산탄데르 고객 만족 예측 - 분류

학습 내용

- 캐글의 산탄데르 고객 만족 데이터 세트에 대해 고객 만족 여부를 XGBoost와 LightGBM을 활용하여 예측

대회 설명

- URL :[https://www.kaggle.com/competitions/santander-customer-satisfaction/overview (https://www.kaggle.com/competitions/santander-customer-satisfaction/overview)](https://www.kaggle.com/competitions/santander-customer-satisfaction/overview)
- 어떤 고객이 행복한 고객입니까? 이를 예측하는 대회
- 평가지표 : AUC - ROC-AUC(ROC 곡선 영역)

데이터 설명

- 데이터 다운로드 : [https://www.kaggle.com/c/santander-customer-satisfaction/data (https://www.kaggle.com/c/santander-customer-satisfaction/data)](https://www.kaggle.com/c/santander-customer-satisfaction/data)
  - train(59MB) : target를 포함한 데이터 셋
  - test(59MB) : target이 없는 데이터 셋
  - sample_submission : 제출용 데이터
- 370개의 피처로 이루어진 데이터
- 피처 이름은 전부 익명처리되어 있음.
- 클래스 레이블 명은 TARGET
  - 값이 1이면 불만족 고객.
  - 값이 0이면 만족 고객

데이터 로드 및 전처리

In [3]:

```python
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
import matplotlib
```

In [4]:

```
train = pd.read_csv("../../dataset/santander_customer/train.csv", encoding='latin-1'
test = pd.read_csv("../../dataset/santander_customer/test.csv", encoding='latin-1')
sub = pd.read_csv("../../dataset/santander_customer/sample_submission.csv")

train.shape, test.shape, sub.shape
```

Out[4]:

```
((76020, 371), (75818, 370), (75818, 2))
```

In [5]:

```
train.head()
```

Out[5]:

	ID	var3	var15	imp_ent_var16_ult1	imp_op_var39_comer_ult1	imp_op_var39_comer_ult3	imp_
0	1	2	23	0.0	0.0	0.0	
1	3	2	34	0.0	0.0	0.0	
2	4	2	23	0.0	0.0	0.0	
3	8	2	37	0.0	195.0	195.0	
4	10	2	39	0.0	0.0	0.0	

5 rows × 371 columns

In [6]:

```
train.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 76020 entries, 0 to 76019
Columns: 371 entries, ID to TARGET
dtypes: float64(111), int64(260)
memory usage: 215.2 MB
```

- 111개의 피처가 float형,
- 260개의 피처가 int형
- 모든 피처가 숫자형이며
- NUll값은 없다.

