상장기업
ESG점수 예측
& 군집별 특징 분석

F조: 강현우 김종민 김채은 김한빈 김형민 황다연

목차

Part 1.

아이디어 소개

I. 아이디어 소개

1. BlackRock

LARRY FINK'S 2020 LETTER TO CEOS

A Fundamental Reshaping of Finance

Dear CEO,

As an asset manager, BlackRock invests on behalf of others, and I am writing to you as an advisor and fiduciary to these clients. The money we manage is not our own. It belongs to people in dozens of countries trying to finance long-term goals like retirement. And we have a deep responsibility to these institutions and individuals – who are shareholders in your company and thousands of others – to promote long-term value.

BlackRock의 CEO Larry Fink

I. 아이디어 소개

II. ESG의 대중화

- 2020년부터 'ESG' 관련 대중 관심도 급증

I. 아이디어 소개

III. ESG란?

- 환경 (Environmental)
- 사회적 책임 (Social)
- 지배구조 (Governance)의 약자
- **기업의 중, 장기적 가치와 지속 가능성에** 큰 영향을 미칠 수 있는 비재무적 지표를 의미

I. 아이디어 소개

IV. ESG의 중요성: See Opportunity

- ESG로의 변화는 거부할 수 없는 흐름
- 두 가지의 Needs
  I. 투자자: 각 기업들의 ESG 준비 상태, 등급 등
  II. 기업: 현재 재무, 비재무 상태에 기반한 ESG 예상 등급

신재생연료 혼합의무화 비율, 2030년까지 '5%'

이필녀 기자 | 승인 2021.01.31 11:00 | 댓글 0

> 산업부, '신재생에너지법 시행령 개정안' 입법예고
> 현 3%에서 단계적 확대...3년 단위로 0.5%p 씩 상향

대기업 2025년부터 ESG 공시 의무화

강계만, 이유섭, 김규식 기자 | 입력 : 2021.01.14 18:00:24 수정 : 2021.01.14 23:27:27 0

1. 아이디어 소개

IV. ESG의 중요성: See Opportunity

- 한국 ESG 등급 산정의 실태는 굉장히 열악함

  I. 기관별로 ESG 등급 산정 Feature 및 가중치 설정이 상이함

  II. 기관별로 ESG 등급 산정 시 재무 정보와 비재무정보가 무차별적으로 섞여있어서, 등급 자체의 신뢰도가 떨어지는 편

  III. 상장 기업 중에서도 ESG 등급이 산정되지 않은 기업이 존재

Who's Good 지속가능발전소

KCGS 한국기업지배구조원

ESG Information, Analytics & Investment 서스틴베스트

1. 아이디어 소개

V. ESG 등급이 없는 기업들: Our Idea

- Our Idea

  I. ESG 등급이 산정되지 않은 기업의 현황을 개략적으로나마 파악할 수 없을까?

  II. ESG 평가와 직, 간접적으로 연관이 있는 Feature를 최대한 많이 고려한다면, Side Effect 까지 고려한 '예상 ESG 등급'을 제공할 수 있지 않을까?

- Our Item

  I. Information: ESG 등급이 산정되지 않은 기업의 '예상 등급'

  II. Unsupervised Learning: Cluster 제공

  III. Supervised Learning: Feature 별 Classification 적용
  ✓ ESG 등급을 가장 잘 설명하는 Feature가 무엇인지 밝혀내고
  ✓ 목적 기업의 예상 등급을 도출

Part 2.

데이터 소개

II. 데이터 소개

I. 데이터 개요

- 목적: 상장 기업들에 대한 정보를 최대한 많이 담고 있는 하나의 데이터셋을 만드는 것!

재무정보

I. 재무제표 데이터 <- 출처: Dart

II. 주가 데이터 <- 출처: Dart

비재무정보

I. 기존 ESG 점수 데이터 <- 출처: ESG 포털

II. 회사 복지 데이터 <- 출처: 사람인

III. 에너지 사용 데이터 <- 출처: NGMS 국가온실가스종합관리시스템

II. 데이터 소개

II. 재무 관련 데이터

- 재무제표, 주가 데이터
  - 회사의 재정적, 재무적 요소를 판단하기 위한 데이터

<class 'pandas.core.frame.DataFrame'>
RangeIndex: 2501 entries, 0 to 2500
Data columns (total 52 columns):

#	Column	Non-Null Count	Dtype
0	회사명	2500 non-null	object
1	종목코드	2500 non-null	float64
2	업종	2500 non-null	object
3	상장일	2500 non-null	object
4	지역	2500 non-null	object
5	영문 종목명	2500 non-null	object
6	시장구분	2500 non-null	object
7	상장주식수	2500 non-null	float64
8	ESG등급	955 non-null	object
9	환경	955 non-null	object
10	사회	955 non-null	object
11	지배구조	955 non-null	object
12	등락률	2380 non-null	float64
13	거래량_x	2380 non-null	float64
14	거래대금_x	2380 non-null	float64
15	온실가스 배출량(tCO2-eq)	326 non-null	float64
16	에너지 사용량(TJ)	326 non-null	float64
17	corp_code	231 non-null	float64
18	stock_name	231 non-null	object
19	corp_cls	231 non-null	object
20	induty_code	231 non-null	float64
21	est_dt	231 non-null	float64
22	2021년 NCSI	60 non-null	float64
23	자산총계	479 non-null	float64
24	자본금	479 non-null	float64
25	자본총계	479 non-null	float64
26	매출액	479 non-null	float64
27	영업이익	479 non-null	float64
28	순이익	479 non-null	float64
29	종가	2479 non-null	float64
30	시가총액	2479 non-null	float64
31	거래량_y	2479 non-null	float64
32	거래대금_y	2479 non-null	float64
33	통근버스운행	1735 non-null	float64
34	본인학자금	1735 non-null	float64
35	복지카드	1736 non-null	float64
36	업무활동비	1736 non-null	float64
37	자격증수당	1736 non-null	float64
38	위험수당	1736 non-null	float64
39	사대보험	1736 non-null	float64
40	워크샵	1736 non-null	float64
41	신입사원교육	1736 non-null	float64
42	직무향상교육	1736 non-null	float64
43	리더십교육	1736 non-null	float64
44	해외연수지원	1736 non-null	float64
45	도서비지원	1736 non-null	float64
46	외국어교육지원	1736 non-null	float64
47	자격증취득지원	1736 non-null	float64
48	총개수	1736 non-null	float64
49	기준가&종가 평균	2380 non-null	float64
50	업종 대분류	2500 non-null	object
51	온실가스/매출액	475 non-null	float64

dtypes: float64(39), object(13)
memory usage: 1016.2+ KB

Our Data Sets

#	Column	Non-Null Count	Dtype
0	corp_code	231 non-null	float64
1	stock_name	231 non-null	object
2	corp_cls	231 non-null	object
3	induty_code	231 non-null	float64
4	est_dt	231 non-null	float64
5	2021년 NCSI	60 non-null	float64
6	자산총계	479 non-null	float64
7	자본금	479 non-null	float64
8	자본총계	479 non-null	float64
9	매출액	479 non-null	float64
10	영업이익	479 non-null	float64
11	순이익	479 non-null	float64

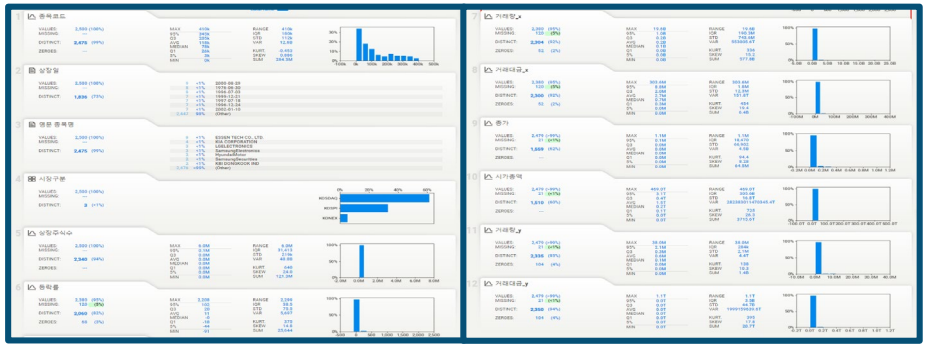
dtypes: float64(10), object(2)

재무제표 데이터

#	Column	Non-Null Count	Dtype
0	종목코드	2500 non-null	int64
1	상장일	2500 non-null	object
2	영문 종목명	2500 non-null	object
3	시장구분	2500 non-null	object
4	상장주식수	2500 non-null	int64
5	등락률	2380 non-null	float64
6	거래량_x	2380 non-null	float64
7	거래대금_x	2380 non-null	float64
8	종가	2479 non-null	float64
9	시가총액	2479 non-null	float64
10	거래량_y	2479 non-null	float64
11	거래대금_y	2479 non-null	float64

dtypes: float64(7), int64(2), object(3)

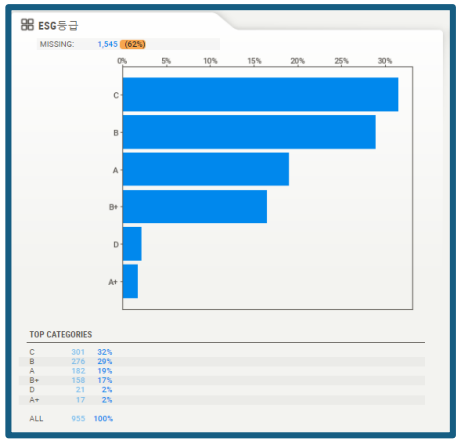
주가 데이터



II. 데이터 소개

III. 비재무 관련 데이터

- ESG 기관 점수 데이터
  - 기업의 E(환경), S(사회적책임), G(지배구조) 각각의 점수와 Total ESG 점수를 A+ ~ D 까지 Grade로 매긴 데이터



ESG 등급 데이터

환경 (E) 데이터

환경	
count	955
unique	6
top	D
freq	305

사회 (S) 데이터

사회	
count	955
unique	6
top	C
freq	300

지배구조 (G) 데이터

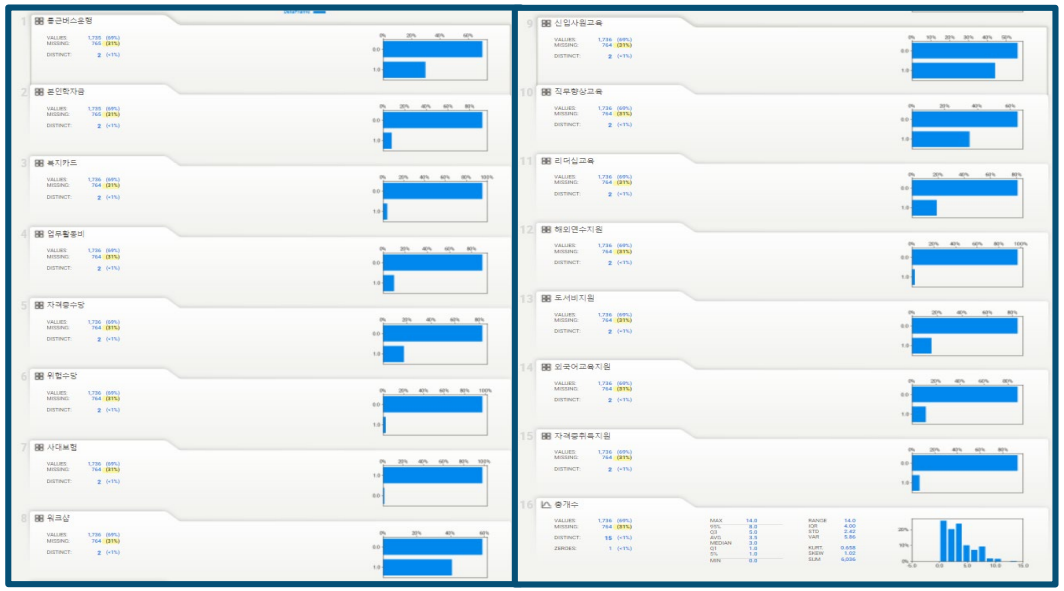
지배구조	
count	955
unique	6
top	B+
freq	399

II. 데이터 소개

III. 비재무 관련 데이터

- 회사 복지 데이터
  - 회사의 다양한 복지 요소 보유 여부에 관한 Binary 데이터
  - 통근버스운행 여부, 학자금 대출 가능 여부, 복지 카드 여부, 각종 교육 및 지원 여부 등으로 구성

#	Column	Non-Null Count	Dtype
0	통근버스운행	1735 non-null	float64
1	본인학자금	1735 non-null	float64
2	복지카드	1736 non-null	float64
3	업무활동비	1736 non-null	float64
4	자격증수당	1736 non-null	float64
5	위험수당	1736 non-null	float64
6	사대보험	1736 non-null	float64
7	워크샵	1736 non-null	float64
8	신입사원교육	1736 non-null	float64
9	직무향상교육	1736 non-null	float64
10	리더십교육	1736 non-null	float64
11	해외연수지원	1736 non-null	float64
12	도서비지원	1736 non-null	float64
13	외국어교육지원	1736 non-null	float64
14	자격증취득지원	1736 non-null	float64
15	총개수	1736 non-null	float64

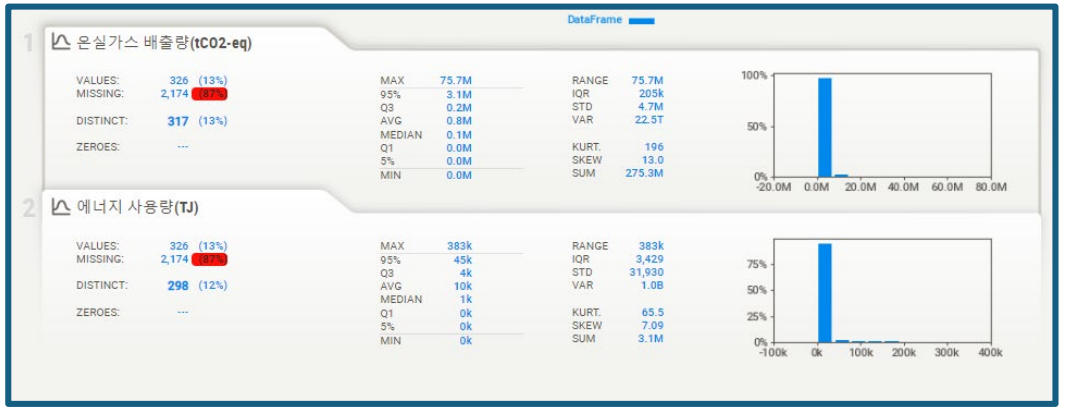


II. 데이터 소개

III. 비재무 관련 데이터

- 에너지 사용 데이터
  - 회사별 연간 온실가스 배출량, 에너지 사용량에 관한 데이터
  - 결측치가 다소 많이 존재한다는 한계

	온실가스 배출량(tCO2-eq)	에너지 사용량(TJ)
count	3.260000e+02	326.000000
mean	8.444162e+05	9551.957055
std	4.739584e+06	31929.664760
min	1.078000e+03	20.000000
25%	2.369100e+04	459.750000
50%	7.754750e+04	1379.500000
75%	2.290500e+05	3889.000000
max	7.567117e+07	382830.000000

