기말고사 보고서
회사채 신용등급 변화 예측

<1조>

IT공학과 1812797 이정민

통계학과 1816793 장은조

통계학과 1816001 홍주리

2021.12

목 차

Ⅰ. 서론

1. 분석 프로젝트 개요

□ 개요

▷ 프로젝트명 : 회사채 신용등급 변화 예측

▷ 프로젝트 도구 : R studio

□ 회사채란?

회사채(corporate bond) 기업이 자금을 조달하는 방법 중 하나로 액면 금액, 만기, 금리 등이 명시되어 있는 회사가 발행하는 채권이다. 이러한 회사채는 주로 기업의 자금 조달의 목적으로 발행되지만 시장의 거래를 통하여 기업의 채무 상환 능력 및 신용 위험(credit risk) 등이 회사채 수익률로 반영되어 채무 기업의 현행 시장 이자율(market interest rate) 측정에도 활용된다. 회사채의 발행은 기업의 입장에서 자금 조달의 유용한 수단이며 기업 회계 기준에 의한 채권, 채무의 현재 가치 측정에 활용되는 이점을 지니나 제약 없는 회사채의 발행은 과도한 차입과 과잉 투자를 이끌 수 있으며 대규모의 기업 부도와 채권자의 손실 등을 야기하여 국가 경제에 심각한 영향을 줄 수 있다. 이에 따라 회사채 발행 적격에 대한 평가 기준이 요구 되었으며 1986년 3월 일반 회사채 및 전환사채 발행 적격기준에 기업의 신용을 평가하는 신용평가제도가 부분적으로 도입 되었다.

국내의 회사채 신용 등급(corporate bond rating)은 금융위원회에 허가를 받은 한국기업평가, NICE평가정보, 한국신용평가의 3개 신용 평가 기관에 의해 공시되고 있으며 각 신용 평가 기관의 내부 평가 기준에 따라 AAA, AA+, . . . , B+, B, B－, CCC, CC, CC, D의 22개 등급으로 표현된다. 여기서 +, 0,－ 표현은 AA~B의 등급에 적용되며 AAA등급부터 BBB등급까지를 투자등급으로 BB등급부터 D등급까지를 투기등급으로 분류하고 있다.

Ⅱ. 데이터 특성

1. 데이터 설명

주어진 데이터는 1998년부터 2020년까지 시간에 따라 관측된 여러 회사들의 데이터이다.
독립 변수는 Id, Y_c(현 시점의 신용등급), Y_b(이전 시점의 신용등급) 외 규모지표, 비재무 지표, 생산성 지표, 생산성 지표, 수익성 지표, 안정성 지표, 현금흐름 지표, 활동성 지표를 대표하는 59개의 독립변수들이 현시점, 이전 시점 두세트로 존재합니다. (자세한 독립변수 설명은 캐글 참고) 총 125개의 변수가 존재한다.

▷ 예측변수(Y) : 이전 대비 현재 신용등급 변화

(+1 : upgrade, 0 : no change, -1 : downgrade)

즉, Y가 이전 등급이 B0 였는데, 현재 등급이 B+가 됐다면, +1 이 될 것이다.

2. 분석 데이터셋

2-1. 기존의 변수들만 활용한 데이터셋 (original_dataset)
Id와 현 시점의 신용등급 변수 및 날짜 관련 변수들을 제외한 123개의 독립변수 존재
* 현 시점의 신용등급을 제외한 이유는 실제 Unseen data에는 현 시점의 신용등급 값이 존재하지 않을거기 때문에 독립변수에서 제외

2-2. 현시점과 이전시점의 변수들의 차이변수 59개를 추가한 데이터셋 (diff_dataset)
original_dataset은 현재와 과거 당시 시점의 값이 명시되어 있다. 이전 대비 현재는 얼만큼 변화했는지에 대한 정도를 새로운 변수로 추가해주었다.
즉 original_dataset 123개의 독립변수 + diff 변수 59개로 총 182개의 독립변수 사용

3. 모델 선정 과정

전체 데이터셋(trainset.csv)을 8:2로 train, valid로 분할한다.
* 이 때, train과 valid의 Y_label(C)의 비율이 동일하게 설정
train 데이터를 5-fold 교차검증을 통해 F1-score의 평균값이 가장 높은 파라미터를 선정
각 모델별 파라미터 튜닝과정을 거친 후, valid data에 적합시킨다. (그림1)
valid data에서 F1-score가 가장 높게 나온 모델을 최종 모델로 선정하게 된다.

