

산탄데르 고객 만족도 분석 보고서

데이터 전처리와 앙상블 학습을 통한 고객만족도 예측



2025 April 14

우 성 욱

<목 차>

[0. 요약 2](#_Toc195525404)

[1. 프로젝트 개요 3](#_Toc195525405)

[1.1. 목표 3](#_Toc195525406)

[1.2. 데이터 3](#_Toc195525407)

[1.3. 주요 과제 3](#_Toc195525408)

[2. 데이터 구성 4](#_Toc195525409)

[2.1. 데이터 구성 4](#_Toc195525410)

[2.2. 이상치, 결측 치 및 중복 데이터 유무 4](#_Toc195525411)

[2.3. 클래스 불균형 여부 4](#_Toc195525412)

[3. 데이터 전처리 5](#_Toc195525413)

[3.1. 결측 치 및 이상치 처리 방법 5](#_Toc195525414)

[4. 프로젝트 흐름 6](#_Toc195525415)

[5. 모델링 7](#_Toc195525416)

[5.1. 사용 모델 7](#_Toc195525417)

[5.2. 모델 사용한 이유 7](#_Toc195525418)

[6. 성능 평가 지표(ROC AUC의 중요성) 8](#_Toc195525419)

[6.1. 클래스 불균형 문제 8](#_Toc195525420)

[6.2. ROC AUC 지표의 장점 8](#_Toc195525421)

[7. 실험 결과 9](#_Toc195525422)

[8. 전체 코드 10](#_Toc195525423)

# 요약

본 보고서는 산탄데르 은행에서 제공한 고객 만족도 데이터셋을 분석하여 불만족 고객을 사전에 식별하고 고객 이탈을 방지하기 위한 예측 모델을 개발한 과정과 결과를 담고 있습니다. 극심한 클래스 불균형, 이상치 존재, 370개의 특성을 가진 고차원 데이터와 같은 여러 과제를 해결하는 과정에서 다양한 전처리 기법과 모델링 접근법을 테스트했으며, 이상치 제거와 SMOTE 조합이 가장 우수한 성능을 보였습니다.

# 프로젝트 개요

## 목표

* 고객 만족 여부를 예측하여 불만족 고객을 사전에 식별
* 식별된 고객에 대한 조치를 통해 이탈 방지에 활용

## 데이터

* Santander 은행에서 제공한 고객 만족도 데이터 셋

(<https://www.kaggle.com/competitions/santander-customer-satisfaction>)

## 주요 과제

* 극심한 클래스 불균형 처리
* 이상치 존재로 이상치 미처리 및 처리 비교

# 데이터 구성

## 데이터 구성

|  |  |  |
| --- | --- | --- |
| 파일명 | 샘플 수 | 설명 |
| train.csv | 76,020 | 학습용 데이터, Target 값 포함 |
| test.csv | 75,818 | 테스트용 데이터, Target 값 미 포함 |

## 이상치, 결측 치 및 중복 데이터 유무

|  |  |
| --- | --- |
| 결측 치 |  |
| 이상 치 | 이상치 존재 |
| 중복 데이터 | 일부 중복 샘플 존재 (모델 성능에 미치는 영향은 미미) |

## 클래스 불균형 여부

아래 <Chart 1 Class 불균형>을 참고하면 목표 값(Target)이 불균형 하는 것을 알 수 있습니다. 목표 값의 비율 Target=0이 약 96%와 Target=1의 비율이 약 4%로 구성 되어 있습니다.

|  |
| --- |
| Chart 1 클래스 불균형 |

# 데이터 전처리

## 3.1. 결측 치 및 이상치 처리 방법

|  |  |
| --- | --- |
| 5% 이하 | 최대값, 평균값, 중앙값으로 대체 |
| 5% ~ 30% | 특정 컬럼 그룹별 평균값, 중앙값으로 대체 |
| 30% 이상 | 삭제하거나 새로운 범주를 생성 |

|  |
| --- |
|  |

IQR(Interquartile Range) 방식으로 이상치 탐색을 하였을 때, 대부분 데이터들이 5% ~ 30% 사이로 구성 되어 있는 것을 알 수 있습니다. 따라서 이상치 처리를 각 컬럼들의 중앙값으로 대체하여 실행하였습니다.

