머신 러닝 모델을 활용한 고객 대출신청 여부 예측

데이터분석 분야 - 퓨처스부문

류동엽

October 14, 2022

목차

분석 배경 및 요약

1. 분석 배경

- 평소 데이터 기반의 고객 행동 예측 분야에 관심
- 금융권 기업의 매출 향상에 직간접적으로 기여 가능한 고객의 대출상품 신청여부 예측을 본 대회를 통해 시도해 보고자 함

2. 분석 요약 및 실험 환경

1) 분석 요약

- 추가 데이터 없이 3개의 데이터 테이블을 모두 사용하여 분석을 수행
- 데이터 전처리 과정을 통해 모델의 예측성능을 향상
- OLS 기반의 통계 분석과 LightGBM 알고리즘을 사용하여 변수 선택
- LightGBM 알고리즘의 하이퍼 파라미터 튜닝 진행

2) 실험 환경

- Google Colaboratory(Colab) Pro 환경 하에서 실험 진행

	Colab Free	Colab Pro	Colab Pro +
Guarantee of resources	Low	High	Even Higher
GPU	K80	K80, T4 and P100	K80, T4 and P100
RAM	16 GB	32 GB	52 GB
Runtime	12 hours	24 hours	24 hours
Background execution	No	No	Yes
Costs	Free	9.99$ per month	49.99$ per month
Target group	Casual user	Regular user	Heavy user

3) 제출파일 요약

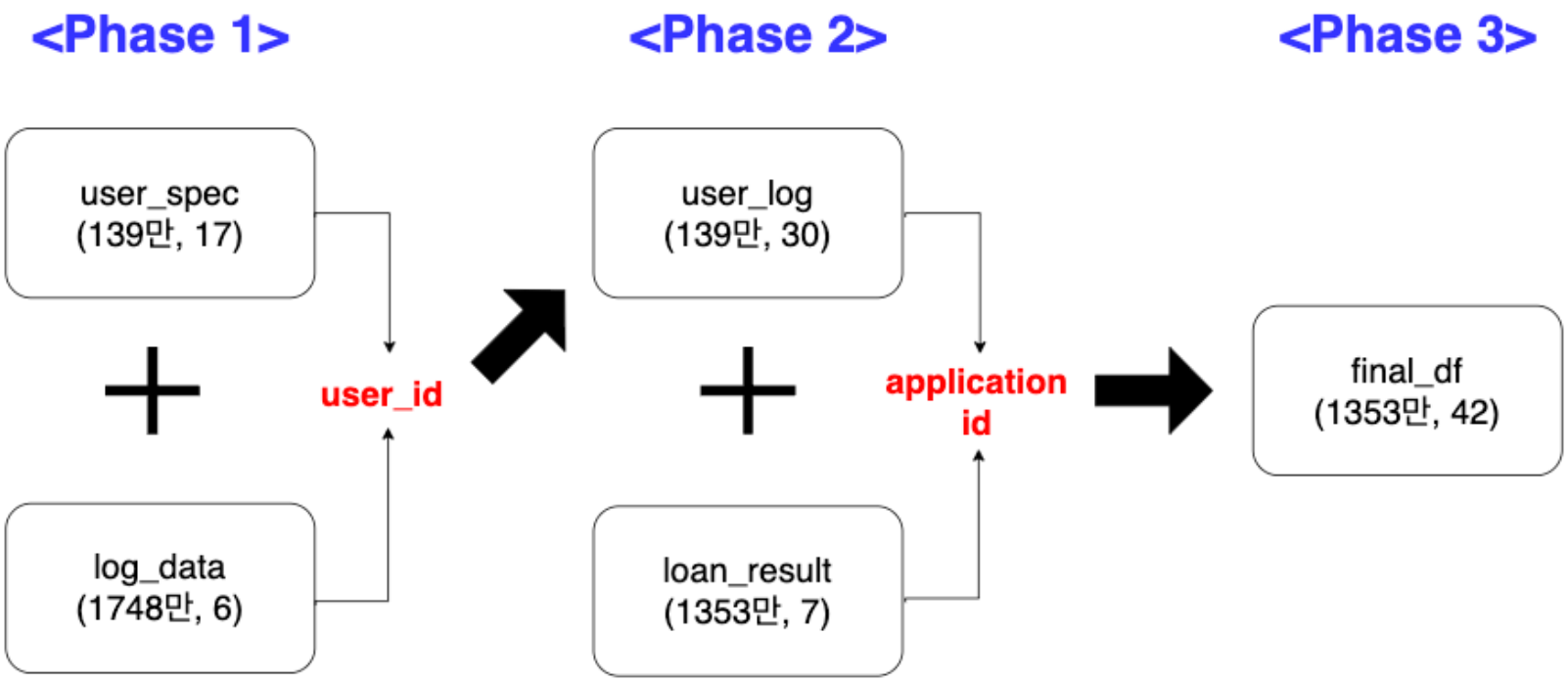
- 전처리, 모델링, 그리고 통계분석 코드는 .ipynb파일로 구성
- train 및 test 데이터가 .csv파일로 존재(전처리 코드에서 생성됨)

