보험 사기자 예측 프로젝트

인천대학교
김영민

2019.7 作

Contents

하나, 분석 절차

- 분석 방법 (KDD 분석 절차)

KDD 분석 절차

둘, 데이터 파악(Selection)

- 데이터 설명

- 보험 사기자를 예측하기 위해 제공되는 파일은 보험가입 고객의 정보(CUST_DATA)와 고객들이 보험 청구 기록(CLAIM_DATA)에 관한 데이터

- 고객의 정보 데이터에서 개인정보는 비식별조치 되었으며 총 25개 변수로 구성

- 보험 청구 데이터의 개인정보는 비식별조치 되었으며 38개 변수로 구성

- 제공한 고객 데이터의 DIVIDED_SET 변수는 학습용 데이터셋은 1, 평가용 데이터셋은 2로 값을 구성

- 보험 사기자 여부를 저장하고 있는 변수는 SIU_CUST_YN 이며 일반 고객이면 'N', 보험 사기자이면 'Y' 의 값을 가지고 평가용 데이터셋에는 값이 할당되어 있지 않음

둘, 데이터 파악(Selection)

- 데이터 설명

고객 정보 데이터(CUST_DATA)

No	변수영문명	변수타입	변수명	변수 설명
1	CUST_ID	N	고객ID	고객을 구분하는 고유번호
2	DIVIDED_SET	N	데이터셋 구분	학습용 Set의 경우 1번 // 평가용 Set의 경우 2번을 부여
3	SIU_CUST_YN	C	보험사기자여부	Y의 경우 '보험사기자' / N의 경우 '일반고객' /평가용 Set에는 미부여
4	SEX	N	성별	성별 1은 '남성' 2는 '여성'
5	AGE	N	연령	고객연령
6	RESI_COST	N	주택가격	고객의 거주지 주택가격 추정값 (단위 : 만원) (0 : 추정불가)
7	RESI_TYPE_CODE	C	거주TYPE	고객의 거주지 형태 - 일반단독주택(11), 다가구단독주택(12), 영업겸용단독주택(13), 아파트(20),연립_다가구주택(30) 상가등 비거주용건물(40),오피스텔(50),숙박업소의 객실 또는 판자집 등(60),기숙사(70),그외(99)
8	FP_CAREER	C	FP경력	(당사 FP로써의) Y : 경력있음 // N : 경력 없음
9	CUST_RGST	C	고객등록년월	최초 당사의 고객으로써의 등록연월
10	CTPR	C	시도구분	고객의 거주 시/도
11	OCCP_GRP1	C	직업그룹코드1	총 8개직업군으로 분류한 코드
12	OCCP_GRP2	C	직업그룹코드2	총 25개직업군으로 분류한 코드
13	TOTALPREM	N	납입총보험료	고객이 지금까지 당사에 실제 납입한 총 보험료의 합계
14	MINCRDT	C	신용등급(최소)	신용등급 확인 중 최소값 (미확인의 경우 6등급에 포함)
15	MAXCRDT	C	신용등급(최대)	신용등급 확인 중 최대값 (미확인의 경우 6등급에 포함)
16	WEDD_YN	C	결혼여부	Y : 결혼함 / N : 결혼안함 (계약 당시에는 결혼하지 않았던 상태 포함)
17	MATE_OCCP_GRP1	C	배우자직업그룹코드1	총 8개직업군으로 분류한 코드
18	MATE_OCCP_GRP2	C	배우자직업그룹코드2	총 25개직업군으로 분류한 코드
19	CHLD_CNT	N	자녀수	고객의 자녀 수
20	LTBN_CHLD_AGE	N	막내자녀연령	고객의 막내자녀 연령
21	MAX_PAYM_YM	C	최대보험료연월	당사에 최대규모의 보험료를 납입했던 연월
22	MAX_PRM	N	최대보험료	당사에 최대규모의 보험료를 납입했던 월보험료 수준
23	CUST_INCM	N	고객추정소득	고객의 연령/직업/보험료 수준등을 통한 고객의 개인 소득 추정금액
24	RCBASE_HSHD_INCM	N	추정가구소득1	고객의 주택가격을 우선하여 정한 가구소득 추정금액
25	JPBASE_HSHD_INCM	N	추정가구소득2	고객의 직업 및 납입보험료 수준을 우선하여 정한 가구소득 추정금액

둘, 데이터 파악(Selection)

- 데이터 설명

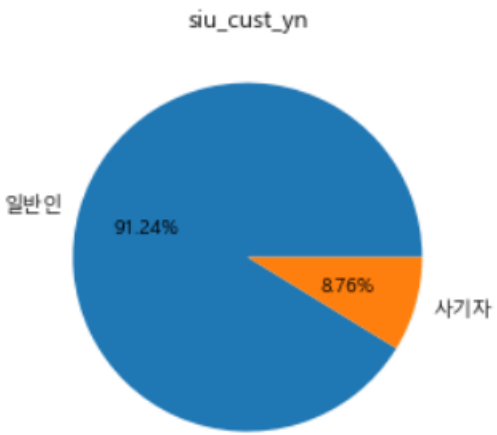
고객 청구 데이터(CLAIM_DATA)

