Film Öneri Sistemi
Movie Recommendation System

1. Zeynep ÖZÇELİK
Yazılım Mühendisliği
Kocaeli Üniversitesi
Kocaeli,Türkiye
ozcelikzeynep03@gmail.com

***Özetçe*— Film öneri sistemleri,izleyicilerin tercihleri doğrultusunda öneriler sunan algoritmalardır.Bu çalışmada,MovieLens20M veriseti üzerinde very işlemleri gerçekleştirilmiştir. İşlenen veriler üzerine Apriori algoritması uygulanması sonucu sık izlenen filmlerden birliktelik kuralları oluşturulmuştur. Popülerlik bazlı ve kişiselleştirilmiş olmak üzere iki öneri türü sunulmuştur. Her iki tür de film türü ve ismine göre filtrelenerek özelleştirilmiştir. Öneriler, Flask tabanlı grafik kullanıcı arayüzünde sunulmuş ve dinamik olarak gözlenmiştir. FP-tree kullanılarak sistemin bellek ve zaman verimliliği artırılmış, algoritmaların doğruluğu ve etkinliği test edilmiştir.**

Anahtar Kelimeler — Film Öneri Sistemi,Popülerlik tabanlı öneri ,Apriori Algoritması,Birliktelik Kuralı,Flask,Python

***Abstract*— Film recommendation systems are algorithms that provide movie suggestions based on the preferences of viewers. In this study, the MovieLens20M dataset is used to create a user-item matrix for the application of the Apriori algorithm, identifying clusters of frequently watched movies. Two types of recommendations are offered: popularity-based and personalized. Both types are further customized based on movie genres and titles. The recommendations are presented through a Flask-based graphical user interface and dynamically observed. The use of FP-tree enhances memory and time efficiency, and the accuracy and effectiveness of the algorithms are tested.**

***Keywords*— Film Recommendation System, Popularity-based Recommendation, Apriori Algorithm, Association Rule, Flask, Python**

I. GİRİŞ

Film öneri sistemleri, izleyicilerin izlenme geçmişlerine göre kişiye özel film önerileri sunan algoritmalardır. Bu sistemler, izleyicilerin izlenme geçmişini takip ederek doğru ve uyumlu öneriler bulmayı hedeflemektedir. Film öneri sistemleri, kişiselleştirilmiş önerilerin yanında genel izleyicilerin tercihlerine göre popülerlik tabanlı önerilerde de bulunmaktadır.

Bu projede , MovieLens20M[3] veri seti kullanılarak, kullanıcı-izlediği film ilişkisini bulunduran bir boolean matrisi oluşturulmuş ve bu matris üzerinde Apriori algoritması uygulanarak sık izlenen film kümeleri oluşmuştur.Bu kümeler ile filmlere düşen öneriler support,confidence ve lift değerlerine göre incelenerek belirlenmiştir.Öneriler popüleriteye ve kişisel tercihe bağlı olarak iki bazda incelenmiştir. Bu iki öneri türü de film türü ve film ismine göre filtrelenerek daha özel sonuçlara indirgenmiştir.

Sistem performansını ve verimliliğini artırmak için sık öge kümeleri FP-tree ağacında depolanmıştır.Bu yöntem,bellek kullanımını ve zaman karmaşıklığını optimize etmeye yardımcı olmuştur.Sistemin doğru önerilerde bulunabilmesi içinde öneriler için yazılan fonksiyonlar test edilmiştir.

II. MATERYAL VE YÖNTEM

Bu proje, makine öğrenme algoritmaları ve veri işleme için hazır kütüphaneler bulundurması sebebiyle Python programlama dili ile yazılmıştır.Veritabanı olarak, genellikle user,filmler ve filmlere verilen oyları içeren CSV dosyaları tercih edilmiştir.

