ASK-U

이탈 유저 떡 하나 더 주기

김진영 이태정 진명훈

Our Slide

01

Intro.

02

Feature &
EDA.

03

Model.

04

Conclusion.

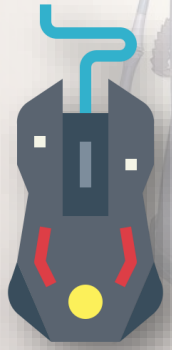
01

Intro.

Lineage

현재의 NCSOFT를 만든,

오늘까지도 NCSOFT를 책임지는 게임



간편한 조작 방식

게임 내
사회 활동 형성

가상의 특별한
전투 경험 부여

01

Intro.

Lineage

유저들은 현실 세계에서 받은 스트레스를 Lineage에서 제공하는

훌륭한 가상 세계의 경험, 다양한 전투와 컨텐츠들로 풀 수 있습니다!

그저 빛… 갓니지

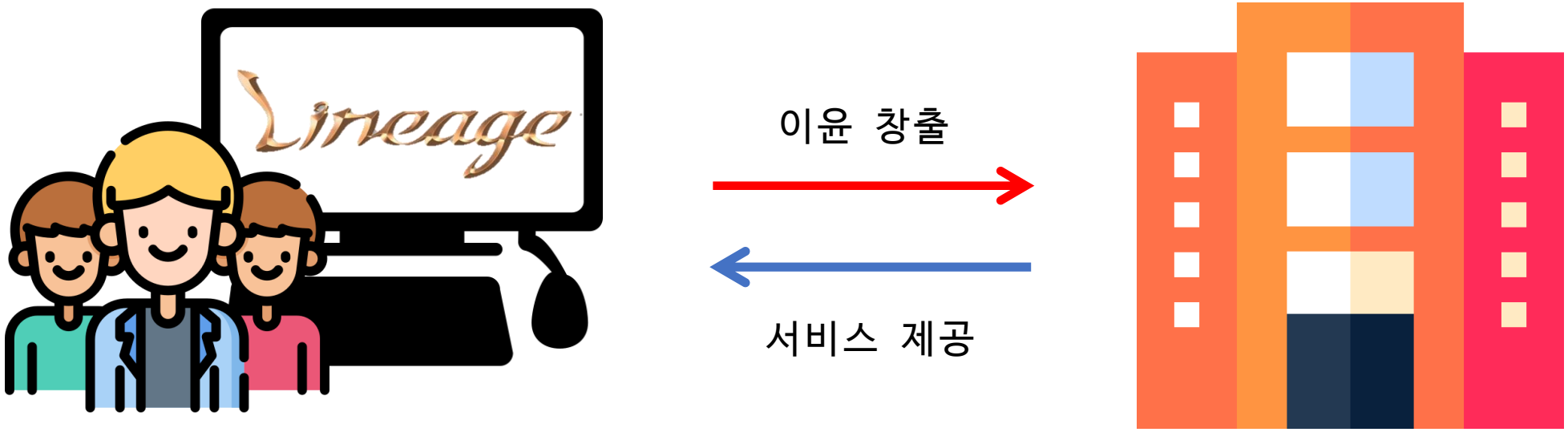
01

Intro.

Lineage

NCSOFT는 유저들에게 리니지 서비스를 제공,

유저는 게임을 즐기며 필요시 과금/정액권을 사용, 회사 수익 창출합니다



01

Intro.

User Churn

그러나.. 유저가 이탈하게 된다면?

=> 연쇄적인 이탈 발생 및 손해 발생

```
def is_Churn(latest_login):
    # 장기 미접속자
    # 7일 동안 접속 X이면
    # 7일을 추가로 관찰하고
    # 미접속시 이탈로 간주
    if today - latest_login >= 14:
        return True
    else: return False
```

Bye... 즐거웠어

이윤 창출

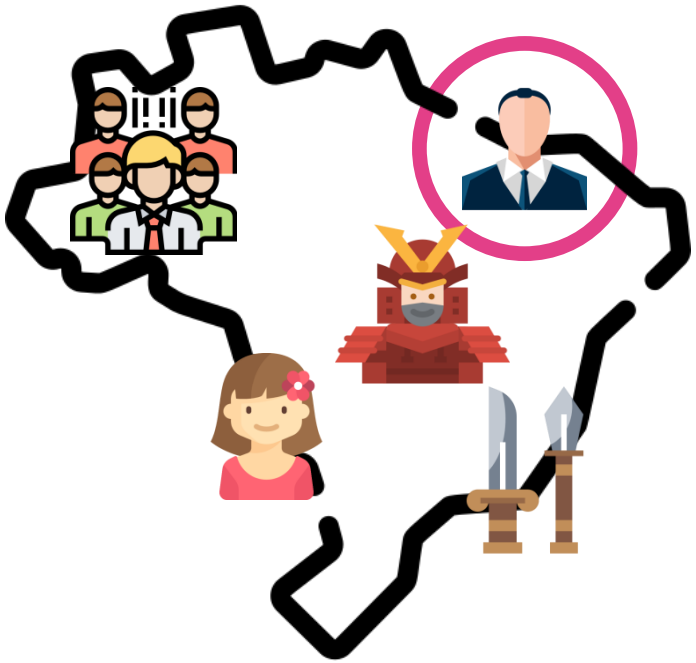
서비스 제공

01

Intro.

User Churn을 어떻게 방지할까?

Lineage



- Play pattern
- Login pattern
- Combat pattern
- Trade pattern
- Relation pattern
- Paid pattern
- Etc…

유저의 게임 내
활동 데이터 기반으로

이탈 이상 징후를 탐지

그리고 적절한 보상 부여

01

Intro.

User Churn을 어떻게 방지할까?

Feature EDA

이탈 징후를 보이는 유저의 특징을 탐색

Modeling

이탈 유저를 판별, 보상을 하기 위한 모델 구축

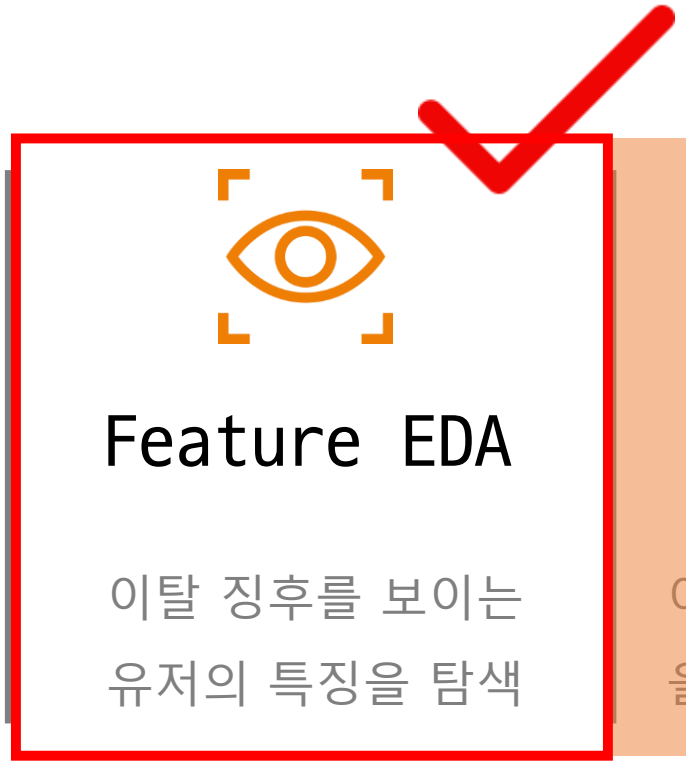
Conclusion

구축한 모델 성능 개선 및 결과, 향후 방향성 제시

02

Feature & EDA.

이탈 징후를 보이는 유저의 특징



Feature EDA

이탈 징후를 보이는 유저의 특징을 탐색

Modeling

이탈 유저를 판별, 보상을 하기 위한 모델 구축

Conclusion

구축한 모델 성능 개선 및 결과, 향후 방향성 제시

02

Feature & EDA.

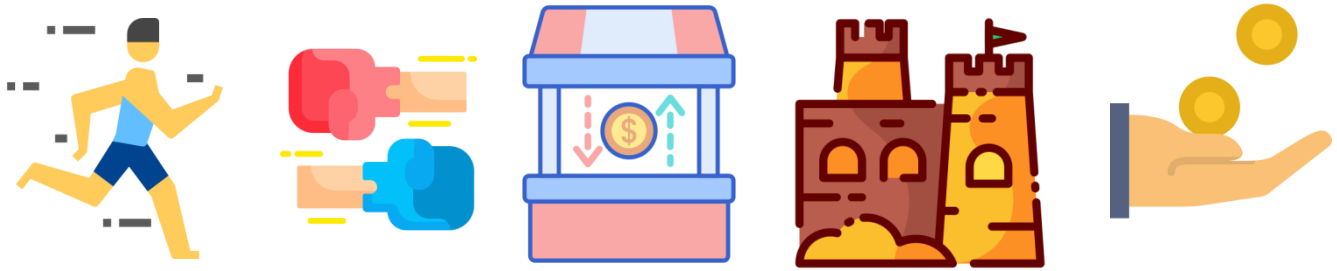
변수를 어떻게 구축할 것인가?

유저 **이탈(장기 미접속)**에는 다양한 이유 존재

- 재미 없어서
- 레벨 올리기가 힘들어서
- 혈맹이 활발하지 않아서
- 일이 생겨서
- 전투에서 져서
- 자꾸 죽어서
- etc

유저가 **과금**을 하는 이유도 크게 다르지 않음

- 추가적인 레벨업을 위해서
- 좋은 아이템을 사기 위해서
- 직업군이 과금이 필요해서
- 과금할 캐릭터가 많아서
- 전투에서 보다 강하게 만들기 위해서
- 혈맹에서 보다 활약을 많이 하기 위해서



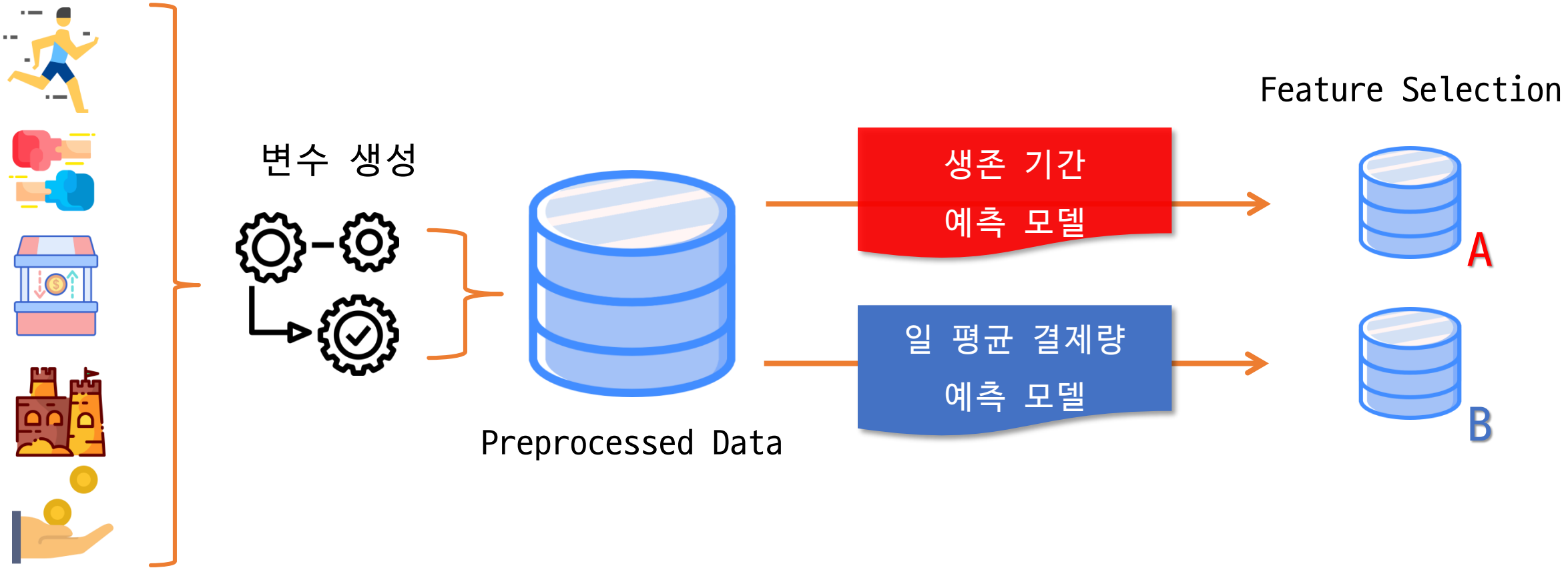
activity combat trade pledge payment

02

Feature & EDA.

변수를 어떻게 구축할 것인가?