구분	파일명	용도	파일 형식
코드 파일	bigcontest_eda	데이터 전처리	ipynb
	bigcontest_modeling	모델링	ipynb
	bigcontest_stats	통계 분석	ipynb
데이터	train	훈련 데이터	csv
	test	테스트 데이터	csv

데이터 전처리

1. 분석 프레임워크

- 제안된 분석 방법론은 3가지 기본 제공 데이터 테이블을 모두 사용
- user_id를 기준으로 log_data와 user_spec 테이블을 병합하고(user_log 테이블 생성),
- application_id를 기준으로 User_log와 loan_result 테이블 병합(final_df 테이블 생성)



2. 유저로그 데이터

- user_id별로 타 컬럼을 집계한 데이터를 생성
  → 유저 정보가 있는 user_spec 테이블과 병합하기 때문

1) time_gap

- user_id 별로 평균 접속 시간차 계산
  (가정1 : 자주 접속한 사람들은 대출상품을 신청할 가능성이 높을 것이다!)

- **수식 : (가장 최근에 접속한 시간 – 최초 접속 시간) / 접속한 횟수**
- 계산된 시간을 유닉스 시간단위로 변경 후 밀리초 단위로 표시

유저별 평균접속 시간차

user_id	
1	7.668748e+08
7	0.000000e+00
9	1.180000e+05
11	1.636624e+08
12	7.816832e+07

2. 유저로그 데이터

2) num_events

- user_id 별로 수행한 행동 종류의 개수(고유값)

(가정2 : 수행한 행동의 종류가 다양할수록 대출상품을 신청할 가능성이 높을 것이다!)

