Home Credit - Credit Risk Model Stability with XGBoost

컴퓨터공학과 나호영

Introduction

신용 기록이 없다는 것은 많은 것을 의미한다. 나이가 어린 고객이나 현금 사용을 선호하는 고객이 대출을 상환할 수 없다는 것을 의미하는 것은 아니기 때문이다. 기존의 금융 제공 업체들은 전통적으로 신용 기록이 거의 존재하지 않는 사람들의 대출을 거절할 가능성이 높았다. 하지만 대출 상환 능력을 판단할 수 있는 예측 모델을 사용한다면 대출이 가장 필요한 사람들에게 더욱 접근성이 높아진다. 본 프로젝트의 목표는 Kaggle에서 주최하는 Feature Competition인 Home Credit - Credit Risk Model Stability의 데이터를 사용하여 대출 상환 가능성을 예측하는 모델을 개발하고, 이를 통해 금융 기관이 보다 정확한 신용 평가를 할 수 있도록 돕는 것을 목적으로 한다. 모델을 개발하는 과정은 데이터 전처리, 특성 엔지니어링, 모델 훈련, 모델 평가 및 특성 중요도 분석 순으로 진행되었다.

Related-Model (XGBoost)

XGBoost(Extreme Gradient Boosting)는 그라디언트 부스팅(Gradient Boosting) 알고리즘을 기반으로 한 강력한 머신러닝 모델이다. XGBoost의 장점은 다음과 같다.

Gradient Boosting 기반의 성능 향상
XGBoost는 그라디언트 부스팅 기법을 사용하여 앙상블 학습을 수행한다. 이는 이전 모델의 오차를 보완하는 새로운 모델을 반복적으로 학습시킴으로써 전체 모델의 성능을 향상시키는 방법이다.

Regularization을 통한 과적합 방지
XGBoost는 모델 복잡도를 제어하기 위해 L1 및 L2 정규화를 포함한 다양한 정규화 기법을 제공한다. 이는 과적합을 방지하고 모델의 일반화 성능을 향상시킨다.

특성 중요도 제공
XGBoost는 각 특성의 중요도를 평가하여 모델이 어떤 특성에 가장 많은 영향을 받는지 이해할 수 있다. 이는 데이터의 해석과 모델 최적화에 유용하다.

그라디언트 부스팅 알고리즘의 강력함과 정규화를 통한 과적합 방지, 빠른 학습 속도와 확장성, 그리고 해석 가능한 특성 중요도 제공등의 장점은 본 프로젝트의 목표인 대출 상환 가능성 예측 모델로 사용하기에 적절하다.

1. Data Pre-Processing

1.1 데이터 형변환
제공되는 데이터는 원시 데이터이기 때문에 데이터들이 다양한 형식과 데이터 유형을 가질 수 있다. 데이터의 일관성을 유지하고 연산 효율성, 메모리 사용을 최적화하기 위해 데이터 형변환을 진행하였다.

1.2 날짜 처리
대출 신청일로부터 경과된 시간,즉 시간적 차이 분석을 용이하게 파악하기 위해 각 데이터 포인트의 대출 신청일로부터 경과된 일수를 계산하여 이를 새로운 특성으로 추가하였다.

1.3 불필요한 열 제거
데이터의 품질을 향상시키고 모델에 적합한 형태로 데이터를 만들기 위해, 70% 이상의 결측치를 가지거나, 고유값이 1개 이하이거나, 200개 이상의 문자열 값을 가지는 열을 제거하였다.

1.4 특성 공학
학습 모델의 성능을 향상시키기 위해 데이터로부터 유용한 특성을 추출하고 새로운 특성을 생성하는 과정이 필요하다. 데이터에서는 수치형, 날짜형, 문자형 등이 존재한다. 수치형과 날짜형에서는 최대값, 마지막값, 평균값을 추출하고 문자형에서는 최대값, 마지막값을 추출하였다.나머지 데이터형에 대해서는 최대값, 마지막값을 추출하였다. 추출된 값들을 리스트로 최종적으로 반환하였다.

1.5 메모리 사용 측정 및 최적화
데이터프레임이 사용하는 전체 메모리 용량을 측정하고 최적화된 데이터 타입을 적용함으로써 메모리 사용량을 줄였다. 데이터셋이 대용량이기 때문에 데이터 로드 및 처리하는 시간을 단축시키고 모델 학습의 속도를 향상시키기 위해 이 과정이 필요하다. 각 열의 데이터 타입을 확인하고 최적화된 데이터 타입으로 변환하였다. 정수형은 최소값과 최대값을 기준으로 해당 열에 필요한 가장 작은 정수형 데이터 타입을 선택하여 메모리 사용량을 줄였다. 부동소수점형 역시 마찬가지로 최소값과 최대값을 기준으로 가장 작은 부동소수점형 데이터 타입을 선택하여 메모리 사용량을 줄였다.