In [9]:

```
cnt=0
for one in train.columns:
    print(one, end="  ")
    cnt += 1
    if cnt%20==0:
        print()
```

```
ID  var3  var15  imp_ent_var16_ult1  imp_op_var39_comer_ult1  imp_op_v
ar39_comer_ult3  imp_op_var40_comer_ult1  imp_op_var40_comer_ult3  imp
_op_var40_efect_ult1  imp_op_var40_efect_ult3  imp_op_var40_ult1  imp_
op_var41_comer_ult1  imp_op_var41_comer_ult3  imp_op_var41_efect_ult1
imp_op_var41_efect_ult3  imp_op_var41_ult1  imp_op_var39_efect_ult1  i
mp_op_var39_efect_ult3  imp_op_var39_ult1  imp_sal_var16_ult1
ind_var1_0  ind_var1  ind_var2_0  ind_var2  ind_var5_0  ind_var5  ind_
var6_0  ind_var6  ind_var8_0  ind_var8  ind_var12_0  ind_var12  ind_va
r13_0  ind_var13_corto_0  ind_var13_corto  ind_var13_largo_0  ind_var1
3_largo  ind_var13_medio_0  ind_var13_medio  ind_var13
ind_var14_0  ind_var14  ind_var17_0  ind_var17  ind_var18_0  ind_var18
ind_var19  ind_var20_0  ind_var20  ind_var24_0  ind_var24  ind_var25_c
te  ind_var26_0  ind_var26_cte  ind_var26  ind_var25_0  ind_var25  ind
_var27_0  ind_var28_0  ind_var28
ind_var27  ind_var29_0  ind_var29  ind_var30_0  ind_var30  ind_var31_0
ind_var31  ind_var32_cte  ind_var32_0  ind_var32  ind_var33_0  ind_var
33  ind_var34_0  ind_var34  ind_var37_cte  ind_var37_0  ind_var37  ind
_var39_0  ind_var40_0  ind_var40
ind_var41_0  ind_var41  ind_var39  ind_var44_0  ind_var44  ind_var46_0
ind_var46  num_var1_0  num_var1  num_var4  num_var5_0  num_var5  num_v
ar6_0  num_var6  num_var8_0  num_var8  num_var12_0  num_var12  num_var
13_0  num_var13_corto_0
num_var13_corto  num_var13_largo_0  num_var13_largo  num_var13_medio_0
num_var13_medio  num_var13  num_var14_0  num_var14  num_var17_0  num_v
ar17  num_var18_0  num_var18  num_var20_0  num_var20  num_var24_0  num
_var24  num_var26_0  num_var26  num_var25_0  num_var25
num_op_var40_hace2  num_op_var40_hace3  num_op_var40_ult1  num_op_var4
0_ult3  num_op_var41_hace2  num_op_var41_hace3  num_op_var41_ult1  num
_op_var41_ult3  num_op_var39_hace2  num_op_var39_hace3  num_op_var39_u
lt1  num_op_var39_ult3  num_var27_0  num_var28_0  num_var28  num_var27
num_var29_0  num_var29  num_var30_0  num_var30
num_var31_0  num_var31  num_var32_0  num_var32  num_var33_0  num_var33
num_var34_0  num_var34  num_var35  num_var37_med_ult2  num_var37_0  nu
m_var37  num_var39_0  num_var40_0  num_var40  num_var41_0  num_var41
num_var39  num_var42_0  num_var42
num_var44_0  num_var44  num_var46_0  num_var46  saldo_var1  saldo_var5
saldo_var6  saldo_var8  saldo_var12  saldo_var13_corto  saldo_var13_la
rgo  saldo_var13_medio  saldo_var13  saldo_var14  saldo_var17  saldo_v
ar18  saldo_var20  saldo_var24  saldo_var26  saldo_var25
saldo_var28  saldo_var27  saldo_var29  saldo_var30  saldo_var31  saldo
_var32  saldo_var33  saldo_var34  saldo_var37  saldo_var40  saldo_var4
1  saldo_var42  saldo_var44  saldo_var46  var36  delta_imp_amort_var18
_1y3  delta_imp_amort_var34_1y3  delta_imp_aport_var13_1y3  delta_imp_
aport_var17_1y3  delta_imp_aport_var33_1y3
delta_imp_compra_var44_1y3  delta_imp_reemb_var13_1y3  delta_imp_reemb
_var17_1y3  delta_imp_reemb_var33_1y3  delta_imp_trasp_var17_in_1y3  d
elta_imp_trasp_var17_out_1y3  delta_imp_trasp_var33_in_1y3  delta_imp_
trasp_var33_out_1y3  delta_imp_venta_var44_1y3  delta_num_aport_var13_
1y3  delta_num_aport_var17_1y3  delta_num_aport_var33_1y3  delta_num_c
ompra_var44_1y3  delta_num_reemb_var13_1y3  delta_num_reemb_var17_1y3
delta_num_reemb_var33_1y3  delta_num_trasp_var17_in_1y3  delta_num_tra
```

```
sp_var17_out_1y3  delta_num_trasp_var33_in_1y3  delta_num_trasp_var33_