▷ 평가 지표 : F1-score

[그림 1]

Ⅲ. 모델 선정 과정 및 결과

1. Multinomial Logistic Regression

Logistic Regression은 두 개의 값만을 가지는 종속변수와 독립변수들 간의 인과관계를 로지스틱 함수를 이용하여 추정하는 통계기법이다. 로지스틱 회귀분석은 어떤 사건(event)이 발생할지에 대한 직접 예측이 아니라 그 사건이 발생할 확률을 예측하는 것이다.
반응변수가 범주형 자료이며, 일반화 선형모형(generalized linear model)의 특수한 경우로 S형 곡선을 그리는 함수 모형이다. 로지스틱 회귀모형은 반응변수가 이힝변수일 경우 glm함수를 사용하지만, 다항변수일 경우에는 nnet패키지의 multinom함수를 사용한다. 다른 모델들에 비해 학습속도가 빠르며, 별도의 파라미터 튜닝과정이 없다는 장점이 있다.

< Original dataset 적합 결과 >

다항 로지스틱 회귀모형을 valid data에 적합시킨 결과, Accuracy는 0.7419가 나왔다. 하지만 train data 및 valid data의 Y가 불균형 데이터로 Accuray를 보는 것은 의미가 없다고 판단된다. 불균형 데이터 평가의 한계를 보완하고자 F1-score를 확인한 결과 0.463이 나왔다.
각 클래스별 민감도(Seneitivity)를 확인한 결과는 아래와 같다.
* 민감도 : 실제 Positive 중에서 Positive라고 예측한 비율

	Class : -1	Class : 0	Class : 1
Sensitivity	0.37838	0.8819	0.43000

[표 1]

소수 클래스인 -1,1에 대한 민감도가 각각 0.378, 0.43으로 class 0의 민감도 대비 낮은 값을 가지고 있다.

< diff dataset 적합 결과 >

다항 로지스틱 회귀모형 모델을 valid data에 적합시킨 결과, F1-Score 가 0.4874이 나왔다.
diff 변수 59개를 추가로 사용했을 때, F_score가 증가했지만 그 증가폭은 다소 작은 편이다.

2. 의사결정나무 C5.0

의사결정나무란 의사결정 rule을 계층적 나무구조로 도표화하여 분류 및 예측할 수 있는 분석방법입니다. 의사결정나무의 종류는 다양한데, 그 중 C5.0은 ID3의 설명 향상을 위해 개발된 알고리즘인 C4.5의 상향버전의 알고리즘으로 엔트로피를 낮추는 방향으로 가지치기를 진행합니다.
C5.0은 가장 중요한 속성만 사용하며, 모든 문제에 적합한 분류기라는 장점이 있다.

▷ 하이퍼 파라미터 튜닝 후보값
trials의 값은 (1,5,10,20,30), rules값은 (T,F)를 후보값으로 정해서 총 10가지의 조합을 생성
* trials : 부스팅 반복수를 의미하며, trails = 1일때는 의사결정나무에 속하지만 trials가 2이상의 값을 가지게 되면 앙상블 기법 중 부스팅에 속하게 되는 것으로 판단.

* rules : 나무가 rule-based model로 분해되는지 여부

< Original dataset 적합 결과 >

C5.0 모델을 valid data에 적합시킨 결과, F1-Score 가 0.389가 나왔다.
다항 로지스틱 회귀모형의 F1-score 0.463보다는 현저히 예측력이 떨어졌다.
각 클래스별 민감도(Seneitivity)를 확인한 결과는 아래와 같다.

	Class : -1	Class : 0	Class : 1
Sensitivity	0.29730	0.8940	0.2900

[표 2]

소수 클래스인 -1,1에 대한 민감도가 각각 0.2973, 0.29으로 class 0의 민감도 대비 낮은 값을 가지고 있다. 다항 로지스틱 회귀모형의 -1,1에 대한 민감도 0.378, 0.43 보다 작은 값을 가지는 것을 볼 수 있고, 이를 통해 다항 로지스틱 회귀모형이 C5.0 모형보다 소수 클래스에 대한 예측력이 높다는 것을 알 수 있다.

< diff dataset 적합 결과 >

C5.0 모델을 valid data에 적합시킨 결과, F1-Score 가 0.5217이 나왔다.
diff 변수 59개를 추가로 사용했을 때, F_score가 0.1557이 증가하였고 예측력의 향상이 유의적인 증가폭이다.

3. eXtrem Gradient Boosting

Gradient descent boosting 기법에 병렬 학습이 지원되도록 구현한 라이브러리이다.
GBM 대비 수행시간이 빠르며, 표준 GBM 경우 과적합 규제기능이 없으나, XGBoost는 자체적으로 과적합 규제 기능으로 강한 내구성을 지닌다. 또한 Early stopping 기능이 있어 더 이상 성능의 발전이 없을 경우 멈춰추는 기능도 있다.