# 프로젝트 흐름

# 모델링

## 사용 모델

* XGBoostClassifier

## 모델 사용한 이유

* 우수한 예측 성능  
  테이블 형태의 구조화된 데이터에서 뛰어난 성능을 보이는 알고리즘으로, 많은 데이터 분석 경진대회에서 최상위 성적을 거둠
* 클래스 불균형 처리 능력  
  본 데이터 셋의 극심한 클래스 불균형(약 4%)에도 안정적인 성능을 보임
* 고차원 데이터 효과적 처리  
  370개 특성을 가진 고차원 데이터에서 자동으로 특성 중요도를 계산하고 중요 변수에 높은 가중치 부여
* 과 적합 방지 메커니즘  
  정규화 파라미터와 early stopping 기능을 통해 과 적합 위험 감소
* 계산 효율성  
  GPU 가속화(tree\_method='gpu\_hist')를 통한 빠른 학습 가능
* 해석 가능성  
  의사결정나무 기반 모델로서 변수 중요도를 통한 예측 결과 해석 가능

# 성능 평가 지표(ROC AUC의 중요성)

## 클래스 불균형 문제

* TARGET = 1 비율이 4%로 **극단적으로 불균형**된 상황

## ROC AUC 지표의 장점

* 분류 임계값에 영향을 받지 않음
* 모델의 전체 분류 능력을 측정
* False Positive Rate와 True Positive Rate를 모두 고려

# 실험 결과

|  |  |  |  |
| --- | --- | --- | --- |
| 실험 명 | 이상치 처리 | 클래스 처리 | ROC AUC |
| 이상치 미 제거  불균형 미 처리 | ❌ | ❌ | 0.8485 |
| 이상치 미 제거  Under Sampling | ❌ | Under Sampling | 0.8362 |
| 이상치 미 제거  SMOTE | ❌ | SMOTE | 0.8250 |
| 이상치 제거  불균형 미 처리 | 중앙값으로 대체 | ❌ | 0.8297 |
| 이상치 제거  Under Sampling | 중앙값으로 대체 | Under Sampling | 0.8198 |
| 이상치 제거  SMOTE | 중앙값으로 대체 | SMOTE | 0.9756 |