out_1y3
delta_num_venta_var44_1y3  imp_amort_var18_hace3  imp_amort_var18_ult1
imp_amort_var34_hace3  imp_amort_var34_ult1  imp_aport_var13_hace3  im
p_aport_var13_ult1  imp_aport_var17_hace3  imp_aport_var17_ult1  imp_a
port_var33_hace3  imp_aport_var33_ult1  imp_var7_emit_ult1  imp_var7_r
ecib_ult1  imp_compra_var44_hace3  imp_compra_var44_ult1  imp_reemb_va
r13_hace3  imp_reemb_var13_ult1  imp_reemb_var17_hace3  imp_reemb_var1
7_ult1  imp_reemb_var33_hace3
imp_reemb_var33_ult1  imp_var43_emit_ult1  imp_trans_var37_ult1  imp_t
rasp_var17_in_hace3  imp_trasp_var17_in_ult1  imp_trasp_var17_out_hace
3  imp_trasp_var17_out_ult1  imp_trasp_var33_in_hace3  imp_trasp_var33
_in_ult1  imp_trasp_var33_out_hace3  imp_trasp_var33_out_ult1  imp_ven
ta_var44_hace3  imp_venta_var44_ult1  ind_var7_emit_ult1  ind_var7_rec
ib_ult1  ind_var10_ult1  ind_var10cte_ult1  ind_var9_cte_ult1  ind_var
9_ult1  ind_var43_emit_ult1
ind_var43_recib_ult1  var21  num_var2_0_ult1  num_var2_ult1  num_aport
_var13_hace3  num_aport_var13_ult1  num_aport_var17_hace3  num_aport_v
ar17_ult1  num_aport_var33_hace3  num_aport_var33_ult1  num_var7_emit_
ult1  num_var7_recib_ult1  num_compra_var44_hace3  num_compra_var44_ul
t1  num_ent_var16_ult1  num_var22_hace2  num_var22_hace3  num_var22_ul
t1  num_var22_ult3  num_med_var22_ult3
num_med_var45_ult3  num_meses_var5_ult3  num_meses_var8_ult3  num_mese
s_var12_ult3  num_meses_var13_corto_ult3  num_meses_var13_largo_ult3
num_meses_var13_medio_ult3  num_meses_var17_ult3  num_meses_var29_ult3
num_meses_var33_ult3  num_meses_var39_vig_ult3  num_meses_var44_ult3
num_op_var39_comer_ult1  num_op_var39_comer_ult3  num_op_var40_comer_u
lt1  num_op_var40_comer_ult3  num_op_var40_efect_ult1  num_op_var40_ef
ect_ult3  num_op_var41_comer_ult1  num_op_var41_comer_ult3
num_op_var41_efect_ult1  num_op_var41_efect_ult3  num_op_var39_efect_u
lt1  num_op_var39_efect_ult3  num_reemb_var13_hace3  num_reemb_var13_u
lt1  num_reemb_var17_hace3  num_reemb_var17_ult1  num_reemb_var33_hace
3  num_reemb_var33_ult1  num_sal_var16_ult1  num_var43_emit_ult1  num_
var43_recib_ult1  num_trasp_var11_ult1  num_trasp_var17_in_hace3  num_
trasp_var17_in_ult1  num_trasp_var17_out_hace3  num_trasp_var17_out_ul
t1  num_trasp_var33_in_hace3  num_trasp_var33_in_ult1
num_trasp_var33_out_hace3  num_trasp_var33_out_ult1  num_venta_var44_h
ace3  num_venta_var44_ult1  num_var45_hace2  num_var45_hace3  num_var4
5_ult1  num_var45_ult3  saldo_var2_ult1  saldo_medio_var5_hace2  saldo
_medio_var5_hace3  saldo_medio_var5_ult1  saldo_medio_var5_ult3  saldo
_medio_var8_hace2  saldo_medio_var8_hace3  saldo_medio_var8_ult1  sald
o_medio_var8_ult3  saldo_medio_var12_hace2  saldo_medio_var12_hace3  s
aldo_medio_var12_ult1
saldo_medio_var12_ult3  saldo_medio_var13_corto_hace2  saldo_medio_var
13_corto_hace3  saldo_medio_var13_corto_ult1  saldo_medio_var13_corto_
ult3  saldo_medio_var13_largo_hace2  saldo_medio_var13_largo_hace3  sa
ldo_medio_var13_largo_ult1  saldo_medio_var13_largo_ult3  saldo_medio_
var13_medio_hace2  saldo_medio_var13_medio_hace3  saldo_medio_var13_me
dio_ult1  saldo_medio_var13_medio_ult3  saldo_medio_var17_hace2  saldo
_medio_var17_hace3  saldo_medio_var17_ult1  saldo_medio_var17_ult3  sa
ldo_medio_var29_hace2  saldo_medio_var29_hace3  saldo_medio_var29_ult1
saldo_medio_var29_ult3  saldo_medio_var33_hace2  saldo_medio_var33_hac
e3  saldo_medio_var33_ult1  saldo_medio_var33_ult3  saldo_medio_var44_
hace2  saldo_medio_var44_hace3  saldo_medio_var44_ult1  saldo_medio_va
r44_ult3  var38  TARGET
```