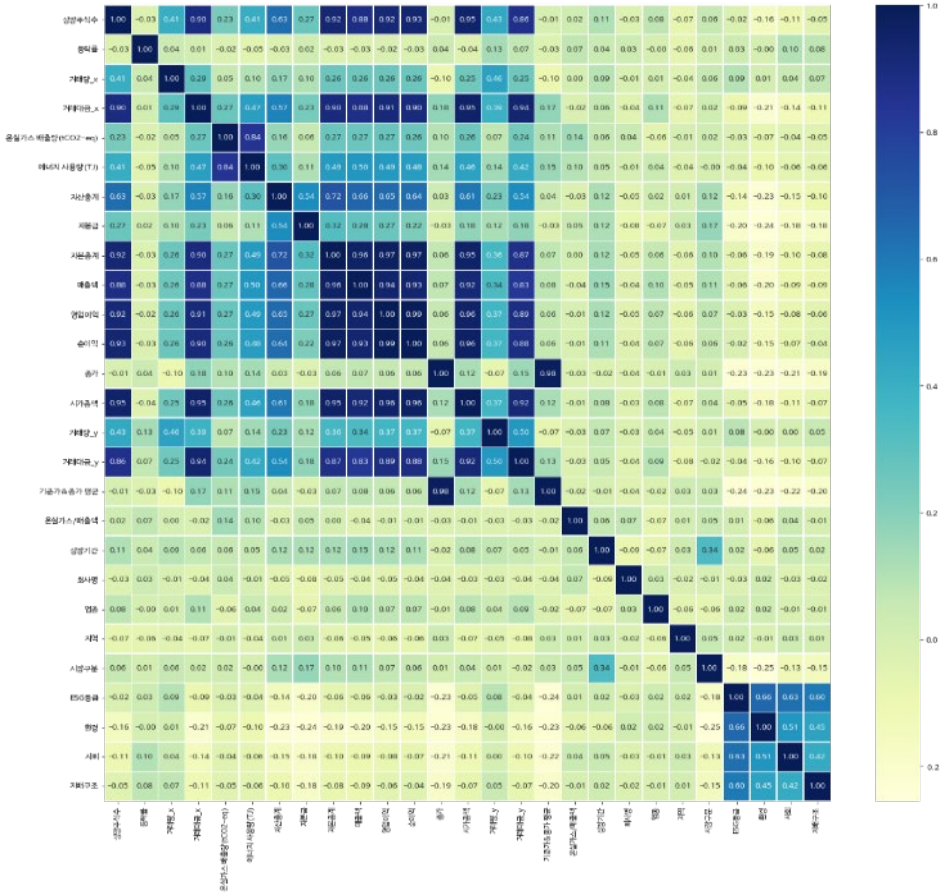


II. 데이터 소개

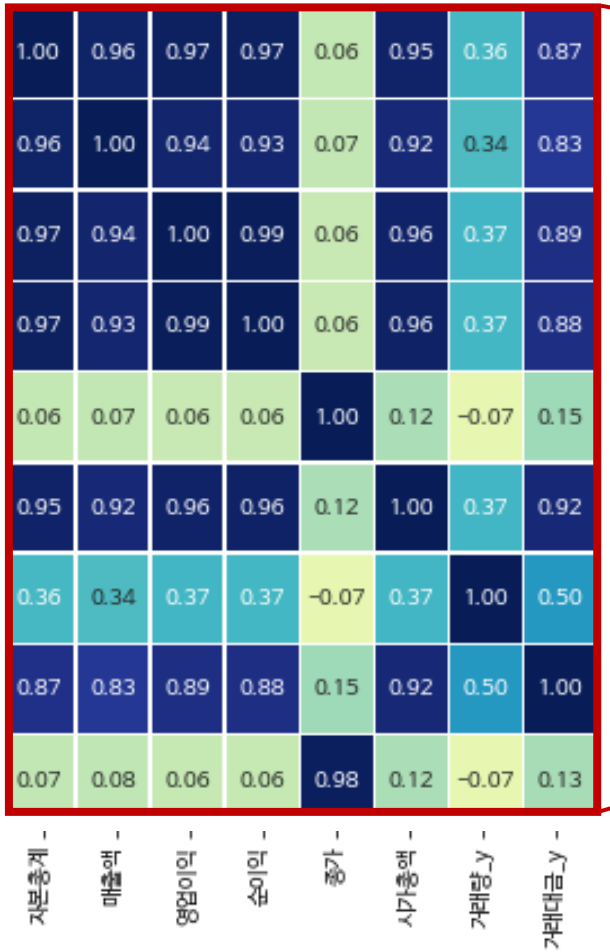
IV. Column별 상관관계

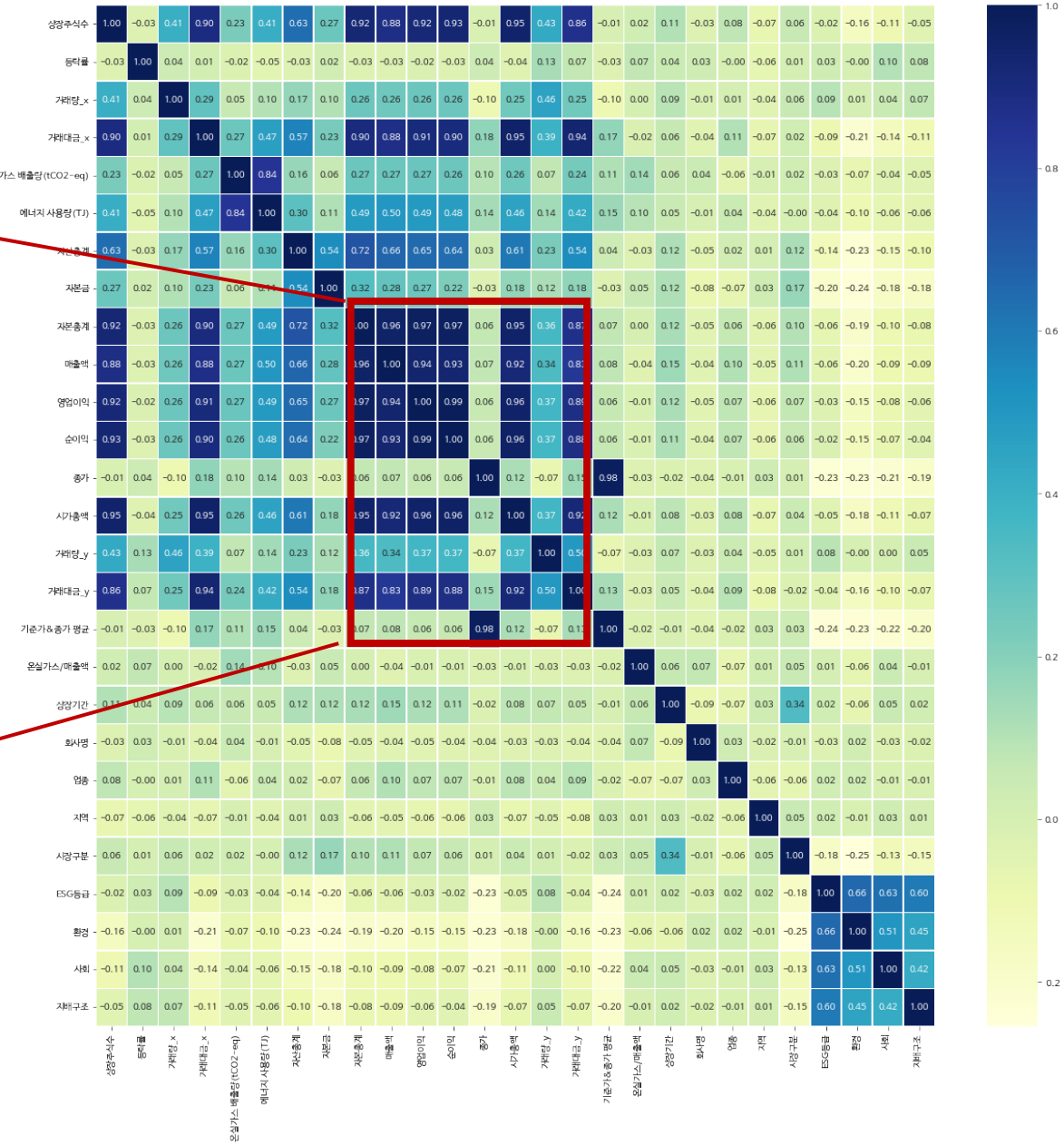
- Feature별 상관관계 그래프는 오른쪽과 같음
- 시가총액, 주가와 같은 재무 데이터는 서로 높은 상관관계 보임
- ESG점수 등급의 상관관계 또한 높게 나타남
- ESG점수 이외에 ESG점수와 두드러지게 상관관계가 높은 변수는 보이지 않음
- 최대한 많은 변수 클러스터링에 사용하기 위해 다중공선성 처리는 우선 하지 않음

Feature별 상관관계 그래프



Feature별 상관관계 그래프





Feature별 상관관계 그래프

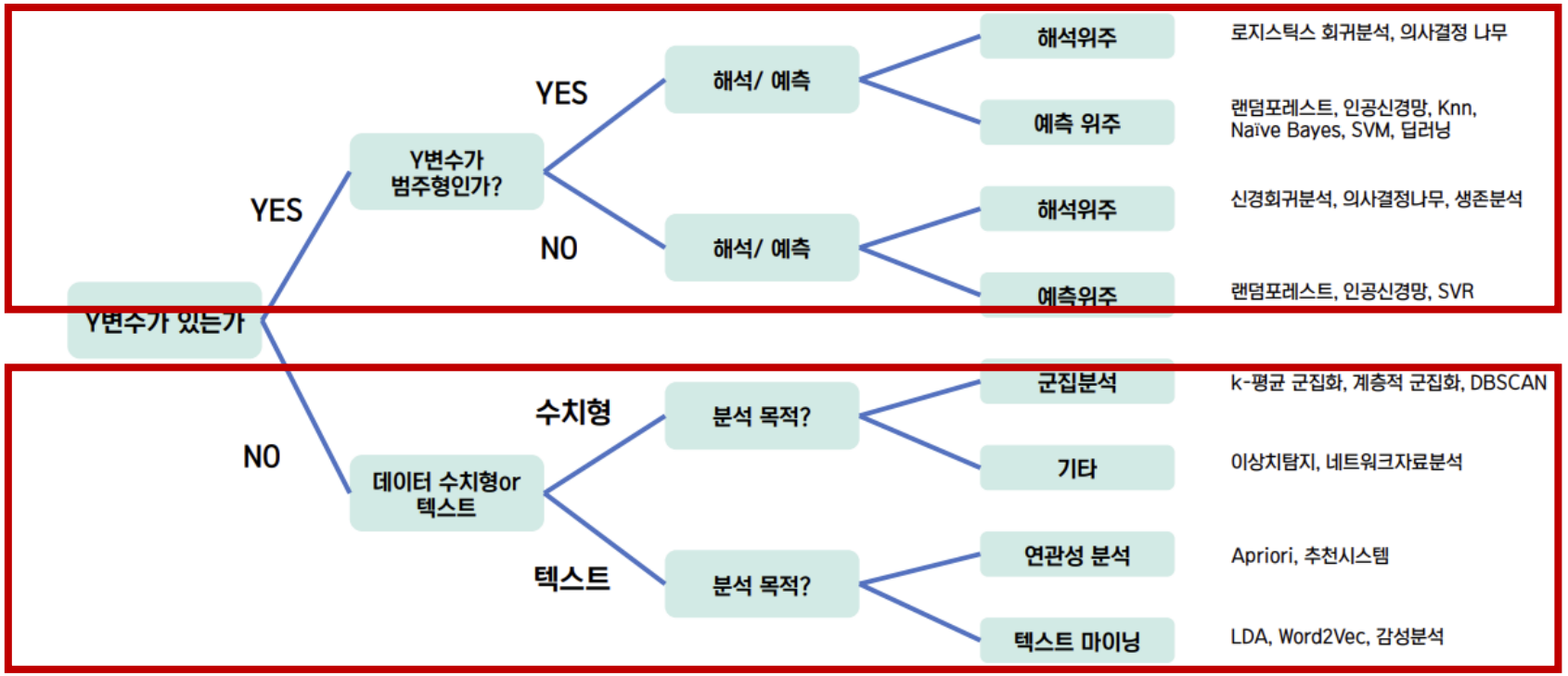
Part 3.

모델 구현

III. 모델 구현

I. 모델 소개

- Unsupervised Learning이란? 우리는 이걸로 Cluster를 구축할 것임. Why?
- Supervised Learning이란? 우리는 이걸로 두 가지 결론을 이끌어낼 것임. Why?



일반적인 경우 지도학습이 적절

확보 데이터의 특성 고려 시 비지도학습에 기반한 역산의 높은 성능 기대

III. 모델 구현

II. 데이터 전처리

1. Label Encoding (Original Encoding)

- 각 문자열 라벨을 단순 정수화 (A -> 0, B -> 1, C -> 2...)
- Decision Tree 모델에 적합하며, 특히 GBDT 모델에서는 기본적인 방법
- 클러스터링의 경우에는 범주형 자료를 숫자형으로 변환하지 않고 진행

	회사명	업종	지역	시장구분	ESG등급	환경	사회	지배구조
0	94	126	16	1	3	3	2	3
1	554	79	8	1	4	4	2	2
2	321	100	8	0	4	5	4	3
3	719	67	8	0	2	5	4	3
4	452	97	10	1	2	5	2	3
...	...	...	...	...	...	...	...	...
950	10	46	8	1	0	0	1	0
951	326	53	8	1	2	5	3	3
952	141	111	8	1	4	4	4	3
953	605	132	8	1	3	5	2	0
954	869	78	1	1	2	5	4	3

955 rows × 8 columns

Label Encoding 예시

*** 추가 고려한 방법론 및 한계**

I. **One-Hot인코딩** – 선형 모델 이외에 부적합 + cost 낭비 심함
II. **더미코딩** – 업종과 같은 feature수가 많아서 feature가 지나치게 비대해짐
III. **이펙트 코딩** – 더미코딩과 같은 문제
IV. **특징Hashing** – Hashing 이후 해석 불가
V. **빈도 인코딩**- 빈도가 같은 feature의 경우 데이터가 합쳐짐

III. 모델 구현

II. 데이터 전처리

2. SMOTE(Synthetic Minority Over-Sampling Technique) – 데이터 비대칭 문제 해결

- **비대칭 문제:** 데이터의 비율 차이가 큰 경우 단순히 우세한 클래스의 모형을 선택하는 모형의 정확도가 높아짐
- **SMOTE :** 낮은 비율로 존재하는 클래스의 데이터를 K-NN 알고리즘(최근접)을 활용하여 새롭게 생성하는 방법. 단순 무작위 추출은 overfitting 문제가 발생할 수도 있으나, SMOTE는 알고리즘에 기반해서 데이터를 생성하므로 과적합 발생 가능성이 상대적으로 작음

```
from imblearn.over_sampling import SMOTE
oversampling_instance = SMOTE(k_neighbors = 3)
```

