

**Gölhisar Uygulamalı Bilimler Yüksekokulu**

**İLERİ PYTHON PROGRAMLAMA II DERSİ**

**GRUP İSMİ:** **PlayWise**

**PROJE KONUSU:Oyun Öneri Sistemi**

**Öğrenci Ad-Soyad:**

**1-Murat ÇELEBİ**

**2-Osman MANDAL**

**3-Eren PERÇİN**

**4-SEMAİ MİRAÇ**

**5-EGE YILMAZ**

MAYIS 2025 BURDUR

# İÇİNDEKİLER

[**ÖNSÖZ**](#_bookmark0) **i**

[İÇİNDEKİLER](#_bookmark1) ii

[ÇİZELGELER DİZİNİ](#_bookmark2) iii

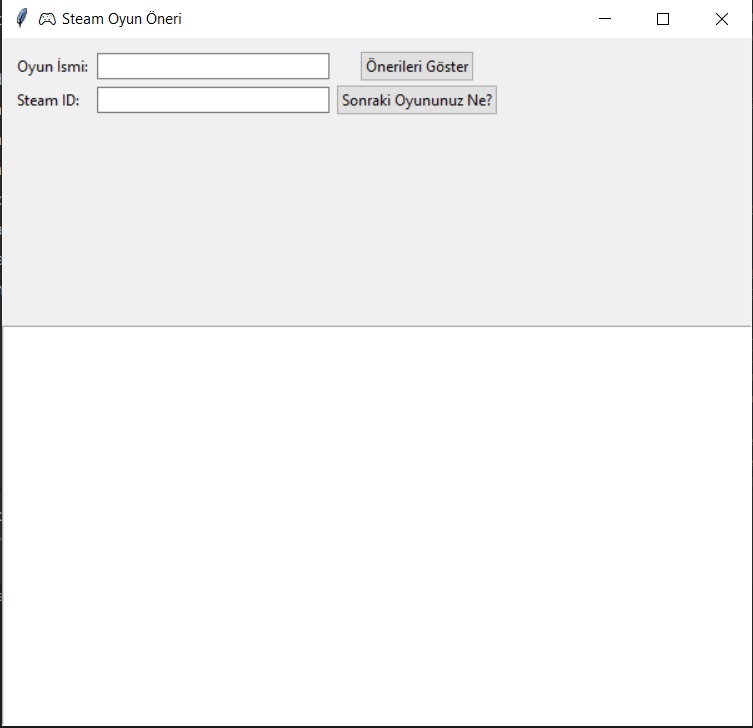
[ŞEKİLLER DİZİNİ](#_bookmark3) iv

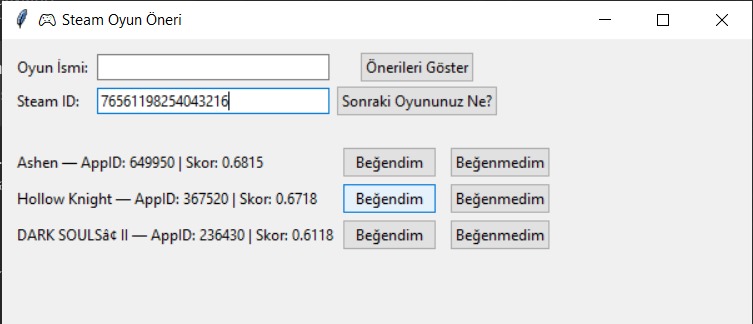
[ÖZET](#_bookmark3) v

[ABSTRACT](#_bookmark4) vi

