İŞBİRLİKÇİ FİLTRELEME YÖNTEMİ KULLANILARAK KULLANICI TABANLI FİLM ÖNERİ SİSTEMİ

Bilgisayar Mühendisliği Bölümü

Kocaeli Üniversitesi

160201023
Erva Nur SEVİM
ervasevim@gmail.com

160201055
Abdullah GEZEK
abdullahgezekk@gmail.com

170201087
Fatma Sıla SEÇGİN
silasecgin@gmail.com

160201070
Abdullah Emre ONUR
eonur98@gmail.com

150201115
Fulya EKER
fulyaeker0@gmail.com

130201020
Yasemin ELVAN
yaseminelvan355@gmail.com

Özet

Çalışmamızda bellek tabanlı işbirlikçi filtreleme yöntemini kullanarak kullanıcıların verdiği oylara yönelik benzer zevklere sahip olan kullanıcıları bulup beğenebileceği filmler önermeye çalıştık. İşbirlikçi filtreleme yöntemleri, geçmiş kullanım verilerine göre bir kullanıcının tercihlerini belirlemeye çalışır. Bu projede, işbirlikçi problemleri çözmek için esnek bir sinir ağı mimarisi gözden geçirilmiş ve daha da geliştirilmiştir. Modelimiz iyi bir tahmin yapmak için en başarılı sonucu vereceği ağırlık parametresinin hesabını yapmaya çalışmaktadır. Sonuçları test etmek için film izleme uygulaması olan Netflix'in kullanıcılara en iyi filmi öneren algoritma yarışması için yayınladığı Netflix prize veri setini[6] kullandık. Sonuçlarımızın başarı oranı gerçekleştirilen testler bölümünde değerlendirilmiştir.

Abstract

In our study, we tried to suggest movies that users might like according to their votes by using memory-based collaborative filtering method. Collaborative filtering methods try to determine a user's preferences based on historical usage data. In this project, a flexible

neural network architecture was reviewed and developed to solve collaborative problems. Our model tries to calculate the weight parameter that will give the most successful result to make a good estimate. To test the results, we used the Netflix data set published by Netflix, which is a movie watching application, for the algorithm contest that suggests the best movie to users. The success rate of results is evaluated in the tests section.

1. Giriş

İnternet'in yaygınlaşması, veri depolama birimlerinin kapasitelerinin artması ve ucuzlaması her geçen gün elektronik ortamda tutulan verilerin artmasına yol açmıştır. Bu çoğu yönden iyi bir şey olarak görülse de verilerin nasıl yorumlanacağı ve bilgiye nasıl ulaşılacağı önemli bir sorun haline gelmiştir. Bu noktada bazı kolaylaştırıcı yöntemler üretilmeye başlanmıştır. Şuan girdiğimiz çoğu sitede karşımıza çıkan öneri sistemlerinin çok iyi çalıştığının hepimiz farkındayız. Öneri sistemleri kullanıcılardan toplanan verileri işleyip anlamlı bilgiler elde etmemizi sağlayan ve bu bilgilerin yorumlanacak düzeye gelmesini sağlayan araçlar olmuşlardır. Dolayısı ile öneri sistemleri aşırı veri ile başa çıkmak için geliştirilen bilgisayar tabanlı zeki tekniklerdir.

Günümüzde özellikle e-ticaret alanında gördüğümüz bu öneri sistemlerini çoğu sistem kullanmaktadır. Şuan film izlediğimiz netflix, amazon gibi sitelerde, müzik dinlediğimiz spotify gibi günlük hayatımızda sıklıkla kullandığımız uygulamalarda bu sistemler kullanılmakta ve oldukça yüksek başarı sonucu vermektedir. Kullanıcının girdiği her sitedeki ürün yelpazesini incelemesi ve kendine uygun olanını bulması bu kadar çok veri içerisinde mümkün değildir. Öneri sistemleri kullanıcıların içerik öğeleri (film, müzik kitap, vb.) hakkındaki geçmişteki ilgi bilgilerini kullanarak daha önce hiç karşılaşmadıkları bir içerik öğesine olan ilgisini hesaplamaya çalışırlar. Yani öneri sistemlerinin amacı kullanıcıların ilgi duyabileceği ürünleri kullanıcıya önermektir. Öneri sistemleri girdi olarak aldıkları bilgiye göre değişik sınıflara ayrılmaktadır. Bu sistemlerden en başarılı olanları ve en tanınmış olanları içerik tabanlı filtreleme ve işbirlikçi filtrelemedir.