No	변수영문명	변수타입	변수명	변수 설명
1	CUST_ID	N	고객ID	고객을 구분하는 고유번호
2	POLY_NO	N	증권번호	청약서번호이면서 동시에 계약성립후에는 증권번호로 사용
3	ACCI_OCCP_GRP1	C	직업그룹코드1	총 8개직업군으로 분류한 코드(사고 당시)
4	ACCI_OCCP_GRP2	C	직업그룹코드2	총 25개직업군으로 분류한 코드(사고 당시)
5	CHANG_FP_YN	C	FP 변경 여부	모집 FP와 청구 당시 수금 FP 와의 동일 여부
6	CNTT_RECP_SQNO	C	계약별접수일련번호	사고접수에 대해 해당 계약건별로 부여하는 번호
7	RECP_DATE	C	사고접수일자	사고가 접수된 일자
8	ORIG_RESN_DATE	C	원사유일자	사고접수시 해당 사고의 최초 사유발생일자
9	RESN_DATE	C	사유일자	보험금 지급사유 발생일자
10	CRNT_PROG_DVSN	C	현재진행구분	현재진행구분 상태 구분 - 접수(11), 심사배정(21), 심사(22), 심사결재(23), 조사(32), 조사결재(33)
11	ACCI_DVSN	C	사고구분	사고원인을 구분함 - 재해(1), 교통재해(2), 질병(3)
12	CAUS_CODE	C	원인코드	사고의 원인에 해당하는 사인코드
13	CAUS_CODE_DTAL	C	원인코드상세	사고의 원인에 해당하는 사인코드_상세정보
14	DSAS_NAME	C	병명	병명
15	DMND_RESN_CODE	C	청구사유코드	지급청구의 원인이 되는 사유코드 - 사망(01), 입원(02), 통원(03), 장해(04), 수술(05), 진단(06), 치료(07), 해지/무효(09)
16	DMND_RSCD_SQNO	N	청구사유코드일련번호	동일 증번, 동일한 청구사유이지만 사유일자가 다른 경우 일련번호를 1씩 증가시킴.
17	HOSP_OTPA_STDT	C	입원/통원시작일자	입원시작일, 통원은 통원시작일 (입원은 무조건 연속된일자만 관리됨)
18	HOSP_OTPA_ENDT	C	입원/통원종료일자	입원종료일, 통원은 통원종료일
19	RESL_CD1	C	결과코드1	사고원인에 대한 결과코드
20	RESL_NM1	C	결과명1(사인내용)	결과내용
21	VLID_HOSP_OTDA	N	유효입원/통원일수	보험금지급대상인 입원일수 또는 통원일수
22	HOUSE_HOSP_DIST	N	고객병원거리	고객 거주지와 병원까지의 거리(km)
23	HOSP_CODE	N	병원코드	병원코드
24	ACCI_HOSP_ADDR	C	병원지역(시도)	병원지역
25	HOSP_SPEC_DVSN	C	병원종별구분	병원종별구분 - 종합병원(10), 병원(20), 요양병원(25), 의원(30), 치과병원(40), 치과의원(45), 보건의료원(60), 약국(70), 한방병원(80), 한의원(85), 해외(90), 의료기관이외(95)
26	CHME_LICE_NO	N	담당의사면허번호	의사면허번호
27	PAYM_DATE	C	지급일자	보험금 지급일자
28	DMND_AMT	N	청구금액	사고보험금청구금액
29	PAYM_AMT	N	지급금액(지급테이블)	실지급금액
30	PMMI_DLNG_YN	C	실손처리여부	실손처리여부
31	SELF_CHAM	N	본인부담금	국민건강보험 적용 금액 중 환자 부담 금액
32	NON_PAY	N	비급여	국민건강보험 미적용 금액
33	TAMT_SFCA	N	전액본인부담금	국민건강보험 미적용 금액
34	PATT_CHRG_TOTA	N	환자부담총액	본인부담금 + 비급여 + 전액본인부담금
35	DSCT_AMT	N	영수할인금액	병원에서 할인해주는 비용
36	COUNT_TRMT_ITEM	N	진료과목개수	실손영수증 내 진료과목의 개수
37	NON_PAY_RATIO	N	실손비급여비율	(비급여 + 전액본인부담금) / (환자부담총액) = (비급여 + 전액본인부담금) / (본인부담금 + 비급여 + 전액본인부담금)
38	HEED_HOSP_YN	C	유의병원여부	금감원 유의 병원 대상 여부

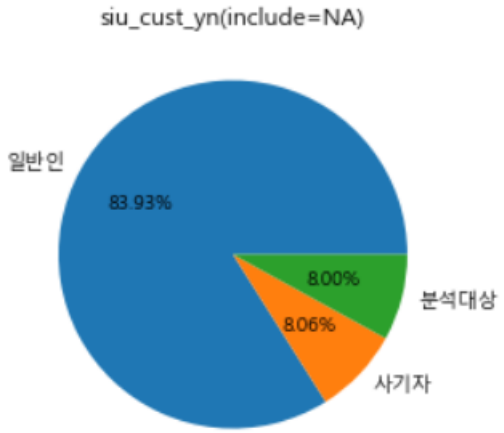
둘, 데이터 파악

- 종속변수 확인

사기자 분포



결측치를 제외한
사기자와 일반인 분포



분석 대상을 포함한
사기자와 일반인
분석대상 분포

셋, 데이터 전처리(Preprocessing)

- 데이터 결측치 확인

```
cust_id              0
divided_set          0
siu_cust_yn       1793
sex                  0
age                  0
resi_cost            0
resi_type_code    1254
fp_career            0
cust_rgst          456
ctpr               621
occp_grp_1         595
occp_grp_2         595
totalprem         5791
mincrdt           9476
maxcrdt           9476
wedd_yn            473
mate_occp_grp_1   11827
mate_occp_grp_2   11827
chld_cnt            473
ltbn_chld_age       473
max_paym_ym        6486
max_prm            6486
cust_incm          5263
rcbase_hshd_incm      0
jpbase_hshd_incm    680
dtype: int64
```

- 데이터 결측치 비율

```
cust_id             0.000000
divided_set         0.000000
siu_cust_yn         8.004464
sex                 0.000000
age                 0.000000
resi_cost           0.000000
resi_type_code      5.598214
fp_career           0.000000
cust_rgst           2.035714
ctpr                2.772321
occp_grp_1          2.656250
occp_grp_2          2.656250
totalprem          25.852679
mincrdt            42.303571
maxcrdt            42.303571
wedd_yn             2.111607
mate_occp_grp_1    52.799107
mate_occp_grp_2    52.799107
chld_cnt            2.111607
ltbn_chld_age       2.111607
max_paym_ym        28.955357
max_prm            28.955357
cust_incm          23.495536
rcbase_hshd_incm    0.000000
jpbase_hshd_incm    3.035714
dtype: float64
```