A. Kullanılan Kütüphaneler Ve Araçlar

- **Pandas** : Veri işleme ve analizi için kullanılmıştır.
- **Numpy**: Veriler üzerinde nümerik hesaplamalar ve veri manipülasyonu için kullanılmıştır.
- **Apriori(mlxtend.frequent_patterns)** : Sık öge kümelerini bulmak için kullanılmıştır
- **Assosiacition rules(mlxtend.frequent_patterns):** Elde edilen sık öge kümeleri ile birliktelik kuralı oluşturmak için kullanılmıştır.
- **Memory_profiler** : Bellek kullanımını izlemek için kullanılmıştır.
- **Matplotlib.pyplot** : Verileri görselleştirmek için kullanılmıştır.
- **Flask** : Web uygulaması geliştirmek için kullanılmıştır.
- **Unittest**: Fonksiyon bazlı testler yapmak için kullanılmıştır.
- **Time**: Kodun çalışma süresini ölçmek için kullanılmıştır.

B. VeriSeti

Bu projede , film öneri sistemi için MovieLens20M veri seti kullanılmıştır.Bu veri seti kullanıcıların 20 milyon

derecelendirme ve izleyicilerin filmlere verdikleri oylardan oluşmaktadır. 138.000 kullanıcı tarafından 27.000 farklı filme dair verilen oyları içermektedir.

Bu proje veriseti içerisinden iki dosya baz alınarak oluşturulmuştur:

- **ratings.csv** : Her satır bir kullanıcını belirli bir filme 1-5 arası oylardan ve bu puanın verildiği zaman damgasını içerir.Bu dosya ile kullanıcıların izlediği filmler ve beğenileri analiz edilmesi için kullanılmıştır.

- **movies.csv**: Bu dosyada her filme ait movieId,title(başlık),genre(tür) bilgileri yer almaktadır.Film önerileri geliştirildiği zaman filmlerin tür bazlı izleme eğilimleri analiz edilirken kullanılmıştır.

C. Veri Yükleme ve İşleme Adımları

Bu projede, MovieLens20M veri seti, gerekli verilerin yüklenmesi, işlenmesi ve temizlenmesi amacıyla çeşitli veri işleme adımlarından geçirilmiştir. Veri işleme süreci, veri setlerinin yüklenmesi,eksik bilgilerin temizlenmesini ,veriler içerisinde ilişkisel tablo ve matris oluşturulmasını içermektedir. Veri işlemede kullanılan fonksiyonlar ve yöntemler aşağıda detaylandırılmıştır:

a) Veri Yükleme

- **load_datasets() :** Bu fonksiyoni ile, veri setindeki her bir dosya csv'den çekilerek pandas dataframe'ine dönüştürülür.

- **preprocess_timestamps(tag_df,rating_df):** Önceden belirlenen .info() komutu ile bilgileri alınan df 'lerin timestamp veri tipi "object'ten" "datetime" formatına çevrilmiştir.

```
def preprocess_timestamps(tag_df, rating_df):

tag_df["timestamp"] = pd.to_datetime(tag_df["timestamp"])
rating_df["timestamp"] =
pd.to_datetime(rating_df["timestamp"])
return tag_df, rating_df
```

b) Veri Temizleme

Her bir DataFrame içindeki NaN değerler belirlenmiş ve temizlenmiştir. Örneğin, link_df DataFrame'inde eksik değerler şöyle tespit edilmiştir:

link_df.isnull().sum() komutu ile link_df içindeki NaN değerler kontrol edilmiştir.

Elde edilen sonuca göre tmdbId sütununda 252 NaN veri olduğu tespit edilmiştir

```
movieId      0
imdbId       0
tmdbId     252
dtype: int64
```

link_df.dropna(subset=['tmdbId']) komutu kullanılarak tmdbId sütunundaki NaN değerlerr bulunan satırlar temizlenmiştir.

c) User-Movie Tablosu ve Matrisi Oluşturma

- **User-Movie Tablosu Oluşturma**

Bu adımda ,her kullanıcın izlediği filmlerin bulunduğu izleyici-film tablosu oluşturulmuştur.

```
izleyici_film_tablosu =
rating_df.groupby('userId')['movieId'].apply(list).reset_index()
izleyici_film_tablosu.columns = ['userId', 'İzlenilen Filmler']
```

- **User-Movie Matrisi Oluşturma**

Bu adımda, matrisin bellek aşımı ve üzerinde yapılan işlemlere bağlı olarak zaman harcaması göz önüne alınarak matrisin oluşmasında belirli kısıtlamalar kullanılmıştır.İlk olarak belirlememiz istenen 10 tane film türü seçimi yapılmıştır. En çok izlenen türlerden ilk 10'nu alınıp bunlar üzerinde 100 tane film seçimi yapılmıştır ve bu filmlerin izlenme sayısı hesaplanıp , sıralama işlemi uygulanmıştır.

