[Objective]

각 고객의 정보를 기반으로 해당 고객이 대출한 돈을 갚을 수 있을지 없을지에 대한 확률을 예측

- 크게 두가지 개념으로 분류됨

1. **Supervised**: The labels are included in the training data and the goal is to train a model to learn to predict the labels from the features

**2. Classification**: The label is a binary variable, 0 (will repay loan on time), 1 (will have difficulty repaying loan)

* 변수 설명 (변수별로 파일이 구성됨. 총 7개의 파일. 매개변수는 SKP\_ID\_CURR)
* **application\_train/application\_test**: SK\_ID\_CURR로 구분됨. Training data는 0과1로 구분된 TARGET 변수를 가지고 있음(1은 예정된 상환일을 지체하여 상환에 어려움을 겪은 경우, 0은 정상적으로 상환한 경우)
  + 종속변수인 TARGET 변수의 분포를 보면, 1인 경우는 전체 중 8% 정도로 나타는데, 이를 통해 굉장히 불균형(highly unbalance)한 데이터임을 알 수 있습니다. 따라서 교차검증을 위한 샘플링을 할 때 주의해야 합니다(아래 SMOTE 내용 참고)
  + 독립변수

<https://medium.com/mighty-data-science-bootcamp/kaggle-%EB%8F%84%EC%A0%84%EA%B8%B0-home-credit-default-risk-part-1-735030d40ee0>

* 알아야 할 기본 정보
* SMOTE란?

구분할 각 분류에 해당하는 데이터의 비율이 반반이 아닌 경우 훈련 데이터 내 비율이 높은 분류 쪽으로 결과를 내놓는 모델을 만들게 될 수 있습니다. 이런 상황을 클래스 불균형(class imbalance)이라고 하며 이를 해결하는 방법으로 가중치(weight)를 조절하거나, 더 많은 비용(cost)을 부과하거나, 훈련 데이터를 직접 조절하는 방법이 있습니다(SMOTE).

SMOTE는 비율이 낮은 분류의 데이터를 고려하여 샘플을 만들어 내는 방법입니다. 기존의 샘플을 주변의 이웃을 고려해 약간씩 이동시킨 점들을 추가하는 방식으로 동작합니다.

* ROC\_AUC\_SCORE란?

이번 Kaggle의 평가방식을 보면 ROC 커브 아래 면적을 구하도록 되어있습니다.

AUC = AUROC (the Area Under a ROC Curve) : ROC 커브의 밑면적을 구한 값이 바로 AUC입니다. 이 값이 1에 가까울수록 성능이 좋은 것으로 판단할 수 있습니다.

1로 예측하는 기준을 쉽게 잡으면 민감도는 높아지는 대신 모든 경우를 1이라고 하므로, 특이도가 낮아집니다. 그러므로 이 두 값이 둘 다 1에 가까워야 의미가 있습니다. 민감도와 특이도는 서로 trade off 관계로 두 값의 합은 항상 1 입니다. AUC 값은 전체적인 민감도와 특이도의 상관 관계를 보여줄 수 있어 편리한 성능 척도 기준입니다.



**Part1. A Gentle Introduction**

Home Credit Default Risk에 대해 전체적으로 훑어보는 커널

1. Encoding

우선, Categorical Data를 인코딩합니다. 2개의 값으로만 구성되어 있는 feature는 Label Encoding을 하고, 2개 이상의 값으로 구성되어 있는 feature는 One-Hot Encoding을 합니다.

* One-Hot-Encoding을 하는 이유는 2개 이상의 값을 가지는 Feature에 대해 Label Encoding을 하면 성능이 떨어지기 때문. Label Encoding이 되면 범주형 데이터 값이 1, 2, 3, 4.. 등과 같이 숫자로 변환됨 이 1, 2, 3, 4… 는 단지 범주를 구분하기 위한 건데 실제 머신러닝 모델을 돌리면 모델이 이 숫자의 크기를 간주함.(즉 ‘4가 1보다 4배 크다’라는 식으로). 실제 1과 4는 각 범주를 숫자로 매핑시켜준 것에 불과한데 이를 크기가 있는 숫자로 인식을 한다면 성능에 문제를 일으키므로 따라서 데이터 값이 2개 이상이면 One-Hot-Encoding을 해주는 것