전체 데이터의 만족(0), 불만족(1) 비율

In [10]:

```
train['TARGET'].value_counts()
```

Out[10]:

```
0    73012
1     3008
Name: TARGET, dtype: int64
```

In [21]:

```
train['TARGET'].value_counts()[0]
```

Out[21]:

```
73012
```

In [32]:

```
satified = train['TARGET'].value_counts()[0]    # 만족
unsatified = train['TARGET'].value_counts()[1]  # 불만족
all_count = train['TARGET'].count()

print("{:.3f}% {:.3f}%".format( (satified/all_count) * 100 ,
                               (unsatified/all_count) * 100 ) )
```

```
96.043% 3.957%
```

In [33]:

```
train.describe()
```

Out[33]:

	ID	var3	var15	imp_ent_var16_ult1	imp_op_var39_comer_ult
count	76020.000000	76020.000000	76020.000000	76020.000000	76020.00000
mean	75964.050723	-1523.199277	33.212865	86.208265	72.36306
std	43781.947379	39033.462364	12.956486	1614.757313	339.31583
min	1.000000	-999999.000000	5.000000	0.000000	0.00000
25%	38104.750000	2.000000	23.000000	0.000000	0.00000
50%	76043.000000	2.000000	28.000000	0.000000	0.00000
75%	113748.750000	2.000000	40.000000	0.000000	0.00000
max	151838.000000	238.000000	105.000000	210000.000000	12888.03000

8 rows × 371 columns

- var3의 최소값이 -999999 - 이상치로 보임

In [34]:

```
train['var3'].value_counts()
```

Out[34]:

```
 2         74165
 8           138
-999999      116
 9           110
 3           108
           ...
 177           1
 87            1
 151           1
 215           1
 191           1
Name: var3, Length: 208, dtype: int64
```

In [35]:

```
# -999999를 가장 많은 나온 값으로 변경
train['var3'].replace(-999999, 2, inplace=True)
```

In [38]:

```
# 실제 확인
train.loc[ train['var3']==-999999, : ]
```

Out[38]:

ID	var3	var15	imp_ent_var16_ult1	imp_op_var39_comer_ult1	imp_op_var39_comer_ult3	imp_o

0 rows × 371 columns

In [39]:

```
## ID 열을 삭제
# train.drop('ID', axis=1, inplace=True)
train = train.loc[  :, "var3":  ]
train.head()
```

Out[39]:

	var3	var15	imp_ent_var16_ult1	imp_op_var39_comer_ult1	imp_op_var39_comer_ult3	imp_op_v
0	2	23	0.0	0.0	0.0	
1	2	34	0.0	0.0	0.0	
2	2	23	0.0	0.0	0.0	
3	2	37	0.0	195.0	195.0	
4	2	39	0.0	0.0	0.0	

5 rows × 370 columns

In [40]:

```
# 피처와 레이블를 지정.
# TARGET를 제외한 열을 입력으로(X), TARGET열을 y로 지정
X = train.iloc[:, :-1]
y = train['TARGET']

X.shape, y.shape
```