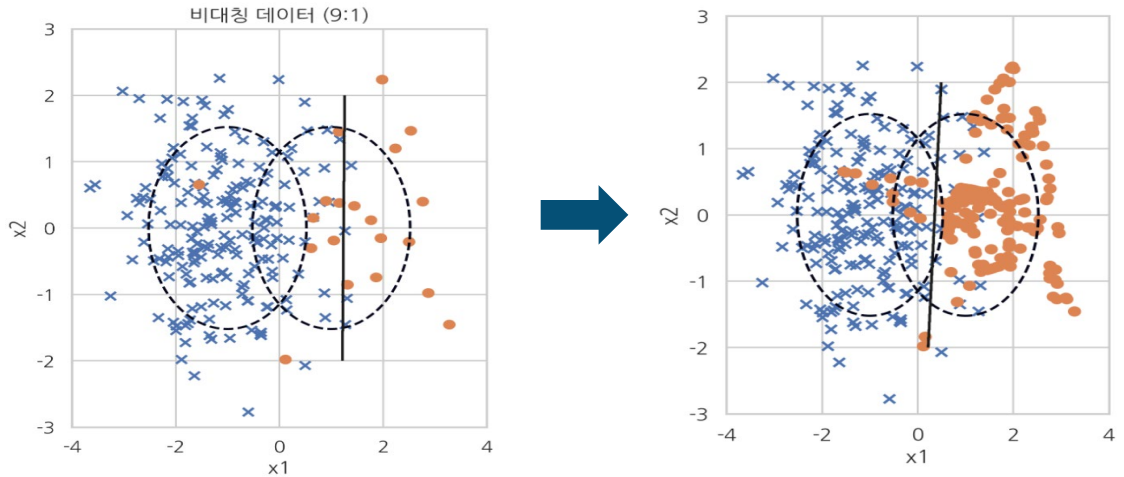
```
o_X, o_y = oversampling_instance.fit_resample(data_4_X, data_4_y)
```

```
data_4_y.value_counts()
```

```
ESG등급
4    301
2    276
0    182
3    158
5     21
1     17
dtype: int64
```

```
o_y.value_counts()
```

```
ESG등급
0    301
1    301
2    301
3    301
4    301
5    301
dtype: int64
```



III. 모델 구현

III. 모델 소개

1. Unsupervised Learning – K-Prototype Model

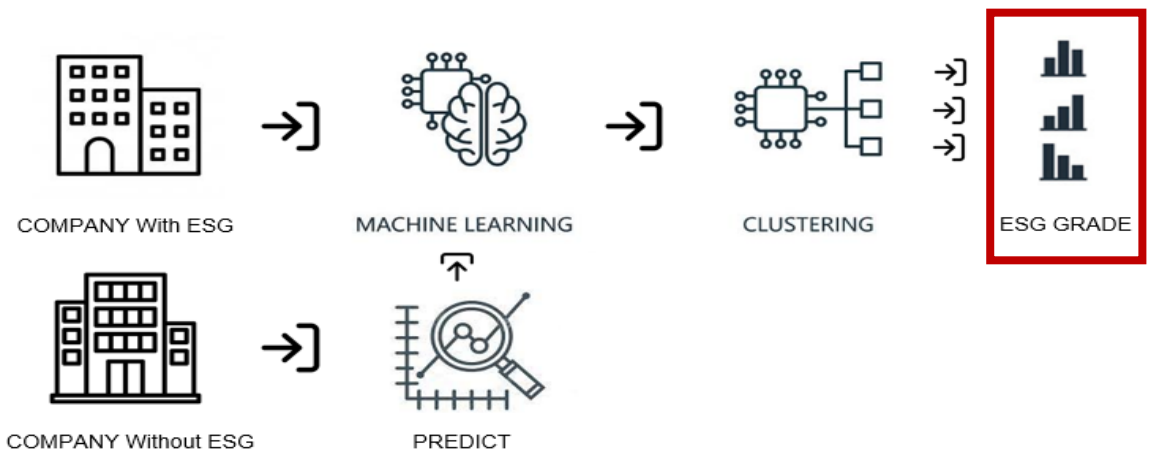
- K-Prototype Clustering은

  K-means(수치형 자료, 평균값 사용)

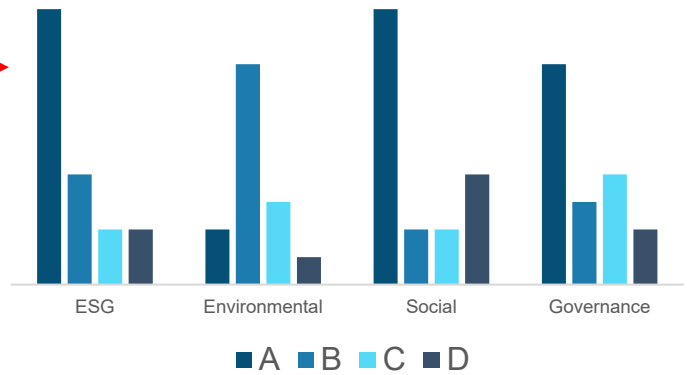
  K-modes(범주형 자료, 최빈값 사용)

  을 동시에 사용함으로써, 범주형 자료의 변환없이 클러스터링이 가능한 알고리즘

- 각 클러스터별로 ESG 등급, Environmental / Social / Governance **등급의 구분이 뚜렷하게 나오는 모델을 성공적인 모델**로 지칭



예시: 1st Cluster의 경우



III. 모델 구현

III. 모델 소개

2. Supervised Learning - Decision Tree, Random Forest, ADA Boost, Light GBM, Gradient Boosting, CAT Boosting, MLP (총 7개 Model)

- 비지도학습 기반의 클러스터링 모델은 기업이 속한 군집의 속성을 바탕으로 ESG 등급을 **확률**로서 예측함
- 지도학습에 기반한 Classification의 성능의 테스트, Y factor에 가장 큰 영향을 주는 X feature의 탐색을 위함
- F1 Score을 활용해 성능 평가
- ESG 등급을 제외한 모델의 지도학습으로 얻은 Label과 실제 Label을 비교하여 예측 성능 평가
- 변수 중요도 확인으로 가장 많은 영향을 끼치는 Feature의 재무적, 비재무적 요소 판단

Y factor
: ESG 등급

X features
X features
X features
X features
X features
X features

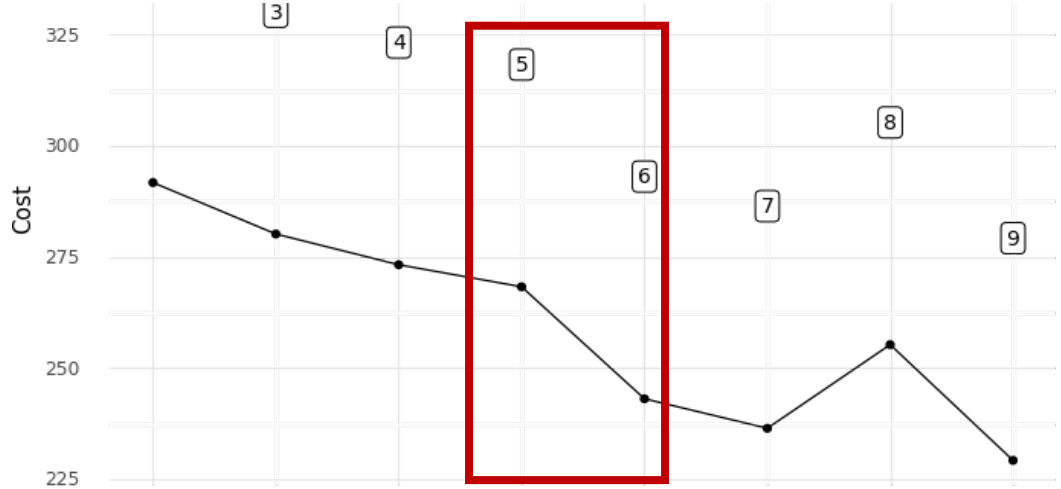
Part 4.

모델 해석

IV. 모델 해석

I. Unsupervised Learning – HyperParameter

- Elbow Method: **최적의 Cluster의 개수 'K'**를 찾는 방법
  - 클러스터의 갯수별 SSE를 계산하여, SSE가 가장 '급격하게 줄어드는 지점 ' 이 최적의 K
  - 임의의 군집 수를 정했을 때보다 Clustering이 더 잘 될 것으로 기대함



- 하이퍼파라미터의 종류
  - ✓ **Cluster의 개수: n_clusters**
  - ✓ 최대 반복 횟수: max_iter
  - ✓ 수치형 변수의 dissimilarity 함수: num_dissim
  - ✓ 범주형 변수의 dissimilarity 함수: cat_dissim
  - ✓ 클러스터당 반복 횟수: n_inint
  - ✓ 가중치: gamma
  - ✓ Verbosity mode: verbose
  - ✓ Job의 개수: n_jobs

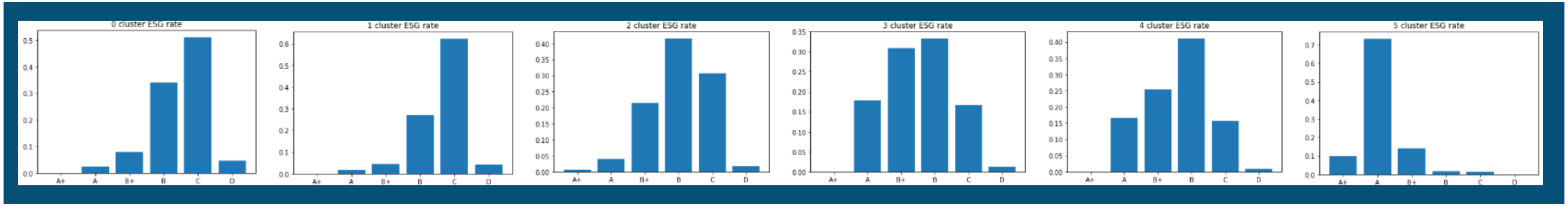
```
kprototype = KPrototypes(n_clusters=8, max_iter=100, num_dissim=euclidean_dissim, cat_dissim=matching_dissim,
                         init='Cao', n_init=10, gamma=None, verbose=0, random_state=None, n_jobs=1)
```

- **n_clusters** : cluster의 수
- **max_iter** : 최대 iteration 횟수
- num_dissim : numerical variable의 dissimilarity 함수 정의
- cat_dissim : categorical variable의 dissimilarity 함수 정의
- init : initialization 방법 정의 (Huang, Cao)
- **n_init** : cost에 따른 iteration당 클러스터링 반복 횟수 설정
- gamma : 가중치 설정
- verbose : verbosity mode
- n_jobs : computation을 위한 job의 수 결정

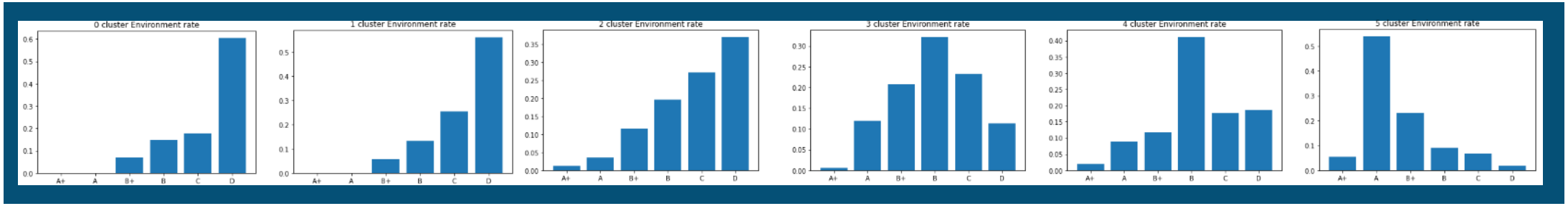
IV. 모델 해석

I. Unsupervised Learning – 결과 해석

- ESG 등급 클러스터 구분은 0, 1번 군집은 C등급, 5번 군집은 A등급이 우세했고,
  이외 군집은 분포 별 1,2,3순위로 설명 가능



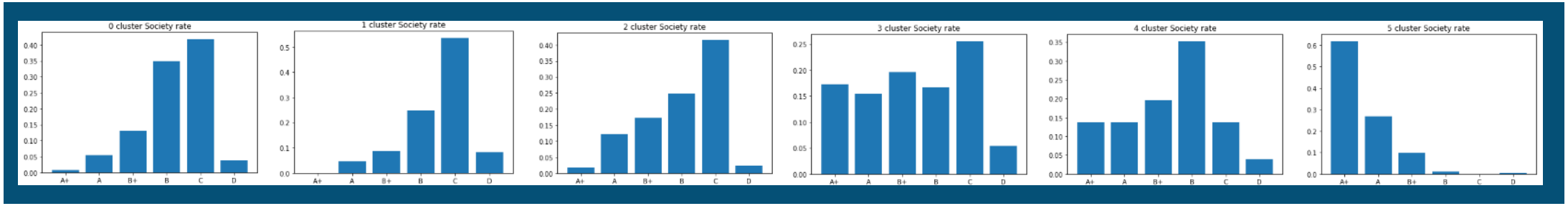
- Environment 클러스터 구분은 0, 1번 군집은 D등급, 5번 군집은 A등급이 우세했고,
  이외 군집은 분포 별 1,2,3순위로 설명 가능



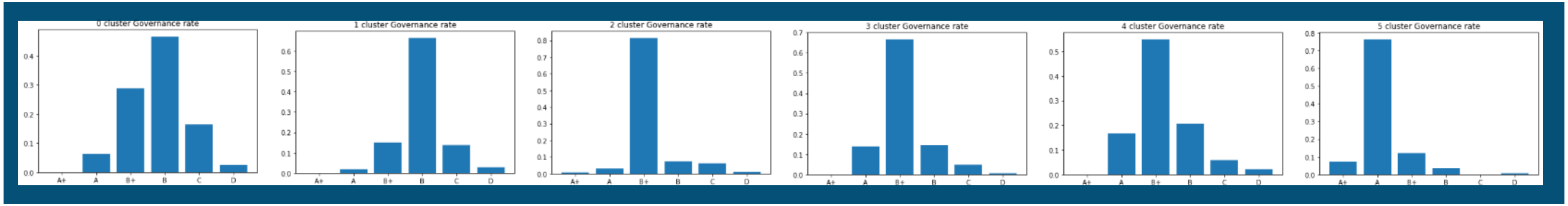
IV. 모델 해석

I. Unsupervised Learning – 결과 해석

- Society 등급 클러스터 구분은 1, 2번 군집은 C등급, 5번 군집은 A+등급이 우세하고,
  이외 군집은 분포 별 1,2,3순위로 설명 가능

