금융데이터를 활용한
“나의 금융생활 정보지수”

날 닮은 너

Index

Our Team

날 닮은 너

고객이 입력한

개인정보와 금융 데이터를 통해

나를 닮은 사람을 찾아내고

금융 상품을 추천하여

금융 생활에 도움을 주고자

'날 닮은 너' 프로젝트를 시작하였습니다.

Question No.1

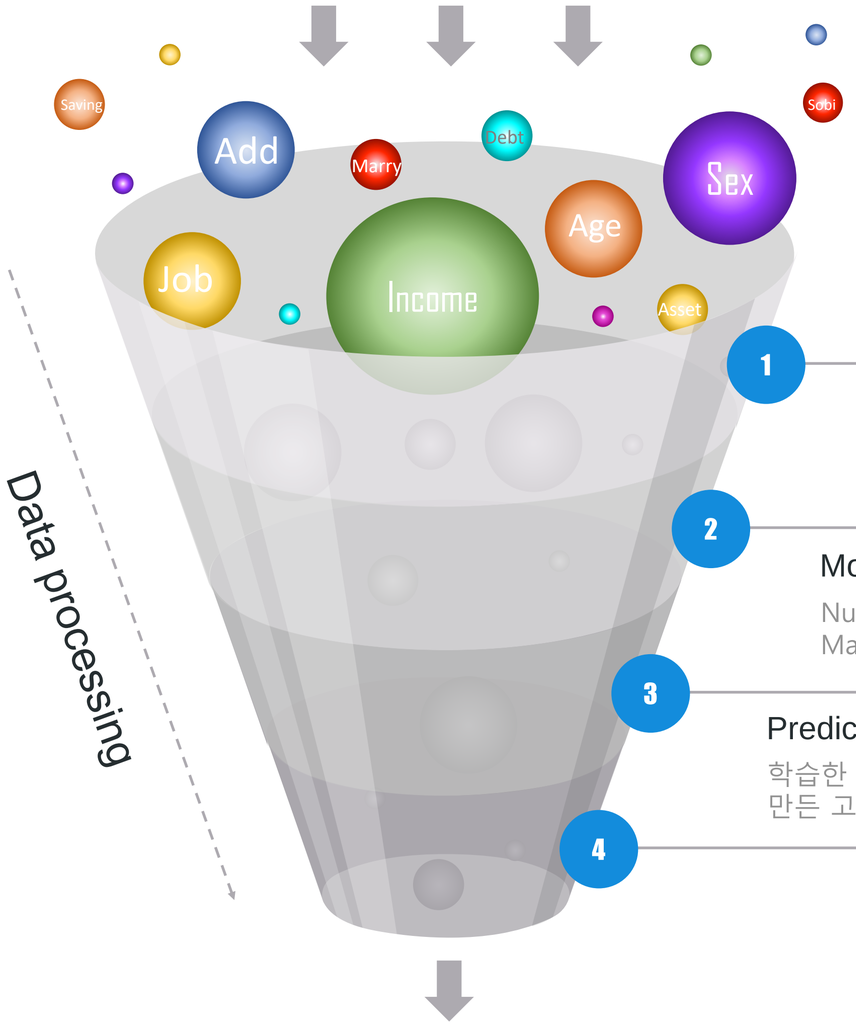
Financial Information Prediction

Overview

Data Modeling

분석 모델을 위한
사용된 주요 라이브러리

- Scikit-learn
- XGBoost
- LightGBM
- Seaborn
- Pandas
- Numpy



1

Data Understanding

각각의 feature의 의미와 결측, 이상 데이터 탐색

2

Data cleaning & Feature engineering

One hot encoding을 통해 명목 형 feature 처리
Ratio 기반의 새로운 field 생성

3

Model Building

Null값 존재 유무에 기반하여 데이터 분할
Machine Learning 기법을 활용하여 데이터를 학습

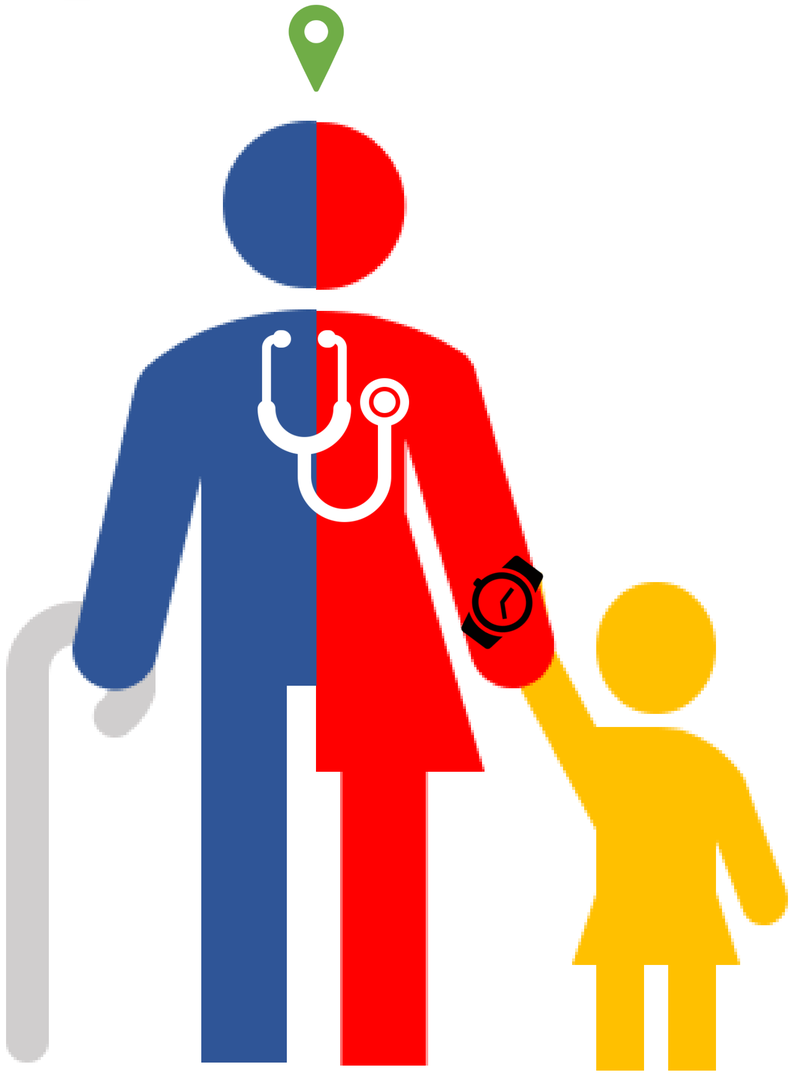
4

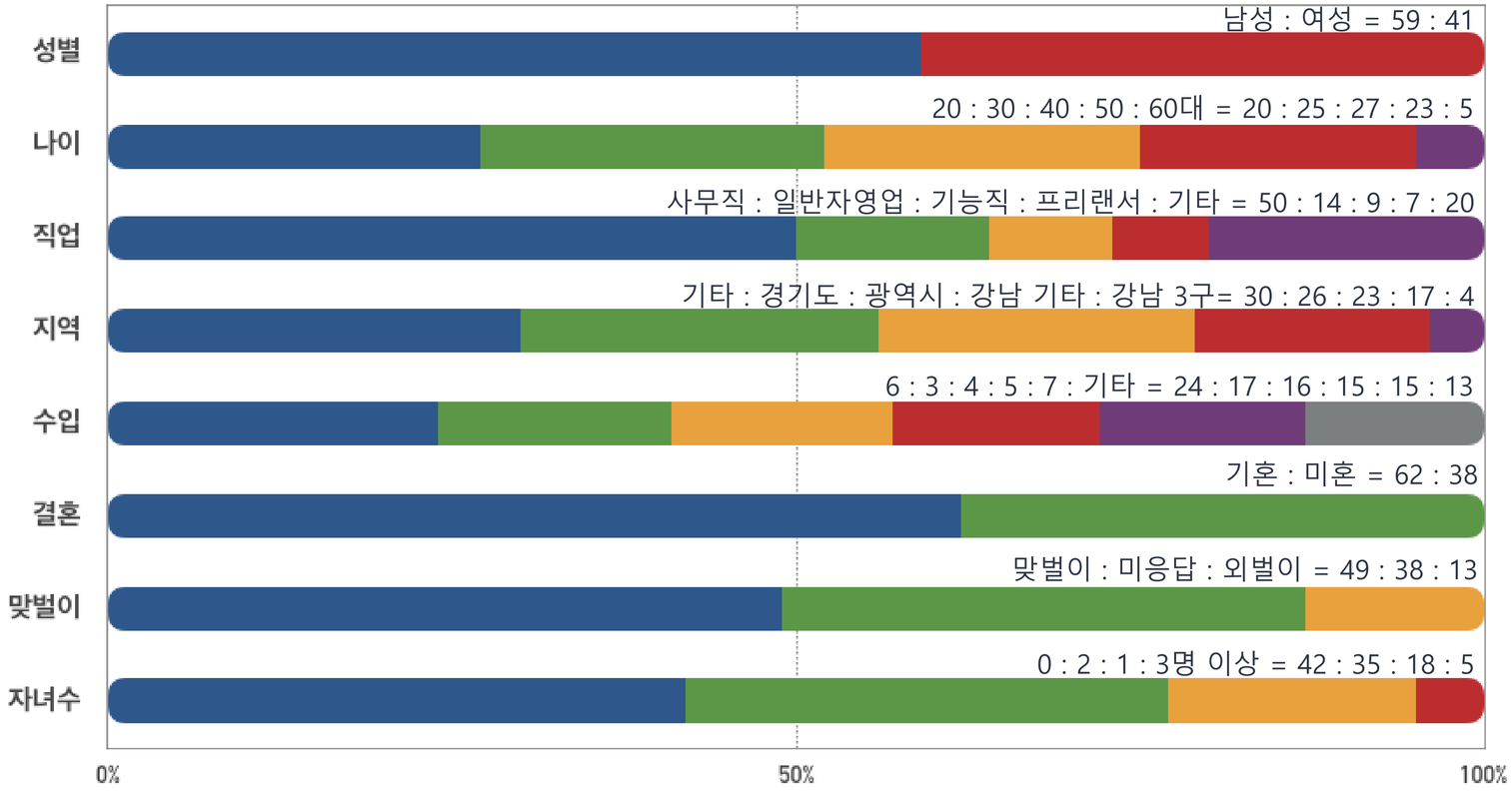
Predictions

학습한 모델을 통하여 8개의 기본정보를 조합하여
만든 고객 유형 마다 금융정보 예측 및 입력

Data Understanding

객체 별 분포도





- 각 feature 별 분포는 대체적으로 균등하게 분포하고 있음
- 하지만, 직업 중 사무직에 과도한 집중 현상이 보여지고 있으며
- 맞벌이 경우 미혼의 경우 미 응답 처리되어 있어 주의가 필요함