# 전체 코드

|  |
| --- |
| Library 로드 |
| # ==============================  # 1. 기본 라이브러리 로드  # ==============================  import pandas as pd # 데이터프레임 처리 라이브러리  import numpy as np # 수학 연산 및 배열 처리 라이브러리  import matplotlib.pyplot as plt # 데이터 시각화 라이브러리  # ==============================  # 2. 머신러닝 모델 관련 라이브러리  # ==============================  # (1) 앙상블 학습 모델  from sklearn.ensemble import VotingClassifier # 여러 개의 분류 모델을 조합하는 투표 기반 앙상블 학습  # (2) 개별 분류 모델  from sklearn.linear\_model import LogisticRegression # 로지스틱 회귀 분류기  from sklearn.neighbors import KNeighborsClassifier # K-최근접 이웃(KNN) 분류기  from sklearn.ensemble import RandomForestClassifier # 랜덤 포레스트 분류기  from sklearn.ensemble import GradientBoostingClassifier # 그래디언트 부스팅 트리 분류기  from sklearn.tree import DecisionTreeClassifier # 결정 트리 분류기  from sklearn.ensemble import AdaBoostClassifier # AdaBoost 분류기  # (3) 부스팅 기반 분류 모델 (XGBoost & LightGBM)  import xgboost as xgb # XGBoost 라이브러리 (트리 기반 부스팅 기법)  from xgboost import XGBClassifier # XGBoost 분류기  from lightgbm import LGBMClassifier # LightGBM 분류기  # ==============================  # 3. 모델 평가 및 성능 지표  # ==============================  from sklearn.metrics import confusion\_matrix # 혼동 행렬  from sklearn.metrics import precision\_score, recall\_score # 정밀도(precision) 및 재현율(recall)  from sklearn.metrics import f1\_score, roc\_auc\_score # F1-score 및 ROC-AUC 점수  from sklearn.metrics import accuracy\_score # 분류 모델의 정확도(accuracy)  # ==============================  # 4. 하이퍼파라미터 최적화  # ==============================  from hyperopt import hp # 하이퍼파라미터 탐색 공간 정의  from hyperopt import STATUS\_OK # 최적화 과정에서 상태 반환  from hyperopt import fmin, tpe, Trials # 최적의 하이퍼파라미터 탐색을 위한 함수들  # ==============================  # 5. 데이터셋 로드 및 데이터 분할  # ==============================  from sklearn.datasets import load\_breast\_cancer # 유방암 데이터셋 (예제 데이터셋)  from sklearn.model\_selection import train\_test\_split # 데이터 분할 (학습/테스트 세트)  from sklearn.model\_selection import cross\_val\_score # 교차 검증을 통한 모델 성능 평가  # ==============================  # 6. 데이터 전처리  # ==============================  from sklearn.preprocessing import StandardScaler # 데이터 표준화 (평균 0, 분산 1 변환)  from sklearn.preprocessing import MinMaxScaler # 데이터 Min-Max 스케일링 (0~1 범위 변환)  from sklearn.impute import SimpleImputer # 결측값 처리  from imblearn.under\_sampling import RandomUnderSampler # 불균형 데이터의 샘플링을 위한 언더샘플링  from imblearn.over\_sampling import RandomOverSampler # 불균형 데이터의 샘플링을 위한 오버샘플링  from imblearn.over\_sampling import SMOTE # SMOTE 기법 (Synthetic Minority Over-sampling Technique)  from scipy.stats import randint # 범위 내에서 랜덤값 생성  # ==============================  # 7. 통계 분석 관련 라이브러리  # ==============================  from scipy.stats import shapiro, skew, kurtosis # 데이터 분포 분석을 위한 함수들  from sklearn.model\_selection import GridSearchCV # 그리드 서치를 통한 하이퍼파라미터 최적화  from sklearn.model\_selection import RandomizedSearchCV # 랜덤 서치를 통한 하이퍼파라미터 최적화  import seaborn as sns # 고급 데이터 시각화 라이브러리  # ==============================  # 8. 기타 유틸리티  # ==============================  import time # 코드 실행 시간 측정  import warnings # 경고 메시지 무시 설정  warnings.filterwarnings('ignore') |
| 이상치 미 제거 및 불균형 미 처리 |
| cust\_train\_df = pd.read\_csv('./santander-customer-satisfaction/train.csv', encoding='latin-1')  cust\_test\_df = pd.read\_csv('./santander-customer-satisfaction/test.csv', encoding='latin-1')  cust\_train\_copy\_df = cust\_train\_df.copy()  cust\_test\_copy\_df = cust\_test\_df.copy()  cust\_train\_copy\_df.drop('ID', axis=1, inplace=True)  if 'ID' in cust\_test\_copy\_df.columns:  ids = cust\_test\_copy\_df['ID']  cust\_test\_copy\_df.drop('ID', axis=1, inplace=True)  else:  ids = cust\_test\_copy\_df.index  X\_features = cust\_train\_copy\_df.drop('TARGET', axis=1)  y\_labels = cust\_train\_copy\_df['TARGET']  X\_train, X\_test, y\_train, y\_test = train\_test\_split(  X\_features, y\_labels,  test\_size=0.3,  random\_state=156,  stratify=y\_labels  )  X\_tr, X\_val, y\_tr, y\_val = train\_test\_split(  X\_train, y\_train,  test\_size=0.3,  random\_state=0  )  xgb\_clf = XGBClassifier(  n\_estimators=500,  learning\_rate=0.05,  max\_depth=5,  random\_state=156,  eval\_metric='auc',  tree\_method='gpu\_hist',  early\_stopping\_rounds=100,  gpu\_id=0  )  xgb\_clf.fit(  X\_tr, y\_tr,  eval\_set=[(X\_val, y\_val)],  verbose=False  )  roc\_score = roc\_auc\_score(y\_test, xgb\_clf.predict\_proba(X\_test)[:, 1])  print(f"ROC AUC Score (이상치 미제거 + 불균형 미처리): {roc\_score:.4f}") |
| 이상치 미 제거 및 Under Sampling |