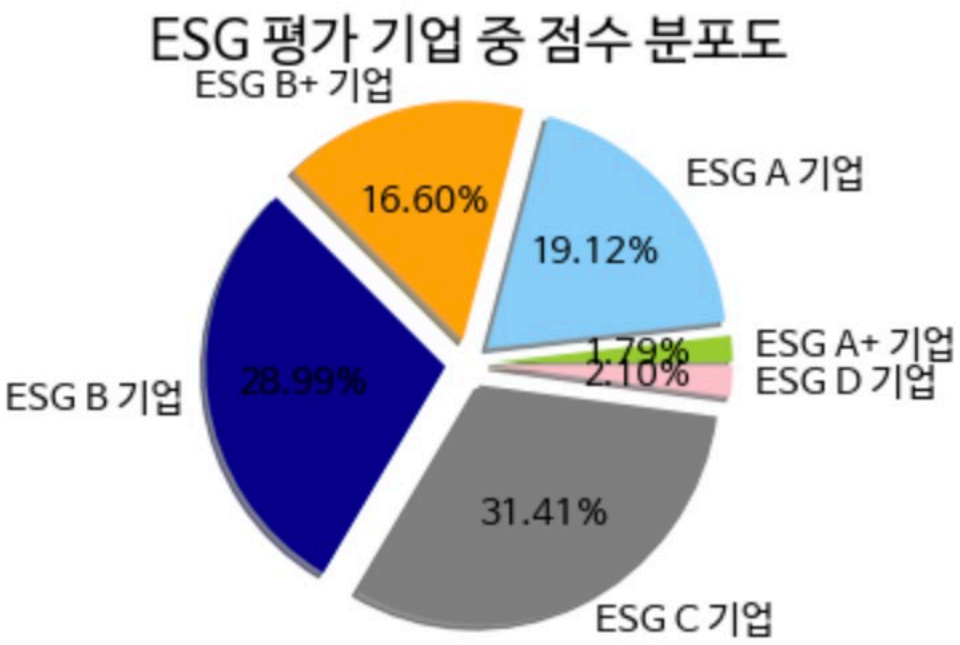


- Governance 클러스터 구분은 0, 1번 군집은 B등급, 2, 3, 4번 군집은 B+등급, 5번 군집은 A등급이 우세함



IV. 모델 해석

I. Unsupervised Learning – Oversampling



총 955개

Oversampling

A+
A
B+
B
C
D

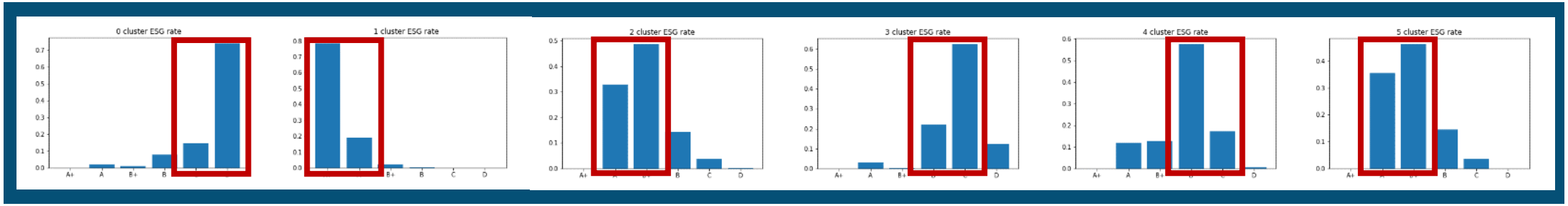
각 301개

각 등급당 기업 수가 **균등하게**
할당되도록 Sample 개수 조정

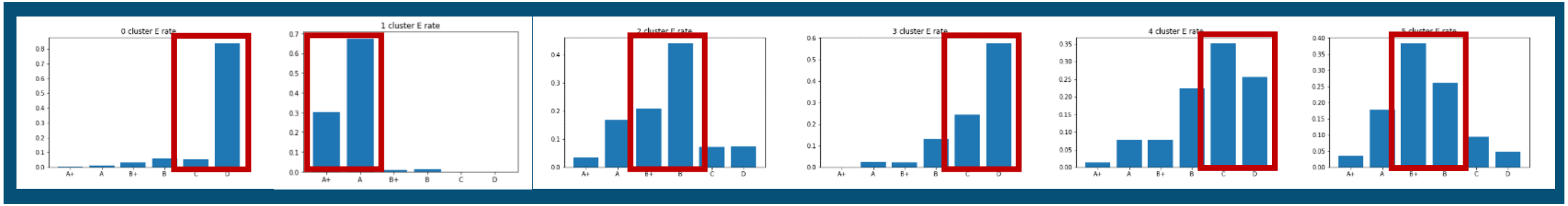
IV. 모델 해석

I. Unsupervised Learning – Oversampling

- Oversampling으로 구현한 ESG 등급 클러스터 구분에서 0번 군집은 D등급, 1번 군집은 A+등급,
  3번 군집은 C등급, 4번 등급은 B등급이 우세했고, 이외 2, 5번 군집은 분포 별 1,2순위로 설명 가능



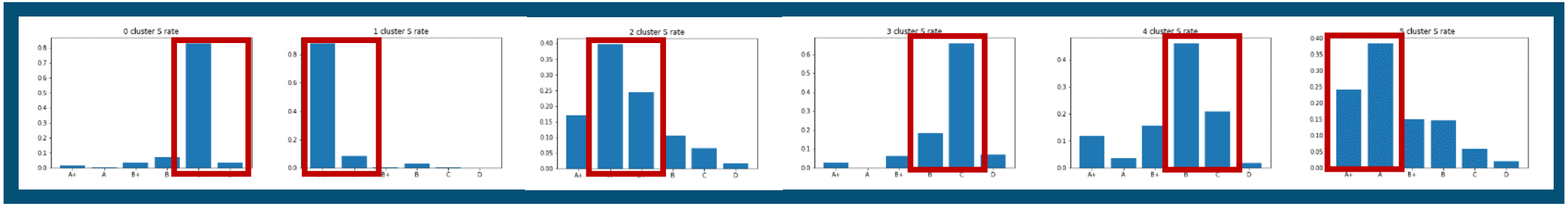
- Oversampling으로 구현한 Environment 클러스터 구분에서 0번 군집은 D등급, 1번 군집은 A등급,
  2번 군집은 B등급, 3번 군집은 D등급이 우세했고, 이외 4, 5번 군집은 분포 별 1,2,3순위로 설명 가능



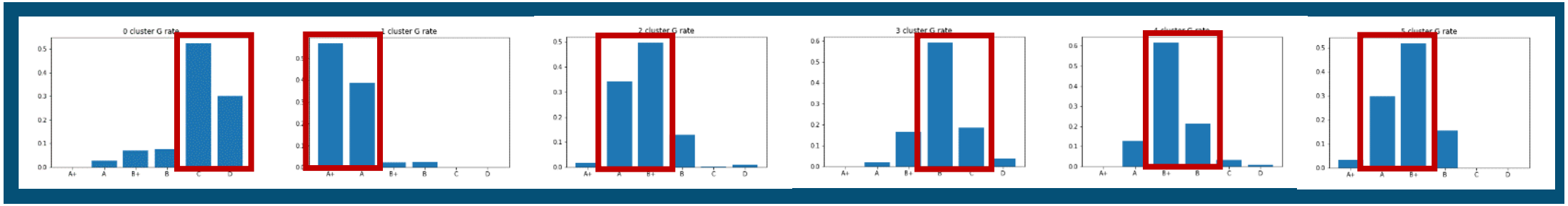
IV. 모델 해석

I. Unsupervised Learning – Oversampling

- Oversampling으로 구현한 Society 클러스터 구분에서 0번 군집은 C등급, 1번 군집은 A+등급,
  3번 군집은 C등급, 4번 등급은 B등급이 우세했고, 이외 2, 5번 군집은 분포 별 1,2순위로 설명 가능



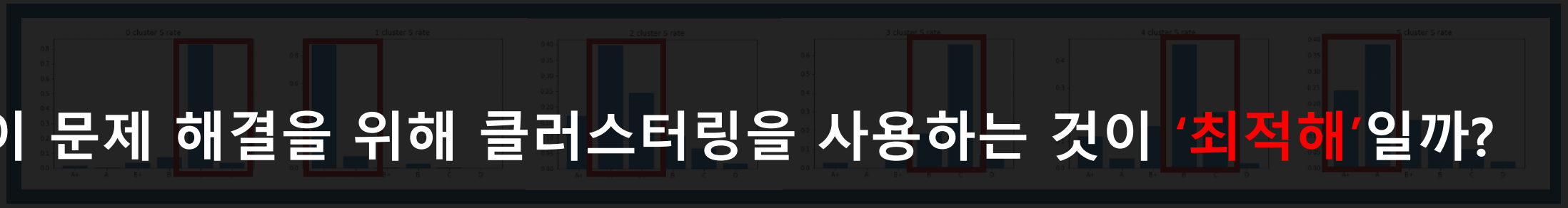
- Oversampling으로 구현한 Governance 클러스터 구분에서 0번 군집은 C등급, 3번 군집은 B등급,
  4번 군집은 B+등급이 우세했고, 이외 1, 2, 5번 군집은 분포 별 1,2순위로 설명 가능



IV. 모델 해석

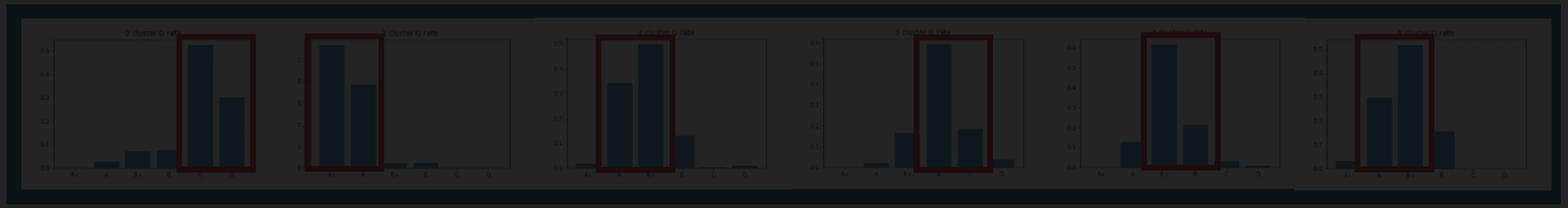
I. Unsupervised Learning – Oversampling

- Oversampling으로 구현한 Society 클러스터 구분에서 0번 군집은 C등급, 1번 군집은 A+등급, 3번 군집은 C등급, 4번 등급은 B등급이 우세했고, 이외 2, 5번 군집은 분포 별 1,2순위로 설명 가능



이 문제 해결을 위해 클러스터링을 사용하는 것이 '최적해'일까?

- Oversampling으로 구현한 Governance 클러스터 구분에서 0번 군집은 C등급, 3번 군집은 B등급, 4번 군집은 B+등급이 우세했고, 이외 1, 2, 5번 군집은 분포 별 1,2순위로 설명 가능



IV. 모델 해석

II. Supervised Learning

- Supervised Learning 모델을 사용할 수밖에 없는 이유

- 비지도학습 기반의 클러스터링 모델은 기업이 속한 군집의 속성을 바탕으로 ESG 등급을 **확률**로서 예측함

- 목적 기업의 예상 ESG 등급을 **분포도가 아닌 라벨값**으로 제시하기 위해서는 지도학습 기반의 모델 구현 필요

- ESG 등급을 목적변수로, 나머지 features들을 설명변수로 갖는 아래의 다양한 **지도학습 기반 모델**을 구현함

    - Decision Tree
    - Random Forest
    - Gradient Boosting
    - Light GBM
    - ADA Boost
    - CAT Boosting
    - MLP

IV. 모델 해석

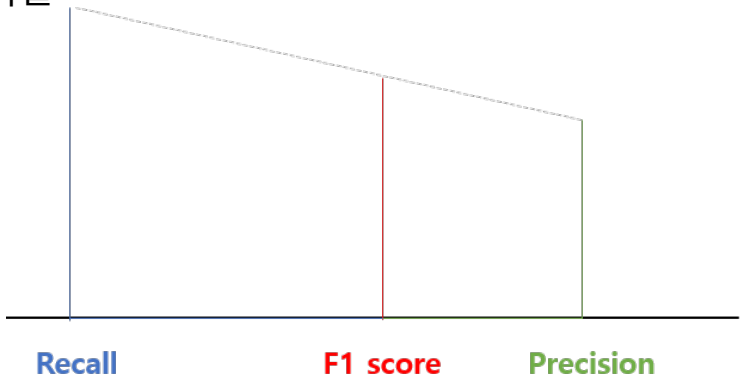
II. Supervised Learning – Decision Tree

- 하이퍼파라미터 Tuning을 이용해, 적절한 시점에 Tree 생성을 중단하여 훈련 데이터셋에 과적합되는 문제 방지

```
clf = DecisionTreeClassifier(random_state=0, criterion = 'entropy', max_depth = 31, min_samples_split = 24, min_samples_leaf = 11, max_leaf_nodes = 25)

# max_depth: 트리의 최대 depth
# min_samples_split: split하기 위해 노드가 가지고 있어야 하는 최소 샘플 개수
# min_samples_leaf: leaf node가 가져야 하는 최소 샘플 개수
# max_leaf_nodes: leaf node의 최대 개수
```

- F1 Score는 정밀도와 재현율의 조화평균값으로, 클래스 간 불균형이 존재할 때 Accuracy 대신 사용함
  I. Precision: 정밀도, Classifier가 참으로 분류한 결과 중, 실제 '참 ' 의 비율
  II. Recall: 재현율, 실제 참값 중에서 Classifier가 참으로 분류한 비율

$$2 * \frac{Precision * Recall}{Precision + Recall}$$



IV. 모델 해석

II. Supervised Learning – F1 Score 설명

- Accuracy가 높지만, Recall과 Precision이 낮은 경우가 존재할 수 있음 $2 * \frac{Precision * Recall}{Precision + Recall}$

	실제 양성	실제 음성
예측 양성	True Positive = 5	False Positive = 3
예측 음성	False Negative = 5	True Negative = 87

$$Accuracy = \frac{5+87}{5+3+5+87} = 0.92$$

$$Precision = \frac{5}{5+3} = 0.625 \quad Recall = \frac{5}{5+5} = 0.5$$

$$F_1 = 2 * \frac{0.5*0.625}{0.5+0.625} = 0.555$$

이런 경우 F1 스코어를 이용하여 해석

IV. 모델 해석

II. Supervised Learning – Decision Tree

- 가지치기 이전의 F1 Score

```
from sklearn.model_selection import cross_validate
import numpy as np

scoring = ['f1_macro']
tree_scores = cross_validate(tree_nottuned, X_train, y_ESG_train_pred, scoring=scoring, cv=5, return_train_score = False)
print("F1 Score of not-tuned tree : {:.3f}".format(np.mean(tree_scores['test_f1_macro'])))
```

```
F1 Score of not-tuned tree : 0.287
```

- Hyperparameter 임의 설정 후 사전 가지치기 수행 했을 때의 F1

```
scoring = ['f1_macro']
tree_scores = cross_validate(tree_prepruned, X_train, y_ESG_train_pred, scoring=scoring, cv=5, return_train_score = False)
print("F1 Score of pre-pruned tree (Hyperparameter 임의 설정) : {:.3f}".format(np.mean(tree_scores['test_f1_macro'])))
```

```
F1 Score of pre-pruned tree (Hyperparameter 임의 설정) : 0.253
```

