영화 추천 프로그램-코드 설명

12194114 구현규

1.입력 받은 dataid를 통해 사용자의 직업과 같은 사람들이 가장 좋아하는 무비 10개 추천

sql0 = "SELECT occupation FROM user WHERE userid="+str(num1)+";"  
cur.execute(sql0)  
rows = cur.fetchall()[0][0]

**----rows에 데이터베이스를 통한 직업 데이터 받기.**

sql2 = "select movieTitle,avg(ratingScore) as ratingScore

from user natural join ratings natural join movie

**----user, movie, rating natural join을 한다.**

where occupation='"+str(rows)+

**----아까 받은 rows의 직업 데이터를 이용하여 해당 유저의 직업을 가진 데이터로 한정**

"' group by movieid

**----movie id로 묶고**

order by ratingScore desc,movieId desc limit 10;"

**----평점순 10개만 출력**

cur.execute(sql2)  
result=cur.fetchall()

**----result에 저장**

2.입력 받은 dataid를 통해 사용자의 나이대(10’s, 20’s, 30’s, 40’s, 50’s, …)와 같은 사람들의 가장 좋아하는 무비 10개 추천

sql3="SELECT age FROM user WHERE userid="+str(num1)+";"  
cur.execute(sql3)  
age = int(cur.fetchall()[0][0])

**----age에 데이터베이스를 통한 나이 데이터 받기.**

min\_age=math.floor(age/10)\*10)

**----1의 자리 버림을 통한 (10’s, 20’s, 30’s, 40’s, 50’s, …)결정**  
max\_age=min\_age+10

**----+10 (ex)10-20같은 범위 결정**

sql4="select movieTitle,avg(ratingScore) as ratingScore

from user natural join ratings natural join movie

**----user, movie, rating natural join을 한다.**

where "+str(min\_age)+" <=age and age < "+str(max\_age)+"

**----자신의 나이대에 속하는 나이대의 데이터만 남긴다**

"' group by movieid

**----movie id로 묶고**

order by ratingScore desc,movieId desc limit 10;"

**----평점순 10개만 출력**  
cur.execute(sql4)  
result2=cur.fetchall()

**----result2에 저장**

3. knn 추천 알고리즘(K=40, 피어슨 상관계수, 유저 기반)

sql = "select userid,movieId,movieTitle,ratingScore from user natural join ratings natural join movie"  
cur.execute(sql)  
rows = cur.fetchall()  
moviedata = pd.DataFrame(data=rows, columns=['userid', 'movieId', 'movieTitle', 'ratingScore'])  
moviedata2 = pd.pivot\_table(moviedata, index='userid', columns='movieId', values='ratingScore', aggfunc='max')

**----moviedata2아래와 같이 인덱스는 userid, columns는 movieid인 데이터프레임 생성한다. 안의 값은 유저가 해당 movieid에 평가한 'ratingScore' 데이터**

**스크린샷, 사각형이(가) 표시된 사진

자동 생성된 설명**

**----다른사람과 유사도 검사를 통해 영화 추천을 받는 유저(A)를 userid로 넣는다**  
  
def cosim(userid, dataframe):  
 movies = []  
 for i in dataframe.loc[userid, :].index:  
 if math.isnan(dataframe.loc[userid, i]) == False:  
 movies.append(i)

**----movies에 유사도 검사유저가 본 영화 저장**  
 U\_df = pd.DataFrame(dataframe.loc[userid, movies]).T

**----U\_df 유저가 유저가 본 영화에 대한 평점을 데이터 프레임에 저장**  
 other\_df = dataframe.loc[:, movies].drop(userid, axis=0)

**----other\_df에는 다른 모든 유저가 유저가 본 영화에 대한 평점을 데이터 프레임에 저장**  
 sim\_dict = {}  
 for user in other\_df.index:

**----다른 유저 집합중 한 유저 선택(B)**  
 u\_list = []  
 df\_list = []  
 ch1 = 0  
 pa1 = 0  
 pa2 = 0  
 sm = [m for m in U\_df.columns if math.isnan(other\_df.loc[user, m]) ==

False]

**----다른 유저(B)가 유저(A)가 본 영화를 전부 보지 못 했기 때문에 NaN이 아닌 영화만 sm에 저장**  
 for i in U\_df.loc[userid, sm].values:  
 u\_list.append(i)

**----** **sm에 있는 영화(유저(A)가 본 영화)에 대한 유저(A)의 평가를 u\_list에 순서대로 저장**  
 for i in other\_df.loc[user, sm].values:  
 df\_list.append(i)

