

دانشگاه اصفهان

دانشکده مهندسی کامپیوتر

گزارش پروژه‌ی پایانی درس جبرخطی کاربردی

سامانه توصیه‌گر با تجزیه ماتریس SVD

اعضای گروه:

علی‌اکبر احراری - ۴۰۰۳۶۱۳۰۰۱

مهرآذین مرزوق – ۴۰۰۳۶۱۳۰۵۵

# توابع مورد استفاده

## تابع CreateMatrix

def CreateMatrix():  
 user\_movie\_ratings = pd.merge(RatingsDf, MoviesDf, on='movieId')[['userId', 'movieId', 'rating']]  
 user\_movie\_ratings\_pivot = user\_movie\_ratings.pivot\_table(index='userId', columns='movieId', values='rating')  
 user\_movie\_ratings\_matrix = user\_movie\_ratings\_pivot.fillna(0).values  
  
 return user\_movie\_ratings\_matrix

عملکرد: ایجاد ماتریس ارزشیابی از اطلاعات فیلم ها و ارزشیابی ها.

مراحل:

* ادغام دو دیتافریم MovieDf و RatingsDf بر اساس شناسه فیلم (movieId).
* تبدیل دیتافریم ادغام شده به فرمت pivot table، که در آن سطرها نشان دهنده کاربران و ستون ها نشان دهنده فیلم ها هستند.
* پر کردن مقادیر خالی (NaN) در ماتریس با صفر.
* بازگرداندن ماتریس ارزشیابی ایجاد شده.

## تابع SVD

def SVD(S):  
 # Initialize matrices  
 m, n = S.shape # m = number of users, n = number of movies  
  
 # S transpose S  
 ST = np.transpose(S)  
 STS = np.matmul(ST, S)  
  
 # Calculate the eigenvalues and eigenvectors  
 eigenvalues, eigenvectors = np.linalg.eig(STS)  
  
 # Filter out negative eigenvalues and their corresponding eigenvectors  
 Landas = eigenvalues[eigenvalues >= 0]  
 V = eigenvectors[:, eigenvalues >= 0]  
 VT = np.transpose(V)  
 r = len(Landas)  
  
 sigmas = np.sqrt(Landas)  
 Sigma = np.diag(sigmas)  
  
 U = np.empty((m, r))  
  
 for i in range(r):  
 U[:, i] = np.matmul(S, V[:, i]) / sigmas[i]  
  
 return U, Sigma, VT, S

عملکرد: اجرای تجزیه ارزش منفرد روی ماتریس ارزشیابی.

مراحل:

* دریافت ماتریس ارزشیابی به عنوان ورودی.
* محاسبه ابعاد ماتریس (تعداد کاربران و تعداد فیلم ها).
* محاسبه ترانهاده ماتریس و ضرب آن در خودش (ST \* S).
* محاسبه مقادیر ویژه و بردارهای ویژه ماتریس حاصل.
* فیلتر کردن مقادیر ویژه و بردارهای ویژه منفی.
* محاسبه ماتریس های U، Sigma و V از مقادیر ویژه و بردارهای ویژه فیلتر شده.
* بازگرداندن ماتریس های U، Sigma، VT و ماتریس S اصلی.

## تابع cosine\_similarity

def cosine\_similarity(u, v):  
 dot\_product = np.dot(u, v)  
 norm\_u = np.linalg.norm(u)  
 norm\_v = np.linalg.norm(v)  
 return dot\_product / (norm\_u \* norm\_v)

عملکرد: محاسبه شباهت کسینوسی بین دو بردار.

مراحل:

* دریافت دو بردار به عنوان ورودی.
* محاسبه ضرب داخلی دو بردار.
* محاسبه نرم هر یک از بردارها.
* محاسبه شباهت کسینوسی با استفاده از فرمول: (ضرب داخلی بردارها) / (حاصلضرب نرم بردارها)
* بازگرداندن مقدار شباهت کسینوسی.

