Multivariate Data Analysis Assignment #4

Ensemble Learning

산업경영공학부 2020170856 이우진

Dataset : **Richter's Predictor: Modeling Earthquake Damage**

해당 데이터셋은 총 260,601개의 건물에 대한 지진 피해 정도를 기록한 데이터셋이다. 첫 번째 컬럼인 building_id는 각 건물의 일련번호이며, 마지막 컬럼인 damage_grade는 해당 건물의 지진 피해 정도로써 분류모형의 종속변수로 사용된다. 이 외 모든 컬럼은 입력변수이다.

https://www.kaggle.com/datasets/mullerismail/richters-predictor-modeling-earthquake-damage

[사전 작업 사항]

[1] 입력 변수의 속성이 numeric 이 아닌 변수들에 대해 1-of-C coding (1-hot encoding) 방식을 통해 명목형(요인형) 변수를 범주의 개수만큼의 이진형(binary) 변수들로 구성되는 dummy variable을 생성하시오.

해당 데이터셋은 260,601개의 행과 40개의 변수를 가지고 있다. 이때, 마지막 칼럼인 damage_grade는 종속변수로 사용되고, building_id를 제외한 나머지 38개의 칼럼은 입력변수로 사용된다. 'building_id' 변수는 학습에 사용하는 변수가 아니므로 이를 삭제한 후, 학습을 진행하였다. 결측치 확인도 진행해보았는데, 결측치가 존재하지 않는 것을 알 수 있었다. 다음으로 pd.get_dummies 함수로 numeric 형식이 아닌 변수들에 대하여 1-of-C coding을 진행하여 dummy variable을 생성하였다. 최종적으로 68개의 입력변수와 1개의 종속변수가 만들어졌다.

[2] 전체 데이터셋에서 10,000개의 instance를 샘플링한 뒤 학습/검증/테스트 데이터셋을 다시 6,000개/2,000개/2,000개로 분할하고 다음 각 물음에 답하시오. 분류 성능을 평가/비교할 때는 3-class classification의 Accuracy와 Balanced Correction Rate (BCR)을 이용하시오.

전체 데이터셋에서 무작위로 10,000개의 instance를 샘플링 하였고, 학습/검증/테스트 데이터셋을 다시 6000개/2000개/2000개로 분할하였다. 이때, 샘플링 이전의 종속변수의 값을 확인해보았을 때, class 1의 개수가 25124, class 2의 개수가 178259, class 3의 개수가 87218개로 불균형 상태인 것을 알 수 있다. 샘플링 이후에도 이러한 비율이 유지되는지 확인해 보았고, 그 결과 class 1의 개수가 990, class 2의 개수가 5743, class 3의 개수가 3268개인 것을 알 수 있었다. 즉, 샘플링 이후에도 각각의 클래스에 대한 비율이 잘 유지된 것을 알 수 있다.

train, valid, test셋으로 나눈 후에도, 각각의 클래스 비율이 유지되는지 확인한 결과 아래와 같이 원래 데이터셋의 클래스 비율과 비슷하게 잘 분할된 것을 확인할 수 있다.

```
damage_grade
2    3430
3    1973
1     597
Name: count, dtype: int64
damage_grade
2    1144
3     649
1     207
Name: count, dtype: int64
damage_grade
2    1168
3     646
1     186
Name: count, dtype: int64
```

다음으로 분류 성능 평가/비교할 때, Accuracy와 BCR을 평가 지표로 사용하기 위하여 아래와 같은 코드를 작성하였다.

```python
def perf_eval(cm: np.array):
    num_classes = cm.shape[0]
    tpr = np.zeros(num_classes)
    tnr = np.zeros(num_classes)

    for i in range(num_classes):
        tp = cm[i, i]
        fn = np.sum(cm[i, :]) - tp
        fp = np.sum(cm[:, i]) - tp
        tn = np.sum(cm) - (tp + fn + fp)

        tpr[i] = tp / (tp + fn) if (tp + fn) > 0 else 0
        tnr[i] = tn / (tn + fp) if (tn + fp) > 0 else 0

    # 전체 정확도 계산
    overall_acc = np.trace(cm) / np.sum(cm)
    # 평균 BCR 계산
    avg_bcr = np.mean(np.sqrt(tpr * tnr))

    return overall_acc, avg_bcr
```

특히, 이번 데이터셋의 경우, 위에서 말한 것과 같이 데이터가 상당히 불균형한 것을 알 수 있다. class 1과 class3의 개수를 더하여도 class2의 개수보다 작은 것을 알 수 있고, 이는 학습을 진행하지 않고 모두 2로만 예측하여도 Accuracy가 50프로가 넘는다고 할 수 있다. 본 과제에서는 class1, class3도 잘 예측하는 것을 목표로 하기에, Accuracy 대신, 각 class들의 예측 결과를 모두 종합적으로 고려할 수 있는 BCR을 기준으로, 여러 모델들과 하이퍼파라미터 조합에 대해 평가하고자 한다.

[Q1] 다음과 같이 세 가지 단일 모형에 대하여 분류 모델을 구축하고 Accuracy 와 BCR 관점에서 분류 정확도를 비교해보시오. CART 와 ANN 의 경우 hyperparameter 후보 값들을 명시하고 Validation dataset을 통해서 최적의 값을 찾아서 Test 에 사용하시오.

1. Multinomial logistic regression (MLR)

먼저, MLR 학습을 진행하기 전, Multinomial logistic regression도 최적의 결과를 얻기 위해 여러 하이퍼파라미터 조합 후보를 정하였고, 그 하이퍼파라미터는 아래와 같다.

1. 'C': [0.01, 0.1, 1, 10, 100]

    Regularization 강도의 역수를 나타내는 파라미터로, C 값이 작을수록 규제가 강해지고, C 값이 클수록 규제가 약해진다.

2. 'solver': ['newton-cg', 'lbfgs', 'sag']

    최적화 알고리즘을 나타내는 파라미터로, newton-cg는 Newton-Conjugate Gradient 방법을 의미한다. lbfgs는 Broyden-Fletcher-Goldfarb-Shanno (BFGS) 알고리즘의 제한된 버전이다. sag는 Stochastic Average Gradient 방법이다.

3. 'max_iter': [100, 300]

    반복 횟수와 관련된 파라미터로, solver가 수렴할 때까지 허용하는 최대 반복 횟수를 설정한다.

해당 파라미터 조합으로 Grid search를 수행하고, 검증 데이터에 대한 BCR 값을 기준으로 최적의 하이퍼파라미터 조합을 찾아보았다. 이때, BCR을 기준으로 최적의 하이퍼파라미터를 찾은 이유는 아래와 같다.

위에서 설명한 것과 같이, 데이터셋이 불균형한 경우, 모델 성능을 평가할 때 Accuracy보다 BCR을 사용하는 것이 더 적절하다고 할 수 있다. BCR은 Accuracy에 비해 각 클래스에 대한 예측 성능을 더 공정하게 평가하고 데이터의 불균형으로 인해 발생하는 편향된 결과를 줄이는 데 도움을 줄 수 있기 때문이다. 실제로, 본 데이터셋에서 학습을 진행하지 않고 모두 class2로 예측을 하더라도, class1과 class3는 하나도 맞추지 못하지만, class2의 비율이 50%가 넘어가기 때문에 Accuracy가 50%가 넘는 것을 알 수 있다. 따라서 이후, 최적의 하이퍼파라미터 조합을 찾고, 분류 성능을 비교는 과정에서, Accuracy보다 BCR을 중점적으로 평가하고자 한다.

Multinomial logistic regression에서 Grid search를 수행하고, 검증 데이터에 대한 AUROC 값을 기준으로 최적의 하이퍼파라미터 조합을 찾아보았고, 그 결과는 다음과 같다.

1.'C': 10

2. 'max_iter': 300

3. 'solver': 'newton-cg'

이때, validation 데이터셋에 대한 BCR 값은 0.4994이다.

validation 데이터셋에 대한 confusion matrix는 다음과 같다.

Actual / Prediction	1	2	3
1	74	124	79
2	35	984	125
3	3	510	136

test 데이터셋에 대하여 confusion matrix를 나타내 보았고, 그 결과는 다음과 같다.

Actual / Prediction	1	2	3
1	60	120	6
2	41	1024	103
3	1	514	131

다음으로 test data set에 대한 평가지표를 나타내면 다음과 같다.

Model	Accuracy	BCR
MLR	0.6075	0.4833

Accuracy 및 BCR, confusion matrix를 종합적으로 보았을 때, 좋은 성능을 보인다고 하기 어려워 보인다. BCR이 0.5도 안 되는 수치를 보여주는 것을 보아, 각 class에 대한 예측을 잘했다고 할 수 없는데, confusion matrix를 살펴보면, class2에 대한 예측은 뛰어난 편이지만, class1과 class3은 맞춘 수보다 틀린 수가 더 많은 것을 알 수 있다. 이는 학습 데이터에서 class2의 비율이 가장 많기 때문에, class2에 편향된 학습이 이루어졌다고 할 수 있다. class1과 class3도 class2로 예측한 것이 가장 많기 때문에, class2를 집중적으로 학습했다고 할 수 있고, 결과적으로 BCR 성능이 낮은 것을 알 수 있다.

