고급 프로젝트 최종 보고서

10대 익명 투표 서비스 개선 및 신규 서비스 개발

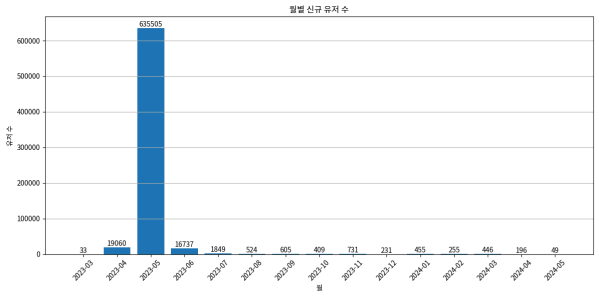
박하진, 여승윤, 오정은, 최지원B

1. 데이터 EDA

1-1 유저 가입/출석/탈퇴 : `accounts_user` , `accounts_attendance` , `accounts_userwithdraw`

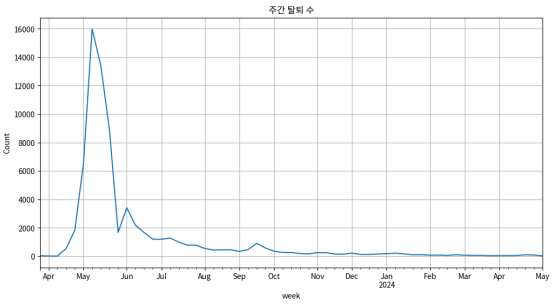
월별 가입자 수

2023년 03월 29일 ~ 2024년 05월 09일 집계



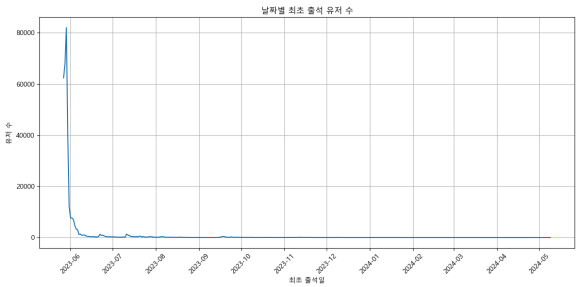
주간 탈퇴 유저 수

2023년 03월 29일 ~ 2024년 05월 09일 집계



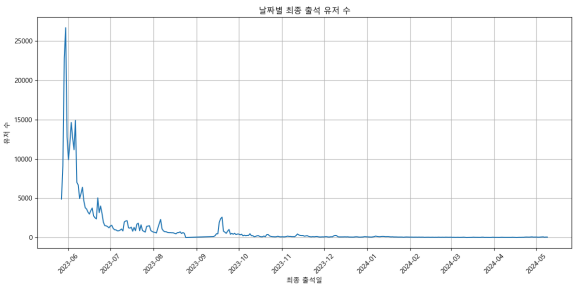
최초 출석 일자별 유저 수

2023년 05월 27일 ~ 2024년 05월 09일 집계



최종 출석 일자별 유저 수

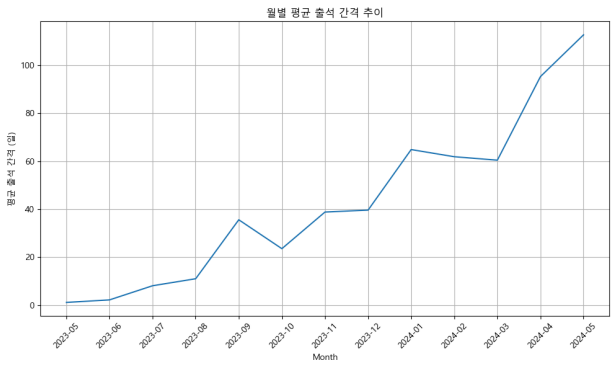
2023년 05월 27일 ~ 2024년 05월 09일 집계



월별 출석 간격 추이

인사이트 요약

- 가입, 탈퇴 시기에 큰 차이가 없고, 최초 출석일, 최종 출석일 분포 역시 유사한 시기에



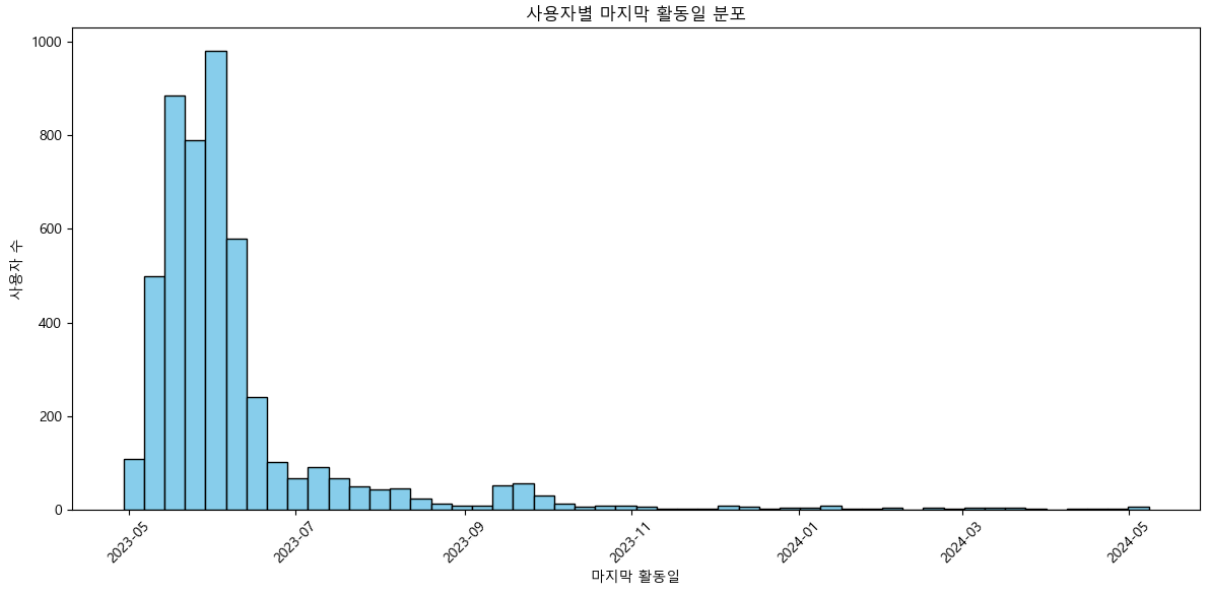
이뤄진 것으로 보아 초기 이탈율이 큰 것으로 보인다.

- 23년 3월 말 서비스 런칭 후 23년 6월부터 급격히 가입, 출석 유저가 감소한다. 약 2달 만에 서비스 이용량이 급감하는 것으로 보인다.
- 월별 평균 출석 간격은 점점 증가하는 것으로 보아, 서비스의 지속성에 문제가 있을 것으로 보인다.

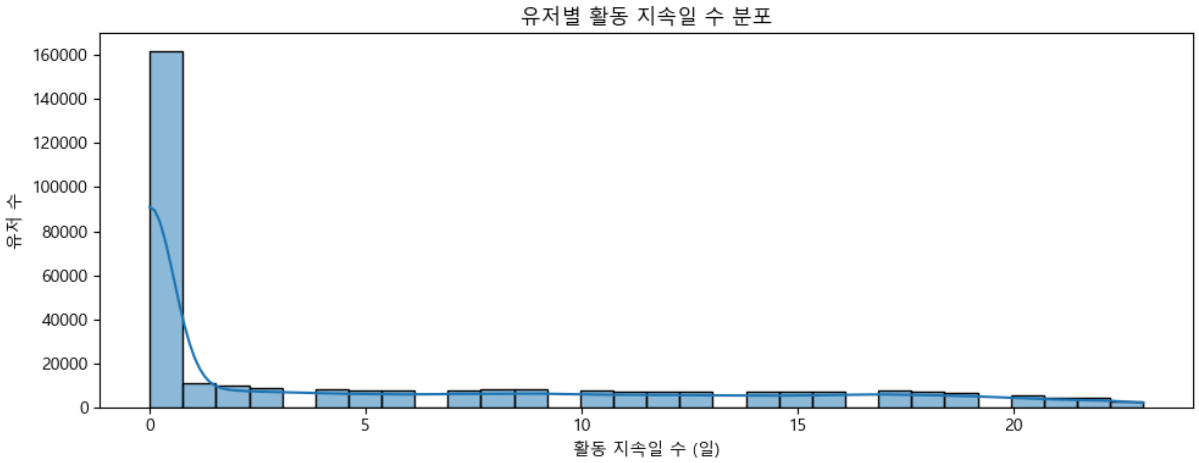
1-2 유저 활동 : accounts_userquestionrecord , hackle_events , hackle_properties

사용자가 실제로 질문에 어떻게 반응했는지, 참여했는지 여부, 읽었는지 여부, 신고를 얼마나 받았는지 등 서비스 내 핵심 액션인 "투표 활동"과 직접 관련된 행동 이력 확인

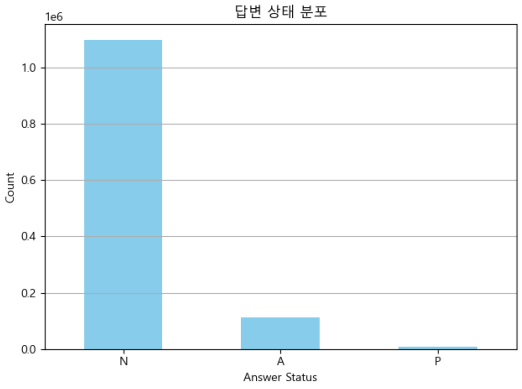
- 사용자 별 마지막 활동(투표)일 분포 그래프



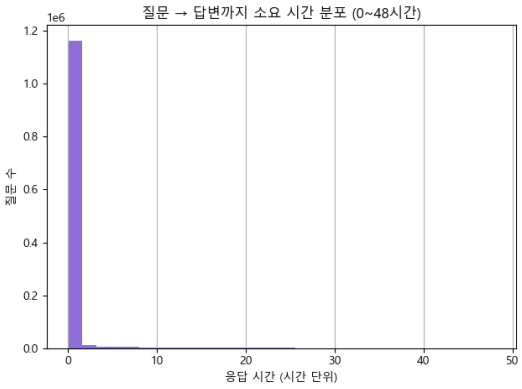
- 사용자 별 활동 지속일 수 분포 그래프



- 답변 상태 분포도 확인 : 투표에 대한 답변이 잘 이루어진 것 같진 않아 보임.



