bu rastgeleliğin arkasında bir mekanizma vardır. Değerlendirilen alt çözümlerden umut vadedenler, aramanın yönünü belirleyerek, ileriki aşamalarda olası en iyi çözüme götürür [14].

GA'ın en büyük dezavantajları yavaş çalışmalarıdır. Her ne kadar arama uzayında gözetimli bir öğrenme yolu izlense de, çok fazla olası çözümün olduğu çok parametreli büyük sistemler, fazla hesaplama gerektirdiği için işlem maliyetleri yüksektir. Ancak son yıllarda bilgisayar sistemlerinin donanımsal özellikleri yeterince güçlendiği için yapılan hesaplamalar, çok daha hızlı bir şekilde gerçekleştirilebilmektedir. Bu sayede *GA* birçok problemin çözümünde kullanılabilir olmuştur. Diğer taraftan model tabanlı *ÖS* öncelikle bir eğitim aşamasından geçip bir model oluşturarak ileriki aşamalarda bu modeli kullanıp öneri ürettiğinden hesaplama maliyetleri eğitim aşamasında çok büyük ölçüde giderilir ve son kullanıcıya hızlı bir şekilde sunulur. Yapılan çalışmada bu tip bir model tabanlı *ÖS* tasarlanmıştır.

2. AMAÇ VE KAPSAM

Yapılan çalışma kapsamında *OF* yöntemlerinin en önemli üç göstergesi olan doğruluk, performans ve gizliliğin iyileştirilmesi amaçlanmıştır. *OF* yöntemi k-En Yakın Komşu (*kNN*) algoritması ile gerçekleştirilmiştir. *kNN* algoritması, AK'ya en benzer k tane kullanıcının daha önce yaptığı ürün puanlamalarını kullanarak *AK*'nın henüz puan vermediği ürünlerden kendisine önerilerde bulunur. Buradaki motivasyon geçmişte benzer seçimler yapmış olan kullanıcıların gelecekte de benzer seçimler yapacağı varsayımıdır. *kNN* algoritması kullanıcılar arasındaki benzerlikleri hesaplarken farklı metrikler kullanabilir. Bunlardan bazıları Pearson korelasyon katsayısı, Vektör Cosine Benzerliği, Genişletilmiş Jaccard Katsayısı gibi metriklerdir. Yapılan çalışmada Pearson Korelasyon Katsayısı kullanılmıştır. *kNN* algoritması ile *GA* birlikte kullanılarak bir model oluşturulmuş ve *kNN* ile üretilen önerileri iyileştirmek amaçlanmıştır [16].

Hafıza tabanlı *OF* yöntemleri, *AK*'nın verilerini sistemdeki diğer kullanıcının verilerine bakarak anlık hesaplamalar yapar ve *AK'ya* öneri üretir. Bu yöntem canlı sistem üzerinde gerçek zamanlı çalıştığından anlık tüm değişimler hesaplamalara dâhil edilebilir; bu da *OF* yönteminin doğruluğunu artırır. İçeriği çok hızlı değişen dinamik sistemlerde hafıza tabanlı *OF* yöntemlerinin kullanılması bu kapsamda değerlendirildiğinde avantaj sağlar. Ancak hafıza tabanlı *OF* yöntemleri tüm hesaplamaları anlık yaptığından büyük sistemlerde performans sorunları baş gösterebilir. Diğer taraftan model tabanlı sistemlerde, önce bir eğitim aşaması ile bir model oluşturur ve bu model kullanılarak öneri üretilir. Bu yöntem içeriği çok sık değişmeyen sistemlerde kullanılabilir. Bu şekilde hem sistemin doğruluğu sağlanmış olur hem de oluşturulan model ile çok daha hızlı bir şekilde öneri üretilir. Yapılan çalışmada model tabanlı bir teknik kullanılmıştır.

Gizliliği sağlanmamış *ÖS*'nde kullanıcıların çok değerli olan satın alma veya ürün puanlama bilgileri açık bir şekilde veri tabanlarında tutulur. Bu durum birçok farklı güvenlik açığını da beraberinde getirir. Kullanıcı verisinin gizliliğinin sağlanmasının yollarından bir tanesi anonim tekniğidir. Bu teknikte kullanıcılar kişisel profil bilgilerini gizli tutarak oylama bilgilerini sisteme gönderir. Bu teknikte veri setinin doğruluğu tam olarak sağlanamaz. Kötü amaçlı kullanıcılar, veri tabanına rastgele birçok veri girişi yapabilir. Diğer taraftan bazı ticari işletmeler belirli ürünlerin beğenilirliğini artırmak için bu ürünlerin puanlamalarını artıracak bir takım girişler yapabilir. Bu tip sahte veriler

ÖS'nin veri tabanını diğer kullanıcılar için de kullanılmaz hale getirir. *ÖS*'nin veri tabanının doğru veriyi tutması kullanıcılara doğru öneri üretilesi için en önemli kuraldır [10, 11, 12]. Yapılan çalışmada kullanıcı verilerinin gizliliğini sağlamak diğer taraftan kullanıcılara üretilecek önerilerin de doğruluğunu en üst seviyede tutmak amaçlanmıştır.

Kullanıcı verilerinin gizliliğinin sağlanması için rastgele bozma tekniği (*RBT*) kullanılmıştır [38]. Bu teknikte, orijinal veriye bazı rastgele veriler eklenerek kullanıcının gerçek verisi saklanmaktadır. Bu şekilde her ne kadar gizlilik sağlanmış olsa da gerçek veri bir miktar bozulmaya uğrar. Ancak yeterli sayıda kullanıcının olduğu sistemlerde kullanıcı verilerine eklenen rastgele veri bu teknikle toplamda sıfır olduğu için sistemin çalışmasında genel doğruluk oranını çok küçük bir hata payıyla sağlamaktadır. Gizlilik ve doğruluk birbirleri ile ters orantılı birer kavramdır. Dolayısıyla bu teknikte de kullanıcı verisine eklenecek rastgele sayıların miktarı ve büyüklüğü arttıkça sistemin hata payını arttıracaktır [12, 13, 38].

Kullanıcı tabanlı *kNN* algoritması *ÖS*'de en sık tercih edilen algoritmalardan birisidir. Yapılan çalışmada *kNN* algoritması *GA* kullanılarak daha performanslı ve daha doğru sonuç üretecek şekilde geliştirilmiştir. Kullanıcılar arasındaki benzerlikler, *GA* operatörleri kullanılarak iyileştirilmiş ve üretilen önerilerdeki hata payının düşürülmesi sağlanmıştır. *OF* ve *GKOF* yöntemleri ayrı ayrı *GA* ile desteklenerek her iki metodun sonuçları *kNN* algoritması ile tasarlanmış geleneksel hafıza tabanlı *OF* ve *GKOF* yöntemleri ile karşılaştırılarak sonuçlar grafikler halinde gösterilmiştir.

Kullanılan yöntem, model tabanlı bir yöntem olduğu için performans açısından hafıza tabanlı yöntemlere göre çok daha hızlı sonuç verebilmektedir. Diğer taraftan model oluşturulurken *GA* operatörleri kullanılarak sistemin doğruluğu iyileştirilmiştir. Sonuç olarak gerek sistemin performans artışı gerek doğruluğunun sağlanması gerekse kullanıcı bilgilerinin gizliliğinin korunması sağlanmıştır.

3. İLGİLİ ÇALIŞMALAR

Dakhel ve Mahdavi [24], *K*-ortalama kümeleme algoritması ile kullanıcıları beğenilerine göre kümeledikten sonra *'oylama algoritması'* adını verdikleri bir teknikle öneri üretmişlerdir. Bu teknikte öneri isteyen *AK* hangi küme içerisinde ise o kümedeki kullanıcılardan öneri istenen ürüne verilen puanlardan en sık verilmiş olan puan öneri olarak *AK*'ya sunulmuştur. Choi ve ark. [26], *ÖS*'ndeki soğuk başlangıç problemi ve veri setlerindeki doluluk oranlarının düşük olmasından kaynaklanan problemlerinden dolayı kullanıcıların ürün puanlama verileri gibi bilgilerin doğruluğu yüksek öneriler üretemeyeceğini ön görmüşlerdir. Bu problemlerin çözümü için ise ürün kategorine göre öneriler üreten bir sistem tasarlamışlardır. Yadav ve ark. [27], *AK*'nın diğer kullanıcılarla olan benzerliklerini bulmak için sezgisel tabanlı Bat algoritmasını kullanmış ve yüksek doğrulukta öneriler üretmişlerdir. Yine bu çalışmada Bat algoritması ile meta sezgisel bir algoritma olan Sanal Arı Kolonisi algoritması karşılaştırılmış ve bu algoritma ile üretilen sonuçlardan daha iyi sonuçlar elde edilmiştir. Wang ve ark. [25] boşluklu yapıdaki veri setlerinde de doğruluğu yüksek seviyede tutabilen bir algoritma önermişlerdir. *'Benzerlik füzyonu'* adını verdikleri bu yöntem hem kullanıcı hem de ürün tabanlı bir yöntemdir.

Holland ve ark. 1960 yılında Michigan Üniversitesinde ilk defa genetik algoritmaları geliştirdiler. Çalışmaları iki aşamadan oluşmuştu: Doğal sistemlerin adaptasyon süreçlerini sanallaştırarak açıklamak ve doğal sistemlerin önemli mekanizmalarını benimseyerek soyut bir sistem geliştirmek [14]. Daha sonraki yıllarda *GA* birçok alandaki çok farklı problemin çözümünde kullanıldı. Bu alanlardan bir tanesi de *ÖS* oldu. Bu bağlamda yapılan çalışmalar genellikle farklı algoritmalar ile *GA* birlikte kullanılarak gerçekleştirilmiştir.

GA kullanılarak birçok *OF* uygulaması yapılmıştır. Bu çalışmalarda genellikle farklı algoritmalar ve teknikler ile *GA* birlikte kullanılarak farklı yaklaşımlar geliştirilmek amaçlanmıştır. *GA* kullanılarak yapılmış olan *GKOF* çalışması literatürde bu kapsamda yapılmış olan ilk çalışmadır. Kim ve Ahn [17], K-ortalama ve SOM kümeleme algoritmalarının karşılaştırmalı analizini yaptıktan sonra K-ortalama kümeleme algoritmasını *GA* ile optimize eden bir çalışma ortaya koymuştur. Bu çalışmada *K*-ortalama kümele algoritmasının başlangıcında belirlenen küme merkezlerinin seçilmesinde herhangi bir mekanizmanın olmadığından, hâlbuki başlangıç küme merkezlerinin farklı seçilmesi halinde sonuçların çok farklı şekillerde çıkabileceğinden

bahsedilmiştir. Ayrıca başlangıç küme merkezlerinin rastgele seçilmesi algoritmanın yerel optimum sonuçlara takılıp kalacağı söylenmiştir. Bu problemlerin çözümü için *K*-ortalama algoritmasının başlangıç küme merkezlerinin seçiminde *GA* tabanlı bir yaklaşım önerilmiştir. Rana ve Jain [6] geleneksel kümeleme algoritmalarının veri setlerini gruplarken zamansal boyutların varlığını ihmal ettikleri için performanslarının büyük oranda azaldığını ön görerek bu problemin *GA* kullanarak bir çözüm üretmişlerdir. Dünyadaki hızlı değişimin zaman içerisinde kullanıcıların tercihlerini de değiştireceği düşüncesi öne çıkmaktadır. Bu kriterler göz önüne alınarak zamansal boyutun da dikkate alındığı bir dinamik *ÖS* geliştirilmiştir. Dao ve ark. [19] 2012 yılında yazdıkları makalelerinde *OF* tekniğinin *GA* gibi yapay zekâ yöntemleri ile birlikte kullanılarak daha iyi sonuçlar almayı amaçlamışlardır. Yeni bir *ÖS* olarak *GA* tabanlı İçerik-Farkındalıklı Ortak Filtreleme sitemi önermişlerdir. Yapılan çalışmada amaçlanan kullanıcı tercihlerine ve ürün içeriklerine göre konum tabanlı bir reklam öneri sistemi tasarlamaktır. İki ürünün içeriklerinin birbirleri ile olan benzerliklerini tespit etmek için *GA* kullanılmıştır. Bobadilla ve ark. [20] yaptıkları çalışmada kullanıcıların ortak oyladıkları ürünler üzerinden bir formülasyon ile benzerlikleri ağırlıklandırarak bu ağırlıklarının kendi belirledikleri bir aralık içerisinde anlamlı olanları *GA*'ya verilerek başarılı olanlar seçilmiştir. Ar ve Bostancı [16] kullanıcıların birbirleri ile olan benzerliklerini farklı metrikler kullanarak hesapladıktan sonra bu değerleri *GA* operatörlerine verip yeni benzerlik değerleri hesaplamış ve daha iyi sonuç veren benzerlik değerlerini ileriki iterasyonlarda kullanmıştır.

OF uygulamalarında gizlilik koruma ile ilgili ilk çalışmalar Canny [21, 22] tarafından 2002 yılında yapıldı. Bu çalışmalarda toplu verilerin bazı şifreleme teknikleri ile dağıtık ortamlarda tutulması önerilmiştir. Bilge ve Polat [23], Polat ve Du [12, 38], Kaleli ve Polat [42], Yargıç ve Bilge [43] makalelerinde, kullanıcıların puanlama verilerine rastgele sayılar eklenerek kullanıcı verilerinin gizliliğini korurken, sistemin hata payını en düşük seviyede tutan *RBT*'nin başarılı bir şekilde kullanılabileceğini göstermişlerdir.

4. ORTAK FİLTRELEME

İçerik tabanlı algoritmalar yapısı itibarı ile ürün ve kullanıcının birtakım özelliklerine bağımlıdır. Bir ürünün özellikleri her açıdan incelenmeli ve ürünün pazarlanabilir yönleri ortaya konulmalıdır. Aynı şekilde kullanıcılara ürün önerilebilmesi için kullanıcıların demografik bilgileri, zevk ve alışkanlıkları gibi kullanıcı profili oluşturulmasında hayati önem arz eden bilgiler doğru bir şekilde alınmalıdır [34]. Ürün ve kullanıcı profillerini çıkarıldıktan sonra ve hangi kullanıcının hangi ürüne ihtiyaç duyabileceği doğru bir mantıkla programlandıktan sonra kullanıcıya özel öneriler üretilebilir. Burada sistemin karşılaşabileceği bir takım problemler vardır. Öncelikle sistem kullanıcı hakkında profil bilgilerine ihtiyaç duyacaktır. Bu bilgiler yaş, cinsiyet, yaşanılan bölge, gelir durumu, medeni durum, engel durumu gibi demografik bilgilerin yanı sıra kullanıcıların tercihlerinin tespit edilebilmesi için ihtiyaç duyulan kullanıcı zevk, beğeni ve ihtiyaçları ve bu ihtiyaçların hobi amaçlı mı iş amaçlı mı olarak kullanılacağı bilgisi gibi birçok veriye ihtiyaç duyulur [35]. Diğer taraftan ürün profilinin çıkarılabilmesi için ürünle ilgili özelliklerin iyi analiz edilmesi gerekmektedir. Profili çıkarılacak fiziksel bir ürünse uzunluk, ağırlık gibi boyutsal özellikler ile içeriksel veya maddesel (katı, sıvı vb.) özelliklerinin çıkarılması gerekmektedir. Aynı şekilde sanal bir ürün analiz edilirken yazılımsal ürünler için farklı, resim, video gibi multimedya içerikler için farklı, çevrimiçi kurslar için çok farklı özellikler çıkarılmalı ve bunların uygulama geliştirme süreçlere dâhil edilmesi gerekmektedir. Bu verilerin doğru bir şekilde çıkarılması ve anlamlı bir şekilde programlanabilmesi oldukça zor bir süreçtir. Kullanıcıların birçoğu kendilerinden çok fazla bilgi isteyen bir sisteme gerek güvenlik amaçlı gerekse ayırdıkları zaman açısından sıcak bakmayacaktır. Ya yanlış ya da eksik bilgiler girerek sonraki aşamaya geçmek isteyecektir. Dolayısı ile sağlıklı çalışan bir öneri üretim sistemi tasarlamak çok zor olacaktır [36].