2. Classification and Regression Tree (CART)

CART 학습을 수행하기 위해 사용된 하이퍼파라미터는 다음과 같다.

1. criterion ["gini", "entropy"]

    이 파라미터는 노드 분할에 사용되는 판단 기준을 결정한다.

    gini의 경우 지니 불순도를 사용하여 분할하고, entropy의 경우 엔트로피(information gain)를 사용하여 분할한다.

impurity가 얼마나 개선되는지 gain의 관점에서 평가하고 분기를 진행하기 때문에, 불순도 계산 함수가 무엇인지에 따라 분기 위치 및 여부가 달라질 수 있다. 따라서 주어진 데이터에 적합한 최적의 분할 기준을 선택하기 위해 위의 후보들을 선정하였다.

2. min_samples_split [10, 25, 50]

이 파라미터는 노드를 분할하기 위해 필요한 최소 샘플 수를 지정한다.

해당 값이 작은 경우, 더 복잡한 트리를 생성할 수 있지만, 과적합의 위험이 존재한다.

반대로 해당 값이 크다면, node가 split되지 않게 해줄 수 있어 과적합을 방지할 수 있다.

3. max_depth [None, 10, 20, 30]

이 파라미터는 트리의 최대 깊이를 제한한다.

더 깊은 트리는 더 complex하게 학습할 수 있지만, 과적합의 위험이 존재한다.

4. min_samples_leaf [10, 20, 40]

이 파라미터는 리프 노드에 필요한 최소 샘플 수를 지정한다.

해당 값이 작은 경우, 더 복잡한 트리를 생성한 수 있지만, 과적합의 위험이 존재한다.

반대로 해당 값이 큰 경우 과적합을 방지할 수 있다.

최적의 하이퍼 파라미터 조합을 찾기 위해 validation data를 기준으로 BCR을 측정하여 가장 높은 수치를 보이는 조합을 선택하였다.

그 결과, 하이퍼 파라미터에 대한 최적의 조합은 다음과 같다.

1. Best criterion: gini

2. Best min_samples_split: 10

3. Best max_depth: None

4. Best min_samples_leaf: 20

이 때, validation 데이터셋에 대한 BCR 값은 0.6262이다.

validation 데이터셋에 대한 confusion matrix는 다음과 같다.

Actual / Prediction	1	2	3
1	84	119	4
2	64	884	196
3	6	326	317

test 데이터셋에 대하여 confusion matrix를 나타내 보았고, 그 결과는 다음과 같다.

Actual / Prediction	1	2	3
1	75	106	5
2	60	921	187
3	8	319	319

다음으로 test data set에 대한 평가지표를 나타내면 다음과 같다.

Model	Accuracy	BCR
MLR	0.6075	0.4833
CART	0.6575	0.6317

MLR과 비교하였을 때 Accuracy, BCR 모두 성능이 개선된 것을 알 수 있다. 특히, BCR 값이 크게 증가한 것을 확인할 수 있는데, 이는 CART 모델이 MLR 모델에 비해, class1과 class3에 대한 예측을 더 잘하였기에 나타난 결과임을 알 수 있다. confusion matrix을 보았을 때, MLR에 비해 CART 모델이 class1, class3 예측을 더 잘하는 것을 알 수 있다. 특히, MLR 모델은 class3를 정확하게 예측하는 비율이 약 20%에 불과하였지만, CART 모델은 class3를 정확하게 예측하는 비율이 약 50%인 것을 보아 CART 모델이 예측을 더 잘한다고 할 수 있다. 전체적으로 class1과 class3을 class2로 잘못 예측하는 수가 줄어들었기 때문에 전체적인 성능이 향상된 것을 알 수 있다.

3. Artificial Neural Network (ANN)

ANN 학습을 수행하기 위해 사용된 하이퍼파라미터는 다음과 같다.

1. hidden_layer_sizes: [(25,), (50,), (100,), (25,25)]

    Neural Network에서 은닉층의 크기를 지정하는 하이퍼파라미터로, 이는 모델의 복잡도를 결정하는 중요한 요소 중 하나이다. 이때, 은닉층의 크기가 클수록 과적합의 위험성을 지니기에 적절한 크기를 선정하는 것이 필요하고, 따라서 여러 경우를 테스트하기 위해, 위와 같이 후보를 설정하였다.

2. learning_rate_init: [0.005, 0.01, 0.1]

learning_rate는 학습 속도를 결정하는 하이퍼파라미터로, 가중치 업데이트의 크기를 조절하여 학습 속도 및 수렴 속도를 조정할 수 있다. 이때, 너무 작은 학습률은 수렴 속도를 저하시키고, local minimum에서 나오지 못할 수도 있다. 또한, 너무 큰 학습률은 수렴하지 않고 발산할 가능성이 있으므로 학습률의 후보를 다음과 같이 3가지를 선정하였다.

3. activation: ['logistic', 'tanh', 'relu']

이 파라미터는 은닉층에서 사용할 활성화 함수를 결정한다. 이 때, logistic은 로지스틱 시그모이드 함수, tanh는 하이퍼볼릭 탄젠트 함수, relu는 ReLU(Rectified Linear Unit) 함수이다.

최적의 하이퍼 파라미터 조합을 찾기 위해 validation data를 기준으로 BCR을 측정하여 가장 높은 수치를 보이는 조합을 선택하고, 그 결과는 다음과 같다.

1. hidden_layer_sizes: (50,)

2. learning_rate_init: 0.01

3. activation: 'relu'

이 때, validation 데이터셋에 대한 BCR 값은 0.5884이다.

validation 데이터셋에 대한 confusion matrix는 다음과 같다.

Actual / Prediction	1	2	3
1	81	102	24
2	51	709	384
3	2	323	324

test 데이터셋에 대하여 confusion matrix를 나타내 보았고, 그 결과는 다음과 같다.

Actual / Prediction	1	2	3
1	66	96	24
2	61	725	382
3	7	326	313

다음으로 test data set에 대한 평가지표를 나타내면 다음과 같다.

Model	Accuracy	BCR
MLR	0.6075	0.4833
CART	0.6575	0.6317
ANN	0.5520	0.5733

ANN의 경우, MLR에 비해 BCR은 높은 수치를 보여주지만 Accuracy의 경우, 3개의 모델 중에서 가장 낮은 성능을 보이는 것을 알 수 있다. confusion matrix를 살펴보면, class1, class3를 정확히 예측하는 비율은 CART와 큰 차이가 없는 것을 알 수 있지만, class2를 다른 모델들에 비해 제대로 예측하지 못하는 것을 알 수 있다. 즉, ANN은 CART와 비교했을 때, class1, class3를 정확하게 예측하는 비율은 비슷하지만, class2를 정확하게 예측하는 비율은 약 20% 정도 떨어졌고, 그런 class2가 데이터셋에서 가장 많은 수를 가진 class이기 때문에, Accuracy 측면에서 가장 낮은 수치를 보이는 것을 알 수 있다.

전체적으로 보았을 때, CART 모델이 Accuracy, BCR 모든 평가지표에서 가장 높은 수치를 보여주는 것을 알 수 있고, 가장 우수한 성능을 보여준다고 할 수 있다. confusion matrix를 보았을 때도, CART 모델이 class2를 정확하게 예측하는 비율을 유지하며, class1과 class3에 대한 예측 성능을 끌어올린 것을 확인할 수 있고, 따라서 가장 높은 평가지표 수치를 보인다고 할 수 있다.

MLR 즉, 로지스틱 회귀 모델은 BCR 측면에서 현저히 낮은 수치를 보여주고, Accuracy도 낮은 수치를 보여주는데, 이는 해당 데이터셋이 독립변수와 종속변수 간의 관계를 선형 함수로 잘 나타낼 수 없는 비선형 데이터 셋이기 때문에, 제대로 예측하지 못했다는 해석이 가능하다.

다음으로 CART 모델은 비선형 관계를 포착할 수 있고, 이상치에 robust한 특징을 지니고 있다. 또한 데이터에서 변수 간의 복잡한 상호작용이 있는 경우, CART는 이를 효과적으로 모델링할 수 있고, 분할 시 클래스 간의 균형을 유지하려 하여, 클래스 불균형을 처리하는 데 상대적으로 유연하다고 할 수 있다. 따라서 3개의 모델 중에서 가장 높은 분류 성능을 보이는 것을 알 수 있다.

마지막으로 ANN의 경우, 비선형적이고 복잡한 데이터를 잘 학습한다는 장점이 있지만, 성능이 상대적으로 낮게 나온 이유는 다음과 같이 해석할 수 있다. 먼저, 해당 데이터셋이 불균형 데이터셋이고, 학습 데이터셋의 크기가 작아서 일 가능성이 있다. 해당 데이터셋은 데이터 라벨이 불균형하고, 학습 데이터 수가 많다고 보기 어렵기 때문에, 데이터 패턴을 파악하는데 있어 어려움이 있고, 결과적으로 과적합 될 가능성이 존재한다. 또한, 하이퍼파라미터 튜닝이 제대로 이루어지지 않았을 가능성도 있다. 더 많은 종류의 하이퍼파라미터를 넓은 범위로 Grid search한다면 더 좋은 성능을 보여줄 가능성이 있다.

