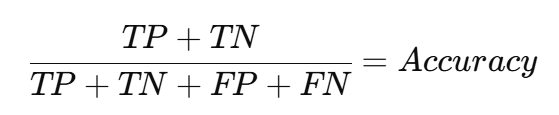
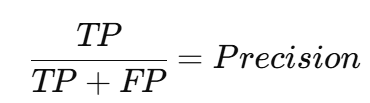
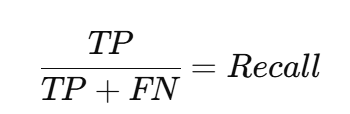
**DATA INTEGRITY - Q4**

Yotam ben moshe

שאלה 1 - מה תהיה ההשפעה של records null על המדדים recall ,precision ,accuracy?

נפרק לחלקים:

1. Accuracy: מדד מדויקת כוללת את כל הנכונים (חיוביים ושליליים). אם נוסיף הרבה null records (כלומר עוד TN) , הACCURACY יעלהֵ.
2. Precision: מתייחס רק לתוצאות חיוביות שחזינו. null records לא נכנסות ל־TP או FP (רק TN), ולכן לא משפיעות ישירות.
3. Recall: מתייחס ליכולת לזהות חיוביים אמיתיים (כלומר מבין כל מה סיווג נכון את החיוביים). אם המודל חוששֵֵ ֵלתת תחזיות חיוביות (כי רוב הנתונים הם שליליים), הוא מפספס הרבה TP ולכן יש הרבה FN. במקרה זה המכנה גדל ללא שינוי כמעט במונה ולכן הRECALL קטן. 

**שאלה 2** - בנו מודל בעל recall גבוה על בסיס -

במאי משותף:ֵ

עבור כל במאי, נחפש את כל זוגות הסרטים שהוא ביים.

אין סינון לפי סרט קלט - כל הזוגות האפשריים.

זהו חיבור **ישיר וברור** בין סרטים - ולכן **recall גבוה** .

drop table if exists gs\_common\_director;

create table gs\_common\_director as

(SELECT

md1.movie\_id AS movie\_id\_source,

md2.movie\_id AS movie\_id\_target,

'director' AS reason

FROM

gs\_movies\_directors as md1

JOIN

gs\_movies\_directors as md2

ON

md1.director\_id = md2.director\_id

WHERE

md1.movie\_id != md2.movie\_id

GROUP BY

md1.movie\_id, md2.movie\_id, reason);

שחקן משותף:

סרטים בהם מופיע אותו שחקן.  
שחקן לרוב שומר על סגנון משחק ולכן זה קישור אפשרי.ֵ  
recall גבוה - יוצר הרבה זוגות (גם אם לא כולם איכותיים), אך עדיין ההמלצה תלויה מאוד במי השחקן.

drop table if exists gs\_common\_actor;

create table gs\_common\_actor as

(SELECT DISTINCT

r1.movie\_id AS movie\_id\_source,

r2.movie\_id AS movie\_id\_target,

'actor' AS reason

FROM

gs\_roles as r1

JOIN

gs\_roles as r2

ON

r1.actor\_id = r2.actor\_id

WHERE

r1.movie\_id != r2.movie\_id

GROUP BY

r1.movie\_id, r2.movie\_id, reason);

ז'אנר משותף:

סרטים בז'אנר משותף, סביר שיהיו רלוונטיים לאותו קהל.  
קל ליישום.ֵ

תורם רבות ל-recall.

drop table if exists gs\_common\_genre;

create table gs\_common\_genre as

(SELECT

mg1.movie\_id AS movie\_id\_source,

mg2.movie\_id AS movie\_id\_target,

'genre' AS reason

FROM

gs\_movies\_genres mg1

JOIN

gs\_movies\_genres mg2

ON

mg1.genre = mg2.genre

WHERE

mg1.movie\_id != mg2.movie\_id

GROUP BY

mg1.movie\_id, mg2.movie\_id, reason);

תפקיד משותף:ֵ

תפקידים חוזרים מרמזים על נושאים דומים.

פחות חזק מבמאי/ז'אנר, אך מועיל כשאין מידע אחר.

recall גבוה מכיוון שלא הרבה יסווגו כFN, אך precision נמוך יחסית כי סיווג על פי תפקיד יכול לגרום להרבה צימודים לא נכונים להסתווג כנכונים. ניתן לראות את ההשפעות בשאלה מספר 1.

drop table if exists gs\_common\_role;

create table gs\_common\_role as

(SELECT

r1.movie\_id AS movie\_id\_source,

r2.movie\_id AS movie\_id\_target,

'role' AS reason

FROM

gs\_roles as r1

JOIN

gs\_roles as r2

ON

r1.role = r2.role

WHERE

r1.movie\_id != r2.movie\_id

GROUP BY

r1.movie\_id, r2.movie\_id, reason);

שנה קרובה (3+-):

סרטים שיצאו באותו פרק זמן, סביר שהם שייכים לאופנת סרטים מסוימת.

לא קשר חזק, אך מועיל בהעדר מידע אחר.ֵ

drop table if exists gs\_close\_years;

create table gs\_close\_years as

(SELECT DISTINCT

m1.id AS movie\_id\_source,

m2.id AS movie\_id\_target,

'year' AS reason

FROM

gs\_movies m1

JOIN

gs\_movies m2

ON

ABS(m1.year - m2.year) <= 3

WHERE

m1.id != m2.id

GROUP BY

m1.id, m2.id, reason);

Collaborative Filtering

לקחתי את הטבלה שיש במטלת הבונוס שמייצגת דירוגים לפי משתמשים ונגדיר "אהבו" כ - recommendation >= 6.