- ex) user_1 이 회원가입과 핀다 앱을 실행 → num_events = 2

3) event 원-핫 인코딩

- user_id 별로 각 행동을 몇 번 수행했는지에 대해 더미 형태로 집계

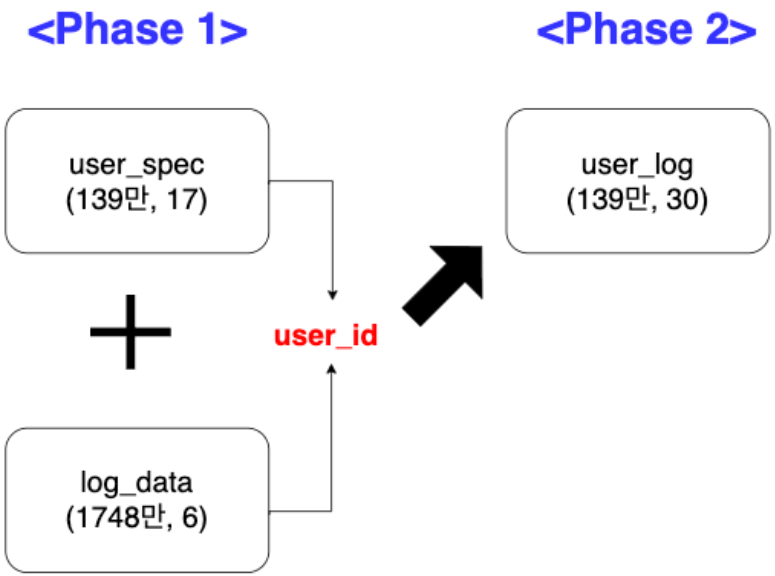
event user_id	CompleteIDCertification	EndLoanApply	GetCreditInfo	Login	OpenApp	SignUp	StartLoanApply	UseDSRCalc	UseLoanManage	UsePrepayCalc	ViewLoanApplyIntro
1	0.0	0.0	3.0	1.0	0.0	0.0	0.0	0.0	1.0	0.0	0.0
7	0.0	0.0	1.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
9	0.0	0.0	3.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
11	4.0	7.0	8.0	0.0	5.0	0.0	4.0	0.0	9.0	1.0	5.0
12	15.0	39.0	5.0	15.0	15.0	0.0	15.0	0.0	1.0	0.0	15.0
...	...	...	...	...	...	...	...	...	...	...	...
879693	6.0	12.0	2.0	0.0	9.0	0.0	17.0	0.0	5.0	0.0	10.0
879694	1.0	3.0	0.0	0.0	0.0	0.0	1.0	0.0	0.0	0.0	1.0
879695	1.0	1.0	0.0	0.0	0.0	0.0	1.0	0.0	0.0	0.0	1.0
879696	1.0	1.0	3.0	2.0	1.0	0.0	0.0	0.0	2.0	0.0	1.0
879698	0.0	0.0	0.0	0.0	1.0	0.0	1.0	0.0	0.0	0.0	1.0

584636 rows × 11 columns

2. 유저로그 데이터

4) user_spec 테이블과 병합

- 생성한 13개의 파생 변수 데이터를 user_id 기준으로 user_spec 테이블과 병합
- 병합된 테이블의 이름은 user_log로 명명



3. 유저스펙 데이터(user_log)

1) 파생변수 목록

- 생성한 6개의 파생 변수 정리

변수명	도출과정	도출 이유
rehabilitation_total	personal_rehabilitation_yn + personal_rehabilitation_complete_yn	개인 회생자 여부 및 납입완료 여부는 추후 대출 신청에 영향을 줄 것이라 판단
income_employment_type	Income_type, employment_type 의 형태로 데이터 변경	변수 통합
age	2022 – birthyear + 1	이후 age_group등의 추가 파생변수 생성
age_group	19세이하 – Teenager / 20~26세 – Student / 27~36세 – Young Adult / 37~60세 – Adult / 61~100 - Elder	이후 해당 변수로 결측치 대체
year_income_age	yearly_income / age	연소득에 비해 나이대가 다르면 대출 신청 가능성에 차이가 있을 것이라 판단
existing_avg_loan_amt	existing_loan_amt / existing_loan_cnt	평균 기대출금액으로 유저별 평균 기대 출금액 계산

3. 유저스펙 데이터(user_log)

2) 결측치 처리

구분	변수명	대체값
범주형	purpose	age_group 세대별 purpose의 최대 빈도값 **ex) Teenager 그룹의 purpose 최대 빈도값 → 생활비로 결측치 대체**
	gender / company_enter_month / employment_type / income_type / houseown_type /	Unknown
연속형	credit_score / yearly_income / desired_amount / existing_loan_cnt / existing_loan_amt existing_avg_loan_amt	age_group 세대별 평균값 **ex) Adult 그룹의yearly_income 평균값으로 yearly_income 결측치 대체**
기타 파생변수	num_events, time_gap, event 원-핫 인코딩(11개)	-1로 결측치 대체