아래 질문들에 대해서는 Base Learner가 CART와 ANN인 경우 [Q1]에서 선택된 hyperparameter를 사용하여 실행하고 그 결과를 이용하여 답하시오.

[Q2] CART의 Bagging 모델을 Bootstrap의 수를 (10, 30, 50, 100, 200, 300)의 순으로 증가시키면서 분류 정확도를 평가해보시오. 최적의 Bootstrap 수는 몇으로 확인되는가? 이 모델은 단일 모형과 비교했을 때 성능의 향상이 있는가?

Q1의 CART에서 가장 좋은 성능을 보였던 파라미터 조합은 다음과 같다.

1. Best criterion: gini

2. Best min_samples_split: 10

3. Best max_depth: None

4. Best min_samples_leaf: 20

따라서, 위의 hyperparameter를 사용한 CART를 base learner로 하여, Bagging 모델 학습을 진행하였다. 이때, Bootstrap의 수를 (10, 30, 50, 100, 200, 300)의 순으로 증가시키면서 학습을 진행하였다. 검증 데이터에 대한 결과는 다음과 같다.

# of Bootstrap	Accuracy	BCR
10	0.6650	0.6216
30	0.6740	0.6159
50	0.6775	0.6224
100	0.6730	0.6129
200	0.6745	0.6144
300	0.6725	0.6079

최적의 Bootstrap의 수는 BCR 값을 기준으로 선정하였고, 그 값은 50이다. 이때, Accuracy도 bootstrap의 수가 50일 때 가장 높은 수치를 보이는 것을 알 수 있다.

결과를 확인해보면 Bootstrap에 관계없이 결과가 모두 큰 차이 없는 것을 알 수 있다. 특히, Bootstrap이 가장 클 때, BCR 값은 가장 낮은 것을 보아 Bootstrap의 수를 무한정 늘리는 것이 성능 향상에 도움이 되지 않는다는 것을 알 수 있다. 이는 Bootstrap의 수가 계속 증가하더라도, 더 이상 개별 트리의 예측 variance를 줄이는 것이 힘들기 때문이라고 할 수 있다.

다음으로 단일 CART 모델과 CART Bagging 모델의 분류 성능을 비교해보았다.

Model	Accuracy	BCR
CART	0.6575	0.6317
CART Bagging	0.7015	0.6433

Bagging 모델이 단일 모델에 비해 Accuracy, BCR 모두 높은 것을 알 수 있고, 따라서 더 좋은 성능을 보인다고 할 수 있다. 특히, Accuracy가 BCR에 비해 더 큰 폭으로 향상되었다.

성능 향상의 원인을 정확하기 파악하기 위해 테스트셋에 대한 confusion matrix도 나타내 보았다.

아래는 Bagging CART 모델의 confusion matrix다.

Actual / Prediction	1	2	3
1	72	111	3
2	31	1010	127
3	2	323	321

아래는 단일 CART 모델의 confusion matrix다.

Actual / Prediction	1	2	3
1	75	106	5
2	60	921	187
3	8	319	319

두 모델의 confusion matrix를 보았을 때, Bagging 모델이 단일 모델보다 class2, class3를 더 정확하게 맞추는 것을 알 수 있다. class1를 정확하게 맞추는 비율은 소폭 감소하였지만, 그 차이는 미미하다고 할 수 있다. 결론적으로, class2와 class3 예측 성능, 특히 class2 예측 성능이 단일 모델에 비해 향상되었기 때문에, Bagging 모델의 성능이 뛰어나다고 할 수 있다.

Bagging CART 모델이 단일 CART 모델보다 더 뛰어난 성능을 보이는 근본적인 이유는 다음과 같이 해석할 수 있다. 먼저, 단일 CART 모델은 데이터의 작은 변화에도 민감하기 때문에 과적합 될 가능성이 있지만, Bagging 모델은 개별 트리의 예측 분산을 감소시키기 때문에 모델의 안정성이 증가한다고 할 수 있다. 즉, 일반화 성능이 향상되었다고 할 수 있고, 개별 트리들을 앙상블 하였기 때문에 다양성도 증가하였다고 할 수 있어, 결론적으로 성능 향상으로 이어졌다고 할 수 있다.

[Q3] Random Forest 모델의 Tree의 수를 (10, 30, 50, 100, 200, 300)의 순으로 증가시키면서 분류 정확도를 평가하고 다음 물음에 답하시오. 학습 과정에서는 변수의 중요도가 산출되도록 학습하시오.

[Q3-1] 최적의 Bootstrap 수는 몇으로 확인되는가?

Bootstrap의 수를 (10, 30, 50, 100, 200, 300)의 순으로 증가시키면서 학습을 진행하였다. 검증 데이

터에 대한 결과는 다음과 같다.

# of Tree	Accuracy	BCR
10	0.6000	0.5680
30	0.6295	0.5928
50	0.6385	0.5998
100	0.6495	0.6062
200	0.6525	0.6074
300	0.6570	0.6108

최적의 Bootstrap의 수는 BCR 값을 기준으로 선정하였고, 그 값은 300이다. 이때, Accuracy도 bootstrap의 수가 300일 때 가장 높은 수치를 보이는 것을 알 수 있다.

Random Forest 모델의 결과를 확인해보면 Tree의 수가 증가할수록 Accuracy 및 BCR이 점점 증가하는 것을 확인할 수 있다. 마지막 Tree의 수인 300까지도 Accuracy 및 BCR이 계속 증가하는 것을 보아, Tree의 수를 더 늘린다면 더 높은 성능을 보일 여지가 있음을 알 수 있다. tree의 수가 증가함에 따라 성능이 증가하는 이유는 다음과 같이 해석할 수 있다. 랜덤 포레스트는 각 트리가 서로 다른 데이터 샘플과 특징을 사용하여 학습하도록 설계되어 있기 때문에, 트리의 수가 증가하면 모델 내의 다양성이 증가하게 된다. 특히, 각 노드에서 최적의 분할 변수를 선택할 때, 전체 변수 중 무작위로 선택된 일부 변수만 고려하므로 트리 간의 상관관계를 줄이고 다양성을 높이기 때문에, Bagging 모델과 다르게 Tree 수에 따른 효과가 더욱 두드러지게 나타나는 것을 알 수 있다.

[Q3-2] 최적의 Tree 수를 기준으로 이 데이터셋에 대해서는 CART Bagging 과 Random Forest 중에서 더 높은 분류 정확도를 나타내는 모형은 무엇인가?

최적의 Tree 수를 기준으로 CART Bagging과 Random Forest의 분류 성능을 나타내 보았다.

Model	Accuracy	BCR
CART Bagging	0.7015	0.6433
Random Forest	0.6735	0.6152

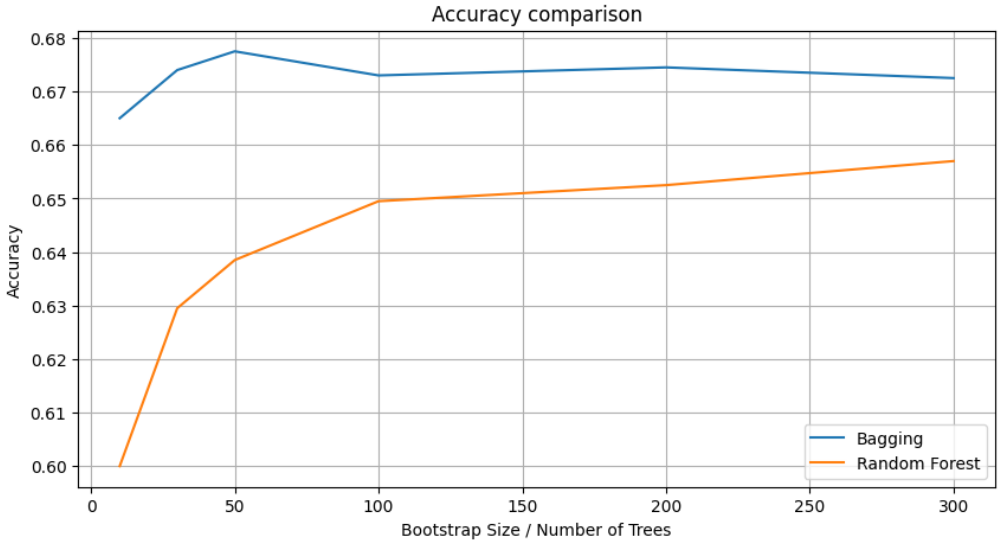
CART Bagging이 Random Forest보다 Accuracy, BCR 두 성능지표에서 더 뛰어난 분류 성능을 나타내는 것을 알 수 있다. 그러나 이 결과를 보고, 무조건 CART Bagging이 Random Forest보다 좋다고 하기엔 무리가 있다. 그 이유는 다음과 같다.