- 데이터 타입 정보

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 22400 entries, 0 to 22399
Data columns (total 25 columns):
cust_id              22400 non-null int64
divided_set          22400 non-null int64
siu_cust_yn          20607 non-null object
sex                  22400 non-null int64
age                  22400 non-null int64
resi_cost            22400 non-null int64
resi_type_code       21146 non-null float64
fp_career            22400 non-null object
cust_rgst            21944 non-null float64
ctpr                 21779 non-null object
occp_grp_1           21805 non-null object
occp_grp_2           21805 non-null object
totalprem            16609 non-null float64
mincrdt              12924 non-null float64
maxcrdt              12924 non-null float64
wedd_yn              21927 non-null object
mate_occp_grp_1      10573 non-null object
mate_occp_grp_2      10573 non-null object
chld_cnt             21927 non-null float64
ltbn_chld_age        21927 non-null float64
max_paym_ym          15914 non-null float64
max_prm              15914 non-null float64
cust_incm            17137 non-null float64
rcbase_hshd_incm     22400 non-null int64
jpbase_hshd_incm     21720 non-null float64
dtypes: float64(11), int64(6), object(8)
memory usage: 4.3+ MB
```

셋, 데이터 전처리 (Preprocessing)

- 결측치 처리(특정값으로 대체)

- 고객의 거주 타입인 resi_type_code 결측치 처리

⇒ 그 외 코드인 99로 처리하려 하였으나 거주 타입의 코드가 99인 고객들 중에 resi_cost가 0(추정불가)인 고객들이 없으므로 최빈값으로 대체(20 : 아파트)

- 납입 총 보험료인 totalprem이 결측치 처리

=〉 결측값인 고객들 중에 물론 수입이 없는 사람들도 있지만 많은 사람들이 수입이 있기 때문에 중위값으로 대체

- 최대 신용등급과 최소 신용등급인 Maxcrdt, mincrdt 결측치 처리

=〉 Maxcrdt이 결측이면 mincrdt도 결측이므로 신용등급 파악 불가. 따라서 maxcrdt은 최대값으로 mincrdt은 최소값으로 대체

- 시도구분 코드인 ctpr의 결측치 처리

=〉 거주지의 최빈값으로 대체(경기)

셋, 데이터 전처리 (Preprocessing)

- 결측치 처리(다른 변수를 고려하여 대체)

- 고객의 수입 cust_incm의 결측치 처리

⇒ Cust_incm에서 직업과 소득이 모두 결측인 경우 occp_grp_1 은 무직으로 처리, cust_incm은 0으로 처리 –〉 그러나 주부인 사람들의 cust_incm이 모두 0이므로 변수의 확장을 방지하기 위해 '1.주부'로 대체

⇒ 직업 코드가 있고 cust_incm이 결측인 데이터는 직업별 평균 소득으로 대체

- 결혼 여부인 wedd_yn 데이터의 결측치 처리

⇒ 결혼 여부가 N인 데이터의 CHLD_CNT , LTBN_CHLD_AGE 은 모두 0으로 처리

⇒ mate_occp_grp_1 , CHLD_CNT , LTBN_CHLD_AGE, 가 모두 결측일때 wedd_yn 도 결측이므로 N으로 대체

셋, 데이터 전처리 (Preprocessing)

- 결측치 처리(랜덤포레스트를 이용한 결측치 대체)

- **직업 코드인 occp_grp_1 인 결측치 처리**

⇒ **fp_career, totalprem,cust_incm 변수를 입력변수로 활용**

⇒ **randomforest 모델을 활용해 직업 코드를 예측하여 결측치 대체**

```
from sklearn.ensemble import RandomForestClassifier

rf = RandomForestClassifier()
```

```
rf.fit(train_x, train_y)
pred_y = rf.predict(test_x)
```

```
pred_y
```

```
array(['4.전문직', '5.서비스', '3.사무직', '3.사무직', '3.사무직', '5.서비스', '3.사무직',
       '3.사무직', '6.제조업', '3.사무직', '5.서비스', '6.제조업', '4.전문직', '5.서비스',
       '2.자영업', '6.제조업', '5.서비스', '5.서비스', '2.자영업', '6.제조업', '3.사무직',
       '4.전문직', '5.서비스', '5.서비스', '2.자영업', '3.사무직', '2.자영업', '3.사무직',
       '5.서비스', '2.자영업', '5.서비스', '4.전문직', '3.사무직', '5.서비스', '5.서비스',
       '5.서비스', '3.사무직', '4.전문직', '5.서비스', '3.사무직', '3.사무직', '3.사무직',
       '3.사무직', '3.사무직', '4.전문직', '4.전문직', '2.자영업', '3.사무직', '5.서비스',
       '5.서비스', '3.사무직', '4.전문직', '6.제조업', '3.사무직', '5.서비스', '4.전문직',
       '3.사무직', '2.자영업', '5.서비스', '5.서비스', '2.자영업', '5.서비스', '6.제조업',
       '3.사무직', '3.사무직', '3.사무직', '3.사무직', '5.서비스', '5.서비스', '3.사무직',
       '6.제조업', '8.기타', '3.사무직', '3.사무직', '6.제조업', '3.사무직', '3.사무직',
       '4.전문직', '3.사무직', '5.서비스', '3.사무직', '3.사무직', '3.사무직', '3.사무직',
       '2.자영업', '5.서비스', '3.사무직', '2.자영업', '3.사무직', '3.사무직', '3.사무직',
       '3.사무직', '3.사무직', '3.사무직', '3.사무직', '5.서비스', '5.서비스', '2.자영업',
       '4.전문직', '3.사무직', '5.서비스', '5.서비스', '3.사무직', '4.전문직', '2.자영업',
       '5.서비스', '3.사무직', '6.제조업', '4.전문직', '6.제조업', '3.사무직', '5.서비스',
       '3.사무직', '8.기타', '4.전문직', '5.서비스', '7.1차산업', '5.서비스', '2.자영업',
       '3.사무직', '3.사무직', '2.자영업', '8.기타'], dtype=object)
```

셋, 데이터 전처리 (Preprocessing)

- 유도변수 생성

- **고객의 나이 변수 age를 19세 이하, 39세 이하, 59세 이하, 나머지 노년으로 구분하여 age_range 변수 추가**