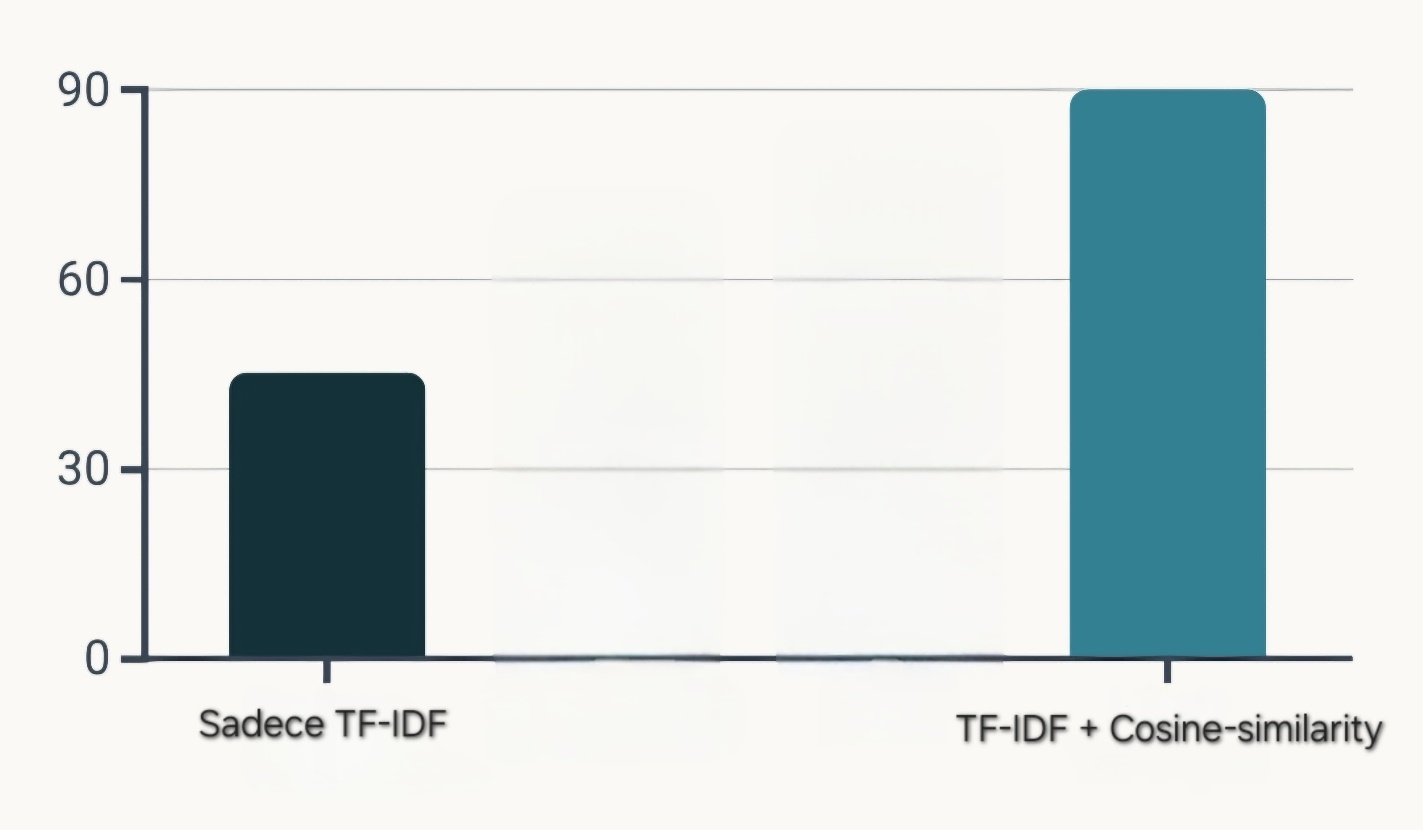
1. [GİRİŞ](#_bookmark5) 1
2. [LİTERATÜR ÖZETİ](#_bookmark6) 2
   1. A Hybrid Recommender System for Steam Games
   2. Video Game Genre Classification Using User Behavior
   3. Analysis of Hybrid Recommendation Systems in Gaming 2
3. [GEREÇ VE YÖNTEM](#_bookmark9) 3
   1. PANDAS ve OS KÜTÜPAHNESİ
   2. RE ve COUNTER KÜTÜPHANESİ
   3. COSİNE SIMILARITY
   4. THREADPOOLEXECUTOR VE JSON
   5. BEAUTİFULSOUP, TKİNTER VE THREADİNG KÜTÜPHANESİ 3
4. [BULGULAR](#_bookmark12) 3
   1. SONUÇ BULGULARI 3
5. [TARTIŞMA](#_bookmark16) 5
6. [SONUÇ VE ÖNERİLER](#_bookmark17) 5