| cust\_train\_df = pd.read\_csv('./santander-customer-satisfaction/train.csv', encoding='latin-1')  cust\_test\_df = pd.read\_csv('./santander-customer-satisfaction/test.csv', encoding='latin-1')  cust\_train\_copy\_df = cust\_train\_df.copy()  cust\_test\_copy\_df = cust\_test\_df.copy()  cust\_train\_copy\_df.drop('ID', axis=1, inplace=True)  if 'ID' in cust\_test\_copy\_df.columns:  ids = cust\_test\_copy\_df['ID']  cust\_test\_copy\_df.drop('ID', axis=1, inplace=True)  else:  ids = cust\_test\_copy\_df.index  X\_features = cust\_train\_copy\_df.drop('TARGET', axis=1)  y\_target = cust\_train\_copy\_df['TARGET']  undersample = RandomUnderSampler(random\_state=42)  X\_resampled, y\_resampled = undersample.fit\_resample(X\_features, y\_target)  X\_train, X\_test, y\_train, y\_test = train\_test\_split(  X\_resampled, y\_resampled,  test\_size=0.3,  random\_state=156,  stratify=y\_resampled  )  X\_tr, X\_val, y\_tr, y\_val = train\_test\_split(  X\_train, y\_train,  test\_size=0.3,  random\_state=0  )  xgb\_clf = XGBClassifier(  n\_estimators=500,  learning\_rate=0.05,  max\_depth=5,  random\_state=156,  eval\_metric='auc',  tree\_method='gpu\_hist',  early\_stopping\_rounds=100,  gpu\_id=0  )  xgb\_clf.fit(  X\_tr, y\_tr,  eval\_set=[(X\_val, y\_val)],  verbose=False  )  roc\_score = roc\_auc\_score(y\_test, xgb\_clf.predict\_proba(X\_test)[:, 1])  print(f"ROC AUC Score (이상치 미제거 + 언더샘플링): {roc\_score:.4f}") |
| 이상치 미 제거 및 SMOTE |
| cust\_train\_df = pd.read\_csv('./santander-customer-satisfaction/train.csv', encoding='latin-1')  cust\_test\_df = pd.read\_csv('./santander-customer-satisfaction/test.csv', encoding='latin-1')  cust\_train\_copy\_df = cust\_train\_df.copy()  cust\_test\_copy\_df = cust\_test\_df.copy()  cust\_train\_copy\_df.drop('ID', axis=1, inplace=True)  if 'ID' in cust\_test\_copy\_df.columns:  ids = cust\_test\_copy\_df['ID']  cust\_test\_copy\_df.drop('ID', axis=1, inplace=True)  else:  ids = cust\_test\_copy\_df.index  X\_features = cust\_train\_copy\_df.drop('TARGET', axis=1)  y\_labels = cust\_train\_copy\_df['TARGET']  X\_train, X\_test, y\_train, y\_test = train\_test\_split(  X\_features, y\_labels,  test\_size=0.3,  random\_state=156,  stratify=y\_labels  )  smote = SMOTE(random\_state=156)  X\_train\_over, y\_train\_over = smote.fit\_resample(X\_train, y\_train)  xgb\_clf = XGBClassifier(  n\_estimators=500,  learning\_rate=0.05,  max\_depth=5,  random\_state=156,  eval\_metric='auc',  tree\_method='gpu\_hist',  early\_stopping\_rounds=100,  gpu\_id=0  )  xgb\_clf.fit(  X\_train\_over, y\_train\_over,  eval\_set=[(X\_test, y\_test)],  verbose=False  )  roc\_score = roc\_auc\_score(y\_test, xgb\_clf.predict\_proba(X\_test)[:, 1])  print(f"ROC AUC Score (이상치 미제거 + SMOTE): {roc\_score:.4f}") |
| 이상치 제거 및 불균형 미처리 |
| cust\_train\_df = pd.read\_csv('./santander-customer-satisfaction/train.csv', encoding='latin-1')  cust\_test\_df = pd.read\_csv('./santander-customer-satisfaction/test.csv', encoding='latin-1')  cust\_train\_copy\_df = cust\_train\_df.copy()  cust\_test\_copy\_df = cust\_test\_df.copy()  numeric\_cols = cust\_train\_copy\_df.select\_dtypes(include=['int64', 'float64']).columns.drop('TARGET')  for col in numeric\_cols:  Q1 = cust\_train\_copy\_df[col].quantile(0.25)  Q3 = cust\_train\_copy\_df[col].quantile(0.75)  IQR = Q3 - Q1  lower\_bound = Q1 - 1.5 \* IQR  upper\_bound = Q3 + 1.5 \* IQR  median = cust\_train\_copy\_df[col].median()    cust\_train\_copy\_df.loc[  (cust\_train\_copy\_df[col] < lower\_bound) | (cust\_train\_copy\_df[col] > upper\_bound),  col  ] = median  cust\_train\_copy\_df.drop('ID', axis=1, inplace=True)  if 'ID' in cust\_test\_copy\_df.columns:  ids = cust\_test\_copy\_df['ID']  cust\_test\_copy\_df.drop('ID', axis=1, inplace=True)  else:  ids = cust\_test\_copy\_df.index  X\_features = cust\_train\_copy\_df.drop('TARGET', axis=1)  y\_labels = cust\_train\_copy\_df['TARGET']  X\_train, X\_test, y\_train, y\_test = train\_test\_split(  X\_features, y\_labels,  test\_size=0.3,  stratify=y\_labels,  random\_state=156  )  X\_tr, X\_val, y\_tr, y\_val = train\_test\_split(  X\_train, y\_train,  test\_size=0.3,  random\_state=0  )  xgb\_clf = XGBClassifier(  n\_estimators=500,  learning\_rate=0.05,  max\_depth=5,  random\_state=156,  eval\_metric='auc',  tree\_method='gpu\_hist',  early\_stopping\_rounds=100,  gpu\_id=0  )  xgb\_clf.fit(  X\_tr, y\_tr,  eval\_set=[(X\_val, y\_val)],  verbose=False  )  roc\_score = roc\_auc\_score(y\_test, xgb\_clf.predict\_proba(X\_test)[:, 1])  print(f"ROC AUC Score: {roc\_score:.4f}") |

