Yazılım Geliştirme I Dersi
Proje Raporu

Hilmi Ege Kara, 220229046
Mühendislik Fakültesi, Yazılım Mühendisliği Bölümü
Kocaeli *Üniversitesi*
Kocaeli, Türkiye
hegekara48@gmail.com

I. GIRIŞ

Bu projede MovieLens20M veri setinde apriori algoritması ile işlemler yapılarak genel ve kişiselleştirilmiş film öneri sistemi yapılması amaçlanmıştır. Bu sistemin kişisel ve popüler olmak üzere isim veya türe göre öneri yapılması amaçlanmıştır. Apriori algoritması, bu verilerdeki sık karşılaşılan kalıpları ve ilişkileri keşfederek, film önerilerini güçlendiren birliktelik kuralları oluşturulmasında kullanılmıştır.

II. MATERYAL VE YÖNTEM

1. MATERYAL

A. MovieLens20M Veri Seti

MovieLens 20M film derecelendirmelerini içeren veri setidir. Kararlı bir yapıya sahiptir. 138.000 kullanıcı tarafından 27.000 filme yapılan 20 milyon derecelendirme ve 465.000 etiket uygulaması içerir. Ayrıca, 1.100 etiket arasında 12 milyon ilişkililik puanı içeren tag genome verisi de bulunmaktadır [1]. Projede kullanılan gerekli veri seti dosyaları şu şekildedir:

1. Movies.csv

Veri setinde bulunan filmlere ait id, başlık ve tür verilerini içirir. Projede kullanılacak olan tür ve isim bilgileri bu veri setinden referans alınarak kullanılmıştır.

2. Rating.csv

Veri setinde bulunan değerlendirmeler bu veri setinde bulunmaktadır. Filmlere ait yapılan değerlendirmeler, hangi kullanıcın bu değerlendirmeyi yaptığı, ve zaman damgası verileri bu dosyada bulunmaktadır.

B. Mlxtend Kütüphanesi

Mlxtend (makine öğrenimi uzantıları), günlük veri bilimi görevleri için faydalı araçlar sunan bir Python kütüphanesidir [2]. Bu kütüphanede bulunan association_rules ve fpgrowth fonksiyonları projede aktif olarak kullanılmıştır.

1. fpgrowth

FP-Growth, birliktelik kuralı öğreniminde uygulamaları olan sık görülen öğe kümelerini çıkaran bir algoritmadır ve yerleşik Apriori algoritmasına popüler bir alternatif olarak ortaya çıkmıştır. Özellikle, ve Apriori sık desen madenciliği algoritmasından farkı, FP-Growth'un aday küme oluşturma gerektirmeyen bir sık desen madenciliği algoritması olmasıdır. İçsel olarak, aday kümeleri açıkça oluşturmadan FP-tree (sık desen ağacı) adı verilen bir veri yapısı kullanır, bu da onu büyük veri setleri için özellikle cazip hale getirir [2]. Fpgrowth algoritmasının, apriori algoritmasından daha az kaynak tüketimi ve daha hızlı çalışması sebebiyle apriori algoritması yerine kullanılmasına karar verilmiştir.

2. association_rules

Kural üretimi, sık desenlerin madenciliğinde yaygın bir görevdir. Bir birliktelik kuralının ilgi derecesini değerlendirmek için farklı metrikler geliştirilmiştir [2]. Bu metrikler şu şekildedir;

- support
- confidence
- lift
- leverage
- conviction
- zhangs_metrics

Lift metriği, öneri sistemlerinde genellikle veri kümelerinin birlikte görülme olasılığını analiz etmek için kullanılır. Bu metrik, bir filmin diğer filmlerle ne kadar ilişkili olduğunu gösterir. Yani, lift, bir filmin diğer filmlerle birlikte izlenme olasılığının, bu iki filmin bağımsız olarak izlenme olasılığına oranını ölçer. Bu nedenle lift metriğinin projede kullanılmasına karar verilmiştir.