IV. 모델 해석

II. Supervised Learning – Decision Tree

- 최적의 Hyperparameter를 찾는 방법: **RandomizedSearchCV**

```
from sklearn.model_selection import RandomizedSearchCV

dt = DecisionTreeClassifier()
dt.fit(X_train, y_ESG_train)

hyperparamters = {'max_depth': list(range(1, 50)),
                  'min_samples_split': list(range(1, 80)),
                  'min_samples_leaf': list(range(1, 100)),
                  'max_leaf_nodes': list(range(1, 100)),
                  'criterion': ['gini', 'entropy']}

# iteration 횟수 설정
n_iter = 1000

# RandomizedSearch 진행
rs = RandomizedSearchCV(dt,
                        param_distributions=hyperparamters,
                        n_iter = n_iter,
                        cv=5, verbose=3,
                        return_train_score = True)

rs.fit(X_train, y_ESG_train)
rs.best_params_ # 가장 좋은 성능을 보였을 때의 parameter
```

IV. 모델 해석

II. Supervised Learning – Decision Tree

- Hyperparameter Tuning 수행 후 모델 성능은 다음과 같음

```
tree_randomized = clf.fit(X_train, y_ESG_train)
draw_decision_tree(tree_randomized, feature_names, class_name_ESG)

tree_tuned_scores = cross_validate(tree_randomized, X_train, y_ESG_train, scoring=scoring, cv=5, return_train_score = False)
print("F1 Score of pre-pruned tree after hyperparameter tuning (train set) : {:.3f}".format(np.mean(tree_tuned_scores['test_f1_macro']))) # train set

tree_tuned_scores = cross_validate(tree_randomized, X_test, y_ESG_test, scoring=scoring, cv=5, return_train_score = False)
print("F1 Score of pre-pruned tree after hyperparameter tuning (test set) : {:.3f}".format(np.mean(tree_tuned_scores['test_f1_macro']))) # test set
```

```
F1 Score of pre-pruned tree after hyperparameter tuning (train set) : 0.263
F1 Score of pre-pruned tree after hyperparameter tuning (test set) : 0.225
```

- Data Oversampling 후 모델 성능은 다음과 같음

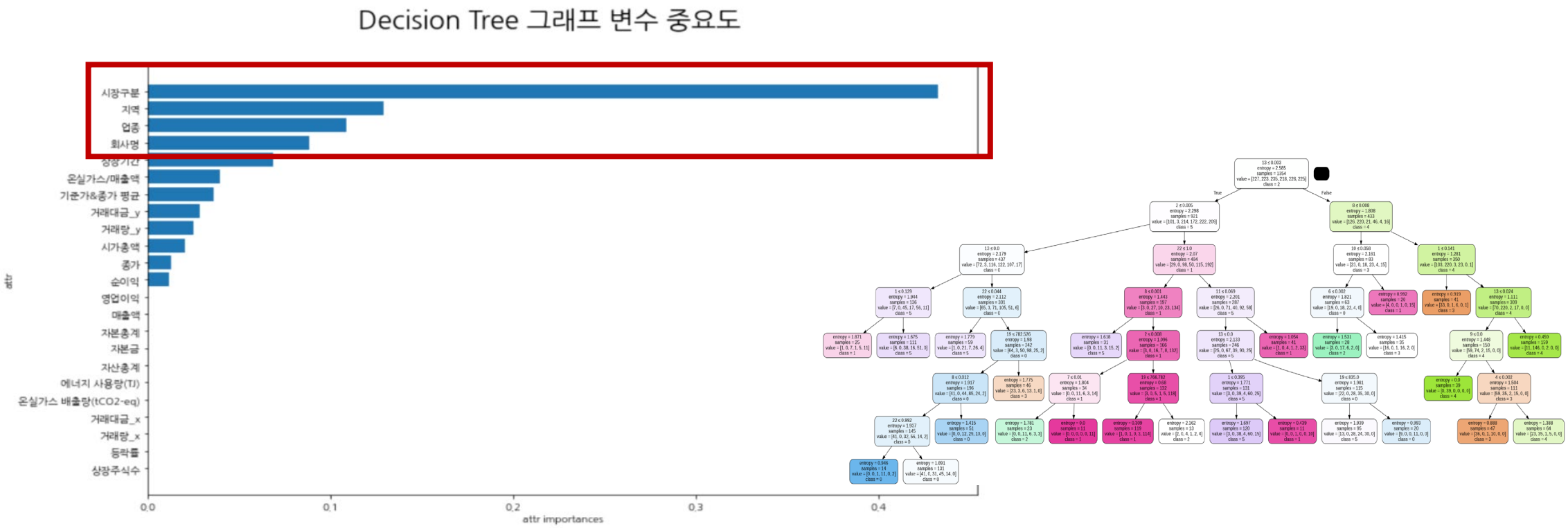
```
tree_tuned_scores = cross_validate(tree_oversampling, o_X_train, o_y_ESG_train, scoring=scoring, cv=5, return_train_score = False)
print("Tree parameter after Oversampling - F1 Score (train set): {:.3f}".format(np.mean(tree_tuned_scores['test_f1_macro']))) # train set

tree_tuned_scores = cross_validate(tree_oversampling, o_X_test, o_y_ESG_test, scoring=scoring, cv=5, return_train_score = False)
print("Tree parameter after Oversampling - F1 Score (test set) : {:.3f}".format(np.mean(tree_tuned_scores['test_f1_macro']))) # test set
```

```
Tree parameter after Oversampling - F1 Score (train set): 0.545
Tree parameter after Oversampling - F1 Score (test set) : 0.494
```

IV. 모델 해석

II. Supervised Learning – Decision Tree

- 시장구분, 지역, 업종이 ESG등급을 잘 설명하는 중요 Feature (시장구분, 지역, 업종 순)
- 아직까지는 ESG 등급이 기업의 특성보다는 업종별 특성의 영향을 많이 받고 있다고 유추할 수 있음



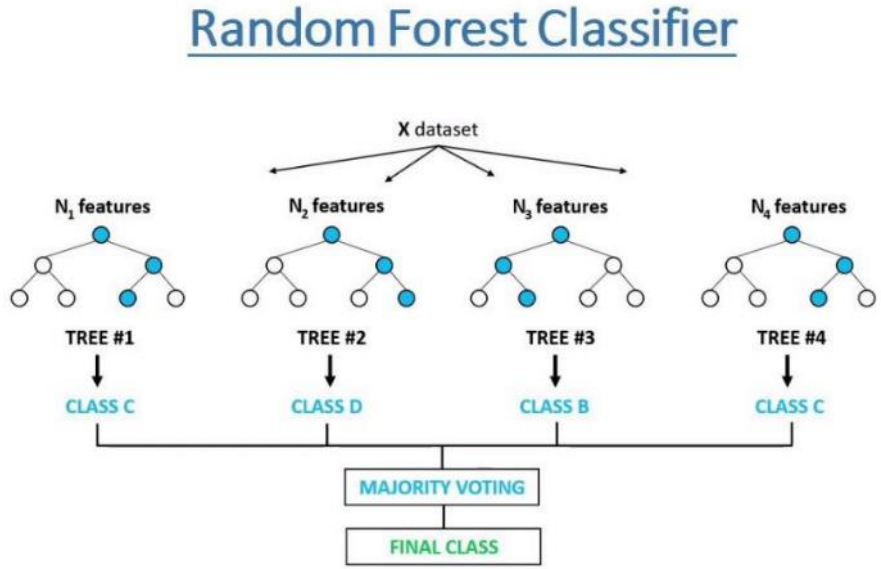
IV. 모델 해석

II. Supervised Learning – Random Forest

• 여러 개의 Decision Tree 분류기가 개별적으로 학습 수행 후 최종적으로 모든 분류기가 voting을 통해 예측 결정

• 개별적인 분류기는 Decision Tree 기반 알고리즘이지만, 개별 Tree가 학습하는 데이터셋은 Bootstrapping 방식으로 일부 데이터가 중첩되게 샘플링된 데이터셋

```
[ ]  # train
     rf_scores_train = cross_validate(rf_clf,X_train, y_ESG_train, scoring=scoring, cv=5, return_train_score = False)
     print("(train set) rf_score before Oversampling - F1 Score: {:.3f}".format(np.mean(rf_scores_train['test_f1_macro'])))

     # test
     rf_scores_test = cross_validate(rf_clf, X_test, y_ESG_test, scoring=scoring, cv=5, return_train_score = False)
     print("(test set) rf_score before Oversampling - F1 Score: {:.3f}".format(np.mean(rf_scores_test['test_f1_macro'])))

     ################## OverSampling ###################################
     # train
     rf_scores_otrain = cross_validate(rf_clf, o_X_train, o_y_ESG_train, scoring=scoring, cv=5, return_train_score = False)
     print("(train set) rf_score after Oversampling - F1 Score: {:.3f}".format(np.mean(rf_scores_otrain['test_f1_macro'])))

     # test
     rf_scores_otest = cross_validate(rf_clf, o_X_test, o_y_ESG_test, scoring=scoring, cv=5, return_train_score = False)
     print("(test set) rf_score after Oversampling - F1 Score: {:.3f}".format(np.mean(rf_scores_otest['test_f1_macro'])))

     (train set) rf_score before Oversampling - F1 Score: 0.376
     (test set) rf_score before Oversampling - F1 Score: 0.362
     (train set) rf_score after Oversampling - F1 Score: 0.731
     (test set) rf_score after Oversampling - F1 Score: 0.614
```



IV. 모델 해석

II. Supervised Learning – Random Forest

- 최적의 하이퍼 파라미터를 찾기 위해 Grid search 2시간 이상 실행했으나 완료되지 않아 종료
- 해결을 위해 RandomizeSearch의 iteration을 늘려서 실행
- But, 일반 모델의 성능이 좋아서 이후 일괄적으로 일반 모델로 적용
- 본 프로젝트의 선택 모델이 클러스터링이기 때문에 하이퍼 파라미터 튜닝 부족은 한계점으로 남겨두기로 함

```
[ ]  from sklearn.ensemble import RandomForestClassifier

     # rf_clf = RandomForestClassifier(criterion = 'entropy', max_depth = 34, max_leaf_nodes = 47, min_samples_leaf = 18, min_samples_split = 38,  random_state
     rf_clf = RandomForestClassifier(random_state = 42)
     # 개별 분류기에 train set 피팅
     rf_clf.fit(X_train, y_ESG_train)

     # train셋으로 prediction
     rf_pred = rf_clf.predict(X_train)

     # 성능 확인
     # accuracy_score 계산
     # print("parameter가 tuning 되지 않았을 때의 Accuracy: {:.3f}".format(accuracy_score(rf_pred,y_ESG_test)))


     # f-1 score 계산
     from sklearn.model_selection import cross_validate
     import numpy as np

     scoring = ['f1_macro']
     rf_scores = cross_validate(rf_clf, X_train, y_ESG_train, scoring=scoring, cv=5, return_train_score = False)
     print("parameter가 tuning 되지 않았을 때의 F1 Score : {:.3f}".format(np.mean(rf_scores['test_f1_macro'])))

     parameter가 tuning 되지 않았을 때의 F1 Score : 0.376
```

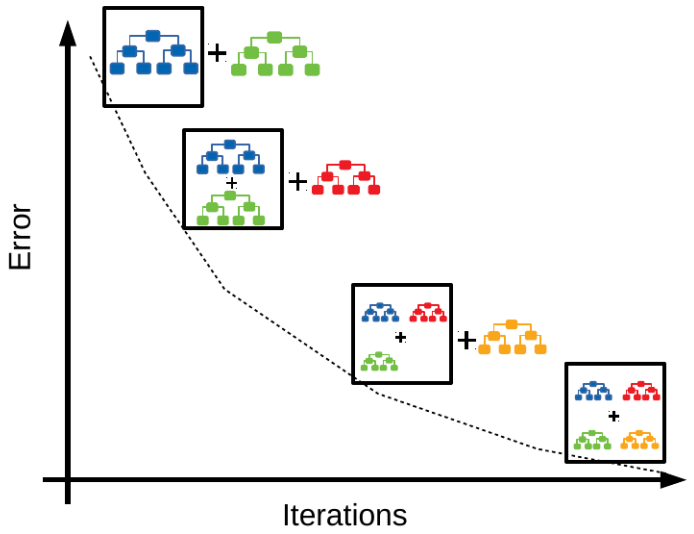
```
from sklearn.model_selection import RandomizedSearchCV
from sklearn.ensemble import RandomForestClassifier

# random forest
rf_clf = RandomForestClassifier(min_samples_split = 2, random_state = 42)

# 개별 분류기에 train set 피팅
rf_clf.fit(X_train, y_ESG_train)

# train셋으로 prediction
rf_pred = rf_clf.predict(X_train)

hyperparamters = {'max_depth': list(range(1, 50)),
                  'min_samples_split': list(range(1, 50)),
                  'min_samples_leaf': list(range(1, 500)),
                  'max_leaf_nodes': list(range(1, 50)),
                  'criterion': ['gini', 'entropy']}

# iteration 횟수를 정해주자!!!!!!!!
n_iter = 1000


# RandomizedSearch 진행
rs = RandomizedSearchCV(rf_clf,
                        param_distributions=hyperparamters,
                        n_iter = n_iter,
                        cv=5, verbose=3,
                        return_train_score = True)

rs.fit(X_train, y_ESG_train)
rs.best_params_ # 가장 좋은 성능을 보였을 때의 parameter
```

IV. 모델 해석

II. Supervised Learning – Gradient Boosting

- 여러 개의 결정 트리를 묶어 강력한 모델을 만드는 앙상블 방법으로, '회귀'와 '분류'에 모두 사용 가능
- Random Forest와 달리, 이전 트리의 오차를 보완하는 방식을 이용해 순차적으로 트리 생성
- Random Forest에 비해 매개변수 설정에 조금 더 민감하지만, 더 높은 정확도를 가짐

```
[ ]  # train
     gb_scores_train = cross_validate(gb_clf,X_train, y_ESG_train, scoring=scoring, cv=5, return_train_score = False)
     print("(train set) gb_score before Oversampling - F1 Score: {:.3f}".format(np.mean(gb_scores_train['test_f1_macro'])))

     # test
     gb_scores_test = cross_validate(gb_clf, X_test, y_ESG_test, scoring=scoring, cv=5, return_train_score = False)
     print("(test set) gb_score before Oversampling - F1 Score: {:.3f}".format(np.mean(gb_scores_test['test_f1_macro'])))

     ################## OverSampling ###################################

     # train
     gb_scores_otrain = cross_validate(gb_clf, o_X_train, o_y_ESG_train, scoring=scoring, cv=5, return_train_score = False)
     print("(train set) gb_score after Oversampling - F1 Score: {:.3f}".format(np.mean(gb_scores_otrain['test_f1_macro'])))

     # test
     gb_scores_otest = cross_validate(gb_clf, o_X_test, o_y_ESG_test, scoring=scoring, cv=5, return_train_score = False)
     print("(test set) gb_score after Oversampling - F1 Score: {:.3f}".format(np.mean(gb_scores_otest['test_f1_macro'])))

     (train set) gb_score before Oversampling - F1 Score: 0.371
     (test set) gb_score before Oversampling - F1 Score: 0.323
     (train set) gb_score after Oversampling - F1 Score: 0.700
     (test set) gb_score after Oversampling - F1 Score: 0.615
```