İşbirlikçi Filtreleme Yöntemleri öneri sistemlerinde en yaygın olarak kullanılan yöntemlerden bir tanesidir. Genellikle kullanıcılar arasındaki benzerlikleri dikkate alarak işlem yapmaktadır. Yani kullanıcıların ilgisini çekebilecek ürünler hakkında bilgi edinmeye çalışırken benzer zevke sahip başka kullanıcıların geçmişte bu ürünlere vermiş oldukları puanları dikkate alarak işlem yapmaktadır. İşbirlikçi filtreleme yaklaşımının altında yatan varsayım,

kullanıcılarının ilgi alanlarına göre, sonraki hareketlerini tahmin ederek uygulama kullanımını sürdürmek amacıyla düzenlenen algoritma yapılarıdır.

İşbirlikçi filtreleme yöntemi bellek tabanlı işbirlikçi filtreleme ve model tabanlı işbirlikçi filtreleme olmak üzere iki ana kısma ayrılmaktadır [2].

Bellek tabanlı işbirlikçi filtreleme yönteminin diğer adı kullanıcı tabanlı işbirlikçi filtrelemedir. Bu yöntemde algoritmalar, kullanıcıların tercihlerini göre kullanıcılar arasındaki ilişkiyi hesaplamaya çalışırlar. Korelasyon değeri öneri yapılırken farklı kullanıcı grupları arasında dinamik olarak hesaplanmaktadır. Gerçek zamanlı öneri sistemlerinde bu dinamik hesaplama büyük bir bellek ve işlem yüküne neden olmaktadır. Bu nedenle, bu tarz öneri sistemleri büyük veri kümeleri ve gerçek zamanlı uygulamalar için uygun değildir. Temel olarak bellek tabanlı işbirlikçi filtreleme algoritmaları öneri isteğinde bulunan aktif kullanıcı ile veritabanındaki diğer kullanıcıların benzerliklerini bulmaya çalışmaktadır. Algoritmaların bellek tabanlı olarak tanımlanmasının nedeni algoritmaların kullanıcı veritabanının bütününde işlem yapmasıdır. Bütün kullanıcıların tercihleri, kullanıcıların ürünlere vermiş olduğu oylarla simgelenmektedir. Aktif kullanıcının oy değeri oyladığı ürünlerin oy değerinin ortalama değeri olmaktadır. Aktif kullanıcının tahmin edilen oyları diğer kullanıcı oylarının ağırlıklı toplamları eklenerek hesaplanmaktadır. Ağırlıklar aktif kullanıcı ile diğer kullanıcıların benzerlikleri tarafından belirlenmektedir bu nedenle kullanıcıların benzerlikleri arttıkça ağırlıklı toplama katkısı da artmaktadır [3].

Model tabanlı işbirlikçi filtreleme diğer bir adıyla ürün tabanlı işbirlikçi filtreleme büyük veri kümelerinde ölçeklenebilir olması nedeniyle güncel uygulamalarda yaygın olarak kullanılmaktadır. Model tabanlı algoritmalar kullanıcılar arasındaki benzerlikleri hesaplamak yerine tercih edilmiş ürünler arasındaki benzerlikleri hesaplamaktadır. Temel olarak model tabanlı işbirlikçi filtreleme algoritmaları öncelikle kullanıcıların tercihlerini derleyerek tanımlayıcı bir model oluşturup ardından modele uygun olarak öneri üretmektedirler.

Biz çalışmamızda bellek tabanlı işbirlikçi filtreleme yöntemini kullanarak kullanıcıların verdiği oylara yönelik benzer zevklere sahip olan kullanıcıları bulup beğenebileceği filmler önermeye çalıştık.

Çalışma sonucunda, kullanıcılara en kısa sürede en fazla ilgilerini çekebilecek filmleri bulmaları konusunda yardımcı olunacaktır.

Geliştirilmiş olan sistemde kullanılmış olan algoritmalar veri setimiz üzerinde test

edilmiştir. Modelin performansı ve başarı oranına sonuçlar kısmında değinilecektir.