# ÇİZELGELER DİZİNİ

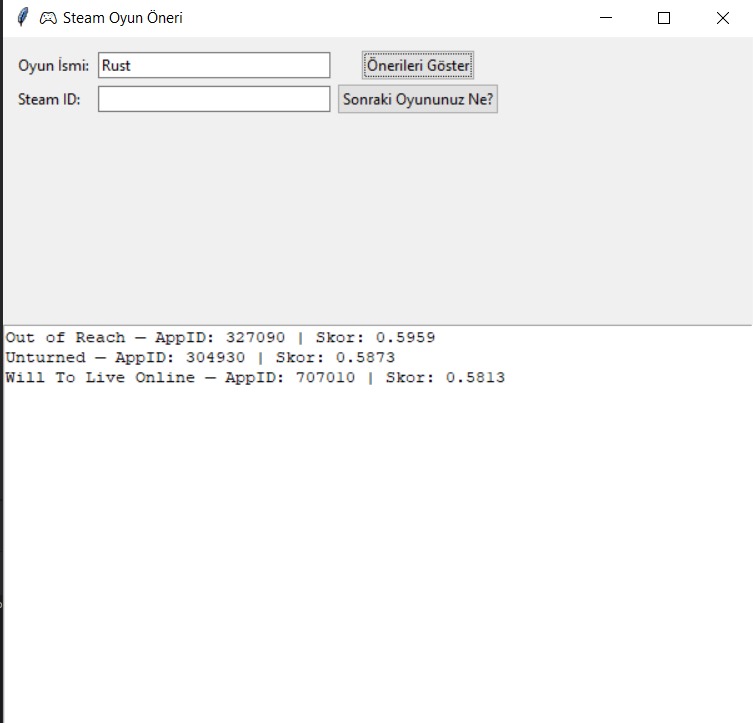
[**Çizelge 1.**](#_bookmark11) Tkinter kullanılarak geliştirilen kullanıcı arayüzünün ekran görüntüsü

[**Çizelge 2.**](#_bookmark11) Kullanıcı geri bildirimlerine göre oyun değerlendirme ve sonuç çıktısı.

# ŞEKİLLER DİZİNİ

Şekil1.TF-IDF’ın tek başına performansı ve TF-IDF + COSINE-SIMILARTY ile beraber çalışınca doğruluk performans karşılaştırması

Şekil2. TF-IDF + COSINE-SIMILARTY’in beraber çalışması sonucu çıktı örneği



# ÖZET

Başlangıçta, sadece oyunların önemli kelimelerini analiz edebiliyorduk, fakat şu anda içeriklerine göre benzer oyunlar öneren bir sistem geliştirdik. Steam'deki devasa veri setini işlemek için Valve’ın sağladığı resmi CSV dosyasını kullandık. Ancak, dosya çok büyük olduğu için veriyi parçalara böldük ve her parçayı sırayla işleyerek veriyi daha hızlı işleyebilmemizi sağladık. Başlangıçta öneri sisteminin doğruluk oranı düşüktü, bu yüzden kelimeler arasındaki anlamsız ifadeleri ayıklamak için "re" kütüphanesini kullandık. Ayrıca, "Counter" kütüphanesiyle temizlediğimiz kelimelerin sıklıklarını sayarak bu verileri karşılaştırma algoritmalarına gönderdik.

Oyunlar arasındaki benzerliği ölçmek için Cosine Similarity algoritmasını tercih ettik, çünkü TF-IDF algoritması yeterli sonuçları vermiyordu. Ancak yine de işlem süresi uzun oluyordu, bu yüzden oyunları benzer kategorilerle sınırlandırarak bu süreyi kısaltmayı başardık. Oyun açıklamaları HTML etiketleriyle karışmıştı, bu nedenle BeautifulSoup kullanarak veriyi temizledik. Kullanıcılar için bir arayüz oluşturmak amacıyla tkinter kütüphanesini kullandık. Bu arayüzde, işlem yaparken donmaları engellemek için threading kütüphanesini entegre ettik. Kullanıcılar, Steam ID’leriyle oyun önerileri alabiliyor ve beğenip beğenmediklerini geri bildirebiliyor. Bu geri bildirimleri depolayarak yapay zekayı geliştirmeye devam ediyoruz.

**Anahtar Kelimeler: veri işleme, CSV, "re", "Counter", Cosine Similarity, TF-IDF, BeautifulSoup, tkinter, threading, kullanıcı geri bildirimi, Steam API**

# ABSTRACT

Initially, we were only able to analyse the important words of games, but now we have developed a system that recommends similar games based on their content. To process the huge dataset from Steam, we used the official CSV file provided by Valve. However, because the file was so large, we split the data into chunks and processed each chunk in turn, allowing us to process the data faster. Initially, the accuracy of the recommendation system was low, so we used the ‘re’ library to extract meaningless phrases between words. We also counted the frequency of the words we cleaned with the ‘Counter’ library and sent this data to the comparison algorithms.