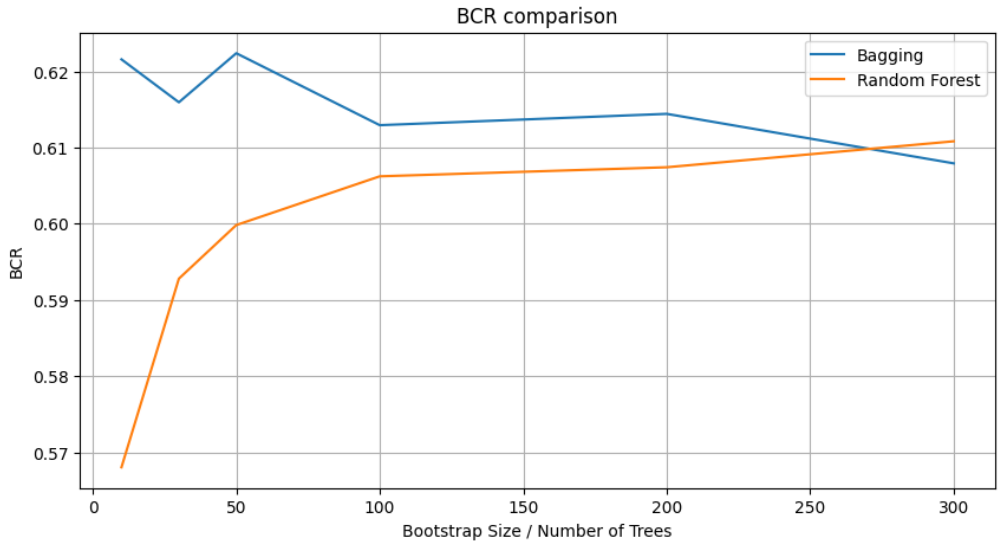
Random Forest는 최적의 분할 변수를 사용할 때, 무작위로 선택된 부분집합을 사용하여 각 Tree를 학습하기 때문에, Tree의 수가 적다면 중요한 패턴을 잡아내기 어려울 수 있다. 그렇지만, Tree

의 수가 많아진다면, 이러한 다양성을 토대로 예측 성능이 더욱 향상되고, 안정화될 수 있다. 실제로, Random Forest에서 위의 표로 나타낸 Tree의 수에 따른 성능지표를 보면, Tree의 수가 더 늘어난다면 성능이 더 높아질 여지를 보인다. 반면, CART Bagging의 경우, 전체 변수 중에서 최적 분기 변수를 선택하기 때문에 Bootstrap의 수가 작더라도 준수한 성능을 보여줄 수 있지만, 그 다양성은 Random Forest에 비해 작다고 할 수 있다. 따라서 정확한 비교를 위해서는 Random Forest의 Tree 수를 더욱 늘려 학습을 진행한 후에, 두 모델을 비교하는 것이 좋아 보인다.

[Q3-3] 각 Tree의 수(Bootstrap의 수)마다 CART Bagging 모형과의 분류 정확도를 비교할 수 있는 그래프를 도시하시오. Tree의 수는 CART Bagging과 Random Forest는 성능 차이에 영향을 미친다고 볼 수 있는가?

각 Tree의 수(Bootstrap의 수)마다 CART Bagging과 Random Forest 성능 차이를 확인할 수 있는 그래프는 아래와 같다. Accuracy와 BCR에 대하여 각각 나타내 보았다.





위의 두 개의 그래프는 검증 데이터로 평가할 때, 사용한 분류 성능 지표를 그래프로 나타낸 것이다.

먼저, Accuracy 그래프를 보았을 때, CART Bagging은 bootstrap size에 따른 차이가 거의 없는 것을 알 수 있다. 반면, Random Forest의 경우, tree의 수가 증가할수록 Accuracy 값이 계속해서 커지는 것을 알 수 있다. 이는 위에서 설명한 것과 같이, CART Bagging은 모든 변수를 사용하여 분할을 수행하지만, Random Forest는 무작위로 선택된 변수 집합을 사용하여 분할을 수행하기 때문에 Random Forest의 다양성이 더 높다고 할 수 있다. CART Bagging은 분할 시, 모든 노드를 고려하기 때문에 각 Tree마다 상관성이 높고 다양성이 낮을 가능성이 있다. 결론적으로 bootstrap size가 증가하여도 더 이상 다양성이 증가하지 않아 성능 개선이 어려울 수 있다. Random Forest의 경우, tree의 수가 작다면 중요한 특징을 제대로 잡아내는 것이 어렵기 때문에, 제대로 된 성능을 보여준다고 보기 어렵다. 하지만 Tree의 수가 증가할수록 더욱 일반화되어 다양성으로 인한 장점이 극대화된다고 할 수 있고, CART Bagging보다 더 나은 성능과 일반화 능력을 보일 가능성이 존재한다고 해석할 수 있다.

다음으로 BCR 그래프를 보면, Accuracy 그래프를 보았을 때와 동일한 해석이 가능한 것을 알 수 있다. CART Bagging의 경우, 오히려 성능이 감소하는 모습을 보이지만, Random Forest의 경우, tree 수에 따라 BCR이 계속해서 증가하는 모습을 보인다. 심지어 검증데이터에선 tree의 수가 최대인 300에 도달했을 때, CART Bagging보다 더 높은 수치를 보이는 것을 알 수 있다. 즉, tree의 수가 더욱 증가한다면, 테스트 데이터셋에서도 CART Bagging보다 더 높은 평가지표 수치를 얻을 가능성이 있다고 할 수 있다.

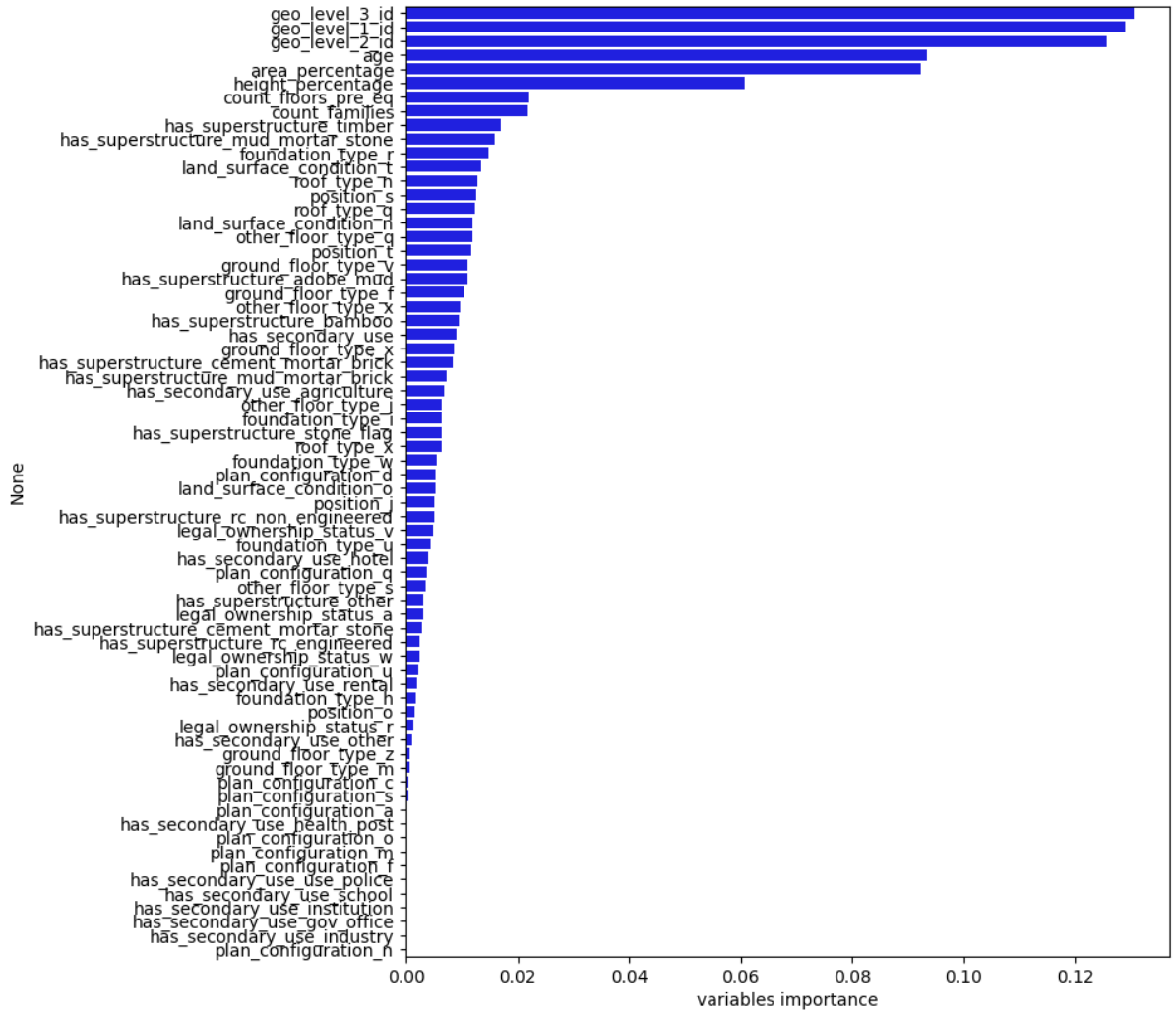
종합적으로 Tree의 수는 Random Forest의 성능 차이에 큰 영향을 미친다고 할 수 있지만, Bootstrap의 수는 CART Bagging에 별다른 성능 차이를 만들지 못한다고 할 수 있다.