1992 yılında Golderg ve ark. *OF* alanında ilk çalışmaları bir e-posta filtreleme sistemi olarak sunmuştur [15]. Yıllar içerisinde *OF* birçok alanda kullanılarak popülerlik kazanmıştır. *OF*, her kullanıcıya özel öneriler üreten bir öneri sistemidir. Öneri üretirken, geçmişte benzer seçimler yapmış kullanıcıların gelecekte de benzer seçimler yapacağını ön gören bir mantıkla çalışır [8]. *OF* sistemi, öneri üretebilmek için bir kullanıcı-ürün matrisi kullanır. Örnek bir kullanıcı-ürün matrisi Tablo-4.1.'de gösterilmiştir. Her kullanıcı-ürün matrisi farklı boyutlarda olabilir. Buradaki matris, görüldüğü üzere

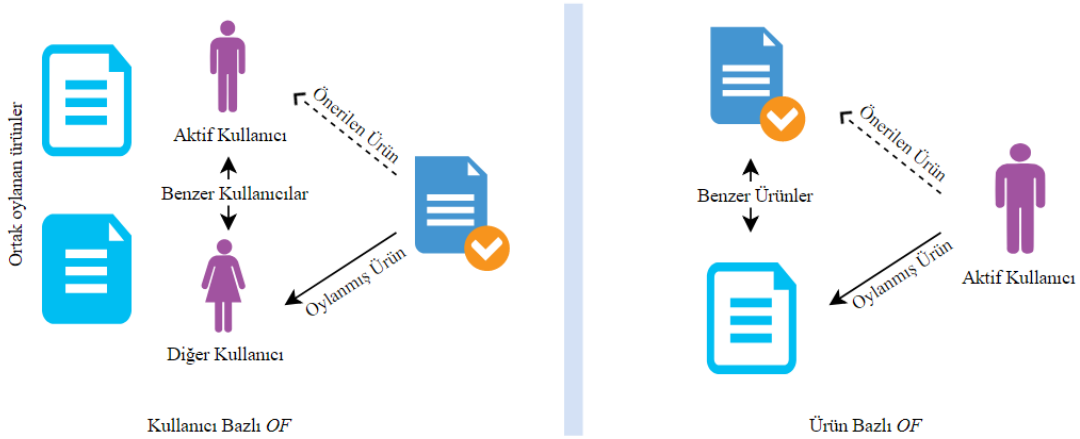
kullanıcılar *{kullanıcı$_1$, kullanıcı2, ... kullanıcı$_n$}*, ürünler ise *{ ürün$_1$, ürün$_2$, ... ürün$_m$}* olacak şekilde U_{nxm} kullanıcı-ürün matrisidir. Örnek matriste kullanıcıların ürünlere verdiği sayısal oylama bilgileri görülmektedir. Buradaki oylama sayısal aralıkları 1 ile 5 puan aralığında seçilmiştir. Bu aralıklar farklı veri setlerinde farklı şekillerde olabilir. Bu yöntemin yanı sıra kullanıcı tercihi, ürünü beğenme veya beğenmeme şeklinde ikili sistemde de tutulabilir. Kullanıcı-ürün matrisi doğrudan veya dolaylı bir şekilde doldurulabilir. Doğrudan doldurma yöntemi adından da anlaşılacağı üzere kullanıcıların ürünleri direkt olarak oyladıkları yöntemdir. Dolaylı doldurma yönteminde ise kullanıcının ürüne puan vermemekle birlikte, ürün sayfasında normalden fazla süre geçirmesi veya ürün sayfasını ziyaret sıklığı, ürüne olumlu yorum yapması gibi bilgiler ile birtakım çıkarsamalar yapılarak kullanıcının ürüne dair düşünceleri tespit edilir [28]. Öneri üretilmesi için kullanıcıdan belirli sayıda ürüne oy vermesi istenir. Bunun sebebi öneri üretilecek kullanıcıya benzer diğer kullanıcıları tespit edebilmektir. Aktif kullanıcı belirli bir sayıda oylama yaptıktan sonra, kendisi ile benzer kullanıcılar bulunur ve bu kullanıcıların daha önce yaptıkları oylamalar içerisinden aktif kullanıcının en beğeneceği ürün veya ürünler gösterilir. *OF* sistemler hafıza tabanlı, model tabanlı ve melez olmak üzere üç sınıfa ayrılır [29].

Tablo 4.1. *Kullanıcı-ürün matrisi*

	ürün$_1$	**ürün$_2$**	**ürün$_3$**	**...**	**ürün$_m$**
kullanıcı$_1$	4	1			3
kullanıcı$_2$	1		5		2
kullanıcı$_3$	5	2	1		5
.					
.					
.					
kullanıcı$_n$	2	5	4		2

OF algoritmaları mantıksal olarak kullanıcı bazlı veya ürün bazlı olarak tasarlanabilir. Her iki yaklaşım da kullanıcı-ürün matrisini kullanır, ancak kullanıcı bazlı yaklaşım *AK* ile diğer kullanıcıların benzerliklerini hesaplar ve bu kullanıcı temelli benzerlikleri kullanarak öneri üretirken, ürün bazlı yaklaşımda kullanıcının öneri istediği

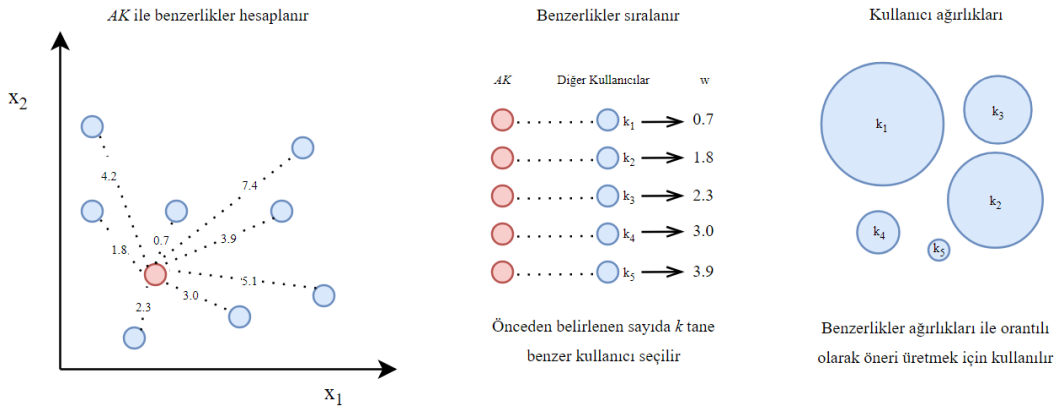
aktif ürüne benzer ürünler bulunarak bir öneri sunulur. Şekil 4.1'de bu anlatım görsel olarak özetlenmiştir.



Şekil 4.1. *Kullanıcı ve Ürün Bazlı Ortak Filtreleme*

4.1. Hafıza Tabanlı Ortak Filtreleme

Hafıza tabanlı *OF* sistemleri çevirim içi olarak gerçek zamanlı işlem yapan canlı sistemlerdir. Aktif kullanıcı *t* anında öneri istediği anda çalışan *OF* sistemi *t* anındaki kullanıcı-ürün matrisindeki bütün puanlamaları kullanarak aktif kullanıcı ile en benzer kullanıcıları bulur ve benzerlikleri hesaplar. Ardından bu benzerlikler ile *AK*'ya öneri üretilir. Yeni öneri istendiğinde bütün prosedür tekrar baştan alınır. Bir başka değişle kullanıcı-ürün matrisinin her istekte tekrar tekrar üzerinden geçilir. Buradaki en önemli nokta sistemin t anındaki kullanıcı-ürün matrisini kullanarak öneri üretmesidir. En güncel kullanıcı-ürün matrisi kullanılarak öneri üretilmektedir. İçeriği çok sık değişen bir kullanıcı-ürün matrisinde doğruluğu yüksek öneri üretmek için hafıza tabanlı *OF* yaklaşımının kullanılması bu açıdan kaçınılmaz olacaktır. Çünkü diğer yaklaşımlarda oluşturulan modeller genellikle *t-1* anında oluşturulacağı için arada geçen sürede benzerlikler t anındaki benzerliklere göre farklılıklar gösterecektir. Bu şekilde çalışan hafıza tabanlı *OF* sistemleri doğruluğu yüksek sistemlerdir, diğer taraftan performans açısından bazı zamanlarda beklentilerin altında kalmaktadırlar. Hafıza tabanlı algoritmalar ürün tabanlı veya kullanıcı tabanlı olarak her iki türlü tasarlanabilir [30-32].



Şekil 4.2. *kNN Algoritması*

Hafıza tabanlı *OF* sistemleri tasarlanırken genellikle Şekil 4.2'de görsel olarak gösterilen en yakın komşuluklar (*kNN*) dikkate alınarak kurgulanır. Aktif kullanıcı ile diğer kullanıcıların benzerlikleri çeşitli metrikler kullanılarak hesaplanır ve hesaplanan benzerlikler ile *AK*'ya en benzer olanlar seçilir. *AK* kendisi için q ürününe öneri istemiş olsun. Öncelikle *AK* ile benzer kullanıcılar bulunmaldır. Yapılan çalışmada birçok öneri sisteminde yaygın olarak kullanılmış Denklem 4.1'de verilen Pearson Korelasyon Katsayısı kullanılmıştır. Kullanıcı benzerliklerini ağırlıklandırmak için kullanılan bu metrik ilk kez GroupLens projesi kapsamında kullanılmıştır [32, 38]. Denklemde görülen ω_{au}, a kullanıcısı (*AK*) ile u kullanıcısı arasındaki benzerliği ifade etmektedir. m' değeri a ve u kullanıcısının ortak oyladıkları ürün sayısı iken, r_{ai} ve r_{ui} kullanıcıların ortak ürünlere verdikleri oylamalardır. Son olarak $\overline{r_a}$ ve $\overline{r_u}$ değerleri ise a ve u kullanıcıların oyladıkları ürünlere verdikleri oyların ortalama değerleridir [8, 33].

$$\omega_{au} = \frac{\sum_{i=1}^{m'}(r_{ai}-\overline{r_a})(r_{ui}-\overline{r_u})}{\sqrt{\sum_{i=1}^{m'}(r_{ai}-\overline{r_a})^2}\sqrt{\sum_{i=1}^{m'}(r_{ui}-\overline{r_u})^2}} \tag{4.1}$$

İki kullanıcı arasındaki benzerliğin (ω_{au}) hesaplanması için Denklem 4.1 kullanılmaktadır. Dolayısıyla bu formül kullanılarak *AK* ile kullanıcı-ürün matrisindeki diğer tüm kullanıcıların benzerlik değerleri hesaplandıktan sonra öneri üretme aşamasına geçilir. Öneri üretirken *AK* ile diğer en benzer k tane kullanıcı tespit edilir [8, 33]. Denklem 4.2 kullanılarak q ürünü için a kullanıcısına öneri üretilir.

$$p_{aq} = \overline{r_a} + \frac{\sum_{u=1}^{k}(r_{uq}-\overline{r_u}) \times \omega_{au}}{\sum_{u=1}^{k}\omega_{au}} \tag{4.2}$$

Üretilen p_{aq} değeri, a kullanıcısına q ürünü için önerilen ürün puanıdır. r_{uq}, benzer kullanıcıların q ürününe verdikleri puan, $\overline{r_u}$ değeri ise ürünlere verdikleri puanların ortalama değeridir.

4.2. Model Tabanlı Ortak Filtreleme

Çevirim içi çalışan hafıza tabanlı algoritmaların zayıf yönü anlık işlem yüklerinin çok fazla olmasıdır. Her öneri üretildiğinde ağır bir hesaplama yapıldığından performans sorunları baş gösterebilir. Diğer taraftan model tabanlı *OF* sistemleri, kullanıcı-ürün matrisinden bir model oluşturduktan sonra önerileri bu model üzerinden verebilir. Model üretilirken hesaplamalar çevirim dışı yapılır, öneri üretilirken bu model kullanılarak hesaplamalar çevirim içi yapılır [37]. Model tabanlı sistemler eğitim ve test olmak üzere iki aşamadan oluşurlar. Eğitim aşamasında veri setinin tamamı kullanılarak ortaya bir model koyulur ve test aşamasında bu model test edilir. İstenen doğruluk ve performans değerleri sağlanmışsa model öneri üretilmek üzere sistem tarafından kullanılır. Test aşamasında alınan sonuçlar beklenen değerlerin altında kalırsa, eğitim aşamasına geri dönülür ve sistem farklı parametrelerle tekrar eğitilir ve tekrar test edilir. Hala istenen performans sağlanamazsa model üretilirken kullanılan teknikte sistemin yapısına uygun iyileştirmeler yapılır ve test aşamasında alınan sonuçlar beklenen değerlere ulaşana kadar bu süreç tekrarlanır. Model, kullanıcı-ürün matrisinin bir özeti niteliğindedir. Matristeki oylama bilgilerinden, öneri üretmek için yeterli; ancak matrisin tamamını bir anlamda içinde barındıran bir tasarım inşa edilmektedir. Model tabanlı sistemlerin dezavantajı, model güncellenene kadar geçen sürede yapılan oylamaların model tarafından içerilmemesidir. Bu yapıda inşa edilen sistemlerde, yeni bir kullanıcı sisteme dâhil olduğunda, model güncelleninceye kadar öneri üretilemez. Aynı şekilde sistemde var olan kullanıcılar yeni oylamalar yaptığında, bu verileri henüz model içermediği için benzerlik hesaplamasında kullanmamış olacaktır. Dolayısıyla model tabanlı sistemler içeriği hızlı değişen kullanıcı-ürün matrislerinde kullanılırken bu kıstas dikkate alınmalıdır. Burada modelin güncellenme sıklığı önem kazanmaktadır. Her sistemin dinamikleri farklı olacağından model güncelleme sıklığı da geliştiricinin deney ve gözlemler yaparak karar vermesi gereken önemli bir parametredir.

Yapılan çalışmada oluşturulan model, kullanıcı bazında benzerlik değerlerini ihtiva eden bir yapıdadır. Her bir kullanıcı için *kNN* ile üretilen benzerlik değerleri eğitim aşamasında *GA* ile geliştirilmiş ve sonuçta elde edilen en iyi benzerlik değerleri benzersiz

kullanıcı numaraları ile eşleştirilmiştir. Ardından kullanıcı bazında üretilen bu benzerlik vektörleri test aşamasında kullanılmıştır. Elde edilen sonuçlar sistem parametreleri değiştirilerek tekrar tekrar hesaplanmış ve en iyi benzerlik değerleri elde edilmiştir. *OF* ve *GKOF* çalışmaları ayrı ayrı yapılmıştır. *GKOF* olarak yapılan çalışmada kullanıcı benzerlikleri hesaplanmadan önce kullanıcı-ürün matrisindeki veriler öncelikle *RBT* ile gizlenmiş [38] daha sonra benzerlik hesaplamaları yapılmıştır. *RBT* kullanılarak kullanıcı verilerinin gizlenmesi ile ilgili işlemler Bölüm 5'de anlatılmıştır.

4.3. Ortak Filtreleme Sistemlerinin Sorunları

Öneri sistemlerinin birincil amaçları olan doğruluk göstergesi, kullanıcılara ne ölçüde kaliteli öneriler sunulduğunu temsil eder [39]. Bir öneri sisteminin birincil amacı doğruluk değerini yüksek tutmak olmalıdır ki kullanıcılar sistemi kullanmaya devam etsinler. Ancak sistemin doğruluğunu olumsuz etkileyen bir takım faktörler vardır. Bunlardan ilki veri setlerinin büyük oranda boş olmasıdır. Kullanıcı-ürün matrislerinde genellikle kullanıcıların birçok ürünü oylamamış oldukları görülür. Öyle ki bazı durumlarda *AK* bir ürün için öneri istediğinde kendisine benzer kullanıcılardan bu ürüne puan veren kullanıcı bulunamadığından herhangi bir sonuç döndürülememektedir. Bunun sebebi kullanıcıların kendilerine sunulan yüzlerce veya binlerce üründen sadece özel olarak ilgilendikleri veya ihtiyaç duydukları ürünlere oy vermeleridir [31]. Öte yandan kullanıcılar arasındaki benzerliklerin hesaplanabilmesi ortak oylanmış ürünlere verilen puanlarla mümkün olabilmektedir. Veri setlerinin büyük boşluklu yapıda olması *OF* algoritmalarının doğruluğu yüksek öneriler üretmesinin en önemli sorunlarından birisidir. Yapılan çalışmada kullanılan veri setinin doluluk oranının çok düşük olmasına rağmen sistemin doğruluk oranında geleneksel *OF* yöntemleriyle elde edilen sonuçlara göre artış sağlanmıştır.

ÖS'nde karşılaşılan bir diğer problem kullanıcı verilerinin gizliliği sağlanmadan sunucularda tutulması problemidir. Bu durum kullanıcı verilerinin Bölüm 1'de anlatıldığı üzere kötü amaçlı kişi ve kurumlarca kullanılarak gerek kullanıcılara zarar verir gerekse öneri sisteminin çalışmasını olumsuz yönde etkileyebilir [38]. Yapılan çalışmada kullanıcı oylama verilerinin gizliği Bölüm 5.2'de anlatıldığı gibi *RBT* tekniği kullanılarak sağlanmıştır. Diğer taraftan gizliliği sağlanmış kullanıcı oylama bilgileri kullanılarak elde edilen sonuçlarda da geleneksel *GKOF* yöntemleriyle alınan sonuçlara göre doğruluk oranları daha iyi bir seviyede ölçülmüştür.

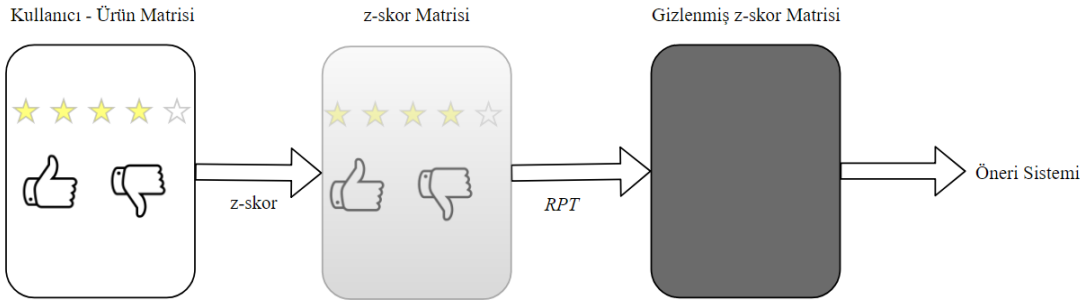
5. ORTAK FİLTRELEME SİSTEMLERİNDE GİZLİLİK KORUMA

OF algoritmalarının en önemli güvenlik problemi, gizliliği sağlanmamış kullanıcı verileri ile çalışmasıdır. Kendisi için bir ürün önerisi isteyen kullanıcının sistem tarafından öncelikle sınıflandırılması gerekecektir. Sınıflandırmanın doğru bir şekilde yapılabilmesinin yolu ise kullanıcının doğru analiz edilmesinden geçer. Bu aşamada kullanıcıya ait oylama bilgilerine ihtiyaç duyulur. Hiçbir oylama yapmamış kullanıcının sistemdeki hangi kullanıcılarla ortak zevklere sahip olduğunun anlaşılması mümkün değildir. Dolayısıyla öneri isteyen kullanıcı hakkında belirli sayıda ürüne oy vermesi istenecek ve ardından kendisine öneri üretilecektir. Ancak sunucularda açık bir şekilde tutulan kullanıcı oylama bilgileri Bölüm 1'de anlatılan saldırılarla kullanılmaz hale getirilebilir veya ticari işletmelerce istismar edilebilir. Bunun aşılması ancak kullanıcı verisinin gizliliğini sağlayarak mümkün olabilir [38]. Yapılan çalışmada kullanıcı verileri *RBT* tekniği kullanılarak gizlenmiş ve gizliliği sağlanmış veri seti üzerinden deneyler yapılmıştır. Ayrıca aynı çalışma, gizliliği sağlanmamış ortak filtreleme olarak da tasarlanıp sonuçlar karşılaştırılmıştır ve sağlanan gizliliğin doğruluk parametresini ne ölçüde etkilendiği ölçülmüştür.