2. İlgili Çalışmalar

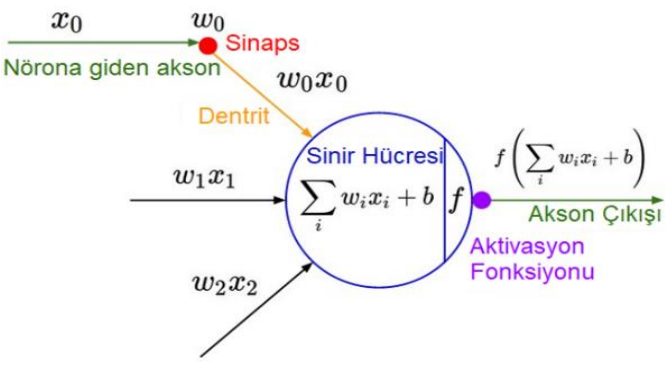
Öneri sistemleri, kullanıcıların daha önceki davranışlarını ve alışkanlıklarını inceleyerek, bir kullanıcıya uygun ürünler önermeyi hedefleyen uygulamalardır. Internet ortamında, bu ürünler, elektronik ticaret sitelerinde yer alan ürünler olabileceği gibi bir Web sitesindeki sayfalar da olabilir. Bu sistemler, kullanıcıların beğenilerine uygun ürünler önermek, kullanıcıya bir Web sitesinde yol göstermek, Web sitesinin yapısını iyileştirmek, kişiselleştirme veya bir Web sitesinin performansını artırma gibi çeşitli amaçlar için kullanılabilir. Literatürde öneri sistemleri için kullanılan çeşitli yöntemler mevcuttur. En sık uygulanan yöntemler olarak, işbirlikçi filtreleme, ilişkilendirme kuralları, Markov modelleri, demetleme yöntemleri, sıralı dizilerin incelenmesi denilebilir. Öneri sistemleri için ilk uygulanan ve bu nedenle de çok yaygın olarak kullanılan yöntemlerden biri işbirlikçi filtreleme yöntemidir.[9]

Geçmiş yıllarda, Model Tabanlı İşbirlikçi Filtreleme Algoritmaları için model oluşturmak için daha çok Bayes Ağları, Kümeleme Yöntemleri ve Kural Tabanlı Yaklaşımlar kullanılmıştır. Günümüzde ise daha çok Matris faktörizasyonuna dayalı yaklaşımlar kullanılmaktadır. Model Tabanlı İşbirlikçi Filtreleme Yöntemi alanında yapılan çalışmalara örnek olarak 1998 yılında Breese ve arkadaşları tarafından yapılan çalışma gösterilebilir.

Bellek Tabanlı İşbirlikçi Filtreleme yöntemlerinin ilk çalışmaları 1994 yılında Resnick tarafından yapılmıştır. Yapılan çalışmada kullanıcılara haber önerebilmek için Pearson Bağıntısı'nı kullanan kullanıcı tabanlı bir öneri sistemi geliştirilmiştir.

3. Mimari

Basit bir sinir ağının yağısı aşağıdaki şekilde gösterildiği gibidir:



Şekil 1: Sinir Ağının Yapısı

Yapay sinir ağının en küçük parçası olarak bilinen perceptron, aşağıdaki gibi lineer bir fonksiyonla ifade edilmektedir ve ilk defa

1957 yılında Frank Rosenblatt tarafından tanımlanmıştır.

$$f(x_1, x_2) = w_1 x_1 + w_2 x_2$$

f(x): x'in değerine bağlı olduğundan bağımlı değişkendir. Girdiye ait skoru verir.

x: bağımsız değişken, girdi.

W: ağırlık parametresi

Derin Öğrenme modelinde yapılan temel işlem; modelin en iyi skoru vereceği W parametresinin hesabını yapmaktır.

Bir katman içindeki nöronların birbirleriyle ilişkileri yoktur. Her bir nöronun işi sistemde akılda kalan bilgiyi sonraki katmana ya da çıkışa aktarmaktır. Arka arkaya iki katmandaki nöronlar birbirlerini çeşitli aktivasyon değerleriyle etkilemekte ve modelin öğrenme seviyesini belirleyen bir aktarım gerçekleştirmektedir. Bu nedenle 'Katman sayısı arttıkça öğrenme performansı artar', denebilir gibi görünse de bu doğru değildir. Çünkü model performansı, sadece girdiler ve katman sayısıyla ilgisi vardır ama bununla belirlenmez. Bir çok farklı hiperparametrenin etkisi çıkış performansını etkilemektedir. Bu hiperparametrelerden biri aktivasyon eğitim kısmında bahsettiğimiz aktivasyon fonksiyonudur.