We preferred the Cosine Similarity algorithm to measure the similarity between the games, because the TF-IDF algorithm did not give sufficient results. However, the processing time was still long, so we managed to reduce this time by limiting the games to similar categories. The game descriptions were mixed with HTML tags, so we cleaned the data using BeautifulSoup. We used the tkinter library to create an interface for users, integrating the threading library to prevent freezes while processing. Users can get game recommendations with their Steam ID and feedback whether they like it or not. We continue to improve artificial intelligence by storing this feedback

**Keywords:** data processing, CSV, ‘re’, ‘Counter’, Cosine Similarity, TF-IDF, BeautifulSoup, tkinter, threading, user feedback, Steam API

# GİRİŞ

Bu projenin temel hedefi, oyuncu kitlesine minimum zaman harcatarak, oynadıkları oyun tarzına en yakın ve en uygun oyunları önermekti. Günümüzde oyuncular, yeni oyun keşfetmek için saatlerini harcayabiliyor ve çoğu zaman kendi zevklerine hitap eden oyunları bulmakta zorlanıyor. Biz de bu sorunu çözmek için öneri sistemleri üzerine yoğunlaştık ve işe TF-IDF algoritmasıyla başladık.

İlk denemelerde, kullanıcının girdiği oyunun açıklama metnindeki kelimeleri ayıklayarak, benzer kelimelere sahip oyunları önermeyi hedefledik. Ancak zamanla fark ettik ki, bu kelimelerin öneri değeri oldukça düşüktü. Sık kullanılan kelimeler ayırt edici olmuyor, önerilen oyunlar da beklenen kaliteyi karşılamıyordu. Üstelik bu işlemleri yapmak uzun sürüyor, öneri sistemi saatler alabiliyordu.

Zaman da bizim için önemliydi. Bu yüzden Valve’den, Steam platformundaki oyunları içeren .csv uzantılı bir veri seti indirerek sistemimizi buna göre yeniden şekillendirdik. Bu veri seti sayesinde oyunların adı, türü, etiketleri ve puanları gibi birçok detaya hızlıca ulaşabildik. Önceden saatler süren işlemler artık bir dakikadan kısa sürede tamamlanabiliyor. Bu da sistemi daha pratik hale getirdi.

Bu aşamadan sonra TF-IDF ile yetinmeyip, öneri kalitesini artırmak adına farklı algoritmaları ve kütüphaneler devreye aldık. Böylece daha sağlam ve hızlı bir öneri altyapısı oluşturmuş olduk.

# LİTERATÜR ÖZETİ

**2.1. A Hybrid Recommender System for Steam Games**

Bu çalışmada, Steam oyunları için hibrit bir öneri sistemi önerilmektedir. Önerilen sistem, içerik tabanlı filtreleme ve işbirlikçi filtreleme tekniklerini birleştirerek, kullanıcıların tercihlerine göre daha hassas oyun önerileri sunmayı hedeflemektedir. Yazarlar, öneri sistemlerinin oyun keşfi süreçlerini iyileştirme potansiyeline sahip olduğunu ve kişiselleştirilmiş oyun önerilerinin kullanıcı deneyimini artırma konusunda önemli bir rol oynadığını vurgulamaktadır.

**2.2. Video Game Genre Classification Using User Behavior**

Bu makale, kullanıcı davranışlarına dayalı bir video oyun sınıflandırma yaklaşımını incelemektedir. Oyun içi etkileşimler ve kullanıcı davranışları, oyun türlerini belirlemede daha etkili bir yöntem olarak önerilmektedir. Araştırma, mevcut tür sınıflandırma yöntemlerine kıyasla daha dinamik ve esnek bir alternatif sunarak, kullanıcı davranışlarının oyun türlerini belirleme süreçlerinde önemli bir rol oynayabileceğini göstermektedir.

**2.3. Analysis of Hybrid Recommendation Systems in Gaming**

Bu çalışma, video oyunlarında hibrit öneri sistemlerinin etkinliğini analiz etmektedir. Özellikle, kullanıcıların önceki oyun tercihleri, oyun içi davranışları ve topluluk yorumları gibi verilerin hibrit modellerde nasıl kullanıldığı ve bu modellerin öneri doğruluğunu nasıl artırdığı tartışılmaktadır. Yazarlar, hibrit sistemlerin oyun keşfi süreçlerinde ve kullanıcı memnuniyetini artırmada önemli bir araç olabileceğine dikkat çekmektedir.

