DATA INTEGRITY - Q4

Yotam ben moshe

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

1. Accuracy: מדד מדויקת כוללת את כל הנכונים (חיוביים ושליליים). אם נוסיף הרבה null records (כלומר עוד TN) , הACCURACY יעלה.

$$\frac{TP + TN}{TP + TN + FP + FN} = Accuracy$$

2. Precision: מתייחס רק לתוצאות חיוביות שחזינו. null records לא נכנסות ל־TP או FP (רק TN), ולכן לא משפיעות ישירות.

$$\frac{TP}{TP + FP} = Precision$$

3. Recall: מתייחס ליכולת לזהות חיוביים אמיתיים (כלומר מבין כל מה סיווג נכון את החיוביים). אם המודל חוששֶ 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.

$$\frac{TP}{TP + FN} = Recall$$

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 - מצאוֿ 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 ממליצים 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.