## تابع get\_recommendation

def get\_recommendations(user\_id):  
 UserMovieRating = CreateMatrix()  
 U, Sigma, VT, S = SVD(UserMovieRating)  
  
 # Get the user vector  
 user\_vector = S[user\_id]  
  
 # Calculate cosine similarities between the user vector and all movie vectors  
 movie\_similarities = np.apply\_along\_axis(cosine\_similarity, 1, VT, user\_vector)  
  
 # Sort movies by similarity scores and get the top N recommendations  
 N = 10 # Adjust the number of recommendations as needed  
 recommended\_movie\_indices = np.argsort(-movie\_similarities)[:N]  
 recommended\_movie\_ids = np.array(MoviesDf['movieId'])[recommended\_movie\_indices]  
  
 recommended\_movies = pd.DataFrame()  
 for i in range(10):  
 recommended\_movies = pd.concat([recommended\_movies,  
 MoviesDf.loc[recommended\_movie\_ids[i]].to\_frame().T], ignore\_index=True)  
  
 return recommended\_movies.drop('movieId', axis=1)

عملکرد: دریافت شناسه کاربر و ارائه توصیه های فیلم بر اساس علایق او.

مراحل:

* دریافت شناسه کاربر به عنوان ورودی.
* ایجاد ماتریس ارزشیابی با استفاده از تابع CreateMatrix.
* اجرای تجزیه ارزش منفرد روی ماتریس ارزشیابی با استفاده از تابع SVD.
* استخراج بردار کاربر از ماتریس S بر اساس شناسه کاربر داده شده.
* محاسبه شباهت کسینوسی بین بردار کاربر و بردارهای فیلم ها (ستون های ماتریس VT).
* مرتب سازی فیلم ها بر اساس میزان شباهت به صورت نزولی.
* انتخاب 10 فیلم با بیشترین شباهت به عنوان توصیه.
* ایجاد یک دیتافریم از اطلاعات فیلم های توصیه شده.
* بازگرداندن دیتافریم حاوی فیلم های توصیه شده.

## بخش اصلی کد

# Load data from CSV files  
MoviesDf = pd.read\_csv("movies.csv")  
RatingsDf = pd.read\_csv("ratings.csv")

# Get user ID as input  
UserId = int(input("Enter user ID: "))  
  
# Get recommendations for the user  
recommendations = get\_recommendations(UserId)  
  
print("Recommended movies for user", UserId, ":")  
print(recommendations)

مراحل:

* دریافت شناسه کاربر از کاربر.
* دریافت توصیه های فیلم برای کاربر با استفاده از تابع get\_recommendations.
* چاپ لیست فیلم های توصیه شده برای کاربر.

# کتابخانه‌های استفاده شده

NumPy

یک کتابخانه قدرتمند برای محاسبات علمی و عددی در پایتون است.

امکان کار با آرایه ها و ماتریس های چند بعدی را فراهم می کند.

شامل توابعی برای انجام عملیات جبر خطی، آماری و ریاضی است.

در این کد، برای ایجاد و دستکاری ماتریس ها، محاسبه تجزیه ارزش منفرد (SVD) و شباهت کسینوسی استفاده می شود.

Pandas

یک کتابخانه برای دستکاری و تحلیل داده ها در پایتون است.

امکان خواندن و نوشتن داده ها از فایل های مختلف مانند CSV و Excel را فراهم می کند.

ساختارهای داده ای مانند DataFrame و Series را ارائه می دهد که برای مدیریت داده های جدولی بسیار مناسب هستند.

در این کد، برای خواندن داده های فیلم ها و ارزشیابی ها از فایل های CSV، ایجاد ماتریس ارزشیابی و دستکاری داده های مربوط به فیلم های توصیه شده استفاده می شود.

# منابع

<https://naomy-gomes.medium.com/the-cosine-similarity-and-its-use-in-recommendation-systems-cb2ebd811ce1>

Google Bard Ai

جزوه درس