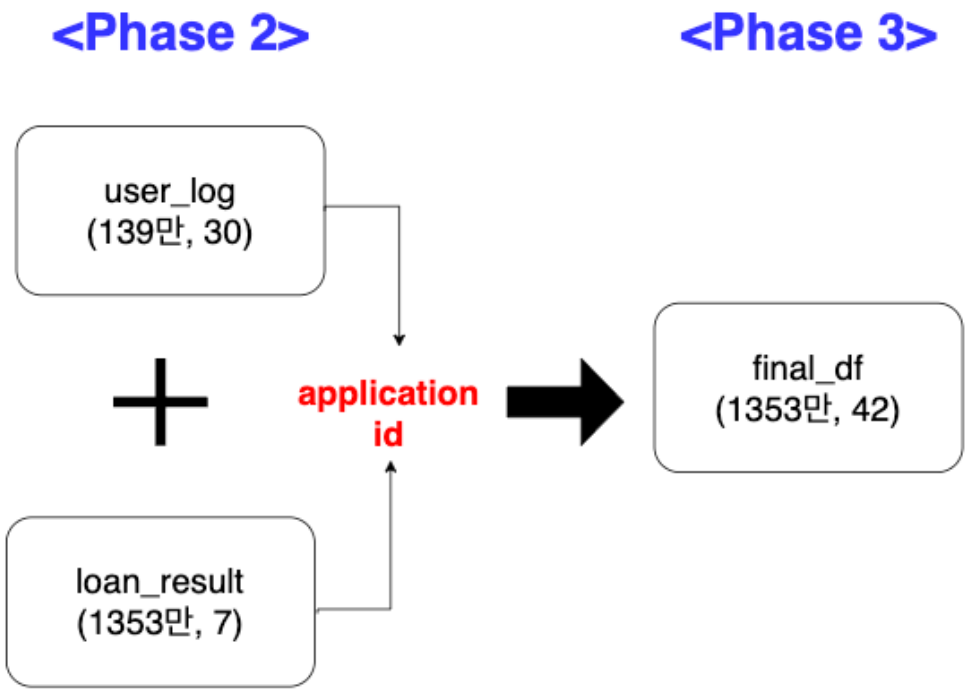
3) 컬럼 제거

- 파생변수 생성에 쓰인 컬럼들을 제거
- birthyear, age, income_type, employment_type

3. 유저스펙 데이터(user_log)

4) loan_result 테이블과 병합

- 전처리를 마친 user_log 테이블을 application_id를 기준으로 loan_result 테이블과 병합
- 병합된 테이블의 이름은 final_df로 명명



4. 최종 데이터(final_df)

1) Null값 가진 Row 제거

- 병합 후, 결측치 조사 → 병합 전 user_log의 변수들에서 동일하게 113개 존재하는 것 확인
- 해당 데이터는 제거

2) 승인한도, 승인금리 결측치 처리

- 승인한도(loan_limit), 승인금리(loan_rate) 변수의 결측치 → 7382개 존재
- 승인한도 및 승인금리 관련 데이터가 없다는 뜻이므로 0으로 결측치 처리

3) LabelEncoding(범주형 변수 전처리)

- string 데이터 타입의 범주형 변수에 적용

(gender, company_enter_month, houseown_type, purpose, income_employment_type, age_group)

4. 최종 데이터(final_df)

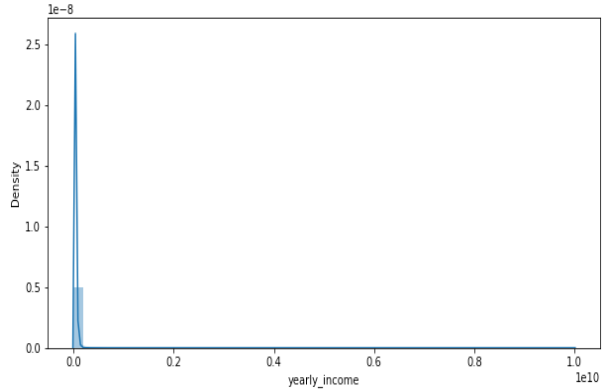
4) 연속형 변수 전처리

- 연속형 변수 중 값 분포가 치우친 **credit_score, yearly_income, desired_amount, existing_loan_amt, existing_avg_loan_amt, year_income_age, loan_limit**에 대해 전처리 수행

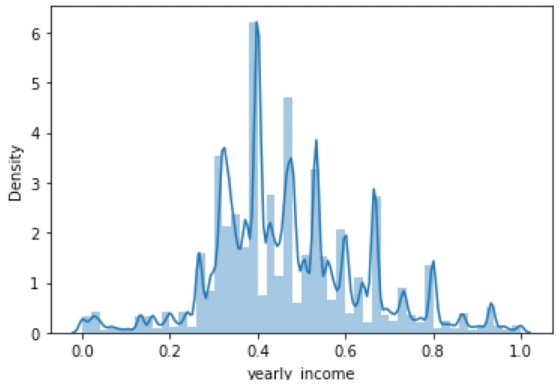
- IQR 범위 이외의 이상치를 제거하고, StandardScaler, MinMaxScaler 의 적용을 통해 연속형 변수의 표준화&정규화 작업 수행