|  |
| --- |
| 이상치 제거 및 UnderSampling |
| cust\_train\_df = pd.read\_csv('./santander-customer-satisfaction/train.csv', encoding='latin-1')  cust\_test\_df = pd.read\_csv('./santander-customer-satisfaction/test.csv', encoding='latin-1')  cust\_train\_copy\_df = cust\_train\_df.copy()  cust\_test\_copy\_df = cust\_test\_df.copy()  numeric\_cols = cust\_train\_copy\_df.select\_dtypes(include=['int64', 'float64']).columns.drop('TARGET')  for col in numeric\_cols:  Q1 = cust\_train\_copy\_df[col].quantile(0.25)  Q3 = cust\_train\_copy\_df[col].quantile(0.75)  IQR = Q3 - Q1  lower\_bound = Q1 - 1.5 \* IQR  upper\_bound = Q3 + 1.5 \* IQR  median = cust\_train\_copy\_df[col].median()    cust\_train\_copy\_df.loc[  (cust\_train\_copy\_df[col] < lower\_bound) | (cust\_train\_copy\_df[col] > upper\_bound),  col  ] = median  cust\_train\_copy\_df.drop('ID', axis=1, inplace=True)  if 'ID' in cust\_test\_copy\_df.columns:  ids = cust\_test\_copy\_df['ID']  cust\_test\_copy\_df.drop('ID', axis=1, inplace=True)  else:  ids = cust\_test\_copy\_df.index  X\_features = cust\_train\_copy\_df.drop('TARGET', axis=1)  y\_target = cust\_train\_copy\_df['TARGET']  undersample = RandomUnderSampler(random\_state=42)  X\_resampled, y\_resampled = undersample.fit\_resample(X\_features, y\_target)  X\_train, X\_test, y\_train, y\_test = train\_test\_split(  X\_resampled, y\_resampled,  test\_size=0.3,  random\_state=156,  stratify=y\_resampled  )  X\_tr, X\_val, y\_tr, y\_val = train\_test\_split(  X\_train, y\_train,  test\_size=0.3,  random\_state=0  )  xgb\_clf = XGBClassifier(  n\_estimators=500,  learning\_rate=0.05,  max\_depth=5,  random\_state=156,  eval\_metric='auc',  tree\_method='gpu\_hist',  early\_stopping\_rounds=100,  gpu\_id=0  )  xgb\_clf.fit(  X\_tr, y\_tr,  eval\_set=[(X\_val, y\_val)],  verbose=False  )  roc\_score = roc\_auc\_score(y\_test, xgb\_clf.predict\_proba(X\_test)[:, 1])  print(f"ROC AUC Score (이상치 제거 + 언더샘플링): {roc\_score:.4f}") |

