

회사채 신용등급 변화 예측

IT 공학과 1812797 이정민
통계학과 1816793 장은조
통계학과 1816001 홍주리

Summary

Data

Y : 신용등급 변화

▷ -1(감소), 0(변화 없음), 1(증가) 로 구성된 범주형 변수

X

▷ X1~ X60, BX1~BX60 , Y_b

총 119개의 변수

1998년 ~ 2016년까지의 데이터를
train 데이터로 활용.

Model

1. C5.0
2. Xgboost
3. Logistic Regression
4. Stacking

Train

1. 기존 데이터의 변수 사용
2. 주어진 변수를 활용하여 새로운 변수 값을 활용

예측 방법

trainset.csv

testset.csv

- ✓ 각 모델의 최종 성능 비교
- ✓ C의 비율 비슷하게 8:2 분할

train

valid



train data 의 5-fold cv를 통해 F-score의 평균값이 가장 높은 파라미터를 선정

예측방법

- ✓ train data 의 5-fold cv를 1번 진행
- ✓ 각 모델별 튜닝 하이퍼 파라미터 선정
- ✓ train으로 학습 후 validation으로 적합해 F_score 확인

```
K = 5
R = 1

set.seed(123)
cv = cvFolds(NROW(train.data), K = K , R = R)

grid <- expand.grid(trials = c(1,5,10,20,30),
                   rules = c(T,F))

result <- foreach(g=1:NROW(grid), .combine=rbind) %do% {
  foreach(r=1:R, .combine=rbind) %do% {
    foreach(k=1:K, .combine=rbind) %do% {
      validation_idx <- cv$subsets[which(cv$which == k), r]
      train <- train.data[-validation_idx, ]
      validation <- train.data[validation_idx, ]

      # 모델 훈련
      m <- C5.0(C ~., data=train ,control=C5.0Control(winnow = F),
               trials=grid[g, "trials"],
               rules=grid[g, "rules"])

      # 예측
      predicted <- predict(m, newdata=validation)

      # 성능 평가
      F_score <- F1_Score(factor(predicted), factor(validation$C))
      return(data.frame(g=g, F_score=F_score))
    }
  }
}
```

예측 방법

```
> result
  g  F_score
1  1 0.3200000
2  1 0.3902439
3  1 0.2285714
4  1 0.2424242
5  1 0.1904761
6  2 0.2800000
7  2 0.3720930
8  2 0.2285714
9  2 0.2424242
10 2 0.2380952
11 3 0.2978723
12 3 0.4000000
13 3 0.2222222
14 3 0.2580645
15 3 0.1395348
16 4 0.2800000
17 4 0.2926829
18 4 0.2777777
19 4 0.3333333
20 4 0.1428571
21 5 0.3265306
22 5 0.3720930
```

```
246 50 0.31372549
247 50 0.21621622
248 50 0.18750000
249 50 0.21428571
250 50 0.10000000
251 51 0.33962264
252 51 0.18604651
253 51 0.23529412
254 51 0.25000000
255 51 0.18604651
256 52 0.32727273
257 52 0.30434783
258 52 0.21621622
259 52 0.24242424
260 52 0.18181818
261 53 0.33333333
262 53 0.31818182
263 53 0.22857143
264 53 0.31250000
265 53 0.18604651
266 54 0.35714286
267 54 0.24489796
268 54 0.15789474
269 54 0.35897436
270 54 0.26666667
```

```
g mean_Fscore
1  5  0.3676747
2  4  0.3596000
3  3  0.3391408
4  1  0.3228629
5  2  0.3226286
6  6  0.3036300
7 10  0.2881754
8  9  0.2824717
9  8  0.2605685
10 7  0.2580884
```

- ✓ g : 한 파라미터의 cv = 5를 의미
- ✓ g로 그룹화 해서 F_score 평균
- ✓ mean_Fscore가 가장 큰 g를 선택

해당 g 번째에 해당되는 파라미터 조합 **최종 선택**
최종 선택 파라미터로 **train 데이터 학습** 후
valid에 적합하여 각 모델별 성능을 확인