C. Tkinter Kütüphanesi

Tkinter kütüphanesi ("Tk arayüzü"), Tcl/Tk GUI araç takımının standart Python arayüzüdür [3]. Hem Tk hem de tkinter, macOS dahil olmak üzere çoğu Unix platformunda ve Windows sistemlerinde mevcut olması, temiz ve sade olması ve kolay yönetilebilir bir masaüstü arayüzü oluşturma

kütüphanesi olması sebebiyle projede kullanılmasına karar verilmiştir.

D. Pandas Kütüphanesi

Pandas, "ilişkisel" veya "etiketli" verilerle çalışmayı kolay ve sezgisel hale getirmek için tasarlanmış, hızlı, esnek ve ifade gücü yüksek veri yapıları sağlayan bir Python paketidir. Amacı, Python'da pratik ve gerçek dünyaya yönelik veri analizi yapmak için temel, yüksek seviyeli bir yapı taşı olmaktır [4]. MovieLens20M veri setini manipüle etmek için kullanılması uygun görülmüştür.

2. YÖNTEM

A. Veri Temizleme

Veri temizleme adımında, beğenilen filmler ile işlemler yapmak ve öneri güvenirliğini artırmak amacıyla, 3 puanın altında verilen değerlendirmeler veri setinden çıkarıldı. Belirlenen eşik, düşük puanlı filmlerin kullanıcıların gerçekten ilgisini çekmeyen ve seyirciyi tatmin etmeyen filmleri yansıtması sebebiyle, öneri sisteminin doğruluğunu artırmayı amaçlamaktadır.

Ek olarak, sistemin kullanıcı davranışlarına dayalı olarak öneriler sunabilmesi ve birliktelik kurallarının daha optimize bir şekilde çalışmasını sağlamak için, 100'den az film izleyen kullanıcıları veri setinden çıkarıldı. Bu kullanıcılar yeterli miktarda veri sağlamadığından öneri doğruluğunu azalttığı gözlemlenmiştir. Yeterli sayıda film izleyen kullanıcılar üzerinden yapılan analiz, öneri algoritmalarının az kullanıcı sayısında daha başarılı olduğu fark edilmiştir.

Bu iki adım, veri setindeki gürültüyü azaltarak, hem fpgrowth algoritmasıyla oluşturulacak sıklık veri kümelerinin hem de association_rules algoritması ile elde edilecek ilişkilerin doğruluğunu ve güvenirliğini artırmak için önemlidir.

Yapılan işlemler neticesinde 34.644 kullanıcı ve 22.608 adet film verisi ile işlemlere devam edilecektir. Oluşturulan veriler yüksek boyutta yer kapladığından ötürü .csv dosyasına dönüştürülmeksizin sonraki adımlarda kullanılmıştır.

B. Sıklık Veri Kümesinin Çıkarılması

Fpgrowth algoritması için kullanıcı-film matrisi oluşturularak her kullanıcının izlediği filmler analiz edilebilir bir yapıya dönüştürüldü. Bu matris, kullanıcılar ve filmler arasındaki etkileşimleri göstererek, izlenen filmlerin ilişkilendirilmesini sağladı.

Verilerden sıklık veri kümelerinin belirlenmesi için 0.2 destek (support) değeri kullanıldı; 34.644 kullanıcı için %20'lik bir destek değeri, bir film çiftinin sıklık veri kümesine girebilmesi için en az 7.000 kez birlikte izlenmiş olması gerektiği anlamına gelir. Bu destek değeri ile sıkça izlenen film kümelerine odaklanılarak, öneri algoritmasının daha anlamlı ve geniş kitlelere hitap eden öneriler oluşturması amaçlandı. Sıklık veri kümeleri, popüler ve birbiriyle ilişkili filmlerin belirlenmesine yardımcı olarak öneri sisteminin isabet oranını artırmakta ve kullanıcıların ilgisini daha iyi yakalamaktadır.