# GEREÇ VE YÖNTEM

## PANDAS VE OS KÜTÜPHANESİ

Valve’den aldığımız veri settinin boyutu büyüktü ve 110000 satırdan fazla olduğu için python’ın bunu okuması çok uzun sürüyordu.Bu yüzden veri settini içinde 10000’er parçalara bölüp içinde geri birleşitrdik bu işlem içinde OS Kütüphanesi kullandık,bu işlediğimiz veriyi de işlemek için PANDAS Kütüphanesini kullandık.

(Kaynak:12) (Kaynak:13)

* 1. RE VE COUNTER KÜTÜPHANESİ

TF-IDF sayesine elde ettiğimiz kelimeler yetersiz ve tam istenildiği gibi değildi.Bu yüzden RE Kütüphanesini kullanarak elde edilen kelimelerin başındaki ve sonunda anlamsız kelimeleri ayırarak daha isabetli ve istenildiği gibi kelimler önerilmesi sağlandı.Counter kütüphanesi sayesinde ayıklanan temiz kelimeler sayılıyor ve algoritmalara iletiliyordu.

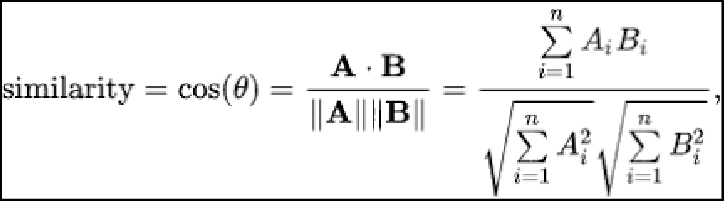




(Kaynak:11) (Kaynak:10)

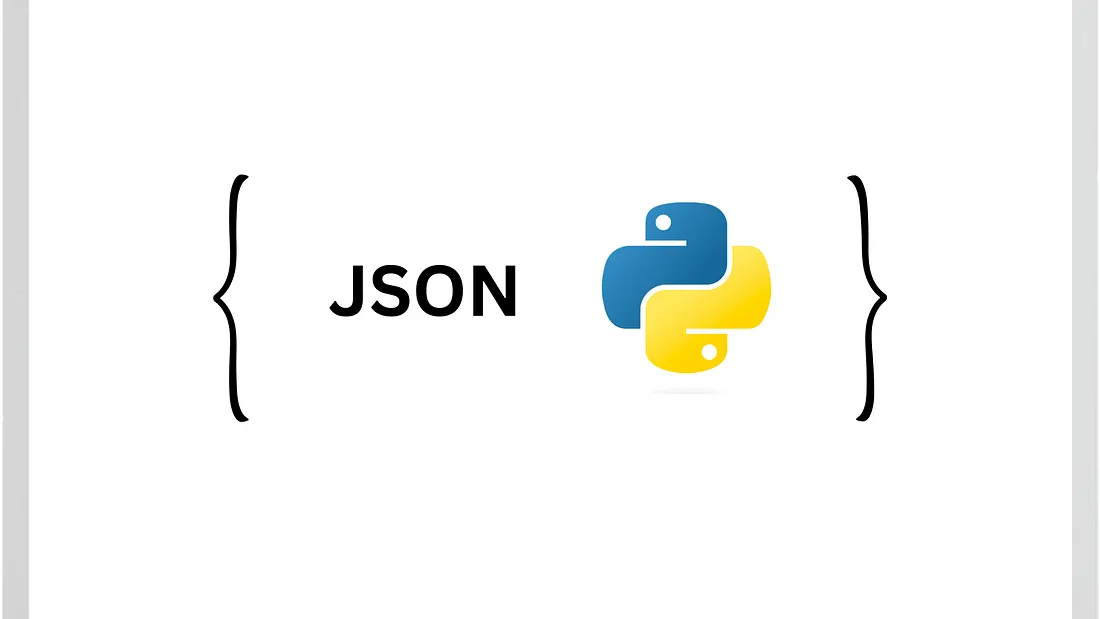
* 1. COSINE SIMILARITY ALGORiTMASI

TF-IDF metinleri vektöre çeviriyordu ama bize doğru kelime verme yüzde %45 civarındaydı.Bu yüzden TF-IDF’I desteklemesi için COSİNE SİMİLARITY Algoritmasını kullandık.Bu algoritma ise değerler arasındaki benzerliğe bakıyordu ama dosyadan veri çekmemize ragmen bu işlem uzun sürüyordu.