- 투표 후 답변까지 소요 시간 분포 확인 : 대부분 즉각적으로 답변

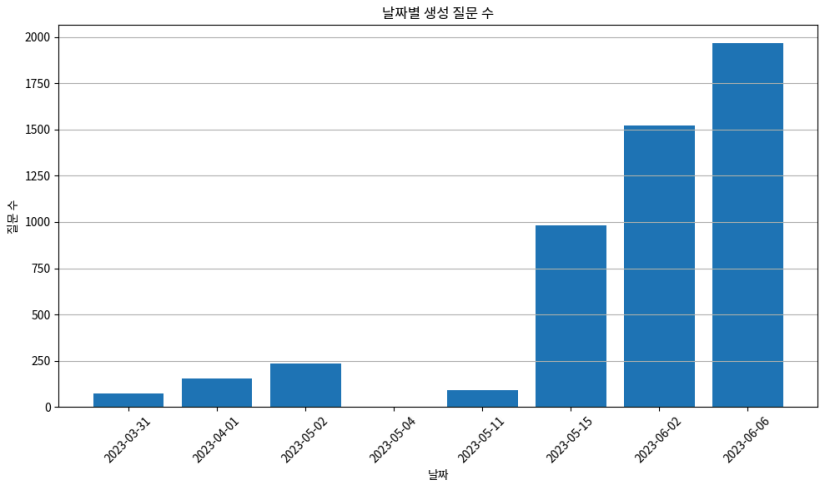


인사이트 요약

- 메인 서비스인 투표 활동의 지속성 역시 낮은 것으로 나타나 서비스의 지속성 부족을 문제로 정의하였으며, 구체적인 분석을 진행하였다.
- 투표에 대한 답변이 이뤄지는 경우, 즉각적으로 응답하지만, 답변 활동 자체가 매우 저조하다. 즉, 유저 간의 상호작용이 매우 저조하다.

1-3 질문 컨텐츠 : polls_question , polls_questionset

- 질문 id 수는 총 5025개, 텍스트 중복을 제외한 질문은 총 3899개이다.
- 질문은 2023.03.31 ~ 2023.06.06까지만 총 8회에 거쳐 생성되었다.



- 유저별로 하루 평균 참여 세트 수는 2.9개이다.

> 인사이트 요약

- 세트 당 질문의 수가 10개로, 하루 평균 3세트에 참여한다면 질문이 약 130일(약 4개월)만에 모두 소진된다.
- 서비스 런칭 후 약 2개월만에 사용량이 급감한 것으로 보아, 컨텐츠 부족의 문제가 있을 것으로 보인다.
- 컨텐츠 소진 속도가 빨라 질문 반복 노출로 인한 컨텐츠 피로감이 유발되었다.

2. 문제 정의 및 원인 파악

> EDA 결과, 익명 투표 **서비스의 지속성이 매우 낮은 것으로 파악**되었다.
> 주요 원인으로는
> **컨텐츠 부족**이 있었으며, 추가적인 원인을 파악하기 위해 세부 분석을 진행하였다.

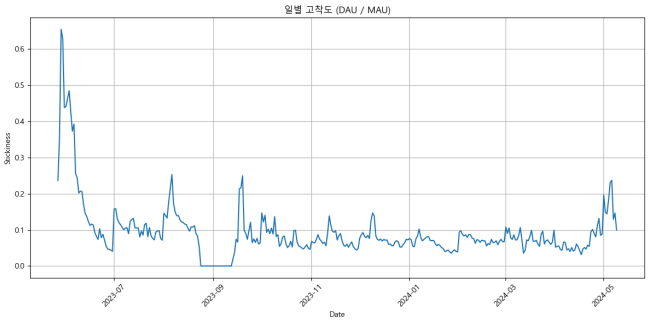
2-1 고착도 분석: `accounts_attendance` , `accounts_userquestionrecord`

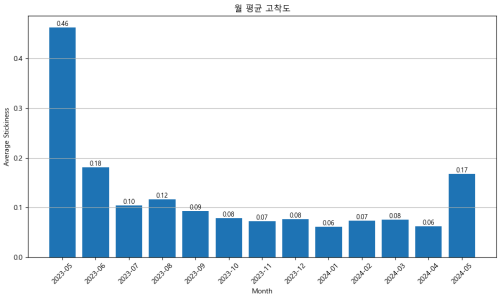
고착도(DAU/MAU)란?

지표	의미
DAU (Daily Active Users)	하루 동안 앱을 사용한 고유 유저 수
MAU (Monthly Active Users)	한 달 동안 앱을 사용한 고유 유저 수
고착도(DAU/MAU 비율)	한 달 동안 이용한 유저들 중 하루에 실제로 이용하는 비율 (높을수록 충성도 ↑)

`accounts_attendance` **: 2023.05.27 ~ 2024.05.09 출석 데이터**

고착도 시각화(일별/월별)



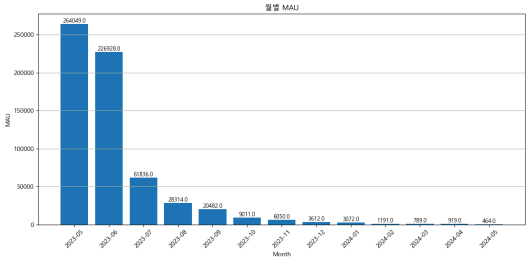


수치 해석

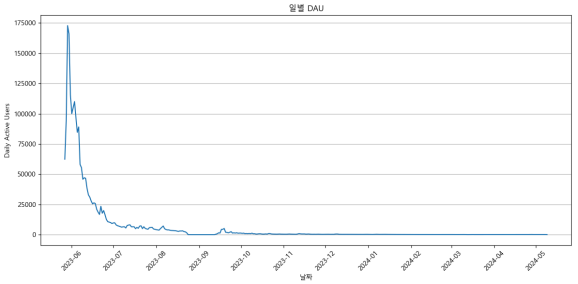
월	DAU/MAU	해석
2023-05	**0.46**	런칭 초기, 테스트 또는 프로모션 등으로 활발한 사용
2023-06 ~ 2023-08	0.10 ~ 0.18	급격한 활용도 감소
2023-09 ~ 2024-04	**0.06 ~ 0.09**	매우 낮은 충성도, 대부분의 사용자가 앱을 떠남
2024-05	**0.17**	**약간 회복세**, 하지만 여전히 매우 낮음.

고착도 하락 원인 보조 지표

- MAU 하락



- DAU 하락



- MAU, DAU 모두 급격하게 하락한 것은 전체 사용자의 관심도가 감소했음을 의미하며, **DAU의 하락이 더욱 급격하여 고착도까지 하락**하였다.

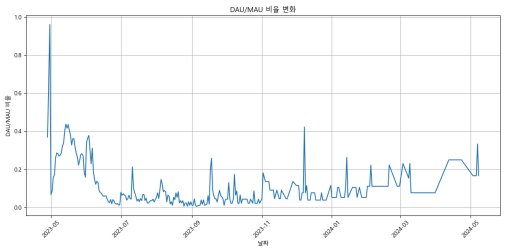
`accounts_userquestionrecord` : 2023.04.28 ~ 2024.05.08 투표 기록 테이블

- 메인 서비스인 투표 기록 데이터를 통해 고착도를 계산하였다.

고착도 시각화

수치 해석

월	DAU/MAU	해석
2023-04	**0.662**	런칭 초기, 테스트 또는 프로모션 등으로



월	DAU/MAU	해석
		활발한 사용
2023-05	0.285	어느 정도 안정적인 관심 유지
2023-06 ~ 2024-01	**0.05 ~ 0.13**	점점 **활용도가 낮아짐**, 대부분 사용자가 앱을 떠남
2024-02 ~ 2024-05	**0.13 → 0.20**	**약간 회복세**, 하지만 여전히 낮음

- 출석 데이터와 고착도 변화 양상이 유사하다.

인사이트 요약

- DAU, MAU가 모두 하락하여 고착도가 하락하였다. 이는 자주 오는 핵심 유저가 이탈했음을 의미한다.
- 서비스 초창기(23.04 ~ 23.05) 유저들이 대거 유입되었다가 서비스의 지속성의 부족으로 두 달 내에 대량 이탈하였다. 남은 유저들 역시 충성도를 유지하지 못했고 유저들의 평균 출석 간격이 100일을 초과하며 매우 저조한 이용량을 보이고 있다.

2-2 유입 Cohort 분석: `accounts_user` , `accounts_userquestionrecord`

- Cohort 분석을 통해 **월별 가입자의 잔존율** 확인, 어떤 시점에 유입된 사용자들이 더 오래 남아 있었는지, 혹은 빠르게 이탈했는지를 파악

Cohort 분석 준비

우선 `accounts_user` 테이블에서 **가입일**과 **활동일** 데이터 활용

- `joined_at` : 사용자의 가입 일자
- `activity_date` : 사용자의 실제 활동 일자 (질문 열람, 응답, 신고 등에서 파생 가능)

Cohort 분석을 위한 테이블 매핑

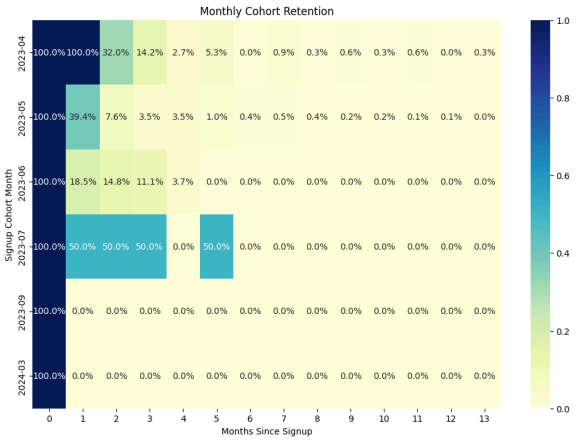
분석용 컬럼	출처 테이블	컬럼명	설명
가입일 (Cohort 기준)	`accounts_user`	`created_at`	사용자 최초 가입일
활동일 (잔존 판단 기준)	`accounts_userquestionrecord`	`created_at`	사용자가 투표에 참여한 시점 (서비스 내 실질적 활동)

분석 흐름 요약

1. `accounts_user`에서 사용자별 **가입 월**을 계산 → `cohort_month`
2. `accounts_userquestionrecord`에서 사용자별 **활동 월**을 계산 → `activity_month`

3. 가입 월과 활동 월의 차이를 계산해서 → `cohort_index`
4. `cohort_month` 기준으로 `cohort_index`에 따른 사용자 수 카운팅
5. Heatmap 형태로 시각화

시각화 및 해석



숫자의 의미

각 셀에 있는 숫자는 **해당 cohort의 가입자 중 몇 %가 해당 월에도 활동했는지**를 나타냄

2023-04 cohort

- `cohort_index = 0` : **1.0 (100%)**
  - → 2023년 4월에 가입한 사용자 중, 4월에 활동한 사용자비율은 100%
- `cohort_index = 1` : **1.0 (100%)**
  - → 가입한 다음 달(5월)에도 활동한 비율이 100%
- `cohort_index = 2` : **0.3195**
  - → 가입 2개월 뒤인 6월에는 약 32%가 활동
- `cohort_index = 3` : **0.142**
  - → 가입 3개월 뒤에는 **14.2%**
- 이후는 점점 감소 (장기 잔존율은 낮음)

2023-05 cohort

- `cohort_index = 0` : 1.0 (5월 활동자 전부)
- `cohort_index = 1` : 0.3935 → 39.4%
- `cohort_index = 2` : 0.0756 → 7.6%
- `cohort_index = 3` : 0.0353 → 3.5%
- 이후 계속 감소

인사이트 요약

- **2023-04 cohort는 비정상적으로 높은 2개월차 retention이 있었다가 빠르게 감소한다.**

  → `1.0 → 1.0 → 0.32 → 0.14 ...`
- **2023-05 cohort 이후부터는 점점 retention이 낮아졌다.**

  → 즉, 서비스 지속 사용률이 점점 떨어지고 있다. 초기 2달의 활동량은 높은 것을 보면 유저들의 서비스에 대한 흥미가 떨어진 것으로 보인다.
- **2023-06, 07, 09, 24-03 cohort 등도 마찬가지로 단기 이탈이 심하다.**

2-3 유저 활동율 분석 : `accounts_userquestionrecord`

분석 전 사용자 확인

항목	
`accounts_user` 기준 고유 사용자 수	**677,085명**
`accounts_userquestionrecord` 에서 활동했지만 `accounts_user` 에 등록되지 않은 사용자 수	**0명**
마지막 가입 일자 (`accounts_user`)	**2024-05-09**
마지막 활동 일자 (`accounts_userquestionrecord`)	**2024-05-08**

해석

1. 가입자와 활동자 데이터가 정확히 연결된다.