1. Decision Tree

- 의사결정 나무 알고리즘 중 하나인 **C5.0**을 활용하였으며 분류문제에 많이 활용된다.
- ID3의 설명 향상을 위해 개발된 알고리즘인 C4.5의 상향버전의 알고리즘

장점

- 모든 문제에 적합하게 사용할 수 있는 분류기
- 결측치, 명목형 변수, 수치 등을 처리할 수 있는 자동성이 높은 모델로 사용에 용이
- 다른 모델에 비해 비교적 높은 효율을 보임

1. Decision Tree

C5.0 parameter

```
C5.0(  
  x,  
  y,  
  trials = 1,  
  rules = FALSE,  
  weights = NULL,  
  control = C5.0Control(),  
  costs = NULL,  
  ...  
)
```

하이퍼 파라미터 튜닝

- trials = 1, 5, 10, 20, 30
- rules = True, False

trials	an integer specifying the number of boosting iterations. A value of one indicates that a single model is used.
rules	A logical: should the tree be decomposed into a rule-based model?

1. Decision Tree

Confusion Matrix and Statistics

		Reference		
Prediction		-1	0	1
-1		22	16	1
0		50	371	70
1		2	28	29

Overall Statistics

Accuracy : 0.7165
95% CI : (0.6782, 0.7526)
No Information Rate : 0.7046
P-Value [Acc > NIR] : 0.28

Kappa : 0.268

McNemar's Test P-Value : 8.062e-08

Statistics by Class:

	Class: -1	Class: 0	Class: 1
Sensitivity	0.29730	0.8940	0.29000
Specificity	0.96699	0.3103	0.93865
Pos Pred Value	0.56410	0.7556	0.49153
Neg Pred Value	0.90545	0.5510	0.86604
Prevalence	0.12564	0.7046	0.16978
Detection Rate	0.03735	0.6299	0.04924
Detection Prevalence	0.06621	0.8336	0.10017
Balanced Accuracy	0.63214	0.6022	0.61433

```
> F1_Score(factor(pred_valid_class_c5), factor(valid.data$C))  
[1] 0.3893805
```

C5.0 의 정확도는 0.7165이며 valid data에 적합한 결과 **F-score가 0.389**가 나왔다.

1. Decision Tree

Confusion Matrix and Statistics

Reference

Statistics by Class:

Submission and Description	Private Score	Public Score	Use for Final Score
sample_submission_c5_tune.csv just now by Eunjojang add submission details	0.44640	0.50628	<input type="checkbox"/>

95% CI : (0.6782, 0.7526)

No Information Rate : 0.7046

P-Value [Acc > NIR] : 0.28

Kappa : 0.268

McNemar's Test P-Value : 8.062e-08

Decision Prevalence : 0.68821 0.68558 0.68617
Balanced Accuracy : 0.63214 0.6022 0.61433

```
> F1_Score(factor(pred_valid_class_c5), factor(valid.data$C))  
[1] 0.3893805
```

Public Score ≒ 0.50628, Private Score ≒ 0.44640

2. XGBoost

여러 개의 의사결정 트리를 조합하여 사용하는 Ensemble 기법 중 하나로 약한 예측 모형들의 학습 에러에 가중치를 두고, 순차적으로 다음 학습 모델에 반영하여 강한 예측 모형을 만듦.

장점

- 병렬 처리로 학습 및 분류를 수행하기 때문에 비교적 수행시간이 빠름
- 과적합 규제 기능이 있음
- 다양한 옵션을 통해 커스터마이징이 가능

2. xgboost

parameter

```
( eta = 0.3,  
  gamma = 0,  
  max_depth = 6,  
  min_child_weight = 1,  
  max_delta_step = 0,  
  subsample = 1,  
  sampling_method = uniform,  
  lamda = 1,  
  alpha = 1  
  .  
)
```

[하이퍼 파라미터 튜닝]

- colsample_bytree = 0.7, 1
- max_depth = 3, 5, 7
- eta = 0.1, 0.05, 0.01
- gamma = 0, 1, 2