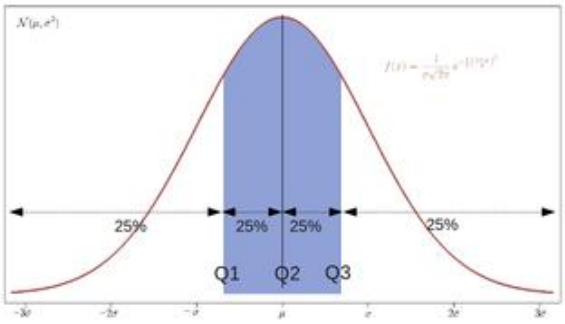
Modelimizde basit bir giriş çıkış katmanı olan perceptron kullandık. Modelimizde gizli katman kullanılmamıştır. Temel olarak basit bir yapay sinir ağında x girdiler, w ağırlıklar olarak tanımlanır ve ağın çıkışına aktarılan değere f(x) yani aktivasyon işlemi uygularız. Daha sonra bunu , nihai çıktı olarak alırız.

Modelimizi eğitmeye başlamadan önce veri setimizdeki düzensizlikleri gidermek ve işlemeye uygun hale getirmek için bazı ön işlemlerden geçirdik. Ön İşleme kısmında bu konuya değinilmiştir.

3.1. Ön İşleme

Gerçek zamanlı uygulamalardan toplanan veriler tamamlanmamış, tutarsız veya eksik olabilmektedir. Modelin başarı performansını etkileyen önemli kriterlerden biri verinin düzenlenmiş şekilde gelmesidir. Veriyi düzenleyip kullanacağımız şekline uygun haline getirmek için bazı ön işleme tekniklerine başvuruyoruz. Veri ön işleme adımları için kullandığımız kütüphaneler numpy, pandas ve seaborn. Burada numpy ve pandası sayısal işlemler ve diziler üzerindeki işlemler için kullanıyoruz. Seaborn kütüphanesi ise veriyi görselleştirebilmemize olanak sağlıyor. Veriyi işleyebilmek için

öncelikle csv dosyasını pandas kütüphanesinin read_csv fonksiyonu ile okuduk. Daha sonra toplam kullanıcı sayısı, toplam oy sayısı ve toplam film sayılarını dosyadan okuyarak parametrelerimize atadık. Daha sonra aykırı verilerin (outlier) temizlenmesi, gürültü verilerinin düzeltilmesi, tutarsızlıkların giderilmesi, eksik değerlerin doldurulması için verileri temizleme aşamasına geçtik. Bu aşamada öncelikle null olan değerleri bulmak için pandas kütüphanesinden dataframe fonksiyonu kullandık. Bu sayede oy olan satırlara false, oy gözükmeyen yani null olan satırlara true değeri atanmış olduk. Bu sayede null olan oyları ayrıştırmış olduk. Daha sonra oluşturduğumuz movie_np dizisinde hangi satırın hangi filme ait olduğunu tuttuk. Bu sayede hangi indexte hangi film id'sine ait oyun olduğunu da bir dizide tutmuş olduk. Filme verilen ortalama oyları ve ortalamanın altında kalan oyları da hesapladık. Burada istisna rastlanan verileri elemek için quantile fonksiyonunu kullandık. Quantile istatistiklerde ve olasılık miktarlarında, bir olasılık dağılımının aralığını eşit olasılıklarla sürekli aralıklara bölen veya bir numunedeki gözlemleri aynı şekilde bölen kesme noktalarıdır[1]. Aşağıda örnek olarak bir Quantile grafiğini gösterilmektedir. Bu grafikte normal olasılık yoğunluk dağılımındaki çeyrekler gösterilmektedir.



Şekil 2: Quantile [8]

Quantile fonksiyonu ile en yoğun olan %80'lik kısmın altında kalan oyları silerek nadir olan verileri işleme almamış olduk. Daha sonra verileri 48000 eğitim verisi ve 1200 test verisi olacak şekilde böldük ve ön işleme işlemlerimizi bu şekilde bitirmiş olduk.