- 활동한 사용자는 모두 `accounts_user` 테이블에 존재하는 사용자이다.
- 즉, **비회원 활동**이나 **탈퇴 후 활동 기록이 남는 문제**는 현재 데이터에선 없다.

2. 활동 데이터는 가입 데이터보다 1일 앞에 종료된다.

- 가입자는 2024-05-09까지 존재하지만, 활동 기록은 2024-05-08까지밖에 없다.
- 이는 **로그 수집 지연**, 또는 **일별 데이터 집계 시점 차이**로 자연스러운 현상이다.

가입자 수 및 활동자 수

- **`accounts_user`의 고유 user ID 수:** `677,085명` → 총 가입자 수
- **`accounts_userquestionrecord`의 고유 user ID 수:** `4,849명` → 질문 기록이 있는 실제 활동자 수

이 수치가 의미하는 것 → 활동률 (Activation Rate) 계산

activation_rate = 4849 / 677085 ≈ 0.0072 → 약 0.72%

> ### 인사이트 요약
>
> - 전체 가입자 중 **질문을 열어보거나 응답을 기록한 사용자는 0.7%에 불과하다.**
> - **99% 이상은 가입만 하고, 서비스의 핵심 기능을 사용하지 않는다.**

→ 즉, **온보딩 문제**, **초기 진입 장벽**, 혹은 **서비스 매력도 부족** 등의 문제가 존재할 수 있다.

- **가입 대비 활동률이 1%도 되지 않는다**는 건 서비스 지속성뿐 아니라 **초기 활성화(activation)** 단계부터 문제가 있을 것이라고 가정할 수 있다.

2-4 투표 세트 완주율 분석: `accounts_userquestionrecord` , `polls_questionset`

- 10개의 질문으로 구성된 한 세트 전체에 모두 투표하는 비율이 얼마나 되는지 확인하고자 한다.

전처리 방식

`accounts_userquestionrecord` 테이블

- 불필요한 컬럼 제거 : `created_at` , `question_id` , `user_id` , `question_piece_id` 제외한 컬럼 제거
- 유저 별, 시간 순 정렬

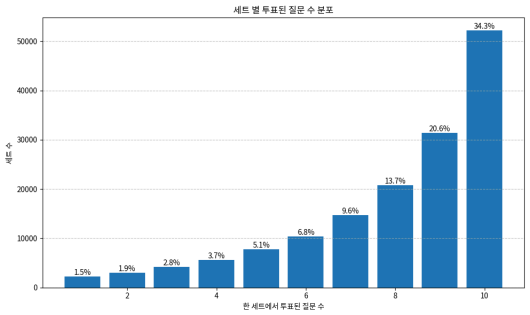
`polls_questionset` 테이블

- 컬럼명 변경: `id` → `set_id`
- 불필요한 컬럼 제거: `opening_time` , `status` 제거
- `question_piece_id_list` 컬럼 정규화: 컬럼명 `question_piece_id`로 설정

→ `user_id`, `question_piece_id` 컬럼 기준으로 `accounts_userquestionrecord 테이블에 머지했다.

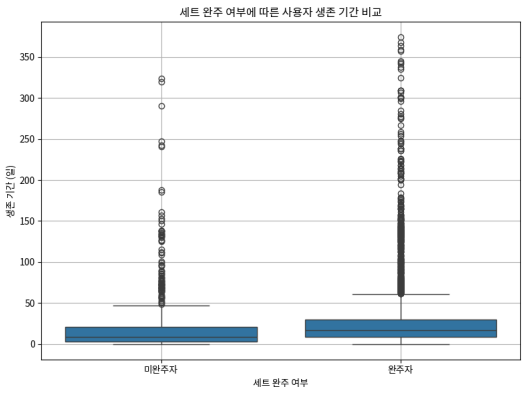
분석 결과

- 세트 별 투표된 질문 수의 분포



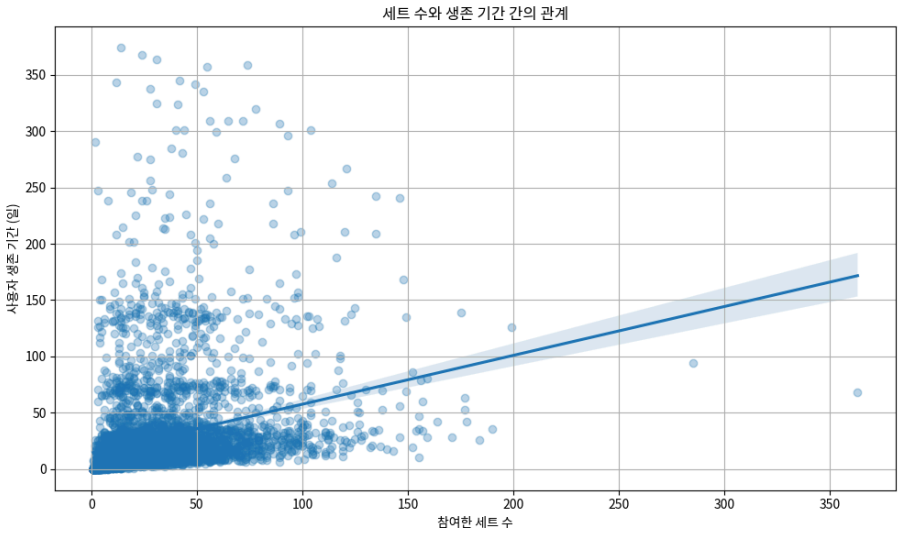
  - 10개 질문에 모두 투표한 경우: 34.3% = **세트 완주율**
  - 평균 투표 질문 갯수 = 7.9개

- 세트 완주 여부에 따른 평균 생존 기간



  - 세트 미완주자 평균 생존 기간 = **18.8일**
  - 세트 완주자 평균 쟁존 기간 = **30.3일**

- 참여한 세트 수와 생존 기간 간의 상관관계



- 상관계수가 **0.28**로 약하지만, 참여한 세트 수와 생존 기간 사이에는 상관관계가 있다.

> **인사이트 요약**

- 신규 컨텐츠가 없는 상태에서 세트당 질문 수는 총 10개로 완주율이 34.3%에 불과하였다.
- 대부분의 유저가 세트에 전부 답변하지 않으며, 세트 미완주자의 생존기간은 완주자에 약 12일 가량 짧은 것으로 보아 세트 완주에 부담을 느끼는 것으로 보인다.
- 투표 참여한 세트의 수가 많을수록 생존 기간이 길다.

3. 문제 원인 분석

3-1 서비스 지속성 하락의 원인 요약

EDA와 데이터 분석 결과, 서비스 지속성 부족의 원인을 다음 3가지로 요약할 수 있다.

- 서비스 흥미도 하락
- 유저 간의 상호작용 부족
- 컨텐츠 부족 및 반복 노출로 인한 피로감 상승

3-2 서비스 지속성 하락 원인 분석 로드맵

1. 사용자 그룹 나누기 (행동 기반 세분화)

> 사용자들이 어떤 방식으로 서비스를 사용하는지를 파악

2. 초기 활성화 단계 분석

> 특정 시점에 유입은 되었지만 빠르게 이탈한 사용자가 많다면
>
> 서비스 초기경험(온보딩)에 문제가 있을 수 있음

3. 행동 패턴/기능 이용 패턴 기반 분석

> 지속성이 낮은 사용자들은 서비스의 어떤 기능을 안 쓰는가?를 파악

4. 이탈자 비이탈자 비교 분석

> 이탈 유저가 떠나기 전에 어떤 행동을 했는지, 그리고 잔존 유저와 어떤 차이가 있었는지를 파악

5. 이탈률 로지스틱 회귀 분석

> 어떤 특성이 이탈 확률(churn probability)에 어떻게 영향을 주는지 분석

3-3 분석 진행 및 결과 해석

1. 사용자 유형 세분화 및 결과

조건	그룹 명	수 (명)	비율 (%)
활동일 수 하루	단발성 사용자	151	3.1%
활동일 수 2일 이상 7일 이하	간헐적 사용자	1644	33.9%
활동일 수 8일 이상	지속 사용자	3,054	63%

인사이트 요약

- 전체 사용자 중 63%가 서비스에 안정적으로 정착한 '지속 사용자'인 것으로 확인된다.
  반면, 약 37%의 사용자는 단발적이거나 간헐적으로만 서비스를 이용한 뒤 더 이상 돌아오지 않는다.
- 왜 이렇게 많은 사용자가 지속 사용자로 정착하지 못했는가? → 초기 경험에 문제가 있었을 가능성이 있다.

2. 초기 활성화 단계 분석

초기 활성화 단계 분석을 통해 왜 사용자들이 금방 떠나는지에 대한 요인 분석 및 해석 결과

지표	설명	결과	해석
가입 후 첫 활동까지 걸린 평균 시간	가입 경험에 문제가 있었는지 확인	**0.63일** (약 15시간)	대부분의 사용자가 **가입 후 하루 이내에 첫 활동 시작** → **온보딩 경험은 크게 문제 없어 보임**
가입 후 평균 활동 지속 일수	얼마나 오래 활동했는지	**28.29일**	평균적으로 한 달 정도는 활동한 사용자들이 많다는 뜻
가입 후 3일 이내 활동 사용자 비율	초기 이탈자 비율 파악	**0.09 (9%)**	**단 9%만이 첫 3일 안에 활동을 마무리**한다는 뜻. 대부분의 사용자는 3일 넘게 활동을 지속
단 1회 활동 후 이탈한 사용자 비율	진입 후 바로 떠난 사용자 파악	**0.03 (3%)**	**단 한 번 활동하고 이탈하는 사용자는 적다**는 뜻. 즉, 사용자들이 서비스에 진입은 잘하고 일정 시간은 머무는 편

인사이트 요약

- 대부분의 사용자는 **온보딩에 성공하고 일정 기간은 활동**한다
- 그런데 왜 DAU/MAU는 낮을까?

→ 이 말은 결국 사용자**들이 "지속적으로 자주" 돌아오지 않는다는 것**이 핵심 문제이다.