|  |
| --- |
| 이상치 제거 및 SMOTE |
| cust\_train\_df = pd.read\_csv('./santander-customer-satisfaction/train.csv', encoding='latin-1')  cust\_test\_df = pd.read\_csv('./santander-customer-satisfaction/test.csv', encoding='latin-1')  cust\_train\_copy\_df = cust\_train\_df.copy()  cust\_test\_copy\_df = cust\_test\_df.copy()  numeric\_cols = cust\_train\_copy\_df.select\_dtypes(include=['int64', 'float64']).columns.drop('TARGET')  for col in numeric\_cols:  Q1 = cust\_train\_copy\_df[col].quantile(0.25)  Q3 = cust\_train\_copy\_df[col].quantile(0.75)  IQR = Q3 - Q1  lower\_bound = Q1 - 1.5 \* IQR  upper\_bound = Q3 + 1.5 \* IQR  median = cust\_train\_copy\_df[col].median()    cust\_train\_copy\_df.loc[  (cust\_train\_copy\_df[col] < lower\_bound) | (cust\_train\_copy\_df[col] > upper\_bound),  col  ] = median  cust\_train\_copy\_df.drop('ID', axis=1, inplace=True)  if 'ID' in cust\_test\_copy\_df.columns:  ids = cust\_test\_copy\_df['ID']  cust\_test\_copy\_df.drop('ID', axis=1, inplace=True)  else:  ids = cust\_test\_copy\_df.index  X\_features = cust\_train\_copy\_df.drop('TARGET', axis=1)  y\_target = cust\_train\_copy\_df['TARGET']  smote = SMOTE(random\_state=42)  X\_resampled, y\_resampled = smote.fit\_resample(X\_features, y\_target)  X\_train, X\_test, y\_train, y\_test = train\_test\_split(  X\_resampled, y\_resampled,  test\_size=0.3,  random\_state=156,  stratify=y\_resampled  )  X\_tr, X\_val, y\_tr, y\_val = train\_test\_split(  X\_train, y\_train,  test\_size=0.3,  random\_state=0  )  xgb\_clf = XGBClassifier(  n\_estimators=500,  learning\_rate=0.05,  max\_depth=5,  random\_state=156,  eval\_metric='auc',  tree\_method='gpu\_hist',  early\_stopping\_rounds=100,  gpu\_id=0  )  xgb\_clf.fit(  X\_tr, y\_tr,  eval\_set=[(X\_val, y\_val)],  verbose=False  )  roc\_score = roc\_auc\_score(y\_test, xgb\_clf.predict\_proba(X\_test)[:, 1])  print(f"ROC AUC Score (이상치 제거 + SMOTE): {roc\_score:.4f}") |