(Kaynak:9)

* 1. THREADPOOLEXECUTOR VE JSON KÜTÜPHANESİ

Tüm gerekli bilgiler elimizdeki dosyada bulunmadığından, appID’leri kullanarak SteamAPI üzerinden oyun bilgilerini çekmeye başladık. Bu işlem uzun sürdüğü için, ağı fazla yormamak ve geçici ban riskini azaltmak amacıyla **ThreadPoolExecutor kütüphanesini** kullanarak aynı anda 10 sorgu gerçekleştirdik. Elde edilen verileri, tekrar sorgulamamak adına .json uzantılı ayrı bir dosyada sakladık. Bu dosyayı okuyabilmek için de **json** kütüphanesini kullandık.

(Kaynak:7)

# 

# BEAUTİFULSOUP, TKİNTER VE THREADİNG KÜTÜPHANESİ

# Dosyada oyunların appID’leri ve açıklamaları yer alıyordu, ancak açıklamalar HTML kodlarıyla karışıktı. Bu nedenle, metni temizlemek için BeautifulSoup kullandık. Ardından bir kullanıcı arayüzüne ihtiyaç duyduğumuz için tkinter ile basit ama işlevsel bir arayüz oluşturduk. Arayüzde işlem sırasında donmalar yaşandığı için threading kütüphanesini ekledik ve böylece arayüz donmadan dosya okuma ve

# API işlemleri yapılabilir hale geldi.

(Kaynak:5) (Kaynak:6)

# 4.BULGULAR

# 4.1 SONUÇ BULGULARI

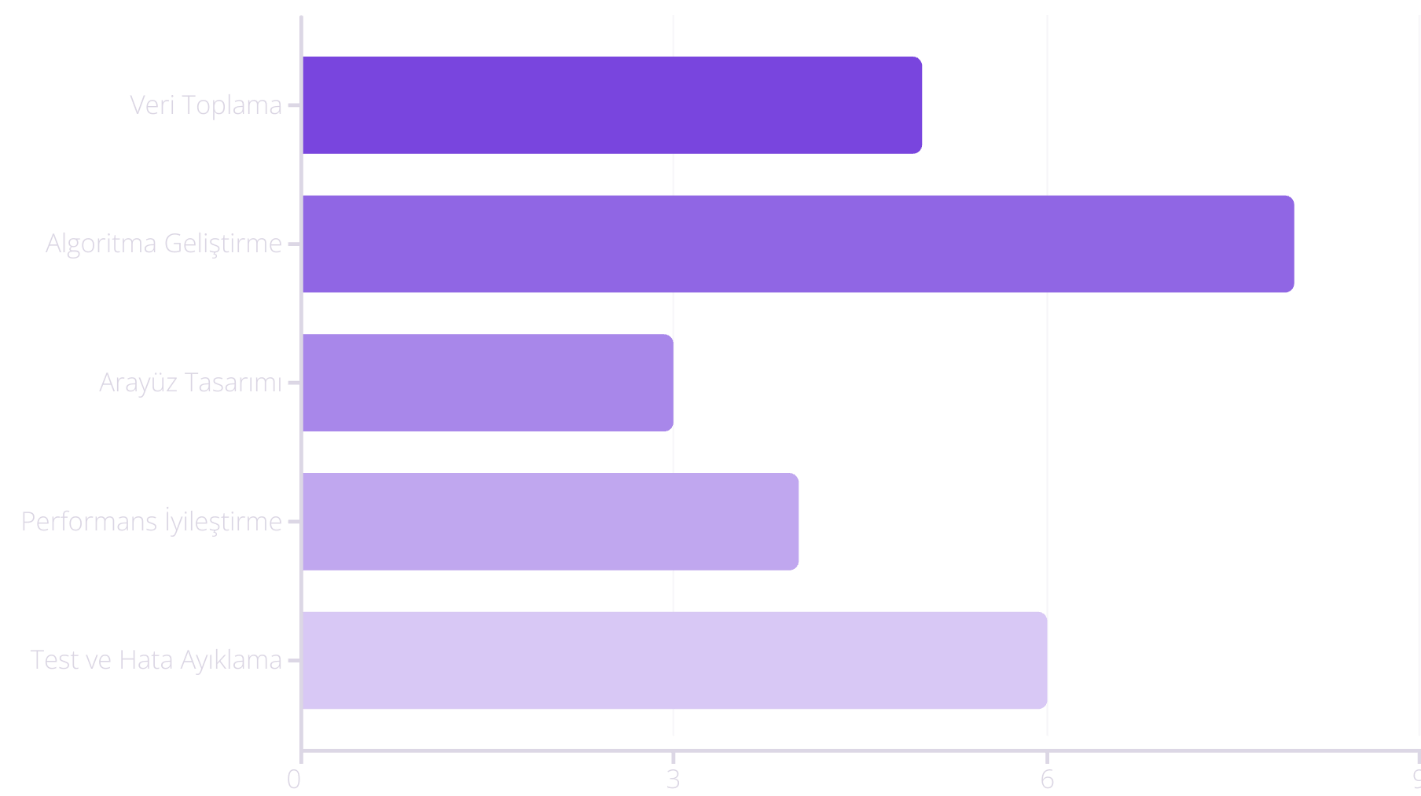
Veri seti çok büyük olduğu için Python bu veriyi okurken çok zaman harcıyordu. Bu yüzden veri, 10.000 satırlık parçalara bölünüp sonra tekrar birleştirildi. Bu işlem veriyi çok daha hızlı işler hale getirdi.

