###### **1페이지**

안녕하세요, 팀 **From Bigcon Import Prize**의 발표를 맡게된 배지원입니다.

###### **2페이지**

저희 발표는 크게 세 파트로 이루어져 있습니다. 먼저 분석 파이프라인을 통해 저희 분석의 개요를 보여드리고, 이를 어떻게 전처리를 하고, 모델을 구축하였는지에 대해 전반적인 진행을 하겠습니다.

###### **3페이지**

먼저 분석 파이프라인에 대해 설명드리도록 하겠습니다.

###### **4페이지**

저희는 **Robust한 결과물**을 제시할 수 있는 방향에 초점을 두어, **시간의 흐름에 따른 영향이 최소화 되도록** 하는 여러 장치를 만들었습니다.

먼저, 데이터적으로는

survival time의 범주를 축소하고, (64 to 11) as를 tas로 재정의 후,

각 target value를 구간을 기준으로 under sampling을 하였습니다.

또한, weekly aggregate의 방법을 사용해 데이터 변환을 실시했습니다.

모델링 단계에서는 **out of fold**라는 CV전략을 이용함으로써, Rubust함을 추구하였습니다.

###### **5페이지**

다음으로 데이터 전처리에 대한 파트입니다.  
데이터 전처리는 크게 데이터 트랜스포메이션과 피쳐엔지니어링의 파트로 나뉘는데요,

모델을 생성하기에 앞서 어떻게 데이터를 변환하였으며, 어떠한 변수를 생성하였는지 말씀드리도록 하겠습니다.

###### **6페이지**

먼저 분석 목적은

적정한 마케팅 비용을 산출하여 이탈할 고객을 생존고객으로 전환 시키기 위함이였습니다. 즉, 이탈할 고객이 얼마나 소비할지에 대해 파악하는 것이 관건이었습니다.

이러한 목적을 이루기 위해서는 Survival time이 64 미만인 유저가 사용할 amount spent 값을 잘 예측하는 것이 중요한데요!

이를 위해서는 주어진 데이터의 세 가지 이슈를 해결해야 했으며,

이는 각각

**target value의 불균형,**

**Data agg의 방식**

**as의 data leakage문제 였습니다.**

차근차근히 각 이슈들과 해결 방법에 대해 말씀드리도록 하겠습니다.

###### **7페이지**

유저 생존기간과 일평균 결제금액의 분포 그래프를 보시면 알 수 있듯이, 저희가 가지고 있는 데이터는 skewed합니다.

유저들은 대부분 survival time이 64로, 즉 대부분 생존하고, amount\_spent는 0으로 소비하지 않는다는 특징을 지닙니다.

데이터가 불균형하였기에 전처리없이 모델을 만들었을 때, 결제금액은 0으로 생존기간 64로 예측하는 문제가 발생하였습니다. 이에 대한 해결책은 이후 이야기드리도록 하겠습니다.

###### **8페이지**

또한 유저의 행동패턴을 정리하기 위해, 데이터를 어떤 방식으로 aggreating 하는 지에 대한 이슈가 있었습니다. Month, Day, week의 3가지의 선택지가 있었는데요,

먼저,

한 달(4주)를 기준으로 유저 데이터를 agg하는 경우 정보 손실이 많고, 저희에게 주어졌던 train label data의 유저 수가 4만명 이었기 때문에 이를 그대로 사용하면 데이터의 수가 작았습니다. 이 때문에 차원의 저주가 발생할 가능성이 있어, 이를 채택하지 않았습니다.

###### **9페이지**

또한,

일간 데이터 기반 예측 시, 유저의 매일 매일의 행동정보를 반영할 수 있지만, 공휴일, 이벤트, 유저 개인 사유 등 지나치게 다양한 노이즈가 반영될 수 있었습니다.