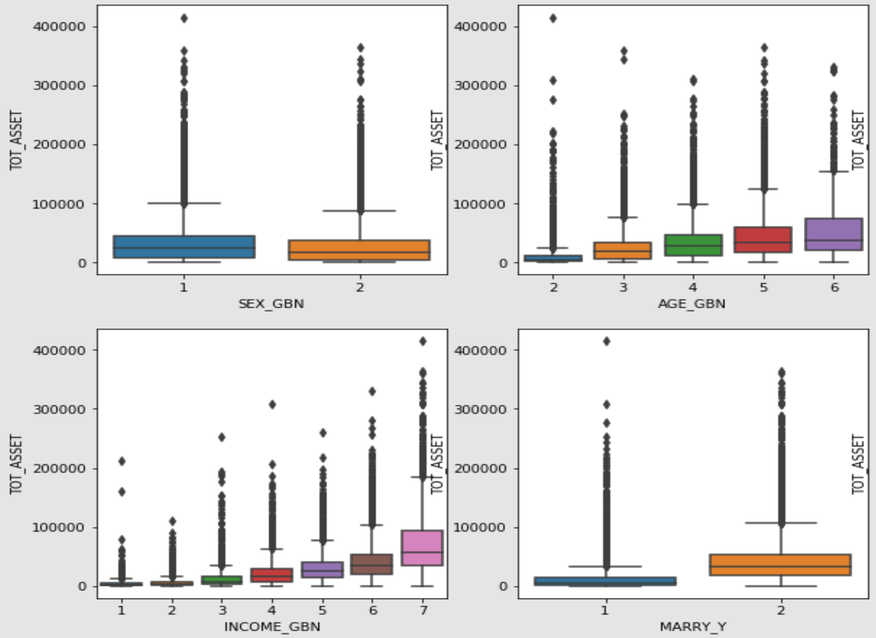
Data Understanding

2가지 속성을 통한 데이터 탐색

- 기혼이면서 자녀 수가 null로 입력되어 있는 고객의 유형은 1212 가지로 이를 실제로 자녀수가 0명인지 또는 미 응답인지 구분해야 하는 고민이 발생
- 하지만, test의 경우 '4'라는 미응답 값이 따로 존재하기 때문에 train의 null 값은 '0'으로 대체
- 또한, 이 분포를 통해서 자녀가 없는 부부의 85%는 맞벌이임을 알 수 있음
- 반면, 706명의 사무직의 경우 filtering 전 Job 필드의 전체 50%를 차지하기 때문에 상대적으로 많은 인원이 몰려있어 정상으로 간주

```
data.iloc[:,:8][(data['NUMCHILD'] ==0) & \
(data['MARRY_Y'] ==2)].apply(lambda x : x.fillna(0).value_counts()).fillna("")
```

	SEX_GBN	AGE_GBN	JOB_GBN	ADD_GBN	INCOME_GBN	MARRY_Y	DOUBLE_IN	NUMCHILD
0.0								1212
1.0	642			48	1		177	
2.0	570	224	706	207	36	1212	1035	
3.0		597	63	279	108			
4.0		256	43	327	200			
5.0		110		351	335			
6.0		25	20		403			
7.0			49		129			
8.0			82					
9.0			134					
10.0			87					
11.0			28					



#성별
남성의 경우 여성보다 상대적으로 많은 자산을 가짐

#연령별
연령대가 증가할수록 전반적으로 많은 자산으로 보유하고 있음

#소득
총소비의 분포와 마찬가지로 소득 구분이 증가할 수록 총자산의 양이 증가함

#결혼여부
본 데이터의 총 자산은 가구당 보유한 총 자산을 의미하기 때문에 기혼의 경우 더 높은 자산을 보유하고 있고 기혼 고객의 95%, 미혼은 49%가 부동산 자산을 보유함

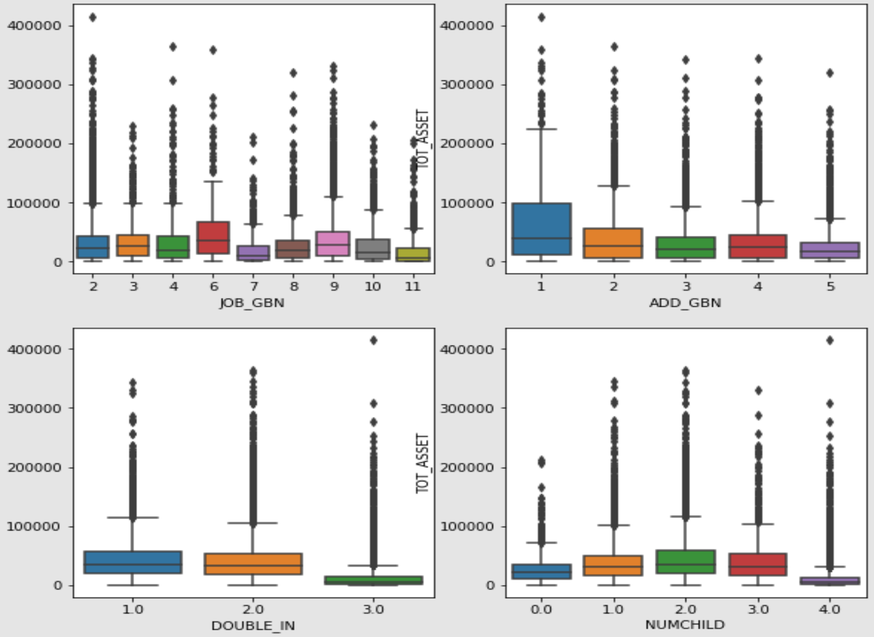
Data Understanding

TOT_ASSET 내 기본 정보 분포

#직업별
직업별 부동산자산의 평균을 확인해보니 전문직(자영업)의 경우 가장 높은 부동산 자산을 보유하여 총자산이 높은 것으로 판단됨

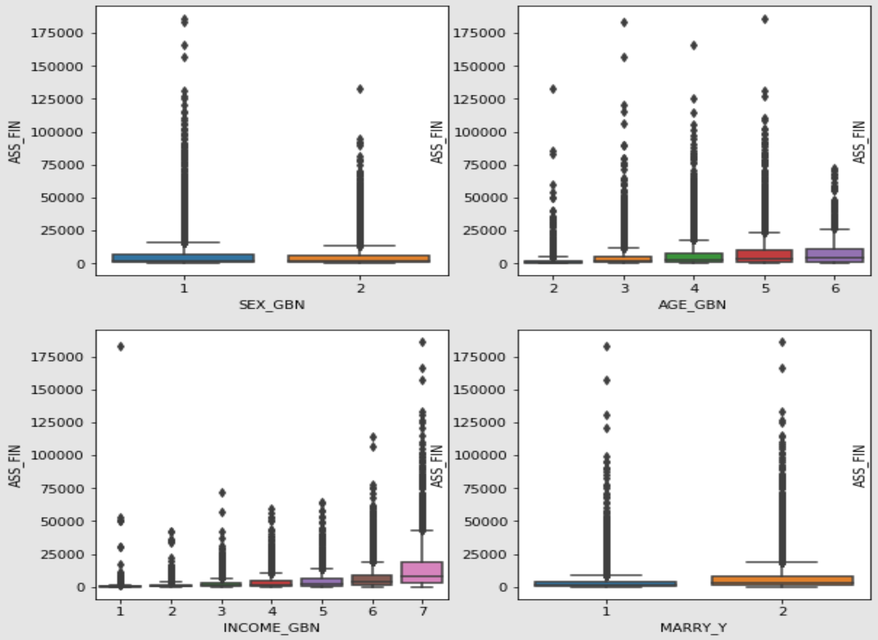
#지역별
강남 3구의 경우 부동산 자산이 타 지역에 비해 크기 때문에 강남 3구에 거주하는 고객의 금융자산이 가장 높다고 판단됨