Şekil 5.1'de görüldüğü gibi kullanıcı profilleri öncelikle z-skorları hesaplanarak normalize edilmiş ardından *RBT* ile gizlilik sağlanmıştır. Buradan anlaşılacağı üzere z-skor kullanıcı verisinde bir ön gizleme sağlamış olur. Ancak z-skor normalizasyon bir formülasyon ile elde edildiğinden işlemler tersine çevrilerek kullanıcı verileri ele geçirilebilir. Bunu engellemek için z-skorları hesaplanan verilerin rastgele üretilen sayılar ile geri dönüştürülemeyecek şekilde gizlenmesi mantığına dayanan *RBT* gibi bir yöntemle gizlenmesi şarttır. Burada izlenen adımlar aşağıdaki gibidir:

- ✓ Kullanıcı belirli sayıda ürüne oy verir
- ✓ Verdiği oyların z-skorlarını hesaplar
- ✓ z-skorları *RBT* ile gizler
- ✓ Gizlenmiş veriyi sunucuya gönderir

Kullanıcı sisteme gizlenmiş verilerini göndermektedir. Yani gizleme işlemini sunucu değil kullanıcı yapmaktadır. Dolayısıyla veri güvenliği ile ilgili kullanıcının aklında herhangi bir şüphe kalmayacaktır.



Şekil 5.1. *Veri Gizleme Adımları*

5.1. Kullanıcı Profillerinin Normalizasyonu

Herlocker ve ark. [40] yaptıkları çalışmada farklı normalizasyon teknikleri kullanarak, kullanıcı oylamalarını bir ön işleme sürecinden geçirerek yapılan önerilerin doğruluğunu iyileştirmeyi amaçlamıştır. z-skor ile normalize edilmiş oylamalar ile alınan hata oranları, normalize edilmemiş oylamalardan alınanlardan daha iyi çıktığı ölçülmüştür. Her kullanıcının z-skorlarının ortalama değeri 0, standart sapması ise 1'dir. a kullanıcısının q ürününe verdiği oyun z-skor puanı Denklem 5.1 gösterildiği gibi hesaplanır. Burada σ_a kullanıcı oylamalarının standart sapması $\overline{r_a}$ ise ortalama değeridir. [8, 38].

$$z_{aq} = \frac{(r_{aq} - \overline{r_a})}{\sigma_a} \tag{5.1}$$

z_{aq} değeri sadece q ürünü için hesaplanan z-skor değerdir. Kullanıcının tüm oylamaları için z-skorlar hesaplanmalıdır. Elde edilen z-skorlar kullanıcılar arasındaki benzerlik hesaplamalarında kullanılır. Denklem 5.2 z-skorları hesaplanan kullanıcıların benzerliklerinin hesaplanması için kullanılır.

$$\omega_{au} = \sum_{i=1}^{m'} z_{ai} \times z_{ui} \tag{5.2}$$

İki kullanıcı arasındaki benzerlik (ω_{au}) değeri hesaplanırken a ve u kullanıcılarının ortak oyladıkları ürünlerin sayısı olan m' tane z-skor değerinin çarpılıp, çarpımların toplanması yeterlidir. Burada yapılan işlem aslında iki tane vektör olan z_a ve z_u'nun skaler çarpımından ibarettir. Dolayısıyla benzerliklerin bulunması aşamasında gerek programlama maliyeti gerekse hesaplama maliyetleri önemli ölçüde azaltılmış olur. Öneri isteyen kullanıcının diğer tüm kullanıcılar ile olan benzerlikleri Denklem 5.2 ile hesaplandıktan sonra *kNN* algoritması kullanılarak en yakın k tane komşusu belirlenir.

Ardından öneri istenen ürünü daha önce oylamış olan kullanıcıların bu ürün için hesapladıkları z-skorları, *AK*'ya olan benzerlik değerleri ile orantılı olarak Denklem 5.3'de gösterilen formül kullanılarak öneri üretilir [38].

$$p_{aq} = \overline{r_a} + \sigma_a \frac{\sum_{i=1}^{k} \omega_{ai} \times z_{iq}}{\sum_{i=1}^{k} \omega_{ai}} \tag{5.3}$$

Burada hesaplanan p_{aq} değeri, a kullanıcısına q ürününün önerilip önerilmemesi kararını vermek için kullanılacaktır. ω_{ai} değeri, a kullanıcısının i kullanıcı ile olan benzerliğini gösterirken z_{iq} değeri i kullanıcısının q ürünü için hesaplanan z-skor değeridir. $\overline{r_a}$ ve σ_a değerleri ise sırası ile a kullanıcısının oylamalarının ortalama ve standart sapma değerleridir.

5.2. Rastgele Bozma Tekniği

RBT kullanılarak kullanıcı oylamalarının gizlenmesi kullanıcıların ürünlere verdiği oylara rastgele üretilen sayılar eklenerek gerçekleştirilir. Bu mantıkla yapılan gizleme işleminde kullanıcının ürünlere verdikleri oylamalarda verdikleri her bir oyun kendi içinde değerlendirmesi yapıldığında mantıklı bir işlem gibi görünmemektedir. Çünkü kullanıcıların verdikleri oylar ilk hallerinde çok farklı bir şekle bürünecektir. Ancak buradaki yaklaşım verilen oyların bireysel olarak değil veri setindeki oyların bütünüyle ele alınmasına dayanır. Yeterince büyük veri setlerine rastgele üretilen sayılar doğru bir mantıkla eklendiğinde, veri setinin gizliliği sağlanmış ve aynı zamanda veri setindeki toplam bilgi korunmuş olacaktır. Gizlenmiş veri setine bakıldığında orijinal bilgiden çok farklı görünecektir. Buradaki bilgi gizleme işlemi her oylama için bireysel olarak yapılmış olduğundan veri setinin yeni haline bakıldığında tanınmaz hale geldiği görülür. Ancak eklenen rastgele sayıların mantığı incelendiğinde sonuçta veri setinin toplam bilgisinin değişmediği ve buradan yüksek doğrulukta öneri üretilebildiği ölçülebilir [38].

RBT ile verilerin gizlenmesinde kullanıcıların z-skor değerlerine rastgele sayılar eklenmektedir. z-skorları hesaplanmış *A* ve *B* kullanıcılarının z-skor vektörleri sırası ile $A = (a_1, a_2, \dots, a_n)$ ve $B = (b_1, b_2, \dots, b_n)$ olsun. *A* ve *B* kullanıcılarının benzerliklerinin hesaplanması için z-skor vektörleri Denklem 5.2'de gösterildiği gibi çarpılmalıdır. Bu skaler çarpım yapılmadan önce verileri gizlemek için tekdüze veya normal dağılımlı $X=(x_1, x_2, \dots, x_n)$ ve $Y=(y_1, y_2, \dots, y_n)$ vektörleri eklenmiş ve bilgi güvenliği sağlanmış

olsun. Eşitlik 5.4, *A* ve *B* kullanıcılarının benzerliklerini veren z-skor vektörlerinin skaler çarpımlarının gizlenmeden önce ve gizlendikten sonraki değerlerinin yaklaşık olarak aynı olduğunu göstermektedir.

$$\sum_{i=1}^{m}(a_i + x_i)(b_i + y_i) = \sum_{i=1}^{m}(a_i b_i + a_i x_i + b_i y_i + x_i y_i) \approx \sum_{i=1}^{m} a_i b_i \quad (5.4)$$

X ve *Y* vektörleri tekdüze veya normal dağılımlı rastgele sayılar içerdiğinden $\sum_{i=1}^{m} a_i x_i$, $\sum_{i=1}^{m} b_i y_i$ ve $\sum_{i=1}^{m} x_i y_i$ değerlerinin yaklaşık değerleri uzun vadede 0 olacaktır. Diğer taraftan Eşitlik 5.5' de görüldüğü üzere *A* vektörüne eklenen normal dağılımlı sayılar yine uzun vadede *A* vektörünün toplam değerini değiştirmeyecektir [38].

$$\sum_{i=1}^{m}(a_i + x_i) = \sum_{i=1}^{m} a_i + \sum_{i=1}^{m} x_i \approx \sum_{i=1}^{m} a_i \quad (5.5)$$

Kullanıcı verilerinin gizlenmesi iki aşamadan oluşur. Birincisi kullanıcının oyladığı verilerin gizlenmesi, ikincisi ise kullanıcının hangi ürünleri oyladığı bilgisinin gizlenmesidir. Yani kullanıcının yaptığı oylamalar gizlenirken oylamadığı bazı ürünlere de sahte oylamalar eklenir. Bu şekilde hem kullanıcının beğendiği ve beğenmediği ürünler, hem de geniş ürün yelpazesinden kullanıcının hangi kategorilerdeki ürünlere ilgi duyduğu bilgisi gizlenmiş olur. Veri gizleme prosedürü ile ilgili adımlar aşağıda listelenmiştir [8].

- ✓ Kullanıcı z-skorlarını hesapla (Denklem 5.1)
- ✓ β_{max} ve σ_{max} parametrelerini belirle
- ✓ $\beta = (0, \beta_{max})$ ve $\sigma = (0, \sigma_{max})$ olacak şekilde β ve σ değerlerini üret
- ✓ $\alpha = \sigma\sqrt{3}$ formülü ile α değerini hesapla
- ✓ Kullanıcının oyladığı ve oylamadığı ürünleri belirle
- ✓ Kullanıcının oyladığı ürünlerin sayısı ile oylamadığı ürünlerin %β'sının toplam sayısını (*T*) hesapla
- ✓ Standart sapması σ ve ortalaması 0 olan normal dağılımda veya [-α, α] aralığında tekdüze dağılımda *T* adet rastgele sayı üret
- ✓ Kullanıcının z-skor vektöründe dolu olan değerlerin tamamına, boş olan değerlerin %β'sına rastgele indeksler üreterek ekle

6. GENETİK ALGORİTMA TABANLI ÖNERİLER ÜRETİLMESİ

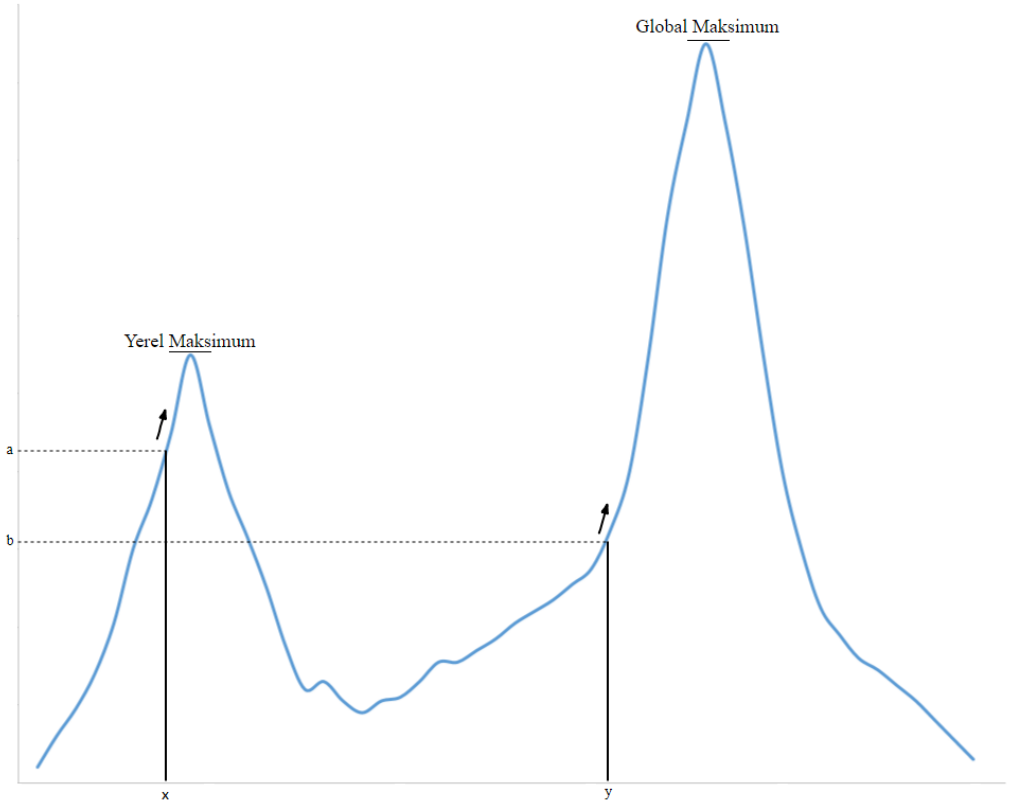
Genetik algoritmalar, biyolojik organizmaların doğal ortamlarında hayatta kalabilmek için geliştirdikleri adaptasyon mekanizmalarını kurgulayarak farklı bir yaklaşım önerirler. Her biyolojik organizma kendi içinde farklı özelliklere sahiptir ve her birinin yaşadıkları ortamların kendine has dinamikleri vardır. Organizmanın hayatta kalabilmesi iki şekilde mümkün olabilir. Ya yaşadığı ortama uyum sağlaması gerekir ya da hayatta kalabileceği farklı bir ortam bulması gerekir. Yaşadığı ortama uyum sağlayabilen organizmalar, karşı cinslerle çiftleşerek yeni nesiller üretirler. Ancak uyum sağlayamamış olanlar çiftleşecek duruma gelemeden yok olacağından yeni nesiller üretemezler. Bu şekilde doğal ortamda sadece uyum sağlamış olan organizmalar sayısal olarak çoğalırken sağlayamayanlar azalacaktır. Farklı organizmaların, yaşadığı ortama uyum sağlaması farklı şekillerde olabilir. Diğer taraftan aynı tür organizmaların da yaşadıkları ortama uyum sağlaması çok farklı şekillerde olabilir. Farklı adaptasyonlar geliştirerek yaşadıkları ortamda hayatta kalmış aynı tür ve karşı cinsten iki birey çiftleşince dünyaya gelen yeni birey genetik olarak her iki atanın da geliştirdiği adaptasyona sahip olacağından çok daha dirençli ve güçlü olacaktır. Bu birey de yine kendisi gibi güçlü bireylerle çiftleşerek daha da güçlü bireyler üretilecektir. Bu durum nesiller boyunca devam ettiğinde ortaya bir ağaç çıkacaktır. Bu ağacın yapraklarını temsil eden son bireyler geçmişte yaşamış tüm atalarının baskın gelen tüm genetik bilgilerine sahip olacaktır. Bazı durumlarda ise doğal bir ortamda bulunan bir canlı türünün bu ortama uyum sağlaması mümkün olamamaktadır. Bu durumda bu tür yaşadığı ortamı değiştirmek zorunda kalır. Eğer kendilerine uygun bir ortam bulamazlarsa türün tüm bireyleri yok olur ve bu tür ortadan kalkar.

Genetik algoritmalar, canlıların geçirdikleri bu biyolojik süreçleri modelleyerek bir yapı ortaya koyarlar. Tıpkı her canlının yaşayabilmesi için doğal bir ortam olması gerektiği gibi genetik algoritma da probleme yönelik modelleyeceği ortama uygun canlıları bu ortama getirmelidir. Model doğru kurgulanmazsa oluşturulan ortamda türetilen bireyler hayatta kalamayacağından sonuç alınamaz. Bu durumda ya yeni bireyler türetilecek ya da ortam değiştirilecektir. Bu durumda modelin arkasındaki mantığın güncellenmesi gerekecektir. Dolayısıyla her genetik algoritma her durumda çalışamaz. Tasarlanan algoritmaların genetik algoritma mantığına sahip olması ancak farklı dinamikler göz önüne alınarak tasarlanırsa mümkün olabilir. Model doğru kurgulandıktan

ve uygun canlı türleri türetildikten sonra genetik algoritma mekanizmalarının sistemi eğitmesi aşamasına geçilir. Bireyler birbirleri ile çiftleştirilir ve belirli bir doğruluğun üzerine çıkanlar ileriki nesillere aktarılır, diğerleri elenir. Genetik algoritmalardaki ana düşünce daha dirençli bireyler yani probleme daha iyi sonuç veren çözümleri aramaktır. Bunu yaparken de canlıların biyolojik süreçlerinden esinlenilmesi gerektiği fikri öne çıkar [14]. Çünkü arzulanan daha doğru sonuçlara ulaşmak için doğanın milyarlarca yıldır kullandığı tekniği kullanır.

Genetik algoritmalar özünde gözetimli birer arama algoritmalarıdır. Bir probleme çözüm olabilecek kompleks bir arama uzayında doğru bir yönlendirme kullanılarak en iyi çözümü bulmayı amaçlar. Doğru yönlendirme yapılmadan aramalar büyük bir uzayda çok fazla zaman alır. Olabilecek her çözümü değerlendirmek yerine arama yönünü doğru belirleyip umut vadeden yönde aramaları kısıtlamak hesaplama süresini etkileyici bir biçimde azaltacaktır. Genetik algoritmalarının uygulanmasının ilk aşaması başlangıç popülasyonunun (*BP*) üretilmesi aşamasıdır. *BP* teknik olarak arama uzayındaki olası çözümlerin bir alt kümesidir. Bu çözümler gözetimli bir öğrenme yöntemi ile yönlendirilerek daha iyi çözümlerin bulunması amaçlanır. *BP*'ndaki her bir alt çözüm birey ya da kromozom olarak adlandırılır. Her bir birey kendisine ait bir gen dizisine sahiptir. Tıpkı biyolojik süreçlerde gerçekleşen iki bireyin çiftleşmesi sonucu her ikisinin genlerinin yeni bireylere aktarılması gibi *GA* ile de bu bireyler çiftleştirerek her iki atanın genleri yeni nesle aktarılır. İki bireyin çiftleştirilmesi işlemine çaprazlama denir. Popülasyondaki her bir birey başka bir birey ile çift oluşturacak şekilde eşleştirilip çaprazlanarak yeni popülasyon oluşturulur. Çaprazlama işlemi yapılırken mutasyon adı verilen bir ekleme yapılır. Biyolojik süreçlerde bir gen kendisini kopyalarken bazı kopyalama hatalarının olduğu gözlemlenmiştir. Bu hatalara mutasyon adı verilmiştir. Mutasyonlar tamamen rastgele meydana gelen kopyalama hatalarıdır. Herhangi bir mantığı yoktur. Genlerde meydana gelen mutasyonlar faydalı, zararlı veya etkisiz olabilirler. Zararlı mutasyonlar bireyin gen bütünlüğüne ciddi zarar verirse o bireyi popülasyonda arka sıralara düşürecektir. Diğer taraftan faydalı mutasyonlar bireyi kendi popülasyonundaki diğer bireylerden daha avantajlı bir konuma getirecek ve aramanın yönünü daha doğru taraflara çekecektir. Her bir popülasyondaki tüm bireyler bir jenerasyon olarak adlandırılır. Bu işlem belirli bir sayıda tekrarlanır ve jenerasyonlar oluşturulur. Yeni jenerasyon oluşturulurken bir önceki popülasyondaki bireyler kullanılır. Bu bireylerin her biri çözülmek istenen problem için olası bir çözümdür. Ancak her birey

iyi çözüm getiremez. Bu nedenle ileriki jenerasyona aktarılacak bireylerin doğru seçilmesi gerekir. Bu seçme işlemi yapılırken uygunluk değeri (*UD*) olarak adlandırılan bir değer kullanılır. Her bireyin problemin çözümüne ne kadar yaklaştığı, bir başka deyişle istenen optimum çözüme ne kadar uygun olduğunun ölçülmesi gerekir. Bu işlem yapılırken her problem için farklı bir yöntem kullanılır ve bir bireyin ileriki aşamaya geçip geçmeyeceğinin bir nevi karar vericisi olan uygunluk değeri hesaplanır. Her bir birey için hesaplanan bu uygunluk değeri teknik olarak bireyin bir sonraki jenerasyona geçip geçmeme olasılığı olarak düşünülebilir. Birey, optimum çözüme ne kadar yakınsa uygunluk değeri o kadar iyi çıkacaktır ve bir sonraki jenerasyona aktarılma olasılığı da diğer bireylere göre yüksek olacaktır. Dolayısıyla her bir jenerasyonda optimum çözüme yakın olan bireyler yüksek olasılıkla bir sonraki popülasyona dahil edilecekken yakın olmayanların dahil edilmesi daha düşük bir olasılıkla gerçekleşecektir. Sonuç olarak her yeni popülasyon beklenen çözüme bir adım daha yaklaşacak ve bireylerden bir tanesi çözüme ulaşacaktır.