```
count    22400.000000
mean        44.734866
std         15.445707
min          2.000000
25%         34.000000
50%         46.000000
75%         56.000000
max         89.000000
Name: age, dtype: float64
```

- Age 칼럼에 대한 기술 통계

최소 나이 : 2세
최대 나이 : 89세

	age	age_range
0	47	중년
1	53	중년
2	60	장년
3	64	장년
4	54	중년

Age와 age_range 칼럼 비교

셋, 데이터 전처리 (Preprocessing)

- 유도변수 생성

- **고객의 자녀의 수 chld_cnt를 0 : no , 1~3 : small , 4~ : big 으로 구분지어 chld_range 변수 추가**

```
count    22400.000000
mean         0.694509
std          0.949046
min          0.000000
25%          0.000000
50%          0.000000
75%          1.000000
max          6.000000
Name: chld_cnt, dtype: float64
```

- chld_cnt 칼럼에 대한 기술 통계

최소 자녀 수 : 0명
최대 자녀 수 : 6명

	chld_cnt	chld_range
0	2.0	small
1	2.0	small
2	0.0	no
3	0.0	no
4	3.0	small

Chld_cnt와 chld_range 칼럼 비교

셋, 데이터 전처리 (Preprocessing)

- 파생 변수 생성

- CLAIM_DATA의 변수를 이용해 파생변수 생성
- 사기자 예측에 영향을 주는 요인
- 평균 입원 일수 : VLID_HOSP_OTDA(통원일/입원일)
- 사고구분 및 청구사유 횟수 : ACCI_DVSN, DMND_RESN_CODE

셋, 데이터 전처리 (Preprocessing)

– 파생 변수 생성

- **고객의 청구 데이터인 claim_data의 vlid_hosp_otda 칼럼을 이용해 평균을 구하고 mean_days 칼럼을 cust_data에 추가**

	cust_id	vlid_hosp_otda
0	5936	2
1	5936	2
2	5936	2
3	1043	6
4	8545	0
5	4734	4
6	9416	0
7	20267	23
8	2778	29
9	9019	13

- vlid_hosp_otda 컬럼과 cust_id 컬럼

Cust_id가 중복되어서 들어가있다.

	cust_id	mean_days
0	1	1.250000
1	2	2.666667
2	3	16.000000
3	4	0.000000
4	5	25.000000

평균 입원 일자를
Mean_days칼럼으로 추가

셋, 데이터 전처리 (Preprocessing)

- 파생 변수 생성

- **사고구분 및 청구사유 코드 : ACCI_DVSN, DMND_RESN_CODE 를 이용하여 사고 원인과 청구 코드 결합 ex) 1_1, 1_2**

고객별 사고구분과 청구 사유 코드 →

	cust_id	acci_dvsn	dmnd_resn_code
0	5936	1	3
1	5936	1	3
2	5936	1	3
3	1043	3	2
4	8545	3	5

	1_1	1_2	1_3	1_4	1_5	1_6	1_7	1_9	2_1	2_2	2_3	2_4	2_5	2_6	2_7	2_9	3_1	3_2	3_3	3_4	3_5	3_6	3_7	3_9
1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	4	0	0	0	0	0
2	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	3	0	0	0	0	0
3	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0
4	0	0	0	0	0	9	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
5	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0

사고 원인과 청구 코드별 청구 횟수

셋, 데이터 전처리 (Preprocessing)

- 컬럼 삭제

Cust_data의 정보

- **차원이 증가하여 overfitting(과적합)이 발생하는 것을 방지하기 위한 변수제거**

```
Index(['cust_id', 'divided_set', 'siu_cust_yn', 'sex', 'age', 'resi_cost',
       'resi_type_code', 'fp_career', 'cust_rgst', 'ctpr', 'occp_grp_1',
       'occp_grp_2', 'totalprem', 'mincrdt', 'maxcrdt', 'wedd_yn',
       'mate_occp_grp_1', 'mate_occp_grp_2', 'chld_cnt', 'ltbn_chld_age',
       'max_paym_ym', 'max_prm', 'cust_incm', 'rcbase_hshd_incm',
       'jpbase_hshd_incm', 'age_range', 'chld_range', 'mean_days', '1_1',
       '1_2', '1_3', '1_4', '1_5', '1_6', '1_7', '1_9', '2_1', '2_2', '2_3',
       '2_4', '2_5', '2_6', '2_7', '2_9', '3_1', '3_2', '3_3', '3_4', '3_5',
       '3_6', '3_7', '3_9'],
      dtype='object')
```

Cust_data의 컬럼

Cust_data의 결측치

```
cust_id              0
divided_set          0
siu_cust_yn       1793
sex                  0
age                  0
resi_cost            0
resi_type_code       0
fp_career            0
cust_rgst          456
ctpr                 0
occp_grp_1           0
occp_grp_2         595
totalprem            0
mincrdt              0
maxcrdt              0
wedd_yn              0
mate_occp_grp_1  11827
mate_occp_grp_2  11827
chld_cnt             0
ltbn_chld_age        0
max_paym_ym       6486
max_prm           6486
cust_incm            0
rcbase_hshd_incm     0
jpbase_hshd_incm   680
age_range            0
chld_range           0
mean_days            0
1_1                  0
1_2                  0
```