2. Feature Engineering

간단한 Feature Engineering으로는 Polynomial feature와 Domain Knowledge feature를 사용합니다. Polynomial feature는 특정 feature를 제곱, 세제곱 하거나 서로의 곱(interaction)을 통해 새로운 다항식으로 구성된 feature를 의미합니다. 중요한 feature인 `EXT\_SOURCE\_1`, `EXT\_SOURCE\_2`, `EXT\_SOURCE\_3`를 기반으로 Polynomial feature를 만들었습니다. 예를 들어, `EXT\_SOURCE\_1^2`, `EXT\_SOURCE\_2^2` 같이 하나의 feature를 제곱, 세제곱 등을 해주거나 `EXT\_SOURCE\_1` x `EXT\_SOURCE\_2`, `EXT\_SOURCE\_1` x `EXT\_SOURCE\_2^2`, `EXT\_SOURCE\_1^2` x `EXT\_SOURCE\_2^2`와 같이 서로를 곱해준 feature를 만들 수 있습니다.

그다음은 아래와 같이 Domain Knowledge feature를 사용했습니다.

- `CREDIT\_INCOME\_PERCENT`: the percentage of the credit amount relative to a client's income  
- `ANNUITY\_INCOME\_PERCENT`: the percentage of the loan annuity relative to a client's income  
- `CREDIT\_TERM`: the length of the payment in months (since the annuity is the monthly amount due  
- `DAYS\_EMPLOYED\_PERCENT`: the percentage of the days employed relative to the client's age

최종적으로 예측은 Logistic Regression과 Random Forest를 사용했습니다.

**Part2.** [**Introduction to Manual Featrue Engineering**](https://www.kaggle.com/willkoehrsen/introduction-to-manual-feature-engineering)

본 커널에서는 각종 Feature Engineering을 해주어 `A Gentle Introduction`보다 더 성능을 향상시켰습니다.

1. Aggregation

우선, Numeric Feature에 대해 Aggregation 한 feature를 추가.

`SK\_ID\_CURR`을 기준으로 Numerica feature의 `mean`, `max`, `min`, `sum`을 하여 이렇게 구한 Aggregation Feature를 기존의 train, test DataFrame에 추가해줬습니다.

Categorical Feature는 One-Hot-Encoding을 해준 뒤 `sum`, `mean` 의 Aggregation을 해주었습니다.

2. 결측값 처리

각 Feature별로 결측치가 90%가 넘어가는 Feature는 drop 해주려고 했으나 결측치가 90% 넘는 Feature는 없었습니다

3. Target값 과의 상관 계수, Feature 간 다중공선성

각 Feature와 Target 간의 상관계수도 구했습니다. 이 상관계수를 기반으로 가장 중요시되는 feature를 도출했습니다. 중요한 feature(즉, 상관계수의 절댓값이 높은 feature)에는 Feature Engineering으로 만든 feature들도 상당수 있었습니다.

또, 각 Feature 간의 다중공선성도 살펴봤습니다. 각 feature 간의 상관계수가 0.8이 넘어가면 둘 중 하나는 drop 해주었습니다.

최종적으로 모델은 LightGBM을 사용했습니다. `A Gentle Introduction`에서 간단한 feature engineering 한 것보다 성능이 크게 향상되었습니다.

**Part3.** [**Stacking Test-Sklearn, XGBoost, CatBoost, LightGBM**](https://www.kaggle.com/eliotbarr/stacking-test-sklearn-xgboost-catboost-lightgbm)

본 커널의 코드는 간단하지만 성능면에서는 뛰어납니다.

최적의 하이퍼파라미터가 이미 정해진 SKlearn, CatBoost, XGBoost, LightGBM를 Stacking하여 최종 예측값을 구했습니다. Feature Engineering은크게 없었으며, 모델의 하이퍼파라미터 및 Stacking에 중점을 둔 커널입니다. 하이퍼파라미터는 이미 다른 커널에서 구해놓은 것을 그대로 가져다가 썼기 때문에 코드가 간단해질 수 있었습니다.