3. 기능별 이용 패턴 분석

- 지속성이 낮은 사용자들이 어떤 기능을 **잘 안 쓰거나**, 어떤 행동을 **자주 했는지**를 파악해서 서비스 지속성 저하의 **행동적 요인**을 분석
  - 보기만 하고 참여 안하는 사용자의 비율 확인
  - 신고 행동이 많은 사용자의 지속성 확인
  - 응답을 잘 받는 사용자 vs 받지 못한 사용자의 지속성 비교

1) 응답 받은 사용자가 거의 없음

> 응답 받은 사용자 수: 461명
>
> 응답 못 받은 사용자 수: **4388명**

  - **대부분의 사용자가 질문에 대한 응답을 한 번도 못 받은 상황.**
  - `answered_count`, `response_rate`가 0인 사용자 비율이 90%이상
  - 사용자 입장에서 보면, 아무리 질문을 올려도 **"피드백이 없음"으로, 이는 서비스 지속성 하락의 치명적인 원인이다.**

2) 기능별 행동 패턴의 양극화

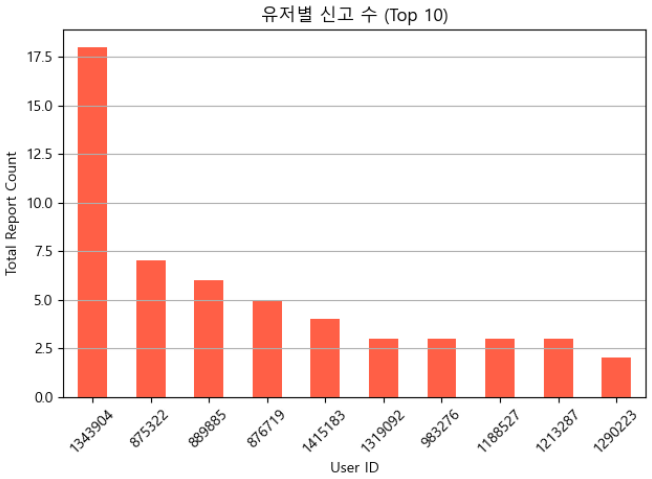
사용자 유형	읽기만 함 (read_only_count 평균)	신고	응답
단발성 사용자	3.9	0.0066	0.1
간헐적 사용자	20.5	0.003	0.1
지속 사용자	**127.4**	**0.0487**	0.09

  - 대부분의 사용자는 **보기만 하고 아무 행동도 하지 않음**
  - **응답은 아예 없고**, 일부 사용자는 **신고**만 진행
  - 지속 사용자조차도 참여 없이 **일방적인 열람 + 신고** 행동만 반복

3) 서비스 구조상 비대칭성

- **질문만 올라오고 응답은 없음** → 사용자 입장에서 허탈 + 피로
- **일방향 콘텐츠 소비 구조** → 사용자 간 상호작용이 거의 없음
- 일부 사용자가 **신고에만 치중** → 부정적 경험 증가 가능성

  ex) 사용자 별 신고 수 top 10



결론: 지속성 저하의 핵심 요인은 유저간의 상호작용 부족이다.

> 응답을 받지 못하는 구조적인 문제 + 일방적 열람과 신고 중심의 사용 패턴

4. 이탈자 비이탈자 비교 분석

- 목표: 특정 시점에 왜 사용자가 떠났는지를 정량/정성적으로 파악

이탈자가 떠나기 전에 어떤 행동을 했는지, 그리고 비이탈자**와 어떤 차이가 있었는지**를 파악

1.
이탈자 정의 - 가입 후 7일 이내, 3일 이상 미참여 : 가입 직후 관심 없으면 이탈로 간주

2.
비교 분석을 위한 주요 행동 변수

변수명	설명
vote_cnt	해당 기간 동안 총 투표 수
unique_target_cnt	투표한 대상자 수
emotion_change_cnt	감정 추론 값 변화 횟수
positive_emotion_ratio	긍정 감정 비율
same_target_vote_ratio	같은 대상에게 반복 투표한 비율
question_diversity_score	참여한 질문의 주제 다양도
daily_activity_span	활동한 일수 (ex. 3일 중 2일 활동)
last_vote_gap	마지막 투표 이후 지난 일수

이탈자와 비이탈자 비율 파악

```
user_df = account_user.copy()
activity_df = accounts_userquestionrecord.copy()

user_df['signup_date'] = pd.to_datetime(user_df['created_at'])
activity_df['activity_date'] = pd.to_datetime(activity_df['created_at'])

merged_df = activity_df.merge(user_df[['id', 'signup_date']], left_on='user_id', right_on='id',
merged_df['days_since_signup'] = (merged_df['activity_date'] - merged_df['signup_date']).

early_activity = merged_df[merged_df['days_since_signup'].between(0, 6)]

early_activity['activity_day'] = early_activity['activity_date'].dt.date
active_days = early_activity.groupby('user_id')['activity_day'].nunique().reset_index()
active_days.columns = ['user_id', 'active_days_in_7days']

active_days['is_churned'] = active_days['active_days_in_7days'].apply(lambda x: 1 if x < 3 e

user_churn_df = user_df[['id', 'signup_date']].rename(columns={'id': 'user_id'})
user_churn_df = user_churn_df.merge(active_days, on='user_id', how='left')
user_churn_df['active_days_in_7days'] = user_churn_df['active_days_in_7days'].fillna(0)
user_churn_df['is_churned'] = user_churn_df['is_churned'].fillna(1).astype(int)

print(user_churn_df[['user_id', 'signup_date', 'active_days_in_7days', 'is_churned']].head())
```

변수 생성 로직을 통해 is_churned 컬럼을 만들고 `is_churned == True` 와 `False` 유저를 나눈 후 **가입 후 7일 이내 활동 일수 분포**를 통해 **이탈자와 비이탈자 비율 파악**

결과

active_days_in_7days	count
0.0	672328
7.0	1109
8.0	924
6.0	818
5.0	678
4.0	506
3.0	312
1.0	207
2.0	203

해석

- `0.0` **일 활동 (672,328명)**: 아예 참여하지 않은 사용자 — 전형적인 이탈자
- `1~2일` **활동**: 관심은 있었지만 꾸준한 참여 X → 이탈자 기준에 포함
- `3일 이상` **활동**: 어느 정도는 서비스에 관심 보인 사용자 → **비이탈자**

이탈자 특성 분석

1. `account_user` **테이블: 기본 사용자 프로필 정보**

컬럼명	설명
id	사용자 ID
gender	성별
point	포인트 수치
friend_id_list	친구 목록 (리스트 형태)
is_push_on	푸시 알림 설정 여부
report_count	신고 횟수
group_id	소속 그룹 ID

2. 이탈 여부 (`user_churn_df`) : 가입 후 7일 이내 3일 이상 참여한 적 없는 사용자 = 이탈자(1), 그렇지 않으면 비이탈자(0)

1. 이탈자 vs 잔존자 특성 비교 (포인트 이상치 제거 후)

항목	이탈자 (Churned)	잔존자 (Retained)	요약
is_push_on (푸시 알림 설정)	0.8429	0.8572	잔존자가 푸시 알림을 약간 더 켜둠 (푸시 설정 비율이 높을수록 잔존 가능성 높음)
point (포인트)	1,540	1,668	잔존자가 평균적으로 약간 더 많은 포인트를 가짐 (활동성 반영 가능)
report_count (신고 수)	0.0362	0.0508	잔존자가 신고를 조금 더 많이 함 (서비스 적극적 사용 신호로 해석 가능)
friend_count (친구 수)	53.1	62.8	**잔존자가 친구 수가 평균적으로 약 10명 더 많음 (사회적 연결성 높을수록 잔존)**
group_id (가입 그룹)	평균 37,107	평균 25,760	그룹 자체는 큰 직접적 영향은 없어 보임 (별다른 차이 없음)

해석

- **푸시 알림을 켜두고, 포인트가 더 많고, 친구 수가 더 많고, 신고도 약간 더 많이 하는** 사용자들이 **잔존 확률이 높음**.

→ 즉, "앱에 적극적으로 관여하는 활동성 높은 사용자"일수록 이탈을 덜 한다는 결론

5. 이탈률 로지스틱 회귀 분석

어떤 특성이 이탈 확률에 어떻게 영향을 주는지 분석 진행

- 각 특성(feature)이 이탈에 미치는 영향(방향과 크기)을 정량적으로 확인

사용된 주요 피처 & 회귀 결과 요약

Feature	설명
is_push_on	푸시 알림 설정 여부
avg_opened_times	질문당 평균 조회 횟수
friend_count	친구 수
report_count_x	신고한 횟수
report_count_y	신고당한 횟수
answered_count	응답한 질문 수
total_actions	전체 서비스 내 활동 수
read_only_count	읽기만 한 질문 수
group_id	소속 학급 ID
read_count	읽은 질문 수

Feature	Coefficient	해석
is_push_on	**−0.3605**	푸시 알림이 켜져 있는 유저는 이탈 확률이 **상대적으로 낮음**. 알림이 사용자 리텐션에 도움이 됨.
friend_count	**−0.2696**	친구 수가 많을수록 이탈 확률이 낮음. **사회적 연결**이 유지에 긍정적 영향.
avg_opened_times	**−0.2567**	질문을 평균적으로 더 많이 열어본 유저는 이탈 확률이 낮음 → **관심도나 호기심이 높음**.
report_count_x	**+0.1585**	유저가 신고를 많이 할수록 이탈 확률이 높음 → **불만이 많거나 부정적 경험 가능성**.
total_actions	**−0.1337**	총 활동량이 많을수록 이탈 확률 낮음. **활동량 자체가 리텐션을 높이는 요인**.
report_count_y	**+0.1332**	다른 피처에서의 신고 수 (ex. 신고 받은 횟수일 가능성)도 많을수록 이탈 확률 ↑. **부정적 상호작용 경험**.
group_id	**+0.1135**	특정 그룹/학급에 따라 이탈 확률 차이 존재. 예: **비활성 그룹에 속한 유저일 가능성**.
read_only_count	**+0.0729**	읽기만 하고 응답하지 않는 유저는 이탈 확률이 조금 높음. **수동적 소비자 유형**.
read_count	**+0.0571**	콘텐츠를 많이 읽더라도, **응답이나 참여가 없다면 유지에 큰 도움은 안 됨**.
answered_count	**−0.0150**	응답 수가 많을수록 아주 약간 이탈 확률이 낮지만, **영향은 미미함**.