#자녀 수
자녀 수 별 총소비 분포와 마찬가지로 2명의 자녀를 보유한 가구가 전반적으로 많은 자산, 특히 부동산을 보유하고 있음



Data Understanding

ASS_FIN 내 기본 정보 분포



#성별

성별은 금융자산 보유액에 큰 차이를 보이지 않지만 남성의 경우 더 넓은 스펙트럼을 보임

#연령별

연령대가 증가할수록 금융자산의 보유액이 증가하지만 60대 이상의 경우 금융자산의 스펙트럼이 줄어듦

#소득별

소득이 증가할수록 금융자산의 보유액이 증가함. 다른 개인정보들에 비해 소득에 따라 확연한 차이를 보임

#결혼여부

기혼의 경우 더 많은 금융자산을 보유함. 결혼을 하며 재산이 합쳐지기 때문으로 판단됨

#지역별

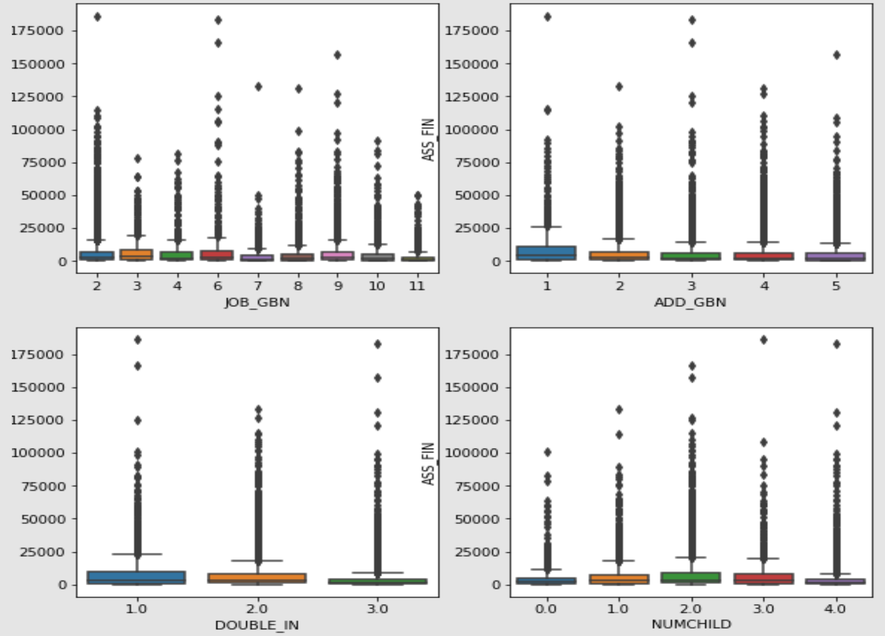
강남 3구에 거주하는 고객의 경우 타 지역에 비해 전반적으로 더 많은 금융자산을 보유함

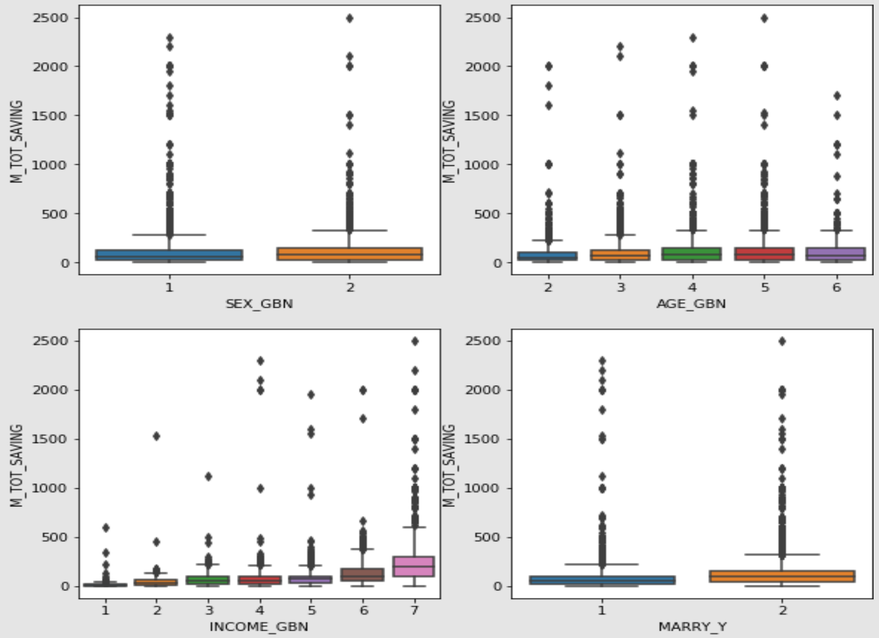
#맞벌이 여부

외벌이의 경우 전반적으로 높은 금융자산을 보유하고 있지만 큰 차이는 보이지 않음

#자녀수

2명의 자녀를 보유한 고객의 경우 전반적으로 많은 금융자산을 보유하고 있지만 큰 차이는 보이지 않음





#연령별
20대부터 40대까지는 저축 액의 양이 증가하는 경향을 보이나 40대부터는 일정하게 유지됨. 연령대가 증가할수록 노년을 대비해 저축 량이 증가하것으로 판단됨

#소득별
소득 구분이 증가할 수록 저축액이 증가함. 다른 개인정보들에 비해 소득 구분에 따라 분포가 확연한 차이를 보이는 것으로 보아 저축액은 소득 구분에 중요한 변수로 판단됨

#결혼여부
기혼 고객의 경우 미혼고객보다 높은 저축액을 보임. 기혼 고객의 경우 가구 합산 월 소득액이 500만원 이상에 해당하는 비율이 56%로 미혼 고객에 비해 45%p 높음

Data Understanding

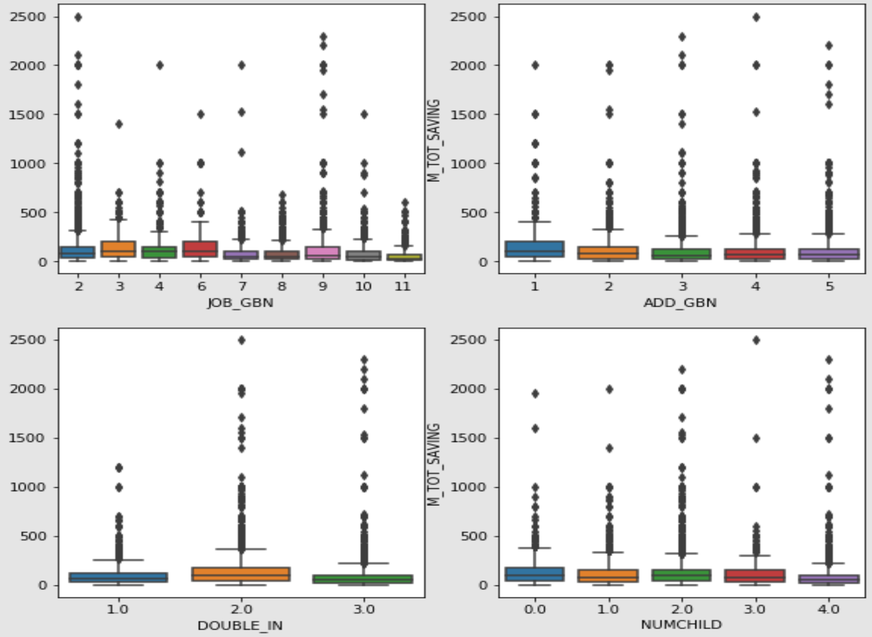
M_TOTAL_SAVING 내
기본 정보 분포

#직업별
공무원의 경우 타 직업 군에 비해 높은 월평균 적금저축액을 보임(공무원 연금 축소 관련 기사 확인). 자영업인 고객의 경우 500만원 이상의 소득을 보유한 비율이 타 직업 군에 비해 높음

#지역별
강남 3구에 거주하는 고객의 저축액이 가장 높은데 이는 강남 3구에 거주하며 가구 합산 월 소득액이 500만원 이상에 해당하는 고객의 비율은 59%로 타 지역에 비해 최소 17%p 이상 높음

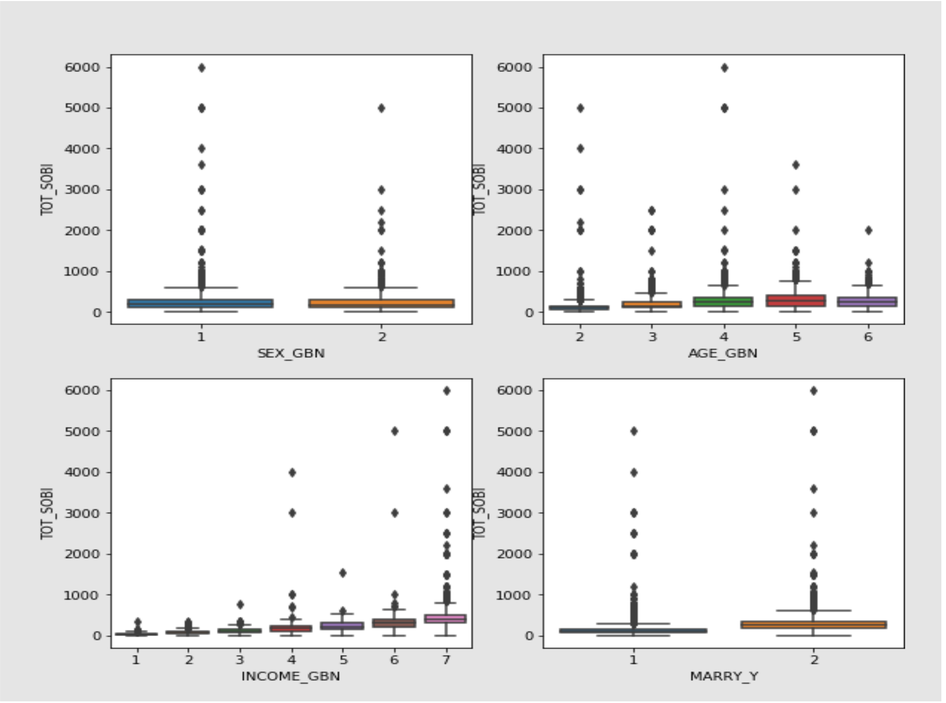
#맞벌이 여부
맞벌이 여부에 따른 가구 합산 월 소득액 비율을 확인해보니 맞벌이고객의 경우 500만원 인 고객이 59%로 외벌이 고객보다 18%p 높음