**그리하여, 절충안으로 Weekly 분석을 선택하였습니다.**

리니지는 매주 패치를 진행하기 때문에, 유저들이 패치일자를 기준으로 플레이스타일이 변하는 경향이 있습니다.

그렇기에 패치를 반영하고, 유저 행동패턴의 경향성까지 확보할 수 있는 weekly를 기준으로 data를 merge하였습니다.

###### **10페이지**

이제 주간 데이터로 변환하는 과정에 대해 설명드리겠습니다.

survival time의 경우,   
각 주차별 유저의 생존일은 **실제 survival time에서 주간 시점의 유저 생존일을 더하면서 계산했습니다.**

45일을 생존한 31번 유저를 예를 들어 보자면, 실제 생존 시간을 기준으로 주차엔 28, 2주차엔 21, 3주차엔 14, 4주차엔 7 씩을 더해 새로운 survival time을 도출했습니다.

이렇게 주차별 새로운 survival time을 구하고, 이를 각각의 유저로 치환했습니다.

amount spent는 일별 payment를 합산해 주차별로 산출했는데요. 그러나 amount spent의 경우 survival time과 의존관계에 있기 때문에, total amount spent로 변수를 재정의하게 되었습니다. 이 부분은 다음 페이지에 설명 드리도록 하겠습니다.

###### **11페이지**

Y label의 이슈를 해결하기 위한 방안으로, 각각의 타겟변수를 재정의하였습니다.

survival time의 경우, 계산량과 언더샘플링의 효율화를 위하여, 카테고리를 64개에서 11개로 축소하였습니다.

amount spent는 유저의 평균 결제액, 즉 실제 결제한 금액을 생존시간으로 나눈 값으로 survival time에 따라 과소 혹은 과대추정되는 등 왜곡이 발생합니다. 이때문에 total amount spent로 변수를 변환하여, 주차별로 tas를 재할당하였습니다. 이 부분은 자세한 설명이 필요하기 때문에 궁금하신 분들은 QnA시간때 질문주시면 감사하겠습니다.

###### **12페이지**

앞서 말씀드린 타겟변수의 skewdness함을 해결하기 위해 under sampling을 진행하였고, 다음과 같이 변환됨을 보실 수 있습니다.

###### **13페이지**

이후, Feature engineering 파트에 도입하게 되는데요!

총 다섯가지의 아이디어를 바탕으로, 피쳐를 생성하였습니다.

첫번째로, 서버에 대한 정보를 반영하기 위해 서버 별로 원핫인코딩을 한 이후, acc-id와 week를 기준으로 sum하였습니다. 즉, 한 유저가 해당 주차에 어떠한 서버에 얼마나 많이 접속하였는지에 대한 변수를 엔지니어링한 것입니다.

###### **14페이지**

두번째로, 5개의 주어진 테이블. (Activity, trade, payment 등) 의 기본 feature에 대해, nunique, min, max 등의 다양한 방법을 적용하며, 요약 통계량 피처를 생성하였습니다. 이를 통해 다양한 특성에 대해 파악하고 정보손실을 최소화하였습니다.

###### **15페이지**

세번째로, 유저플레이 성향을 반영하는 변수들을 생성하였습니다.

sum(each feature) / playtime 로 나눈 파생 변수 인데요!

예를 들어, 일주일 내 동일한 solo\_exp를 획득한 유저라도, 각기 다른 playtime을 가진다면, 두 유저의 게임 스타일과 성향은 다르다고 할 수 있습니다. 이 때문에 전체 게임 플레이에서 해당 요소가 차지하는 비중을 반영할 수 있는 변수를 만들었습니다.

###### **16페이지**

네번째로

유저의 주차 별 행동양식 변수를 생성하였습니다. 각 feature별로 PCA를 시행하여 한 유저의 4주차 정보를 하나의 Component로 축소한 것인데요. 이를 통해 각각의 주차별 피쳐의 변화를 반영하고자 하였습니다.