Sonuç olarak alınan verilerden izleyici ve izlenme bilgileri (1-0) olacak şekilde bir matris oluşturulmuştur.

```
user_movie_matrix=filtered_ratings.pivot_table(index="userI
d", columns="movieId", values="rating", aggfunc="size",
fill_value=0)

user_movie_matrix = user_movie_matrix.applymap(lambda x:
1 if x > 0 else 0)
```

D. Apriori Uygulanması ile Sık Öge Kümeleri ve Birliktelik Kuralı Tespiti

Apriori algoritması, ilişki kuralı öğrenimi için kullanılan gözetimsiz bir makine öğrenimi algoritmasıdır.[1] Bu algoritma, veri kümeleri içindeki ilişkilerin sıklığını tespit etmek için kullanılmaktadır. Bu projede, izlenen filmler arasındaki ilişkiyi ve izlenme sıklığını tespit etmek için Apriori algoritması uygulanmıştır.

a) Sık Öge Kümeleri

Apriori algoritmasına belirli bir support eşiği belirlendikten sonra veri kümesi üzerinde sık öge kümeleri oluşturur.Bu kodda da en az %20 sıklıkla görülen öge kümeleri alınmıştır.

```
f_items=apriori(user_movie_matrix,min_support=0.2,use_col
names=True)
```

b) Birliktelik Kuralı Oluşturulması

Birliktelik kuralı ögeler arasındaki ilişkiyi tanımlar ve hangi ögelerin birlikte bulunabileceğini "support","confidence" ve "lift" metriklerine göre belirlenir:

- **Support(Destek)** : Yalnızca bir filmin veya film kombinasyonlarının birlikte görülme oranıdır.[2]
- **Confidence(Güven)**: Ögelerin çift olarak değerlendirilmesidir. X filminin izlendiğinde Y filminin izlenme olasılığı.[2]
- **Lift** : Bir film izlendiğinde diğer filmin izlenme olasılığının kaç kat artacağını gösteren metriktir.[2]

```
rules=association_rules(f_items_sorted,metric='confidence',mi
n_threshold=0.1)
```

Bu projede , sık öge kümeleri üzerinden %10 güvenilirlik(confidence) oranı ile birliktelik kuralları oluşturulmuştur.

E. Popülerlik Tabanlı Öneri Sistemi

Bu projede, popülerlik tabanlı öneri sistemi, izleyicilerin tercihleri doğrultusunda en popüler 50 filmin belirlenmesi amacıyla kullanılmaktadır.

a) Film İsmine Göre Öneri Sistemi

İzleyicilerin izlenme geçmişleri dikkate alınarak en çok izlemeyi tercih ettikleri 50 filmi bulmayı amaçlamaktadır. Bu öneri sistemi için genel izleyiciye bağlı sık film kümelerine özel çıkarılan kuralların destek, güven ve lift değerleri baz alınarak öneri sistemi kurulmuştur.

Bu öneri sistemi için yazılmış olan popüler_öneriler_film_ismi() fonksiyonu, Apriori algoritması ile oluşturulmuş kurallardan elde edilen consequents (önerilen filmler) sütunundaki movieId'leri çeker. Elde edilen movieId'ler kullanılarak ilgili film bilgileri (başlık, tür, vb.) kural metrik bilgileri ile birleştirilir. Bu birleştirme işlemi sonucunda, her film için support, confidence ve lift metrikleri elde edilir.

b) Film Türüne Göre Öneri Sistemi

Bu sistemde, izlenme geçmiş verilerini dikkate alarak popüler verileri belirlendikten sonra , baz alınan film türüne bağlı olarak öneriler sunulmaktadır.

Projede bu algoritma popüler_öneri_film_türü() fonksiyonu ile belirtilmiştir.Kullanıcı seçimine bağlı olarak filtrenen film türünü içeren 50 film sıralanır.