Şekil 6.1. *GA araması ile en iyi çözüme gidilmesi*

Genetik algoritmalardaki önemli problemlerden bir tanesi aramaların yerel optimum çözümlere takılıp kalmasıdır [41]. Bir popülasyondaki bazı bireylerin uygunluk değeri diğer bireylerden daha iyi olacaktır. Dolayısıyla bu bireylerin seçilme olasılıkları ilk jenerasyonlarda daha yüksek olacaktır. Şekil 6.1'de gösterildiği gibi ilk jenerasyonlarda *x* bireyinin *y* bireyinden daha uygun olması onun global maksimum çözüme daha yakın olacağı anlamına gelmez. Aksine *x* bireyinin ve onun çocuklarının çok fazla seçilmesi durumunda aramanın yerel maksimuma yaklaşacağı görülmektedir. Bu durumu engellemek için popülasyon büyüklüğünün probleme uygun olarak deney ve gözlemler ile seçilmesi, çaprazlama işlemlerinin popülasyondaki bireylerin gen çeşitliliğini koruyacak şekilde yapılması ve mutasyon oranının yine deneylerle en optimum değerde seçilmesi gereklidir.

Genetik algoritmanın uygulamaya konması için öncelikle yapılması gereken başlangıç popülasyonunun oluşturulmasıdır. *OF* ve *GKOF* yöntemleri ile *AK*'nın kendisine en benzer *k* tane komşusu ile benzerlikleri kullanılarak *BP* üretilmiştir. Genetik algoritmaya iki yöntemin benzerlik değerleri verilir ve *GA* bu benzerlikler iyileştirerek daha iyi sonuçlar veren çözümleri arar. *a* kullanıcısının oylamaları $a=(r_1, r_2, \dots, r_m)$ vektörü ile temsil edilir. Buradaki *r* değerleri *OF* yönteminde gerçek oy değerlerini *GKOF* yönteminde gizlenmiş z-skor değerlerini temsil etmektedir. *a* kullanıcısının kendisine en benzer *k* tane kullanıcıyla olan benzerlik değerleri $w_a=(w_{a1}, w_{a2}, \dots, w_{ak})$ vektörü ile tanımlanır. Daha sonra *BP* üretmek için *w* dizisi çoklanır. Burada yapılan işlem *w* vektöründeki elemanların rastgele üretilmiş olan *k* tane indeksle yerlerinin değiştirilmesi şeklindedir.

$$p\_w_{a1} = (w_{a8}, w_{a19}, \dots, w_{a3})$$
$$p\_w_{a2} = (w_{a23}, w_{a1k}, \dots, w_{a5}) \quad (6.1)$$

Örnek olarak üretilen başlangıç popülasyonundaki iki birey eşitlik 6.1'de gösterilen p_w_{a1} ve p_w_{a2} şeklinde olabilir. Burada her bir p_w vektörü popülasyondaki bir bireyi temsil ederken her bir w_{ai} değeri bireyin bir genini temsil eder. Toplamda *p* popülasyon büyüklüğü kadar birey üretilir.

$$g_{a1} = (p\_w_{a1}, p\_w_{a2}, \dots, p\_w_{ap}) \quad (6.2)$$

Eşitlik 6.2 matrisi ile de g_{a1} olarak adlandırılan birinci jenerasyon oluşturulmuş olur. Burada her bir p_w_a, *kx1* boyutunda bir vektör ve g_a matrisi bu vektörlerin

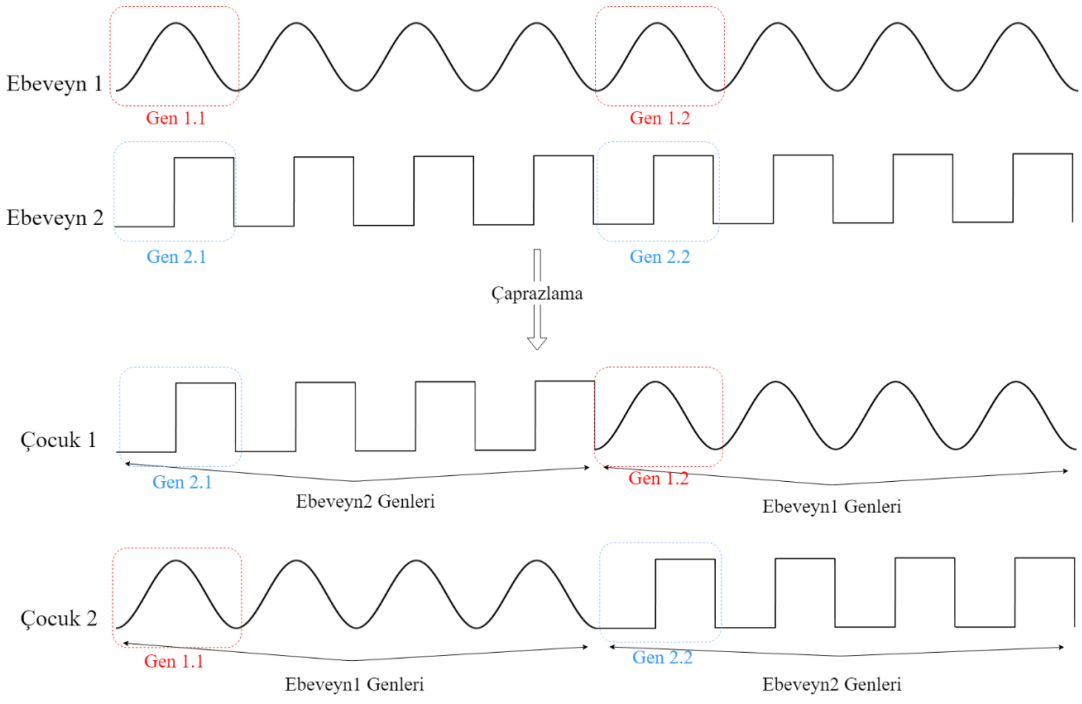
birleşiminden oluşan *pxk* boyutlarındaki bir matristir. Elde edilen başlangıç popülasyonu ile belirli bir iterasyon sayısı kadar yeni jenerasyonlar oluşturulur. Her jenerasyonda oluşturulan bireyler genetik algoritma operatörleri ile geliştirilerek uygunluk değerleri daha yüksek yeni bireyler üretilir ve problemin en optimum çözümünün bulunması amaçlanır [16].

6.1. Genetik Algoritma Operatörleri

Genetik algoritmalar üç ana aşamadan olur: Çaprazlama, mutasyon ve seçme. Başlangıç popülasyonu oluşturulduktan sonra popülasyondaki bireyler birbirleri ile çaprazlanarak yeni bireyler oluşturulur. Çaprazlama ile oluşturulan yeni bireylere oluşturulma aşamasında çok küçük olasılıklarla mutasyonlar eklenir. Son aşamada ise bu bireylerin beklenen çözüme ne kadar yakın oldukları ölçülür ve bazı bireyler uygunlukları oranında seçilme ihtimalleri hesaplanarak yeni jenerasyon için seçilirler.

6.1.1. Çaprazlama

Çaprazlama, popülasyon içerisinden rastgele seçilmiş iki bireyin genlerinin belirli oranlarla alınarak iki yeni bireyin oluşturulması işlemidir. Şekil 6.3 çaprazlama işlemini görsel olarak açıklamaktadır.



Şekil 6.2. *İki atadan iki yeni çocuğun üretildiği çaprazlama işlemi*

Burada görüldüğü üzere *Ebeveyn1* ve *Ebeveyn2* olarak adlandırılan iki atadan *Çocuk1* ve *Çocuk2* olarak adlandırılan iki yeni birey üretilmiştir. Genleri sinüs dalgası olarak sembolize edilmiş olan *Ebeveyn1*'in ilk dört geni *Çocuk2*'ye, son dört geni ise *Çocuk1*'ye geçmiştir. Genleri kare dalga olarak üretilen *Ebeveyn2*'nin ise aynı şekilde ilk dört geni *Çocuk1*'e, son dört geni de *Çocuk2*'ye geçmiştir.

İki bireyin genlerinin çaprazlanması sırasında genleri temsil eden *k* tane komşularının benzerlik değerleri birbirleri ile çaprazlanır. Çaprazlama işlemi yapılırken her bir atanın sahip olduğu *k* tane genden hangi çocuğa ne kadar verileceğinin önceden belirlenmiş olması gerekir. Bu bağlamda eşitlik 6.3'de gösterilen [18] *p_w* bireylerinin genleri birbirleri ile çaprazlanırken her atadan belirli bir miktarda gen alınarak çocuk bireyler üretilmiştir. İki atadan üretilen iki yeni çocuk birey olan *p_w'* bireylerine Bölüm 6.1.2'de anlatıldığı gibi mutasyonlar eklenerek sonraki aşamaya geçilmiştir.

$$p\_w'_{a1} = p\_w_{a1} \,.\, c + p\_w_{a2} \,.\, (1\text{-}c)$$
$$p\_w'_{a2} = p\_w_{a2} \,.\, c + p\_w_{a1} \,.\, (1\text{-}c) \qquad (6.3)$$

Bu işlem yapılırken [0, 1] aralığında bir *c* parametresi belirlenmiş her iki atadan *c* ve *1-c* oranlarında genler alınarak çocuk bireylere aktarılmıştır. Eşitlik 6.1'de gösterildiği gibi *p_w* vektörleri bireylerin benzerlik değerlerini ihtiva etmektedir. *a* kullanıcısının en yakın *k* komşusunun benzerlik değerlerini içeren ve popülasyondan rastgele seçilmiş birinci birey olan p_w_{a1} vektörünün *k* tane elemanından oransal olarak *c* tanesi birinci çocuk olan $p\_w'_{a1}$ vektörüne, *1-c* tanesi ise ikinci çocuk olan $p\_w'_{a2}$ vektörüne aktarılmıştır. Yine aynı mantıkla popülasyondan rastgele seçilen ikinci ata olan p_w_{a2} vektörünün oransal olarak *c-1* tanesi birinci çocuk olan $p\_w'_{a1}$ vektörüne, *c* tanesi ise ikinci çocuk olan $p\_w'_{a2}$ vektörüne aktarılmıştır. Burada iki atadan alınıp çocuklara aktarılan her bir benzerlik, bireyin bir genini temsil etmektedir. Atalardan çocuklara genler aktarılırken çaprazlanacak iki ata rastgele seçilmiş gerekli olan genetik çeşitlilik sağlanmıştır. Çaprazlanacak bireylerin rastgele değil de belirli bir sırada yapılması aynı ataların çocuklarının sürekli birbirleri ile çaprazlanmasına neden olacak ve farklı bölgelerde arama yapan bireylerin arama uzayının diğer bölgelerine erişmesini engelleyecektir. *c* parametresi belirlenirken Bölüm 7'de anlatıldığı gibi bir takım deneyler yapılmış ve optimum değer seçilmiştir. Yapılan çalışmada *OF* ve *GKOF* için deneyler ayrı ayrı yapılmış ve her iki yöntem için en uygun *c* parametresi bulunmuştur.

Algoritma 6.1. Çaprazlama

Require: Çaprazlama Katsayısı (*c*)

1: function CrossOver(*populationSims*)

2: *(p, k) ← size(populationSims)* ← *Popülasyon büyüklüğü (p) ve her popülasyondaki benzerlik sayısı (k)*

3: *populationSims = randomReorder(populationSims)* ← *Popülasyondaki bireylerin sıralarını rastgele değiştir.*

4: *minus1c ← (1 – c)*

5: $populationSimsCrossedOver_{pxk} \leftarrow 0$

6: for *i ← 1:2 to p* **do**

7: **for** *j ← 1 to k* **do**

8: *newSim1 = populationSims(i, j) × c +* ← *İki bireyin benzerlik değerlerini çaprazla*

populationSims(i+1, j) × minus1c

9: *newSim2 = populationSims(i+1, j) × c +*

populationSims(i, j) × minus1c

10: *populationSimsCrossedOver (i, j) = newSim1* ← *Çaprazlanmış yeni benzerlik*

11: *populationSimsCrossedOver (i+1, j) = newSim2*

12: **end for**

13: **end for**

14: return *populationSimsCrossedOver*

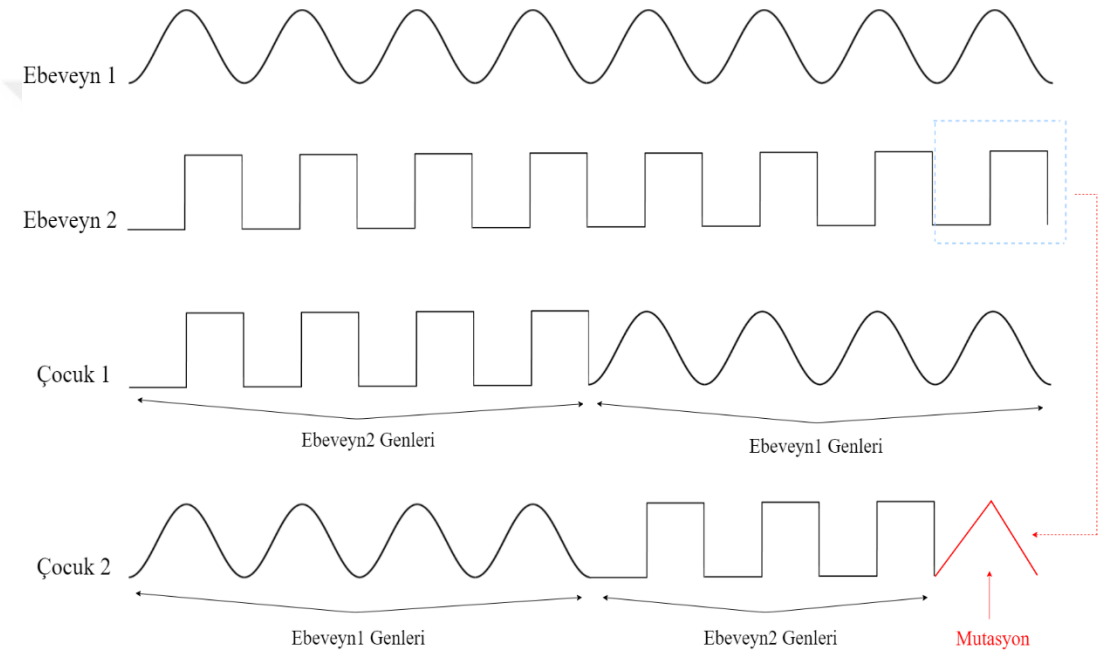
15: end function

Algoritma 6.1 çaprazlama işleminin adımlarını göstermektedir. Burada görüldüğü üzere popülasyondaki bireylerin indeksleri rastgele değiştirildikten sonra bireyler ikişer ikişer döngüye sokulur ve iki atadan belirli sayıda gen alınarak yeni bireyler üretilir.

6.1.2. Mutasyon

Mutasyonlar bireylerin genetik kodlarında meydana gelen rastgele değişimleridir. Biyolojik süreçlerdeki gen kopyalamalarında meydana gelen hatalar bir diğer deyişle mutasyonlar, oluşan yeni genlerin atalarına %100 benzerlik göstermemelerine neden olur. Bu durum her ne kadar kötü bir sonuç gibi görünse de bazen çok faydalı sonuçlar doğurabilmektedir. Şekil 6.1'de gösterildiği gibi bir bireyin yerel optimum sonuçlara takılıp kalmaması için çaprazlamanın yanı sıra bazı bireylerin arama yönünü küçük değişikliklerle değiştirmesi gerekmektedir. Burada mutasyonlar devreye girmektedir. Şekil 6.4'de görüldüğü üzere *Çocuk1* iki ebeveynin genlerini kusursuz bir şekilde

almıştır. Ancak *Çocuk2*'ye, *Ebeveyn1*'in genleri doğru kopyalanırken *Ebeveyn2*'den alınan genlerden bir tanesinin kopyalanması esnasında mutasyon meydana gelmiştir. Bu durumda *Çocuk2* iki atanın ortak genlerinin yanı sıra başka bir gene daha sahip olmuştur. Bu mutasyonun *Çocuk2* için faydalı mı yoksa zararlı mı olduğunu anlamak ancak uygunluk değeri ölçüldüğünde anlaşılabilir. Faydalı bir mutasyon bu yeni bireyi popülasyondaki seçkin bireylerden bir tanesi yapabileceği gibi zararlı bir mutasyon onu istenmeyen bir yönde değiştirerek sonraki jenerasyonlarda yer alamamasına neden olabilir.