3.2. Eğitim

Bildiğimiz üzere nöronlar aslında ağırlıkların ve girdilerin doğrusal kombinasyonu olan birer matematik fonksiyonlardır. Her girdiye karşılık gelen ağırlıklar ile çarpıp toplanır. Matematiksel ifadesini şu şekilde gösterebiliriz;

$$f(x_1, x_2) = w_1 x_1 + w_2 x_2$$

Nöronlar giriş olarak birkaç sayı alır. Her giriş için nöron bir ağırlık değeri atar. Atanmış bu sayıları içeren vektöre ağırlık vektörü denir. Her nöronu benzersiz kılan şey sahip olduğu bu ağırlıklardır. Test sırasında belirlenen bu

ağırlık değerleri ile girdilerin çıktısı oluşturulur. Ağırlık değerlerinin belirlendiği kısım ise burada anlatacağımız eğitim aşamasıdır. Bu aşamada ağımızı ayarlamak için değiştireceğimiz rakamların nihai ve optimum sonucu, test aşamasında kullanacağımız değerleri oluşturacaktır. Kısaca eğitim aşaması girdilerin çıktılarla iyi bir eşleşmesini oluşturmak için, eğitim veri setinin kullanarak ağırlıkların güncellenmesi işlemidir.

Sinir ağımıza ilk başladığımızda ağırlıklarımızı rastgele başlatırız. Beklediğimiz üzere bu bize güzel bir sonuç vermeyecektir. Amacımız eğitim setimizle kötü performans gösteren bir sinir ağı başlatıp, sonuçlara göre ağırlık değerlerini güncelleyip ağın performansını arttırdıktan sonra optimum değerler ile nihai ağı oluşturmak olduğu için baştaki sonucun kötülüğü ile şuan ilgilenmiyoruz. Rastgele değerleri Neural Network classındaki init fonksiyonunda atadık. Bunun için numpy kütüphanesinin random değerler oluşturma fonksiyonunu kullandık. Daha sonra üretilen bu değerleri statik synaptic_weights parametremize atadık. Eğitim aşamasının gerçekleşeceği train fonksiyonunda parametre olarak eğitim veri setimizdeki girdi ve çıktı değerlerini ve iterasyon sayısını aldık. Parametre olarak gönderilen iterasyon sayısı eğitim aşamamızdaki adım sayısını belirliyor. For döngüsü ile iterasyon sayısı kadar eğitim işleminin tekrarlanmasını sağladık. Döngünün ilk adımında think fonksiyonundan çıktı değerimizi alıyoruz. Think fonksiyonunda ağın çıktısına aktarılan değere aktivasyon işleminini uyguluyoruz.

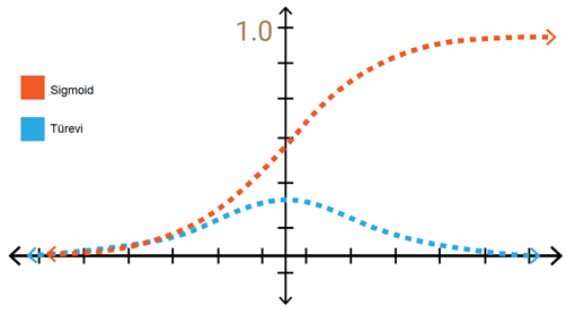
Aktivasyon fonksiyonları sinir ağlarındaki yeri oldukça önemlidir ve performanstaki etkileri büyüktür. Yapay sinir ağlarına doğrusal olmayan gerçek dünya özelliklerini tanıtmak için aktivasyon fonksiyonuna ihtiyaç duyuyoruz. Temel olarak basit bir yapay sinir ağında x girdiler, w ağırlıklar olarak tanımlanır ve ağın çıkışına aktarılan değere f(x) yani aktivasyon işlemi uygularız. Eğer aktivasyon fonksiyonu uygulanmazsa çıkış sinyali basit bir doğrusal fonksiyon olur. Doğrusal fonksiyonlar yalnızca tek dereceli polinomlardır. Aktivasyon fonksiyonu kullanılmayan bir sinir ağı sınırlı öğrenme gücüne sahip bir doğrusal bağlanım (linear regression) gibi davranacaktır. Ama biz sinir ağımızın doğrusal olmayan durumları da öğrenmesini istiyoruz. Çünkü sinir ağımıza öğrenmesi için karmaşık gerçek dünya bilgileri vereceğiz. Çok katmanlı derin sinir ağları bu sayede verilerden anlamlı özellikleri öğrenebilir. Ağırlıklar ile ilgili hata değerlerini hesaplamak için yapay sinir ağında hatanın geriye yayılımı algoritması uygulanmaktadır. Optimizasyon stratejisini belirlemek ve hata oranını minimize etmek gerekmektedir. Uygun