추가적으로, Random Forest에서 최적의 모델에 대한 변수 중요도는 다음과 같다.

	variables importance
geo_level_3_id	0.130482
geo_level_1_id	0.129077
geo_level_2_id	0.125702
age	0.093476
area_percentage	0.092357
...	...
has_secondary_use_school	0.000045
has_secondary_use_institution	0.000031
has_secondary_use_gov_office	0.000029
has_secondary_use_industry	0.000014
plan_configuration_n	0.000000

68 rows × 1 columns

변수 중요도를 표로 보는 것에는 한계가 있어 아래와 같이 그래프로 나타내보았다.



변수 중요도를 통해, 각 변수가 예측 모델에서 차지하는 기여도를 알 수 있다. 뒤의 문제에서 자세하게 설명하겠지만, 'geo_level_3_id', 'geo_level_1_id', 'geo_level_2_id'가 예측에 있어 큰 영향을 미치는 것을 알 수 있고, 'age', 'area_percentage'가 다음으로 큰 영향을 미치는 것을 알 수 있다.

[Q4] [Q1]에서 찾은 최적의 hyperparameter를 이용하여 ANN 단일모형을 10번 반복하여 테스트 정확도를 평가해보시오. Accuracy 와 BCR의 평균 및 표준편차를 기록하시오.

Q1에서 찾은 ANN의 최적의 하이퍼파라미터는 다음과 같다.

1. activation: 'relu'

2. hidden_layer_sizes: (50,)

3. learning_rate_init: 0.01

위의 하이퍼파라미터 조합으로 ANN을 10번 반복하여 학습하였고, 분류 정확도를 구하였다. 그 결과는 다음과 같다.

ANN	Accuracy	BCR
Mean	0.4314	0.4369
Standard deviation	0.1270	0.1060

결과를 확인해보면, Accuracy와 BCR 값이 모두 낮은 수치를 보여주고 있고, 표준편차도 큰 것을 알 수 있다. 아래는 [Q1]에서 같은 하이퍼파리미터 조합을 통해 구했던 평가지표이다.

Model	Accuracy	BCR
ANN	0.5520	0.5733

같은 하이퍼파라미터를 사용했는데, 각각의 ANN 모델이 성능적으로 큰 차이를 보이고, 평균적으로 낮은 성능을 보이는 이유는 다음과 같이 해석할 수 있다.

먼저, 가중치 초기화가 어떻게 되었는지에 따라 모델의 최종 성능에 영향을 미칠 수 있다는 것이다. ANN의 가중치는 학습 시작 시 무작위로 초기화 되기 때문에, 다른 학습 경로를 유도하여 모델의 최종 성능이 달라질 수 있다. 따라서, 특정 초기 가중치에서는 지역 최소값에 빠져, 제대로 학습되지 않았을 가능성이 있다.

다음으로, 훈련 데이터의 다양성 부족이 원인이 될 수 있다. 6000개의 데이터로는 다양한 패턴을 충분히 포괄하지 못하기 때문에, 학습이 불안정해지고, 데이터셋에 없는 패턴이나 극단 값에 제대로 대응하지 못할 수 있다.

마지막으로 과적합의 가능성도 존재한다. 데이터가 적으면 모델이 훈련 데이터에 과적합 되어 일반화 성능이 떨어질 수 있고, 검증 데이터셋이나 테스트 데이터셋에서의 성능이 다소 낮아질 수 있다.

이러한 문제를 해결하기 위한 방법은 다음과 같다.

Xavier Initialization와 같은 가중치 초기화 기법으로, 초기 가중치의 영향을 줄일 수 있다. 또한, 학습데이터의 수를 더 크게 하고, Early stopping을 통해 과적합을 방지하고 일반화 성능을 높일 수 있다.

결론적으로, [Q1]에서 구한 하이퍼파라미터 조합은 최적의 하이퍼파라미터 조합이 아닐 가능성이 있다. 우연히 좋은 초기값에서 시작하여 평균보다 좋은 결과를 얻었고, 따라서 이때의 하이퍼파라

미터 조합이 최적이라고 하였지만, 동일한 파라미터로 10번 학습시켜 평균을 구한 결과, 훨씬 더 낮은 성능을 보이는 것을 알 수 있었다. 따라서 평균 성능을 고려하여 최적의 하이퍼파라미터 조합을 선택한다면, 다른 조합이 선택될 가능성도 존재한다고 생각한다.

[Q5] ANN Bagging 모델에 대해 다음 물음에 답하시오.

[Q5-1] Bootstrap의 수를 (10, 30, 50, 100, 200, 300)의 순으로 증가시키면서 Accuracy 와 BCR을 구해보시오.

Q1의 ANN에서 가장 좋은 성능을 보였던 파라미터 조합은 다음과 같다.

1. activation: 'relu'

2. hidden_layer_sizes: (50,)

3. learning_rate_init: 0.01

따라서, 위의 hyperparameter를 사용한 ANN을 base learner로 하여, Bagging 모델 학습을 진행하였다. 이때, Bootstrap의 수를 (10, 30, 50, 100, 200, 300)의 순으로 증가시키면서 학습을 진행하였다. 검증 데이터에 대한 결과는 다음과 같다.

# of Bootstrap	Accuracy	BCR
10	0.5755	0.2792
30	0.5780	0.4746
50	0.5735	0.3689
100	0.5720	0.3608
200	0.5730	0.3481
300	0.5725	0.3481

전체적으로 Bootstrap 수에 따라, Accuracy는 큰 차이가 없는 것을 알 수 있지만, BCR은 Bootstrap 수에 따라 차이가 존재한다. Bootstrap 수가 제일 작은 10에서, BCR 값이 0.2792로 매우 낮은 수치를 보여주고 있고, Bootstrap 수가 30일 때, 가장 높은 수치인 0.4746을 보여주고 있다. Bootstrap 수가 50 이상일 때는 모두 비슷한 성능을 보여주는 것을 알 수 있다.

[Q5-2] 최적의 Bootstrap 수는 몇으로 확인되는가?

가장 최적의 Bootstrap 수는 BCR 값이 가장 높은 30이다. 이때, Accuracy 값도 0.5780으로 가장 큰 값을 가지는 것을 확인할 수 있다.

[Q5-3] 이 모델은 최적의 단일 모형과 비교했을 때 어떠한 성능의 차이가 있는가?

아래 표는 최적의 ANN 단일 모델과 ANN Bagging의 성능을 나타낸 것이다.

Model	Accuracy	BCR
ANN	0.5520	0.5733
ANN Bagging	0.5785	0.4594

Accuracy에서 Bagging 모델이 단일 모델보다 약간 더 높은 수치를 보이지만, BCR에서는 단일 모델이 훨씬 더 높은 수치를 보이는 것을 알 수 있다. 결론적으로 BCR에서 단일 모델의 수치가 훨씬 더 높기 때문에, Bagging보다 단일 모델의 성능이 더 뛰어나다고 할 수 있다. 원래라면, Bagging이라는 앙상블 기법을 통해, 더 일반화되고 더 좋은 성능의 모델을 얻을 수 있지만, 이 경우 Bagging보다 단일 모델의 성능이 더 뛰어나다. 그 이유는 다음과 같이 해석할 수 있다.

ANN Bagging에서 앙상블 효과가 제대로 나타나지 않았을 가능성이 있다. Bagging은 여러 개의 모델을 결합하여 성능을 향상시킨다. 이때, 각 모델이 다양성을 가지고 상호 보완성을 가져야 성능 향상이 일어나는데, 각 모델이 단일 모델보다 낮은 성능을 보인다면, 상호 보완성을 띄지 않을 수 있다. 실제로, [Q4]에서 ANN 단일모형을 10번 반복하여 평가한 결과, 모델의 평균 성능이 좋지 않은 것을 확인할 수 있었고, 이들이 결합되면 결론적으로 성능 하락이 나타날 수 있다.

[Q6] Adaptive Boosting(AdaBoost)에 대해 다음 물음에 답하시오.

[Q6-1] Hyperparameter 후보 값들을 명시하고, Validation dataset을 통해 최적의 hyperparameter 값을 찾아보시오.

Adaboost 학습을 수행하기 위해 사용된 하이퍼파라미터는 다음과 같다.

1. n_estimators : [50, 100, 200]

    이 파라미터는 앙상블에 사용되는 weak learner의 개수를 지정한다. 즉, 사용할 결정 트리의 개수를 의미하는 것으로, 이 값이 클수록 앙상블의 복잡도가 증가하고, 더 많은 weak learner가 결합되어 모델의 편향이 감소하지만, 계산 시간이 증가한다.

2. learning_rate : [0.01, 0.1, 0.5, 1.0]

    이 파라미터는 각 weak learner가 모델에 기여하는 정도를 조절한다. 즉, 각 weak learner의 가중치 조절에 사용되는 것으로, learning rate가 작을수록 각 weak learner의 영향이 작아진다.

Grid search를 수행하고, 검증 데이터에 대한 BCR 값을 기준으로 최적의 하이퍼파라미터 조합을 찾아보았고, 그 결과는 다음과 같다.

1. n_estimators : 200

2. learning_rate : 1.0

이 때, 최적의 하이퍼파라미터 조합에 대한 검증 데이터 BCR 값은 0.6040이다.

[Q6-2] 최적의 hyperparameter 값을 이용하여 AdaBoost 모델을 학습한 뒤, Test dataset에 적용하여 먼저 구축된 모델들과 분류 성능을 비교해보시오.