#자녀 수
자녀수가 줄어들수록 저축액이 감소하는 경향을 보임. 자녀 수에 따른 소비 분포와는 상반 되는 것으로 **보아 소비와 저축이 반비례 관계인 것을 확인 할 수 있음**



Data Understanding

TOT_SOBI 내 기본 정보 분포



#연령별
20대부터 50대까지 연령대가 증가할수록 소비도 증가하는 형태를 보이지만 60대에서 역전됨. 정년 퇴직 또는 고령자의 경우 연금을 제외한 추가적인 소득이 없어 소비를 절제하는 것으로 해석됨

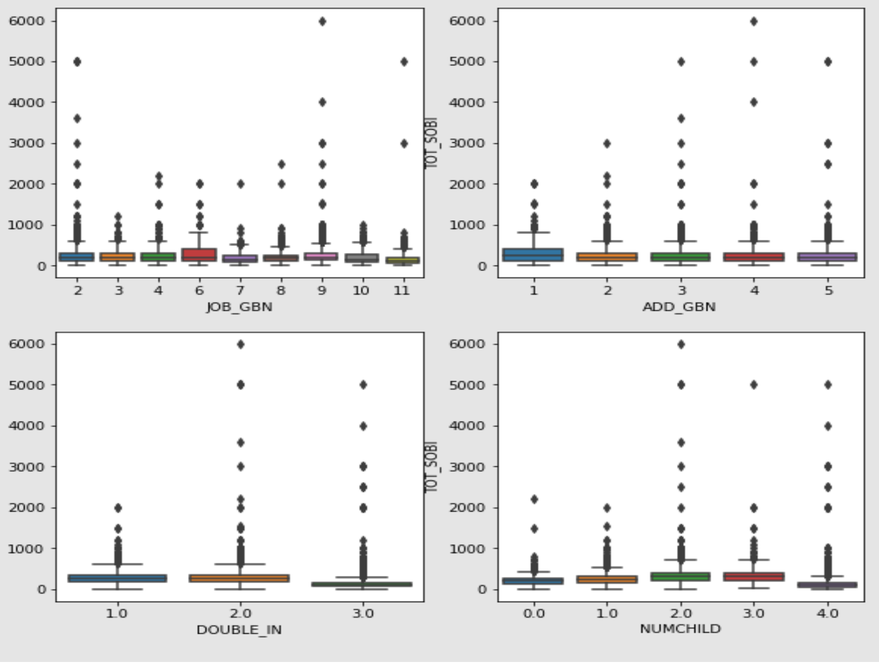
#소득별
소득이 증가할수록 소비가 증가하며, 소득 구간에 대한 구분 중 700만원 이상인 고객들이 한 구간에 포함되어 있기 때문에 많은 이상 치를 보이는 것으로 해석됨. 다른 개인정보들에 비해 소득 구간에 따른 차이가 확연한 것으로 보아 소비에 가장 민감한 변수인 것으로 예상됨

#결혼여부
미혼과 기혼의 평균 자녀 수는 0.15명, 1.58명으로 상대적으로 많은 가구 원을 보유한 기혼이 높은 소비를 보이는 것으로 해석됨

#직업별
자영업의 경우 전반적으로 높은 소비 보이며, 대학원생 및 기타의 경우 가장 낮은 소비를 보임. 이는 전문직(자영업) 종사자의 경우 500만원 이상의 소득을 보유한 고객이 56%로 타 직업에 비해 10%p 이상 높음

#지역별
강남3구의 소비가 타 지역에 비해 전반적으로 높으며, 강남 3구를 제외한 타 지역의 경우 소비에 대해 넓은 스펙트럼을 보임

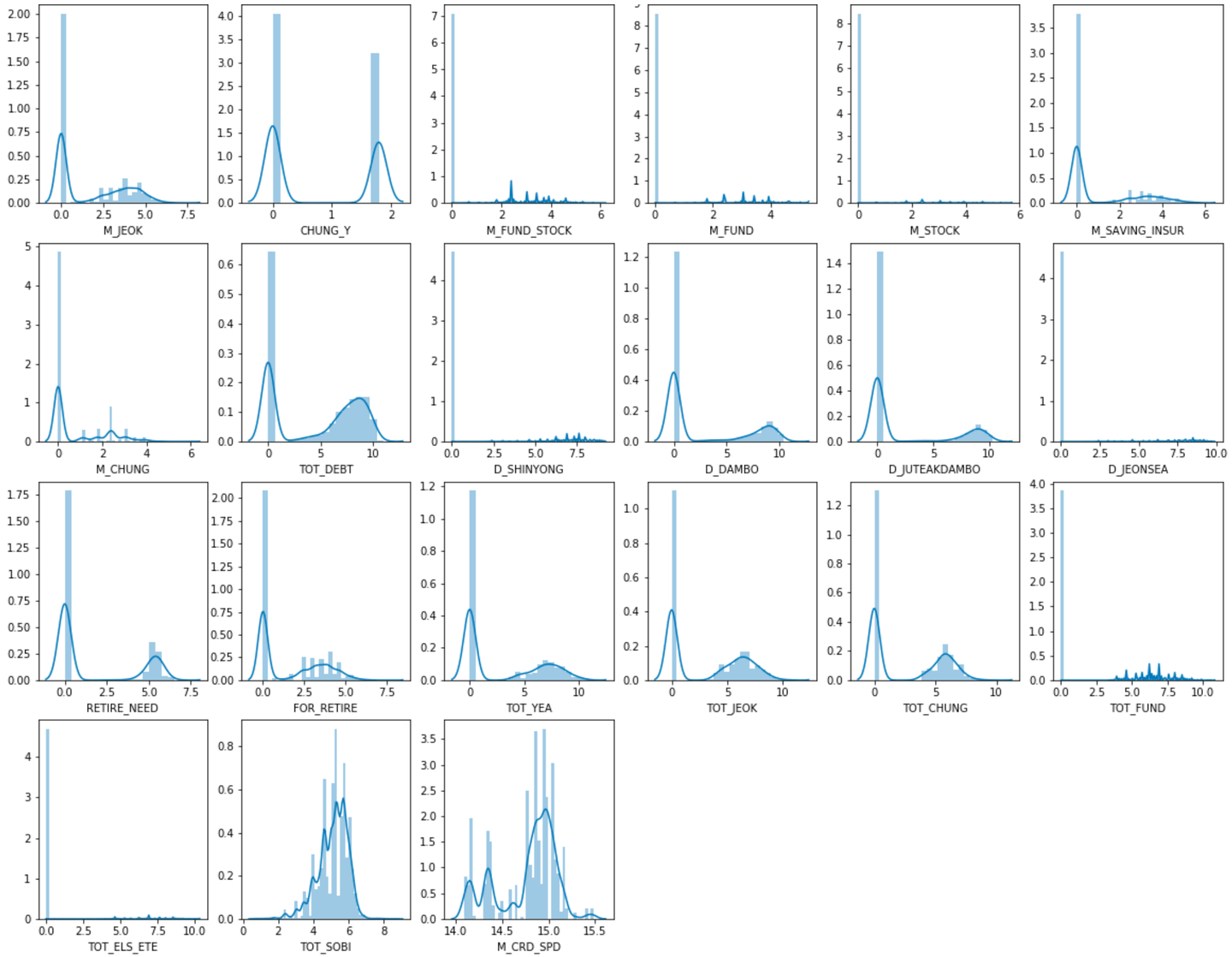
#맞벌이
맞벌이 여부 조사에 응답 의무가 없는 미혼자의 경우 전반적으로 낮은 소비를 보임. 맞벌이와 외벌이 고객의 평균 소비는 비슷하나 맞벌이의 경우 더 넓은 스펙트럼을 보임. 맞벌이고객과 외벌이고객의 총소비 상위 20명의 자녀 수 비율을 확인해보니 자녀 없음의 비율이 각각 30%, 10%로 딩크족 부부의 높은 소비로 인해 넓은 스펙트럼을 보이는 것으로 해석됨