Şekil 6.3. *Çaprazlama esnasında ortaya çıkan mutasyon*

Mutasyonlar çok küçük olasılıklarla meydana gelirler. Zaten bir bireyin gen yapısı gereğinden fazla değiştirilirse işlevsel olarak anlamsız bir hale gelecek ve sonuçta uygunluk değeri büyük oranda düşecektir. Burada yapılması gereken uygun bir mutasyon oranı (m) belirlemektir. Bu oran belirlenirken her problemin kendi dinamikleri esas alınarak yine deney ve gözlemlerle en uygun değer bulunmalıdır. Bu kapsamda *OF* ve *GKOF* yöntemlerinde elde edilen benzerlik değerleri farklı şekillerde hesaplandığından ikisi için ayrı ayrı deneylerle uygun mutasyon oranları belirlenmiştir. Eşitlik 6.1'de gösterilen ve her biri bir bireyi temsil eden p_w vektörlerindeki her bir gen (w) mutasyona uğrayıp uğramayacağına karar verilmek üzere değerlendirilir. Popülasyondaki tüm

bireylerin tüm genleri için [0, 1] aralığında birer rastgele sayı üretilir, eğer bu sayı mutasyon olma oranının üstündeyse ilgili genin değerine rastgele bir değer yazılır. Bu şekilde Algoritma 6.2'de de mutasyon sürecinin işlem adımlarında gösterildiği gibi popülasyondaki bireylerin genleri küçük olasılıklarla değiştirilir.

Algoritma 6.2. Mutasyon Ekleme

Require: Mutasyon Olasılığı (*m*)

1: function Mutation(*populationSims*)

2: *(p, k) ← size(populationSims)* ← *Popülasyon büyüklüğü (p) ve her popülasyondaki benzerlik sayısı (k)*

3: $mutationArray_{pxk}$ *= randomPermutation(p, k, 1)* ← *[0, 1] aralığında rastgele sayı üret*

4: for *i ← 1 to p* **do**

5: **for** *j ← 1 to k* **do**

6: **if** *mutationArray(i, j) < m* **then**

7: *populationSims(i,j) = rand(1)* ← *Benzerlik değerini [0, 1] aralığında rastgele bir sayıyla değiştir*

8: **end if**

9: **end for**

10: **end for**

11: return *populationSims*

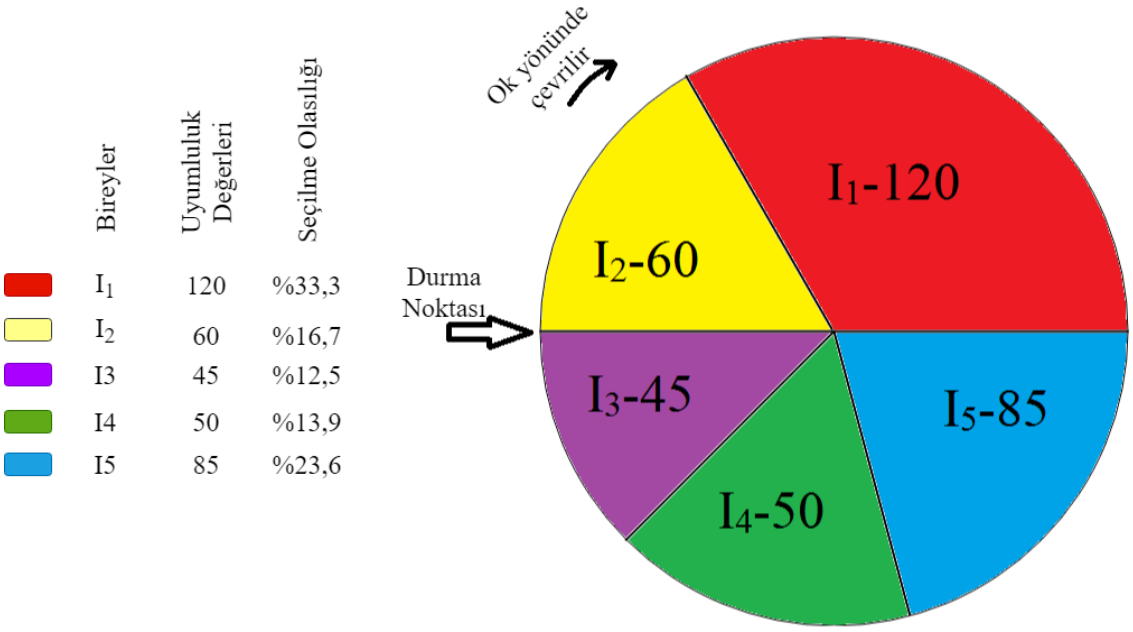
12: end function

6.1.1. Seçme

Genetik algoritmaların seçme aşaması hangi bireylerin ileriki jenerasyona aktarılacağının karar verildiği aşamadır. Bu işlem her bir bireyin problemin çözümüne ne kadar yaklaştığı ölçülerek gerçekleştirilir. Uygunluk değeri adı verilen bu değer her problemde farklı şekilde anlamlandırılır. Uygunluk fonksiyonu (*UF*), bireyin uygunluk değerini ölçmek için kullanılır. Bir birey problemin çözümüne ne kadar uygunsa seçilme olasılığı o kadar yüksektir. Burada seçilecek bireyler Eşitlik 6.3'de gösterilen çaprazlama sonunda oluşan *p_w'* bireylerinin mutasyona uğradıktan sonraki versiyonlarıdır. Yapılan çalışmada uygunluk değerini hesaplayacak olan uygunluk fonksiyonu olarak Denklem 6.4'de gösterilen ortalama mutlak hata (*OMH*) fonksiyonu kullanılmıştır.

$$OMH = \frac{\sum_{i=1}^{n} |p'_i - p_i|}{n} \tag{6.4}$$

Burada n tane ürüne sistemin ürettiği öneriler p', kullanıcının ürüne verdiği gerçek oy değerini ise p ile gösterilmiştir. *OMH*'nın hesaplanması için öncelikle öneri üretilmesi gerekmektedir. Bireyler eğitim aşamasında öncelikle değeri önceden bilinen seçili ürüne öneriler üretir daha sonra üretilen bu önerilerin *OMH* formülü ile hata oranları hesaplanır. Hata oranı düşük olan bireylerin uygunluk değerleri daha küçük çıkacaktır. Bu sayede ileriki jenerasyonlara aktarılma olasılıkları daha yüksek olacaktır.



Şekil 6.4. *Roulette-Wheel Algoritması*

UD hesaplanan bireyler Şekil 6.5'de gösterilen *Roulette-Wheel* algoritması ile seçilerek sonraki aşamaya geçilir. Burada yapılan işlem her bir bireyin uygunluk değerini büyüklükleri ile orantılı olarak bir çarkın içerisine yazdıktan sonra çarkı çevirerek hangi bireyin seçileceğine karar vermek olarak özetlenebilir.

Algoritma 6.3. Roulette-Wheel Seçme Algoritması

1: function RouletteWheel(*mae, populationSims*)

2: $populationSimsSelected_{pxk} \leftarrow 0$

3: $p \leftarrow size(mae)$ ← *Popülasyon büyüklüğü*

4: $maeDivTo1_{px1} = 1/mae$

5: $sumDivTo1_{px1} = maeDivTo1 / sum(maeDivTo1)$

6: $cumSum_{px1}$ *= cumulativeSum(sumDivTo1)* ← *Kümülatif toplam, [0, 1] aralığında*

7: for *i* ← *1 to p* **do**

8: *index = getIndex(cumSum, rand(1), gt)* ← *Üretilen rastgele sayının kümülatif toplamda denk geldiği indeksi bul*

9: *populationSimsSelected(i) = populationSims(index)*

10: end for

11: return *populationSimsSelected*

12: end function

Algoritma 6.3 *Roulette-Wheel* algoritmasının işlem adımlarını göstermektedir. Burada görüldüğü üzere öncelikle her bir bireyin *OMH* bilgileri *mae* dizinde tutulmaktadır. Beş adet bireyden oluşan örnek bir *OMH* dizisi Tablo 6.1' de gösterilmiştir.

Tablo 6.1. *Örnek OMH dizisi ve diğer değişkenler*

	birey$_1$	**birey$_2$**	**birey$_3$**	**birey$_4$**	**birey$_5$**
OMH	0,8	0,5	0,4	1,0	0,8
1/OMH	1,25	2	2,5	1	1,25
Toplama Bölüm	0,15625	0,25	0,3125	0,125	0,15625
Kümülatif Toplam	0,15625	0,40625	0,71875	0,84375	1

OMH bir hata fonksiyonu olduğu için amaç bu değerin küçük olmasını sağlamaktır. Bu nedenle öncelikle *OMH* dizisindeki elemanlar *1*'e bölünür ve dizi tersine çevrilir. Bu şekilde küçük olan *OMH* değerlerinin daha büyük, büyük olan *OMH* değerlerinin daha küçük halde temsil edildiği *maeDivTo1* dizisi elde edilir. Tablo 6.1'de bu dizi *1/OMH* satırında görülmektedir. *1/OMH* değerlerinin toplamı hesaplanır. Örnekteki toplam değerin *8* olduğu hesaplanabilir. *1/OMH* dizisindeki elemanların toplam değer olan *8*'e bölünmesi ile Tablo 6.1'de görülen *toplama bölüm* dizisi oluşturulur. *toplama bölüm* satırında gösterilen değerler bireylerin seçilme olasılıklarını vermektedir. Bu dizideki elemanların kümülatif toplamları *kümülatif toplam* satırında hesaplanır. Tablo 6.1'de görüldüğü üzere kümülatif toplam [0, 1] aralığına indirgenmiştir. Bu aşamadan sonra Algoritma 6.3'de gösterildiği gibi popülasyon büyüklüğü sayısında bir döngü ile [0, 1] aralığında rastgele sayılar üretilerek, üretilen her sayının *kümülatif toplam* dizisinde denk geldiği aralıktan hangi iki değerin arasında ise o değerlerden büyük

olanın indeksindeki birey seçilir. Örneğin rastgele üretilen sayı *0,28* ise $birey_2$, *0,74* ise $birey_4$ seçilecektir. Burada dikkat edilirse *OMH* değeri *0,4* olan $birey_3$ ün denk geldiği aralık *toplama bölüm* sütununda hesaplanan *0,3125* iken *OMH* değeri *1* $birey_4$ ün denk geldiği aralık *0,125*'dir. Yani *OMH* değeri küçük olan bireylerin seçilme ihtimaller daha yüksektir. Diğer taraftan *OMH* değerleri eşit olan $birey_1$ ve $birey_5$ eşit olasılıklarla seçilecektir. Bireyler *OMH* değerleri ile ters orantılı olasılıkları ile seçildikten sonra seçilen bireyler bir sonraki jenerasyona aktarılır ve tekrar çaprazlama ve mutasyon işlemlerine tabi tutulur ve bu işlemler belirli bir iterasyon boyunca tekrarlanarak jenerasyonlar oluşturulur. Bu jenerasyonlar boyunca oluşturulan popülasyonlardaki bireylerden bir tanesi beklenen hata değerine en yakın sonucu üretecektir ve bu bireyin temsil ettiği benzerlikler *AK* için en iyi sonucu vermesi beklenen benzerlikler olarak kaydedilecektir.

7. DENEYSEL ÇALIŞMALAR VE SONUÇLARIN EMPİRİK ANALİZİ

Yapılan çalışmalar *OF* ve *GKOF* yöntemleri için ayrı ayrı gerçekleştirilmiş ve sonuçları analiz edilmiştir. Yapılan her bir deneyde genetik algoritmada kullanılan değişkenlerin en doğru değerlere eşitlenmesi amaçlanmıştır. Ardından en iyi deney sonuçlarına göre *OF* ve *GKOF* sistemleri test edilerek yapılan çalışmanın sistemin genel doğruluğu üzerindeki etkisi gözlemlenmiş ve sonuçlar tablo ve grafiklerle gösterilmiştir.

7.1. Veri Seti

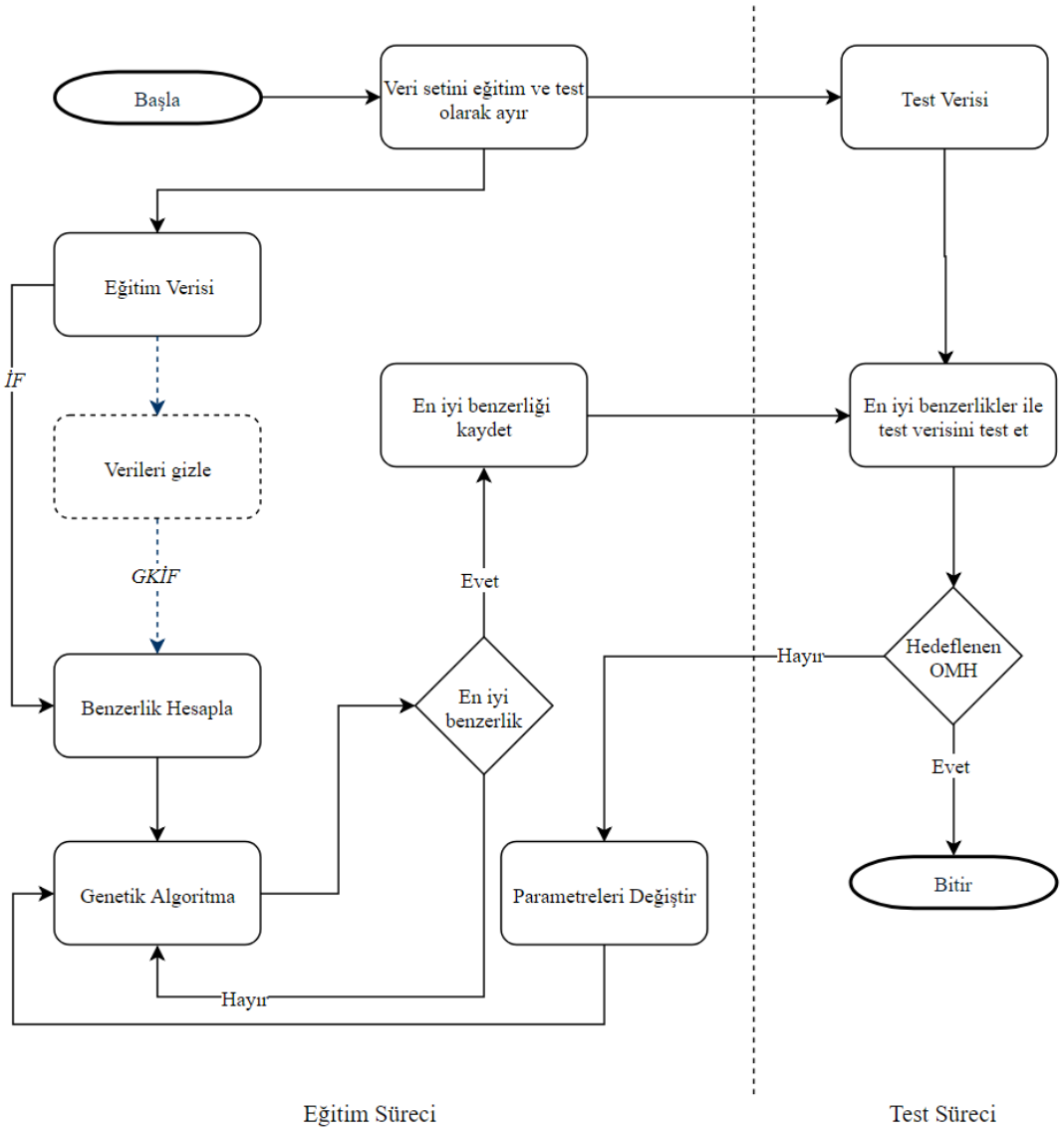
Yapılan çalışmada kullanılan veri seti, Minnesota Üniversitesi'nde GroupLens (http://www.grouplens.org) tarafından toplanmış halka açık bir şekilde yayınlanan MovieLens Public (MLP) veri setidir. MLP veri setinde 943 kullanıcı toplamda 1682 filme oy vermiştir. Her kullanıcının en az 20 ürüne oy verdiği veri setinde toplam oylanmış ürün sayısı 100.000'dir. Buradan da anlaşılacağı üzere veri setindeki doluluk oranı yaklaşık %6,3'dür. Yapılan oylamalar [1, 5] aralığında olup en beğenilen ürüne 5 yıldız verilirken en az beğenilen ürüne ise 1 yıldız verilerek yapılmıştır. Oylanmamış veriler ise 0 olarak gösterilmektedir.

Yapılan deneylerde kullanılmak üzere veri seti, eğitim verisi ve test verisi olarak ikiye ayrılmıştır. Her kullanıcının 5 tane verisi alınarak test verisi olarak saklanmış ve geriye kalan oylamalar eğitim amaçlı kullanılmıştır. Sonuç olarak 4715 tane oylama verisi test için ayrılmış ve bu oylamaların yerlerine 0 değeri yazılmış, geriye kalan yaklaşık 95.000 tane oylama verisi ise eğitim aşamasında sistemin eğitilmesinde kullanılmıştır.

7.2. Eğitim ve Test Süreçleri

Genetik algoritma kullanılarak en yakın komşuluk esasına dayanan *kNN* algoritması ile elde edilen benzerlik değerlerinden daha iyi benzerlik değerleri türetmek mantığı ile yapılan çalışma eğitim ve test aşaması olarak iki adımda gerçekleştirilmiştir. Eğitim aşamasında Şekil 7.1'de gösterildiği gibi geriye her kullanıcı için en optimum benzerlik değerlerini döndüren bir model oluşturulmuştur. Elde edilen optimum benzerlik değerleri daha sonra bir test aşamasından geçirilmiştir. Test aşamasında elde edilen değerler istenen sonuçlara ulaşana dek eğitim aşaması farklı parametreler ile tekrarlanmış ve sonuçta gerek *OF* gerekse *GKOF* yöntemleri için en optimum benzerlik değerleri bulunmuştur.