Popüler film öneri fonksiyonunda kullanılmak üzere 2. sıklık veri kümesi oluşturulmuştur. Sıklık kümesi oluşturulurken min_support parametresi 0.15 olarak ayarlanmıştır. Oluşturulan veri kümesi kural oluşturma işlemine tabii tutulmadan direkt olarak kullanarak sıklık kümesinde bulunan elemanların çokluğundan yararlanılmıştır. Oluşturulan veri kümesi .csv dosyasına dönüştürülerek saklanmıştır.

Oluşturulan sıklık veri kümeleri çok yüksek boyuta sahip olmasından ötürü .csv dosyasına dönüştürülmeksizin sonraki adımlarda kullanılmıştır.

C. Kuralların Oluşturulması

Sıklık veri kümeleri oluşturulduktan sonra, veriler arasındaki ilişkileri belirlemek amacıyla association_rules algoritması kullanılarak kurallar çıkarıldı. Bu adımda, iki film arasındaki bağın gücünü ölçmek için 'lift' metriği tercih edildi. Lift değeri, iki filmin birlikte izlenme olasılığının, her iki filmin ayrı ayrı izlenme olasılıklarına göre ne kadar arttığını ifade eder. Daha isabetli öneriler yapıldığı gözlemlendiği için film isimlerine göre yapılan önerilerde kullanılmak üzere 1.7 eşik değeri, tür kullanılarak yapılan önerilerde 1.2 eşik değeri kullanıldı.

Bu eşik değeri sayesinde yalnızca birbirleriyle anlamlı şekilde ilişkilendirilmiş film çiftleri seçildi. Bu da öneri sisteminin daha güçlü ve isabetli öneriler sunmasını sağlayarak, kullanıcıların daha ilgili olacağı film önerilerinin yapılmasına katkı sağladı.

Oluşturulan kurallar öneri sisteminde kullanılmak üzere .csv dosyalarında dönüştürüldü ve kaydedildi. Daha sonrasında öneri yapılırken bu dosyalar kullanılması hedeflendi.

D. Öneri Oluşturma Fonksiyonu

Öneri oluşturma fonksiyonu, izlenen filmlerle ilişkili yeni film önerileri sunmak amacıyla geliştirilmiştir. İlk olarak, izlenen filmlerle ilişkili kurallar belirlenir. Bu adımda, her bir kuralın öncül kümesindeki(antecedents) filmlerin, kullanıcının izlediği filmlerle örtüşüp örtüşmediği kontrol edilir. Bu filtreleme, yalnızca kullanıcının izlediği filmlerle ilişkili kuralları seçmeyi sağlar.

Sonrasında, seçilen kurallar, kuralların lift değerine göre sıralanır. Yüksek lift değeri, iki filmin birlikte izlenme olasılığının arttığını gösterir ve bu da öneri algoritmasının daha doğru ve etkili sonuçlar üretmesine yardımcı olur.

En yüksek lift değerine sahip olan kurallar, kullanıcının izlediği filmlerle ilişkilendirilmiş yeni filmleri öneri listesine ekler. Sonuç olarak, kullanıcıya önerilecek filmler bir araya getirilir ve en uygun filmler, kullanıcının geçmiş tercihleri ve

film ilişkileri doğrultusunda sıralanır. Bu süreç, öneri sisteminin doğruluğunu artırarak, kullanıcının ilgisini çekecek önerilerin sunulmasını sağlar.

E. Masaüstü Uygulamasının Oluşturulması

Daha sonra, tkinter kütüphanesi kullanılarak bir masaüstü uygulaması oluşturuldu. Bu uygulama, kullanıcıların film önerilerini kolayca alabilmesi için grafiksel bir arayüz sunmaktadır. Uygulamanın temel işlevselliği, öneri oluşturma sürecini kullanıcı dostu bir şekilde sunmaktır.

Gerekli fonksiyonlar, kullanıcı etkileşimini sağlamak için oluşturulmuştur. Bu fonksiyonlar, kullanıcıdan gelen inputları alarak, öneri algoritmalarını çalıştırmakta ve sonuçları kullanıcıya gösterecek şekilde düzenlemektedir. Tkinter, basit ve etkili bir grafiksel arayüz oluşturmak için kullanılarak, önerilerin hızlı bir şekilde görüntülenebilmesi sağlanmıştır.