```
1  cust_data1.shape
```

```
(22400, 52)
```

Cust_data의 행과 열 개수

변수의 개수가 너무 많다.

셋, 데이터 전처리 (Preprocessing)

- 컬럼 삭제

- occp_grp_2 , mate_occp_grp2 => 세부 분류 한 것이므로 대체 값 존재하므로 삭제
- Age와 chld_cnt => age_range 와 chld_range 컬럼이 있으므로 삭제
- 분산이 0인 칼럼 삭제: 모두 같은 값이라는 의미 => 2_7 , 2_9 칼럼 삭제
- max_prm 유의미하지 않다고 판단
- max_paym_ym은 연도를 가르키기 때문에 삭제 필요
- 고객 등록연월(Cust_rgst)도 같은 이유로 삭제
- 추정가구 소득 jpbase_hshd_incm은 rcbase와 기준만 다른 것이므로 삭제
- mate_occp_grp_1 유의미하지 않은 데이터 (결측치가 많다.)
- ctpr 지역과 상관이 없다고 판단

삭제 후 칼럼 →

```
Index(['cust_id', 'divided_set', 'siu_cust_yn', 'sex', 'resi_cost',
       'resi_type_code', 'fp_career', 'occp_grp_1', 'totalprem', 'mincrdt',
       'maxcrdt', 'wedd_yn', 'ltbn_chld_age', 'cust_incm', 'rcbase_hshd_incm',
       'age_range', 'chld_range', 'mean_days', '1_1', '1_2', '1_3', '1_4',
       '1_5', '1_6', '1_7', '1_9', '2_1', '2_2', '2_3', '2_4', '2_5', '2_6',
       '3_1', '3_2', '3_3', '3_4', '3_5', '3_6', '3_7', '3_9'],
      dtype='object')
```

셋, 데이터 전처리 (Preprocessing)

- 모델링을 위한 Encoding

- Object type 변수를 OneHotEncoding을 통해 int type으로 변경
- 단점 : 칼럼의 수 추가
- sex, age_range ,fp_career ,wedd_yn ,chld_range ,occp_grp_1 의 값들을 OneHotEncoding 실행

s1	a0	a1	a2	a3	f0	f1	w0	w1	c0	c1	c2	o0	o1	o2	o3	o4	o5	o6	o7
1.0	0.0	0.0	1.0	0.0	1.0	0.0	0.0	1.0	0.0	0.0	1.0	0.0	0.0	1.0	0.0	0.0	0.0	0.0	0.0
0.0	0.0	0.0	1.0	0.0	1.0	0.0	0.0	1.0	0.0	0.0	1.0	0.0	0.0	1.0	0.0	0.0	0.0	0.0	0.0
0.0	0.0	1.0	0.0	0.0	1.0	0.0	1.0	0.0	0.0	1.0	0.0	0.0	0.0	0.0	0.0	1.0	0.0	0.0	0.0
1.0	0.0	1.0	0.0	0.0	0.0	1.0	1.0	0.0	0.0	1.0	0.0	0.0	1.0	0.0	0.0	0.0	0.0	0.0	0.0
1.0	0.0	0.0	1.0	0.0	0.0	1.0	0.0	1.0	0.0	0.0	1.0	0.0	1.0	0.0	0.0	0.0	0.0	0.0	0.0

넷, 모델링(Transformation)

- 학습데이터와 테스트 데이터 분리

- **학습 데이터 : divided set == 1**
- **테스트 데이터 : divided set == 2**

- **종속 변수 : siu_cust_yn**
- **독립 변수 : cust_id, divided set , siu_cust_yn 을 제외한 모든 변수**

```
train=tmp[tmp['divided_set']==1]
test=tmp[tmp['divided_set']==2]
```

```
train_x=train.loc[:,train.columns.difference(['cust_id','siu_cust_yn','divided_set'])]
train_y=train['siu_cust_yn']
```

```
test_x=test.loc[:,test.columns.difference(['cust_id','siu_cust_yn','divided_set'])]
test_y=test['siu_cust_yn']
```

넷, 모델링(Data Mining)

- 모델 선택

분류 기법
모델 3가지

넷, 모델링 (Data Mining)

- 모델링

- RandomForest , Extratree , XGBoost 모델링(제약조건 없음)
- GridSearch를 이용해 모델링
- 제약조건

```
rf.parms={'max_depth':[None,5,10],'criterion':['gini','entropy'],'max_features':[None,'sqrt','log2']}
```

```
et.parms={'max_depth':[None,5,10],'criterion':['gini','entropy'],'max_features':[None,'sqrt','log2']}
```

```
xgb.parms={'max_depth':[3,5,10],'gamma':[0,0.5,1],'max_features':['auto','log','log2'],'eta':[0.05,0.1,0.3]}
```

- GridSearch 를 이용한 best_parm