위에서 구한 최적의 하이퍼파라미터 조합을 이용하여 AdaBoost 모델을 학습하고, 테스트 데이터셋에 대한 평가 지표를 구하면 다음과 같다. 이때, 먼저 구축된 다른 모델들의 평가 지표도 같이 나타내었다.

Model	Accuracy	BCR
MLR	0.6075	0.4833
CART	0.6575	0.6317
ANN	0.5520	0.5733
CART Bagging	0.7015	0.6433
Random Forest	0.6735	0.6152
ANN Bagging	0.5785	0.4594
AdaBoost	0.6540	0.6151

보다 정확한 분류 성능을 확인하기 위해 테스트 데이터셋에 대한 confusion matrix도 나타냈다.

Actual / Prediction	1	2	3
1	76	108	2
2	56	948	164
3	1	361	284

다음은 비교를 위해 현재 BCR에서 가장 좋은 성능을 보이고 있는 CART Bagging의 confusion matrix도 나타냈다.

Actual / Prediction	1	2	3
1	72	111	3
2	31	1010	127
3	2	323	321

위의 모델 중, AdaBoost가 유일하게 boosting을 사용한 모델이다. BCR을 기준으로 보았을 때, CART Bagging의 성능이 가장 좋고, CART, Random Forest, AdaBoost 순으로 성능이 좋은 것을 알 수 있다. Accuracy를 보았을 때도, CART Bagging이 유일하게 70%가 넘는 수치를 보여주고, Random Forest, CART, AdaBoost 순으로 성능이 좋은 것을 알 수 있다. 지금까지 구축한 다른 모

델들과 비교했을 때, 중간 정도의 성능을 보인다고 할 수 있다. confusion matrix를 보았을 때, AdaBoost 모델은, 현재 가장 뛰어난 성능을 보이는 CART Bagging보다 class1은 더 정확하게 예측하지만 class2, class3 예측 성능은 다소 떨어지는 모습을 보여준다. 같은 Decision tree 기반 모델인 CART와 비교했을 때도, 다소 뒤떨어지는 성능을 보이는 것을 보아, 앙상블 기법을 사용했음에도 아쉬운 성능을 보인다고 할 수 있다. AdaBoost가 아쉬운 성능을 보이는 이유는 다음과 같다.

Decision Tree는 노이즈에 민감한 모델인데, 데이터에 노이즈가 많은 경우, 트리는 이러한 노이즈를 과적합할 가능성이 높다. 특히, Boosting 알고리즘은 이러한 트리를 반복적으로 학습하기 때문에 노이즈가 강조될 수 있다. 현재 사용하는 데이터셋은 샘플링을 진행한 데이터셋이기 때문에, 그 크기가 작은 편이라고 할 수 있고, 따라서 과적합 및 노이즈가 강조되었을 가능성이 있다. 결론적으로, 해당 데이터셋에서는 AdaBoost보다 더 일반화된 모델을 사용하는 것이 좋다고 할 수 있다.

[Q7] Gradient Boosting Machine(GBM)에 대해 다음 물음에 답하시오.

[Q7-1] Hyperparameter 후보 값들을 명시하고, Validation dataset을 통해 최적의 hyperparameter 값을 찾아보시오.

Gradient Boosting Machine(GBM) 학습을 수행하기 위해 사용된 하이퍼파라미터는 다음과 같다.

1. n_estimators : [50, 100, 200]

    이 파라미터는 앙상블에 사용되는 weak learner의 개수를 지정한다. 즉, 사용할 결정 트리의 개수를 의미하는 것으로, 이 값이 클수록 앙상블의 복잡도가 증가하고, 더 많은 weak learner가 결합되어 모델의 편향이 감소하지만, 계산 시간이 증가한다.

2. learning_rate : [0.01, 0.1, 0.5]

    이 파라미터는 각 weak learner가 모델에 기여하는 정도를 조절한다. 즉, 각 weak learner의 가중치 조절에 사용되는 것으로, learning rate가 작을수록 각 weak learner의 영향이 작아진다.

3. max_depth : [5, 7, 10]

    각 결정 트리의 최대 깊이를 나타낸다. 트리의 깊이가 깊을수록 더 많은 정보를 학습할 수 있지만, 과적합의 위험이 증가한다.

4. subsample : [0.7, 0.8, 0.9]

    각 트리를 학습할 때, 사용할 데이터의 비율을 나타낸다. 전체 데이터의 일부만을 사용하여 트리를 학습함으로써 모델의 분산을 줄이고 일반화 성능을 향상시킬 수 있다. 이 값이 작을수록 과적합되는 것을 방지할 수 있다.

Grid search를 수행하고, 검증 데이터에 대한 BCR 값을 기준으로 최적의 하이퍼파라미터 조합을 찾아보았고, 그 결과는 다음과 같다.

1. n_estimators : 200

2. learning_rate : 0.5

3. max_depth : 5

4. subsample : 0.8

이 때, 최적의 하이퍼파라미터 조합에 대한 검증 데이터 BCR 값은 0.6425이다.

[Q7-2] 최적의 hyperparameter 값을 이용하여 GBM 모델을 학습(변수의 중요도가 산출되도록 학습)한 뒤, Test dataset에 적용하여 먼저 구축된 모델들과 분류 성능을 비교해보시오

위에서 구한 최적의 하이퍼파라미터 조합을 이용하여 GBM 모델을 학습하고, 테스트 데이터셋에 대한 평가 지표를 구하면 다음과 같다. 이때, 먼저 구축된 모델들의 평가 지표도 같이 나타내었다.

Model	Accuracy	BCR
MLR	0.6075	0.4833
CART	0.6575	0.6317
ANN	0.5520	0.5733
CART Bagging	0.7015	0.6433
Random Forest	0.6735	0.6152
ANN Bagging	0.5785	0.4594
AdaBoost	0.6540	0.6151
GBM	0.6785	0.6631

보다 정확한 분류 성능을 확인하기 위해 테스트 데이터셋에 대한 confusion matrix도 나타냈다.

Actual / Prediction	1	2	3
1	73	104	9
2	61	894	213
3	8	248	390

다음은 비교를 위해 전에 BCR에서 가장 좋은 성능을 보이고 있던 CART Bagging의 confusion matrix도 나타냈다.

Actual / Prediction	1	2	3
1	72	111	3
2	31	1010	127
3	2	323	321

먼저, GBM의 성능 지표를 살펴보면, Accuracy의 경우 CART Bagging 다음으로 성능이 좋은 것을 알 수 있고, BCR의 경우 0.6631로 전체 모델 중 가장 성능이 좋은 것을 알 수 있다. 이러한 결과가 나오는 이유를 정확하게 확인하기 위해 confusion matrix를 확인해보았는데, GBM이 CART Bagging보다 class2 예측 성능은 떨어졌지만, class3를 더 정확하게 예측하는 것을 확인할 수 있다. 즉, 다른 모델들에 비해 class3에 해당하는 부분을 집중적으로 학습하였기 때문에, class2 예측 성능이 다소 떨어졌더라도 BCR 수치가 높은 것을 확인할 수 있다. 다만, 가장 많은 수를 가진 class2의 예측 성능이 떨어졌기 때문에, Accuracy 값은 CART Bagging에 비해 낮은 수치를 보이는 것을 알 수 있다.

GBM의 성능이 좋게 나온 이유는 다음과 같이 해석할 수 있다. GBM은 weak learners를 결합하여 점진적으로 모델 성능을 향상시키는 앙상블 학습을 진행하기 때문에, 모델의 복잡성을 효과적으로 증가시켰다고 할 수 있다. AdaBoost도 이와 동일한 특징을 가지지만, AdaBoost에 비해 GBM의 성능이 뛰어난 이유는 여러 이유가 있을 수 있다. 먼저, GBM은 더 복잡한 트리를 weak learner로 사용할 수 있어 데이터의 복잡한 패턴을 학습할 수 있다. 또한, GBM은 각 learner에서 residual error를 줄이는 방식으로 학습하기 때문에, 보다 안정적이고 일반화된 방식으로 작동하며, 과적합 위험이 상대적으로 적다고 할 수 있다. 마지막으로 GBM은 subsampling 기법을 사용하여 학습할 때 샘플의 일부만 사용하게 되고, 이는 과적합을 줄이고 모델의 일반화 성능을 높이는데 도움을 준다. 결론적으로 GBM이 데이터 패턴을 더 효과적으로 학습하고, 과적합이 방지되어 더 나은 일반화 성능을 보인다고 할 수 있고, 결론적으로 현재 데이터셋에서 BCR 기준 가장 뛰어난 성능을 보여준다고 할 수 있다.