optimizasyon algoritmasını seçmek de ayrı bir konudur. Çünkü her bir aktivasyon fonksiyonunun kendine özgü avantaj ve dezavantajları vardır. Seçilebilecek aktivasyon fonksiyonları ve formülleri örnek olarak şöyle gösterilebilir.

AKTİVASYON FONKSİYON	DENKLEM	ARALIK
Doğrusal Fonksiyon	$f(x) = x$	$(-\infty, \infty)$
Basamak Fonksiyonu	$f(x) = \begin{cases} 0 \; için \; x < 0 \\ 1 \; için \; x \geq 0 \end{cases}$	$\{0, 1\}$
Sigmoid Fonksiyon	$f(x) = \sigma(x) = \frac{1}{1 + e^{-x}}$	$(0, 1)$
Hiperbolik Tanjant Fonksiyonu	$f(x) = \tanh(x) = \frac{(e^{x} - e^{-x})}{(e^{x} + e^{-x})}$	$(-1, 1)$
ReLU	$f(x) = \begin{cases} 0 \; için \; x < 0 \\ x \; için \; x \geq 0 \end{cases}$	$[0, \infty)$
Leaky (Sızıntı) ReLU	$f(x) = \begin{cases} 0.01 \; için \; x < 0 \\ x \; için \; x \geq 0 \end{cases}$	$(-\infty, \infty)$
Swish Fonksiyonu	$f(x) = 2x\sigma(\beta x) = \begin{cases} \beta = 0 \; için \; f(x) = x \\ \beta \to \infty \; için \; f(x) = 2\max(0, x) \end{cases}$	$(-\infty, \infty)$

Şekil 3: Aktivasyon Fonksiyonlarının Matematiksel İfadeleri [4]

Biz projemizde sigmoid aktivasyon fonksiyonunu seçtik. Grafiği şu şekildedir;



Şekil 4: Sigmoid Fonksiyonu ve türevi [4]

Özellikleri ise şu şekilde sıralanabilir;

- Sigmoid aktivasyon fonksiyonun kullanılmasının en önemli sebebi değeri 0 ve 1 arasına sıkıştırmasıdır. Bu yüzden bir olayın olma olasılığını bulan modellerde kullanılır. Çünkü olasılık fonksiyonu 0 ile 1 arasında değer alır.

- Sigmoid fonksiyonun türevi vardır. Sigmoid fonksiyonu monoton fonksiyondur fakat türevi monoton değildir.

- Sigmoid fonksiyonu, eğitim esnasında modelin takılıp kalmasına sebep olabilir. Bu nedenle karmaşık makine öğrenimi modellerinde kullanımı önerilmez.

- Softmax aktivasyon fonksiyonu verileri üç veya daha fazla sayıda sınıfa ayırmak için kullanılır. Sigmoid ise sadece iki sınıfa ayırır.

Projemizin tanımına uygun özellikler olduğu için sigmoid fonksiyonunu seçtik. Çünkü bizim de çıkarmak istediğimiz sonuç verileri iki sınıfa (öner, önerme) ayırmaktı.

Think fonksiyonundan çıktı değerini aldıktan sonra döngünün ikinci adımında orijinal çıktıdan üretilen çıktımızı çıkarıp hata değerimizi bulduk [5]. Daha sonra eğitim girdileri ile sigmoid türevini çarparak ağırlık matrisi ile topladık. Bu şekilde ağırlık değerimiz güncelledik. Bu şekilde iterasyon

adımı kadar döngü tekrar ederek ve ağırlık değerlenin en optimum değerlerini oluşturduk.