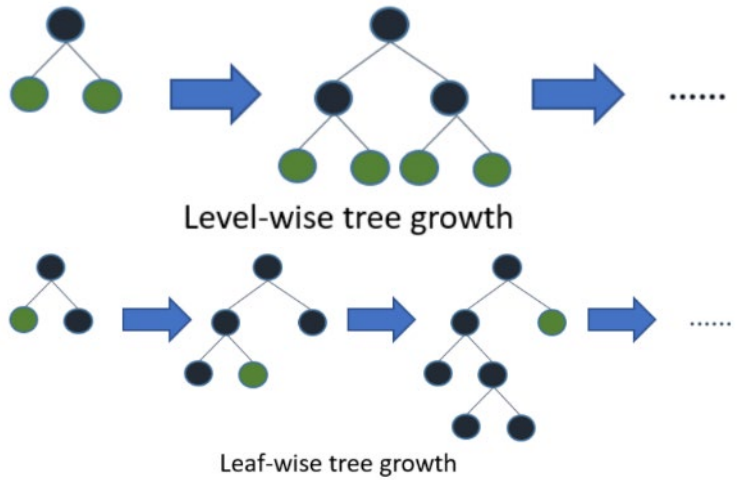


IV. 모델 해석

II. Supervised Learning – Light GBM

- Gradient Boosting 기반의 알고리즘으로, 학습 시간이 느린 XGBoost의 단점을 보완하여 속도와 성능이 개선된 알고리즘
- Gradient-based One-Side Sampling을 적용하여 Information gain을 계산할 때 큰 Gradients를 가진 데이터 인스턴스는 유지하고 작은 Gradients를 가진 데이터 인스턴스는 무작위로 샘플링
- leaf-wise (리프 중심 트리 분할) 방식을 채택하여 트리가 깊어지면서 소요되는 시간과 메모리를 절약함

```
[ ]  # train
     LGBM_scores_train = cross_validate(lgbm_clf,X_train, y_ESG_train, scoring=scoring, cv=5, return_train_score = False)
     print("(train set) LGBM_score before Oversampling - F1 Score: {:.3f}".format(np.mean(LGBM_scores_train['test_f1_macro'])))

     # test
     LGBM_scores_test = cross_validate(lgbm_clf, X_test, y_ESG_test, scoring=scoring, cv=5, return_train_score = False)
     print("(test set) LGBM_score before Oversampling - F1 Score: {:.3f}".format(np.mean(LGBM_scores_test['test_f1_macro'])))

     ################## OverSampling ###################################

     # train
     LGBM_scores_otrain = cross_validate(lgbm_clf, o_X_train, o_y_ESG_train, scoring=scoring, cv=5, return_train_score = False)
     print("(train set) LGBM_score after Oversampling - F1 Score: {:.3f}".format(np.mean(LGBM_scores_otrain['test_f1_macro'])))

     # test
     LGBM_scores_otest = cross_validate(lgbm_clf, o_X_test, o_y_ESG_test, scoring=scoring, cv=5, return_train_score = False)
     print("(test set) LGBM_score after Oversampling - F1 Score: {:.3f}".format(np.mean(LGBM_scores_otest['test_f1_macro'])))

     (train set) LGBM_score before Oversampling - F1 Score: 0.387
     (test set) LGBM_score before Oversampling - F1 Score: 0.371
     (train set) LGBM_score after Oversampling - F1 Score: 0.727
     (test set) LGBM_score after Oversampling - F1 Score: 0.621
```

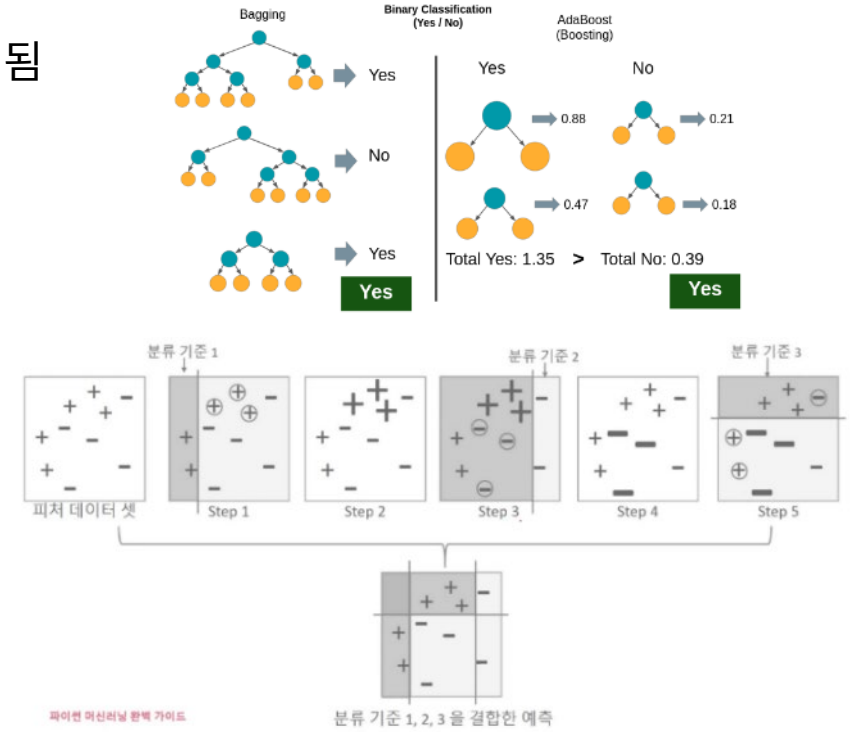


IV. 모델 해석

II. Supervised Learning – ADA Boost

- 약한 분류기들이 상호보완 하도록 순차적으로 학습하고, 이들을 조합하여 최종적으로 강한 분류기의 성능을 향상시킴
- 이전 분류기가 잘못 분류한 샘플의 가중치를 상호보완적(adaptive)으로 바꿔가며 잘못 분류되는 데이터에 더 집중하며 학습해나감
- 약한 분류기들을 조합하여 하나의 분류기를 만들기 때문에 boosting이 됨

```
[ ]  # train
     Ab_scores_train = cross_validate(Ab_clf,X_train, y_ESG_train, scoring=scoring, cv=5, return_train_score = False)
     print("(train set) ADA_score before Oversampling - F1 Score: {:.3f}".format(np.mean(Ab_scores_train['test_f1_macro'])))

     # test
     Ab_scores_test = cross_validate(Ab_clf, X_test, y_ESG_test, scoring=scoring, cv=5, return_train_score = False)
     print("(test set) ADA_score before Oversampling - F1 Score: {:.3f}".format(np.mean(Ab_scores_test['test_f1_macro'])))

     ################## OverSampling ###################################

     # train
     Ab_scores_otrain = cross_validate(Ab_clf, o_X_train, o_y_ESG_train, scoring=scoring, cv=5, return_train_score = False)
     print("(train set) ADA_score after Oversampling - F1 Score: {:.3f}".format(np.mean(Ab_scores_otrain['test_f1_macro'])))

     # test
     Ab_scores_otest = cross_validate(Ab_clf, o_X_test, o_y_ESG_test, scoring=scoring, cv=5, return_train_score = False)
     print("(test set) ADA_score after Oversampling - F1 Score: {:.3f}".format(np.mean(Ab_scores_otest['test_f1_macro'])))

     (train set) ADA_score before Oversampling - F1 Score: 0.190
     (test set) ADA_score before Oversampling - F1 Score: 0.222
     (train set) ADA_score after Oversampling - F1 Score: 0.396
     (test set) ADA_score after Oversampling - F1 Score: 0.231
```



IV. 모델 해석

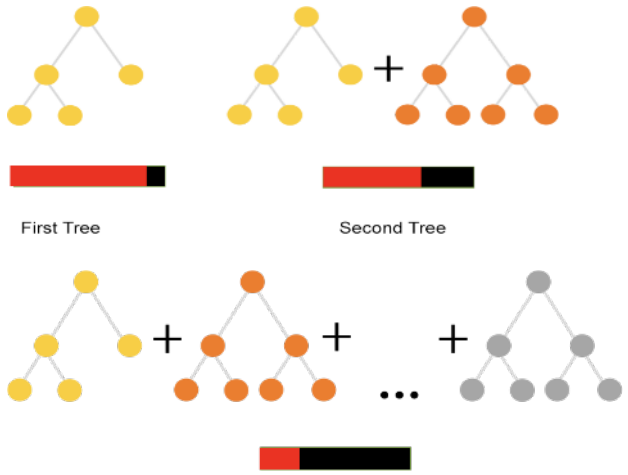
II. Supervised Learning – CAT Boosting

- 의사결정 트리 라이브러리를 강화한 Gradient Boosting 모델
- 하이퍼파라미터 튜닝 부담을 줄여주며, CPU 및 GPU 성능 향상을 지원
- Decision Tree 구성 시 Train set과 Test set간 관측치 오차 및 Unbiased Residual을 구하고 가중치를 업데이트하는 방식으로 과적합 문제를 줄여줌
- 범주를 다른 수준(ex. 숫자)으로 치환하지 않아도 되는 비수치 범주화 기능을 지원

```
4:    learn: 1.5927646    total: 142ms    remaining: 28.3s    993:    learn: 0.0415507    total: 24.8s    remaining: 150ms
5:    learn: 1.5624503    total: 167ms    remaining: 27.6s    994:    learn: 0.0415024    total: 24.9s    remaining: 125ms
6:    learn: 1.5303382    total: 190ms    remaining: 27s      995:    learn: 0.0414727    total: 24.9s    remaining: 99.9ms
7:    learn: 1.4958669    total: 215ms    remaining: 26.6s    996:    learn: 0.0414146    total: 24.9s    remaining: 74.9ms
8:    learn: 1.4697197    total: 239ms    remaining: 26.3s    997:    learn: 0.0413391    total: 24.9s    remaining: 50ms
9:    learn: 1.4439904    total: 268ms    remaining: 26.5s    998:    learn: 0.0412870    total: 25s      remaining: 25ms
10:   learn: 1.4176825    total: 293ms    remaining: 26.3s    999:    learn: 0.0412165    total: 25s      remaining: 0us
```

```
[ ]  # train
     print("(train set) CAT_score after Oversampling - F1 Score: {:.3f}".format(np.mean(CAT_scores_train['test_f1_macro'])))

     # test
     print("(test set) CAT_score after Oversampling - F1 Score: {:.3f}".format(np.mean(CAT_scores_test['test_f1_macro'])))
```

```
(train set) CAT_score after Oversampling - F1 Score: 0.720
(test set) CAT_score after Oversampling - F1 Score: 0.634
```



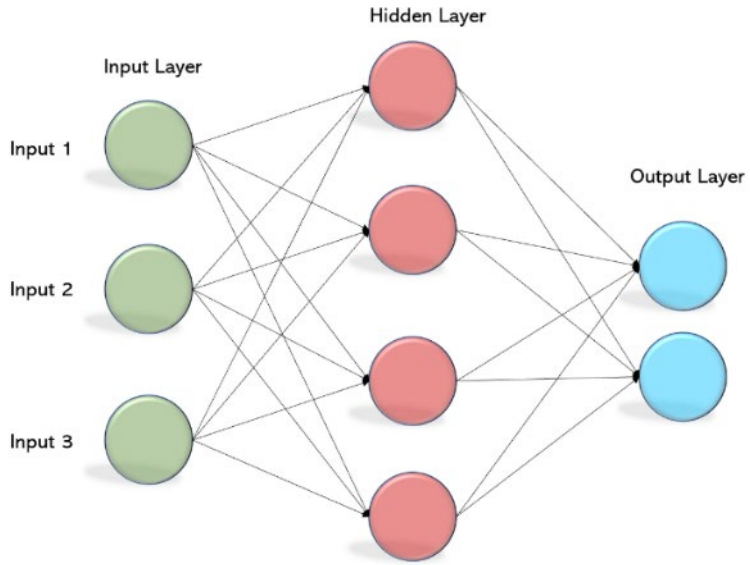
IV. 모델 해석

II. Supervised Learning – MLP

- 입력층과 출력층 사이에 하나 이상의 중간층 (은닉층) 이 존재하는 신경망으로, 퍼셉트론으로 이루어진 층 여러 개를 순차적으로 붙여놓은 형태

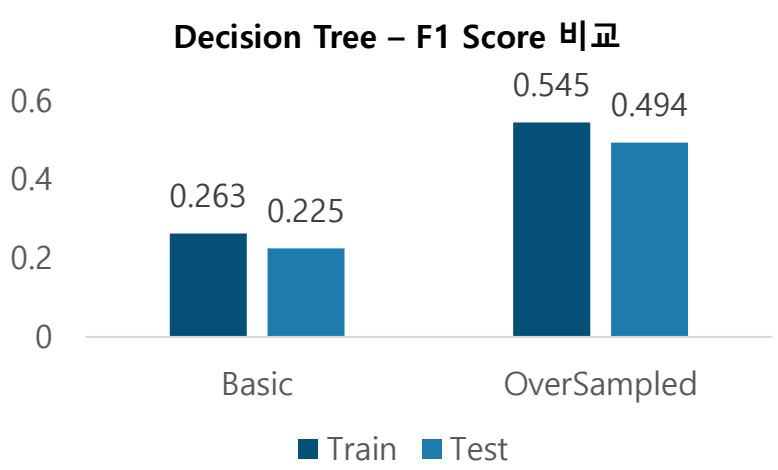
- 층을 여러 겹으로 쌓아가면서 비선형적으로 분류가 가능하며, 각 층의 오차는 역전파 알고리즘을 통해 업데이트해 나감

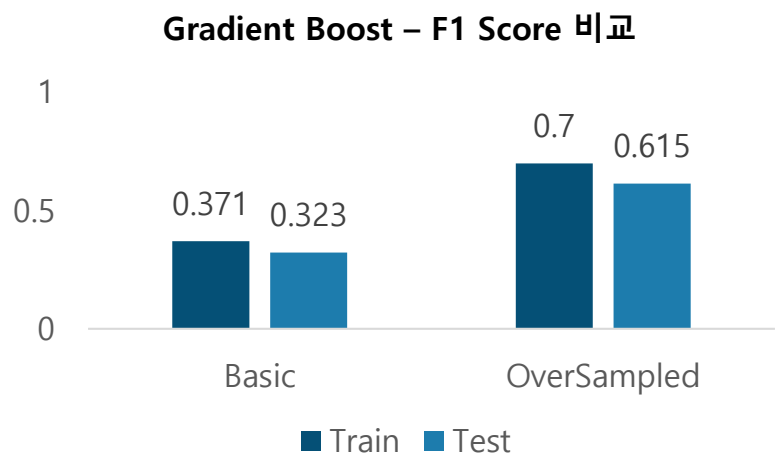
```
[ ]  # train
     MLP_scores_train = cross_validate(MLP_clf,X_train, y_ESG_train, scoring=scoring, cv=5, return_train_score = False)
     print("(train set) MLP_score before Oversampling - F1 Score: {:.3f}".format(np.mean(MLP_scores_train['test_f1_macro'])))

     # test
     MLP_scores_test = cross_validate(MLP_clf, X_test, y_ESG_test, scoring=scoring, cv=5, return_train_score = False)
     print("(test set) MLP_score before Oversampling - F1 Score: {:.3f}".format(np.mean(MLP_scores_test['test_f1_macro'])))

     ################## OverSampling ###################################

     # train
     MLP_scores_otrain = cross_validate(MLP_clf, o_X_train, o_y_ESG_train, scoring=scoring, cv=5, return_train_score = False)
     print("(train set) MLP_score after Oversampling - F1 Score: {:.3f}".format(np.mean(MLP_scores_otrain['test_f1_macro'])))

     # test
     MLP_scores_otest = cross_validate(MLP_clf, o_X_test, o_y_ESG_test, scoring=scoring, cv=5, return_train_score = False)
     print("(test set) MLP_score after Oversampling - F1 Score: {:.3f}".format(np.mean(MLP_scores_otest['test_f1_macro'])))

     (train set) MLP_score before Oversampling - F1 Score: 0.155
     (test set) MLP_score before Oversampling - F1 Score: 0.175
     (train set) MLP_score after Oversampling - F1 Score: 0.336
     (test set) MLP_score after Oversampling - F1 Score: 0.364
```