Şekil 7.1'de genetik algoritma ile öneri sisteminin geliştirilmesinin eğitim ve test aşamalarının ana hatlarının akış diyagramı gösterilmektedir. Yapılan çalışmada *OF* ve *GKOF* yöntemleri ayrı ayrı ele alınmış ve her iki yöntemin sonuçları gösterilmiştir. *GKOF* yöntemi uygulama aşamasında *OF* yöntemine bir katman daha eklenerek gerçekleştirilir. Bu katman gizlilik katmanıdır. *OF* yönteminde benzerlikler Denklem 4.1 kullanılarak benzerlikler hesaplanır ve bu benzerlikler ile Denklem 4.2 ile öneri üretilir. *GKOF* yönteminde Denklem 5.1 ile önce z-skorlar hesaplanır, bu z-skorlar Bölüm 5.2'de anlatıldığı gibi gizlenir, Denklem 5.2 ile z-skorlar skaler çarpılarak benzerlikler hesaplanır ve Denklem 5.3 kullanılarak öneri üretilir [38].



Şekil 7.1. *Genetik Algoritma Eğitim ve Test Akış Diyagramı*

Eğitim aşamasında bir model ortaya koymak için öncelikle sistem değişkenlerinin optimum değerlerinin bulunması gerekmektedir. Mutasyon oranı (*m*), çaprazlama katsayısı (*c*), popülasyon büyüklüğü (*p*) ve jenerasyon sayısı (*g*) değerlerinin gerek *OF* gerekse *GKOF* yöntemleri için belirlenmesi gerekmektedir. Şekil 7.1 eğitim ve test süreçlerinde izlenen yolu göstermektedir. Öncelikle her bir sistem değişkenine mantıklı bir ilk değer ataması yapıldıktan sonra, bir tane sistem değişkeni farklı değerler ile değiştirilir ve diğer değişkenler sabit tutularak sistem eğitilir. Daha sonra bu değişkenin her bir değeri için elde edilen sonuçlar karşılaştırılır, en iyi test sonucunu veren değişken değeri bulunmuş olur. Ardından aynı şekilde diğer değişkenler tek tek eğitim ve test aşamasından geçirildikten sonra tüm sistem değişkenlerinin optimum değerleri belirlenir. *OF* ve *GKOF* yöntemlerinin benzerlik hesabı ve öneri üretim aşaması farklı olduğundan her iki yöntem için süreçler ayrı ayrı yapılmış ve her iki yönteme en uygun sistem değişkenleri bulunmuştur.

Algoritma 7.1. Genetik Algoritma Eğitim Aşaması

Require: Eğitim kullanıcı-ürün matrisi *(U_{nxm})*, Jenerasyon Sayısı (*g*), Popülasyon Büyüklüğü *(p)*, Komşuluk Sayısı (*k*)

1: function TrainGA

Initialize: $GAsims_{nxk} \leftarrow 0$ ← *Üretilecek benzerliklere ilk değer ataması*

2: for all *kullanıcılar* $\in U(i \leftarrow 1...n)$ **do**

3: **for all** *ürünler* $\in U(j \leftarrow 1...m)$ **do**

4: $temp \leftarrow U(i,j)$

5: $U(i,j) \leftarrow 0$ ← *i kullanıcısının j verisini 0 yap, bu veri üzerinden sistemi eğit*

6: **if** *method is CF* **then**

7: $sim_{nx1} = pcc(U(i), i)$ ← *Denklem 4.1 ile benzerlik hesala (OF)*

8: **else**

Veri Gizleme

9: $zscore_{mx1} = calcZscore(U(i))$ ← *j verisi 0 yapıldıktan sonraki z-skor*

10: $allZscores = getOtherZscores(i)$ ← *i dışındaki kullanıcıların z-skorları*

11: **for** $t \leftarrow 1$ *to n-1* **do**

12: $sim(t)=dot(zscore, allZscores(t))$ ← *Denklem 5.2 ile benzerlik hesapla (GKOF)*

13: **end for**

14: $sim = dataDisguise(sim)$ ← *Verileri Gizle*

```
15:     end if
16:     sim = scale(sim, -1, 1)                          ← Benzerlikleri [-1, 1] aralığına ölçekle
    Genetik Algoritma
17:     sortedSim_nx1 = sort(sim, desc)                  ← Benzerlikleri büyükten küçüğe sırala
18:     bestSim_nx1 = first(sortedSim, k)                ← İlk k benzerliği al
19:     minMae ← Inf
20:     for t ← 1 to p do                                ← Başlangıç popülasyonu oluştur
21:       rndIndex_kx1 = randomPermutation(k)            ← k tane rastgele indeks üret
22:       populationSims_pxk(t) = index_at(bestSim, rndIndex)   ← İndekslerdeki benzerlik değerlerini al
23:     end for
24:     for t ← 1 to g do
25:       populationSims = crossOver(populationSims)     ← Algoritma 6.1 ile çaprazlama yap
26:       populationSims = mutation(populationSims)      ← Algoritma 6.2 ile mutasyon ekle
27:       populationPreds_px1 = calcPrediction(populationSims)   ← Denklem 4.2 (OF) veya Denklem 5.3
                                                         (GKOF) ile öneri hesapla
28:       for q ← 1 to p do
29:         mae_px1(q) = MAE(temp, populationPreds(q))   ← Denklem 6.4 ile her popülasyondaki
                                                         her birey için OMH hesapla
30:         if mae(q) < minMae then
31:           minMae = mae(q)
32:           GAsims(i,:) = populationSims(q,:)
33:         end if
34:       end for
35:       populationSims =rouletteWheel(mae, populationSims)   ← Algoritma 6.3 ile seçme işlemi yap
36:     end for
37:   end for
38: end for
39: return GAsims
40: end function
```

Algoritma 7.1 ile eğitim aşamasının adımları anlatılmıştır. Eğitim verisi olarak ayrılmış U_{nxm} matrisindeki n tane kullanıcının m tane oylaması mevcuttur. Her bir kullanıcının eğitim verisindeki her bir oylama değeri geçici bir değişkende tutularak matriste bu hücreye 0 değer yazılır. Ardından bu oylama değeri için öneri üretilir. Bu işlem tüm kullanıcıların eğitim için ayrılan tüm oylamaları için yapılır. Sistem tüm

kullanıcıların tüm eğitim verileri ile eğitildikten sonra elde edilen *GAsims* matrisini test etmek üzere döndürür. Algoritma 7.1'de görüldüğü üzere *OF* ve *GKOF* yöntemleri için ayrı ayrı benzerlikler hesaplandıktan sonra büyükten küçüğe doğru sıralanarak en yüksek değere sahip olan *k* tane benzerlik genetik algoritma ile önceki bölümlerde anlatıldığı gibi daha iyi sonuç verecek şekilde geliştirilmiş ve *UD*'i yüksek olanlar ileriki jenerasyonlar için seçilmiştir. Her kullanıcı için en düşük *OMH* değerini veren benzerlik değerleri *GAsims* matrisinde saklanmış ve bu matris test aşamasında kullanılmıştır.

Eğitim aşamasında elde edilen değişken değerleri ve en iyi sonuçları döndüren benzerlik matrisi Algoritma 7.2'de gösterilen test aşamasıyla her kullanıcı için ayrılan 5 tane oylama bilgisi kullanılarak bu değerlerin doğruluğu test edilmiştir.

Algoritma 7.2. Test Algoritması

Require: Test kullanıcı-ürün matrisi *(T_{nx5})*,

1: function Test(*GAsims*) ← *Eğitim aşamasında üretilen optimum benzerlikler*

2: *avgMae ← 0*

3: for *i ← 1 to n* **do**

4: **for** *j ← 1 to 5* **do**

5: *pr = calcPrediction(GAsims(i), i, j)*

6: *avgMae += MAE(pr, T(i, j))* ← *OMH ları topla*

7: **end for**

8: end for

9: return *avgMae/(n×5)* ← *GAsims için ortalama OMH*

10: end function

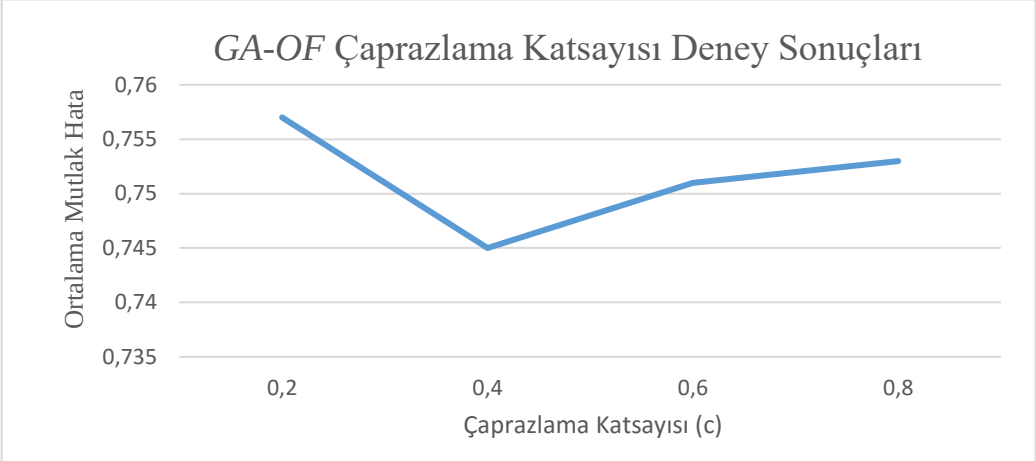
Algoritma 7.2'de görüldüğü üzere *GAsims* matrisi bir önceki aşamada üretilen en iyi benzerlik değerini içermektedir. Bu benzerlik değerleri test aşamasında kullanılan *nx5* tane oylama bilgisi üzerinde test edilerek sonuçlar gözlemlenmiştir.

7.3. Genetik Algoritma Tabanlı OF Deney Sonuçları

Genetik algoritma yaklaşımının geleneksel *kNN* algoritması ile karşılaştırılması yapılmadan önce sistem değişkenlerinin doğru değerlere sabitlenmesi gereklidir. Bu kapsamda *GA* sistem değişkenlerinin belirlenmesi için bir takım deneyler yapılmıştır. *GA* sistem değişkenlerinin hesaplanması esnasında, bir değişken test edilirken diğerleri sabit

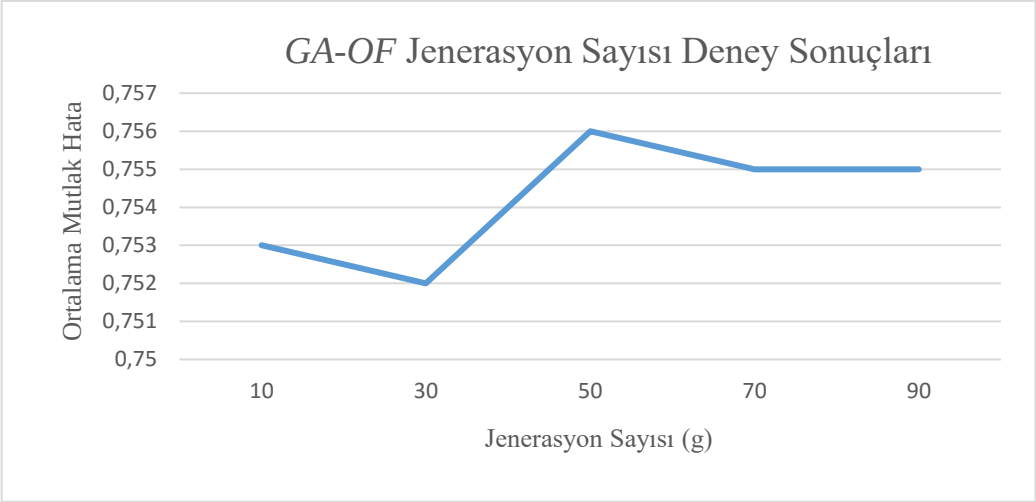
tutulmuştur. Bu değişken için en iyi sonuç elde edildikten sonra sıradaki değişkene geçilmiş ve bu şekilde tüm değişkenler için optimum değerler bulunmuştur. Sistem değişkenlerine hangi değerlerin atanacağının kararı verilirken Denklem 6.4'de verilen *OMH* fonksiyonu kullanılmıştır. Hangi değişkenin *OMH* değeri daha düşükse o değişken seçilmiştir. Yapılan deneylerde bulunan değerler *k* komşuluk değeri *100* değerine sabitlenerek yapılmıştır.

Şekil 7.2 *GA* tabanlı *OF* yöntemi için çaprazlama katsayısının (*c*) bulunması için yapılan deney sonuçları göstermektedir. Buradan görüleceği üzere *c=(0.2, 0.4, 0.6, 0.8)* değerleri için yapılan deneylerde *OF* yöntemi için en iyi sonucu veren katsayı değeri *0.4* olarak ölçülmüştür.



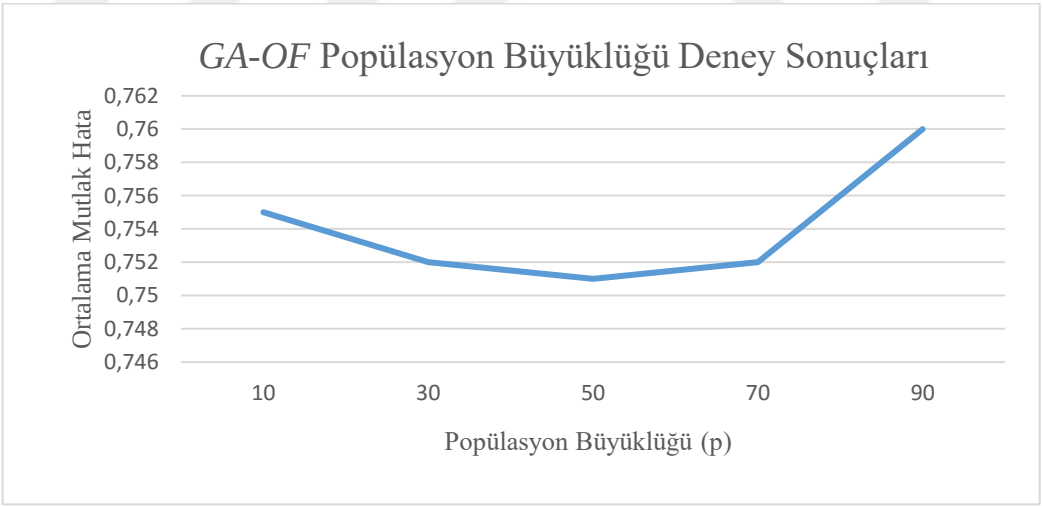
Şekil 7.2. *GA-OF Çaprazlama Deneyi*

Şekil 7.3'de *GA* tabanlı *OF* yöntemi için gerekli olan iterasyon sayısını gösteren jenerasyon sayısı (*g*) değerinin hesaplanması için yapılan deney sonuçları göstermektedir. *g=(10, 30, 50, 70, 90)* değerleri için yapılan deneylerde *OF* yöntemi için en iyi sonucu veren jenerasyon sayısı *30* olarak ölçülmüştür.



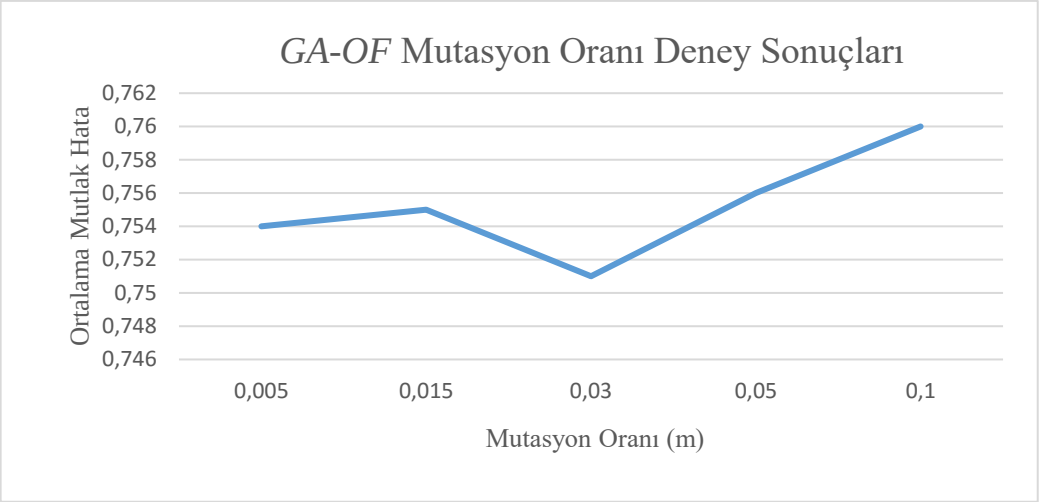
Şekil 7.3. *GA-OF Jenerasyon Sayısı Deneyi*

Şekil 7.4'de yine *GA* tabanlı *OF* yöntemi için oluşturulan popülasyonların büyüklüklerini tutan (*p*) değerinin hesaplanması için yapılan deney sonuçları göstermektedir. *p=(10, 30, 50, 70, 90)* değerleri için yapılan deneylerde *OF* yöntemi için en iyi sonucu veren popülasyon sayısı *50* olarak bulunmuştur.