**drop table if exists personal\_movies\_ranking;**

create table personal\_movies\_ranking

(movie\_id int,

recommendation int not null,

suggested\_by varchar(255) not null,

justification varchar(255) not null,

comment varchar(255),

PRIMARY KEY (movie\_id, suggested\_by),

CONSTRAINT CHK\_personal\_recommendation CHECK (recommendation >=1 AND 10 >= recommendation),

CONSTRAINT CHK\_personal\_justification CHECK (length(justification) >=10),

FOREIGN KEY (movie\_id) REFERENCES movies(Id)

);

Recall גבוה מכיוון כל זוג סרטים שאהב אותו משתמש נכנס לרשימה, לוכד קשרים סמויים שלא מתגלים רק לפי תוכן הסרטים.

drop table if exists gs\_high\_rating;

create table gs\_high\_rating as

(SELECT

l1.movie\_id AS movie\_id\_source,

l2.movie\_id AS movie\_id\_target,

'Collaborative Filtering - Like' AS reason

FROM

(SELECT

suggested\_by,

movie\_id

FROM

personal\_movies\_ranking

WHERE

recommendation >= 6) as l1

JOIN

(SELECT

suggested\_by ,

movie\_id

FROM

personal\_movies\_ranking

WHERE

recommendation >= 6) as l2

ON

l1.suggested\_by = l2.suggested\_by

WHERE

l1.movie\_id != l2.movie\_id);

אותו קוד רק ללא סינון לפי ציון - אלה רק לפי אותו בן אדם

drop table if exists gs\_same\_person;

create table gs\_same\_person as

(SELECT

l1.movie\_id AS movie\_id\_source,

l2.movie\_id AS movie\_id\_target,

'Collaborative Filtering - same person' AS reason

FROM

(SELECT

suggested\_by,

movie\_id

FROM

personal\_movies\_ranking) as l1

JOIN

(SELECT

suggested\_by ,

movie\_id

FROM

personal\_movies\_ranking) as l2

ON

l1.suggested\_by = l2.suggested\_by

WHERE

l1.movie\_id != l2.movie\_id);

**שאלה 3 -** אחדו את כל המודלים לטבלה אחת

drop table if exists high\_recall\_recommendations;

CREATE table high\_recall\_recommendations

(

movie\_id\_source INT,

movie\_id\_target INT,

reason varchar(50),  
PRIMARY KEY (movie\_id\_source,movie\_id\_target,reason)

);

insert into high\_recall\_recommendations

select \*

from

gs\_common\_director;

insert into high\_recall\_recommendations

select \*

from

gs\_common\_actor;

insert into high\_recall\_recommendations

select \*

from

gs\_common\_genre ;

insert into high\_recall\_recommendations

select \*

from

gs\_common\_role ;

insert into high\_recall\_recommendations

select \*

from

gs\_close\_years ;

insert into high\_recall\_recommendations

select \*

from

gs\_high\_rating ;

insert into high\_recall\_recommendations

select \*

from

gs\_same\_person ;

טבלה נוספת שבנויה מטבלת כל ההמלצות - טבלה זו מכילה רק ערך UNIQUE של ההמלצות ללא כפילויות, עם אגרגציה על הממליץ.

drop table if exists high\_recall\_recommendations\_agg;

create table high\_recall\_recommendations\_agg as

select

movie\_id\_source,

movie\_id\_target,

group\_concat(distinct reason separator ', ') as reasons

from high\_recall\_recommendations

group by movie\_id\_source, movie\_id\_target;

שאלה 4 - מצאוֵ את כל ההמלצות שהן NULL RECORD

select gs.\*

from

movies\_recommendations as gs

left join

high\_recall\_recommendations\_agg as high\_recall\_model

on

gs.base\_movie\_id = high\_recall\_model.movie\_id\_source

and

gs.recommended\_movie\_id = high\_recall\_model.movie\_id\_target

where

gs.recommendation <=5 # negative recommendations

and

high\_recall\_model.movie\_id\_source is null; # No model recommended on them

הסבר -

אם NR בעצם מייצג TN כלומר כל ההמלצות שסווגו רעות והן באמת רעות אזי כל ההמלצות שקיבלו בטבלת הGS דירוג 5 ומטה הן TN - המלצות רעות שסווגו כרעות. עושים LEFT JOIN לטבלה עם כל ההמלצות שיש להן RECALL גבוה כדי למצוא את כל ההמלצות הטובות מבין כל ההמלצות

**movies\_recommendations** - מייצג את כל ההמלצות (GS)

**high\_recall\_recommendations**  - כל ההמלצות מהמודלים עם RECALL גבוה.

שאלה 5 - האם יש ממליציםֵֵ הנוטים במיוחד להמליץ על NULL RECORD

##Q5

SELECT

suggested\_by,

COUNT(\*) AS count\_null\_records

FROM

(select gs.\*

from

movies\_recommendations as gs

left join

high\_recall\_recommendations\_agg as high\_recall\_model

on

gs.base\_movie\_id = high\_recall\_model.movie\_id\_source

and

gs.recommended\_movie\_id = high\_recall\_model.movie\_id\_target

where

gs.recommendation <=5

and

high\_recall\_model.movie\_id\_source is null) as null\_records

group by

suggested\_by

having

count\_null\_records >= 20

ORDER BY

count\_null\_records DESC;

הסבר -

בשאילתא בסעיף 4 מצאנו את כל הNR (הסבר נמצא בסעיף) כלומר השאילתה מחזירה המלצות רעות שסווגו כרעות. אני משתמש בשאילתה של סעיף 4 כ - Subquery, ביצעתי GROUP לפי מי שהמליץ, וספרתי כמה פעמים כל ממליץ המליץ המלצה שהיא TN. הוספתי תנאי לכמות המלצות של TN כדי לבודד את הממליצים שיש להם נטיה להמליץ המלצות כאלה.