인사이트 요약

- **이탈 확률을 낮추는 주요 요인**

  → is_push_on, friend_count, avg_opened_times, total_actions

  → 알림 유지 + 친구 네트워크 + 관심도 + 활동량

- **이탈 확률을 높이는 주요 요인**

  → report_count_x, report_count_y, read_only_count, group_id

  → 부정 경험, 수동적 활동, 그룹 특성

Feature	분류	이유	계수
total_actions	실제 행동	사용자가 서비스를 얼마나 많이 이용했는지를 보여주는 전체 활동량이므로 행동 지표	-0.1337
avg_opened_times	실제 행동	질문을 열어본 횟수의 평균으로, 콘텐츠 소비 빈도라는 사용 패턴을 의미	-0.2567
read_only_count	실제 행동	읽기만 하고 응답하지 않은 행동 패턴. 수동적 사용 경향	0.0729
read_count	실제 행동	전체 읽은 횟수. 콘텐츠 소비 중심의 행동 지표	0.0571
answered_count	실제 행동	질문에 대한 응답 횟수. 유저의 참여 행동 지표	-0.015
is_push_on	경험적 요인	알림 설정 여부는 서비스와의 관계 유지/재방문 유도 경험에 가까움	**-0.3605**
friend_count	관계적 요인	친구 수는 사회적 연결, 유저 간 관계를 나타냄	**-0.2696**
report_count_x	경험적 요인	신고 많이 한 유저 → 부정적 사용자 경험 가능성	0.1585
report_count_y	경험적 요인	신고 받은 유저일 가능성 → 부정적인 상호작용 경험	0.1332
group_id	관계적 요인	속한 그룹의 특성에 따라 서비스 유지율이 달라짐 → 소속감, 커뮤니티 특성 등 관계 요인	0.1135

- 단순히 많이 사용했는지(행동량)보다도 **어떤 경험을 했고, 어떤 관계를 맺었는지가 이탈 여부에 더 결정적이기에 경험과 관계를 긍정적으로 할 수 있도록 기존 서비스 개선 또는 신규 서비스 개발이 필요**

3 - 4 유료 결제(초성 확인 서비스) 요인 분석

- 초성을 확인한 record와 확인하지 않은 record 간의 클래스 **불균형**이 심하다.

확인	미확인
59,153	1,128,039

→ 즉, 현재 서비스의 가장 기본적인 수익 모델(초성 확인)의 **수익성이 낮음**

> 따라서, 결제율을 예측하는 것보다는 **어떤 피쳐가 결제율에 유의미한 영향을 끼쳤는지 확인하고자 개발 진행하였다.**

모델 피쳐 정의

- 타겟 변수: `accounts_userquestionrecord` 테이블의 'status' 컬럼
  - C,B: 초성 확인 X
  - I: 초성 확인 O

피쳐	테이블	컬럼	완료	메모
사용 기간	accounts_user	2024-05-09 기준에서 created_at 날짜 뺀 값	O	
성별	accounts_user	gender	O	

피쳐	테이블	컬럼	완료	메모
차단 횟수	accounts_user	block_user_id_list	O	
차단 당한 횟수	accounts_user	block_user_id_list에 나온 아이디 카운트	O	
숨김 횟수	accounts_user	hide_user_id_list	O	
숨김 당한 횟수	accounts_user	hide_user_id_list에 나온 아이디 카운트	O	
신고 횟수	accounts_user	report_count	O	
친구 요청 횟수 대비 '수락' 비율	accounts_friendrequest	status (A)	O	
친구 요청 횟수 대비 '대기' 비율	accounts_friendrequest	status (P)	O	
친구 요청 횟수 대비 '거절' 비율	accounts_friendrequest	status (R)	X	의미 없어서 제거
친구 수	accounts_user	friend_id_list에서 id 개수 세서 friend_count 컬럼 추가	O	
같은 학급 유저 수	accounts_user	group_id	O	
투표한 횟수	accounts_userquestionrecord	user_id	O	
투표 받은 횟수	accounts_userquestionrecord	chosen_user_id	O	
받은 질문 호감 지수	polls_question + accounts_userquestionrecord 조인	Gemini API로 태깅	O	

질문 호감 지수 자동 태깅 (Gemini API)

- 프롬프트
  - 다음은 어떤 사람이 이성에게 특정 질문을 했을 때, 그 질문이 얼마나 이성적인 관심(호감)을 나타내는지를 평가하는 작업입니다.
  - 각 질문에 대해 호감 점수(0~5점)를 매겨주세요:
    - **0점: 단순 정보/우정 기반, 이성적 감정 없음**
    - **1~2점: 아주 미묘한 관심만 느껴짐**
    - **3~4점: 약간의 감정 또는 암시가 있음**
    - **5점: 명확한 이성적 관심 또는 감정 표현**
- 태깅 결과

id	question_text	hogam_score
99	가장 신비한 매력이 있는 사람은?	4
101	미래의 틱톡커는?	1
102	여기서 제일 특이한 친구는?	1
103	가장 지켜주고 싶은 사람은?	5
104	내 어깨를 내어줄 수 있는 사람은?	4
...	...	...
4556	제일 파격적일 것 같은 사람은?	1
4557	고마운 기억이 있는 사람은?	3
4558	발로 글씨를 제일 잘 쓸것 같은 사람	0

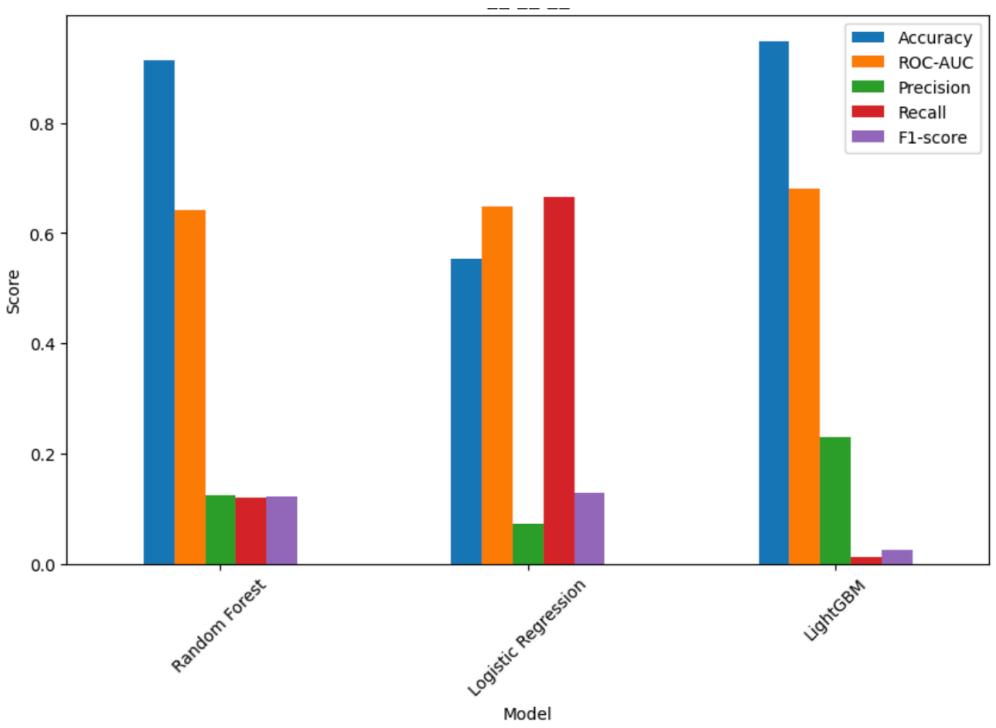
모델 학습 결과(1)

- 성능 평가 지표 선정
  - `Accracy` : 전체 중 맞게 예측한 비율
  - `ROC-AUC Score` : 모델이 0과 1을 얼마나 잘 구분하는지 나타내는 지표
  - `F1 Score` : Precision과 Recall의 조화 평균

  → 불균형한 데이터에서 유용한 `ROC-AUC` , `F1 Score` 기준으로 성능 평가
- 클래스 불균형 해결 위해 **피쳐 추가 및 SMOTE 기법과 scale_pos_weight 하이퍼파라미터 적용해서 소수 클래스에 가중치 부여**

결과 해석

- 모든 모델이 accuracy는 매우 높지만 이는 데이터의 클래스가 매우 **불균형**하기 때문에 **무의미**
- SMOTE 기법 적용, 소수 클래스 가중치 적용, threshold 최적화 등 적용해도 F1 score가 0.1대에서 **개선되지 않았음**

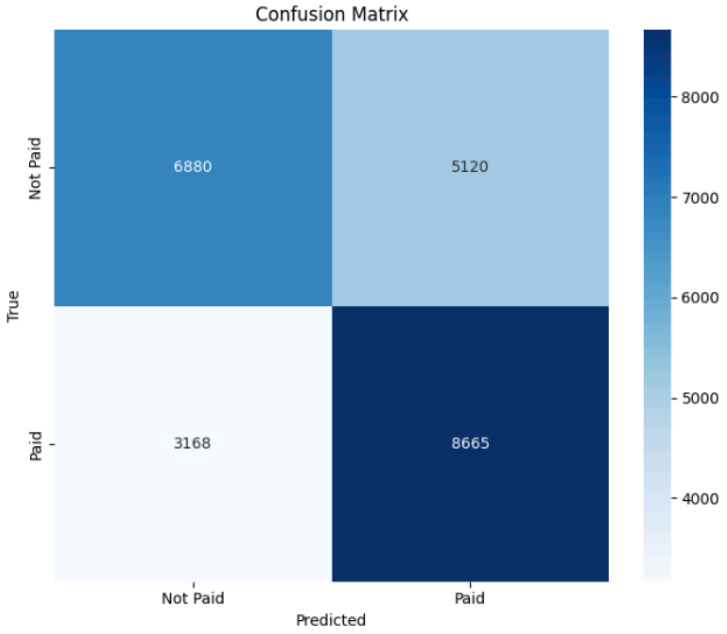


모델 학습 결과 (2)

- 모델: XGBoost 모델
- 클래스 불균형 해결: 클래스 0에서 **랜덤**하게 60,000행 추출 → 클래스 0과 1의 비율을 1:1로 임의로 맞춰줌