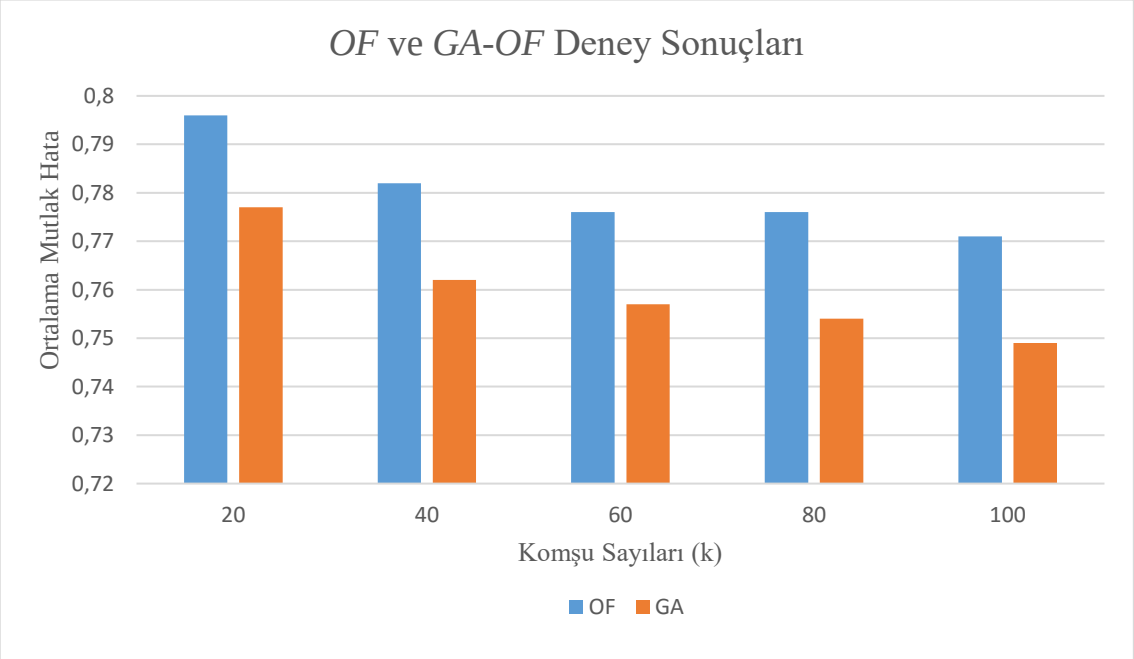
Şekil 7.4. *GA-OF Popülasyon Büyüklüğü Deneyi*

Son olarak Şekil 7.5 *GA* tabanlı *OF* yönteminde bireylerin genlerinin mutasyona uğrama olasılıklarını gösteren mutasyon oranı (*m*) değerinin hesaplanması için yapılan deney sonuçları göstermektedir. *m=(0.005, 0.01, 0.03, 0.05, 0.1)* değerleri için yapılan deneylerde *OF* yöntemi için en iyi sonucu veren mutasyon oranı sayısı *0.03* olarak ölçülmüştür.



Şekil 7.5. *GA-OF Mutasyon Oranı Deneyi*

Tüm sistem değişkenleri en iyi değerlere set edildikten sonra *GA* tabanlı *OF* ile benzerlik değerleri değiştirilmeden öneri hesaplanan geleneksel *OF* yönteminin sonuçları karşılaştırılmıştır. Test aşamasında kullanılmak üzere ayrılmış her kullanıcının 5 tane oylama bilgisine her iki yöntem ile öneri üretilmiş ve sonuçlar Şekil 7.6'de gösterilmiştir. Farklı *k* değerleri için yapılan bu karşılaştırmalar benzerlik değerlerinin *GA* ile iyileştirilmesinden sonra üretilen önerilerin *OMH* değerlerinin geleneksel *OF* yöntemi ile elde edilen önerilerin *OMH* değerlerinden daha iyi olduğu gösterilmektedir.



Şekil 7.6. *Farklı Komşu Sayılarına Göre OF ve GA-OF Karşılaştırması*

Son olarak elde edilen deney sonuçlarının istatistiksel olarak anlamlı olduğunu göstermek için t-test yapılmış ve sonuçları Tablo 7.1'de gösterilmiştir. Serbestlik derecesi 18 olarak belirlenen t-test sonuçlarının *GA* tabanlı *OF* için %95 oranında anlamlı olduğu görülmektedir.

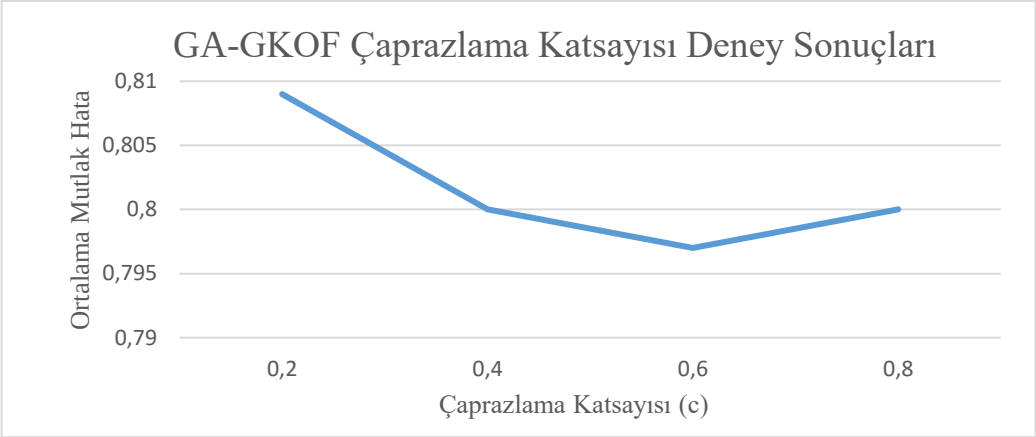
Tablo 7.1. *Geleneksel OF ve GA-OF t-test sonuçları*

k	t	p
20	3,119	0,006
40	3,679	0,002
60	2,840	0,011
80	3,685	0,003
100	3,325	0,003

7.4. Genetik Algoritma Tabanlı GKOF Deney Sonuçları

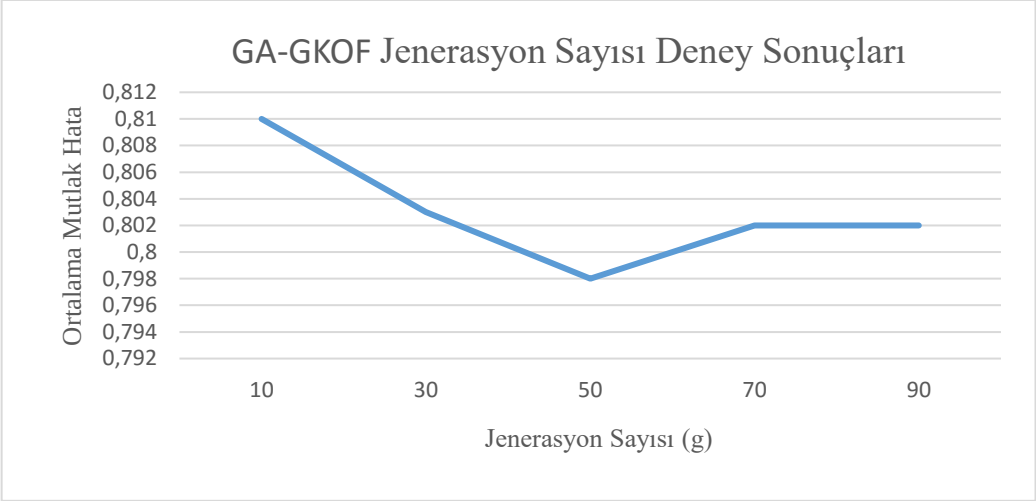
Gizliliği koruyan *OF* kapsamında deneyler benzer şekilde yapılmıştır. Ancak burada elbette eğitim aşamasında üretilen benzerlik değerleri Bölüm 5'de anlatıldığı gibi hesaplanmış ve üretilen öneriler test edilmiştir. *GKOF* ile elde edilen optimum sistem değişkenleri *OF* için üretilenlerden kısmen farklı ölçülmüştür. Bu durum *OF* ve *GKOF* sistemlerinin dinamiklerinin farklı olduğu düşünülünce mantıklı olduğu anlaşılmaktadır. Yine bu kısımda da sistem değişkenlerinin sonuçları iyileştirme oranları ölçülürken Denklem 6.4'de verilen *OMH* fonksiyonu kullanılmış ve *k* komşuluk değeri *100* olarak belirlenmiştir. *OF* yönteminden farklı olarak *GKOF* yönteminde Bölüm 5'de bahsedilen *RBT* tekniğinin parametreleri olan β_{max} ve σ_{max} değerleri de deney boyunca sabit tutulmuş ve β_{max} değeri için *20*, σ_{max} değeri için ise *2* değeri seçilerek deneyler gerçekleştirilmiştir.

Şekil 7.7'da *GA* tabanlı *GKOF* yönteminde çaprazlama işlemi için kullanılan *c* katsayısının *c=(0.2, 0.4, 0.6, 0.8)* değerleri için yapılan deney sonuçları görülmektedir. Burada görüldüğü üzere en düşük *OMH* değerini veren *0.6* katsayı değeri *GKOF* için en iyi çaprazlama katsayısı olarak belirlenmiştir.



Şekil 7.7. *GA-GKOF Çaprazlama Deneyi*

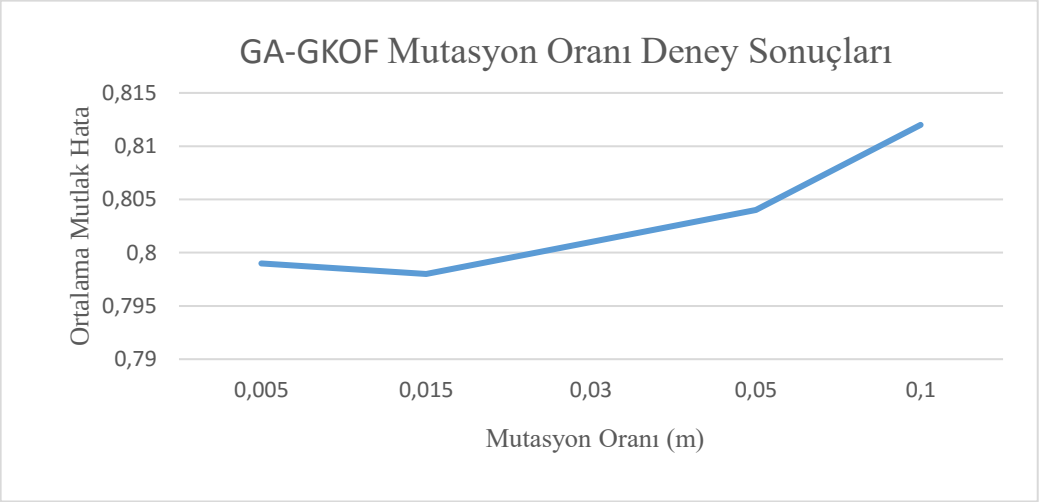
Şekil 7.8'de yine *GA* tabanlı *GKOF* yönteminde kullanılan popülasyonların kaç defa yeni popülasyonlara dönüştürüleceğinin göstergesi olan *g* değerinin *g=(10, 30, 50, 70, 90)* değerleri için yapılan deney sonuçları görülmektedir. *OMH* değeri en küçük jenerasyon sayısı görüldüğü üzere *50* olarak ölçülmüştür.



Şekil 7.8. *GA-GKOF Jenerasyon Sayısı Deneyi*

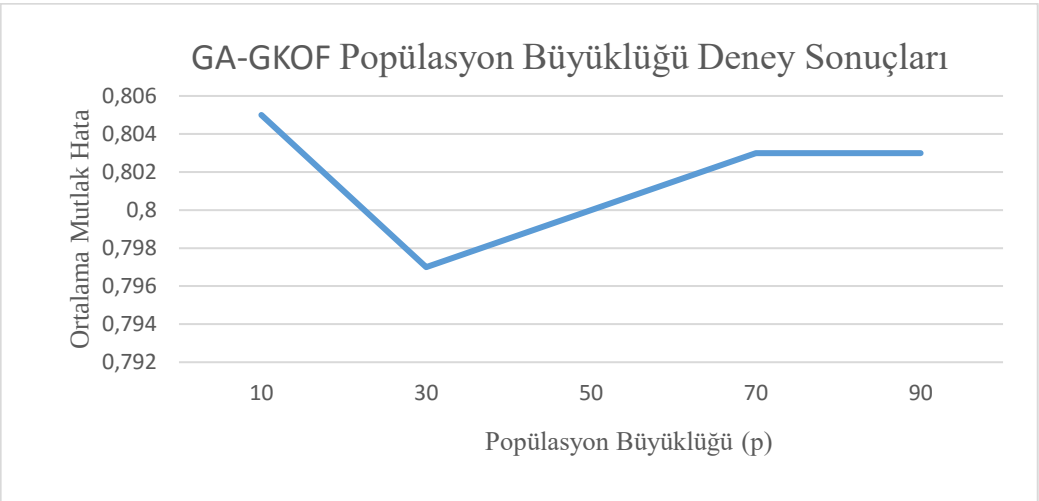
Şekil 7.9'de *GA* tabanlı *GKOF* yöntemi için kullanılan mutasyon oranlarını tutan (*m*) değerinin hesaplanması için *m=(0.005, 0.01, 0.03, 0.05, 0.1)* değerleri için yapılan deney sonuçları gösterilmektedir. Burada da görüldüğü üzere en iyi test sonuçları *0,015* mutasyon oranı değeri ile alınmıştır. *GA* tabanlı *OF* yönteminde en iyi sonucu veren mutasyon oranı *0,03* olarak ölçülmüştür. Burada *GKOF* yönteminde kullanıcıların z-skor bilgilerine rastgele sayılar eklendiği düşünüldüğünde bu sonucun mantıklı olduğu

görülmektedir. Çünkü genetik algoritmanın çalışmasında hayati önem taşıyan gen çeşitliliği *GKOF* yönteminde verilere eklenen rastgele sayılar ile kısmen sağlanmış olur. Bu sayede mutasyonlara *OF* yönteminde olduğu kadar ihtiyaç kalmaz.



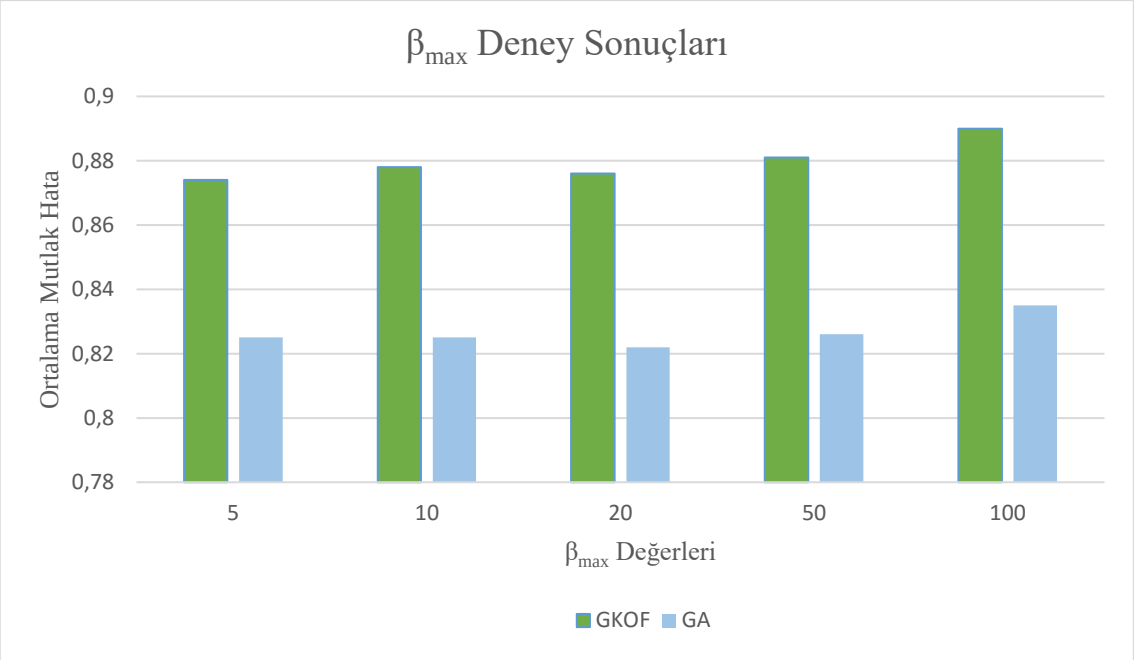
Şekil 7.9. *GA-GKOF Mutasyon Oranı Deneyi*

Şekil 7.10'da *GA* tabanlı *GKOF* yönteminde oluşturulan popülasyonların büyüklüklerini tutan (*p*) değerinin hesaplanması için; *p=(10, 30, 50, 70, 90)* değerleri ile yapılan deney sonuçları gösterilmiştir. Yapılan deneylerde *GKOF* yöntemi için *30* popülasyon büyüklüğü en iyi sonucu veren değer olarak belirlenmiştir.



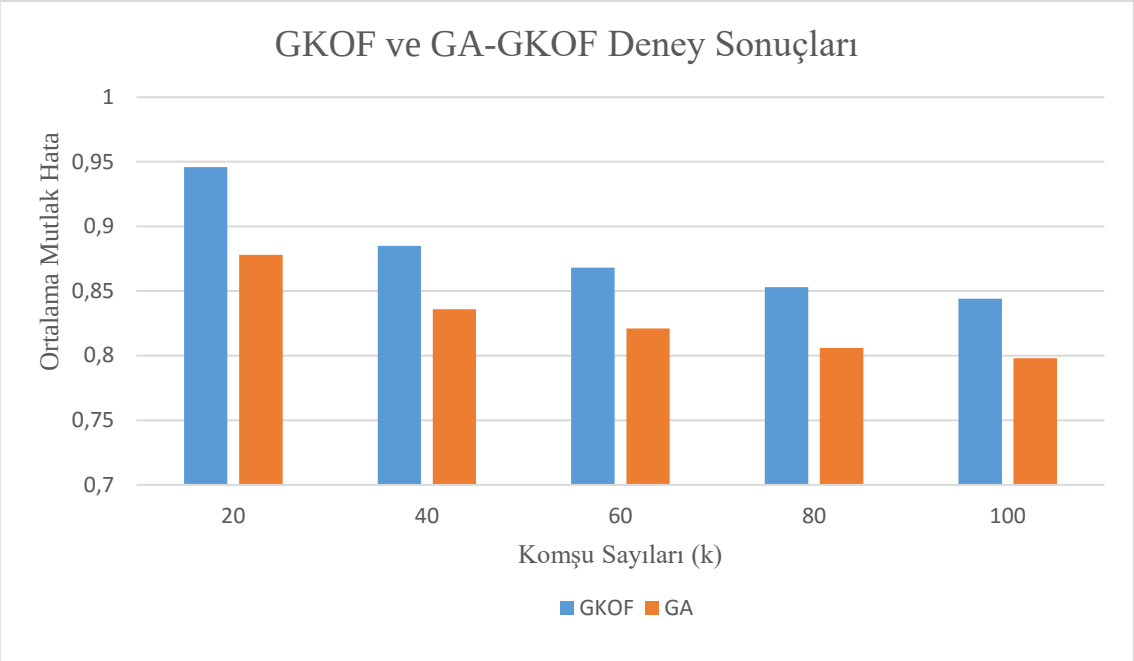
Şekil 7.10. *GA-GKOF Popülasyon Büyüklüğü Deneyi*

OF'den farklı olarak *GKOF* kapsamında ekstra olarak β_{max} değerinin seçilmesi için bir dizi deney yapılmıştır. Burada da yine diğer değişkenler sabit tutularak β_{max} farklı değerler ile test edilmiş ve sonuçlar Şekil 7.11'de gösterilmiştir. Burada yapılan deneylerde *k* komşuluk değeri *50* olarak seçilmiştir. Görüldüğü üzere *GA* tabanlı *GKOF* yöntemi ile elde edilen *OMH* değerleri geleneksel *GKOF* yöntemi ile elde edilen *OMH* değerlerinden daha düşüktür. β_{max} değeri *GA* tabanlı *GKOF* yöntemi için *20* olarak seçilmiştir. Şekil 7.11'de görüldüğü üzere en düşük *OMH* değeri β_{max} *20* olarak belirlendiği zaman alınmıştır. Bu sayede hem kullanıcıların oylamadıkları ürünlerin de gizliliği yeterince sağlanmış hem de kullanılan yöntemin doğruluğu maksimize edilmiştir.