예측력과 과적합 방지를 위한 하이퍼 파라미터 조합

2. xgboost

Confusion Matrix and Statistics

	Reference		
Prediction	0	1	2
0	13	13	2
1	29	301	59
2	1	9	14

Overall Statistics

Accuracy : 0.7438
95% CI : (0.7003, 0.7839)
No Information Rate : 0.7324
P-Value [Acc > NIR] : 0.3165

Kappa : 0.243

McNemar's Test P-Value : 2.239e-09

Statistics by Class:

	Class: 0	Class: 1	Class: 2
Sensitivity	0.30233	0.9319	0.18667
Specificity	0.96231	0.2542	0.97268
Pos Pred Value	0.46429	0.7738	0.58333
Neg Pred Value	0.92736	0.5769	0.85372
Prevalence	0.09751	0.7324	0.17007
Detection Rate	0.02948	0.6825	0.03175
Detection Prevalence	0.06349	0.8821	0.05442
Balanced Accuracy	0.63232	0.5931	0.57967

```
> F1_Score(factor(pred_valid_class), factor(valid_label))  
[1] 0.3661972
```

Xgboot 의 정확도는 0.7438이며 valid data에 적합한 결과 **F-score가 0.366**이 나왔다.

2. xgboost

Confusion Matrix and Statistics

Reference

Statistics by Class:

Submission and Description	Private Score	Public Score	Use for Final Score
sample_submission_xgb_tune.csv a few seconds ago by Eunjojang add submission details	0.36324	0.45918	<input type="checkbox"/>

No Information Rate : 0.7324

P-Value [Acc > NIR] : 0.3165

Kappa : 0.243

Mcnemar's Test P-Value : 2.239e-09

```
> F1_Score(factor(pred_valid_class), factor(valid_label))  
[1] 0.3661972
```

Public Score ≒ 0.45918, Private Score ≒ 0.36324

3. Logistic Regression

회귀를 사용하여 데이터가 어떤 범주에 속할 확률을 0에서 1사이의 값으로 예측하고
그 확률에 따라 가능성이 더 높은 범주에 속하는 것으로 분류해주는 지도 학습 알고리즘.

장점

- 이진 분류 뿐만 아니라 다중 클래스 분류로 확장이 가능.
- 간단하고 쉬운 방법.

3. Logistic Regression

parameter

```
multinom(formula, data, weights, subset, na.action,  
          contrasts = NULL, Hess = FALSE, summ = 0, censored = FALSE,  
          model = FALSE, ...)
```

```
multinom(C ~., data = train_data)
```

- ✓ 로지스틱 회귀분석은 별도의 하이퍼 파라미터가 없어서 튜닝 과정 생략
- ✓ 초기에 나눈 train data로 학습 후 바로 valid data에 적합

3. Logistic Regression

```
> confusionMatrix(factor(pred_valid_class_logit), factor(valid.data$C))
```

Confusion Matrix and Statistics

	Reference		
Prediction	-1	0	1
-1	28	18	1
0	45	366	56
1	1	31	43

Overall Statistics

Accuracy : 0.7419
95% CI : (0.7046, 0.7768)
No Information Rate : 0.7046
P-Value [Acc > NIR] : 0.0249800

Kappa : 0.3701

McNemar's Test P-Value : 0.0003072

Statistics by Class:

	Class: -1	Class: 0	Class: 1
Sensitivity	0.37838	0.8819	0.43000
Specificity	0.96311	0.4195	0.93456
Pos Pred Value	0.59574	0.7837	0.57333
Neg Pred Value	0.91513	0.5984	0.88911
Prevalence	0.12564	0.7046	0.16978
Detection Rate	0.04754	0.6214	0.07301
Detection Prevalence	0.07980	0.7929	0.12733
Balanced Accuracy	0.67074	0.6507	0.68228

```
> |
```

```
> F1_Score(factor(pred_valid_class_logit), factor(valid.data$C))  
[1] 0.4628099
```

로지스틱 회귀의 정확도는 0.7419이며 valid data에 적합한 결과 **F-score가 0.463**이 나왔다.