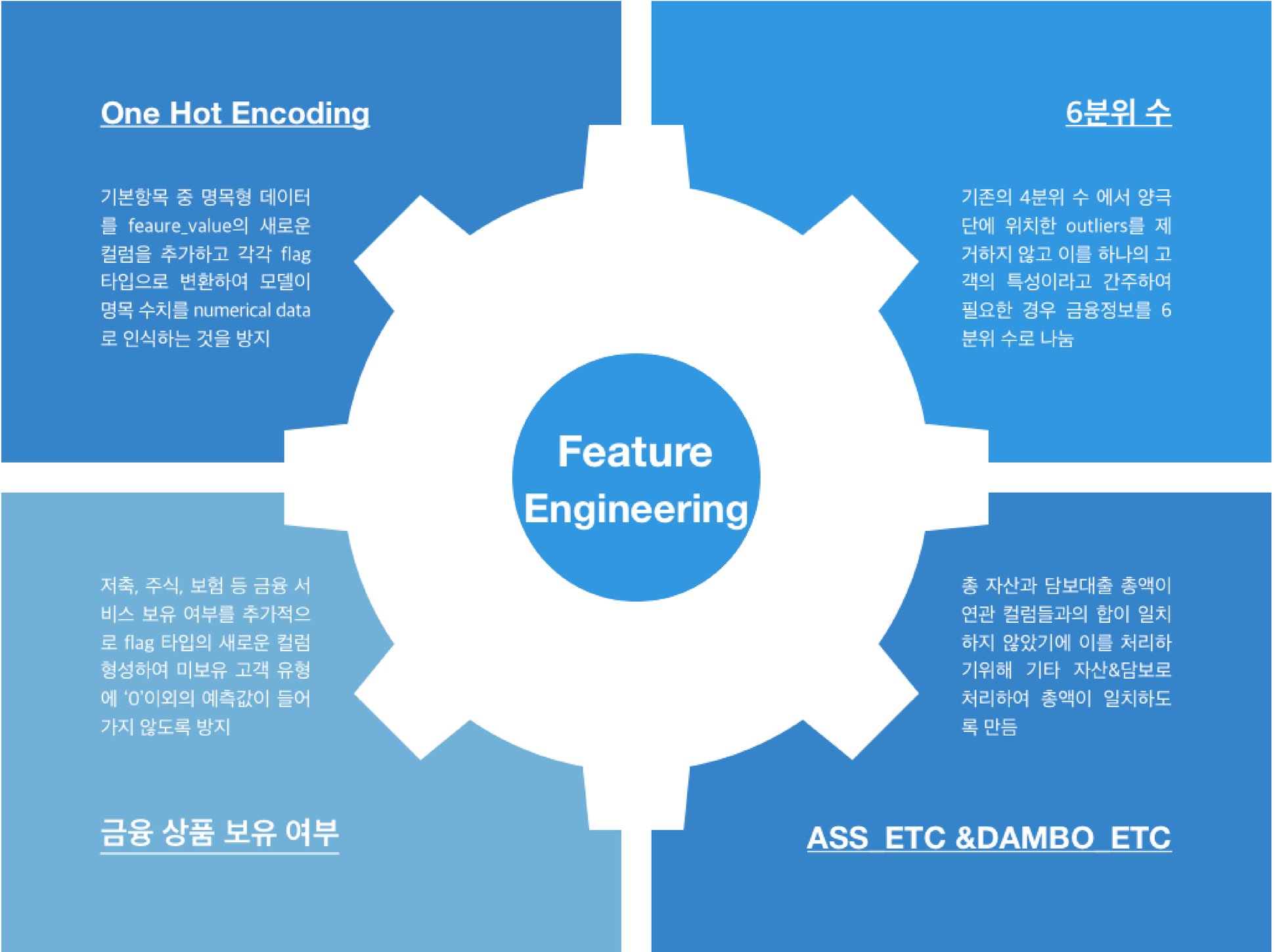


Data Understanding

금융정보 데이터 내 null 분포도

- 금융자산 중 전체 총합을 의미하는 컬럼을 제외하곤 대부분 50%이상의 고객은 금융상품에 가입되어 있지 않음을 보여줌
- 이를 활용하여 금융자산, 월 총 저축액 그리고 월 총 소비 금액을 예측하는데 노이즈가 발생할 것이라 가설을 세움
- 실제로 random forest regression을 이용하여 기본정보 8가지를 input하여 금융상품 가입 유무 및 금액을 예측 시도 결과 높은 RMSE 결과를 보여줌
- 때문에 이를 전체적으로 예측하기보다는 feature engineering 용도로 사용하기로 결정





Feature Engineering

- 금융상품 보유 여부를 활용하여 상품별 보유 금액을 예측하여 이를 합한 뒤 실제 전체 금액과 비교 결과, 편차가 크다는 것을 발견
- 부분을 각각 예측하여 전체를 예측하는 방법보다는 오히려 큰 부분인 금융자산, 월별 총 저축 & 소비 금액을 예측하는 것이 더 나은 결과를 이끌 수 있겠다고 가정을 세움

Model Building

Processing

Data Separations

- Null이 존재하는 경우의 수를 기준으로 Data를 분할
- 결혼, 맞벌이 유무, 자녀 수에 존재하는 Null의 유무에 따라 총 8가지 Data set이 형성
- 예를 들어 결혼, 맞벌이, 자녀 수의 Null을 유무를 조합하면 아래와 같음.
- (oxx, xox, xxo, oox, oxo, xoo, ooo, xxx)

Data Matching

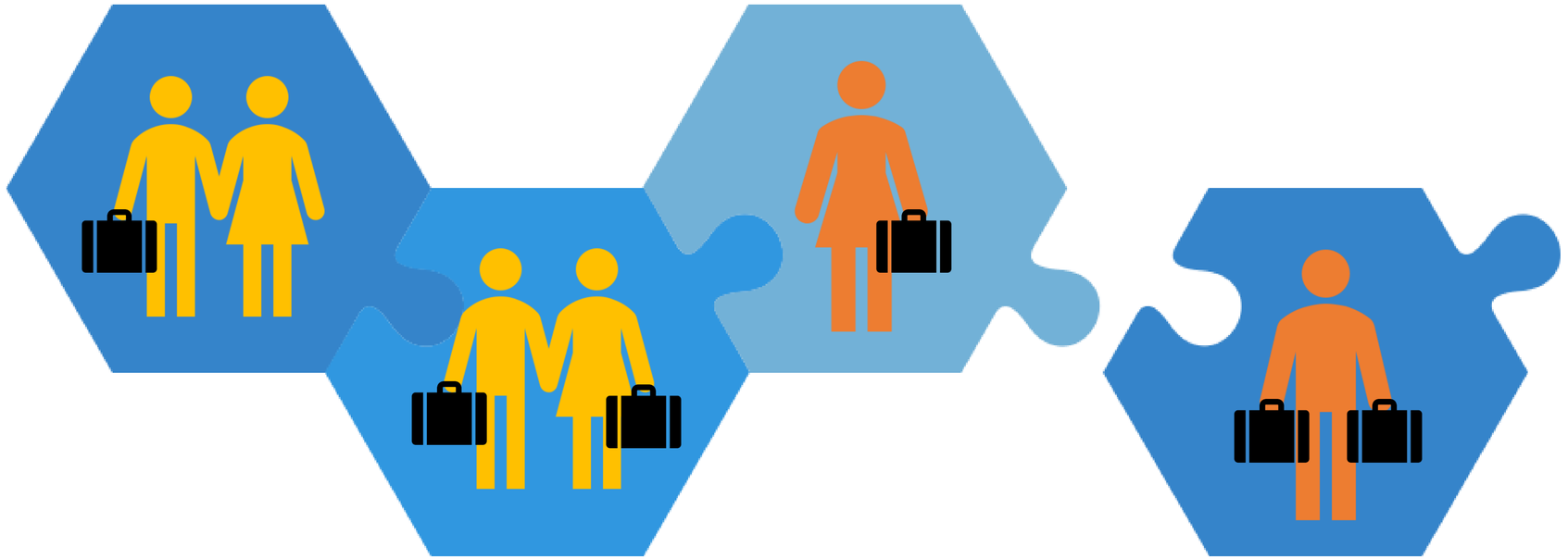
- 총 8개의 조건에 맞춰서 train과 test을 매칭하여 모델을 설계함
- Train의 경우 null이 존재하는 경우는 맞벌이 유무 밖에 존재하지 않기 때문에 이를 해결하기 위해 test에 Null이 존재할 경우 train의 해당 feature을 제외하여 동일한 조건으로 설정하여 모델을 설계

Feature Selection

- 금융자산, 부채 그리고 소비를 위한 각각의 모델에 entropy 지수를 활용하여 특정 설명변수 중 의미가 상대적으로 약한 것을 제거하여 각각의 모델을 좀더 심플하게 만들어 해당 feature의 설명력을 높이고자 하였음
- 이를 기반으로 test의 기본정보 8가지를 활용하여 자산, 부채 그리고 소비를 예측

Model Building

Data Separation



	idx	SEX_GBN	AGE_GBN	JOB_GBN	ADD_GBN	INCOME_GBN	MARRY_Y	DOUBLE_IN	NUMCHILD
0	5	0	2	2	4	2	1	NaN	0.0
1	7	0	3	2	2	3	1	NaN	0.0
2	16	0	2	11	5	2	1	NaN	0.0
3	18	1	2	9	5	3	1	NaN	0.0
4	29	0	3	2	5	2	1	NaN	0.0

- 기혼의 경우 맞벌이 유무를 묻는 것은 타당하나 미혼의 경우는 논리적이지 못함
- 그러다 보니, train 데이터에 미혼인 경우 맞벌이 컬럼의 value가 null로 처리됨
- 이러한 경우가 전체 데이터의 38%가 되기에 학습시킬 경우 문제가 있다고 판단
- 이를 해결하고자, train 데이터를 결혼 유무로 분리

Model Building

Data Separation &
Data Matching

- Train과 달리 Test의 컬럼 중 'MARRY_Y'와 'NUMCHILD'에 null이 경우의 수로 포함되어 있음
- 총 8가지 케이스가 존재하며 null을 nominal value로 지정하기 보단 test dataset을 분리하여 각각의 모델을 설계함
- 해당 컬럼에 null이 존재하면 train의 모델에서 해당 컬럼을 제외한 채 모델을 돌림으로써 test와 같은 조건으로 만듬

Test Data

	SEX_GBN	AGE_GBN	JOB_GBN	ADD_GBN	INCOME_GBN	MARRY_Y	DOUBLE_IN	NUMCHILD
0	1	2	2	1	1	1	1	0
1	1	2	2	1	1	2	1	0
2	1	2	2	1	1	1	2	0
3	1	2	2	1	1	2	2	0
4	1	2	2	1	1	1	1	1

- test 중 null이 존재하지 않는 케이스의 경우 2가지 train에서 null이 존재하지 않은 분리된 데이터를 이용해 학습

- test의 맞벌이 컬럼에 null이 존재할 경우 train 중 null이 존재하는 분리된 데이터를 사용해 학습
- 단 위의 데이터는 미혼의 경우이기에 해당 테스트의 데이터를 결혼 유무로 분리 후 학습시켜야 함

	SEX_GBN	AGE_GBN	JOB_GBN	ADD_GBN	INCOME_GBN	MARRY_Y	NUMCHILD
0	1	2	2	1	1	1	0
1	1	2	2	1	1	2	0
2	1	2	2	1	1	1	1
3	1	2	2	1	1	2	1