```
1 rf_cv.best_params_
```

```
{'criterion': 'entropy', 'max_depth': 10, 'max_features': None}
```

```
1 et_cv.best_params_
```

```
{'criterion': 'entropy', 'max_depth': 10, 'max_features': None}
```

```
1 xgb_cv.best_params_
```

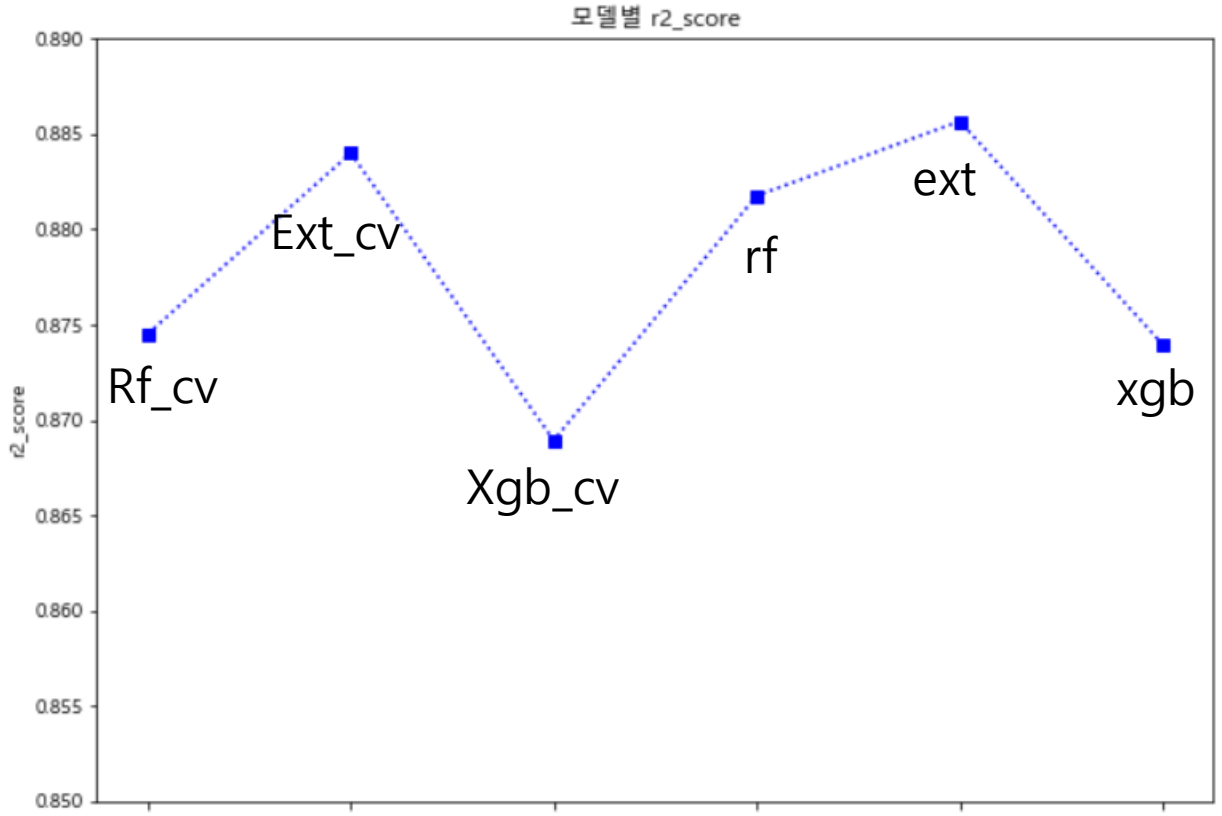
```
{'eta': 0.05, 'gamma': 1, 'max_depth': 5, 'max_features': 'auto'}
```

넷, 모델링 (Data Mining)

- 모델별 r2 socre 비교

Answer data와 Comparison

- RandomForest –〉 Extratree –〉 Xgboost –〉 gridsearch(RandomForest) –〉 gridsearch(Extratree) –〉 gridsearch(Xgboost) 순서




```
{'rf_cv': 0.8745119910764082,
 'et_cv': 0.8839933073061907,
 'xgb_cv': 0.8689347462353597,
 'rf': 0.8817624093697713,
 'et': 0.8856664807585053,
 'xgb': 0.8739542665923034}
```

R2 score values

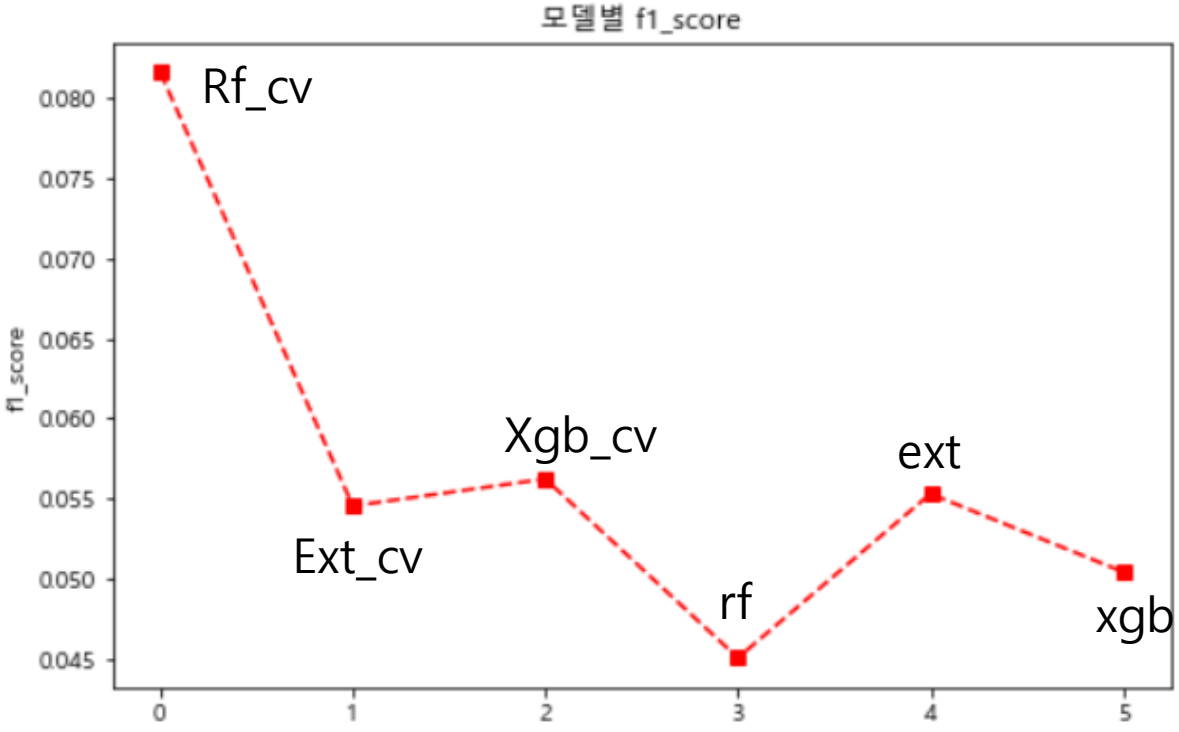
넷, 모델링

- 모델별 f1 socre 비교

Answer data와 Comparison

- gridsearch(RandomForest)–〉gridsearch(Extratree)

–〉gridsearch(Xgboost)–〉RandomForest –〉 Extratree –〉 Xgboost 순서