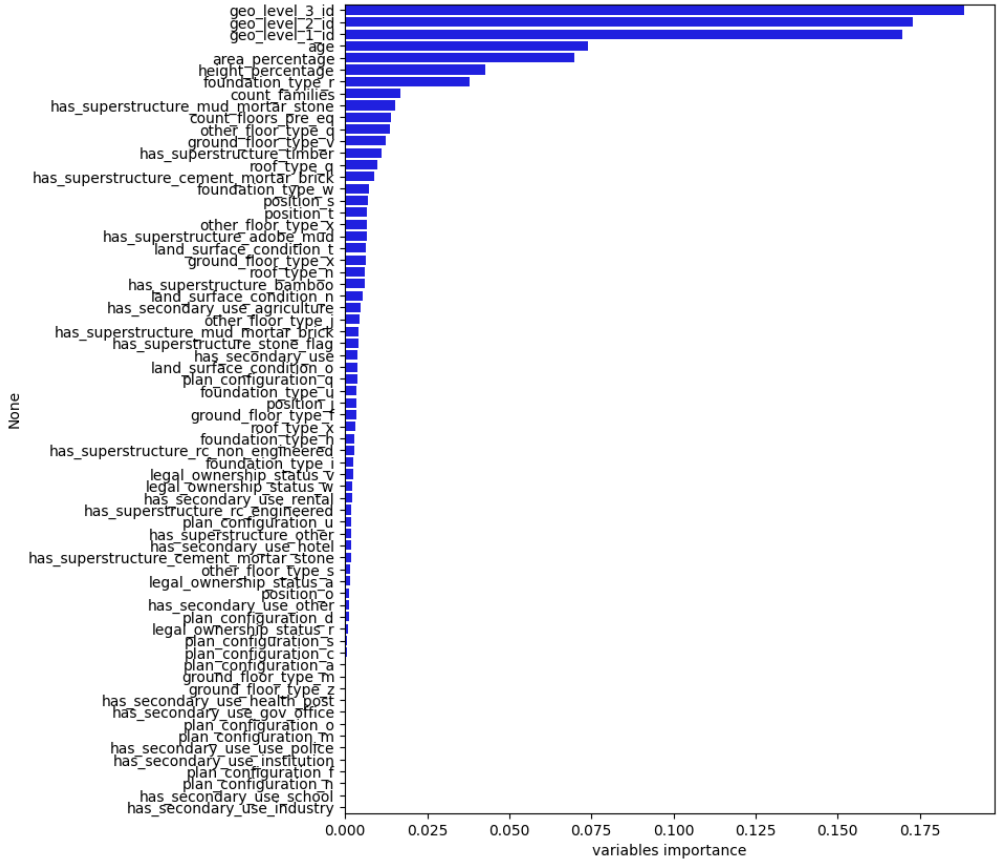
[Q7-3] 산출된 변수의 중요도를 해석해보고, Random Forest 모델에서 산출된 주요 변수와 비교해보시오.

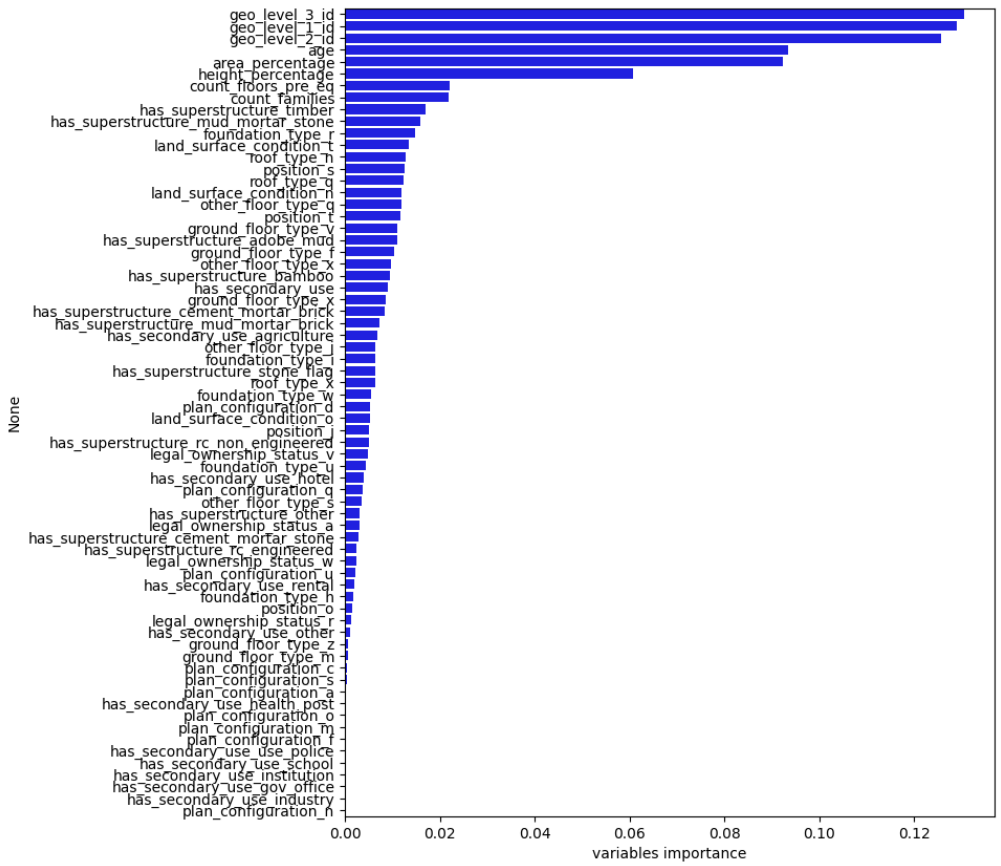
GBM과 RF를 통해 산출된 변수의 중요도는 다음과 같다. 좌측이 GBM 모델에서의 변수 중요도, 우측이 RF 모델에서의 변수 중요도이다.

	variables importance
geo_level_3_id	0.188394
geo_level_2_id	0.172993
geo_level_1_id	0.169598
age	0.073972
area_percentage	0.069832
...	...
has_secondary_use_institution	0.000022
plan_configuration_f	0.000000
plan_configuration_n	0.000000
has_secondary_use_school	0.000000
has_secondary_use_industry	0.000000

	variables importance
geo_level_3_id	0.130482
geo_level_1_id	0.129077
geo_level_2_id	0.125702
age	0.093476
area_percentage	0.092357
...	...
has_secondary_use_school	0.000045
has_secondary_use_institution	0.000031
has_secondary_use_gov_office	0.000029
has_secondary_use_industry	0.000014
plan_configuration_n	0.000000

다음은 그래프로 나타낸 변수 중요도로, 위의 그래프가 GBM 모델, 아래가 RF 모델이다.





먼저, GBM에서 변수 중요도가 높게 산출된 변수들을 확인해보면 'geo_level_3_id', 'geo_level_2_id', 'geo_level_1_id' 순으로 중요도가 높은 것을 알 수 있고, 다음으로 'age', 'area_percentage', 'height_percentage' 순으로 중요도가 높은 것을 알 수 있다.

다음으로, RF에서 변수 중요도가 높게 산출된 변수들을 확인하면 'geo_level_3_id', 'geo_level_1_id', 'geo_level_2_id' 순으로 중요도가 높은 것을 알 수 있고, 다음으로 'age', 'area_percentage', 'height_percentage' 순으로 중요도가 높은 것을 알 수 있다.

이 둘을 비교해보았을 때, 중요한 변수로 선정된 변수가 동일하다는 것을 알 수 있다. 물론, GBM에서의 2,3순위가 RF에서는 3,2순위로 순서가 조금 바뀌는 경우도 존재하지만, 전체적으로 중요도가 높은 상위 6개의 변수들은 GBM과 RF에서 동일한 것을 알 수 있다. 즉, 상위 6개의 변수들은 class를 분류하는데 있어, 중요한 역할을 수행한다고 할 수 있다. 특히, 'geo_level_3_id', 'geo_level_2_id', 'geo_level_1_id'는 다른 변수들에 비해 훨씬 높은 중요도를 가지고 있고, 셋의 중요도에 큰 차이가 없어, GBM과 RF 모델에서 예측에 있어 중요한 역할을 하고 있다고 볼 수 있다.

두 개의 그래프를 확인해보았을 때, 하나 다른 점은 중요도 점수 차이이다. GBM의 경우, 중요도 순위가 높을수록 높은 가중치를 가지고 있지만, RF의 경우, GBM에 비해 낮은 가중치를 가지고 있음을 알 수 있다. 또한, RF는 GBM에 비해 중요도가 더 분산되어 있다고 할 수 있다. 표를 보았을 때도, GBM의 하위 4개의 노드는 중요도가 0에 수렴하지만, RF의 경우는 중요도가 가장 낮은 변수를 제외하고, 0보다 큰 중요도를 가지고 있다. 이러한 결과를 보이는 이유는 Boosting과 Bagging의 차이라고 할 수 있다. GBM은 Boosting 방법을 사용하는데, 이때 잔여오차를 줄이기 위해 모든 변수를 고려하여 최적의 변수를 선택한다. 따라서 중요한 변수가 반복적으로 선택될 가능성이 높고, 이는 특정 변수에 중요도가 집중되는 경향을 보인다. 그러나 RF의 경우, 변수 샘플링을 통해 무작위성을 도입하기 때문에, 변수 중요도가 다양하게 분산될 가능성이 높아진다. 따라서 위와 같은 이유로 GBM에 비해 RF의 변수 중요도가 더 분산되어 있다는 것을 해석할 수 있다.

[Q8] 총 여덟 가지의 모델(Multinomial logistic regression, CART, ANN, CART Bagging, ANN Bagging, Random Forest, AdaBoost, GBM) 중 BCR 관점에서 가장 우수한 분류 정확도를 나타내는 모형은 무엇인가?