- 결혼과 맞벌이 컬럼에 null이 있는 경우, 해당 컬럼을 제외한 채 모델을 핏
- 이러한 경우 분리된 train이 아닌 전체를 사용함

	SEX_GBN	AGE_GBN	JOB_GBN	ADD_GBN	INCOME_GBN	NUMCHILD
0	1	2	2	1	1	0
1	1	2	2	1	1	1
2	1	2	2	1	1	2

Model Building

Feature Selection

6분위를 통한 Outlier 처리

- 금융 자산을 보유한 상위 1%는 10% 그룹내에서도 절대적으로 자산이 차이를 보일 수 있음.
- 실제 train의 분포를 살펴보아도 이러한 경우가 보였으며 이를 단순히 제거하기엔 최종 결과로 그룹이 형성이 될 때 이러한 유형의 고객이 지점에 방문하였을 때 제안할 수 있는 방안이 존재 하지 않음
- 이를 해결 하기 위해 양 극단의 outliers를 1, 6번 그룹으로 부여함

```
def sep6_percentile(data, col, name = 'GROUP_1'):

    qt = list(data[col].describe()[3:8])
    IQR1_5 = qt[3] + ((qt[3] - qt[1]) * 1.5) ; qt.insert(-1, IQR1_5)
    IQR3 = qt[3] + ((qt[3] - qt[1]) * 3); qt.insert(-1, IQR3)

    for i in range(len(qt) - 1):
        idx = list(data[data[col].between(qt[i], qt[i+1], inclusive=True)].index)
        data.loc[np.array(idx), name] = i+1
    data[name] = data[name].astype(int)
    return data
```

Model Building

Feature Selection

Entropy를 활용한 Feature Scoring

```
def get_entropy(data, col):
    set_col = list(set(data[col]))
    set_col.sort()
    print(set_col)
    for i in set_col:
        a = data[data[col] == i].iloc[:,:8].apply(lambda x : x.value_counts()).fillna(0)

        group_cnt = sum(a.iloc[:,1].values[np.where(a.iloc[:,1].values !=0)])

        if col in a.columns:
            search_col = a.columns.drop(col)
        else :
            search_col = a.columns

        print('\n')
        print(col + '_' + str(i))
        for j in search_col:
            wh = np.where(a[j].values in a[j].values !=0 )

            for_ent = 0

            for k in range(len(wh)):
                for_ent += -(a[j].values[wh[0][k]] / group_cnt * math.log(a[j].values[wh[0][k]] / group_cnt,2))
            print(j, '=>', '[',round(for_ent,4),']', end=' ')
```

- Linear regression 모델을 설계하는데 있어서 'Simple is better' 원칙으로 feature의 수를 줄이고자 함
- 이를 위해 Entropy 지수를 활용하여 0.5에 가까운 점수를 받은 각각의 feature를 제거
- 금융자산, 총 저축 금액 그리고 소비 예측 모델에 이를 각각 적용함

```
NCOME_GBN => [ 0.2807 ] MARR

OME_GBN => [ 0.113 ] MARRY_Y
```

```
GROUP_1_3
SEX_GBN => [ 0.4682 ] AGE_GBN => [ 0.4079 ] JOB_GBN => [ 0.4771 ] ADD_GBN => [ 0.1923 ] INCOME_GBN => [ 0.036 ] MARRY
_Y => [ 0.5291 ] DOUBLE_IN => [ 0.3764 ] NUMCHILD => [ 0.2924 ]

GROUP_1_4
SEX_GBN => [ 0.4461 ] AGE_GBN => [ 0.2222 ] JOB_GBN => [ 0.4769 ] ADD_GBN => [ 0.2388 ] INCOME_GBN => [ 0.0379 ] MARR
Y_Y => [ 0.4963 ] DOUBLE_IN => [ 0.426 ] NUMCHILD => [ 0.2546 ]

GROUP_1_5
SEX_GBN => [ 0.4228 ] AGE_GBN => [ 0.1735 ] JOB_GBN => [ 0.4848 ] ADD_GBN => [ 0.2435 ] INCOME_GBN => [ 0.0121 ] MARR
Y_Y => [ 0.4479 ] DOUBLE_IN => [ 0.4768 ] NUMCHILD => [ 0.2125 ]

GROUP_1_6
SEX_GBN => [ 0.3711 ] AGE_GBN => [ 0.1597 ] JOB_GBN => [ 0.4971 ] ADD_GBN => [ 0.3346 ] INCOME_GBN => [ 0.056 ] MARRY
_Y => [ 0.4594 ] DOUBLE_IN => [ 0.4676 ] NUMCHILD => [ 0.1754 ]
```

GOOD => INCOME_GBN, AGE_GBN, ADD_GBN, NUMCHILD
BAD => MARRY_Y, JOB_GBN, SEX_GBN

Model Building

Model Selection

```
models = {'rf': RandomForestRegressor(n_estimators=500, max_depth=10, random_state=42, n_jobs=-1),

          'lasso': make_pipeline(RobustScaler(), Lasso(alpha =0.5, random_state=3)),

          'Enet' : make_pipeline(RobustScaler(), ElasticNet(alpha=0.5, l1_ratio=.9, random_state=3)),

          'KRR' :  KernelRidge(alpha=0.6, kernel='polynomial', degree=2, coef0=2.5),

          'xgb':   xgb.XGBRegressor(colsample_bytree=0.4603, gamma=0.0468,
                                    learning_rate=0.05, max_depth=3,
                                    min_child_weight=1.7817, n_estimators=2200,
                                    reg_alpha=0.4640, reg_lambda=0.8571,
                                    subsample=0.5213, silent=1,
                                    random_state =7, nthread = -1),

          'lgbm': lgb.LGBMRegressor(objective='regression',num_leaves=5,
                                    learning_rate=0.05, n_estimators=720,
                                    max_bin = 55, bagging_fraction = 0.8,
                                    bagging_freq = 5, feature_fraction = 0.2319,
                                    feature_fraction_seed=9, bagging_seed=9,
                                    min_data_in_leaf =6, min_sum_hessian_in_leaf = 11)
        }
```

```
def score(model, X_test, y_true):
    model_prediction = model.predict(X_test)
    model_mse = mean_squared_error(y_true, model_prediction)
    model_rmse = np.sqrt(model_mse)
    model_mae = mean_absolute_error(y_true, model_prediction)
    return model_rmse, model_mae
```

```
def make_stack_model(data, target, models):
    #sc_train = make_dummy(data, ind_col, sep_col, target)
    #sc_train = pd.concat([sc_train, data.iloc[:,ind_col:]],axis = 1)
    sc_train = data

    x_train, x_test, y_train, y_test = train_test_split\
    (sc_train.drop(target, axis = 1), sc_train[target], test_size=0.3, random_state = 0)
    sc_dic = {}

    for name in models.keys():
        models[name].fit(x_train, y_train)
        sc_dic[name] = score(models[name], x_test, y_test)

    res_sort = sorted(sc_dic.items(), key= operator.itemgetter(1))
    res_sort = [ res_sort[:4][i][0] for i in range(4) ]

    meta = models[res_sort[0]]
    stack_model = StackingCVClassifier\
    (classifiers=[models[res_sort[1]], models[res_sort[2]],models[res_sort[3]]],meta_classifier=meta)

    return stack_model
```

Model Evaluation & Selection

- 대표적인 Regression model들을 사용하여 최종 타겟을 각각 예측함
- 이를 RMSE와 MAE로 비교하여 그 결과가 우수한 순서로 배정하고 1순위를 ‘meta classifier’로 지정한 ‘stack model’을 생성

Model Building

Final Result

```
li = ['data_ooo','data_oox','data_oxo','data_oxx','data_xoo','data_xox','data_xxo','data_xxx']
```

```
def processing(train_data, target_data, target = 'TOT_SOBI'):
    data_set = target_data
    data = train_data
    for i in li:
        set_col = data_set.columns[1:7][(data_set.loc[dic[i][0],'AGE_GBN':'NUMCHILD'] != 4).values]
        check = [i for i in ['MARRY_Y','DOUBLE_IN','NUMCHILD'] if i in set_col]

        test_set = data_set.loc[dic[i],set_col].astype('int')



        if check != []:
            train_set = data[flatten([set_col, data.iloc[:1,6:]])]\
            [((data[set_col][check].values == 4).sum(axis = 1) != len(check)).tolist()].fillna(0).astype('int')

        else :
            train_set = data.loc[data[set_col].dropna().index,\
                                 flatten([set_col, data.iloc[:1,6:]])].fillna(0).astype('int')

        #train_set['NEW_TOT_SOBI'] = np.sqrt(train_set['TOT_SOBI']\
        #* np.ceil(train_set['M_CRD_SPD'] / 10000)).astype('int')


        if 'INCOME_GBN' in set_col.values:
            dum_train = make_dummy(train_set,ind_col= len(set_col), sep = ['INCOME_GBN'], target = target)
            dum_test = make_dummy(test_set[set_col],ind_col= len(set_col), sep = ['INCOME_GBN'])

        else:
            dum_train = make_dummy(train_set,ind_col = len(set_col), sep = None, target = target)
            dum_test = make_dummy(test_set[set_col],ind_col= len(set_col))

        dum_test = dum_test[np.array(dum_test.columns.intersection(dum_train.columns))].dropna()
        dum_train = dum_train[flatten([np.array(dum_test.columns.intersection(dum_train.columns)), target])].dropna()

        model = make_stack_model(dum_train, target, models)
        model.fit(dum_train.iloc[:,:-1].values, np.log1p(dum_train[target].values))
        pred = model.predict(dum_test.values)
        data_set.loc[test_set.index, target] = np.expm1(pred)
```