```
{'pred_rf_cv': 0.0816326530612245,
 'pred_et_cv': 0.05454545454545454,
 'pred_xgb_cv': 0.05622489959839357,
 'pred_rf': 0.04504504504504505,
 'pred_et': 0.05529953917050691,
 'pred_xgb': 0.05042016806722689}
```

F1_score values

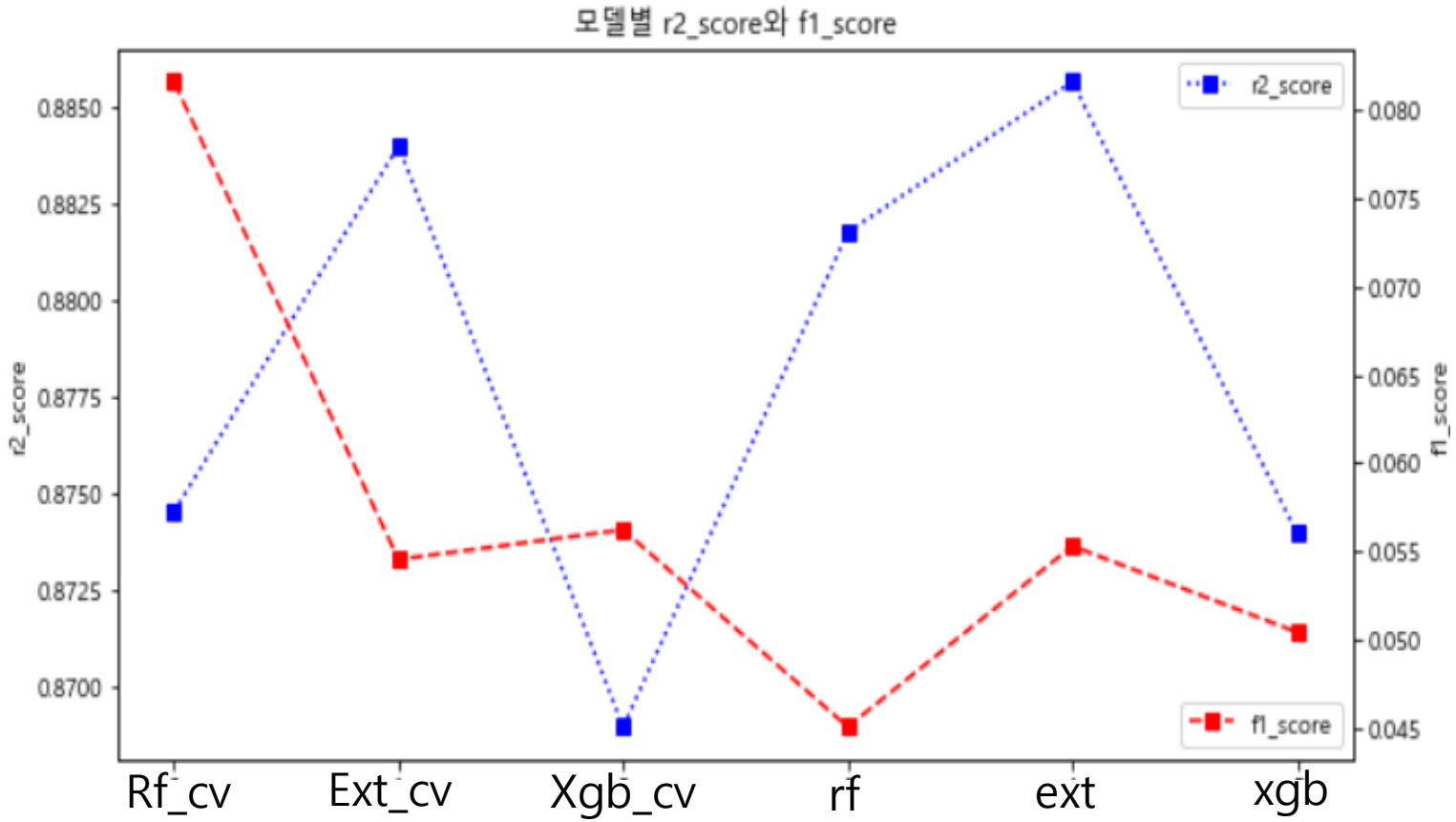
넷, 모델링

- r2 score 와 f1 score

Answer data와 Comparison

- gridsearch(RandomForest)–〉gridsearch(Extratree)

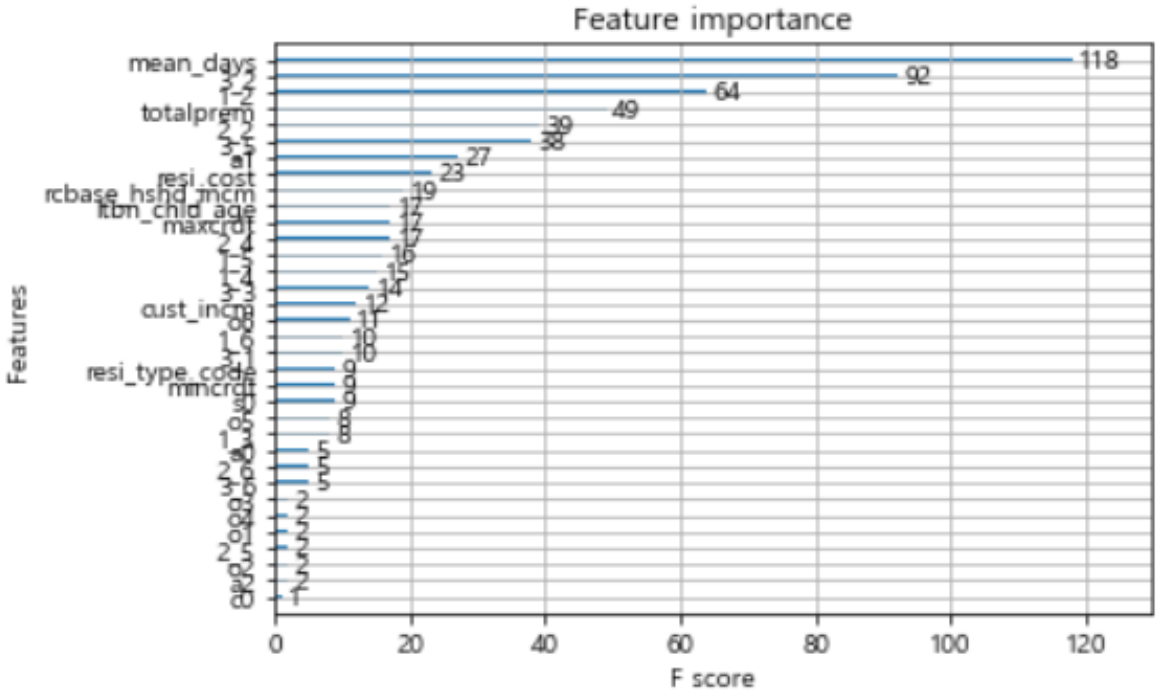
–〉gridsearch(Xgboost)–〉RandomForest –〉 Extratree –〉 Xgboost 순서



다섯, 결과해석(Evaluation)

- Xgboost Classifier

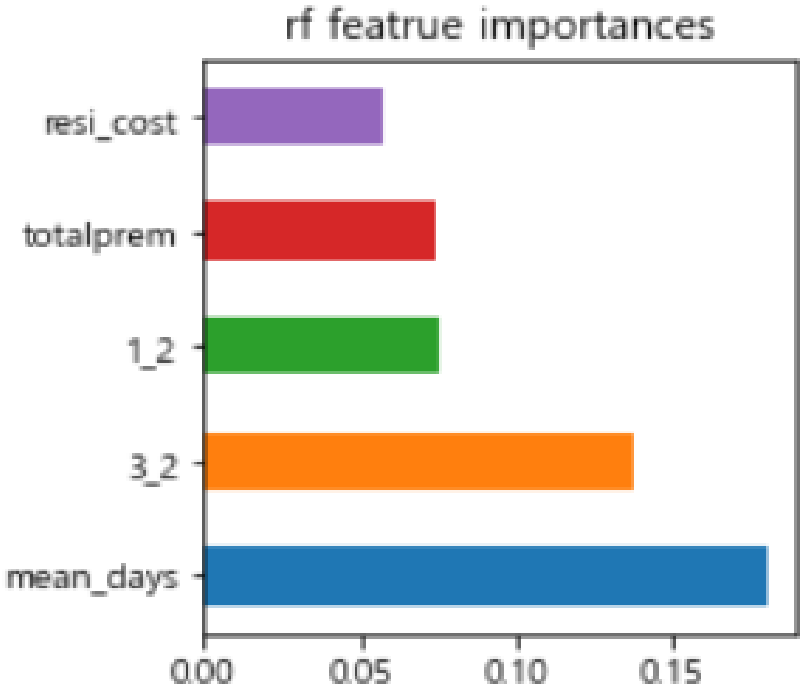
- Xgboost 에서 사기자 여부를 결정한 가장 큰 요인
- Mean_days(평균 입원 일 수)가 가장 컸고, 그 다음으로 3_2(질병으로 인한 입원)이 중요한 변수이다.