3. Logistic Regression

```
> confusionMatrix(factor(pred_valid_class_logit), factor(valid.data$C))
```

Confusion Matrix and Statistics

	Reference		
Prediction	-1	0	1

Statistics by Class:

Submission and Description

Private Score

Public Score

Use for Final Score

[sample_submission_multinom_.csv](#)

0.49979

0.51718

☐

just now by [Eunjojang](#)

[add submission details](#)

```
No Information Rate : 0.7046
P-Value [Acc > NIR] : 0.0249800
```

```
Kappa : 0.3701
```

```
McNemar's Test P-Value : 0.0003072
```

```
Balanced Accuracy      0.67074      0.6507      0.68228
```

```
> F1_Score(factor(pred_valid_class_logit), factor(valid.data$C))
[1] 0.4628099
```

Public Score ≒ 0.51718, Private Score ≒ 0.49979

4. Stacking

여러 모델들을 활용해 각각의 예측 결과를 도출한 뒤 그 예측 결과를 결합해 최종 예측 결과를 만들어 내는 알고리즘으로 n 개의 모델을 학습 데이터를 활용하여 학습 모델을 생성한 뒤 학습을 마치면 예측한 값들을 합쳐서 최종적으로 예측하는 방법이다.

장점

- 여러 모델을 활용하기 때문에 단일 모델을 사용했을 때 보다 성능이 향상.

4. Stacking(1)

Statistics by Class:

	Class: 0	Class: 1	Class: 2
Sensitivity	0.30233	0.9319	0.18667
Specificity	0.96231	0.2542	0.97268
Pos Pred Value	0.46429	0.7738	0.58333
Neg Pred Value	0.92736	0.5769	0.85372

Statistics by Class:

	Class: -1	Class: 0	Class: 1
Sensitivity	0.37838	0.8819	0.43000
Specificity	0.96311	0.4195	0.93456
Pos Pred Value	0.59574	0.7837	0.57333
Neg Pred Value	0.91513	0.5984	0.88911
Prevalence	0.12564	0.7046	0.16978

Statistics by Class:

	Class: -1	Class: 0	Class: 1
Sensitivity	0.29730	0.8940	0.29000
Specificity	0.96699	0.3103	0.93865
Pos Pred Value	0.56410	0.7556	0.49153
Neg Pred Value	0.90545	0.5510	0.86604
Prevalence	0.12564	0.7046	0.16978



```
> weight_df
      minus1  zero  plus1
xgboost 0.30233 0.9319 0.18667
logis    0.37838 0.8819 0.43000
C5       0.29730 0.8940 0.29000
```

- ✓ 각 모델별(C5.0/XGB/로지스틱) 각 C의 범주별 민감도를 가져와 weight_df 에 담는다.
- ✓ 로지스틱 회귀모델이 소수 클래스 -1,1에 대한 예측도가 다른 모델들과 비교했을 때 높은 편

4. Stacking(1)

```
> weight_df
      minus1  zero  plus1
xgboost 0.30233 0.9319 0.18667
logis    0.37838 0.8819 0.43000
C5       0.29730 0.8940 0.29000
```



```
> weight_df
      minus1  zero  plus1 minus1_ratio zero_ratio plus1_ratio
xgboost 0.30233 0.9319 0.18667      0.31      0.34      0.21
logis    0.37838 0.8819 0.43000      0.39      0.33      0.47
C5       0.29730 0.8940 0.29000      0.30      0.33      0.32
```

- ✓ 각 열의 스케일을 맞추기 위해 각 열의 합계로 나눠준다.

4. Stacking(1)

```
> weight_df
      minus1  zero  plus1
xgboost 0.30233 0.9319 0.18667
logis    0.37838 0.8819 0.43000
C5       0.29730 0.8940 0.29000
```

	minus1	zero	plus1	minus1_ratio	zero_ratio	plus1_ratio
xgboost	0.30233	0.9319	0.18667	0.31	0.34	0.21
logis	0.37838	0.8819	0.43000	0.39	0.33	0.47
C5	0.29730	0.8940	0.29000	0.30	0.33	0.32



```
> result_df <- pred_valid_prob_logit
> head(result_df)
      minus1  zero  plus1
3  0.4165267 0.4583849 0.08257237
15 0.2691080 0.6623386 0.07625331
21 0.1719698 0.5518016 0.17175123
22 0.2119291 0.7011236 0.07845508
28 0.3040678 0.4768663 0.16414673
44 0.2698467 0.4105471 0.25561197
```