F. Kişiselleştirilmiş Öneri Sistemi

Kişiselleştirilmiş Öneri Sistemleri bölümünde, izleyicilerin izleme geçmişleri dikkate alınarak, kullanıcıya özgü öneriler sunulmaktadır. Bu sistem, izlenen filmlere benzer veya aynı filmi izleyen diğer kullanıcıların tercih ettikleri filmleri önerir.

a) Film İsmine Göre Öneri Sistemi

Bu öneri sisteminde, kullanıcının izlenme geçmişi ve geçmişte izlemeyi tercih ettiği film türlerine bağlı olarak önerilerde bulunmuştur.Bu algoritma ilgili projede kişiselleştirilmiş_öneri_film_ismi() fonksiyonu ile gösterilmiştir.

İlk olarak, kullanıcının geçmiş izleme verilerinden izlemeyi tercih ettiği film türleri çıkarılır ve sıralanır.Sonrasında,birliktelik kuralları ile izlenen filmler ile ilişikili film önerilerine bulunulur.Bu öneriler kurallarda belirlenen destek,güven ve lift metriklerine göre sıralanır. Bu sıralama kriterlerindeki metriklere ek olarak kullanıcının izlemeyi tercih ettiği film türleri de baz alınarak kullanıcı beğenileri göre kişiselleştirilmiş önerilerde bulunulur.

b) Film Türüne Göre Öneri Sistemi

Kişiye özel olan önerilerin, film türü ve izlenme geçmişi baz alınarak yapıldığı sistemlerdir.Film ismine göre yapılan önerilerden farkı film türü bu sistemde sıralama metriği olarak değil, filtreleme aracı olarak kullanılmıştır. Bu algoritma ilgili projede kişiselleştirilmiş_öneri_film_türü() fonksiyonu ile gösterilmiştir.

Bu sistemin algoritması,kullanıcılara kurallardan gelen öneri verileri, kullanıcının izlediği filmlerin türlerine göre filtrelenmesi ile başlar.Kullanıcının izlediği filmlere (antecedents) karşı önerilen filmler (consequents) arasından benzersiz filmlerin seçimi yapılır ve destek,güven,lift metriklerinin sırasına göre öneri sistemi kurulur.

G. Arayüz(GUI)

Bu arayüz, kullanıcıların film öneri sistemi verileriyle kolay etkileşim kurmasını sağlamak için Flask kullanılarak geliştirilmiştir. Arayüz ,oluşturulan algoritmalar özelinde Öneri Türleri,Filtreleme Türleri,Kullanıcı ID seçimi ve Film Türü seçimlerini listbox aracılığıyla yapılacak şekilde ayarlanmıştır.

III. DEĞERLENDİRME ÖLÇÜTLERİ

A.) Destek,Güven ve Lift Testi

Birliktelik kuralı sonucu oluşan support,confidence ve lift değerleri default olarak belirlenen bir kurala bağlı olarak doğru öneri yapılıp yapılmadığı test edilecektir.

Örnek: 296'idli filme önerilen 593 id'li film için belirlenen kural support,confidence ve lift metriklerine göre değerlendirilecektir.

a.) Support(Destek)

Support bir verideki sık bulunma durumunu ifade eder.Yani belirli bir kombinasyonun kaç adet bulunduğunun oranıdır.

Pulp Fiction -> Silence of Lambs,The ,
Support :0.344516

b.) Confidence (Güven)

Confidence , antecedent değeri doğru olduğunda , consequent değerinin doğru olması gerekmektedir.Yani antecedent'e ait bir film izlendiğinde consequent'e ait bir filmin izlenme oranıdır.

Pulp Fiction -> Silence of Lambs,The ,
Confidence : 0.7537..

c.) Lift (Kaldırma)

Lift iki öge arasındaki ilişkiyi belirlemektedir.Ögelerinden birbirine olan bağımlılığını ölçer.Eğer lift değeri 1'den büyükse pozitif ilişki ,düşükse negatif ilişki vardır.

Pulp Fiction -> Silence of Lambs,The ,
Lift:1.5509

Test edilen kriterlere göre popüler filmler listesinde 593 id'li filmin bulunması gerekmektedir.Popüler ilk 5 film aşağıdaki gibidir.