TF-IDF yöntemiyle elde edilen kelimeler yeterince isabetli değildi. RE kütüphanesiyle kelimelerin başındaki ve sonundaki gereksiz ifadeler temizlendi. Counter kütüphanesi sayesinde en sık geçen kelimeler sayıldı ve daha doğru sonuçlar elde edildi.

Cosine Similarity algoritması, metinler arasındaki benzerliği hesaplayarak önerilerin doğruluğunu artırdı. Ancak verinin boyutu büyük olduğundan bu işlemler biraz uzun sürdü.Steam API üzerinden veri çekmek zaman aldığı için ThreadPoolExecutor kütüphanesi ile aynı anda 10 sorgu yapılarak bu işlem hızlandırıldı. Çekilen veriler JSON dosyasında saklanarak tekrar veri çekme ihtiyacı ortadan kaldırıldı.

Oyun açıklamalarında HTML kodları bulunduğundan BeautifulSoup ile bu kodlar temizlendi. Kullanıcıya basit bir arayüz sunmak için tkinter kullanıldı. Arayüzün donmaması için threading kütüphanesiyle işlemler arka planda yürütüldü. Böylece sistem hem daha hızlı hem de daha kullanışlı hale geldi.

Sonuç olarak, sistem hem performans olarak hızlandı hem de önerdiği oyunlar daha anlamlı hale geldi. Gerek kullanılan kütüphaneler gerekse yapılan optimizasyonlar sayesinde proje istediğimiz seviyeye ulaştı.



# TARTIŞMA

# 

Projenin ilk aşamasında sadece oyun içeriklerinden anlamlı kelimeleri ayıklayarak öneri yapmaya çalıştık. Bu yöntemle alınan sonuçlar sınırlıydı ve önerilerin doğruluğu düşüktü. Vize sonrası yapılan geliştirmelerle hem öneri kalitesi hem de sistemin hızı önemli ölçüde artırıldı.

Büyük veri seti nedeniyle yaşanan okuma ve işleme sorunları, dosyanın parçalara ayrılması ve Pandas–OS kütüphaneleriyle çözülerek sistem performansı iyileştirildi. İlk olarak yalnızca TF-IDF algoritması kullanılsa da, RE ve Counter kütüphaneleriyle içerik temizliği yapılarak daha anlamlı kelimeler elde edildi. Cosine Similarity algoritması ile oyunlar arasında benzerlik oranı hesaplandı ve daha isabetli öneriler sunuldu.

Veri yetersizliğinde Steam API kullanıldı, fakat bu süreç yavaş ilerliyordu. ThreadPoolExecutor ile bu sorun aşıldı, JSON dosyaları ile veri tekrar kullanılabilir hale getirildi. HTML karışıklıkları BeautifulSoup ile giderildi. Kullanıcı arayüzü tkinter ile geliştirildi, threading ile donmalar önlendi. Son olarak, kullanıcıdan alınan geri bildirimlerin yapay zeka sistemin kendini geliştirmesi sağlandı.