***이상치 제거 전 train/test 데이터를 분할했으며, 이상치 제거는 train데이터에 대해서만 수행**

***train/test 데이터에는 Scaler를 따로 적용**



<전처리 전 yearly_income 분포>



<전처리 후 yearly_income 분포>

4. 최종 데이터(final_df)

5) 시간 타입 변수 전처리

변수명	시간	근거
insert_time	연/월/일 3가지의 컬럼 추가	계정을 생성한 연/월/일 이 대출 신청여부와 관계가 있을 것이라 가정
loanapply_insert_time	일/시간/분 3가지의 컬럼 추가	한도조회한 날짜 / 시간 / 분 이 대출 신청여부와 관계가 있을 것이라 가정

- **String 형태의 데이터 타입을 datetime64 타입으로 변한 후 전처리함**

6) 최종 출력물

- train.csv (730만, 42)
- test.csv (326만, 42)

모델링

1. 알고리즘 설명

1) LightGBM

- 해당 실험에서는 LightGBM분류기(LGBMClassifier) 사용
- Boosting 계열 알고리즘으로, Gradient Boosting 프레임워크로 학습하는 Tree 기반 모델
- Level-Wise 기반의 타 Tree 모델들과는 달리, Leaf-Wise를 통해 수평적으로 Tree의 Leaf 확장

→ **학습이 빠르며, 더 많은 손실을 줄일 수 있음**

- 해당 실험에 사용할 train 데이터셋은 약 730만개의 대용량 데이터이므로,
  빠르게 손실을 줄여나갈 수 있는 LightGBM 모델 선정

2) 통계 분석

- OLS 기반 회귀모델로 종속변수에 대한 독립변수들의 상관관계 도출
- statsmodels.api 패키지 사용
- 추후 최종 예측에 사용할 변수 선택에 활용 예정

2. 변수 및 파라미터 선정

1) 변수 선정

① 통계 분석 기반 변수 제거

- 각각의 독립변수에 대해 t-test 값(p-value) > 0.01이면 유의미하지 않은 변수로 판단하여 제거
- 이후, 남은 독립변수들에 대해 모델의 설명력(R-squared)을 감소시키는 변수 제거