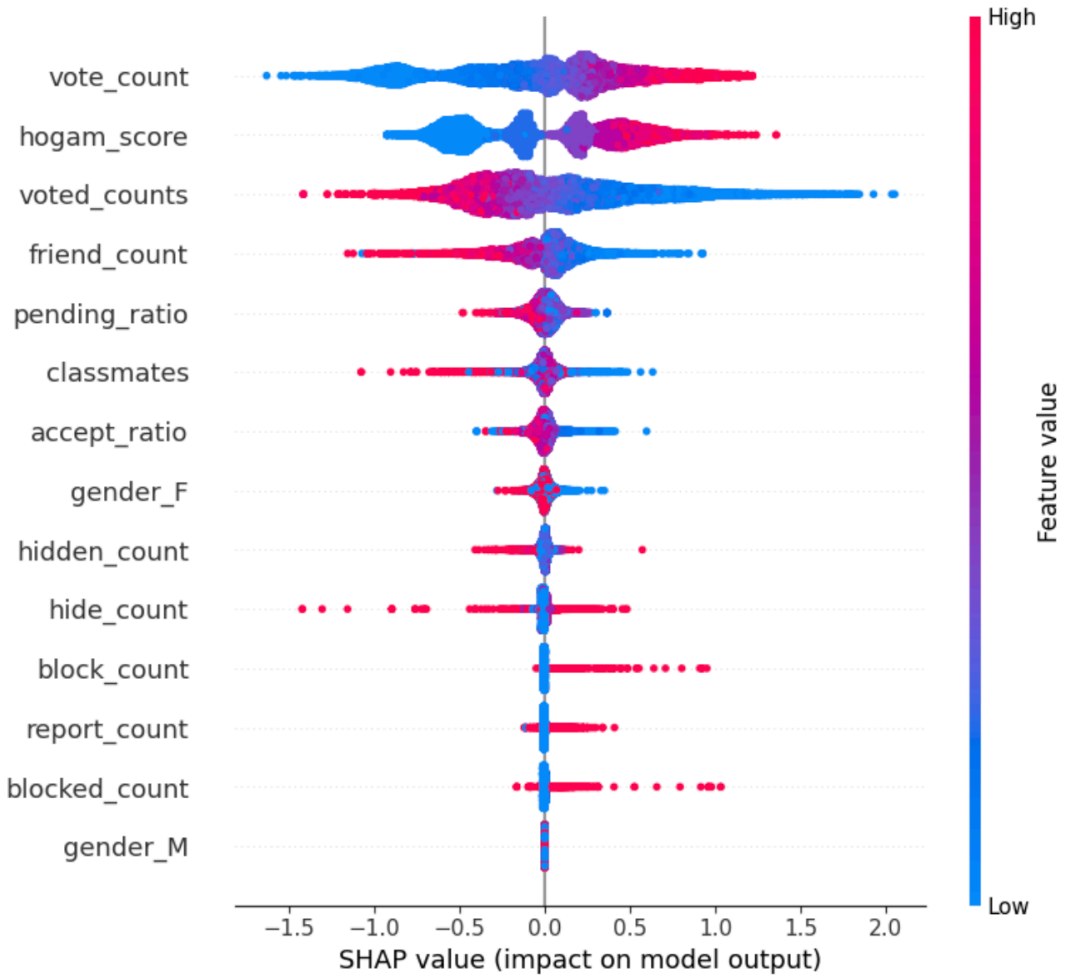
결과 정리

- 클래스 불균형이 해소되어 F1 Score가 0.68대로 크게 성능 개선됨
  - ROC-AUC Score: 0.7109
  - Average Precision Score: 0.6936
  - F1 Score: **0.6765**



피쳐 중요도 해석

- SHAP 기법 사용
  - 머신러닝 모델의 예측 결과를 해석 가능하게 만드는 기법
  - 개별 피처가 모델 예측에 얼마나 영향을 미쳤는지 수치 제시 → 모델 결과 해석에 용이함
  - 각 피처가 예측값을 얼마나 증가(+) 또는 감소(-)시키는지
- 주요 피쳐 해석
  - vote_count: 값이 높을수록 결제 확률이 증가하는 경향
  - hogam_score: 높은 호감 점수의 질문일수록 결제 확률이 증가하는 경향
  - voted_counts: 값이 높을수록 결제 확률 감소
  - friend_count: 친구 수 많을수록 결제 확률 감소
  - classmates: 큰 영향은 없지만 약간의 음의 기여



서비스 수익성 개선 방향성 요약

- **친구 수 많을수록 결제율 감소 → 친구 수 보다는 유저 간의 상호작용 증가 필요성**
- **투표 참여 활발할수록 결제율 증가 → 투표 컨텐츠 참여도 향상 시 수익성 증가**
- **호감도 높은 질문일수록 결제율 증가 → 질문 카테고리 선택시 수익 창출 가능성**

4. 개선 방안

4-1 세트당 질문 수 조정 A/B 테스트 기획

기획 배경

- 새로운 질문이 23년 6월 이후로 추가되지 않았다. > **컨텐츠 부족 & 반복 노출로 인한 컨텐츠 피로감**
- 추가 질문이 없는데도 세트당 질문 수가 10개였고 세트 완주율은 약 35%에 불과하였다. > **높은 세트 완주 허들: 진입 부담 상승**
    1. 미완주자들의 이탈 속도가 더 빠르다 (미완주자 : 18일, 완주자: 30일)
    2. 많은 세트를 투표한 사용자일수록 생존기간이 길다 (상관계수: 0.28)

실험 가설

- 세트 당 질문의 수가 줄어들면 장기 재방문율이 증가할 것이다.

평가 지표: 장기 재방문율

- 유저가 유입 후 60일 이내에 최소 한 번 이상 참여했는가?
- 초기 이용량은 양호하지만, 2개월만에 급락하는 이용량을 개선하기 위함이므로 장기적인 지표를 사용하였다.
- 기존 A그룹 대비 전환율이 상대적으로 14.96퍼센트 올라 13.5% 이상이 되면 장기 재방문율이 유의미하게 증가한 것이다.

	A그룹	B그룹	비고
그룹 인원 비율	2000	2000	
세트 당 질문 수	10개	8개	
실험 기간	신규 진입 후 6개월	신규 진입 후 6개월	
장기 재방문율	11.76%	13.5%	상대 전환 증가율: 14.96%

기대 효과

- 세트 당 질문의 수가 줄어들면 질문 소진 속도가 줄어들고 세트 완주 부담이 줄어 더 많은 세트를 완주하며 더 오래 서비스를 즐길 수 있을 것이다.

4-2 신규 서비스 기획 : 관계 발전 서사 모델

기획 배경

- 본 서비스의 핵심 기능인 투표 서비스를 전체 사용자 대비 0.7% 밖에 사용하지 않았으며, 이는 초성 확인 기능이 투표자들의 익명성을 침해해 사용자들이 투표행위 자체에 소극적인 태도를 보이는 것으로 판단하였다.
- **초성 확인 기능은 본 서비스의 주요 수익 모델이지만, 즉각적으로 '정보 격차'를 해소함으로써 유저들의 지속적인 흥미와 호기심을 유도하지 못해 실제 결제 행동으로 이어지는 비율이 낮다고 판단하였다.**
- **본 서비스는 소셜 네트워크 플랫폼의 형태를 띠고 있으나, 답변 기능 외에 사용자 간 상호작용을 지원하는 기능이 부족해 네트워크 형성과 유지가 어렵고, 이로 인해 초기 이탈률이 매우 높다고 판단하였다.**

신규 서비스 제안

관계 발전 서사 모델 - **관계 발전 서사 모델**은 익명 투표 앱 환경에서 사용자 간의 투표 활동 데이터를 실시간으로 분석하여, 두 사용자 사이에서 발생하는 **미묘한 감정적 흐름과 관계 변화를 전지적 관찰자 시점에서 서사 형태로 구현하고 전달하는 인공지능(Gemini) 기반 시스템**. 이 모델의 핵심 목표는 단순한 투표 결과 알림을 넘어, 각 상호작용에 의미와 스토리를 부여함으로써 사용자들의 서비스 몰입도를 높이고 지속적인 관계 발전에 대한 흥미를 유발하는 것이다.

- 기존 모델과의 차이점

신규 서비스에서는 투표를 한 사용자의 정보는 기본적으로 익명으로 유지되며, 사용자 간의 상호작용을 통해 관계가 발전함에 따라 상대방을 암시하는 문구가 생성되어 이를 통해 추론할 수 있다. 또한, 기존 서비스와 마찬가지로 양 유저 간의 총 투표 회수가 28회를 초과한 시점부터는 투표한 사용자의 초성 확인 기능이 추가적으로 제공된다. 이러한 변경을 통해 신규 서비스는 강화된 익명성을 바탕으로 사용자의 적극적인 투표 활동과 지속적인 참여를 유도하는 동시에, 충분한 상호작용 이후에는 제한적인 정보 공개를 통해 관계 발전을 지원할 수 있을 것으로 기대된다.

모델 파이프라인

*** 새로운 투표 행동이 발생할 때 마다 투표 질문에 관련된 감정을 분석하여 해당 유저가 속해 있는 유저쌍의 관계지표를 업데이트하 고 해당 정보를 기반으로 서사 문구생성하여 유저에게 노출.

```
1.[질문 감정 추론 모델 (Gemini)] → 감정 분류 (이성적 관심, 우정, 호기심 등)
    ↓
2.[유저 행동 기록 DB] ← 클릭, 투표 등 실시간 저장
    ↓
3.[관계 구조화] → 유저 쌍별 감정/행동 기반 관계 지표 생성
    ↓
4.[문구 생성 시스템 (Prompt Template + Gemini)] → 관계 상태 기반으로 서사 문구 동적 생성
```

1. 질문 감정 추론 모델

질문 감정 추론 모델은 gemini 에 기반하여 사용자가 투표한 질문의 텍스트를 분석하여 투표 행동 안에 담긴 감정과 의도를 예측하고 분류하는 모델이다.

- 모델의 주요 목표

- '질문의 내용', '사용된 단어', '이성/동성 여부' 를 고려하여 사용자의 투표 감정을 '이성적 관심', '호기심', '우정', '칭찬', '거리감'

으로 분류

- 코드 구현 (질문 감정 분류)

```
import os
import time
import json
import pandas as pd
import google.generativeai as genai
from google.auth import load_credentials_from_file
from google.generativeai import configure
```

```
# Configure the Gemini API with your API key
genai.configure(api_key=google_api_key)  # Replace with your actual API key
# os.environ["GOOGLE_APPLICATION_CREDENTIALS"] = "/home/twins032337/Final/bigqu

# 모델 초기화
model = genai.GenerativeModel(model_name="models/gemini-2.0-flash")


def build_batch_prompt(question_texts):
    # joined_questions = "\n".join(f'"{q}"' for q in question_texts)
    return f"""
다음 질문을 보고, 사용자가 어떤 감정에서 이 질문에 투표할 수 있을지 상위 3개의 감정을 추론해주세요.
각 질문에는 **투표자의 성별과 대상과의 관계(이성/동성)** 에 따라 투표의도가 달라질 수 있으므로, 이
결론적으로, 각 질문마다 이성간의 투표일 때 동성간의 투표일때는 나누어 감정을 분류해주세요.

카테고리 목록:
1. 이성적 관심 – 질문이 호감도나 이성적 관심과 관련되어 있다면
2. 우정 – 친구 사이에서의 행동, 정서적 유대감에 관한 질문이라면
3. 호기심 – 관계 발전, 상대방에 대한 탐구에 관한 질문이라면
4. 칭찬 – 상대방의 장점을 칭찬하는 질문이라면
5. 거리감  – 상대방에게 감정적 거리감을 느끼거나 부정적 의미에 관련한 질문이라면

각 질문마다 해당 카테고리들을 관련도 높은 순서대로 최대 3개까지 나열해주세요.
2순위, 3순위는 필요할 때만 적어도 됩니다.
출력은 아래 JSON 형식을 반드시 따라야 합니다:

반드시 아래 형식에 맞는 순수 JSON 객체만 출력해주세요.
절대 코드블럭 마크다운을 포함하지 마세요.

{{
  "질문1": {{
     "동성":["1순위", "2순위", "3순위"],
     "이성":["1순위", "2순위", "3순위"]
  }},
  ...
}}

카테고리는 반드시 아래 중 하나여야 합니다: "이성적 관심", "우정", "호기심", "칭찬", "거리감"

아래 질문들에 대해 위와 같은 형식으로 분류 결과를 JSON으로 출력해주세요:
주의사항: 만약 특정 질문에 대해 감정 분류가 어렵거나 판단이 불가능한 경우에도, 아래와 같은 JSON 형

{question_texts}
```

```
""".strip()

# 결과 저장 리스트
results = []

# 데이터프레임에서 질문을 batch로 나누어 처리
batch_size = 100
for i in range(0, len(q_df), batch_size):
    batch = q_df.iloc[i:i+batch_size]
    # print(batch)
    prompt = build_batch_prompt(batch["question_text"].tolist())
    print(batch["question_text"].tolist())

    try:
        response = model.generate_content(prompt)
        content = response.candidates[0].content.parts[0].text.strip()
        print(content)

        if content:
            try:
                json_result = extract_json_from_markdown(content)
            except json.JSONDecodeError as e:
                print(f"JSON 디코딩 오류: {e}")
                json_result = {}
        else:
            print("응답이 비어 있습니다.")
            json_result = {}

        for idx, row in batch.iterrows():
            q_text = row["question_text"]
            q_id = row["id"]
            categories_by_relation = json_result.get(q_text, {})

            for relation in ["동성", "이성"]:
                categories = categories_by_relation.get(relation, [])
                cat1 = categories[0] if len(categories) > 0 else ""
                cat2 = categories[1] if len(categories) > 1 else ""
                cat3 = categories[2] if len(categories) > 2 else ""
                results.append({
                    "question_id": q_id,
                    "question_text": q_text,
                    "relation": relation,
                    "1st": cat1,
                    "2nd": cat2,
                    "3rd": cat3
```

```
            })

        print(f"{i+1} ~ {i+len(batch)} 처리 완료")
        time.sleep(1.5)

    except Exception as e:
        print(f"오류 발생 (batch {i}~{i+batch_size}): {e}")

# 최종 결과 DataFrame 생성
classified_df = pd.DataFrame(results)
print(classified_df)
```

원본 질문 텍스트

	id	question_text
0	99	가장 신비한 매력이 있는 사람은?
1	100	"이 사람으로 한 번 살아보고 싶다" 하는 사람은?
2	101	미래의 틱톡커는?
3	102	여기서 제일 특이한 친구는?
4	103	가장 지켜주고 싶은 사람은?