# SONUÇ VE ÖNERİLER

Bu proje, başlangıçta yalnızca kelime analizine dayalı basit bir öneri sistemi iken, yapılan iyileştirmelerle çok daha gelişmiş bir oyun öneri sistemine dönüşmüştür. Uygulama artık bir oyunun açıklamasına ve içeriğine bakarak ona benzer oyunlar önerebilmekte, kullanıcıdan alınan geri bildirimlerle kendini geliştirme potansiyeline sahip bir yapı sunmaktadır.

Projenin başarısı, kullanılan farklı Python kütüphanelerinin doğru yerlerde ve doğru amaçlarla konfigüre edilmesiyle sağlanmıştır. Büyük veri setlerinin işlenmesi, metinlerin temizlenmesi, benzerlik hesaplamaları ve kullanıcı deneyimi gibi birçok konuda karşılaşılan problemler aşılmıştır.

Ancak öneri sisteminin hala geliştirilmesi gereken yönleri vardır. Önerilerin hazırlanması hâlâ yaklaşık 30-40 saniye sürmekte, bu da kullanıcı deneyimini olumsuz etkileyebilir. Bu yüzden:

* Steam API’ye olan bağımlılık azaltılmalı,
* Veriler daha önceden işlenmiş ara dosyalarda saklanarak öneriler hızlandırılmalı,
* Kullanıcıya daha spesifik sorular sorabilen bir chatbot sistemi entegre edilmeli,
* Sadece Steam değil, Epic Games, Xbox gibi platformlardaki oyunlarla fiyat ve içerik karşılaştırması yapılabilmeli.

Gelecekte yapılacak bu geliştirmeler sayesinde proje daha kararlı, hızlı ve kapsamlı bir öneri sistemine dönüşebilir.

# KAYNAKLAR

1. Doherty, S. M., Keebler, J. R., Davidson, S. S., Palmer, E. M., & Frederick, C. M. (2018). Recategorization of video game genres. *Proceedings of the Human Factors and Ergonomics Society Annual Meeting, 62*(1), 2099-2103. https://doi.org/10.1177/1541931218621473
2. ResearchGate. (2020). A Hybrid Recommender System for Steam Games. Retrieved from <https://www.researchgate.net/publication/340216044_A_Hybrid_Recommender_System_for_Steam_Games>
3. ArXiv. (2022). 2202.03392: Title. Retrieved from <https://ar5iv.labs.arxiv.org/html/2202.03392>
4. Real Python. (2021). Introduction to Python Threading. Retrieved from <https://realpython.com/intro-to-python-threading/>
5. Sinanerdinc. (2020). Python BeautifulSoup Modülü. Retrieved from <https://www.sinanerdinc.com/python-beautifulsoup-modulu>
6. Dizinbilgi. (2020). Tkinter Kütüphanesi Nedir?. Retrieved from [https://medium.com/@dizinbilgi/tkinter-kütüphanesi-nedir-15288f131450](https://medium.com/@dizinbilgi/tkinter-k%C3%BCt%C3%BCphanesi-nedir-15288f131450)
7. GitConnected. (2020). JSON Data Handling in Python Made Easy. Retrieved from <https://levelup.gitconnected.com/json-data-handling-in-python-made-easy-6876d17a79f>
8. GeeksforGeeks. (2021). How to Use ThreadPoolExecutor in Python3. Retrieved from <https://www.geeksforgeeks.org/how-to-use-threadpoolexecutor-in-python3/>
9. ResearchGate. (2021). Cosine Similarity Formula. Retrieved from <https://www.researchgate.net/figure/Cosine-similarity-formula_fig2_345471138>
10. GeeksforGeeks. (2020). Python Counter Objects: Elements. Retrieved from <https://www.geeksforgeeks.org/python-counter-objects-elements/>
11. Sinanerdinc. (2020). Python re Modülü. Retrieved from <https://www.sinanerdinc.com/python-re-modulu>
12. Aykut Uludağ. (2020). Python os Modülü. Retrieved from https://medium.com/@aykutuludag/python-os-modülüsü-1e6eced93069
13. Hatice Özbolat. (2021). Pandas Kütüphanesi: Veri Analizi ve Manipülasyonun Güçlü Aracı. Retrieved from https://haticeozbolat17.medium.com/pandas-kütüphanesi-veri-analizi-ve-manipülasyonun-güçlü-araçı-a7f883e828b8
14. Wikipedia. (2021). Valve logo. Retrieved from <https://tr.m.wikipedia.org/wiki/Dosya:Valve_logo.svg>