▷ 튜닝 파라미터 및 선정 이유
max_depth는 클수록 training set의 error는 줄일 수 있으나, 과적합을 일으킬 수 있다.
colsample_bytree 는 낮을수록 과적합을 방지해준다.
eta는 너무 크면 overshooting & 과적합 문제 발생하며, 너무 작으면 학습 시간이 오래 걸린다.
gamma는 분할을 수행하는데 필요한 최소 손실 감소를 지정해준다.
과적합의 이슈를 해결하기 위해 4개의 파라미터를 선정하였다.

▷ 파라미터 튜닝 후보값
max_depth의 값은 (3,5,7), colsample_bytree의 값은 (0.7,1), eta의 값은 (0.1,0.05,0.01), gamma의 값은 (0,1,2) 를 후보값으로 정해서 총 54가지 파라미터 조합을 생성시켰다.

< Original dataset 적합 결과 >

XGB 모델을 valid data에 적합시킨 결과, F1-Score 가 0.366이 나왔다.

다항 로지스틱 회귀모형의 F1-score 0.463보다는 현저히 예측력이 떨어졌다.
각 클래스별 민감도(Seneitivity)를 확인한 결과는 아래와 같다.

	Class : -1	Class : 0	Class : 1
Sensitivity	0.30233	0.9319	0.18667

[표 3]

소수 클래스인 -1,1에 대한 민감도가 각각 0.30233, 0.18667로 class 0의 민감도 대비 낮은 값을 가지고 있다. 0에 대한 민감도는 가장 높게 나왔다.

< diff dataset 적합 결과 >

XGB 모델을 valid data에 적합시킨 결과, F1-Score 가 0.3697이 나왔다.
diff 변수 59개를 추가로 사용했을 때, F_score가 증가했지만 그 증가폭은 다소 작은 편이다.

4. Stacking_weight

스태킹이란 여러 모델들을 활용해 각각의 예측 결과를 도출한 뒤 그 예측 결과를 결합해 최종 예측 결과를 만들어 내는 알고리즘으로 n개의 모델을 학습 데이터를 활용하여 학습 모델을 생성한 뒤 학습을 마치면 예측한 값들을 합쳐서 최종적으로 예측하는 방법이다. 여러 모델을 활용하기 때문에 단일 모델을 사용했을 때 보다 성능이 향상되고 편향이 줄어드는 장점이 있다.

다항 로지스틱 회귀모형, C5,0, XGB 3개의 모델을 predict 할 때 type 옵션을 prob로 설정해 각 클래스에 해당되는 확률값을 output으로 구한다.
3개의 모델의 각 클래스별 확률값의 평균을 구할 것이다. 이 때 모두 더해서 3으로 나누는 단순평균이 아닌 각 모델별&클래스별 가졌던 민감도 (표1~표3)를 가중치로 하여 평균을 내줄 것이다.
즉, 다항 로지스틱 회귀모형이 class -1,1의 민감도가 높으니 -1,1에 대한 예측 확률값은 로지스틱의 값에 가중치가 더 많이 부여될 것이다. 이 때, 클래스 불균형으로 0에 대한 민감도가 다른 클래스 민감도보다 압도적으로 높은 값을 가지기 때문에 이를 보정하고자 클래스별 합계로 나눠준다.

각 클래스별 가중평균을 구한 후, 가장 높은 확률의 값을 가지는 class를 최종 예측값으로 output.

< Original dataset 적합 결과 >

가중평균 Stacking을 valid data에 적합시킨 결과, F-score가 0.427이 나왔다.

	Class : -1	Class : 0	Class : 1
Sensitivity	0.33784	0.8819	0.4

[표 4]

소수 클래스인 -1,1에 대한 민감도가 각각 0.378, 0.43으로 class 0의 민감도 대비 낮은 값을 가지고 있다.

< diff dataset 적합 결과 >

Stacking_weight 모델을 valid data에 적합시킨 결과, F1-Score 가 0.4878이 나왔다.
diff 변수 59개를 추가로 사용했을 때, F_score가 증가했지만 그 증가폭은 다소 작은 편이다.

5. Stacking_majority vote

앞서 Stacking_weight는 예측값을 class로 반환하지 않고, prob로 반환하여 각 클래스별 설정한 가중치를 통해 가중평균으로 최종 예측을 했다. Stacking_majority_vote는 각 모델의 예측값을 class로 반환한 후, 최빈값을 사용하여 최종 예측값으로 output 했다. 즉, 3개의 모델의 예측값에 다수결의 법칙이 적용된 결과이다.