Bu süreç, kullanıcıların öneri sistemini doğrudan etkileşimli bir ortamda kullanabilmesini ve önerilen filmleri görsel olarak inceleyebilmesini sağlamaktadır

III. DEĞERLENDIRME ÖLÇÜTLERI

Değerlendirme ölçütleri başlığı 3 alt ana başlığa ayrılabilir;

A. Hız

1. Apriori - Fpgrowth Algoritması

Apriori algoritması, iteratif bir algoritmadır. Başlangıçta tüm öğe kümelerinin sıklığı hesaplanır, sonra her bir öğe kümesi büyütülür. Kombinasyonlar kullanılarak sayma işlemi yapılır. Her bir adımda kombinasyon sayısı 1 artırılır. Bu işlem, kütleli olmayan öğe kümeleri elenene kadar devam eder.[2]

FP-Growth algoritması, sıklıkla kullanılan öğe kümelerini “Frequent Pattern Tree” (FP-Tree) adı verilen bir yapı ile temsil eder. Bu, Apriori’ye kıyasla daha hızlı çalışmasına olanak tanır.[2]

İki algoritmayı kıyaslamak için test ortamı oluşturulmuştur. Test ortamında sadece 3’ten fazla puan almış filmler ve en az 100 film izlemiş kullanıcılar matrisi kullanılmıştır. İki algoritmada da min_support parametresi 0.25 olarak belirlenmiş olup işlemi tamamlama süreleri şu şekildedir:

Apriori : 2 dakika 35 saniye
Fpgrowth : 26 saniye

2. Linear ve Tree Veri Yapısı

Sistem genel olarak .csv dosyalarında tutulan kuralları linear search ile gezerek çalışmaktadır fakat Kişisel-Tür öneri sisteminin yavaş öneri vermesi sebebiyle heap-tree veri yapısının kullanılmasına karar verilmiştir. Fakat kurulan veri yapısının linear aramadan 3.5 saniye daha yavaş öneri vermesi sebebiyle tren yapısı kaldırılmış olup linear search ile arama yapılmaya devam edilmiştir.

B. Kaynak Tüketimi

1. Apriori - Fpgrowth Algoritması

Apriori algoritması linear şekilde sayarak, fpgrowt algoritması fp-tree ağacından verileri çekerek fpgrowth ağacının daha az kaynak tüketmesinin beklenmektedir. Hız başlığında yapılan test ortamı tekrar oluşturularak algoritmaların kaynak tüketimi test edilmiştir. Buna göre kullanılan kaynak miktarları (ram) şu şekildedir;

Apriori : 14.36 GB
Fpgrowth : 1.54 GB

C. Doğruluk

Oluşturulan test.py dosyası ile oluşturulan arayüze doğru türlerin basılıp basılmadığı, kullanılan filmlerin eksik veri içerip içermediği test edilmiştir.

IV. SONUÇLAR VE TARTIŞMA

Yukarıdaki materyaller ve yöntemler kullanılarak oluşturulan sistemin yaptığı önerilerin sağlık öneriler olması sonucunda (örneğin “Toy Story” filmine “Toy Story 2” filminin önerilmesi, “Shrek” filmini izleyen kullanıcıya “Shrek 2” filmini önerilmesi) sistemin başarılı bir şekilde çalıştığı düşünülmektedir. Yapılan proje neticesinde veri madenciliğinde kullanılan apriori, fpgrowth, association_rules algoritmaları; veri işlemede kullanılan panda kütüphanesi; masaüstü arayüz yapımında kullanılan tkinter kütüphanesi hakkında bilgi ve deneyim kazanılmıştır.

KAYNAKÇA

1. https://grouplens.org/datasets/movielens/20m/
2. https://rasbt.github.io/mlxtend/
3. https://docs.python.org/3/library/tkinter.html
4. https://pandas.pydata.org/docs/getting_started/overview.html