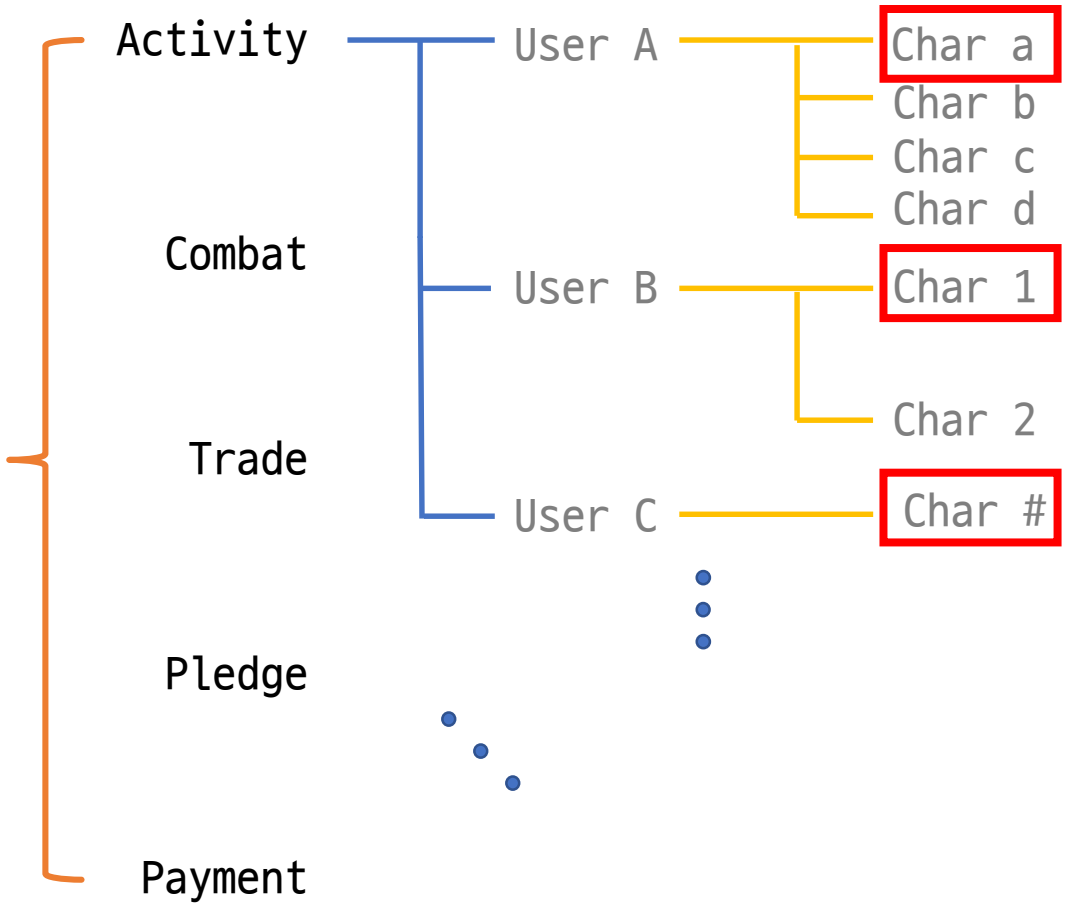
앞선 아이디어를 바탕으로 Feature를 구축
그 후, 생존 기간, 일 평균 결제량 모델 별
우수한 Feature 선정하는 방식으로 진행



02

Feature & EDA.

어떻게 데이터셋을 구성할 것인가?



Train 40,000명의 유저에 대해 각기 다른 개수의 character를 육성한다.

예측을 위해

1. 주캐의 특성을 유저의 특성으로
2. 사용하는 캐릭터를 반영하여

위 두 가지 방식으로 Char 차원을 축소하여 데이터 셋을 구성

주캐의 선정 기준은

주캐 = argmax(char's level)

02

Feature & EDA.

Make Feature (Activity)

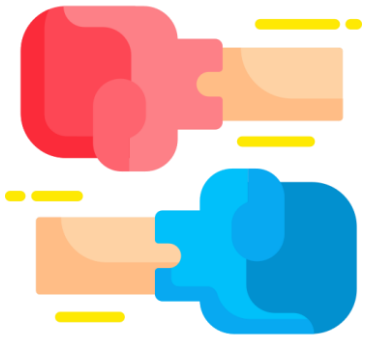
activity

Playtime	• 유저 별 일일 플레이 시간 3개 군집화
Login	• 주차 별 접속 횟수 • 28일 간 접속 패턴 군집화 • 총 접속일 수 계산
Char	• 유저 별 캐릭터 수, 일 별 사용 캐릭터 수
Give up	• Death – Revive, 게임에 대한 의지 대변
Exp	• Solo, Party, Quest에서 얻은 Exp로 군집화 후 numbering
Game Money Change	• 유저 단위, 캐릭터 단위로 game money 증/감 count

02

Feature & EDA.

Make Feature (Combat)



combat

Combat

- 전투 횟수
- 캐릭터 별 전투 데이터 편차(STD)
- 유저 별 전투 데이터 편차(STD)

Level

- 보유 캐릭터의 max level – min level
- 유저가 보유한 캐릭터의 max level(주캐의 Lev)
- 부캐의 평균 레벨

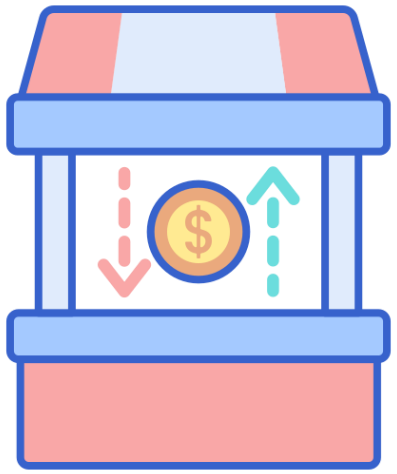
Class

- 유저 별 보유한 클래스(직업)의 수

02

Feature & EDA.

Make Feature (Trade)



trade

Item

- 매수/매도 아이템 수량 및 가격
- 아이템 별 매수/매도 횟수
- 판매 – 구매 물량
- 판매/구매 금액
- Margin = 판매 금액 – 구매 금액

Trade Type

- 교환 창/개인 상점 별 매수/매도 여부
- 교환 창/개인 상점 별 거래 합

User/Char

- 유저 별 Item, Trade Type feature 생성
- 주력 캐릭터 Item, Trade Type feature 생성
- 부수 캐릭터 Item, Trade Type feature 생성

Trade Time

- 일 평균 거래 횟수
- 평균 매수/매도 일자 (매수매도 일자 / 로그인 횟수)
- 주 판매 시간, 주 구매 시간 = Avg(매수매도 시간)
- Avg(총 거래 시간) = (주 판매 시간 + 주 구매 시간) / 2
- 24시간 중 1시간 단위의 주 거래 시간

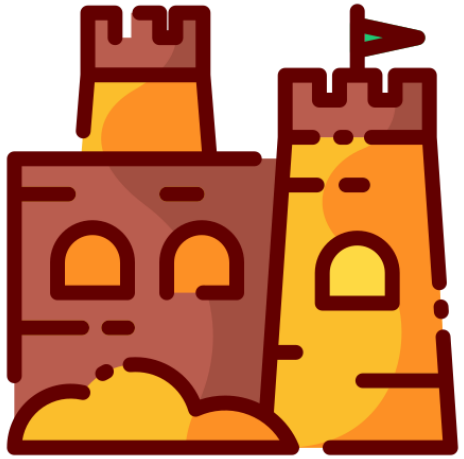
User2Other

- 매수/매도 상대방의 수/합/비율(count, sum, ratio)
  (장사꾼 여부 간접적 판별 가능)

02

Feature & EDA.

Make Feature (Pledge)



pledge

Pledge

- 가입한 혈맹의 순위 (혈맹원 수 기준)
- 혈맹원 합
- 혈맹 활동일

User

- 유저 별 가입한 혈맹의 수
- 주 캐릭터의 혈맹 활동
- 부 캐릭터의 혈맹 활동

Server

- a 서버 접속 횟수
- b 서버 접속 횟수

02

Feature & EDA.

Make Feature (Payment)

payment

Payment

- 일 평균 결제량 : 유저 별 28일간 pay의 합 / 결제일 수
- 비 접속 결제 : login하지 않은 상태에서의 결제 유무
- 28일 평균 결제량 : 유저 별 28일간 pay의 합 / 28

Pattern

- 결제 패턴 28일 간의 결제 패턴을 군집화, numbering

Count

- 유저 별 결제한 일 수

02

Feature & EDA.

Make Feature (Flatten)

acc_id 기준으로 groupby하여 Day를 압축시키면

특정 일자의 데이터(이벤트)가 미치는 영향을 잃어버리게 됩니다.

ID	Week	X_feature
2	1	0.5
	2	1.1
	3	2.6
	4	0.7

Groupby.SUM()

ID	SUM_1~4(X_feature)
2	4.9
5	2.6

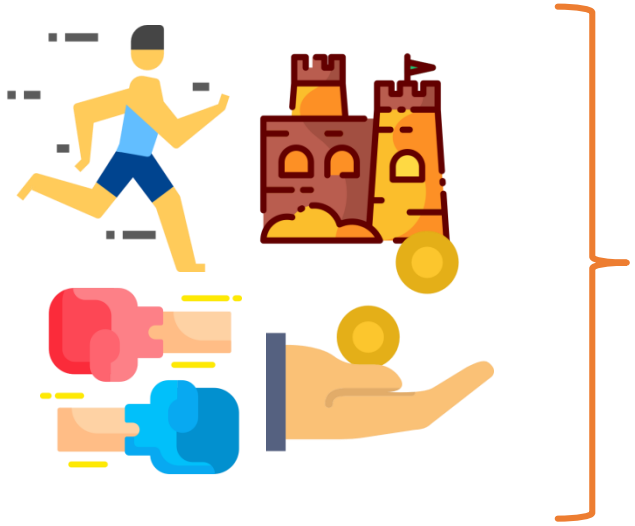
주차별로 다른 정보가 유실됨!

02

Feature & EDA.

Make Feature (Flatten)

따라서 종합 통계량만을 계산하는 것이 아니라
주차별로 데이터의 통계량을 계산하여 Feature로 만들어주는 방법을 활용했습니다. (주차 별 데이터는 7일 씩 SUM을 했습니다)



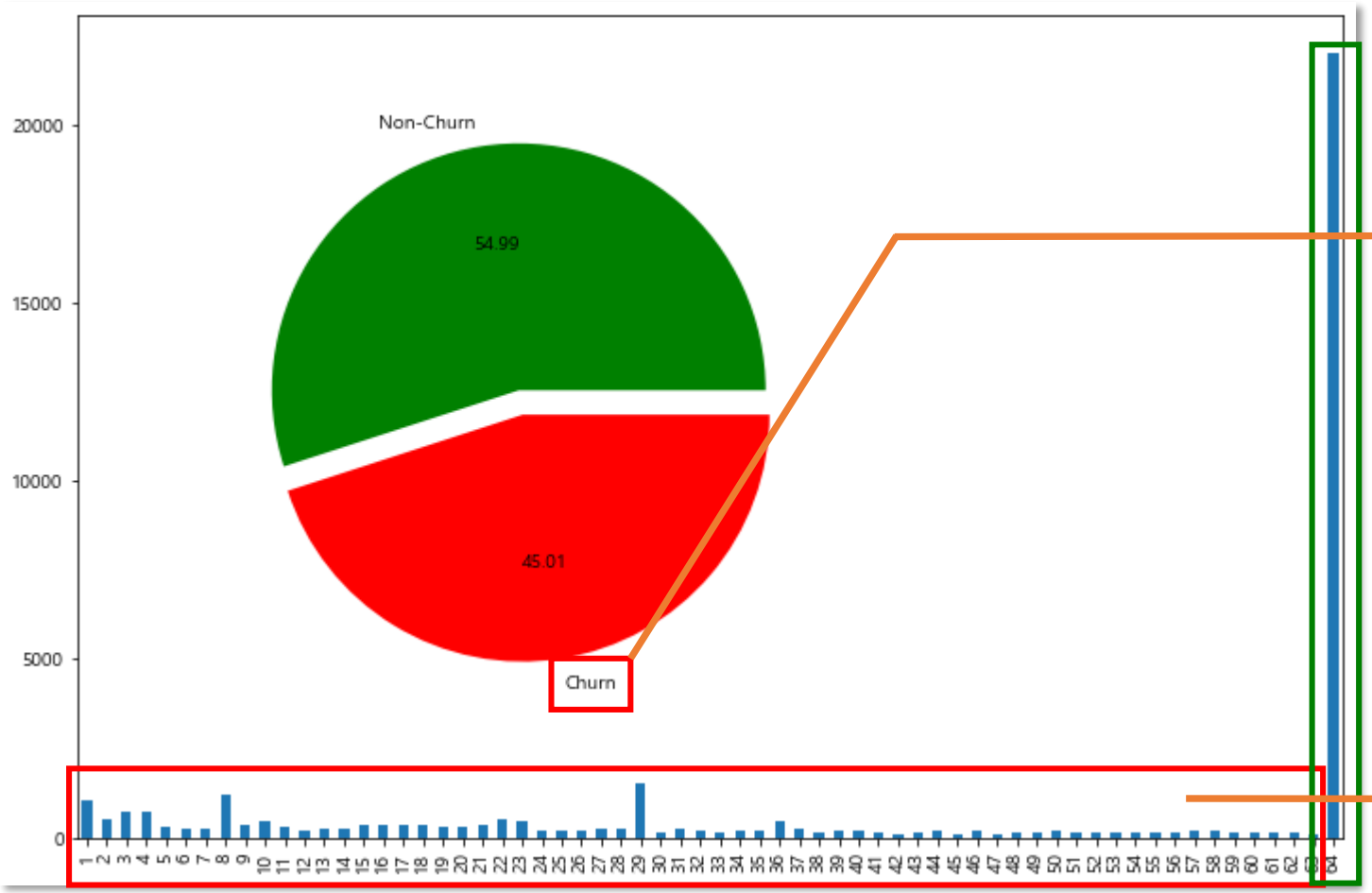
ID	SUM_1(X)	SUM_2(X)	SUM_3(X)	SUM_4(X)
2	0.5	1.1	2.6	0.7
5	0	1.2	0.1	1.3
⋮				

주차에 민감한 변수 Ex) Give up, Combat, pledge_rank, payment

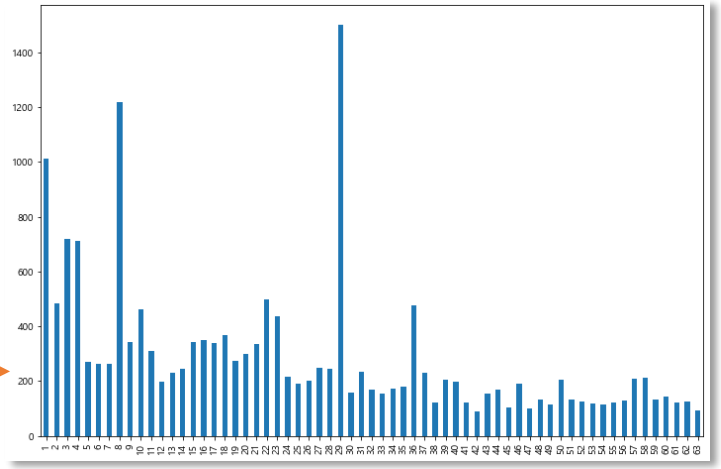
02

Feature & EDA.