4. Veri Kümesi

Kullandığımız veri kümesi film izleme uygulaması olan Netflix, filmler için kullanıcı derecelendirmelerini tahmin etmek üzerine yaptığı yarışmada kullanılması için yayınlamıştır. Veri seti, 2019 itibariyle Netflix'te mevcut olan filmlerden oluşmaktadır. Veri setinde her bir kullanıcının id'si, oy verdiği filmlerin id'si, 1 ile 5 arasında verilmiş olan oy ve YYYY-MM-DD formatında oyun verildiği tarih bulunmaktadır. Veri seti, 17 770 filmden, 480 189 kullanıcının oy verdiği toplam 100 480 507 oyu içerir. Biz projemizde kısıtlamaya giderek, 60 filmden, 176710 kullanıcının oy verdiği toplam 280107 oyu baz alarak işlemlerimizi gerçekleştirdik. Verilen oylar 1-5 aralığındadır. Daha doğru sonuç alabilmek adına veriler, veri temizleme işleminden geçirilerek kalan veriler üzerinden eğitim aşaması tamamlanmıştır. Eğitilen veriler üzerinden öneri sonuçları elde edilmiştir.

5. Gerçekleştirilen Testler

Modelimizin test verileri üzerindeki performansını hata matrisi kullanarak hesapladık. Makine öğrenmesinde kullanılan sınıflandırma modellerinin performansını değerlendirmek için hedef niteliğe ait tahminlerin ve gerçek değerlerin karşılaştırıldığı hata matrisi sıklıkla kullanılmaktadır. . Hata matrisi tahmin edilen ve gerçek değerlerle 4 farklı kombinasyonlu bir tablodur. Recall (Geri Çağırma), Precision (Hassasiyet) , Specificity (Özgünlük) , Accuracy (Doğruluk) ölçmek için kullanışlıdır. Sınıflandırma tahminleri şu dört değerlendirmeden birine sahip olacaktır:

1. True Pozitive (TP): Olumlu tahmin ettiniz ve bu doğru
2. True Negative (TN): Olumsuz tahmin ettiniz ve bu doğru.
3. False Positive (FP): Olumlu tahmin ettiniz ve bu yanlış.
4. False Negative(FN): Olumsuz tahmin ettiniz ve bu yanlış.

Modelimizin test verileri üzerindeki değerleri şu şekildedir.

true_positive = 6825
true_negative = 482
false_positive = 2987
false_negative = 1706

Bu değerleri hata matrisi üzerinde gösterilmiş hali tablo 1'de gösterilmiştir.

	POSITIVE	NEGATIVE
POSITIVE	6825	2987
NEGATIVE	1706	482

Tablo 1: Hata Matrisi

Bu tablo üzerinden accuracy (doğruluk), errror rate (hata oranı), recall (geri çağırma), precision (hassasiyet) ve F-measure yani F1 skoru parametrelerini açıklayıp hesaplayalım.

Doğruluk (Accuracy) değeri modelde doğru tahmin ettiğimiz alanların toplam veri kümesine oranı ile hesaplanmaktır. Formülü şu şekildedir;

$$\frac{TP + TN}{TOPLAM}$$

Modelimizin doğruluk oranı 0,61'dir.

Hata Oranı (Error Rate) değeri yanlış tahminlerin oranıdır. Formülü şu şekildedir;

$$\frac{FN + FP}{TOPLAM}$$

Modelimizin hata oranı 0,39'dir.

Recall (Geri Çağırma): Tüm pozitif değerlerden, ne kadar doğru tahmin ettiğimizdir. Formülü şu şekildedir;

$$\frac{TP}{TP + FN}$$

Modelimizin recall değeri 0,8'dir.

Precision (Hassasiyet) tüm değerlerden, ne kadar doğru tahmin ettiğimizdir. Formülü şu şekildedir;

$$\frac{TP}{TP + FP}$$

Modelimizin hassasiyet değeri 0,7'dir.

F1 Score (F Measure) hassasiyet ve geri çağırmayı aynı anda ölçmeye yardım eder. Doğruluk yerine F1 Score değerinin kullanılmasının en temel sebebi eşit dağılmayan veri kümelerinde hatalı bir model seçimi yapmamaktır. Formülü şu şekildedir.

$$\mathbf{2} * \frac{\text{Hassasiyet} * \text{Geri Çağırma}}{\text{Hassasiyet} + \text{Geri Çağırma}}$$

Modelimizin f1 skoru 0,6'dır.