2. Training Model

XGBoost with StratifiedGroupKFold

2.1 StratifiedGroupKFold 설정

데이터셋을 k-fold cross-validation을 위해 분할할 때 사용되는 방법으로 일반적인 StratifiedKFold와 달리 그룹 정보를 고려하여 데이터를 분할한다. 각 fold에서 클래스의 레이블 분포를 보존하면서 그룹간의 데이터가 고르게 분할되도록 도와주기 때문에 모델이 더 일반화된 성능 평가를 가능하게 한다. 본 프로젝트에서는 5개의 fold로 데이터를 분할하고 shuffle을 false로 설정하여 데이터를 섞지 않고 순서대로 분할하였다. 데이터를 여러 개의 fold로 나누어 학습과 검증을 번갈아가며 수행함으로써, 모델이 특정 데이터셋에 과적합되지 않고 다양한 데이터에서 일반화될 수 있도록 했다.

2.2 모델 파라미터 설정

params 딕셔너리에 XGBoost 모델의 주요 파라미터들을 설정한다. 예를 들어, booster, objective, eval_metric, max_depth, learning_rate 등의 파라미터가 포함된다. tree_method는 GPU를 사용할 수 있으면 'gpu_hist'로 설정하고, 그렇지 않으면 'auto'로 설정한다. random_state는 재현 가능성을 위해 설정한 난수 시드이다.

2.3 5-fold cross-validation

데이터셋을 5개의 fold로 나누고, 각 fold에 대해 학습 데이터와 검증 데이터를 나눈 뒤 xgb.DMatrix를 사용하여 XGBoost가 사용할 수 있는 형식으로 데이터를 변환하였다. 모델의 학습을 진행하면서 early_stopping_rounds를 설정하여 학습 중에 성능이 개선되지 않으면 학습을 중단하였다. 또한 각 fold에서 검증 세트에 대해 최적의 반복 횟수, best_iteration을 결정하고 해당 반복까지의 예측을 수행하며 AUC 점수를 계산하고 모든 fold에서 계산된 AUC 점수를 cv_scores 리스트에 저장하였다.

2.4 성능 모니터링 및 결과 분석

XGBoost를 사용하여 5개의 fold에서 교차 검증을 수행하고 각 fold에서 훈련 세트와 검증 세트에 대한 AUC 점수, 즉 성능 지표에 대한 모니터링을 진행하였다. 다음은 XGBoost 모델 훈련 과정 중에서 verbose_eval 옵션을 사용하여 초기 라운드와 반복이 진행된 라운드의 성능 지표에 관한 로그이다.

```
[0]     train-auc:0.70237       valid-auc:0.64751
[181]   train-auc:0.98820       valid-auc:0.73343
[0]     train-auc:0.70782       valid-auc:0.67067
[200]   train-auc:0.98912       valid-auc:0.74143
[213]   train-auc:0.98998       valid-auc:0.74031
[0]     train-auc:0.67743       valid-auc:0.67698
[197]   train-auc:0.98654       valid-auc:0.77845
[0]     train-auc:0.68986       valid-auc:0.62665
[187]   train-auc:0.98509       valid-auc:0.74116
[0]     train-auc:0.72283       valid-auc:0.63371
[200]   train-auc:0.98908       valid-auc:0.72326
[202]   train-auc:0.98922       valid-auc:0.72340
```

결과를 살펴보면 학습 초기에는 Train-AUC와 Valid-AUC 모두 낮은 값을 가지고 시작한다. 라운드가 진행됨에 따라 Train-AUC와 Valid-AUC 모두 점차 증가하며 학습 데이터에 대해 성능이 올라가는 것을 확인할 수 있다.

아래는 훈련이 종료된 후에 Cross-Validation을 통해 얻은 각 Fold 별 최대 AUC 점수를 나타낸 결과이다.

```
CV AUC scores:  [0.7397231066556114, 0.7456849007133438, 0.7854618401478609, 0.7417501725327813, 0.7369001802359127]
Maximum CV AUC score:  0.7854618401478609
```

	Maximum CV AUC Score
Fold 1	0.739723106655611
Fold 2	0.745684900713344
Fold 3	0.785461840147861
Fold 4	0.741750172532781
Fold 5	0.736900180235913

이를 살펴보면 Fold 3에서의 AUC 점수가 가장 높은 것을 확인할 수 있다. 위의 로그와 비교해보면 Fold 3의 초기 라운드에서 Train-AUC와 Valid-AUC 의 차이가 거의 나지 않는 것을 확인할 수 있다. 이는 모델이 학습 데이터에 대해 잘 일반화되었음을 나타낸다. 따라서 본 프로젝트에서는 Fold 3을 나타내는 fitted_models[2] 를 최종 모델로 선택한다.