Out[40]:

```
((76020, 369), (76020,))
```

데이터 나누기

- 학습용 80%, 자체 검증용 20%

In [43]:

```
from sklearn.model_selection import train_test_split

X_train , X_test, y_train, y_test = train_test_split(X, y,
                                                     stratify=y,
                                                     test_size=0.2, random_state=0)

X_train.shape, X_test.shape, y_train.shape, y_test.shape
```

Out[43]:

```
((60816, 369), (15204, 369), (60816,), (15204,))
```

In [48]:

```
y_test[0:10]
```

Out[48]:

```
19379    0
66921    0
12415    0
9735     0
17997    0
67089    0
63376    0
6461     0
33577    0
59255    0
Name: TARGET, dtype: int64
```

In [45]:

```
## target(레이블) 분포비율
print( "학습용 레이블 분포 비율 : \n" , y_train.value_counts() / y_train.count() )
print( "테스트용 레이블 분포 비율 : \n" , y_train.value_counts() / y_train.count() )
```

```
학습용 레이블 분포 비율 :
 0    0.960438
1    0.039562
Name: TARGET, dtype: float64
테스트용 레이블 분포 비율 :
 0    0.960438
1    0.039562
Name: TARGET, dtype: float64
```

모델 생성 및 학습, 그리고 평가해 보기

In [47]:

```
%%time

from xgboost import XGBClassifier
from sklearn.metrics import roc_auc_score

xgb_model = XGBClassifier(n_estimators=500, random_state=156)
xgb_model.fit(X_train, y_train,
              early_stopping_rounds=100,
              eval_metric='auc',
              eval_set=[(X_train, y_train), (X_test, y_test)])
```

```
[0]	validation_0-auc:0.82570	validation_1-auc:0.79283
[1]	validation_0-auc:0.84010	validation_1-auc:0.80737
[2]	validation_0-auc:0.84361	validation_1-auc:0.81021
[3]	validation_0-auc:0.84783	validation_1-auc:0.81287
[4]	validation_0-auc:0.85123	validation_1-auc:0.81469
[5]	validation_0-auc:0.85518	validation_1-auc:0.81860
[6]	validation_0-auc:0.85922	validation_1-auc:0.81977
[7]	validation_0-auc:0.86238	validation_1-auc:0.82034
[8]	validation_0-auc:0.86570	validation_1-auc:0.82147
[9]	validation_0-auc:0.86798	validation_1-auc:0.82301
[10]	validation_0-auc:0.87104	validation_1-auc:0.82379
[11]	validation_0-auc:0.87448	validation_1-auc:0.82456
[12]	validation_0-auc:0.87687	validation_1-auc:0.82401
[13]	validation_0-auc:0.87918	validation_1-auc:0.82467
[14]	validation_0-auc:0.88081	validation_1-auc:0.82508
[15]	validation_0-auc:0.88331	validation_1-auc:0.82379
[16]	validation_0-auc:0.88569	validation_1-auc:0.82457
[17]	validation_0-auc:0.88674	validation_1-auc:0.82453
[18]	validation_0-auc:0.88885	validation_1-auc:0.82354
```

In [50]:

```
# 0의 예측 확률, 1의 예측 확률
pred_prob = xgb_model.predict_proba(X_test)[:, 1]
pred_prob
```

Out[50]:

```
array([0.00690398, 0.02649283, 0.01910355, ..., 0.01988643, 0.0117861
5,
       0.00611465], dtype=float32)
```

In [51]:

```
# 실제값(y_test)와 예측값(pred_prob)
xgb_roc_score = roc_auc_score(y_test, pred_prob, average='macro')
print("ROC AUC : {0:.4f}".format(xgb_roc_score))
```

```
ROC AUC : 0.8251
```

임계값을 정해서 평가해보기

In [68]:

```
pred_01 = pred_prob > 0.1
pred_01
```

Out[68]:

```
array([False, False, False, ..., False, False, False])
```

In [69]:

```
# 실제값(y_test)와 예측값(pred_prob)
xgb_roc_score = roc_auc_score(y_test, pred_01, average='macro')
print("ROC AUC : {0:.4f}".format(xgb_roc_score))
```

```
ROC AUC : 0.7161
```