1-Silence of the Lambs, The (1991)-593
2-Forrest Gump (1994)
3-Shawshank Redemption, The
4-Jurassic Park (1993)
5-Terminator 2: Judgment Day (1991)

Bu kombinasyonda , support %34.45, confidence %75.38 ve lift 1.55'tir. Bu, (593) ve (296) öğelerinin birlikte görülme sıklığı ve kullanıcıların bu filmi birlikte izleme oranı diğer film kombinasyonlarına göre daha fazla demektir.

B.) Öneri Doğrulama Testleri(Birim Testler)

Öneri doğruluğu testleri, film sisteminin sunduğu önerilerin doğru ve güvenilir şekilde öneriler sunduğunu belirlemek için yapılmaktadır.

a) Popüler Filmler Üzerine Testler

Bu test, popüler film önerilerini test etmek için kullanılır.Bu testlerde izlenme sayısı ve verilen oylara doğru orantılı olarak öneriler sunulmalıdır.

Testin Amacı: Filmler arasından en çok izlenen ve en yüksek oyları alanlar öneri listesinde bulunmalıdır.

Beklenen Sonuç: "Pulp Fiction (1994)", "Forrest Gump (1994)", "Shawshank Redemption (1994)" gibi popüler filmlerin önerilmesi.

Kod gösterimi:

```
top_movies = popüler_öneriler_film_ismi(self.movie_df, self.rules)
self.assertIn('Pulp Fiction (1994)', top_movies['title'].values)
self.assertIn('Forrest Gump (1994)', top_movies['title'].values)
self.assertIn('Shawshank Redemption, The (1994)', top_movies['title'].values)
```

Test Sonucu : OK

b) Film Türü Bazlı Popüler Öneriler Üzerine Testler

Bu test, Film türü bazlı popüler 50 filmi öneren fonksiyonun öneri sisteminin doğruluğunu test etmektedir.

Testin Amacı: Kullanıcıya belirli bir türde (örneğin, 'Drama') film önerileri yapılmalıdır.

Beklenen Sonuç: "Drama" türüne ait filmler (örneğin "Pulp Fiction (1994)", "Forrest Gump (1994)") doğru şekilde önerilmelidir.

Kod Gösterimi :

```
top_drama_movies = get_top_50_for_genre_with_rules('Drama', self.movie_df, self.rules, self.rating_df, self.genre_viewer_counts)

self.assertIn('Pulp Fiction (1994)', top_drama_movies['title'].values)
```

self.assertIn('Forrest Gump (1994)', top_drama_movies['title'].values)

Test Sonucu : OK

c) Kişiselleştirilmiş Film İsimleri Üzerine Testler

Bu test kullanıcıların, geçmiş izlenme tercihleri ve beğenilerine göre film önerilerinin doğruluğunu test etmektedir.Kullanıcıya beğenebileceği filmler üzerinden testler uygulanmıştır.

Testin Amacı: Kullanıcıya, önceki izlemeleri ve puanlamalarına göre doğru filmler önerilmelidir.

Beklenen Sonuç: "Jurassic Park (1993)", "Forrest Gump (1994)", " Fight Club (1999)" gibi filmler önerilmelidir. Bu filmler, kullanıcının önceki tercihleriyle uyumlu olmalıdır.

Kod Gösterimi :

```
recommended_movies = kişiselleştirilmiş_öneri_film_ismi(self.rating_df, self.movie_df, self.rules, user_id=318)

self.assertIn('Jurassic Park (1993)', recommended_movies['title'].values)
self.assertIn('Forrest Gump (1994)', recommended_movies['title'].values)
self.assertIn('Fight Club (1999)', recommended_movies['title'].values)
```

Test Sonucu : FAILED (failures=1)(Fight Club (1999) öneri sisteminde bulunamadı) %87,5 olasılıkla doğru önerilmiştir.

d) Kişiselleştirilmiş Film Türü Önerileri Üzerine Testler

Bu test, kullanıcıların izlediği filmlere göre önerilen filmlerin belirli türe ait filmlere ait olmasını test etmektedir.

Testin Amacı: Kullanıcılara sadece belirli bir türdeki filmler önerilmelidir.