2.5 Feature Importance

특성 중요도 시각화

XGBoost 라이브러리에서는 학습된 특성의 중요도를 그래프로 시각화해주는 plot_importance 함수를 제공한다. 오른쪽 그래프는 선택된 fitted_models[2] 모델을 바탕으로 중요도 유형을 weight으로 설정하여 각 특성이 얼마나 많은 트리 분기에서 사용되었는지 나타내는 그래프를 시각화한 결과이다.

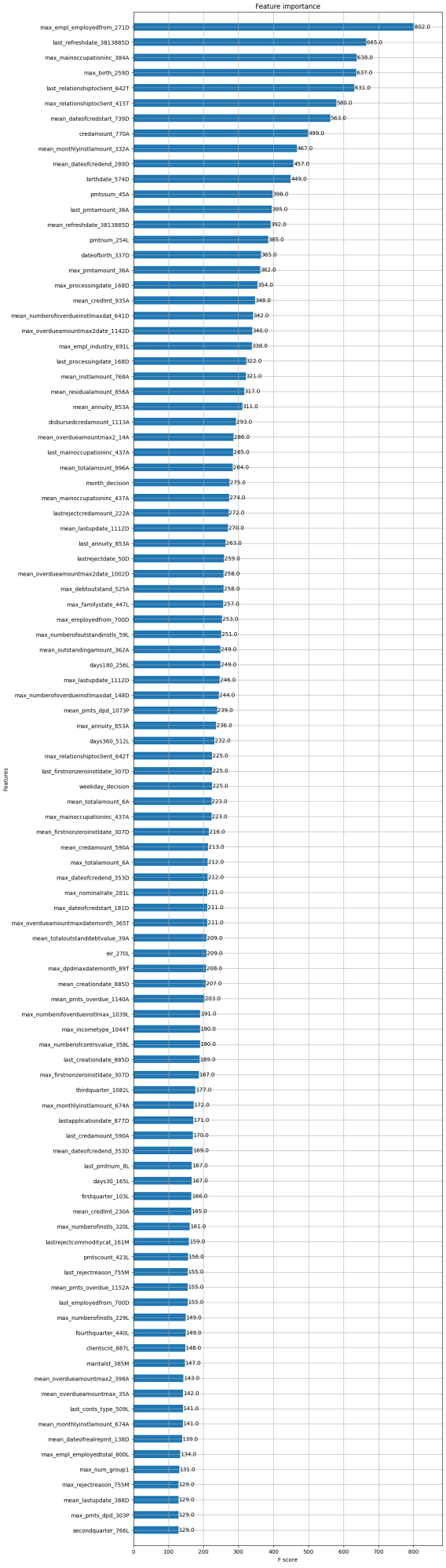
중요도가 높은 특성들은 모델 예측에 중요한 역할을 한다고 볼 수 있다. 이러한 특성들은 모델 성능에 긍정적인 영향을 미칠 가능성이 높다고 해석된다. 중요도가 낮은 특성들은 모델 에측에 크게 기여하지 않는 것으로 볼 수 있다. 따라서 이러한 특성들을 제거함으로써 모델의 복잡성을 줄이고, 계산 비용을 절감하며, Overfitting 가능성을 낮출 수 있다.

특성 중요도 데이터 처리

get_score 함수를 사용하여 계산된 특성의 중요도를 딕셔너리 형태로 반환받는다. 이를 데이터프레임으로 변환하는데, 특성과 그에 대응하는 중요도 값을 포함하여 변환한다. 이후 중요도 기준으로 내림차순으로 정렬한다.

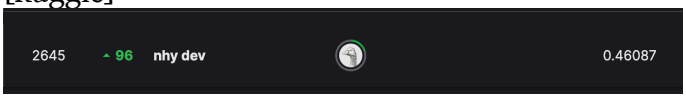
```
                          features  importance
0         max_empl_employedfrom_271D       802.0
1          last_refreshdate_3813885D       665.0
2         max_mainoccupationinc_384A       638.0
3                    max_birth_259D       637.0
4     last_relationshiptoclient_642T       631.0
..                               ...         ...
292  pctinstlsallpaidlate4d_3546849L         1.0
293                  last_sex_738L         1.0
294                clientscnt_493L         1.0
295     last_empls_economicalst_849M         1.0
296    last_empls_employer_name_740M         1.0

[297 rows x 2 columns]
Number of features which are not important: 173
```

정렬된 특성들을 바탕으로 중요하지 않은 특성들을 제거했다. 이 때 임계값을 설정하는 것이 중요한데 본 프로젝트에서는 80으로 설정하여 중요도가 80 미만인 특성들을 제거하였다. 총 173개의 특성들이 임계값 미만에 해당되는 제거 대상 특성인 것을 확인할 수 있다.