	coef	std err	t	P>|t|	[0.025	0.975]
gender	0.0020	0.000	13.632	0.000	0.002	0.002
credit_score	-0.0003	6.15e-07	-510.973	0.000	-0.000	-0.000
yearly_income	-4.266e-13	5.32e-13	-0.802	0.423	-1.47e-12	6.16e-13
company_enter_month	-0.0001	6.89e-07	-195.241	0.000	-0.000	-0.000
houseown_type	-0.0015	5.65e-05	-26.566	0.000	-0.002	-0.001
desired_amount	-1.43e-11	3.01e-13	-47.479	0.000	-1.49e-11	-1.37e-11
personal_rehabilitation_yn	0.0367	0.002	18.556	0.000	0.033	0.041
personal_rehabilitation_complete_yn	-0.0339	0.003	-12.068	0.000	-0.039	-0.028
existing_loan_cnt	0.0039	3.09e-05	125.177	0.000	0.004	0.004
existing_loan_amt	-7.19e-11	1.27e-12	-56.783	0.000	-7.44e-11	-6.94e-11
num_event	0.0682	0.000	182.460	0.000	0.067	0.069
time_gap	-2.402e-11	4.73e-13	-50.806	0.000	-2.49e-11	-2.31e-11
CompleteIDCertification	-0.0704	0.000	-187.794	0.000	-0.071	-0.070
EndLoanApply	-0.0668	0.000	-178.824	0.000	-0.068	-0.066
GetCreditInfo	-0.0683	0.000	-182.689	0.000	-0.069	-0.068
Login	-0.0685	0.000	-183.031	0.000	-0.069	-0.068
OpenApp	-0.0682	0.000	-182.369	0.000	-0.069	-0.067
SignUp	-0.0601	0.000	-144.322	0.000	-0.061	-0.059
StartLoanApply	-0.0689	0.000	-183.999	0.000	-0.070	-0.068
UseDSRCalc	-0.0698	0.000	-153.015	0.000	-0.071	-0.069
UseLoanManage	-0.0682	0.000	-182.366	0.000	-0.069	-0.068
UsePrepayCalc	-0.0684	0.000	-165.997	0.000	-0.069	-0.068
ViewLoanApplyIntro	-0.0679	0.000	-181.186	0.000	-0.069	-0.067
rehabailitation_total	0.0028	0.001	2.452	0.014	0.001	0.005
age_group	0.0013	3.89e-05	34.237	0.000	0.001	0.001
existing_avg_loan_amt	8.951e-11	3.28e-12	27.263	0.000	8.31e-11	9.59e-11
loan_limit	7.435e-11	3.21e-12	23.189	0.000	6.81e-11	8.06e-11
loan_rate	-0.0058	1.82e-05	-320.025	0.000	-0.006	-0.006
insert_year	0.0002	3.99e-07	526.650	0.000	0.000	0.000
insert_month	0.0016	8.54e-05	18.815	0.000	0.001	0.002
insert_day	-0.0497	0.001	-71.734	0.000	-0.051	-0.048
loan_day	0.0499	0.001	71.963	0.000	0.049	0.051
loan_hour	-0.0011	1.31e-05	-87.591	0.000	-0.001	-0.001
loan_minute	-2.426e-05	4.03e-06	-6.022	0.000	-3.22e-05	-1.64e-05

2. 변수 및 파라미터 선정

1) 변수 선정

① 통계 분석 기반 변수 제거

- 통계 분석 기반으로 제거해야 하는 변수 목록

구분	변수명
T-test 결과 P-value > 0.01	**yearly_income(연소득), rehabilitation_total**
제거시 모델 설명력 증가	**gender(성별), personal_rehabilitation_complete_yn(개인회생자여부), houseown_type(주거소유형태), existing_avg_loan_amt(평균기대출금액), year_income_age, loan_limit(대출한도), insert_year, loan_minute**

2. 변수 및 파라미터 선정

2) 변수 선정

② 머신 러닝 모델 기반 변수 제거

- 다양한 독립변수들을 조합하여 모델의 성능 개선 유무 확인
- 머신 러닝 기반으로 제거해야 하는 변수 목록

변수명
gender(성별), personal_rehabilitation_yn(개인회생자 여부), personal_rehabilitation_complete_yn(납입완료 여부), SignUp(유저별 회원가입 횟수)

2. 변수 및 파라미터 선정

3) 파라미터 선정

- LightGBM 분류기 모델에 사용할 파라미터 선정

파라미터명	기능	파라미터 설정값
n_estimators	학습할 Boosting Tree 개수(=학습 횟수)	1000
max_depth	Tree 깊이	6
boosting_type	Boosting 타입	gbdt
boost_from_average	정답 레이블이 불균형한 경우 False를 설정해 모델에게 알림	False
random_state	Seed 설정	42
n_jobs	모델 학습에 사용할 CPU 코어 수	-1(가용 가능한 전체)

3. 평가 지표

- 모델의 성능을 측정하고 변수 및 파라미터를 선정하기 위해 평가 지표 사용
- Accuracy, Precision, Recall, F1-Score, Roc_Auc의 5가지 지표 사용