총 8개의 모델에 대한 Accuracy 및 BCR을 나타낸 표는 아래와 같다.

Model	Accuracy	BCR
MLR	0.6075	0.4833
CART	0.6575	0.6317
ANN	0.5520	0.5733
CART Bagging	0.7015	0.6433

Random Forest	0.6735	0.6152
ANN Bagging	0.5785	0.4594
AdaBoost	0.6540	0.6151
GBM	0.6785	0.6631

이 중 BCR을 기준으로 가장 우수한 분류 정확도를 나타내는 모형은 GBM이다.

BCR을 기준으로 보았을 때, GBM > CART Bagging > CART > Random Forest > AdaBoost > ANN > MLR > ANN Bagging 순으로 분류 성능이 좋다고 할 수 있다.

이 중, GBM, CART Bagging, CART가 BCR에서 다른 모델들에 비해 높은 수치를 보이는데, 이들은 모두 decision tree를 기반으로 하는 방법을 사용한다. Random Forest, AdaBoost도 ANN에 비해 높은 성능을 보이는데, 이를 토대로 현재 데이터셋에서 ANN에 비해 Decision Tree가 더 좋은 성능을 보인다고 할 수 있다. 이러한 결과는 ANN은 데이터셋이 불균형하고 노이즈가 많은 경우, 과적합 되기 쉽지만, Decision Tree는 상대적으로 단순한 모델이고, 앙상블 기법을 통해 과적합을 줄이는 것이 가능하기 때문에 나온 결과라고 해석할 수 있다. 이러한 이유로 ANN에서 과적합 방지를 위해 드롭아웃, L1,L2 정규화 등 다양한 정규화 기법을 적용한다면, 성능 개선의 여지가 있다고 생각한다.

ANN, ANN Bagging이 BCR에서 낮은 수치를 보인다는 점에서 현재 데이터셋이 불균형 데이터이고, 샘플링을 통해 데이터셋의 크기도 작아졌기 때문에, 학습에 어려움이 있었다고 해석할 수 있다. 또한, ANN은 다른 모델들에 비해 많은 하이퍼파라미터들이 있기 때문에, 모델을 최적화하는 것이 상대적으로 어려운 편이다. 그렇기에 ANN에서 적절한 하이퍼파라미터 튜닝이 이루어진다면 성능 개선의 여지는 충분히 있다고 할 수 있다.

그러나 이때, ANN의 하이퍼파라미터 튜닝이 잘 되어 성능이 개선되어 Decision Tree 기반 모델의 성능을 뛰어넘는다고 하더라도, 성능 차이가 크지 않다면, 변수 중요도를 파악할 수 있는 Decision Tree 기반 모델을 사용하는 것이 좋을 수 있다. ANN은 블랙박스 모델로 내부 작동 원리를 이해하기 어려워 설명력에 있어 Decision Tree보다 떨어지기 때문이다.

마지막으로 Multinomial logistic regression의 BCR 성능이 가장 낮은 것을 보아, 해당 데이터셋은 독립변수와 종속변수 간의 관계를 선형 함수로 잘 나타낼 수 없는 비선형 데이터셋이라는 해석도 가능하다.

[Extra Question]

이 데이터셋은 아래 표와 같이 Class2>Class3>Class1 순으로 높은 비중을 차지하고 있으며, 범주의 불균형이 상당한 수준이다. [Q8]에서 선정된 가장 우수한 모델(알고리즘 및 hyperparameter)에 대해서 데이터 전처리 관점에서 불균형을 해소하여 분류 성능을 향상시킬 수 있는 아이디어를 제시하고 실험을 통해 검증해보시오.

해당 데이터셋은 class1의 개수가 25124, class2의 개수가 148259, class3의 개수가 87218로 범주의 불균형이 상당한 수준이다. class1과 class3를 더한 개수가 class2를 넘지 못하기 때문에 모두 class2로 예측하여도 accuracy가 50%가 넘는 결과를 보여준다. 또한, 위에서 여러 모델들을 학습한 결과 class2에 비해 class1, class3의 예측 정확도는 낮은 것을 알 수 있었고, 이러한 데이터 불균형으로 인해 BCR 수치가 낮게 나오는 것을 알 수 있었다.

클래스 불균형을 해결하는 방법은 크게 2가지 접근법이 존재한다. 첫번째로, 데이터 수준 접근법으로 resampling 기법이 있다. 해당 기법은 다시 Oversampling과 Undersampling으로 나뉜다. 두번째로, 알고리즘 수준 접근법으로 클래스 가중치 조정, 앙상블 기법 등이 존재한다. 앙상블 기법은 앞선 문제에서 사용하였을 때, GBM과 같이 단일 모델보다 더 나은 성능을 보여주는 것을 확인하였다.

본 문제에서는 데이터 전처리 관점에서 불균형을 해소하여 분류 성능을 향상시키기 위해, resampling을 사용하고자 한다. Oversampling은 소수 클래스의 데이터를 복제하여 데이터셋의 균형을 맞추는 것이고, Undersampling을 다수 클래스의 데이터를 줄여 데이터셋의 균형을 맞추는 것이다.

현재 sampling을 진행한 데이터의 수가 10000개이고, 그 중 테스트 데이터셋은 6000개이다. 현재 데이터셋의 크기도 크지 않은 편인데, Undersampling을 진행할 경우 데이터셋의 크기가 더욱 작아지고, 결과적으로 과적합 문제가 커질 가능성이 높다. 따라서 Oversampling을 진행하여 class1, class3의 데이터 수를 class2의 데이터 수와 동일하게 하고자 한다.

현재 학습 데이터셋에 대하여 Oversampling을 진행한 결과는 다음과 같다.

```
샘플링 전 클래스별 샘플 개수: Counter({2: 3430, 3: 1973, 1: 597})
샘플링 후 클래스별 샘플 개수: Counter({2: 3430, 3: 3430, 1: 3430})
```

validation 데이터셋과 test 데이터셋의 경우, 다른 모델들과 동일한 데이터셋으로 평가해야 공정한 평가가 가능하기 때문에 Oversampling을 진행하지 않았다.

이후, Q8에서 가장 우수한 모델로 선정하였던 GBM을 사용하여 학습을 진행하였고, 그때의 하이퍼파라미터 조합은 아래와 같다.

1. n_estimators : 200

2. learning_rate : 0.5

3. max_depth : 5

4. subsample : 0.8

학습을 진행한 결과, 검증 데이터셋에서의 Accuracy는 0.6425, BCR은 0.6670 값이 나왔다.

다음으로 학습 데이터 셋에 대한 분류 성능 평가는 다음과 같다.

Model	Accuracy	BCR
MLR	0.6075	0.4833
CART	0.6575	0.6317
ANN	0.5520	0.5733
CART Bagging	0.7015	0.6433
Random Forest	0.6735	0.6152
ANN Bagging	0.5785	0.4594
AdaBoost	0.6540	0.6151
GBM	0.6785	0.6631
GBM (resampling)	0.6685	0.6746

이전에 BCR 값이 가장 높았던 GBM 모델에 비해 이번에 새롭게 학습한 GBM(resampling) 모델의 BCR 값이 더 높아진 것을 확인할 수 있고, Accuracy는 소폭 감소한 것을 확인할 수 있다. 즉, 기존보다 더 balance 있게 변수들을 예측하는 것을 확인할 수 있는데, 더 정확하게 결과를 확인하기 위해 confusion matrix도 나타내 보았다.

아래는 새롭게 학습한 GBM(resampling)의 confusion matrix이다.

Actual / Prediction	1	2	3
1	83	94	9
2	78	859	231
3	10	248	395

다음은 기존 GBM 모델의 confusion matrix이다.

Actual / Prediction	1	2	3
1	73	104	9
2	61	894	213
3	8	248	390

두 개의 confusion matrix의 결과를 확인하고 한 가지 놀란 사실이 있다. 본 과제에서 class3 예측 성능은 증가한 경우가 많았지만, class1 예측 성능은 유의미하게 차이 나는 경우가 별로 없었다.

그러나 이번에 새롭게 학습한 GBM 모델의 경우, class1 예측에 성공한 수가 처음으로 80을 넘은 것을 확인할 수 있다. 또한, class3 예측 성공도 소폭 증가한 것을 알 수 있다. 이때, class1의 예측 성능이 조금 증가한 것으로 볼 수 있지만, class1의 전체 개수가 200개도 되지 않기 때문에 10개를 더 맞췄다는 것은 비율로 했을 때, 전체에서 5% 더 맞췄다고 할 수 있다. 물론 class2 예측 성공률은 전에 비해 낮아진 것을 확인할 수 있지만, class1과 class3 예측을 전보다 더 잘한다는 점에서 더 밸런스 있는 예측을 한다고 할 수 있다. 결론적으로 데이터 불균형 해소를 위한 oversampling이 효과가 있었고, 그에 따라 성능 개선이 이루어졌다고 할 수 있다.

지금은 데이터 전처리 중에서도 간단한 oversampling 기법을 사용하여 성능 개선을 시도하였는데, 불균형 문제 해결을 위한 다양한 방법을 접목시킨다면 이보다 더 좋은 성능을 얻을 수 있을 것으로 예상된다.