GridSearchCV를 이용한 하이퍼 파라미터 튜닝

- max_depth, min_child_weight, colsample_bytree
- 먼저 2-3개 정도의 파라미터를 최적화 시킨 후,최적 파라미터를 기반으로 1-2개 파라미터를 결합하여 튜닝을 수행

In [70]:

```
%%time

from sklearn.model_selection import GridSearchCV

# 우선 하이퍼 파라미터 수행 속도를 향상을 위해 100으로
xgb_model1 = XGBClassifier(n_estimators=100, use_label_encoder=False)
params = {"max_depth":[5,7],
          "min_child_weight":[1,3],
          "colsample_bytree":[0.5, 0.75]}

gridcv = GridSearchCV(xgb_model1, param_grid=params, cv=3)
gridcv.fit(X_train, y_train, early_stopping_rounds=30,
            eval_metric='auc',
            eval_set = [(X_train, y_train), (X_test, y_test)])
```

```
/Users/toto/Documents/anaconda3/lib/python3.8/site-packages/xgboos
t/sklearn.py:793: UserWarning: `eval_metric` in `fit` method is dep
recated for better compatibility with scikit-learn, use `eval_metri
c` in constructor or`set_params` instead.
  warnings.warn(
/Users/toto/Documents/anaconda3/lib/python3.8/site-packages/xgboos
t/sklearn.py:793: UserWarning: `early_stopping_rounds` in `fit` met
hod is deprecated for better compatibility with scikit-learn, use `
early_stopping_rounds` in constructor or`set_params` instead.
  warnings.warn(
```

In [73]:

```
print("GridSearchCV  최적 파라미터 : ", gridcv.best_params_ )

pred_prob = gridcv.predict_proba(X_test)[:, 1]

# average='macro' : 각 레이블에 대한 측정항목을 계산하고 가중치가 적용되지 않은 평균을 찾습니다.
# default='macro'
xgb_roc_score = roc_auc_score(y_test, pred_prob, average='macro')
print("ROC AUC : {0:4f}".format(xgb_roc_score))
```

```
GridSearchCV  최적 파라미터 :  {'colsample_bytree': 0.5, 'max_depth': 5,
'min_child_weight': 3}
ROC AUC : 0.824543
```

실습해 보기

- colsample_bytree : 0.5, max_depth : 5, min_child_weight : 3로 설정
- n_estimators = 1000으로 증가, learning_rate를 조정해보고, reg_alpha를 추가하여 ROC_AUC의 값을 구해보자.

In [74]:

```
%%time

xgb_model_l = XGBClassifier(n_estimators=1000,
                            random_state= 77,
                            learning_rate=0.02,
                            max_depth=5,
                            min_child_weight=3,
                            colsample_bytree=0.5,
                            reg_alpha=0.03)

# 성능 평가 지표를 auc로, 조기 중단 파라미터 값은 200으로 설정하고 학습 수행
xgb_model_l.fit(X_train, y_train,
                early_stopping_rounds=200,
                eval_metric='auc',
                eval_set=[(X_train, y_train), (X_test, y_test)])
```

```
/Users/toto/Documents/anaconda3/lib/python3.8/site-packages/xgboos
t/sklearn.py:793: UserWarning: `eval_metric` in `fit` method is dep
recated for better compatibility with scikit-learn, use `eval_metri
c` in constructor or`set_params` instead.
  warnings.warn(
/Users/toto/Documents/anaconda3/lib/python3.8/site-packages/xgboos
t/sklearn.py:793: UserWarning: `early_stopping_rounds` in `fit` met
hod is deprecated for better compatibility with scikit-learn, use `
early_stopping_rounds` in constructor or`set_params` instead.
  warnings.warn(
```

In [75]:

```
pred_prob = xgb_model_l.predict_proba(X_test)[:, 1]
xgb_roc_score = roc_auc_score(y_test, pred_prob, average='macro')
print("ROC AUC : {0:4f}".format(xgb_roc_score))
```

```
ROC AUC : 0.826601
```

메모

- XGBoost는 GBM을 기반으로 하고 있기에, 수행시간이 어느정도 걸립니다.
- 앙상블 계열 알고리즘에서 하이퍼 파라미터 튜닝으로 성능 수치 개선이 급격하게 되는 경우는 많지 않습니다.