감정 분류 모델 적용 후

	question_id	question_text	relation	1st	2nd	3rd
0	99	가장 신비한 매력이 있는 사람은?	동성	호기심	우정	NaN
1	99	가장 신비한 매력이 있는 사람은?	이성	이성적 관심	호기심	NaN
2	100	"이 사람으로 한 번 살아보고 싶다" 하는 사람은?	동성	호기심	우정	NaN
3	100	"이 사람으로 한 번 살아보고 싶다" 하는 사람은?	이성	이성적 관심	호기심	NaN
4	101	미래의 틱톡커는?	동성	호기심	우정	NaN

2. 유저 행동기록 데이터 전처리

- 과정 목표

실시간으로 유저 업데이트되는 서비스의 실현 가능성을 확인하기 위해 현재 보유하고 있는 행동기록 데이터를 'timestamp'를 기준으로 정렬하고 '투표자 성별', '투표받은 사용자 성별', '동성/이성', '질문/감정 분류 결과', ' 질문 텍스트'등의 정보를 포함하는 데이터프레임 생성

- 데이터프레임 형태

	From	To	From_gender	To_gender	timestamp	question_id	relation	question_text	1st	2nd	3rd
0	849436	849469	F	F	2023-04-28 12:27:49	252	동성	손이 가장 이쁘게 생겼을거 같은 사람은?	칭찬	호기심	NaN
1	849436	849446	F	F	2023-04-28 12:28:02	244	동성	대학교에서 학생회장할 것 같은 사람은?	칭찬	우정	NaN
2	849436	849454	F	F	2023-04-28 12:28:09	183	동성	나의 자존감을 가장 많이 높여줬던 사람은?	우정	칭찬	NaN
3	849436	847375	F	F	2023-04-28 12:28:16	101	동성	미래의 틱톡커는?	호기심	우정	NaN

3. 관계 구조화 및 관계 지표 생성

제공해주신 투표 기록 데이터를 바탕으로, 서비스 기획 목표에 부합하는 사용자 관계 판단을 위해 유저 쌍별 그룹화(groupby)를 수행하고, LLM 모델의 효율적인 분석을 돕는 보조 지표들을 추가했습니다. 이러한 보조 지표들은 초기 기획 단계에서 설정되었으며, 실제 문구 생성 과정에서의 결과와 피드백을 반영하여 유동적으로 수정 및 보완되었습니다. 최종적으로 추가된 관계 지표들은 LLM 모델이 사용자 간의 미묘한 상호작용 패턴과 감정 변화를 더 정확하게 이해하고, 그에 따라 맥락에 맞는 관계 발전 서사 문구를 생성하는 데 중요한 역할을 수행했습니다. 이걸 간결한 어투로 써달라고 보고서 작성에 사용할거니가

제공된 투표 기록 데이터를 유저 쌍별로 묶고, 서비스 기획 목표에 맞춰 LLM 모델의 관계 판단을 돕는 보조 지표들을 추가했습니다. 이 지표들은 초기 설계 후 실제 문구 생성 결과에 따라 유동적으로 수정되었으며, 최종적으로 LLM 모델이 사용자 간의 미묘한 상호작용과 감정 변화를 정확히 파악하여 맥락에 맞는 서사 문구를 생성하는 데 기여했습니다.

제공된 투표 기록 데이터를 유저 쌍별로 묶고, 서비스 기획 목표에 맞춰 LLM 모델의 관계 판단을 돕는 보조 지표들을 추가하였다. 이 지표들은 초기 설계 후 실제 문구 생성 결과에 따라 유동적으로 수정되었으며, 최종적으로 LLM 모델이 사용자 간의 미묘한 상호작용과 감정 변화를 정확히 파악하여 맥락에 맞는 서사 문구를 생성하는 데 기여하는 방향으로 설정하였다.

- 관계 지표 설명

지표 이름	설명
`total_votes`	현재까지 해당 유저쌍 간에 발생한 총 상호작용(투표) 횟수
`A_to_B_count`	A 유저가 B 유저에게 보낸 투표 횟수
`B_to_A_count`	B 유저가 A 유저에게 보낸 투표 횟수
`last_vote_direction`	현재 투표의 방향(A→B 또는 B→A) - 투표의 방향에 따라 문구를 제공 받는 주체를 판단
`first_vote_timestamp`	해당 유저쌍 간 최초로 투표가 이루어진 시각
`last_vote_timestamp`	현재 투표가 이루어진 시각
`vote_intervals`	지금까지의 투표 간 시간 간격들의 리스트 - 유저간의 최근 상호작용 빈도를 반영하기 위함
`interaction_stage`	현재 유저쌍의 상호작용 단계 (`단방향` , `초기상호작용` , `반복상호작용` 등으로 분류) - 상호작용 단계를 통해 관계진척도를 효율적으로 반영하기 위함
`is_reciprocal`	해당 유저쌍이 서로에게 투표한 적이 있는지 여부 (True/False) - 교차 투표 발생시점 이후로 상대방을 유추할 수 있는 문구 생성
`recent_reciprocation`	최근 3회 내에 상호작용의 방향성이 양방향으로 나타났는지 여부 - 최근 유저들간 감정 변향성을 반영하기 위함
`emotion_list`	마지막 투표까지의 감정 시퀀스 리스트

- 코드 구현

```
import pandas as pd
import numpy as np

def generate_user_pair_progressive_metrics(df_pair): # df_pair = 각 유저쌍
df_pair['timestamp'] = pd.to_datetime(df_pair['timestamp'])
df_pair = df_pair.sort_values('timestamp').reset_index(drop=True)

A, B = df_pair['From'].iloc[0], df_pair['To'].iloc[0]
results = []
emotions = [] # 감정 누적 리스트

gender_relation = df_pair['relation'].iloc[0] # '이성' 또는 '동성'


    for i in range(1, len(df_pair) + 1):
        sub_df = df_pair.iloc[:i]
        total_votes = i

        if total_votes > 27:  # 28회차 이상 부터는 초성확인 기능으로 대체, 상세정보 요약
            results.append({
                'round': i,
                'total_votes': total_votes,
                'A_to_B_count': None,
                'B_to_A_count': None,
                'first_vote_timestamp': None,
                'last_vote_timestamp': None,
                'vote_intervals': None,
                'last_vote_direction': None,
                'interaction_stage': None,
                'is_reciprocal': None,
                'recent_reciprocation': None,
                'top_emotion': None
            })
            continue

        A_to_B_count = (sub_df['From'] == A).sum()
        B_to_A_count = (sub_df['From'] == B).sum()
        first_vote_timestamp = sub_df['timestamp'].iloc[0]
        last_vote_timestamp = sub_df['timestamp'].iloc[-1]
        vote_intervals = sub_df['timestamp'].diff().dt.total_seconds().div(60).dropna().tolist
()
        vote_intervals = [round(x, 1) for x in vote_intervals]
        last_vote_direction = 'A→B' if sub_df['From'].iloc[-1] == A else 'B→A'
```

```
        if A_to_B_count == 0 or B_to_A_count == 0: # 상호작용 단계 분류
            interaction_stage = '단방향'
        elif A_to_B_count <= 2 and B_to_A_count <= 2:
            interaction_stage = '초기 상호작용'
        elif A_to_B_count >= 3 and B_to_A_count >= 3:
            interaction_stage = '반복 상호작용'


        is_reciprocal = A_to_B_count > 0 and B_to_A_count > 0 # 상호작용 여부
        recent_3 = sub_df.tail(3)
        recent_reciprocation = 'Yes' if recent_3['From'].nunique() > 1 else 'No'

        emotion = sub_df.iloc[-1]['1st'] # 감정시퀀스 추가
        if pd.notna(emotion):
            emotions.append(emotion)
        emotion_sequence = emotions.copy()
        question_text = sub_df.iloc[-1]['question_text'] # 질문 텍스트

        results.append({
            'round': i,
            'total_votes': total_votes,
            'gender_relation':gender_relation,
            'A_to_B_count': A_to_B_count,
            'B_to_A_count': B_to_A_count,
            'first_vote_timestamp': first_vote_timestamp,
            'last_vote_timestamp': last_vote_timestamp,
            'vote_intervals': vote_intervals,
            'last_vote_direction': last_vote_direction,
            'interaction_stage': interaction_stage,
            'is_reciprocal': is_reciprocal,
            'recent_reciprocation': recent_reciprocation,
            'emotion_sequence' : emotions,
            'question_text' : question_text
        })

    return pd.DataFrame(results)
```

4. 문구 생성 시스템

Gemini api를 이용한 자동 문구 시스템 구축

- 설계 목표

1. 프롬프트 설계 과정에서 익명 투표를 통한 **관계 발전**이라는 서비스의 핵심 컨셉을 명확히 정의 및 분석 가이드라인 제공

2. LLM 이 이해하지 못하는 관계 변화를 보조지표 추가를 통해 반영
3. 기획 목적을 왜곡하지 않는 범위에서 LLM의 연산 부담을 줄이고 응답 속도를 향상시키기 위해, 프롬프트의 토큰 수를 효율적으로 관리

- 코드 구현

```
import os
import time
import json
import pandas as pd
import google.generativeai as genai
from dotenv import load_dotenv

def generate_messages(df, batch_size=5, sleep_sec=1):
# 1. 환경변수 로드 및 모델 설정
load_dotenv()
genai.configure(api_key=os.getenv("GOOGLE_API_KEY"))
model = genai.GenerativeModel(model_name="models/gemini-1.5-pro")

# 2. 단일 배치 처리 함수 정의
def generate_message(df_pair):
    df_copy = df_pair.copy()
    for col in df_copy.columns:
        if pd.api.types.is_datetime64_any_dtype(df_copy[col]):
            df_copy[col] = df_copy[col].dt.strftime('%Y-%m-%d %H:%M:%S')

    def convert_to_serializable(obj):
        if isinstance(obj, pd.Timestamp):
            return obj.strftime('%Y-%m-%d %H:%M:%S')
        elif isinstance(obj, pd.Timedelta):
            return str(obj.total_seconds())
        elif isinstance(obj, list):
            return [convert_to_serializable(i) for i in obj]
        elif isinstance(obj, dict):
            return {k: convert_to_serializable(v) for k, v in obj.items()}
        else:
            return obj

    data_list = [convert_to_serializable(record) for record in df_copy.to_dict(orient='record
s')]

    system_prompt = """
```

당신은 감정적으로 섬세하고 맥락을 잘 이해하는 어시스턴트입니다.