Beklenen Sonuç: Kullanıcıya, yalnızca seçilen türdeki filmler önerilmelidir. Örneğin, 'Drama' türündeki "Pulp Fiction (1994)" ve "Forrest Gump (1994)" gibi filmler önerilmelidir.

Kod gösterimi:

```
user_id = 318
selected_genre = 'Drama'
recommended_movies = kişiselleştirilmiş_öneri_film_türü(self.rules, self.rating_df, self.movie_df, user_id, selected_genre)
self.assertIn('PulpFiction(1994)',recommended_movies['title'].values)
self.assertIn('ForrestGump(1994)',recommended_movies['title'].values)
self.assertIn('ShawshankRedemption,The(1994)',recommended_movies['title'].values)
```

Test Sonucu : OK

d.) FP-Tree ve Apriori Performans Karşılaştırması

Fp-tree, apriori ile oluşan sık öge kümelerinin ağaç üzerinde sıklık değerlerine göre sıralanması ile gereksiz gezinmelerden kaçınarak bellek ve işlem süresinin azaltılmasında yer alır.

a) Bellek Kullanımı

Fptree , veri kümesinin sadece sık ögelerini tuttuğu için işlem olmayan veriler devre dışı bırakılır böylece bellekten tasarruf edilmiş olur.

b) Zaman Verimliliği

Apriori'ye kıyasla her veriyi taramadan , sık ögeleri doğrudan iletilen bir yapı kurduğu için işlem zamanı kısalır.

c) Öneri Fonksiyonları Üzerinde Performans Analizi

Fp-tree ve apriori algoritmaları ile kullanılmış olan 4 farklı öneri fonksiyonu üzerinde zaman ve bellek yönetimi üzerine testler gerçekleşmiştir.Test sonuçları Tablo1 'deki gibidir.

Fonksiyon	Apriori Zaman (s)	Fptree Zaman (s)	Apriori Bellek (MiB)	Fptree Bellek (MiB)
popüler_öneriler_film_ismi()	0.013	0.011	0.2148	0.0039
popüler_öneriler_film_türü()	0.026	0.019	0.2695	0.2343
kişiselleştirilmiş_öneri_film_türü()	0.18	0.1018	2.917	2.917
kişiselleştirilmiş_öneri_film_ismi()	2.1871	1.0266	1.878	0.093

Tablo 1 : Fp-Tree ve Apriori'ye Bağlı Performans Tablosu

IV. SONUÇLAR VE TARTIŞMA

Bu proje , iki farklı öneri sistemi yaklaşımı kullanarak kullanıcılara film önerileri sunmayı başarmıştır.Popülerlik tabanlı sistem, genel izleyici seçimine bağlı olarak en çok tercih edilen ve sevilen filmlerin önerilmesi ile

oluşmuştur.Kişiselleştirilmiş öneriler ise tek bir kullanıcıya ait izlenme geçmişine bağlı olarak sunulan önerilerdir.

Proje hazırlanması sürecinde veri temizleme ve işleme bölümü zaman alıcı olup ilgili bölümde yapılan işlemler bellekte yüksek alanlarda işlem yapması bu projenin zorlanan yanlarındandır.İkinci zorlanılan konu ise popüler filmler için oluşturan kuralların toplam 50 tane olacak şekilde öneri sunulmamasıdır.Bu sebeple oluşturulan matris üzerinde sürekli değişimler yapılmıştır.

Gelecekte, bu öneri sisteminde kullanılmayan veri dosyaları da eklenerek sistem çeşitli parametreler ile konfigüre edilip kullanıcıya iyi analiz edilmiş ve dinamik öneriler sunulabilir. Böylece öneri performansı ve kullanıcı memnuniyeti artırılabilir.

KAYNAKLAR

[1] https://www.ibm.com/topics/apriori-algorithm

[2] https://medium.com/@nrmnbabalik/apriori-algoritmas%C4%B1-c15d19d9d39e

[3] https://www.kaggle.com/datasets/grouplens/movielens-20m-dataset

[4] https://www.softwaretestinghelp.com/fp-growth-algorithm-data-mining/

[5] https://ab.org.tr/ab16/bildiri/46.pdf

[6] https://medium.com/ai-insights-cobet/why-fp-tree-outperforms-apriori-a-python-code-deep-dive-85aa09bece2c