다섯, 결과해석(Evaluation)

- Randomforest Classifier

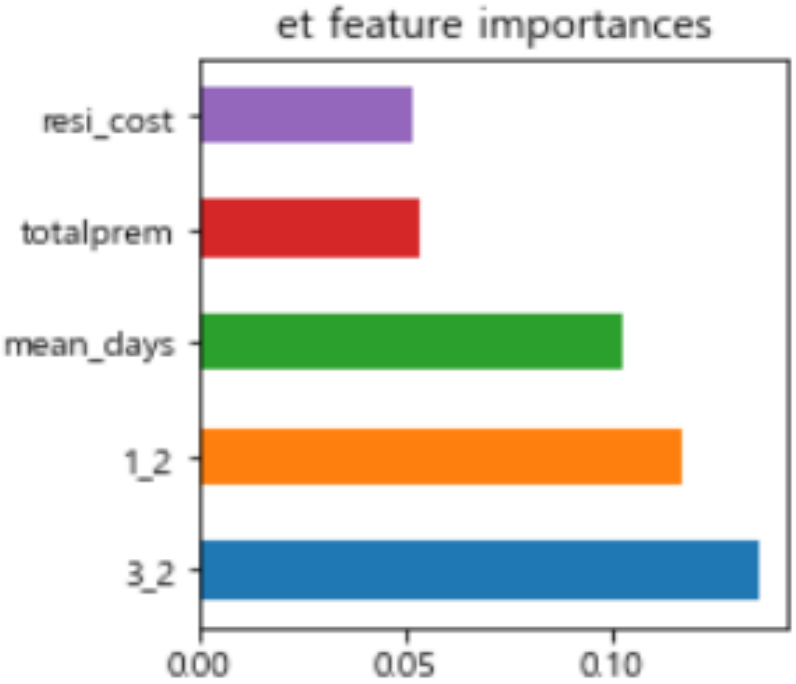
- Randomforest 에서 사기자 여부를 결정한 가장 큰 요인
- Mean_days(평균 입원 일 수)가 가장 컸고, 그 다음으로 3_2(질병으로 인한 입원)이 중요한 변수이다.



다섯, 결과해석(Evaluation)

- Extratree Classifier

- Extratree 에서 사기자 여부를 결정한 가장 큰 요인
- 3_2(질병으로 인한 입원) 가 가장 컸고, 그 다음으로 1_2(재해로 인한 입원)이 중요한 변수이다.



다섯, 결과 해석

- 평균 입원 일 수, 질병으로 인한 입원 요인이 모델에서 사기자를 분류하는데 가장 큰 역할을 하였다.
- 1_2, 3_2, mean_days 칼럼의 공통점은 입원이다.
- 사기자를 예측하는데 가장 크게 고려해야 할 점은 **입원여부, 입원 일자**이다.

여섯, 보완점 파악

– 파생 변수 생성

- 컬럼의 삭제를 좀 더 데이터 분석을 기반으로 삭제해야한다.
- 모델링 과정에 많은 시도가 없었다. (모델이 3개밖에 없었다.)
- GridSearch 에서 더 많은 parameter 조합을 시도했어야 한다.

감사합니다.