현재 우리는 익명 투표앱을 운영 중이며, 사용자는 특정 질문에 대해 다른 사용자에게 투표할 수 있습니다.
투표를 받은 사용자는 투표자의 정체를 알 수 없지만, 투표를 받을 때마다 생성된 특정문구를 전달받습니다.

당신은 json 형태의 데이터를 기반으로 두 사용자 간의 감정적 흐름과 상호작용 패턴을 분석하고,
각 투표마다 사용자에게 보여줄 감정 서사 기반 문구를 **전지적 관찰자 시점**에서 생성하는 역할을 합니다. 문구는 질문 텍스트를 반영해 생성해주세요.

중고등학생을 대상으로 하는 앱이므로 문장은 트렌디하고 캐주얼해야 하며, 친근한 어투로 표현해줘.
반복적인 상호작용이나 감정의 누적, 양방향성의 발생 등은 관계발전으로 문장에 자연스럽게 반영되어야 합니다.

문장은 의미심장하고 감정적으로 여운을 남겨야 하며,
특히 상호작용이 양방향으로 발생한 이후에는 서로를 점점 더 의식하게 되는 분위기를 암시해야 합니다.

생성되는 문구에서는 투표한 사용자나 투표받은 사용자를 특정하여 지칭하지 마세요. "A", "B"와 같은 명칭 대신, 익명의 관계 속에서 발생하는 감정과 변화를 중심으로 서술해 주세요.

당신의 응답은 입력받은 JSON 리스트 각 항목마다 대응되는 문구가 포함된 JSON 리스트 형태여야 합니다.
각 항목에 "generated_message"라는 key를 추가하여 해당 문장을 포함시켜 주세요.

generated_message는 반드시 연애나 이성적 관심을 나타낼 필요는 없으며, 동성 관계라면 로맨스적 뉘앙스보다 우정의 뉘앙스를 강조하여 사회적 맥락에 맞춰 자연스럽게 표현해 주세요.

데이터내 각 컬럼의 의미는 다음과 같습니다.

'total_votes' - 현재까지 해당 유저쌍 간에 발생한 총 상호작용(투표) 횟수
'gender_relation' - '이성' 또는 '동성'
'A_to_B_count' - A 유저가 B 유저에게 보낸 투표 횟수
'B_to_A_count' - B 유저가 A 유저에게 보낸 투표 횟수
'last_vote_direction' - 현재 투표의 방향 (A→B 또는 B→A)
'first_vote_timestamp' - 해당 유저쌍 간 최초로 투표가 이루어진 시각
'last_vote_timestamp' - 현재 투표가 이루어진 시각
'vote_intervals' - 지금까지의 투표 간 시간 간격들의 리스트 (단위: 초 또는 분)
'interaction_stage' - 현재 유저쌍의 상호작용 단계 ('단방향', '초기상호작용', '반복상호작용' 등으로 분류)
'is_reciprocal' - 해당 유저쌍이 서로에게 투표한 적이 있는지 여부 (True 또는 False)
'recent_reciprocation' - 최근 3회 내에 상호작용의 방향성이 양방향으로 나타났는지 여부

```
    'emotion_list' - 마지막 투표까지의 감정 시퀀스 리스트
    'question_text' - 질문 텍스트


    응답 형식 예시는 다음과 같습니다:

    [
    {
        "total_votes": 1,
        "A_to_B_count": 1,
        ...
        "generated_message": "😂너를 생각하면 웃음이 나는 사람이 있나 봐! ..."
    },
    ...
    ]
    이 형식을 정확히 따라주세요.
    """

    response = model.generate_content(
        [
            {"role": "user", "parts": [system_prompt]},
            {"role": "user", "parts": [
                "다음 JSON 리스트에 대해 각 항목별로 감정적 서사 문장을 'generated_message' 필드에 넣어 반환해주세요.",
                json.dumps(data_list, ensure_ascii=False)
            ]}
        ],
        generation_config={
            "temperature": 0.9,
            "top_p": 1.0,
            "top_k": 40,
            "max_output_tokens": 2048
        }
    )

    try:
        result = json.loads(response.text)
        result_df = pd.DataFrame(result)
        return pd.concat([df_pair.reset_index(drop=True), result_df[['generated_message']]], axis=1)
    except Exception as e:
        print("⚠️ JSON 파싱 오류 발생:", e)
        print("Gemini 응답 내용:\\n", response.text)

        import re
```

```
        messages = re.findall(r'"generated_message"\\s*:\\s*"([^"]+)"', response.text)
        if messages:
            while len(messages) < len(df_pair):
                messages.append("")
            return df_pair.reset_index(drop=True).assign(generated_message=messages[:len
(df_pair)])
        else:
            return df_pair.reset_index(drop=True).assign(generated_message="메시지 생성 실
패")

# 3. 전체 배치 처리 루프
all_results = []
num_rows = len(df)

for start in range(0, num_rows, batch_size):
    end = min(start + batch_size, num_rows)
    batch_df = df.iloc[start:end]
    print(f"📦 Processing batch {start} to {end}...")

    try:
        result = generate_message(batch_df)
        all_results.append(result)
    except Exception as e:
        print(f"❌ Error in batch {start}-{end}: {e}")
        fallback_df = batch_df.copy()
        fallback_df["generated_message"] = "메시지 생성 실패"
        all_results.append(fallback_df)

    time.sleep(sleep_sec)

return pd.concat(all_results, ignore_index=True)
```

- 문구 생성 샘플 (A = user_id : 1577440, B = user_id : 1577437)

```
[
{
  "round": 1,
  "total_votes": 1,
  "gender_relation": "이성",
  "vote_direction": "A→B",
 ...
  "question_text": "침대에 인형 가득 쌓아놓을 것 같은 사람은?",
  "generated_message": "🧸 침대 가득 쌓인 곰돌이처럼, 누군가에게 포근한 인상을 남겼나 봐요!
```

첫인상, 꽤 괜찮았던 거 같죠? 😉"
},
{
"round": 2,
"total_votes": 2,
"gender_relation": "이성",
"vote_direction": "A→B",
...
"question_text": "끈기가 제일 좋은 사람은?",
"generated_message": "💪 누군가 당신의 끈기에 주목했어요! 쉽게 포기하지 않는 모습이 멋있어 보였나 봐요. 두 번째 투표라니... 좀 신경 쓰이는데요? 🤔"
},
{
"round": 3,
"total_votes": 3,
"gender_relation": "이성",
"vote_direction": "B→A",
...
"question_text": "눈물이 많은 친구는?",
"generated_message": "💧 서로에게 투표했네요! 눈물이 많다는 건, 그만큼 감수성이 풍부하다는 뜻이기도 하죠. 혹시... 서로를 좀 더 알고 싶어진 걸까요? 묘한 긴장감이 흐르는 것 같아요. 😊"
},
{
"round": 4,
"total_votes": 4,
"gender_relation": "이성",
"vote_direction": "B→A",
...
"question_text": "새해 첫날에 보고 싶은 친구는?",
"generated_message": "🎉 새해 첫날부터 보고 싶을 만큼, 특별한 존재가 되었나 봐요! 서로를 향한 투표가 쌓여가면서, 둘 사이의 분위기가 점점 더 깊어지는 느낌이에요. 🥰"
},
{
"round": 5,
"total_votes": 5,
"gender_relation": "이성",
"vote_direction": "A→B",
...
"question_text": "가장 김밥을 잘 만들 것 같은 사람은?",
"generated_message": "🍙 김밥처럼 둘 사이에 뭔가 꽁꽁 싸매진 비밀이 있는 것 같아요! 투표할 때마다 서로를 떠올리는 모습... 상상만 해도 흐뭇하네요. 이 관계, 어디까지 갈지 궁금해지는데요? 🤭"
}
]

```
[
  {
    "round": 6,
    "total_votes": 6,
    "gender_relation": "이성",
    "vote_direction": "B→A",
   ...
    "question_text": "과거로 돌아간다면 더 빨리 친해지고 싶은 친구는?",
    "generated_message": "과거로 돌아가 다시 시작하고 싶을 만큼 특별한 인연이라니, 시간을 거슬러 마음이 전해졌네요.🕰️ 서로에게 더 빨리 다가가고 싶은 마음, 설렘과 아쉬움이 교차하는 순간이에요."
  },
  {
    "round": 7,
    "total_votes": 7,
    "gender_relation": "이성",
    "vote_direction": "B→A",
   ...
    "question_text": "눈이 제일 큰 사람은?",
    "generated_message": "👀 끌리는 눈빛, 묘한 분위기! 눈에 띄는 매력을 가진 사람에게 시선이 집중되는 건 어쩔 수 없나 봐요. ✨ 그 눈빛에 담긴 이야기가 궁금해지는 순간이에요."
```


신규 서비스 제한점

해당 신규 서비스는 Gemini-Pro 모델을 활용하여 사용자 간 투표 데이터를 기반으로 감정 서사 문구를 자동 생성하는 기능을 제공한다. 이 과정에서 모델 호출이 빈번하게 발생하며, 특히 사용자 간 투표 횟수가 많을수록 API 호출량이 증가하여 이에 따른 운영 비용도 비례적으로 상승하는 구조를 가지고 있다.

그러나 현재 해당 서비스가 실제 사용자 경험 개선에 얼마나 기여하며, 궁극적으로 유료 결제 전환율 증가로 이어질지는 불확실한 상황이다. 즉, 서비스 운영에 따른 비용이 향후 유료 전환을 통해 기대되는 수익을 초과할 가능성도 존재한다.

따라서 본 서비스를 전면적으로 도입하기보다는, 기존 서비스와 신규 서비스를 병행 운영하면서 A/B 테스트를 통해 유저의 반응과 결제 전환율, 체류 시간 등의 주요 지표를 비교·분석할 필요가 있다. 이를 통해 비용 대비 효과성을 정량적으로 검토하고, 유의미한 성과가 확인될 경우 점진적으로 서비스 적용 범위를 확대하는 것이 바람직하다.

4-3 추가 개선안

- 컨텐츠 부족을 해소하기 위한 주기적인 신규 질문 업데이트
- 질문 반복 노출로 인한 피로도를 해소하기 위한 질문 카테고리화 : 유료 광고 시청 후 원하는 카테고리의 질문 선택 가능