6. Sonuçlar

Projemizde, kullanıcı tabanlı işbirlikçi filtreleme kullanarak film öneri sistemi yapılmış, tahmin verileriyle çıkan sonuçlar karşılaştırılmıştır. Verisetimiz, belirli işlemlerden geçirilerek veriler eğitilmiş, tahmin sonuçları ile çıkan sonuçlar karşılaştırılıp doğru tahmin oranı belirlenmiştir. Yapay sinir ağlarının yapısı öğrenilip, eğitim ve test aşamalarının nasıl gerçekleştiğine dair kazanımlar elde edilmiştir. Öneri sistemlerinin araştırılması ile de makine

öğrenmesi alanında öğrenim sağlanmıştır. İleride kullanılabilecek veri madenciliği, yapay sinir ağları ve yeni makine öğrenmesi yöntemleri ile öneri sisteminde geliştirmeler devam edilebilecektir. Böylece farklı alanlarda elde edilmiş kazanımlar bulunmaktadır.

7. Kaynaklar

[1] Richard P.Lippman, "An Introduction to Computing with Neural Nets", 1987

[2] Herlocker J. L., "Evaluating collaborative filtering recommender Systems", In ACM Transactions on Information Systems, 2004

[3] Billsus, D., and Pazzani, M.,. "Learning Collaborative Filters. Proceedings of ICML'98", 1998

[4] https://medium.com/@ayyucekizrak/derin-%C3%B6%C4%9Frenme-i%C3%A7in-aktivasyon-fonksiyonlar%C4%B1n%C4%B1n-kar%C5%9F%C4%B1la%C5%9Ft%C4%B1r%C4%B1lmas%C4%B1-cee17fd1d9cd

[5] Josef Feigl ve Martin Bogdan, "Collaborative Filtering With Neural Networks", 2017

[6] https://www.kaggle.com/netflix-inc/netflix-prize-data

[7] F. Kaya, "İşbirlikçi Filtrelemeye Dayalı Web Tabanlı Öneri Sistemi Geliştirilmesi", 2012

[8] **https://en.wikipedia.org/wiki/Quantile**

[9] Özcan A., "Sosyal ağ verisi kullanarak mobil telefonlar için öneri altyapısı tasarlanması", İstanbul Teknik Üniversitesi, Fen Bilimleri Enstitüsü, İstanbul, 2010

[10] Teuvo Kohonen, "An introduction to Neural Computing", 1988

[11] Anil K.Jain, Jianchang Mao ve K.Mohuiddin, "Artificial Neural Networks: a Tutorial", 1996

[12] Gözde Özbal, "A Ccontent Boosted Collaborative Filtering Approach for Movie Recommendation Based On Local & Global User Similarity and Missing Data Prediction", 2009

[13] Hilal Karaman, "A Content Based Movie Recommendation System Empowered By Collaborative Missing Data Prediction", 2010

[14] Masahiro Morita ve Yoichi Shinoda, "Information Filtering Based on User Behavior Analysis and Best Match Text Retrieval", 1994

[15] Robin Burke, "Integrating Knowledge-Based and Collaborative-filtering Recommender Systems", 1999

[16] Yehuda Koren ve Robert Bell, "Advances in Collaborative Filtering", 2015

[17] Andrey Feuerverger, Yu He ve Shashi Khatri, "Statistical Significance of the Netflix Challenge", 2012

[18] Robert M. Bell ve Yehuda Koren, "Improved Neighborhood-based Collaborative Filtering", 2007

[19] Jun Wang, Arjen P. de Vries1 ve Marcel J.T. Reinders, "Unifying User-based and Item-based Collaborative Filtering Approaches by Similarity Fusion",2006

[20] Gui-Rong Xue1, Chenxi Lin1, Qiang Yang, WenSi Xi, Hua-Jun Zeng, Yong Yu ve Zheng Chen, "Scalable Collaborative Filtering Using Cluster-based Smoothing",2005

[21] Badrul Sarwar, George Karypis, Joseph Konstan ve John Ried, "Item-Based Collaborative Filtering Recommendation Algorithms", 2001

[22] Ben Schafer, Joseph Konstan ve John Ried, "Recommender Systems in E-commerce", 1999