**----** **sm에 있는 영화(유저(A)가 본 영화)에 대한 다른 유저(B)의 평가를 df\_list에 순서대로 저장**  
 for i in range(len(sm)):  
 ch1 += (u\_list[i] \* df\_list[i])  
 pa1 += u\_list[i] \* u\_list[i]  
 pa2 += df\_list[i] \* df\_list[i]  
 ch2 = sum(u\_list) \* sum(df\_list)  
 pa3 = (pa1 - sum(u\_list) \* sum(u\_list) / (len(sm) + 0.000000001)) \* (  
 pa2 - sum(df\_list) \* sum(df\_list) / (len(sm) + 0.000000001))  
 result = (ch1 - ch2 / (len(sm) + 0.000000001)) / ((pow(pa3, 1 / 2) + 0.000000001))

**----** **피어슨 상관 계수(아래 우측 이미지)를 좌측 같이 변형하여 계산하는 과정, 분모가 0이 되는 것을 고려하여 0.000000001추가**

텍스트, 폰트, 라인, 화이트이(가) 표시된 사진

자동 생성된 설명  
 sim\_dict[user] = result

**----sim\_dict에 다른 유저(B)의 userid와 해당 유저와 유저(A)와의 상관 계수를 딕셔너리로 저장**  
 ''''''  
 sim\_mat = sorted(sim\_dict.items(), key=operator.itemgetter(1), reverse=True)[:40]

**----sim\_mat에는 모든 다른 유저와의 상관 계수 중 k=40임으로 상위 40명을 저장**

**----knn계산과정----**  
 recommend\_list = list(set(dataframe.columns) - set(U\_df.columns))

**---- 유저(A)가 안본 영화 리스트를 recommend\_list 저장**   
 others\_k = [i[0] for i in sim\_mat]

**---- 유사도 높은 40명의 userid를 저장**  
 recommender = dict()  
 for movie in recommend\_list:

**---- 영화 하나를 결정**  
 rating = []  
 sim = []  
 for person in others\_k:  
 if math.isnan(dataframe.loc[person, movie]) == False:  
 rating.append(dataframe.loc[person, movie])  
 sim.append(sim\_dict[person])

**---- 다른 유저 한 명씩 불러오고 위에서 결정된 영화를 해당 유저가 봤다면 영화 평점과 유사도 데이터를 순서대로 rating,sim에 저장**  
  
 aa = len(rating)  
 use\_sum = 0  
 for i in range(aa):  
 use\_sum += sim[i] \* rating[i]  
 pred = use\_sum / (sum(sim) + 0.0000001)  
 recommender[movie] = pred

**----아래 계산대로 유저(A)가 안 본 영화의 예상 평점을 recommender dict에 저장**

텍스트, 폰트, 라인, 화이트이(가) 표시된 사진

자동 생성된 설명  
  
 return sorted(recommender.items(), key=operator.itemgetter(1), reverse=True)[:10]

**---- recommender dict에서 평점 높은 10개 영화 출력**

result\_list=cosim(int(num1),moviedata2)

**----맨위에서 입력 받은useid를 기반으로 추천 상위 10개 movieid를 result\_list에 저장**  
movielist=[]   
for i in range(10):  
 sql7="select movieid, movietitle

from movie where movieid="+str(result\_list[i][0])+"

order by movieId desc limit 1;"  
 cur.execute(sql7)  
 movie = cur.fetchall()  
 removie=movie[0]  
 movielist.append((removie[0],removie[1]))

**----추천 상위 10개 영화movieid를 통해 movietitle을 찾고 momovielist에 같이 저장**

4. 사용 언어: 파이썬

-파이썬 플라스크 라이브러리로 설계

-복잡한 프로그램이 아니고 시간 문제상 최대한 간단한 구조로 설계

-아래와 같이 매우 간단하 구조

-주 작동원리는 temp에 있는 html에 "GET", "POST" 작동이 있을 경우

routes.py로가 알맞은 동작 수행 후 리턴 값 반납

텍스트, 스크린샷, 소프트웨어, 멀티미디어 소프트웨어이(가) 표시된 사진

자동 생성된 설명

-실행 화면-

텍스트, 스크린샷, 콜라주, 그래픽 디자인이(가) 표시된 사진

자동 생성된 설명

-영화 이미지는 고정 이미지(default함수를 찾을 수가 없어 고정 이미지 사용)

-id입력시 해당 아이디에 맞는 3가지 값만 텍스트로 출력

-넷플릭스 로고 옆의 모든 버튼은 다시 메인(다른 페이지 없음)으로 돌아오는 것으로 구현