잔존 기간 예측을 위한 EDA



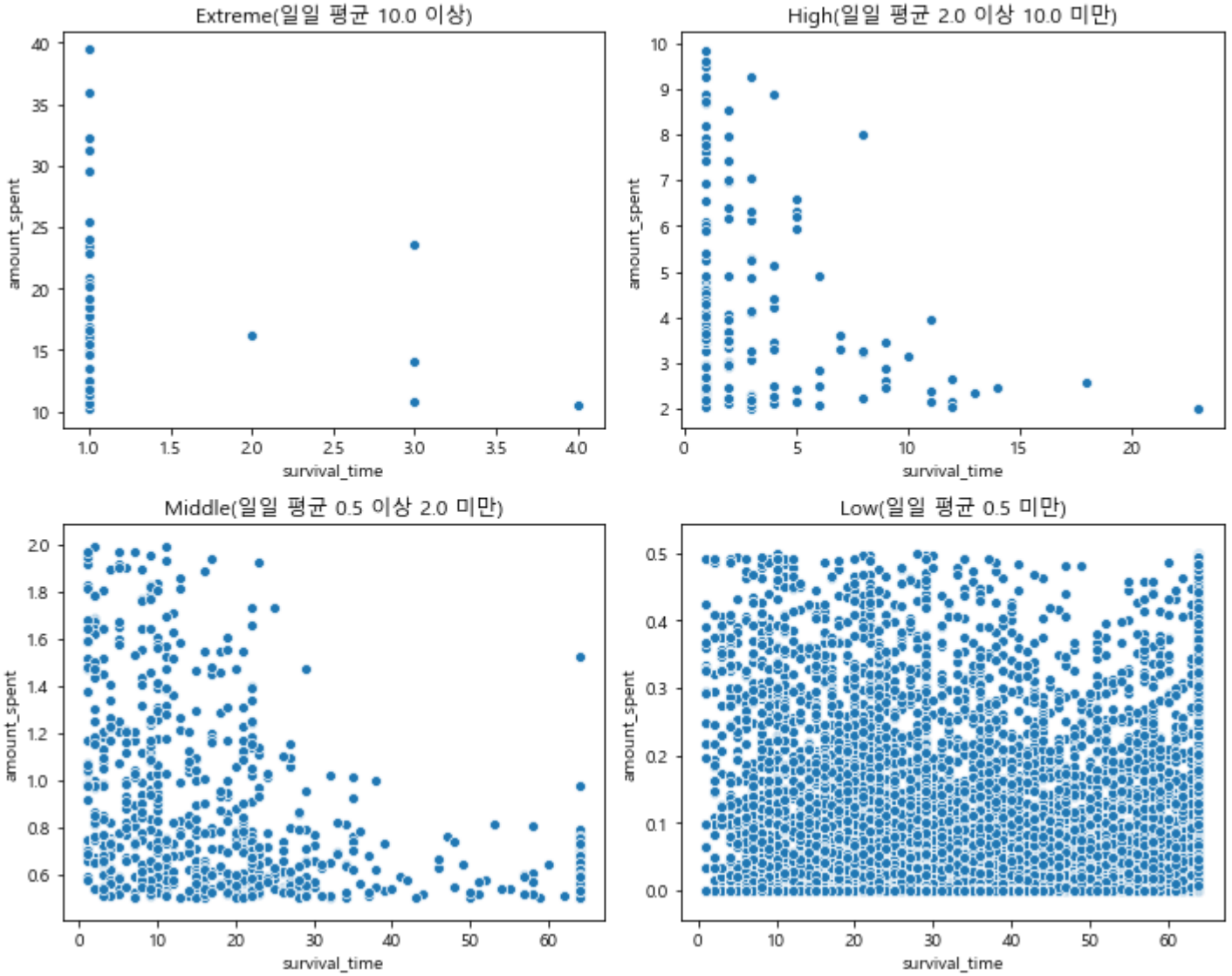
- 이탈 유저와 잔존 유저의 비율은 상이
- Class 불균형 문제
- 예측 실시 시, 1~10일 혹은 28일, 64일로 결과가 쏠릴 확률이 큼



02

Feature & EDA.

과금의 양을 예측하기 위한 EDA



Amount Spent(일 평균 결제량)

0.5 미만의 금액을 사용한 유저 군에서 생존 기간에 따른 밀집 현상은 존재하지 않고

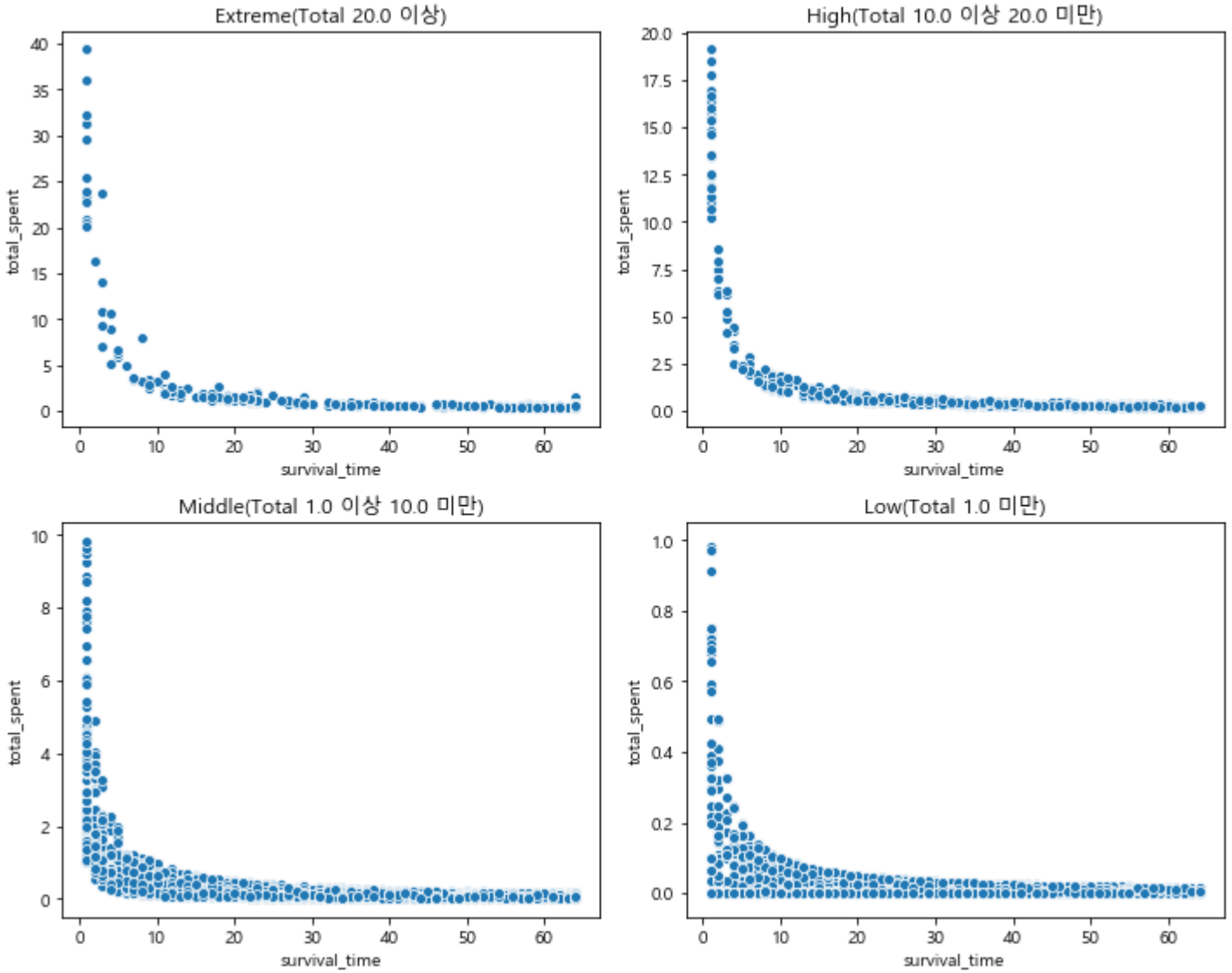
2.0, 10.0, 그리고 그 이상으로 갈수록 일 평균 결제 량이 굉장히 높으면서 잔존 기간은 짧아지는, 그래프에서 왼쪽으로 치우쳐지는 것을 확인할 수 있다.

또한 1~4일 조기에 이탈하는 유저들의 결제량이 압도적으로 높다는 점도 괄목할 대상이다.

02

Feature & EDA.

과금의 양을 예측하기 위한 EDA



Amount spent * Survival time

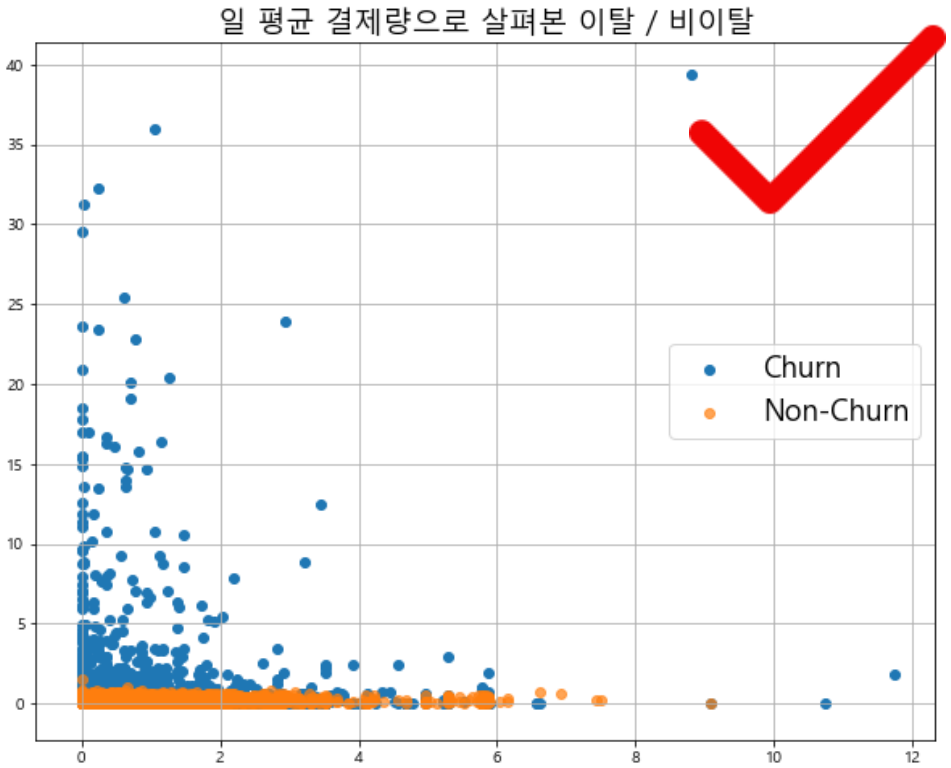
전체 소비량을 대상으로 결제 량을 예측하기 위해 이에 대한 plot을 시각화

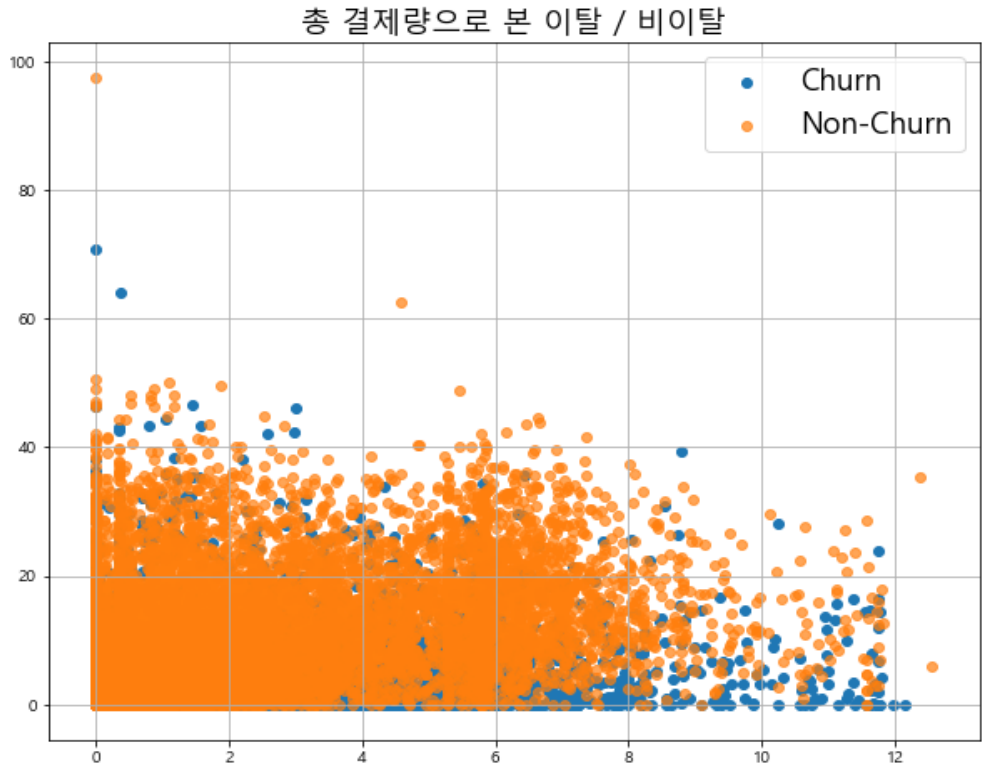
그러나, 일정 과금 수준별로 cut했으나 잔존 유저의 경우 조금씩 결제를 해도 64일 동안 결제가 진행되기 때문에 threshold로 나눴을 때 딱딱 분류되지 않는 모습을 확인할 수 있었다.