각 특징의 중요도 시각화

- xgboost 모듈의 시각화 기능을 갖는 plot_importance() 메서드를 이용

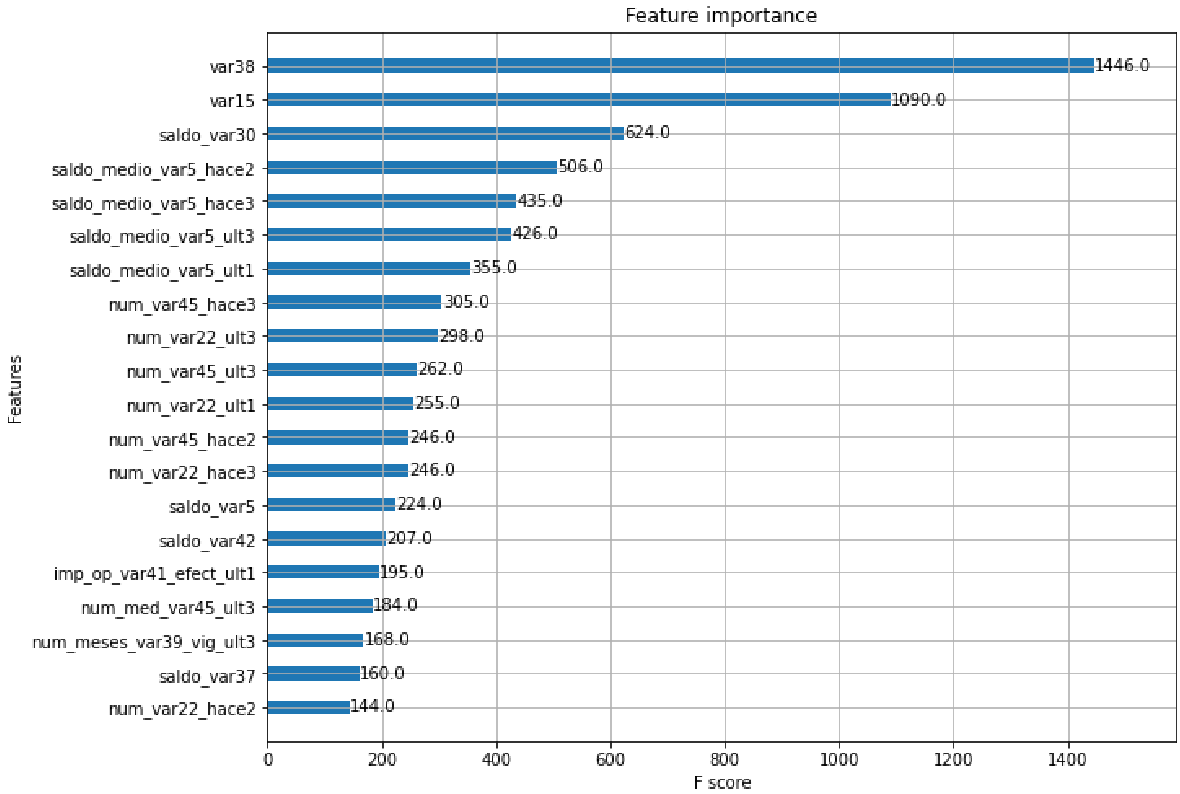
In [76]:

```
from xgboost import plot_importance
import matplotlib.pyplot as plt

fig, ax = plt.subplots(1,1, figsize=(10,8))
plot_importance(xgb_model_l, ax=ax, max_num_features=20, height=0.4)
```

Out[76]:

```
<AxesSubplot:title={'center':'Feature importance'}, xlabel='F score',
ylabel='Features'>
```



- var38, var15, saldo_var30 등이 유의한 변수 TOP3로 선정