예를 들어, playtime의 경우 음수일수록 total amount spent가 높은 경향을 보였습니다. 이렇듯 각 피쳐의 변화에 따라 유저의 생존기간과 결제금액에 영향을 미칠 것이라 판단하여 변수를 생성하였습니다.

###### **17페이지**

마지막으로, 유저가 언제 진입하였는지에 대한 정보를 주기 위하여 유저의 게임 시작 시기에 대한 라벨을 지정하였습니다. 기존의 1,2,3,4 주차로 agg하며, 잃게 된 이전 접속 정보를 전달하기 위하여 이러한 변수를 생성하게 되었습니다.

###### **18페이지**

fe를 통해 결과적으로 2600개의 column을 생성하여 이를 분석에 활용하였습니다.

###### **19페이지**

다음으로 모델링 파트입니다.

###### **20페이지**

저희는 LightGBM 모델을 앙상블하여 최종 모델을 생성하였는데요! 빠르고 효율적이며 높은 성능을 보이기 때문에 이를 선택하였습니다.

또한 CV 전략은 Out of fold라는 개념을 이용하였습니다.

oof란 기본적으로 kfold를 이용하는 방법으로써

트레인 셋을 서브셋으로 나눠 k개의 모델을 생성하고, 이를 train target value와 비교하면서, 점수를 산출하는 방법입니다.

저희의 train set과 test set을 구축한 과정은 다음과 같습니다.

train data는 1, 2, 3, 4주차의 유저들을 활동 기록을 agg함으로써 산출하였고 이를 이용하여 모델을 학습시켰습니다.

이렇게 학습시킨 모델에 대해, test data를 적합하였는데, 이 때 test data는 1, 2, 3주차의 정보는 사용하지 않았고, 4주차의 정보만을 agg하여 사용하였습니다. 1, 2, 3주차를 사용하지 않은 이유는, 주차별로 모델링이 되기 때문입니다.

###### **21페이지**

또한, 저희는 survival time을 분류하는 모델과 total amount spent를 예측하는 모델을 만들었습니다. lgb의 feature importance를 통해 각각의 피쳐를 선택하였습니다.

survival time의 경우,   
유의미한 변수로 유저가 한 주 간 az서버에 얼마나 접속하였는지, 평균 ao 서버에 접속량 등 대체적으로 서버에 대한 영향력이 높았습니다. 세번째 피쳐는 거래시각을 원핫-인코딩하여 만들어 놓은 변수로, 주차별로 13시 대에 거래를 수행한 최솟값의 행동양식입니다.

또한, tas의 경우에는 주간 ao 서버 총 접속량과 주별 최대 ad서버 접속량의 PCA값으로 유저의 행동양식에 대한 영향이 높았습니다.

###### **22페이지**

영향력 높은 변수에 따라 survival\_time과, amount\_spent가 어떻게 변화하는지에 대해 알아보도록 하겠습니다.

survival time의 경우, 접속시간이 적을 때, **접속시간 대비 npc킬의 비율이 낮을 때** 이탈하는 경향을 보였습니다.

total amount spent의 경우, 각 feature들의 영향력이 어떻게 발휘되는지는 추후 필요시 appendix를 통해 보다 상세히 보여드리도록 하겠습니다.

###### **23페이지**

ncsoft에서 제공한 모듈을 통해 score를 도출했을 때, **최종적으로 oof 기준 38,000점의 점수를 달성할 수 있었습니다.**

로버스트를 추구한 저희의 분석 전략이 잘 통했다면, test1과 test2도 이와 비슷한 결과가 나왔을 것이라고 생각합니다.

저희의 분석이 고객의 데이터에서 숨은 그림을 찾을 수 있는 인사이트가 되었으면 좋겠습니다.

###### **24페이지**

지금까지 고객의 행동 데이터에 숨겨진 잔존가치를 찾기위한 FBIP 팀의 여정이었습니다. 감사합니다.