02

Feature & EDA.

User Churn을 어떻게 방지할까?





- 일 평균으로 보면 과금 유저의 결제 량이 압도적으로 높은 것을 확인할 수 있으나
- 총 결제량으로 볼 경우, 기간이 가미되어 차이를 확인할 수 없었다.

02

Feature & EDA.

최종 데이터 셋 및 라벨

X_train

(40000, 366)

Preprocessed Data

y_train1 잔존 기간 (40000, 1)

y_train2 일 평균 결제 량 (40000, 1)

03

Model.

어떤 방향으로 모델을 구축할 것인가?

Feature EDA

이탈 징후를 보이는 유저의 특징을 탐색

Modeling

이탈 유저를 판별, 보상을 하기 위한 모델 구축

Conclusion

구축한 모델 성능 개선 및 결과, 향후 방향성 제시

03

Model.

어떤 방향으로 모델을 구축할 것인가?

$$Objective : Maximize\ E(r)$$

$$E(r) = (T \times \hat{R}) \times \gamma - \hat{C}$$

$$where\ T = \begin{cases} 0, if\ \hat{t} = 64\ or\ t = 64 \\ 30 \times e^{-\frac{(t-\hat{t})^2}{2\times 15^2}}, otherwise \end{cases}$$

$$\hat{R} = predicted\ amount\ spent$$

$$\hat{C} = 0.3 \times \hat{R}\ if\ \hat{t} < 64\ else\ 0$$

$$\gamma = conversion\ rate$$

$$T \times \hat{R}$$

생존 기간과 일 평균 과금 량

예측 모델을 각각 구성!

03

Model.

Baseline 모델 – Data

- Label만 다른 두 데이터셋 생성
- 일 평균 결제 량은 결제 량이 0인 유저 배제

DataSet1

(40000, 366)

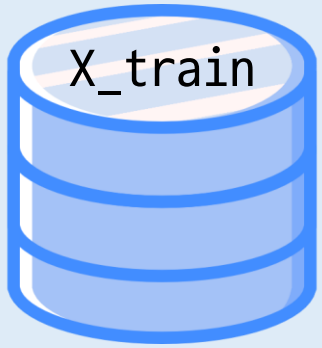
Preprocessed Data

y_train1

잔존 기간

(40000, 1)

DataSet2



23562

(40000, 366)

Preprocessed Data

23562

y_train2

일 평균 결제 량

(40000, 1)

03

Model.

Baseline 모델 – TrainTestSplit

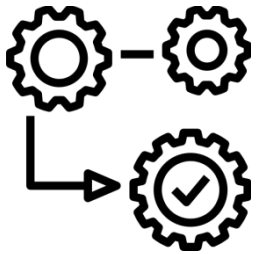
DataSet1

(40000, 366+1)

DataSet2

(23562, 366+1)

Train-Test-Split



8 : 2

DataSet1

X_train	y_train1
(32000, 366)	(32000, 1)
X_test	y_test1
(8000, 366)	(8000, 1)

DataSet2

X_train	y_train2
(18850, 366)	(18850, 1)
X_test	y_test2
(4712, 366)	(4712, 1)

03

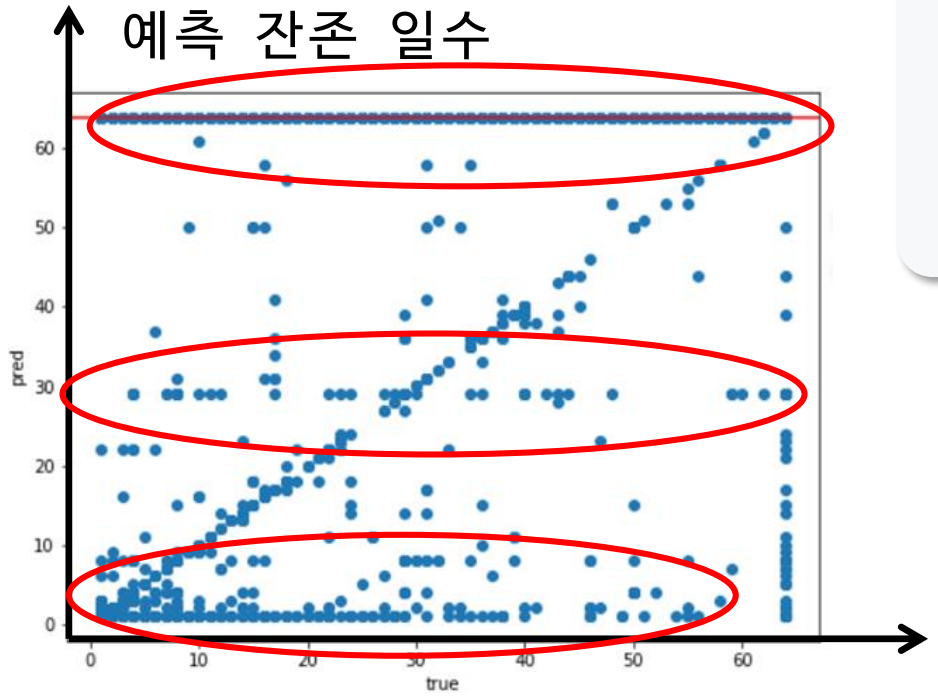
Model.

Baseline 모델 – 잔존 기간 예측

DataSet1

생존 기간
예측 모델

XGBClassifier




```
# 잔존 기간 모델 파라미터 (다중분류)
xgb_params = {
    'learning_rate' : 0.02,
    'gamma' : 0.3,
    'min_child_weight' : 3,
    'nthread' : 15,
    'max_depth' : 30,
    'subsample' : 0.9,
    'eval_metric' : 'merror',
    'colsample_bytree' : 0.8,
    'n_estimators' : 500,
    'max_leaves' : 300,
    'objective' : 'multi:softprob',
    'num_class' : 64
}
```

- 빈도가 제일 많던
  1~4일, 28일, 64일이라고
  예측한 오답의 수가 많았다.

03

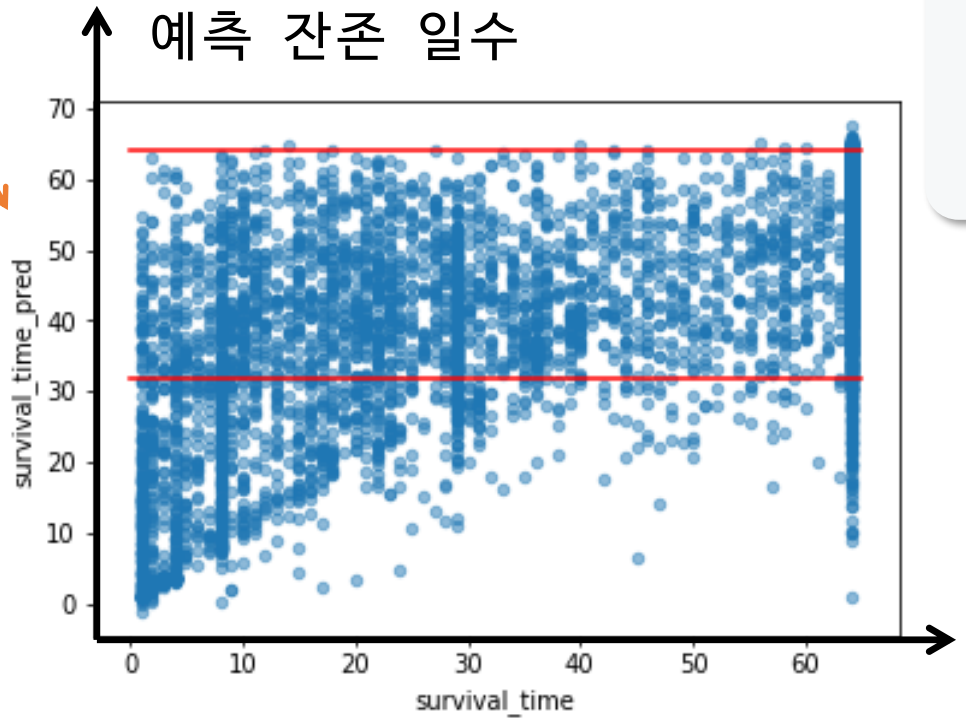
Model.

Baseline 모델 – 잔존 기간 예측

DataSet1

생존 기간
예측 모델

XGBRegressor




```
# 잔존 기간 모델 파라미터 (회귀)
xgb_params = {
    'learning_rate' : 0.01,
    'gamma' : 0,
    'min_child_weight' : 1,
    'nthread' : 15,
    'max_depth' : 10,
    'subsample' : 0.5,
    'eval_metric' : 'rmse',
    'colsample_bytree' : 0.6,
    'n_estimators' : 500,
    'max_leaves' : 300,
    'objective' : 'reg:squarederror'
}
```

- 빈도가 제일 많던
  1~4일, 28일, 64일이라고
  예측한 오답의 수가 많았다.

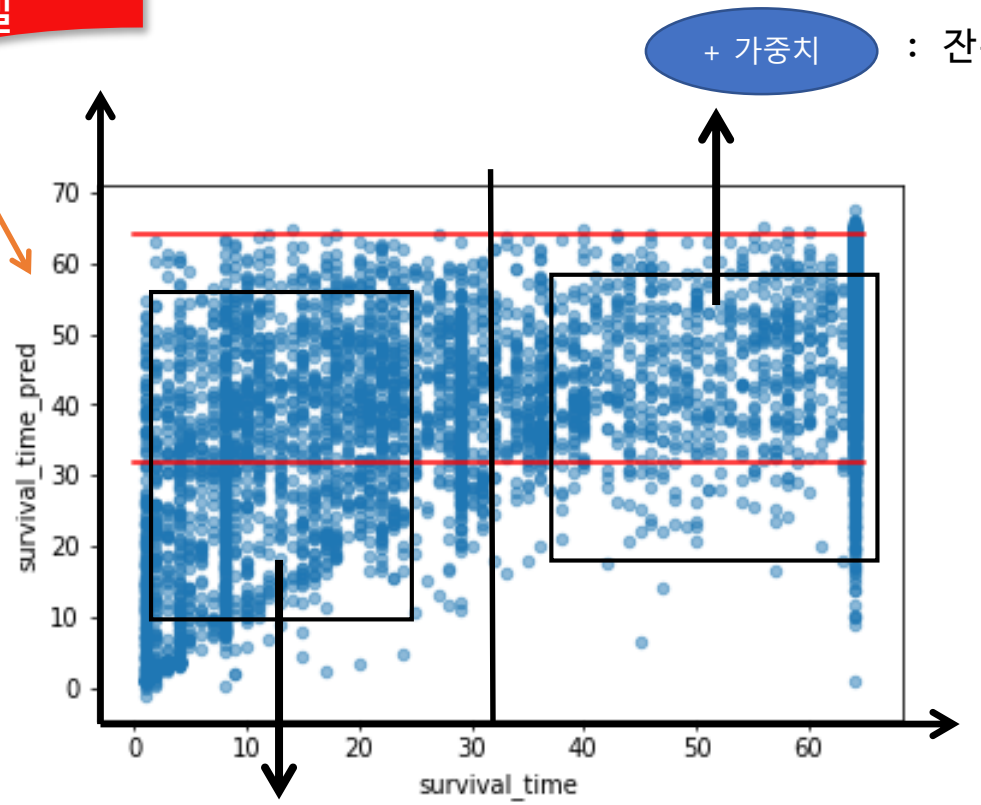
03

Model.

회귀 모델 – 잔존 기간 예측

생존 기간
예측 모델

XGBRegressor



- 여러 방법을 써봤지만 예측 오차를 줄이는 데에 한계가 있었다
- 가장 기대이익이 높은 생존기간 1인 유저와, 기대이익이 0인 잔존 유저를 최대한 가려내기 위해 가중치를 적용했다.
- 특히 64일의 예측 능력이 매우 떨어지므로 64인 경우 가장 큰 가중치를 준다면 모델의 성능이 좋아질 것이다.