Şekil 7.11. *GKOF ve GA-GKOF için β_{max} Deneyleri*

Son olarak tüm sistem değişkenlerinin optimum değerleri hesaplanmış ve bu değerler kullanılarak iki yöntem karşılaştırılmıştır. *GA* tabanlı *GKOF* yöntemi ile farklı *k* değerleri için elde edilen *OMH* değerleri, geleneksel *GKOF* yöntemi ile elde edilen *OMH* değerleri ile karşılaştırılarak sonuçlar Şekil 7.12'de gösterilmiştir. Bu karşılaştırma yine *5* tane oylama değeri ile saklanan test ürünleri ile yapılmıştır. Görüldüğü üzere *GA* tabanlı *GKOF* yöntemi kullanılarak elde edilen sonuçlar geleneksel *GKOF* yönteminde elde edilen sonuçlara göre daha düşük ortalama *OMH* değerleri elde etmiştir.



Şekil 7.12. *Farklı Komşu Sayılarına Göre GKOF ve GA-GKOF Karşılaştırması*

Fraklı *k* komşuluk değerleri için her iki yöntemde elde edilen deney sonuçlarının t-test sonuçları Tablo 7.2'de gösterilmiştir. Serbestlik derecesi burada da 18 olarak belirlenen t-test sonuçlarının *GA* tabanlı *GKOF* için %99 oranında anlamlı olduğu görülmektedir.

Tablo 7.2. *Geleneksel GKOF ve GA-GKOF t-test sonuçları*

k	t	p
20	14,344	0,000
40	12,080	0,000
60	10,811	0,000
80	10,806	0,000
100	10,584	0,000

8. DEĞERLENDİRME VE SONUÇ

Tez kapsamında yapılan bu çalışmada *kNN* tabanlı geleneksel öneri sistemlerinden hem ortak filtreleme hem de gizliliği korunmuş ortak filtreleme yöntemleri ayrı ayrı ele alınmış ve bu yöntemler genetik algoritma ile geliştirilerek daha iyi bir seviyeye getirilmesi amaçlanmıştır. Gizliliği korunmuş ortak filtreleme yöntemi ile kullanıcı verilerinin güvenliği sağlanırken üretilen önerilerdeki doğruluk oranlarından bir miktar taviz verilmektedir. Bu bağlamda *kNN* tabanlı geleneksel *GKOF* sistemlerinin daha doğru sonuçlar üretmesi için farklı teknikler ile desteklenmesi gerekmektedir. Bu şekilde veri güvenliği sağlanırken öneri sistemin doğruluk göstergesi de iyileştirilmiş olacaktır. Yapılan çalışma genetik algoritma ile eğitim aşamasında oluşturulan model kullanılarak *GA* tabanlı *GKOF* yönteminin doğruluğu gizlilikten taviz verilmeden artırılmıştır.

Deney sonuçlarından da görüldüğü üzere genetik algoritma her iki yöntemde de iyileştirme sağlamıştır. Ancak *GKOF* yöntemindeki iyileştirme oranı daha yüksektir. Yapılan gözlemlerde bunun sebebinin verileri gizlemek için kullanılan *RBT* tekniğinin orijinal verilere rastgele veri eklemesi olduğu anlaşılmıştır. Her ne kadar kullanıcıların toplanmış verisinde bir değişiklik olmasa da her bir oy bilgisi değiştirildiği için doğruluk bir miktar azalmaktadır. *kNN* tabanlı öneri sistemlerinde üretilecek öneri kullanıcıların birbirleri ile olan benzerlikleri ile hesaplanır. *RBT* tekniği ile kullanıcı oylamaları değiştirildiği için benzerlikler de değişecektir. Bu sebeple sistem doğruluğunun bir miktar azalması beklenen bir sonuçtur. Kullanılan yöntemde kısmen değişmiş olan bu benzerlik değerleri daha doğru sonuç üreten benzerlik değerler ile güncellendiği için sistemin doğruluğu artırılabilmiştir. Öteki taraftan *GA* tabanlı *OF* yönteminde de iyileştirme sağlanabilmiş ancak *GKOF* yönteminde elde edilen sonuçlara göre belirtilen sebeplerden dolayı iyileştirme miktarı oransal olarak daha düşük kalmıştır.

Genetik algoritma ile sistemin eğitilmesi aşaması her ne kadar uzun sürse de elde edilen en iyi benzerlik değerleri ile öneri üretilmesi işlemi, hafıza tabanlı geleneksel yöntemlere göre çok daha hızlı bir şekilde yapılabilmektedir. Sisteme ilk kez giren kullanıcıların oylamaları ve eski kullanıcıların eklediği yeni oylamaların hesaplanan en iyi benzerlik değerlerine dâhil edilmesi gerekir. Deneysel çalışmalarla elde edilen sistem parametreleri kullanılarak eğitim aşaması belirli aralıklarla tekrarlanıp üretilen en iyi benzerlikler sürekli güncel tutularak sistemin en doğru önerileri üretmesi sağlanacaktır.

Yapılan çalışmada *OF* ve *GKOF* yöntemleri ile elde edilen benzerlik değerlerinden en iyi k tane benzerlik değeri alınarak genetik algoritmaya verilmiş ve bu benzerlik değerlerinin geliştirilmesi amaçlanmıştır. Bu çalışmada sadece benzerlik değerleri iyileştirilmiş aktif kullanıcının kendisine yakın olmayan komşulukları dikkate alınmamıştır. Çalışmanın daha ileriye götürülmesi için en yakın komşulukların değil tüm komşulukların veya benzerlikleri belirli bir oranda olan komşulukların tamamı ele alınarak bunlar genetik algoritma ile iyileştirilerek sonuçlar daha da geliştirilebilir. Bu şekilde gerek benzerlikler optimum değerlere eşitlenecek gerekse aktif kullanıcıya benzer komşuluklar sadece *kNN* yönteminde elde edilen en yakın komşuluklar olmayacak, bundan daha fazlası kullanılmış olacaktır. Bu sayede optimum komşuluklar da yine genetik algoritma ile taranarak benzerlik oranları ile birlikte belirlenmiş olacaktır. Bu sayede genetik algoritmanın arama uzayı genişletilmiş olacak ve oluşturulan her bir bireyin ihtiva ettiği benzerlikler en yakın k komşuluk sınırlarının dışına çıkmış olacaktır. Dolayısıyla genetik algoritma için önemli bir parametre olan gen çeşitliliği daha da ileri götürülebilir. Bu şekilde yapılacak bir çalışmada elde edilecek sonuçlar doğruluğu belirli oranlarda daha yüksek seviyelere taşıyabilir.

Gizliliğin sağlanması üretilen önerilerin doğruluklarında bir miktar azalmaya sebep olmaktadır. Bu çalışmada *RBT* tekniği kullanılarak kullanıcı-ürün matrisinin gizliliği sağlanmıştır. Gizlilik sağlandıktan sonra elde edilen benzerlik değerlerinde bir miktar bozulma olacağından bu benzerlikler genetik algoritma ile geliştirilerek daha iyi sonuçlar alınması sağlanmıştır. Burada gizlilik sağlaması için kullanılan teknik değiştirilerek farklı metotlarla kullanıcı-ürün matrisi gizlenebilir ve buradan elde edilen benzerlikler genetik algoritma ile geliştirildikten sonra sonuçlar tekrar analiz edilerek karşılaştırılabilir.

Yapılan çalışma kullanıcı bazlı bir ortak filtreleme çalışmasıdır. Bu çalışma benzer şekilde ürün bazlı ortak filtreleme olarak da tasarlanabilir. Bu şekilde ürünler arasında hesaplanan benzerlikler genetik algoritma ile benzer şekilde geliştirilerek sistem eğitilebilir ve ardından bu benzerlikler test aşamasında kullanılarak sonuçlar elde edilebilir. Bu şekilde yapılacak deneysel çalışmalar kullanıcı bazlı ortak filtreleme yöntemi ile elde edilen sonuçlarla birlikte ele alınabilir ve her iki yöntemin gerek *OF* gerekse *GKOF* teknikleri kullanıldığında elde ettikleri test sonuçlarının karşılaştırmalı analizleri yapılabilir.

KAYNAKÇA

[1] Wang, Z., Yu, X., Feng, N., and Wang Z. (2014). An improved collaborative movie recommendation system using computational intelligence. *Journal of Visual Languages & Computing*, 25(6), 667-675.

[2] Vijayasarathy, L. R. (2004). Predicting Consumer Intentions to Use Online Shopping: The Case for an Augmented Technology Acceptance Model. *Information and Management*, 41(6), 747–762.

[3] Mahmood, T. and Ricci, F. (2009). Improving recommender systems with adaptive conversational strategies. *Proceedings of the 20th ACM Conference on Hypertext and Hypermedia*, New York, USA: ACM, 73-82

[4] Burke, R. (2007). Hybrid Web recommender systems. *The adaptive web*. Springer Berlin Heidelberg. 4321(1), 377-408.

[5] Resnick, P. and Varian, H. R. (1997). Recommender Systems. *Communications of the ACM*, 40(3), 56-58.

[6] Rana C. and Jain S. (2014). An evolutionary clustering algorithm based on temporal features for dynamic recommender systems. *Swarm and Evolutionary Computation*, 14, 21-30.

[7] Ackerman, M. S., Cranor, L. F., and Reagle, J. (1999). Privacy in e-commerce: Examining user scenarios and privacy preferences. *Proceedings of the 1st ACM Conference on Electronic Commerce (EC'99)*, MI, USA, 1-8.

[8] Bilge, A. and Polat, H., (2013). A scalable privacy-preserving recommendation scheme via bisecting k-means clustering. *Information Processing and Management*, 49(4) 912– 927.

[9] Ameur, E., Brassard, G., Fernandez, J. M., and Mani Onana, F. S. (2008). Alambic: A privacy-preserving recommender system for electronic commerce. *International Journal of Information Security*, 7(5), 307–334.

[10] M. K. Reiter, A. D. Rubin. (1998). Crowds: anonymity for web transaction. *ACM Transactions on Information and System Security*, 1(1), 66–92.

[11] P. F. Syverson, D. M. Goldschlag, and M. G. Reed. (1997). Anonymous connections and onion routing. *In Proceedings of 1997 IEEE Symposium on Security and Privacy*, Oakland, California, USA, May 5-7.

[12] Polat, H., and Du, W. (2005). Privacy-preserving collaborative filtering. *International Journal of Electronic Commerce*, 9(4), 9–35.

[13] Batmaz, Z. and Polat, H. (2016). Randomization-based Privacy-preserving Frameworks for Collaborative Filtering. *Procedia Computer Science*, 96, 33-42.

[14] Goldberg, D. E. (1989). *Genetic Algorithms in Search, Optimization, and Machine Learning*. Boston: Addison-Wesley Publishing Company.

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

[16] Ar, Y. and Bostanci, E. (2016). A genetic algorithm solution to the collaborative filtering problem. *Expert Systems With Applications*, 61, 122-128.

[17] Kim, k. and Ahn, H. (2008). A recommender system using GA K-means clustering in an online shopping market. *Expert Systems With Applications*, 34(2), 1200-1209.

[18] Michalewicz, Z. (1996). Genetic algorithms + Data structures = Evolution programs. *Artificial intelligence*. Heidelberg: Springer.

[19] Dao, T. H., Jeong, S.R. and Ahn, H. (2012). A novel recommendation model of location-based advertising: Context-Aware Collaborative Filtering using GA approach. *Expert Systems With Applications*, 39(3), 3731-3739.

[20] Bobadilla, J., Ortega, F. and Alcala, J. (2011). Improving collaborative filtering recommender system results and performance using genetic algorithms. *Knowledge-Based Systems*, 24(8), 1310-1316.

[21] Canny, J. (2002a). Collaborative filtering with privacy. *In Proceedings of the 2002 IEEE symposium on security and privacy*, Berkeley, CA, USA, 45–57.

[22] Canny, J. (2002b). Collaborative filtering with privacy via factor analysis. *In Proceedings of the 25th annual international ACM SIGIR conference on research and development in information retrieval*. Tampere, Finland, 238-245

[23] Bilge, A. and Polat, H. (2012b). An improved privacy-preserving DWT-based collaborative filtering scheme. *Expert Systems with Applications*, 39(3), 3841–3854.

[24] Dakhel, G. M. And Mahdavi, M. (2011). A New Collaborative Filtering Algorithm Using K-means Clustering and Neighbors' Voting. *11th International Conference on Hybrid Intelligent Systems*, Melacca, Malaysia Dec. 5-8

[25] Wang, J., P. de Vries, A. and Reinders, M. J. T. (2006). Unifying user-based and item-based collaborative filtering approaches by similarity fusion. *In Proceedings of the 29th annual international ACM SIGIR conference on Research and development in information retrieval (SIGIR '06)*, New York, NY, USA, 501-508.

[26] Choi, S., Ko, S., Han, Y. (2012) A movie recommendation algorithm based on genre correlations, *Expert System with Applications*, 39(9), 8079-8085.

[27] Yadav, S., Vikesh, Shreyam, Nagpal, S. (2018). An Improved Collaborative Filtering Based Recommender System using Bat Algorithm. *International Conference on Computational Intelligence and Data Science (ICCIDS 2018)*, Dwarka, Delhi, India 1795-1803.

[28] Miller, B. N., Konstan, J. A. and Riedl, J. (2004). PocketLens: toward a personal recommender system, *ACM Transactions on Information Systems*, 22(3), 437–476.

[29] Bobadilla, J., Ortega, F., Hernando, A. and Gutiérrez, A. (2013). Recommender systems survey. *Knowledge-Based Systems*, 46, 109–132.

[30] Shardanand, U. (1994). Social information filtering for music recommendation. Doctoral dissertation, *Massachusetts Institute of Technology*.

[31] Sarwar, B., Karypis, G., Konstan, J. and Reidl, J. (2001). Item-based collaborative filtering recommendation algorithms. *In Proceedings of the 10th International Conference on World Wide Web (WWW'01).* ACM, New York, NY, 285–295.

[32] Resnick, P., Iacovou, N., Suchak, M., Bergstrom, P. and Riedl, J. (1994). Grouplens: an open architecture for collaborative filtering of netnews. *In Proceedings of the ACM Conference on Computer Supported Cooperative Work (CSCW'94)*, New York, NY, 175–186.

[33] Herlocker, J. L., Konstan, J. A., Terveen, L. G., and Riedl, J. T. (2004). Evaluating collaborative filtering recommender systems. *ACM Transactions on Information Systems*, 22(1), 5–53.

[34] Souvik, D., Niloy G. and Pabitra, M. (2008) Feature weighting in content based recommendation system using social network analysis. *In Proc. of the 17th International Conference on World Wide Web*, 1041–1042.

[35] Cacheda, F., Carneiro, V., Fernández, D. and Formoso, V. (2011). Comparison of collaborative filtering algorithms: Limitations of current techniques and proposals for scalable, high-performance recommender systems. *ACM Transactions on the Web*, 5(1), 1-33.

[36] Candillier, L., Meyer, F., Fessant, F. and Jack, K. (2009). State-of-the-art recommender systems. *Collaborative and Social Information Retrieval and Access-Techniques for Improved User Modeling*, 1-22.

[37] Breese, J., Heckerman, D. and Kadie, C.(1998). Empirical analysis of predictive algorithms for collaborative filtering. *Proceedings of the 14th Conference on Uncertainty in Artificial Intelligence*, San Francisco, CA, USA: Morgan Kaufmann Publishers Inc., 43-52.

[38] Polat, H. And Du, W. (2003). Privacy-Preserving Collaborative Filtering using Randomized Perturbation Techniques. *Third IEEE International Conference on Data Mining*. Melbourne, FL, USA, Nov 22-22

[39] Park, D.H., Kim, H.K., Choi, Y. and Kim, J.K. (2012). A literature review and classification of recommender systems research. *Expert Systems with Applications*, 39(11), 10059–10072.

[40] J. Herlocker, J. Konstan, A. Borchers, and J. Riedl. (1999). An algorithmic framework for performing collaborative filtering. *Proceedings of the 22nd annual international ACM SIGIR conference on Research and development in information retrieval*, 230-237.

[41] Balabanovic, M., and Shoham, Y. (1997). Combining content-based and collaborative recommendation. *Communications of the ACM*, 40, 66–72.

[42] Kaleli, C., and Polat, H. (1997). Privacy-preserving SOM-based recommendations on horizontally distributed data. *Knowledge-Based Systems*, 33, 124-135.

[43] Yargıç, A., and Bilge, A. (1997). Privacy-preserving multi-criteria collaborative filtering. *Information Processing & Management*, 56(3), 994-1009.

ÖZGEÇMİŞ

Adı Soyadı : Mustafa Kemal BİRGİN

Yabancı Dil : İngilizce

Doğum Yeri ve Yılı: Malatya / 1985

E-Posta : mkbirgin@anadolu.edu.tr

Eğitim ve Mesleki Geçmişi:

- 2009, Karadeniz Teknik Üniversitesi, Mühendislik Fakültesi, Bilgisayar Mühendisliği Bölümü