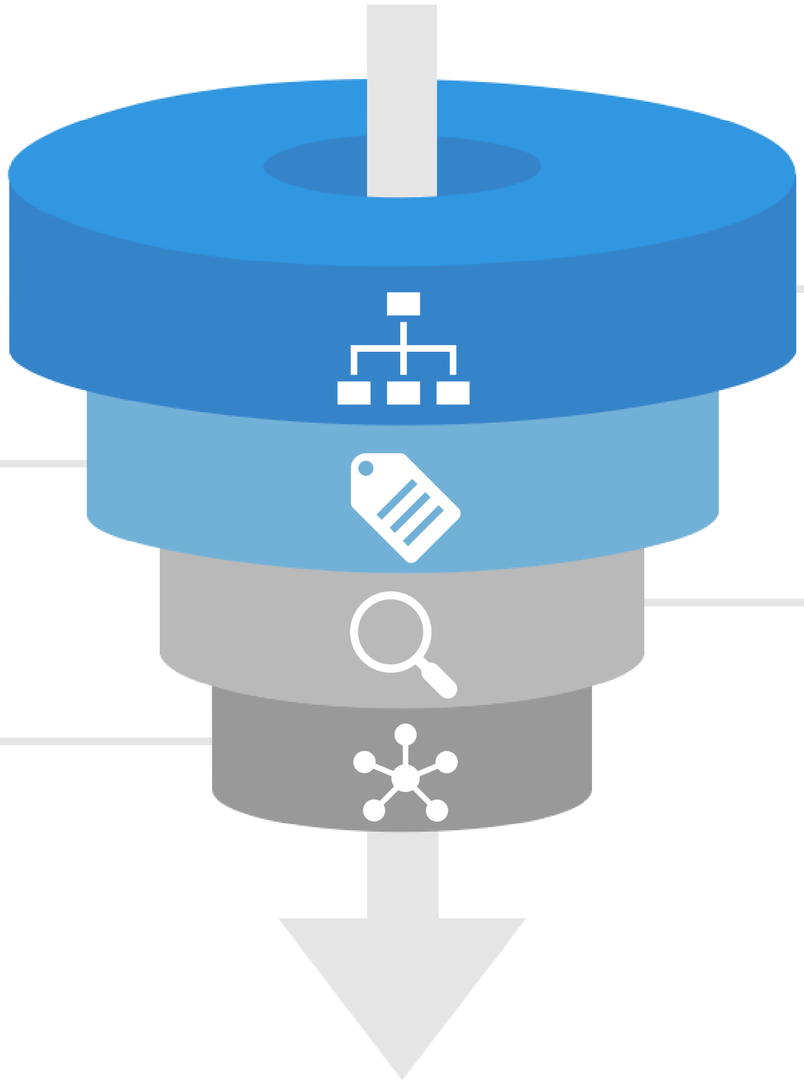
								NUMCHILD	ASS_FIN	M_TOT_SAVING	TOT_SOBI
								0.0	563.544955	35.556404	30.905158
								0.0	446.547546	22.229364	35.945343
								0.0	745.639160	11.234036	35.017663
3	1	2	2	1	1	1.0	2.0	0.0	563.544955	35.556404	30.905158
4	1	2	2	1	1	2.0	2.0	0.0	471.632699	30.146671	32.401138
5	1	2	2	1	1	4.0	2.0	0.0	304.827301	9.655828	18.822453

Question No.2

Peer Grouping

Overview

Clustering



예측된 결과를 6분위로 그룹화

KModes를 사용하여
Peer_Group id 생성

각 Peer_Group 간 특성이
존재하는지 탐색

Mapping table 생성

Model Building

Convert Data Type

6분위를 통한 Data Type 변환

- 예측된 금융자산, 저축 그리고 소비 데이터는 연속형 데이터로 군집화를 위해 명목 형으로 변환 작업이 필요했음

- 앞에서의 작업에서 논리적 일관성을 유지하기 위해 6분위를 사용하여 양 극단치에 1과 6을 부여해주는 6분위 수를 사용하여 데이터타입을 변환

	SEX_GBN	AGE_GBN	JOB_GBN	ADD_GBN	INCOME_GBN	MARRY_Y	DOUBLE_IN	NUMCHILD	ASS_GROUP	SAVING_GROUP	SOBI_GROUP
0	1	2	2	1	1	1	1	0	1	1	1
1	1	2	2	1	1	2	1	0	1	1	1
2	1	2	2	1	1	4	1	0	1	1	1
3	1	2	2	1	1	1	2	0	1	1	1
4	1	2	2	1	1	2	2	0	1	1	1
141745	2	6	11	5	7	2	2	4	5	4	4
141746	2	6	11	5	7	4	2	4	4	4	4
141747	2	6	11	5	7	1	4	4	4	4	4
141748	2	6	11	5	7	2	4	4	5	4	4
141749	2	6	11	5	7	4	4	4	5	4	4

Model Building

Generate Peer_Group id

KModes를 사용한 Peer_Group 생성

- 다양한 고객의 성향을 펀드투자 유형에 따라 반영할 수 있다고 가정
- 펀드 유형(5), 상품 속성(4) 그리고 위험등급(6) 가지를 조합하면 120개가 나오고 이를 n_clusters로 대입 (신한은행 기준)
- 그룹내의 평균과 중앙값이 근접해 있으며 각 그룹간 특성들이 분리 되어 있음을 볼 수 있음

```
from kmodes.kmodes import KModes

km = KModes(n_clusters=120, init='Huang', n_init=5, verbose=0)

km_pre = km.fit_predict(cat_data)

M2_set = pd.concat([pd.Series(km_pre, name = 'Peer_Group'), data.iloc[:,:11]], axis =1)
```

```
M2_set.groupby('Peer_Group').agg(['mean','medi
```

Group	mean	median	std	min								
0	1076.586747	984.398196	731.147337	72.144203	19839.583225	30.840641	26.867670	21.416210	3.433541	431.368927	119.832879	119.100018
1	5318.474856	4493.801854	3735.700712	248.022629	53657.089844	146.044352	138.422178	67.563329	8.411779	521.268059	275.729246	269.431012
2	1212.917992	1074.046509	1569.506503	273.757690	49258.863281	30.760400	27.884169	19.997236	2.949056	279.277130	124.461865	116.698957
3	3567.912125	3569.029932	1763.308024	467.066489	49258.863281	75.710169	66.249733	46.468303	8.411779	521.268059	184.333717	175.736702
4	6198.195477	5678.242432	4051.306679	248.022629	49258.863281	144.847196	154.711014	38.033678	3.433541	375.739337	331.335727	323.017987
5	2341.618668	2203.802407	810.907264	431.200552	8393.705817	60.273307	57.088064	19.769039	5.137300	521.268059	195.273793	195.225243

Model Building

Explore Peer Group

Peer Group 특징 분석

- 그룹118번에는 672명의 유형이 들어가 있으며, 싱글인 20대 여성이 주로 분포 되어 있음을 볼 수 있다. 또한 이중 약 40%가 강남 기타 구역에서 자영업을 하고 있으며 소득은 200미만으로 보이고 있음

- 반면 그룹 119번은 성별은 남성이 70%로 주를 이루고 있으나 연령대는 60에 이상으로 확연히 나이와 경기도 주민으로 유추할 수 있음. 또한 상대적으로 소득이 낮고 맞벌이를 통해 생활을 유지하는 것으로 예측

	Peer_Group	SEX_GBN	AGE_GBN	JOB_GBN	ADD_GBN	INCOME_GBN	MARRY_Y	DOUBLE_IN	NUMCHILD
0.0									31
1.0		89			80	160	507	84	63
2.0		583	369	38	392	508	65	63	391
3.0			53	39	64	3			82
4.0			136	54	88	1	100	525	105
5.0			34		48				
6.0			80	54					
7.0				46					
8.0				67					
9.0				260					
10.0				61					
11.0				53					
118.0	672								

	Peer_Group	SEX_GBN	AGE_GBN	JOB_GBN	ADD_GBN	INCOME_GBN	MARRY_Y	DOUBLE_IN	NUMCHILD
0.0									50
1.0		375			24	465	97	42	67
2.0		146	49	39	30	43	389	429	40
3.0			52	29	68	12			301
4.0			75	32	340		35	50	63
5.0			37		59				
6.0			308	36					
7.0				12		1			
8.0				44					
9.0				25					
10.0				45					
11.0				259					
119.0	521								

Mapping Table

Final Result

idx2	AGE_GBN	JOB_GBN	ADD_GBN	INCOME_GBN	MARRY_Y	DOUBLE_IN	NUMCHILD	ASS_FIN	M_TOT_SAVING	TOT_SOBI	PeerGroupNo
1	2	2	1	1	1	1	0	149.272428	18.691837	29.998798	12
2	2	2	1	1	2	1	0	393.793818	13.060307	26.676347	41
3	2	2	1	1	4	1	0	483.979332	5.877057	33.261362	62
4	2	2	1	1	1	2	0	149.272428	18.691837	29.998798	41
5	2	2	1	1	2	2	0	416.393802	13.540199	12.658689	41
6	2	2	1	1	4	2	0	625.474231	12.545228	21.619373	41
7	2	2	1	1	1	4	0	81.040711	14.214459	31.085495	72
8	2	2	1	1	2	4	0	578.829651	12.841987	11.654006	37
9	2	2	1	1	4	4	0	691.466982	3.826207	27.694914	14
10	2	2	1	1	1	1	1	149.272428	18.691837	29.998798	72
141728	6	11	5	7	2	2	2	7023.340218	165.322641	341.294479	31
141729	6	11	5	7	4	2	2	7861.679253	185.588409	335.441367	101
141730	6	11	5	7	1	4	2	3477.578409	92.336568	382.026955	117
141731	6	11	5	7	2	4	2	6103.046875	165.637541	330.524383	4
141732	6	11	5	7	4	4	2	7646.721394	130.501755	373.192939	47
141733	6	11	5	7	1	1	3	4564.534342	98.829735	278.022117	21