< Original dataset 적합 결과 >

다수결의 법칙 Stacking을 valid에 적합시킨 결과, F1-score를 확인한 결과 0.46246이 나왔다.

< diff dataset 적합 결과 >

다수결의 법칙 Stacking을 valid data에 적합시킨 결과, F1-Score 가 0.4839이 나왔다.
diff 변수 59개를 추가로 사용했을 때, F1_score가 증가했지만 그 증가폭은 다소 작은 편이다.

6. 최종 모델 선정

최종 모델 선정의 지표는 valid의 F1-score값이다.
(original_dataset & diff_dataset) + 5개의 모델 조합 중 valid의 F1-score가 가장 큰 값을 가지는 조합은 diff_dataset의 C5.0으로 이를 최종 testset.csv에 적합하여 캐글에 제출하였다.

Ⅳ. 분석 결과 요약

Valid 기준	기존 F1-score	차이 반영 F1-score
C5.0	0.389	0.5217 ▲
XGBoost	0.366	0.3697 ▲
Logistic Regression	0.463	0.4874 ▲
Stacking	0.427	0.4878 ▲
Stacking	-	0.4839

우리가 가진 trainset.csv를 train과 valid 데이터셋으로 쪼개서, 구했던 F1-score이다.
5개의 모형 모두 공통점으로 Original dataset에 적용한 것 보다 diff dataset에 적용한 결과가 좀 더 예측력이 좋게 나왔다. 이는 diff 변수가 Y를 예측하는데 있어서 영향력을 주었다고 생각한다.

	Public F1-score	Private F1-score
C5.0	0.50628	0.44640
XGBoost	0.45918	0.36324
Logistic Regression	0.51718	0.499979
Stacking(가중편균)	0.52449	0.45529
Stacking(다수결법칙)	0.54217	0.46246

위의 표는 [original dataset]로 학습시킨 각 모델을 Y_label이 없는 testset.csv에 적합시킨 후, 캐글에 올린 결과이다.
Public F1-score는 testset.csv의 80%에 대한 F1-score이며, Private F1-score는 나머지 20%에 대한 F1-score이다. 80%에 대한 F-score를 확인했을 때, 상위2개의 모델이 Stacking 이 선정됐다.
전반적으로 Public 대비 Private의 F1-score가 낮게 평가가 되고 있는데, 이는 20%의 데이터로만 측정된 결과이다 보니 차이가 난다고 생각한다.

	Public F1-score	Private F1-score
C5.0	0.50297	0.42976
XGBoost	0.44749	0.36342
Logistic Regression	0.52273	0.46130
Stacking(가중편균)	0.49722	0.40643
Stacking(다수결법칙)	0.51961	0.46386

위의 표는 [diff dataset]로 학습시킨 각 모델을 Y_label이 없는 testset.csv에 적합시킨 후, 캐글에 올린 결과이다.
Public F1-score 상위 모델은 Stacking과 로지스틱 회귀모형이 선정됐다.
상위의 개념없이 보면 5개의 모델 중 다항 로지스틱 회귀모형과 두가지의 Stacking 모형이 valid F1-score, Public F1-score, Private F1-score에 상대적으로 높게 나오는 경향이 있었다.
최종 모델로 선정된 C5.0 모델의 경우 valid의 F1-Score는 0.5점대로 높았지만, 최종 Private F1-score는 0.42976으로 다소 차이가 생겼고 이는 과적합의 원인과 Private의 20% 비율로 인한 결과라고 판단된다. XGBoost의 경우 모든 경우에서 F1-score가 상대적으로 낮게 나왔다. 이에 대한 이유로는 과적합을 조정하기 위한 파라미터들을 주로 사용하다보니 오히려 Underfitting이 되지 않았나 판단이 된다.

Ⅴ. 아쉬운 점

▷ Stacking 모델에서 3개의 모델이 아닌 좀 더 많은 모델들을 사용해서 합쳤다면, 좀 더 편향 편향을 없애고 성능을 좀 더 올릴 수 있었을거라 생각한다.

▷ 변수선택법을 통해 변수 선정과정을 거쳤다면 예측력이 향상될거라고 생각한다.

▷ trainset.csv를 train과 valid로 잘라, train데이터로만 학습을 시켰기 때문에 정보를 모두 활용하지 못했다.

▷ Y__b 이전 시점의 신용등급의 범주가 너무 많아서 이를 좀 더 큰 카테고리로 묶었다면 좀 더 좋은 결과가 나왔을 거라 생각한다.