3. Result

[kaggle]



위는 Home Credit - Credit Risk Model Stability에 참가한 Leaderboard 결과이다. 3,856 참가 팀 중에서 2645 등을 기록하였다.

4. Discussion with Comparing to LightGBM

대회가 끝나고 나서 리더보드에 상위 등수를 기록한 팀의 코드들을 살펴보니 LightGBM 모델을 선택한 것을 확인할 수 있었다. XGBoost와 LightGBM을 비교하기 위해 XGBoost에 LGBM과 비슷한 파라미터를 사용하고 테스트 세트와 검증 세트 예측을 위한 함수도 같은 것을 사용했더니 XGBoost의 성능과 LightGBM의 성능이 유사한 것을 확인할 수 있었다. 하지만 실제로 제출하니 XGBoost의 점수가 LightGBM 보다 확연히 낮은 것을 확인할 수 있었다. 이에 대한 이유는 다음과 같은 가능성들이 존재한다.

알고리즘의 차이
XGBoost와 LightGBM은 모두 부스팅 알고리즘을 기반으로 하지만 구체적인 트리 구성 방법이나 학습 방식에서 차이가 존재한다. 예를 들어, Leaf-wise 트리 확장 방법을 사용하는 XGBoost는 데이터가 크고 깊이가 깊은 트리를 만들 수 있지만 Leaf-wise는 과적합 가능성이 있고 데이터가 작을 때는 성능이 떨어질 수 있다. 반면 LightGBM은 Level-wise 트리 확장 방법을 사용하며 데이터가 많고 적절한 깊이의 트리를 효과적으로 구성할 수 있다.

하이퍼 파라미터
XGBoost와 LightGBM은 각각 다른 하이퍼 파라미터를 가지고 있으며 이들을 최적화하는 방법에도 차이가 있을 수 있다. 같은 매개변수를 사용하더라도 알고리즘의 구현 방식에 따라 최적의 하이퍼 파라미터 값이 다를 수 있다.

이외에도 다양한 이유들이 존재할 수 있지만, 테스트 세트와 검증 세트에서 비슷한 성능을 내는데 실제 제출 후 결과에서 성능이 확연히 차이나는데에 대한 이유를 정확히는 알 수 없었다.

5. Conclusion

이 프로젝트에서는 Kaggle의 Home Credit - Credit Risk Model Stability 대회를 통해 XGBoost를 사용하여 대출 상환 가능성을 예측하는 모델을 개발하고 평가하였다. XGBoost는 그라디언트 부스팅 기법을 통해 강력한 성능을 발휘하며, 특히 정규화 기법을 통해 과적합을 방지하고 데이터의 해석 가능성을 제공하는 등의 장점을 가지고 있다.

모델 개발 과정에서는 데이터 전처리, 특성 엔지니어링, 모델 훈련 및 평가를 진행하였다. 데이터 전처리 단계에서는 데이터 형변환, 날짜 처리, 불필요한 열 제거, 메모리 최적화 등을 수행하여 데이터의 일관성과 효율성을 유지하였다. 특성 엔지니어링에서는 다양한 수치형, 날짜형, 문자형 데이터에서 유용한 특성을 추출하고 새로운 특성을 생성하여 모델의 성능을 향상시켰다.

모델은 StratifiedGroupKFold를 통해 교차 검증을 수행하였고, XGBoost의 파라미터를 최적화하여 각 fold에서 최적의 성능을 도출하였다. 특성 중요도 분석을 통해 모델이 예측에 중요한 역할을 하는 특성들을 확인하고, 필요 없는 특성들을 제거하여 모델의 복잡성을 줄이는 작업을 수행하였다.

최종적으로 Kaggle 대회에서 제출한 결과는 아쉽게도 상위 등수에 미치지 못하고 2645등을 기록하였다. 다른 상위 팀들의 코드를 분석한 결과, LightGBM을 사용한 것으로 확인되었으며, XGBoost와의 비교에서 테스트 세트와 검증 세트에서 비슷한 성능을 보였지만 실제 제출 결과에서는 성능 차이가 발생한 것으로 보인다. 이는 XGBoost와 LightGBM의 알고리즘적인 차이나 하이퍼 파라미터 설정의 차이 등이 원인일 수 있다.