03

Model.

회귀 모델 – 타겟 가중치

생존 기간
예측 모델

$$\text{가중치} = \begin{cases} -: (y-32)\cdot\ln(32-y), & if\ y < 32 \\ +: (y-32)\cdot\ln(y-31), & if\ y \geq 32 \end{cases}$$

- 총 생존기간 64의 중간인 32일을 기점으로 32일과의 차이가 클수록 큰 가중치가 적용되도록 설계
- 점진적으로 큰 가중치 적용을 위해 자연로그를 사용
- 32일 미만이면 음의 가중치를, 32 이상이면 양의 가중치 적용

예시)

survival_time	w_survival_time	survival_time_pred
35	4.158883	36.048416
9	-72.116367	93.886017
10	-68.002934	1.269660
44	30.779392	23.257292
49	49.136320	38.401604
64	111.888242	104.129059

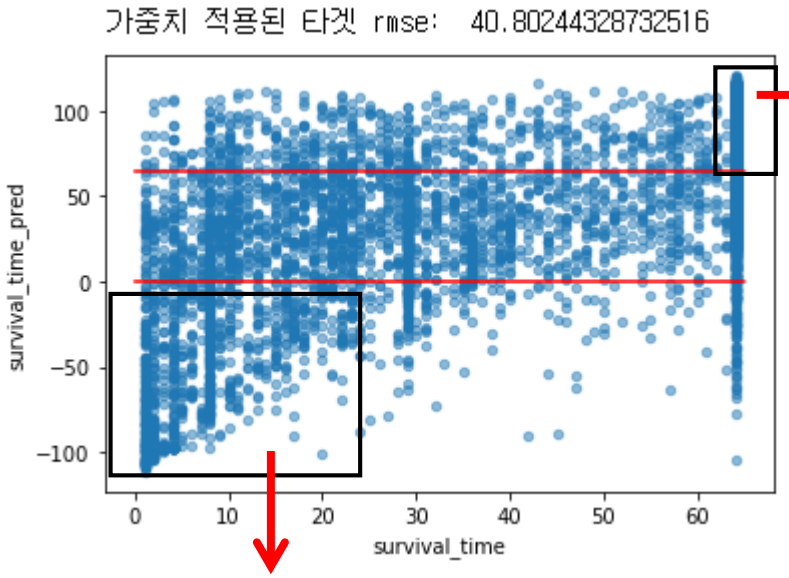
03

Model.

회귀 모델 – 가중치 적용 결과

생존 기간 예측 모델

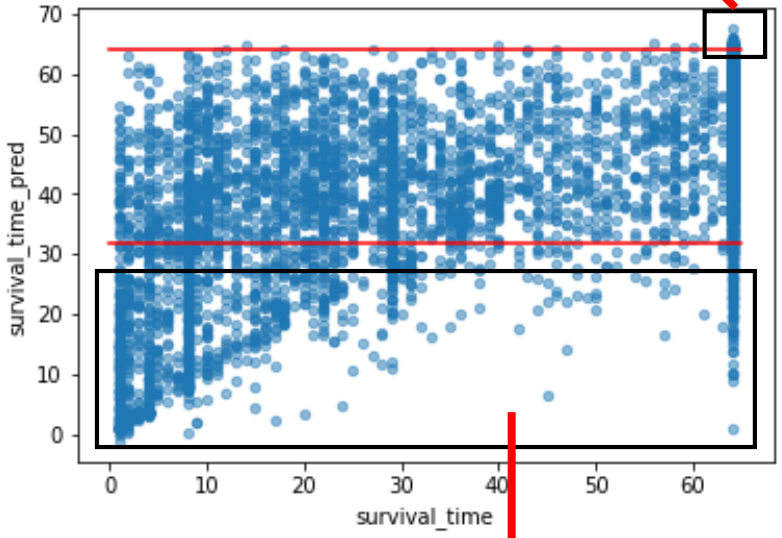
가중치 적용한 모델



잔존유저의 예측값 분포가 올라감

0 이하로 예측된 유저의 대부분이 생존기간 25이하

기존 모델



1~21일 사이에 잔존 유저의 비율이 높음

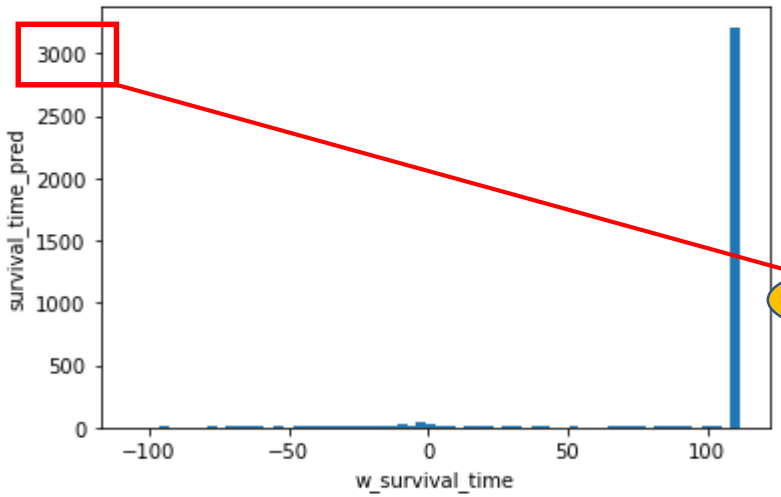
03

Model.

Baseline 모델 – 가중치 적용 결과

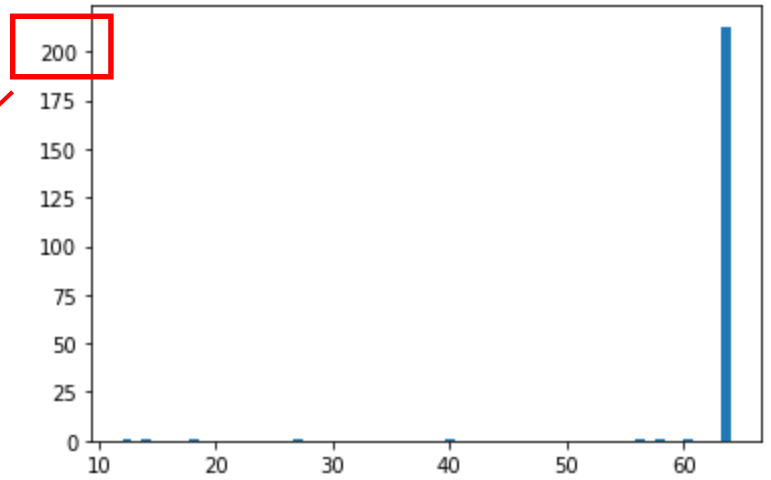
생존 기간 예측 모델

가중치 모델의 예측값 64이상의 실제 생존기간 빈도

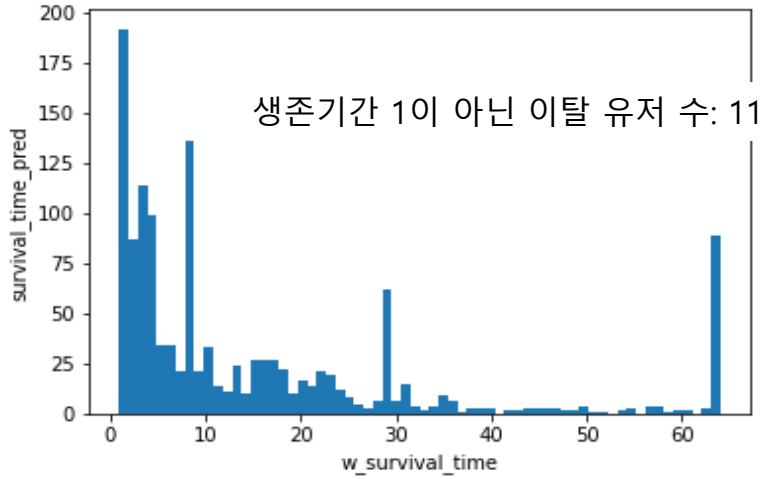


성능 향상

기존 모델의 예측값 64이상의 실제 생존기간 빈도

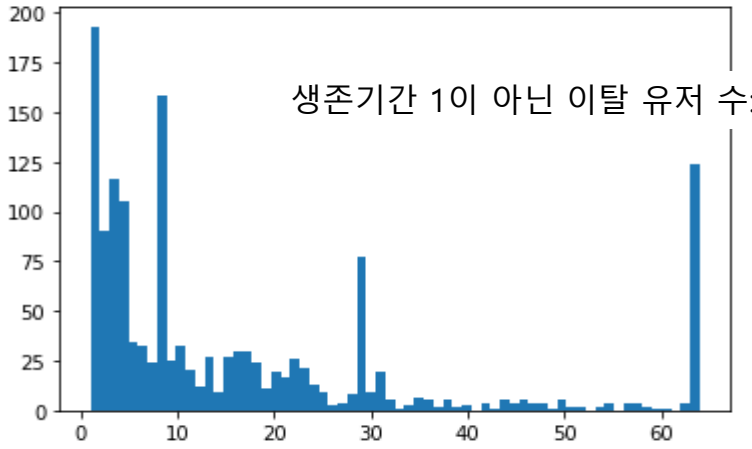


가중치 모델의 예측값 0 미만의 실제 생존기간 빈도(1302명)



생존기간 1이 아닌 이탈 유저 수: 1110명

기존 모델의 예측값 32 이하의 실제 생존기간 빈도(1439명)



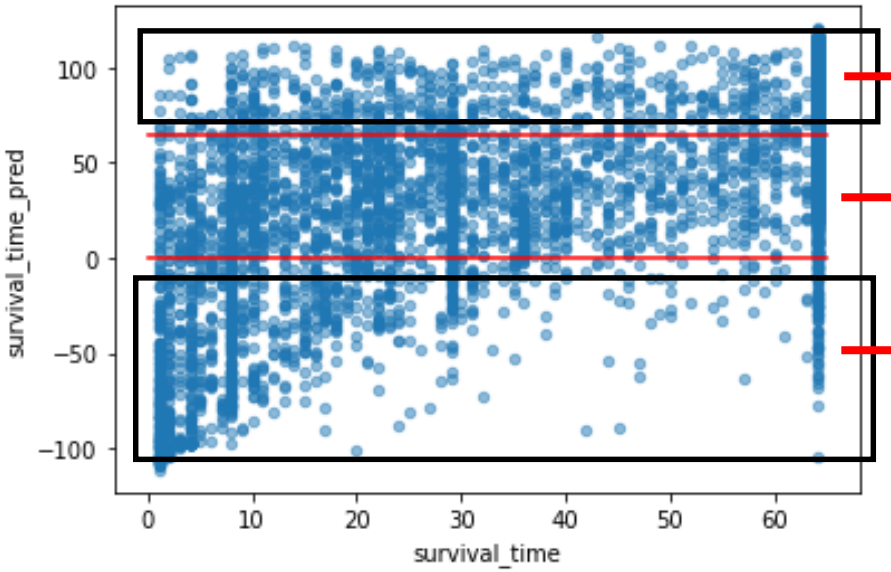
생존기간 1이 아닌 이탈 유저 수: 1300명

03

Model.

회귀 모델 – 가중치 적용 결과

생존 기간
예측 모델



64 이상으로 예측된 값은 생존기간 64로 변경

변경 없음

0 미만으로 예측된 값은 생존기간 1로 변경

Test 데이터 기대이익 향상

03

Model.

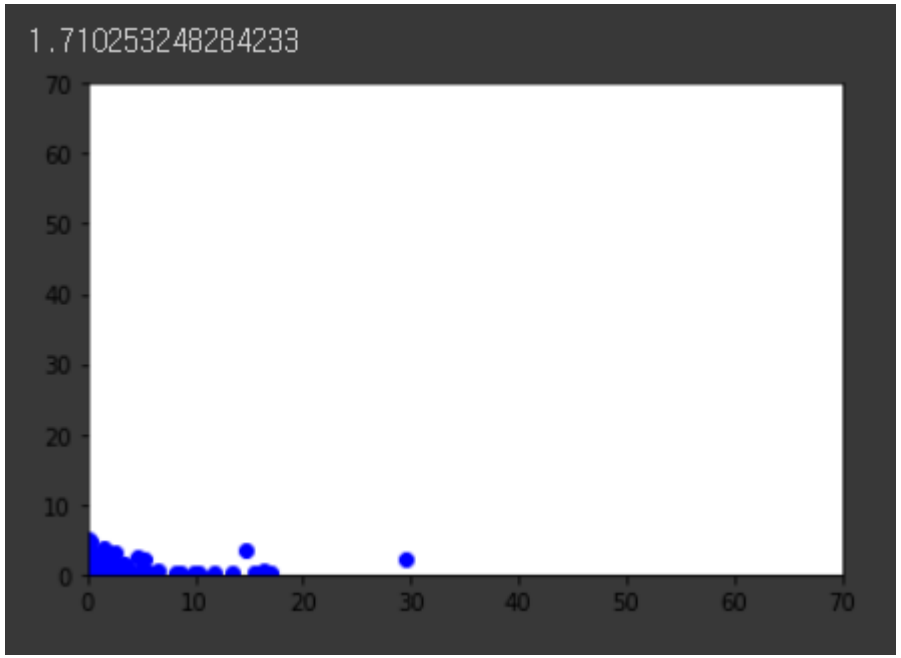
Baseline 모델 –일평균결제량 예측

DataSet2

일평균결제량 예측 모델

XGBRegressor

예측 일평균결제량



실제 일평균결제량

```
# 일평균결제량 모델 파라미터
xgb_params = {
    'learning_rate' : 0.02,
    'gamma' : 0.3,
    'min_child_weight' : 3,
    'nthread' : 15,
    'max_depth' : 30,
    'subsample' : 0.9,
    'eval_metric' : 'merror',
    'colsample_bytree' : 0.8,
    'n_estimators' : 500,
    'max_leaves' : 300,
    'objective' : 'multi:softprob',
    'num_class' : 64
}
```