Question No.3

Asset Distributions

Peer Group 금액분포

Final Result

Peer Goup No.	비교대상컬럼	백분위수1	백분위수2	백분위수3	백분위수4	백분위수5	백분위수96	백분위수97	백분위수98	백분위수99	백분위수100
1	금융자산	439.980847	517.938921	528.854553	547.489929	570.082306	1780.977295	1804.16687	1966.234375	2657.786025	19839.58323
1	월저축금액	8.93113449	9.68348136	10.5022052	13.4823733	14.754992	55.69929123	59.26026657	61.68400284	140.392334	431.368927
1	월소비금액	54.3963766	60.434052	65.263555	65.3139535	69.3746132	172.0930585	174.8102284	182.4559631	283.9578857	447.6935012
2	금융자산	1418.64417	1774.71567	2229.53101	2522.0268	2682.76172	9861.216086	10618.52104	11503.17653	13399.7002	53657.08984
2	월저축금액	33.6443892	42.709386	52.4347898	56.9178411	59.781931	298.6232922	298.6232922	306.8197833	364.1718186	521.2680589
2	월소비금액	129.511983	140.611474	148.929806	149.349901	149.349901	397.0550251	410.2305908	413.2218085	440.7299084	624.9556885
3	금융자산	432.577772	478.664825	571.657017	621.694895	682.36033	1710.762294	1983.990349	3346.150879	4689.720703	49258.86328
3	월저축금액	9.07532644	12.8670441	13.7173491	14.630147	15.988182	51.8099786	60.77039106	70.27116039	107.8630108	279.2771301
3	월소비금액	71.1947556	72.8817479	79.8483096	82.236271	93.3987045	181.1881708	189.5910496	230.5829568	261.2492163	498.6551819
118	금융자산	997.42157	1720.95117	1842.28088	1871.84673	2041.9608	10612.53223	13621.09464	15412.37695	15575.14913	53657.08984
118	월저축금액	11.2954493	40.1633339	40.1633339	40.1633339	46.397292	266.327301	267.8132209	267.8132209	310.1730655	521.2680589
118	월소비금액	140.611474	160.190323	163.846916	164.242735	164.254946	458.2828447	473.3107621	485.2654105	513.9254837	584.900958
119	금융자산	99.0607877	99.0607877	104.146161	126.19853	129.156215	885.9407346	992.4607544	1124.985474	1224.491577	1862.849985
119	월저축금액	2.24152322	3.70665726	3.91788666	4.7817623	5.18052492	43.07281306	47.71622054	49.64726168	60.78238312	68.15310048
119	월소비금액	20.213787	22.6875301	28.1907288	29.9476539	29.9642989	96.74160767	103.6842804	104.7694523	112.2829735	125.9888858
120	금융자산	129.286894	189.312803	279.206131	284.656891	284.656891	1648.733521	1648.733521	1722.206934	3079.360296	6977.977261
120	월저축금액	2.2506641	2.29592621	4.94746325	6.05180311	7.26584277	19.86538368	23.75506983	30.47696357	30.47696357	143.5720215
120	월소비금액	12.6081142	14.015488	16.1163813	17.1283314	17.1283314	89.07105853	90.40971443	105.9487263	137.9664158	448.0298956

Question No.4

Alternative Data Collection

고객의 라이프스타일

보유한 카드 종류와 혜택을 통해 역으로 고객의 성향 분석

Life Stages

싱글, 커플, 패밀리 카테고리 내에서 추천하는 카드를 통하여 고객의 결혼여부 파악 가능

Education

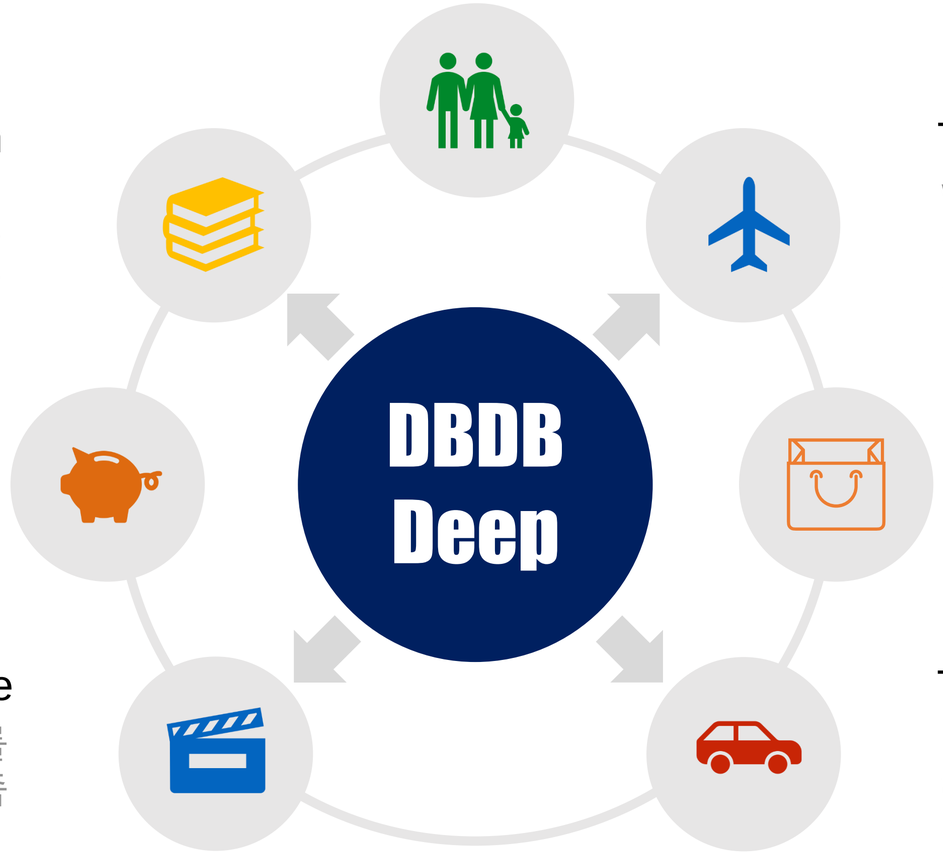
연령에 따라 20대 같은 경우 학생, 30대 이상의 경우 자녀가 있을 가능성이 존재함

Finances

보험, 은퇴자금, CMA 통장 여부와 주거래 은행이 신한은행인지 확인 가능

Life Style

문화생활과 외식이 잦은 고객인지를 통하여 소비성향을 단편적으로 예측



Travel & Leisure

Work-life-balance를 중시하는 직장인, 적립 유형에 따른 패키지/자유여행 선호를 예측 가능

Shopping

Online vs Offline 쇼핑 선호도를 통하여 IT 서비스 친숙도 평가

Transports

대중교통 할인과 주유 서비스 혜택을 통하여 차량 보유 여부 분석

지점 방문에 따른 고객 유형

지역/시간/목적/빈도를 통합하여 주 생활 지역과 직업 예측

01. Where

고객이 방문한 지점을 통해 거주지역 또는 직장의 위치 파악 가능. 주 지점과 방문 지역이 일치할 경우 실 거주지 추측이 가능하며 반대로 불일치할 경우 근무지가 지점과 가까울 가능성이 보임

02. When

시간, 요일에 따라 직업의 유형을 추측한다면 회사원의 경우 점심시간대를 선호하지만 상대적으로 오픈 시간이 늦은 자영업자는 오전시간대를 이용할 가능성이 존재.

03. For What

대출, 예금, 보험, 환전 등 다양한 목적에 따라 고객은 지점을 방문. 그 목적에 따라 고객의 직업 뿐만 아니라 관심있는 상품을 파악하여 데이터화 한다면 금융생활 정보지수에 반영 가능.

04. Frequency

특정한 목적을 가지고 지점을 방문하였지만 이는 단기성일 가능성이 존재. 때문에 빈도수를 활용하여 은행 서비스의 주 목적을 파악할 수 있어야 함.

Self-Check

데이터 분류 예측 문제에서 train과 test 데이터를 분류하지 않고 하나의 모델로 모델링 하였을 때 오차 값이 큰 문제 발생. 이를 MARRY_Y, DOUBLE_IN, NUMCHILD 을 기준으로 각 컬럼에 대하여 NULL값이 있는 경우, 없는 경우의 8개 그룹으로 나누어 각 그룹에 대해 모델링하여 오차를 줄여나갈 수 있었음

명목 형 변수를 그룹화 할 수 없는 문제가 존재하였음. 사용자의 기본 입력 정보는 명목형 변수들인데, K-Means 알고리즘은 명목 형 변수들에 대하여 클러스터링을 할 수 없어서 K-Modes 알고리즘을 사용함. K-Modes 알고리즘은 명목 형 변수를 군집화 할 수 있는 알고리즘으로 기본 개인정보 8개와 금융정보 3가지를 명목형 변수로 변환하여 120개의 그룹으로 클러스터링함

머신러닝, 딥러닝, K-Modes 알고리즘에 대한 수학적 이해가 부족하여 parameters를 효율적으로 활용하지 못하였음.

Greedy search를 활용하여 최적화 된 그룹의 수를 구했다면 더 나은 결과를 얻을 수 있었지만 시간적, 컴퓨터 성능의 제한적 환경으로 인해 대체 값으로 신한은행의 펀드 상품의 총 유형인 120가지를 사용함

120개로 분류한 모든 각 그룹마다 가지고 있는 특성을 찾아낼 수 있으면 고객에게 필요한 금융 서비스를 제안할 수 있을 것으로 보임

Q & A

감사합니다