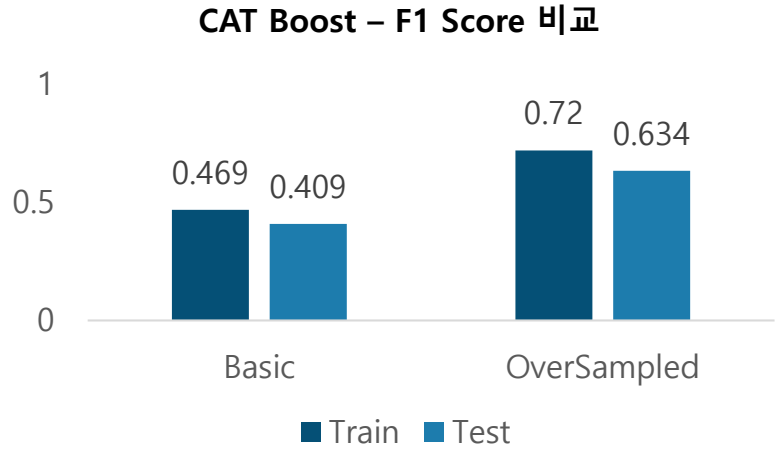


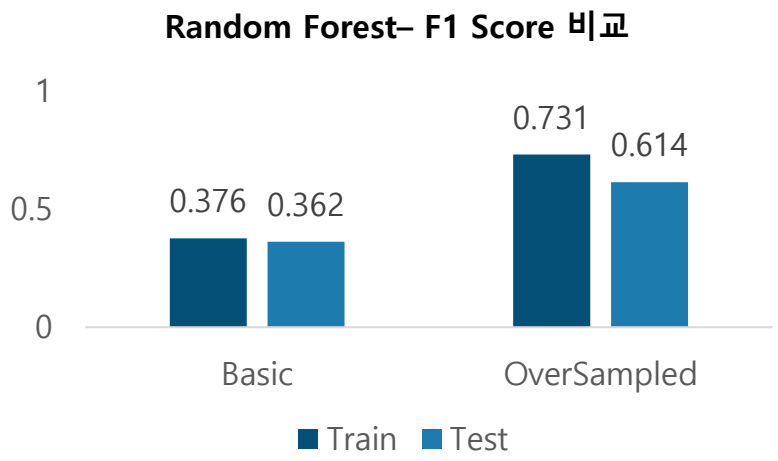
IV. 모델 해석

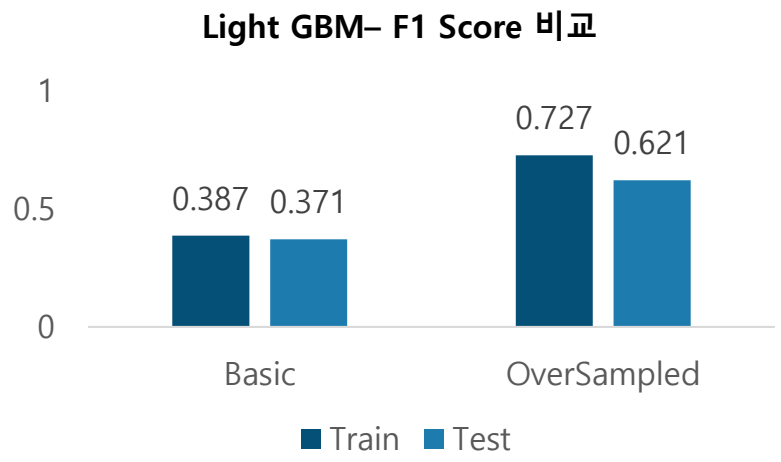
II. Supervised Learning – F1 score 확인 & OverFitting 확인

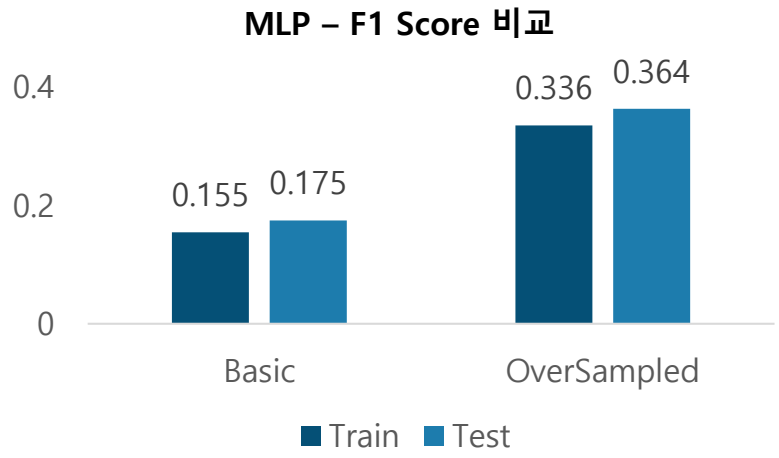






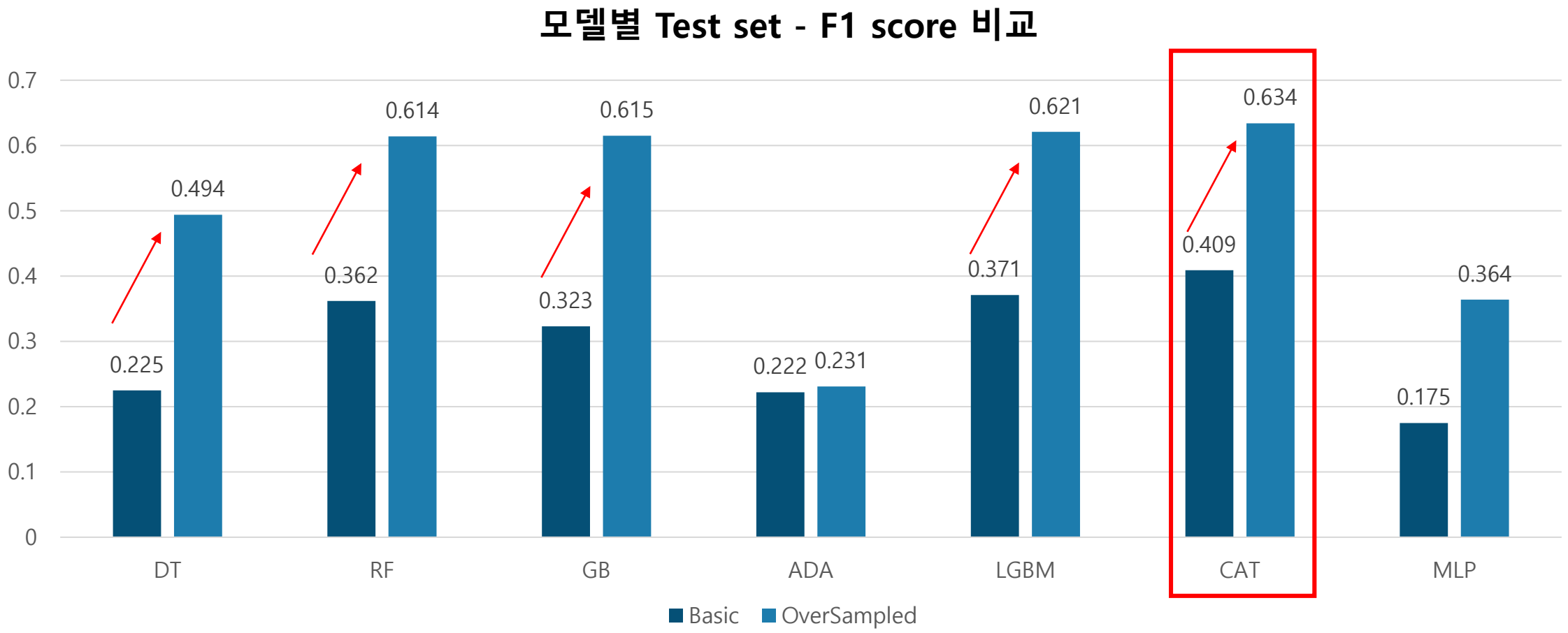






IV. 모델 해석

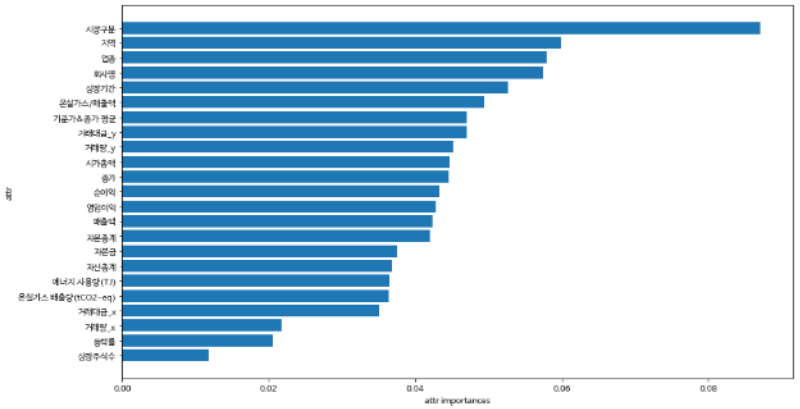
II. Supervised Learning – F1 Score 확인



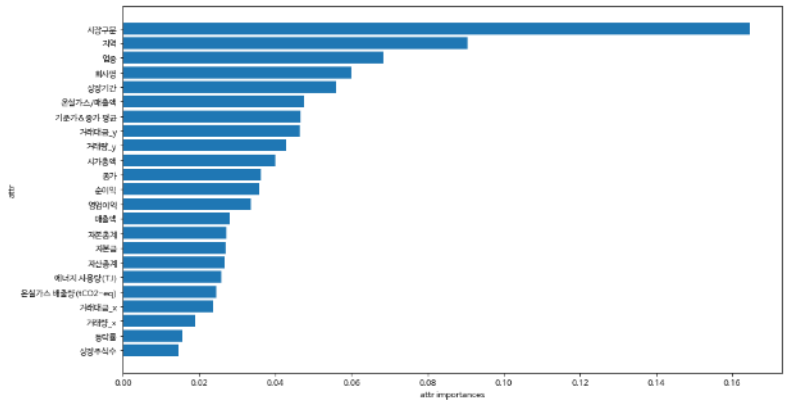
IV. 모델 해석

II. Supervised Learning – 모델별 변수 중요도 확인

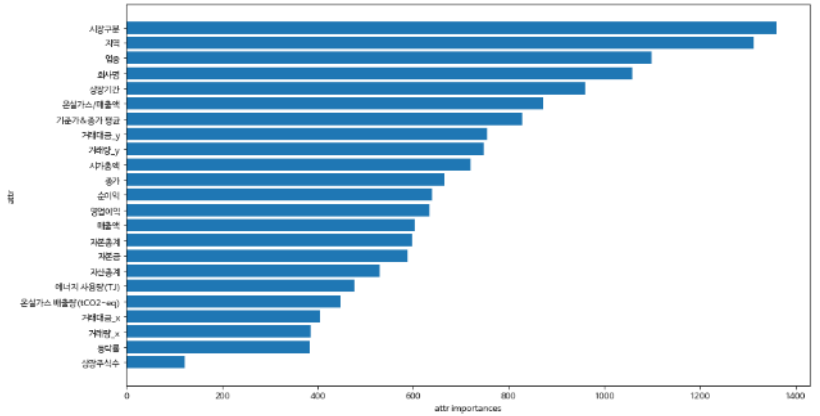
Random Forest 그래프 변수 중요도



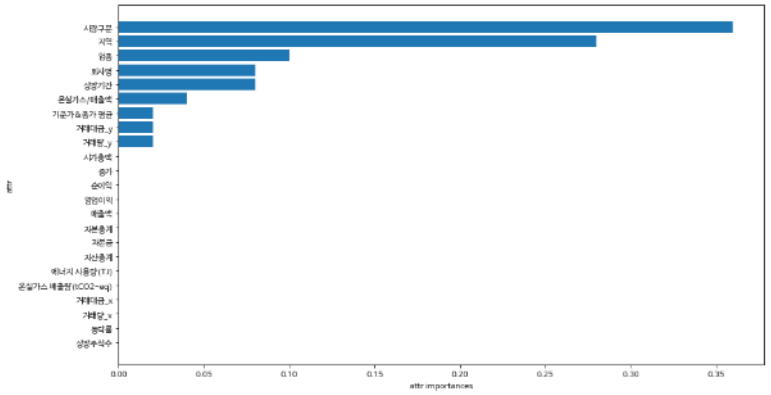
Gradient Boost 그래프 변수 중요도



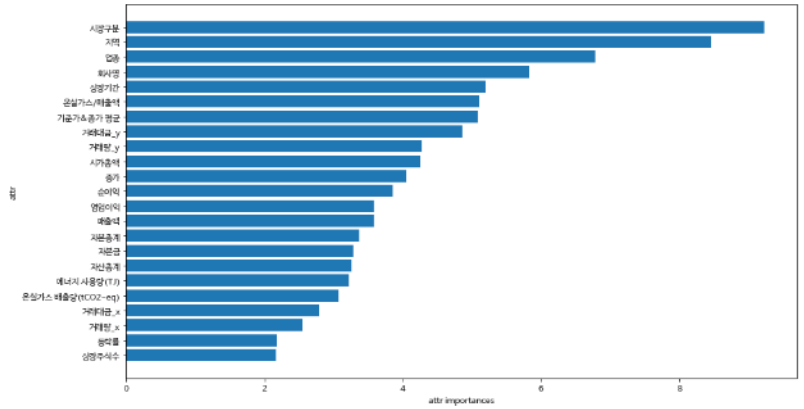
lgbm 그래프 변수 중요도



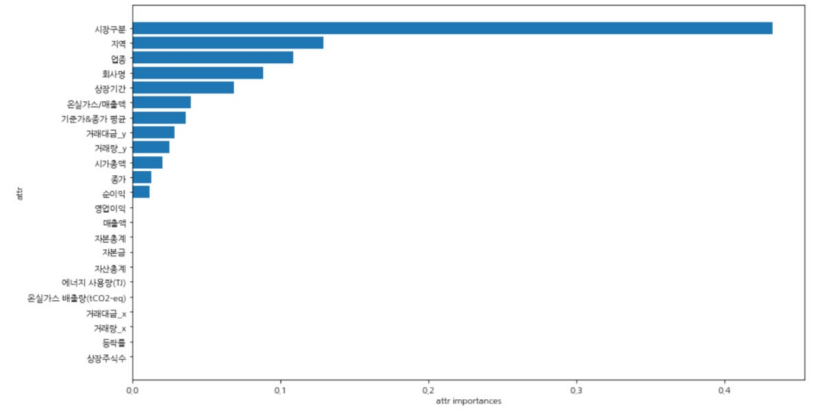
ADABoost 그래프 변수 중요도



CATBoost 그래프 변수 중요도



Decision Tree 그래프 변수 중요도



Part 5.