- Rmse를 최저로 할수록 모든 값이 0에 가까워졌다.
- 무과금 유저 및 저과금 유저가 상당수이기 때문

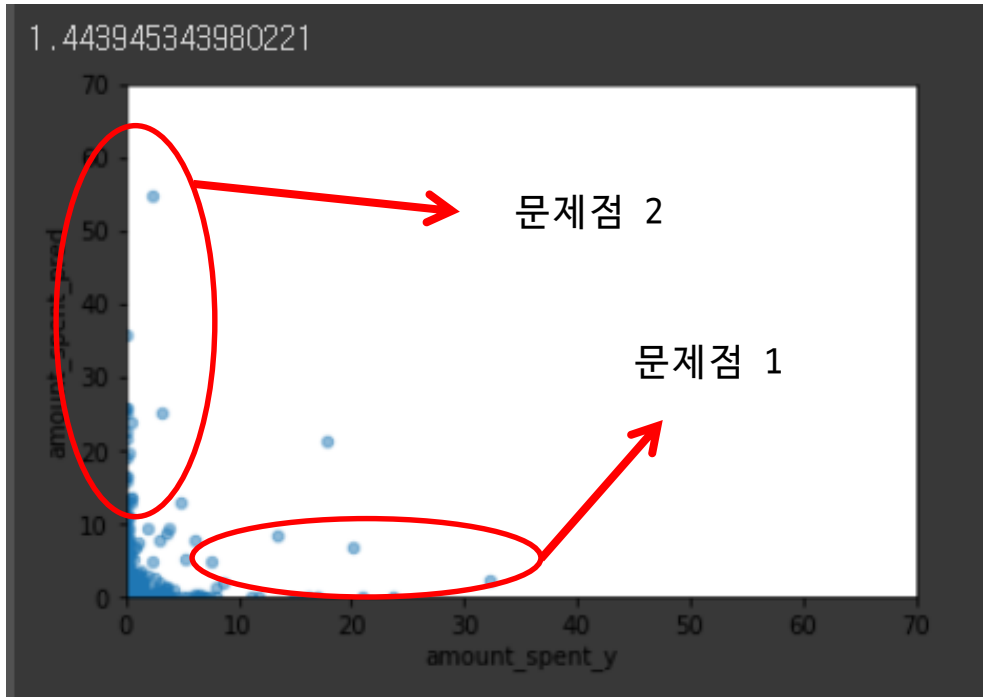
03

Model.

회귀 모델 –가중치 적용

일평균결제량 예측 모델

가중치 = $1.6 \cdot y \cdot \ln y$



- 기대이익이 높은 고과금 유저에 대한 결제량 예측 성능을 높이기 위해 가중치 적용
- 높은 일평균결제량에 대한 높은 가중치를 주기 위해 자연로그 사용

그러나

- 무과금 유저 때문에 가중치를 적용해도 예측값이 0에 가까움(문제점 1)
- 이를 보완하기 위해 가중치를 높이면 일평균결제량이 0 부근인 경우에 대한 오차가 매우 커짐(문제점 2)

03

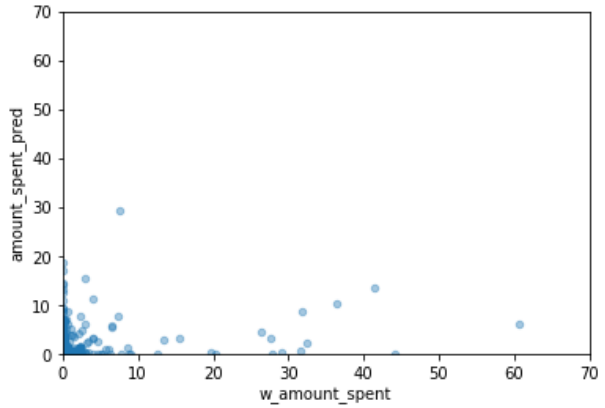
Model.

회귀 모델 –가중치 적용: 과금 유저 대상

일평균결제량 예측 모델

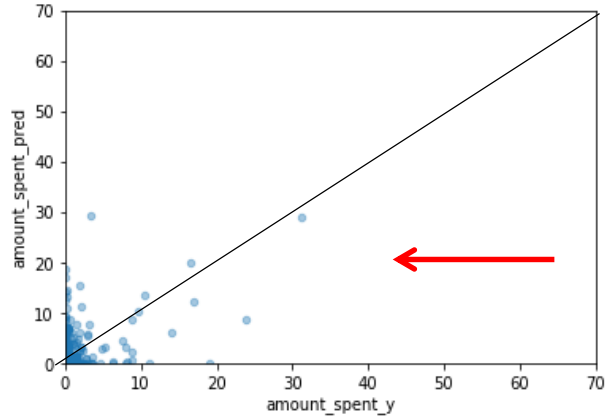
- X축: 가중된 타겟, y: 가중된 타겟 예측값

가중치 적용 라벨 rmse: 3.7992352026295313



- X축: 타겟, y: 예측값

실제 타겟의 rmse: 1.1680619619942905



- 문제점 1, 문제점 2를 해결하기 위해 과금 유저만을 대상으로 타겟에 가중치를 적용하여 모델 생성
- 가중된 타겟을 본래 타겟으로 변경했을 때
  - 낮은 값은 거의 미동이 없음 (가중치가 작기 때문)
  - 높은 예측값은 오른쪽에서 왼쪽으로 이동
- 그 결과, 높은 일평균결제량의 예측 성능 향상

예시)

acc_id	amount_spent_y	w_amount_spent	amount_spent_pred
24702	23.952066	123.284410	8.745134
51116	16.940902	78.255670	12.422235
65542	31.252524	173.693844	29.023355
89231	14.003362	60.679913	6.099175
92683	19.138915	91.948061	0.078109
111027	16.638483	76.406125	20.233248

03

Model.

selectfrommodel

- Label만 다른 두 데이터셋 생성
- 일 평균 결제 량은 결제 량이 0인 유저 배제

Feature 81

```
'playtime_count', 'solo_exp_count',
'quest_exp_count', 'fishing_count',
'game_money_change_count', 'login_clt', 'deff_level',
'level_max', 'day_count', 'char_count', 'day_sum',
'level', 'minus_am', 'plus_am', 'num_trade_s',
'num_trade_ex', 'num_trade', 'sell_amount',
'amount_diff', 'item_5_s_m', 'type_ex_s_m',
'trade_ratio', 'tday_count_t', 'source_count',
'day_y', 'random_defender_cnt_y', 'log_in_freq',
'a_server_num', 'b_server_num',
'combat_char_cnt_main', 'combat_play_time_main',
'pledge_member_num_main', 'pledge_member_num_sub',
'playtime_1', 'npc_kill_1', 'solo_exp_1',
'party_exp_1', 'private_shop_1', 'playtime_2',
'npc_kill_2', 'solo_exp_2', 'party_exp_2', 'death_2',
'fishing_2', 'playtime_3', 'solo_exp_3',
'party_exp_3', 'private_shop_3', 'playtime_4',
'npc_kill_4', 'solo_exp_4', 'party_exp_4',
'rich_monster_4', 'fishing_4', 'private_shop_4',
'game_money_change_4', 'forgive_4', 'pledge_cnt_3',
'num_opponent_4', 'play_char_cnt_1',
'combat_char_cnt_1', 'etc_cnt_1_y',
'combat_play_time_1', 'combat_char_cnt_2',
'pledge_combat_cnt_2', 'random_defender_cnt_2_y',
'temp_cnt_2_y', 'combat_play_time_2',
'play_char_cnt_3', 'combat_char_cnt_3',
'random_defender_cnt_3_y', 'temp_cnt_3_y',
'combat_play_time_3', 'pledge_rank_3',
'combat_char_cnt_4', 'temp_cnt_4_y', 'etc_cnt_4_y',
'combat_play_time_4', 'pledge_rank_4',
'amount_spent_1', 'amount_spent_4'
```

Feature 104

```
'playtime', 'quest_exp', 'game_money_change',
'playtime_count', 'npc_kill_count', 'solo_exp_count',
'quest_exp_count', 'game_money_change_count',
'day_attack', 'day_temp', 'std_same', 'acc_std_same',
'level_max', 'day_count', 'char_max', 'day_sum',
'C_1', 'C_4', 'C_5', 'C_7', 'level', 'day_mean',
'day_x', 'amount_spent', 'pay_count', 'pay_mean',
'non_login_pay', 'minuss', 'pluss', 'minus_am',
'plus_am', 'day_s', 'item_amount_s', 'item_6_s',
'item_amount_t', 'item_price_t', 'item_2_t',
'num_trade_s', 'num_trade_shop', 'num_trade',
'sell_amount', 'buy_amount', 'margin', 'amount_diff',
'day_s_m', 'item_amount_s_m', 'item_2_s_m',
'item_6_s_m', 'type_shop_s_m', 'day_t_m',
'item_price_t_m', 'day_s_nd', 'item_6_s_nd',
'type_ex_s_nd', 'day_t_nd', 'item_amount_t_nd',
'item_1_t_nd', 'item_2_t_nd', 'trade_ratio',
'hour_s', 'hour_mean', 'hour_s_700', 'hour_t_100',
'hour_t_300', 'source_sum', 'source_merchant_ratio',
'play_char_cnt', 'temp_cnt_y', 'pledge_rank',
'pledge_member_num', 'play_char_cnt_main',
'random_defender_cnt_main', 'play_char_cnt_sub',
'same_pledge_cnt_sub', 'solo_exp_1', 'quest_exp_1',
'playtime_2', 'quest_exp_2', 'playtime_4',
'npc_kill_4', 'solo_exp_4', 'quest_exp_4',
'fishing_4', 'game_money_change_4',
'same_pledge_cnt_1_x', 'pledge_cnt_4',
'play_char_cnt_1', 'combat_play_time_1',
'play_char_cnt_2', 'play_char_cnt_3',
'combat_char_cnt_3', 'random_defender_cnt_3_y',
'temp_cnt_3_y', 'etc_cnt_3_y', 'combat_play_time_3',
'play_char_cnt_4', 'pledge_rank_4', 'amount_spent_1',
'amount_spent_3', 'amount_spent_4', 'week1_log',
'week2_log', 'week3_log', 'week4_log'
```

03

Model.

최종 Process

일자	TEST1 점수	TEST2 점수	합계
2019-09-06 04:43:26 PM	10748.1	3068.55	13816.6
2019-09-06 12:21:01 AM	10925	2539.09	13464.1
2019-09-06 10:23:08 AM	10925	2514.49	13439.5
2019-09-06 10:28:52 AM	10917	2514.49	13431.5
2019-09-06 01:34:08 AM	10914.9	2440.85	13355.7

DataSet1

X_train	y_train1
(32000, 81)	(32000, 1)
X_test	**y_test1**
(8000, 81)	(8000, 1)

생존 기간 예측 모델

$$\text{가중치} = \begin{matrix}(y-32)\cdot\ln(y-32), if\ y<32\\ (y-32)\cdot\ln(y-31),\ if\ y\geq 32\end{matrix}$$

$$\hat{y}=1, if\ \hat{y}\leq 0$$
$$\hat{y}=64, if\ \hat{y}\geq 64$$

DataSet2

X_train	y_train2
(18850, 104)	(18850, 1)
X_test	**y_test2**
(4712, 104)	(4712, 1)

일평균결제량 예측 모델

$$\text{가중치} = (1.6\cdot y\cdot \ln y)$$

$$\hat{y} = 0,\ \text{if}\ \hat{y} < 0$$

04

Conclusion.

결론 총총총.

Feature EDA

이탈 징후를 보이는 유저의 특징을 탐색

Modeling

이탈 유저를 판별, 보상을 하기 위한 모델 구축

Conclusion

구축한 모델 성능 개선 및 결과, 향후 방향성 제시

04

Conclusion.