- ✓ predict(model, valid, type = 'prob') 를 통해 각 범주별(-1/0/1) 확률값을 구함
- ✓ xgboost, 로지스틱, C5.0 각 범주별 확률값에 가중치를 곱해서 더해준다. (단순평균 대신 가중평균 사용)
- ✓ 높은 확률의 범주를 최종 class로 예측

4. Stacking(1)

```
> valid.data$C <- as.factor(valid.data$C)
> confusionMatrix(factor(pred_list), factor(valid.data$C))
Confusion Matrix and Statistics
```

	Reference			
Prediction	-1	0	1	
-1	25	17	1	
0	47	366	59	
1	2	32	40	

Overall Statistics

Accuracy : 0.7317
95% CI : (0.694, 0.7671)
No Information Rate : 0.7046
P-Value [Acc > NIR] : 0.07983

Kappa : 0.3374

McNemar's Test P-Value : 5.368e-05

Statistics by Class:

	Class: -1	Class: 0	Class: 1
Sensitivity	0.33784	0.8819	0.40000
Specificity	0.96505	0.3908	0.93047
Pos Pred Value	0.58140	0.7754	0.54054
Neg Pred Value	0.91026	0.5812	0.88350
Prevalence	0.12564	0.7046	0.16978
Detection Rate	0.04244	0.6214	0.06791
Detection Prevalence	0.07301	0.8014	0.12564
Balanced Accuracy	0.65144	0.6364	0.66524

```
> F1_Score(factor(pred_list), factor(valid.data$C))
[1] 0.4273504
```

Stacking(가중평균)의 정확도는 0.7317이며 valid data에 적합한 결과 **F-score가 0.427**이 나왔다.

4. Stacking

```
confusionMatrix(factor(pred_list), factor(valid.data$C))
```

Confusion Matrix and Statistics

Statistics by Class:

Class: -1 Class: 0 Class: 1

Submission and Description

Private Score

Public Score

Use for Final Score

[sample_submission_stacking.csv](#)

0.45529

0.52449

☐

just now by [Eunjojang](#)

[add submission details](#)

P-Value [Acc > NIR] : 0.07983

Kappa : 0.3374

McNemar's Test P-Value : 5.368e-05

```
> F1_Score(factor(pred_list), factor(valid.data$C))
```

[1] 0.4273504

Public Score는 0.52449, Private Score는 0.45529

4. Stacking(2)

Submission and Description	Private Score	Public Score	Use for Final Score
sample_submission_stacking__.csv just now by Eunjojang add submission details	0.46246	0.54217	<input type="checkbox"/>

- ✓ `predict(model, valid, type = 'class')` 를 통해 각 모델별 예측값을 구한다.
- ✓ 3개의 모델에서 나온 예측값 중 최빈값을 최종 예측값으로 선정 (투표, 다수결의 법칙 적용)

Public Score는 0.54217, Private Score는 0.46246

최종결과

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

변수 추가

각 변수의 현재 값과 과거 값의 차이를 59개의 추가적인 변수로 사용하여 동일하게 학습

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

최종결과

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

기존 변수들만 사용 ▲

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

추가 변수들 사용

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

아쉬운점

- ✓ 변수 선택, 파생 변수, 변수 변환 등 변수들의 의미 부족으로 다양하게 활용하지 못한 점
 - ⇒ 중요도가 낮은 변수 제거 및 도메인 지식에 의한 변수 변환 등의 방법을 결합하면 성능이 더 좋아질 것으로 기대
- ✓ C5.0 , Xgboost 튜닝 과정에서 다양한 파라미터와 후보값들을 고려하지 못한 점
 - ⇒ 후보값들을 다양하게 고려했다면 높은 예측력과 과적합을 막을 수 있었을 것으로 기대
- ✓ 다양한 모델을 적용해보지 못한 점
 - ⇒ 스택킹_다수결의법칙을 적용할 때 좀 더 많은 모델의 의견이 있었다면 다수결의 신뢰성이 높아졌을 것으로 기대