→ 단순 Accuracy 만을 사용하여 모델이 편향되는 것을 방지하기 위함

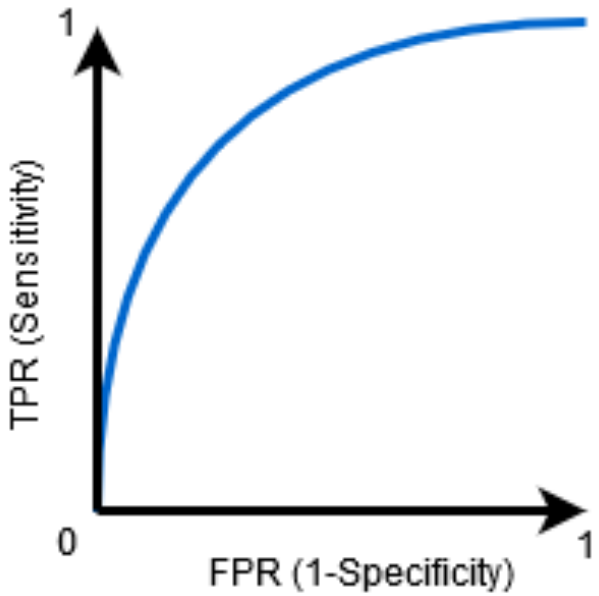
$$Accuracy = \frac{T_p + T_n}{T_p + T_n + F_p + F_n}$$

$$Precision = \frac{T_p}{T_p + F_p}$$

$$Recall = \frac{T_p}{T_p + T_n}$$

$$F_1 = 2 \cdot \frac{precision \cdot recall}{precision + recall}$$

<Accuracy/Precision/Recall/F1-Score>



<Roc-Auc>

최종 예측 결과

1. 모델 예측 결과

1) 통계 분석 및 머신 러닝 기반 변수 사용 결과

- 실험 결과 머신 러닝 기반 변수를 사용하여 테스트 데이터를 예측하기로 결정

* OLS 회귀모델 예측 정확도가 많은 독립변수 사용시 크게 떨어지는 것으로 해석됨

* OLS 회귀모델은 선형성을 가정하므로 독립변수 개수가 많아질수록 회귀계수 추정량 분산 증가

→ 선형 예측 모형의 복잡도가 크게 증가해 예측 정확도가 감소함

평가지표	통계 분석 변수 사용 결과	머신 러닝 변수 사용 결과
Accuracy	0.9409	0.9415
Precision	0.5763	0.5911
Recall	0.0935	0.1125
F1-score	0.1609	0.1890
Roc-Auc	0.8766	0.8852

1. 모델 예측 결과

2) 머신 러닝 기반 test 데이터 예측

- 선정된 변수 및 파라미터를 사용하여 LightGBM 분류기로 test 데이터를 예측
- 예측 방식 : StratifiedKFold 기법 사용
- 데이터를 5분할로 나누고, 각 분할된 데이터를 학습한 후 테스트 데이터 예측
- 이후, 5개의 테스트 데이터 예측값의 평균 계산
  → 평균값이 0.5 이상이면 1로, 그렇지 않으면 0으로 분류

2. 추후 발전방향

1) 통계 분석

- 실험에 사용한 OLS 회귀모델은 독립변수가 많을수록 모델의 설명력 및 예측력이 떨어짐
- 모델 예측 향상에 도움을 주기 위해서는 변수들을 더 잘 통제할 수 있는 모델 사용 필요
- Ridge 분류기 기법을 사용하여 설명력 및 예측력 향상 가능

→ Ridge는 linear 모델에 L2 규제를 부여하여 상대적으로 큰 coef값을 갖는 변수 영향력을 감소시킴

2) 추가 머신 러닝 모델과의 비교

- 실험 단계에서 컴퓨팅 자원상의 문제로 LightGBM 분류기만을 사용하였음

→ 본 모델은 평가 지표 중 Recall 및 F1-Score에서 약점을 보임

- Imbalance한 데이터를 보다 잘 분류 하는 다양한 머신러닝 분류기 기법을 사용하여 모델들 간 분류 성능 비교 가능
- 높은 성능을 보이는 모델들 간의 앙상블 또한 실행 가능

2. 추후 발전방향

3) 변수 재선정

- 도메인 지식을 활용하여 모델의 예측 성능을 향상시킬 수 있는 파생 변수 생성 필요
- 실험 단계에서는 LightGBM에 수동으로 변수를 투입/방출하는 형식으로 최적의 변수 조합을 선정했으나, 보다 다양하고 정밀한 기법을 사용할 필요가 있음

→ Python 패키지에서 사용 가능하다면 StepWise 등의 기법으로 최적 변수 조합을 찾아야 함

Thank You!