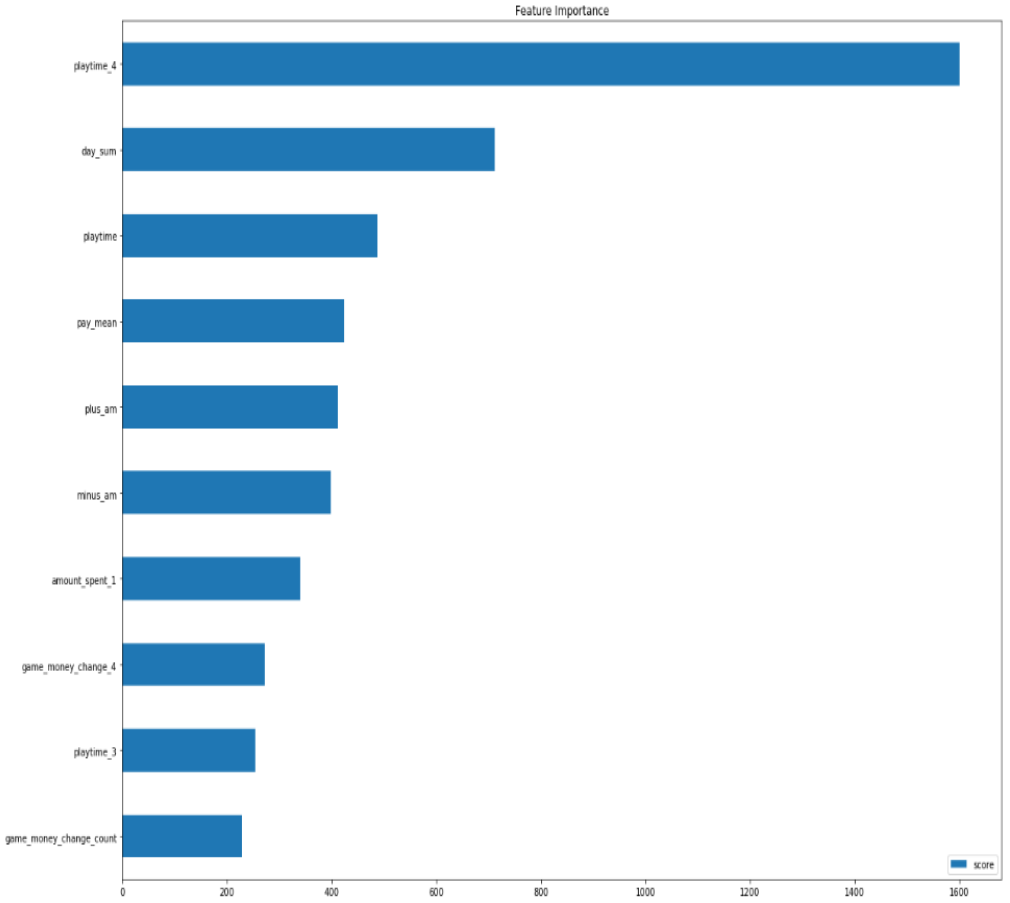
결론 총총총.

일평균결제량 결론

1. 접속 관련 특성
 - playtime_3, platime_4, day_sum
   : 마지막 주의 접속 여부가 중요

2. 결제 관련 특성
 - pay_mean, amount_spent_1
   : 월평균 결제량과 첫 주 결제량이 중요

3. 게임머니 관련 특성
 - plus_am, minus_am, game_money_change_4, game_money_change_count
   : 게임 머니의 양의 값, 음의 값, 마지막 주가 중요

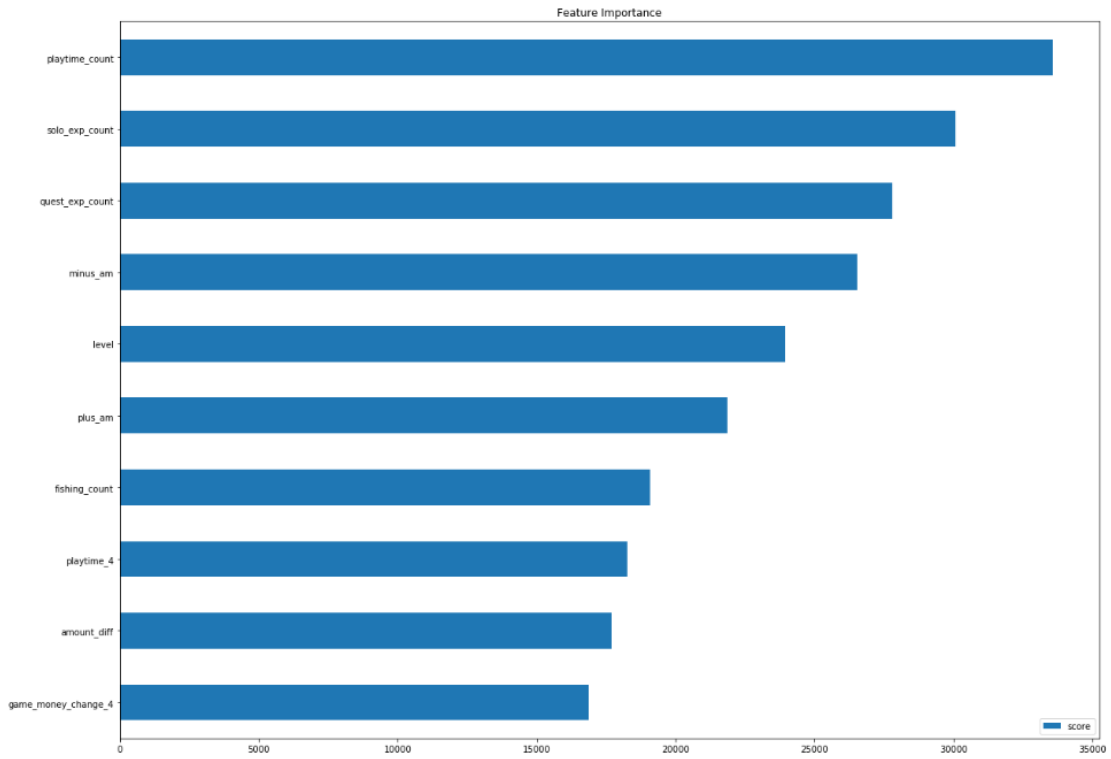


결론: 마지막 주에 아이템을 정리하고 초기에 이탈하는 유저에 맞는 특성들이 선택됨

04

Conclusion.

결론 총총총.



Result(이탈 원인 분석)

이탈 예측 model의 feature importance

1. 중요한 요인들의 특징을 살펴보면 새로 생성한 count 값들이 주요 변수로 작용 -> 게임 로그 데이터의 경우 시간의 변화, 대규모 패치 등에 따라 얼마든지 바뀔 수 있음 (시간에 강건하지 못함), 하지만 전체 대비 count들은 시간이 변해도, 게임 내 패치가 이루어져 전반적인 로그데이터의 수치가 변화해도 영향을 미치지 않음 -> 시간이 강건한 모델을 만드는데 중요한 작용을 함

2. 중요도 6위 이후로 보면 column명 뒤에 숫자가 붙은 변수들이 주로 등장함 -> 주차별 데이터들을 합산하여 flatten한 변수들 -> 시간적 변화가 중요하게 작용한다는 점을 알 수 있음

04

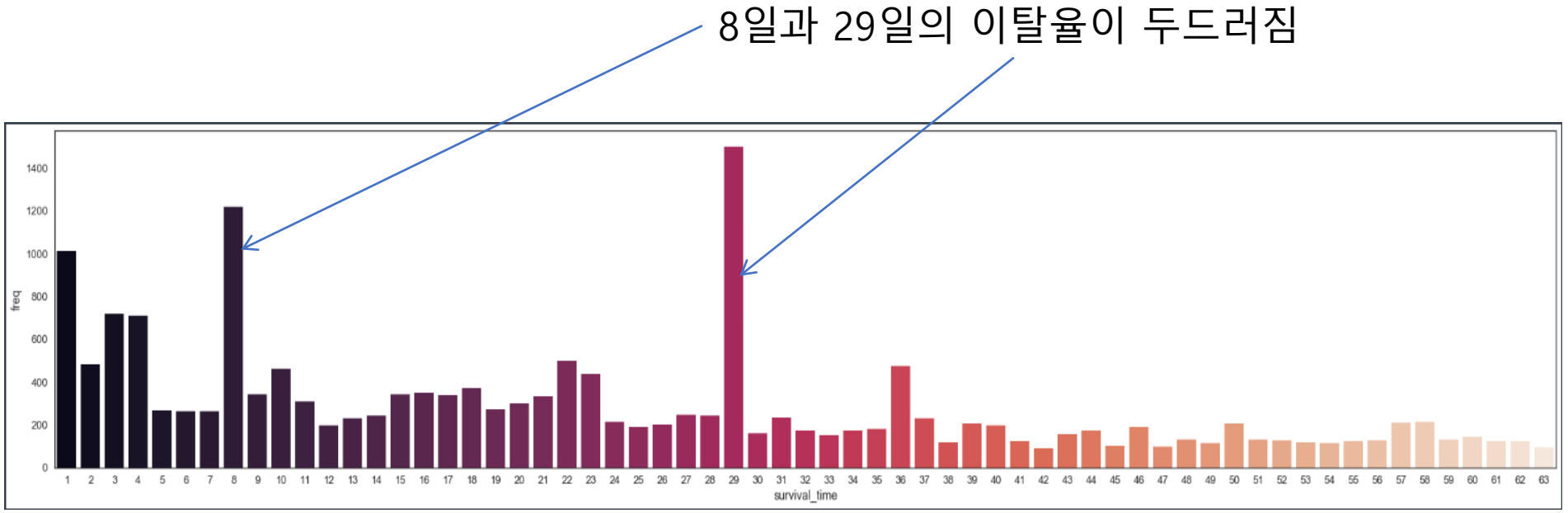
Conclusion.

결론 총총총.

변수 중요도에 따른 이탈 원인 추정 (Causality)

이탈 추정 이유

- 평소 이탈과 유입은 일정한 비율로 일어난다.
- 하지만 특정 시점에서 이탈율이 높아지는 경우가 있다.
- 이러한 구간의 원인을 분석하고 해결하지 못한다면 새로운 유저의 유입을 높여도 리텐션이 어려울 것이다.



04

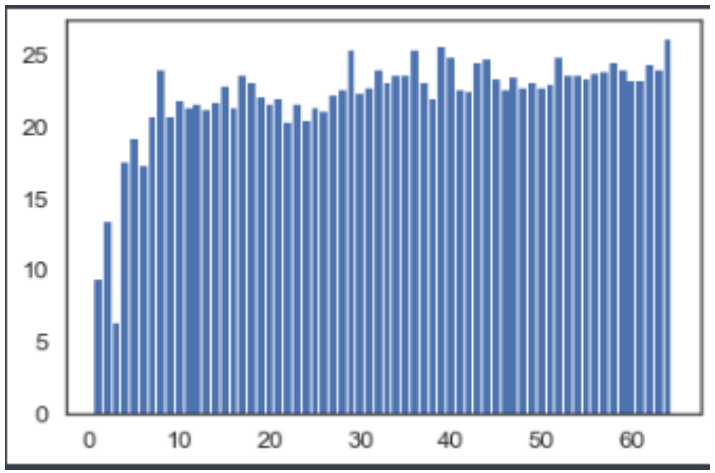
Conclusion.

변수 중요도에 따른 이탈 원인 추정

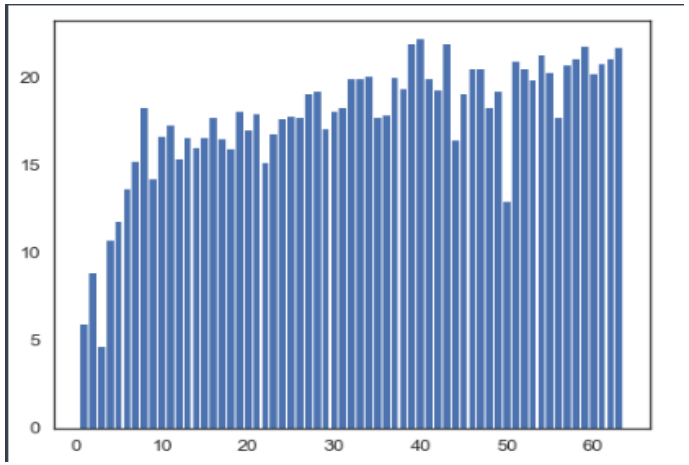
전반적으로 64일(잔존)으로 갈수록 상승하는 모습을 보여줌

하지만 이탈율이 높았던 8일과 29일의 특성은 확인하기 힘듬

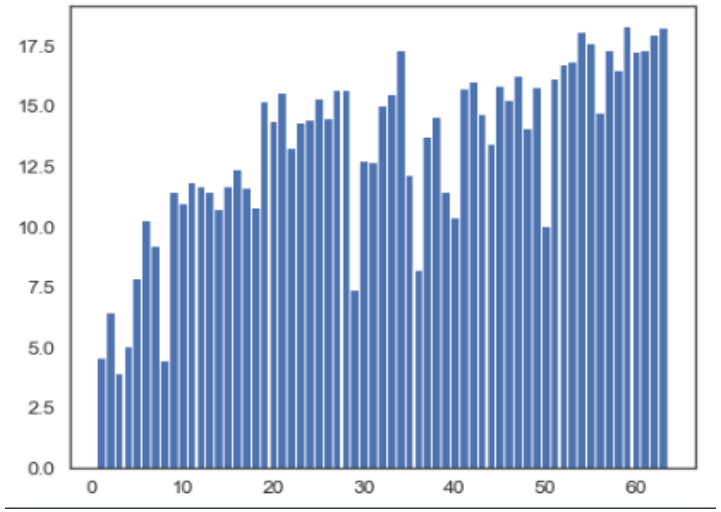
Rank.1 playtime_count



Rank.2 solo_exp_count



Rank.3 quest_exp_count

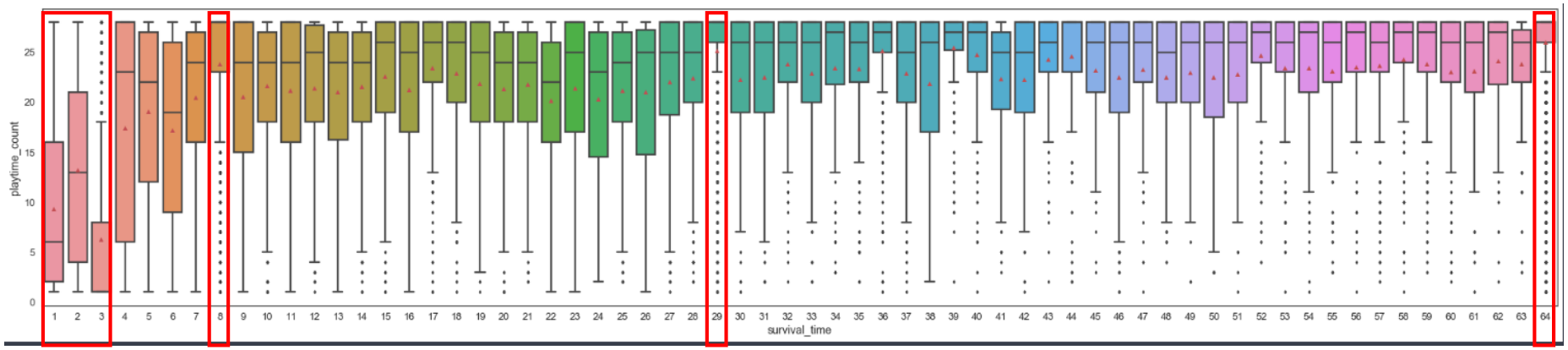


04

Conclusion.