결과 해석

V. 결과 해석

I. Unsupervised 결과 해석

- 성과

  I. 각 클러스터별로, 50%가 넘는 Dominant 등급을 제공하는 클러스터 구분에 성공함. 특히 Oversampling을 사용한 경우 각 클러스터별로 70%가 넘는 Dominant 등급을 제공하는 클러스터 추출이 가능했음

- 한계

  I. 어디까지나 클러스터링에 의존한 분포를 제공하고 있으므로, Prediction 값의 정확도를 보장할 수 없음

  II. 각 클러스터별로 평균 50~70%의 Dominant를 가지고 있는 클러스터 분포의 제공이 소비자들에게 그렇게까지 유용하다고 장담할 수 없음

V. 결과 해석

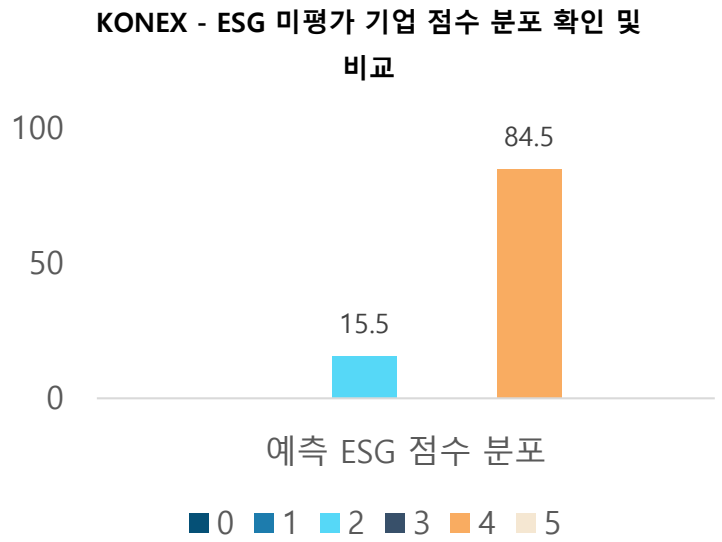
II. Supervised 결과 해석

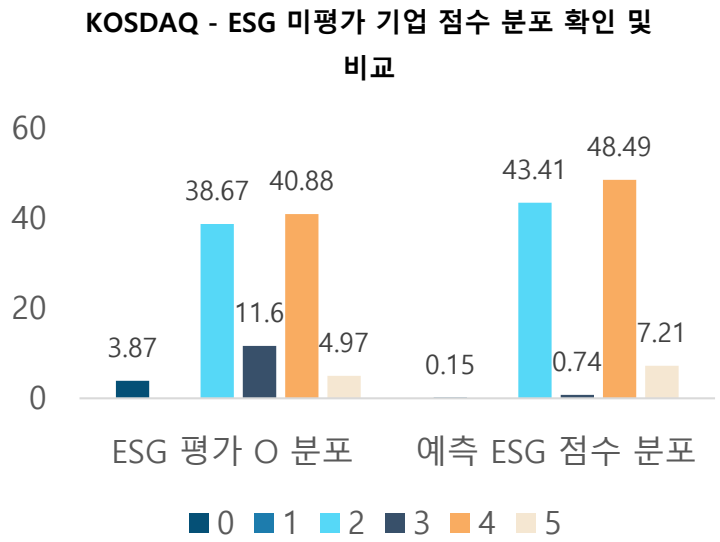
- 성과

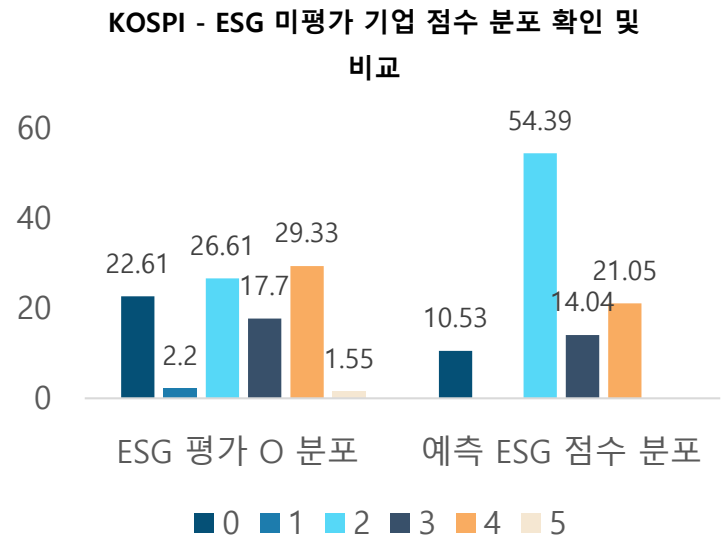
  I. 지도학습 방법론에서는 '학습 ' 을 통해 보다 세부적으로 예측할 수 있었다.
  II. 또한 모델의 예측력을 올릴 수 있었다.

- 한계

  I. 데이터 부족으로 인한 불균형으로 인해 클래스 분포가 고르지 못하다.



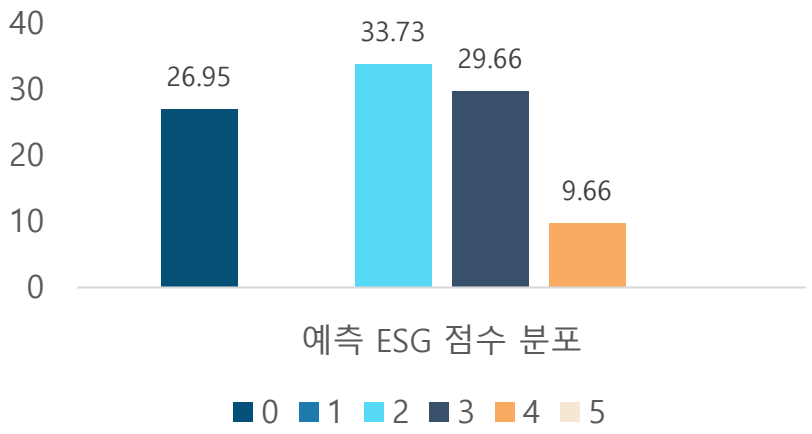




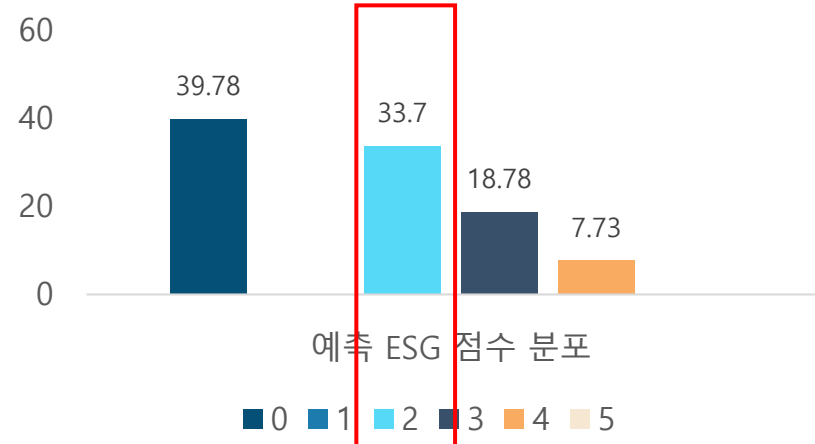
V. 결과 해석

II. Unsupervised vs Supervised 결과 비교

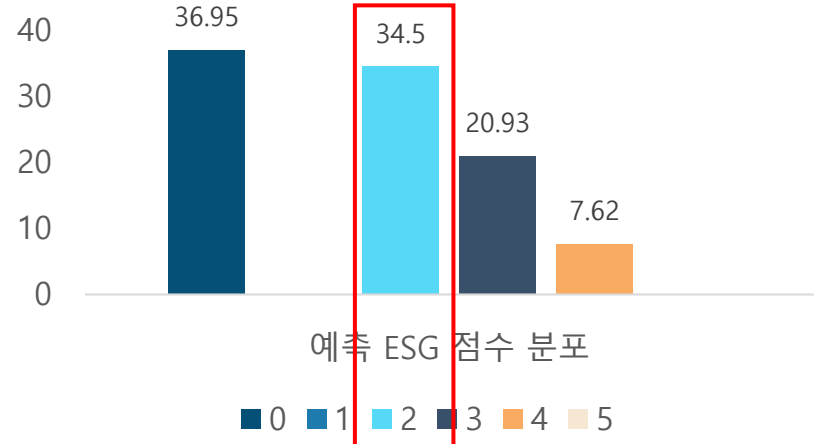
KONEX - ESG 미평가 기업 점수 분포 확인 및 비교



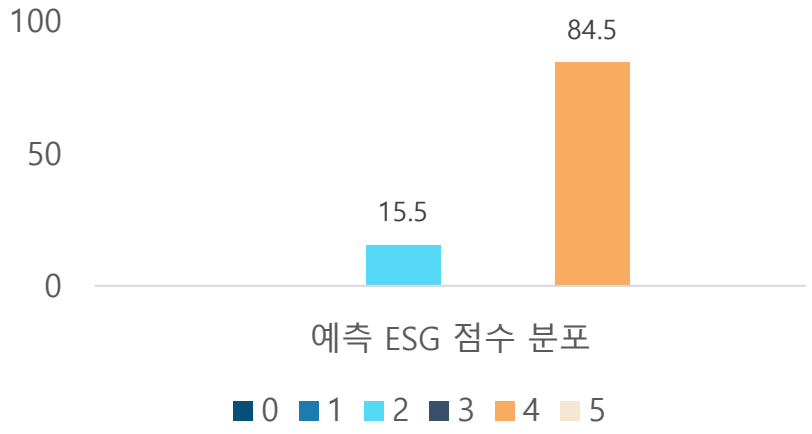
KOSDAQ - ESG 미평가 기업 점수 분포 확인 및 비교



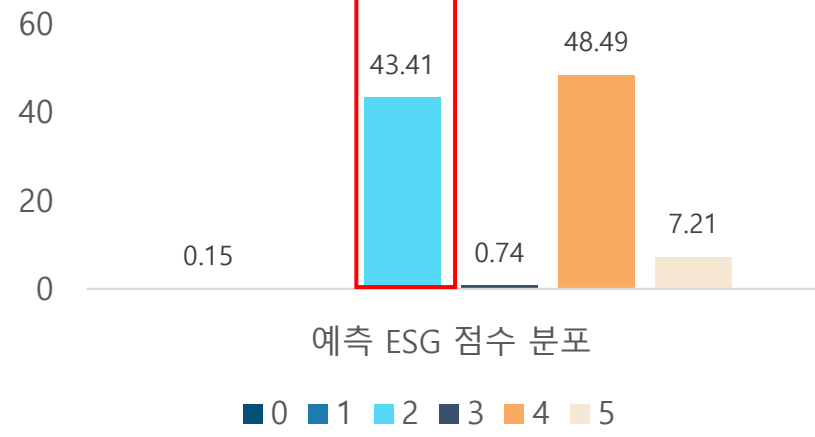
KOSPI - ESG 미평가 기업 점수 분포 확인 및 비교



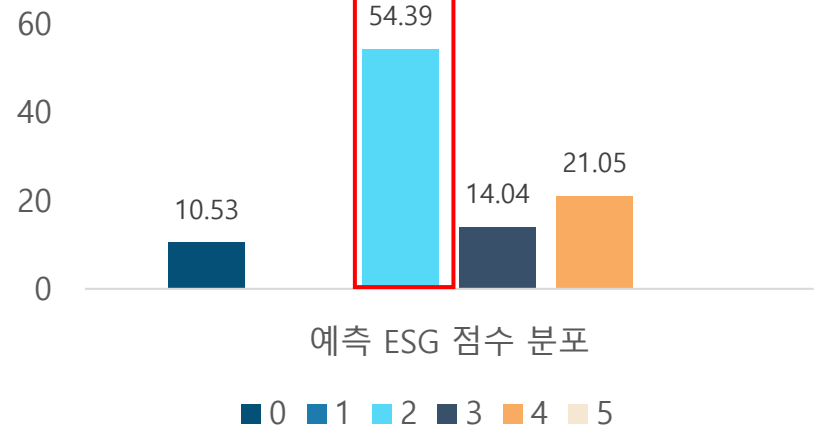
KONEX - ESG 미평가 기업 점수 분포 확인 및 비교



KOSDAQ - ESG 미평가 기업 점수 분포 확인 및 비교



KOSPI - ESG 미평가 기업 점수 분포 확인 및 비교



프로젝트를 마치며

프로젝트 한계 & 배운점

프로젝트 한계

I. 한계

- ESG 점수가 부여되지 않았던 기업들의 데이터에 결측치가 많아 모델의 정확도를 담보할 수 없으며, Hypertuning 이외에 모델 성능의 향상 방법이 모호함
- KCGS를 제외한 다른 두 기관(서스틴베스트, ESG연구소)의 클러스터 및 Classification 결과를 도출하지 못했음. 특히 서스틴베스트는 ESG 등급 산정에 있어서 자체적으로 재무적인 정보에 단순 가중치를 두는 것이 아니라 등급의 구분을 줌으로써 분석이 힘든 Multi-Binary Dataset을 제공함
- 데이터 부족으로 클래스 분포가 고르지 못하며, 결측치를 처리하는 과정에서 기존 데이터에 많이 의존하여 왜곡된 해석 결과가 있을 수 있음
- 지도학습 모델의 경우, 각 모델별 Feature 중요도는 train 데이터셋으로부터 얻은 통계량으로 계산된 중요도이기 때문에 test 데이터셋의 경우에는 중요도가 다르게 나타날 수 있음

프로젝트 한계 & 방향

2. 차후 진행 방향

- 지도학습 기반의 모델링 과정을 거쳐 목적 기업의 예상 ESG 등급을 라벨값으로 제시하지 않는 비지도학습 한계점을 보완할 수 있음
- 각 기관별로 ESG 등급 산정 기준을 명확히 하여 ESG 등급을 예측하기 위한 데이터를 수집하는 방법론을 보강할 필요가 있음
- 기업을 설명하는 데이터셋 보강
- 각 평가기관별 ESG 등급의 차이를 구체적으로 밝혀내고, 해당 사항을 이용해 모델을 보정할 수 있음