변수 중요도에 따른 이탈 원인 추정

Rank.1 playtime_count



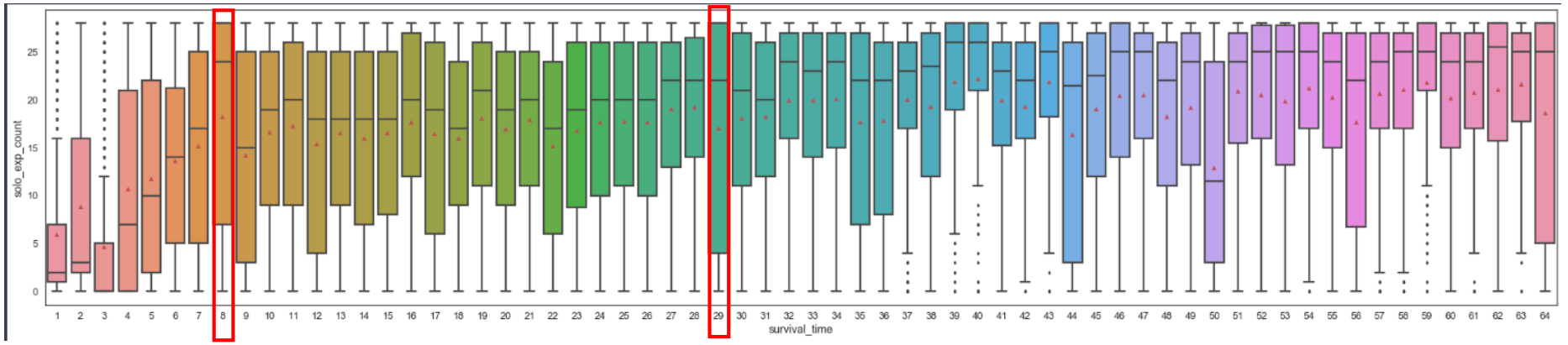
- 확실히 잔존에 가까울수록 playtime_count의 데이터 분포와 평균이 높음
- 8일과 29일의 이탈자가 실제로 상당수에 다다른다.
- 8일, 29일, 64(잔존)만 playtim_count의 분포가 주변 이탈일 대비 상당히 높다
- 그 외에 이탈율이 높았던 1일~3일의 경우 게임을 플레이 한 일 수가 타 군에 비해 현저하게 낮음을 볼 수 있다.
- 알 수 있는 점 : 이탈율이 높은 구간은 playtime이 현저하게 낮거나 현저하게 높다.
- 현저하게 낮은 구간은 리니지에 흥미를 못느끼거나 시간을 투자할 여건이 되지 못해 이탈한 것으로 추정
- 현저하게 높은 구간(8,29일)은 열심히 게임을 즐겼는데 왜 이탈율이 높은 것인가

04

Conclusion.

변수 중요도에 따른 이탈 원인 추정

Rank.2 solo_exp_count



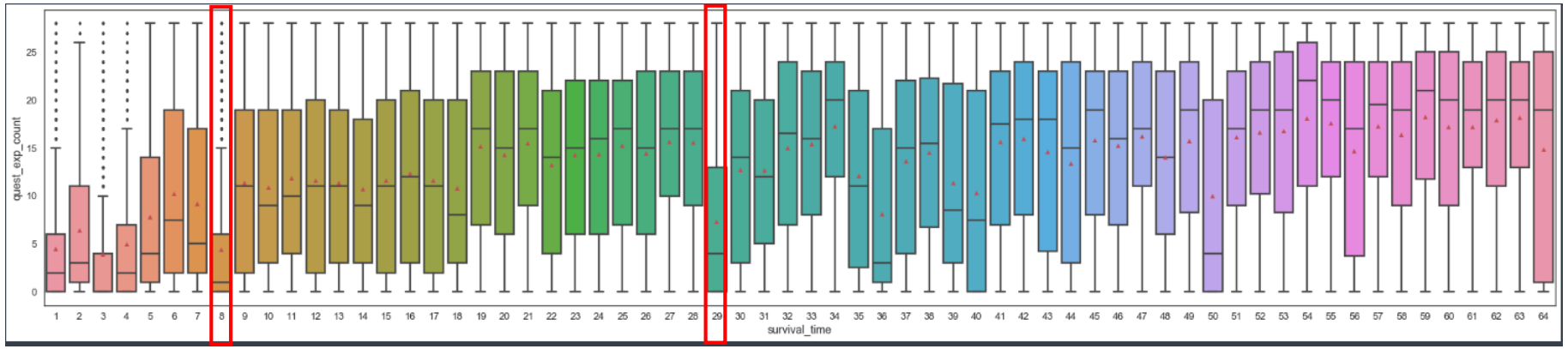
- 8일 이탈 유저들의 경우 혼자 사냥을 즐기는 횟수의 평균과 중앙값이 두드러지게 높다는 것을 확인할 수 있다.
- 29일 이탈 유저들의 경우 혼자 사냥을 즐기는 횟수는 특이점을 보이지 않았다.
- 29일의 이탈 원인은 다른 부분에 있을 것이다.

04

Conclusion.

변수 중요도에 따른 이탈 원인 추정

Rank.3 quest_exp_count



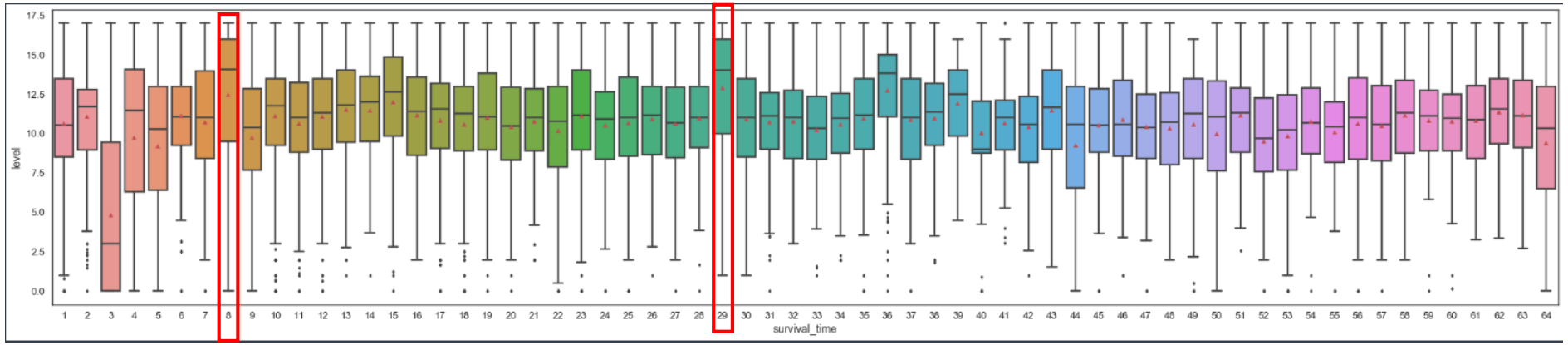
- 8일과 29일 모두 퀘스트를 진행 한 횟수가 상대적으로 낮다.
- 그렇다면 8일과 29일은 리니지 내 다양한 컨텐츠를 즐기지 않고 오로지 사냥에만 집중하여 흥미를 잃은 것인가?

04

Conclusion.

변수 중요도에 따른 이탈 원인 추정

Rank.5 level



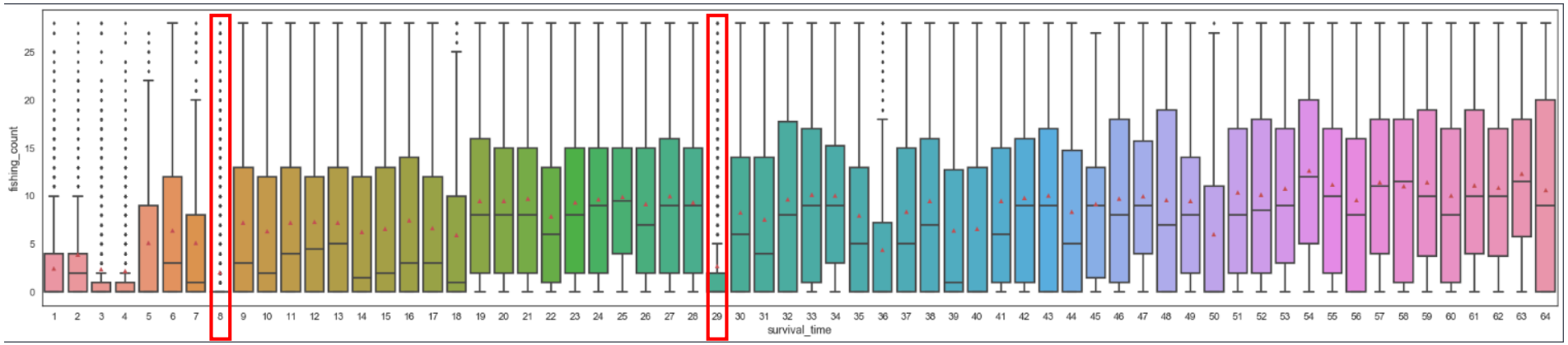
- 앞서 세운 가설을 기반으로 해석하자면 level의 분포만 보아도 상대적으로 높다는 것을 알 수 있다.
- 두드러지게 많았던 플레이 시간의 대부분을 level up에만 사용.
- 이탈율이 높았던 1~3일차를 보면 3일차 혼자 다른 분포를 띠는 것을 확인.
- 플레이 시간이 적다 하더라도 1,2일 이탈 유저는 level이 뒤쳐지지 않으나 3일에 이탈한 유저의 경우 level이 상당히 낮은 것을 확인 할 수 있다. -> 게임 자체에 숙련도 및 진입 장벽과는 무관하고 단순히 할여할 수 있는 시간 부족이 이탈 원인으로 추정

04

Conclusion.

변수 중요도에 따른 이탈 원인 추정

Rank.7 fishing_count



- 리니지라는 게임 특성상 현실에 가까운 다양한 컨텐츠를 반영함으로써 유저의 충성도를 높이는 장점을 보유하고 있는데 8일과 29일 이탈 유저의 경우 이러한 컨텐츠를 전혀 즐기지 않는 것으로 확인되었다.

04

Conclusion.

결론 및 이탈 대응책

결론

- 1~3일차 이탈 유저의 이탈 원인은 타 class에 비해 게임에 할여할 수 있는 시간이 상대적으로 적음을 알 수 있다.
- 3일차의 경우 실제로 게임에 투자할 수 있는 시간이 부족하여 이탈하는 경우이지만 1,2일차의 경우 혈맹에 가입한 비율(사회성 평가 지표)이 높고 level up속도 또한 평이한데 이탈율이 높다는 점은 리니지 자체 시스템에 대한 숙련도 부족, 진입장벽 등이 원인으로 추정된다.
- 8일과 29일의 경우 플레이 시간과 솔로 경험치 획득량이 굉장히 높지만 fishing, quest 등의 컨텐츠 진행율이 굉장히 낮다.
- 이 또한 자율성이 높은 게임을 온전히 즐기지 못한다는 점이 이탈 원인으로 보인다.
- 특히 정액제(30일 또는 300시간)를 고려했을 때 29일은 다음 달의 이탈 여부를 결정하게 되는 중요한 시기이다.
- 8일 또한, 리니지 게임 내 다양한 이벤트(신규 유저 3, 7일 무료, 코카콜라 포인트 제공 등)를 통해 추가 플레이 후 2번째 정액제를 신청하게 되는 시점이다.
- 따라서, 1~3일, 8일, 29일이 유저 리텐션을 결정 짓는 전략적 시기이다.

이탈 대응책

- 이탈 유저의 공통적인 특징은 다양한 컨텐츠를 즐기지 못하고 플레이 시간에 대부분을 사냥에 집중한다는 점과 솔로 사냥 시간이 상대적으로 높다는 점이다.
- 먼저, 다양한 컨텐츠를 즐길 수 있게 레벨 구간 별 가이드 라인을 제공한다면 복잡한 구조의 게임에 대한 진입장벽을 낮출뿐만 아니라 플레이의 다양성을 가져와 리텐션 및 충성도를 높일 수 있을 것이다.
- 이외에도 혈맹뿐만 아니라 파티와 커뮤니티 시스템을 강화하고 추천한다면 타인과의 자연스러운 교류를 통해 새로운 정보들을 취합하고 색다른 즐거움을 선사할 수 있다고 판단한다.