과제 2

: 의사 결정 트리(Decision_Tree)

이름 : 김진석
학번 : 202210829
학과 : 휴먼지능정보공학과

목차

1. Decision_Tree

2. Entropy
    2.1 기존 코드
    2.2 변경 코드

3. Information Gain
    3.1 기존 코드
    3.2 변경 코드

4. Result

1. Decision_Tree

Decision_Tree는 데이터를 분류하거나 예측하는 데 사용되는 기계 러닝 알고리즘 중 하나입니다. Decision_Tree는 데이터의 특성을 기반으로 분기 구조를 형성하여 데이터를 분류하거나 예측합니다. 특히, 설명이 쉽고 일반화 성능이 좋아서 데이터 분석에 많이 사용됩니다. 그러나 과적 합에 취약하고 복잡한 데이터에 적합하지 않을 수 있다는 단점이 있습니다.

2. Entropy

Entropy는 상태의 무질서 도를 나타내는 척도로 사용됩니다. 상태의 무질서도가 클수록 Entropy가 높습니다. Entropy는 발생 가능한 사건의 확률 분포를 사용하여 계산됩니다.

Entropy는 다음과 같이 정의할 수 있습니다.

$$H(t, \mathcal{D}) = - \sum_{l \in levels(t)} (P(t = l) \times log_2(P(t = l)))$$

Entropy 수식 정의

여기서,

- H(t, D)는 섀넌 엔트로피
- p(t=L)는 L번째 사건의 확률

이다.

2.1 기존 코드

```
entropy = np.sum([(-counts[i] / np.sum(counts)) * np.log2(counts[i] / np.sum(counts))
                  for i in range(len(elements))])
```

• 코드 분석

1. counts 변수에는 target_col의 각 고유값에 대한 빈도가 저장됩니다.
2. np.sum(counts) 함수를 사용하여 counts의 합을 계산합니다.
3. np.log2(counts / np.sum(counts)) 함수를 사용하여 각 고유값의 정보량을 계산합니다.
4. -counts / np.sum(counts) 함수를 사용하여 각 고유값의 확률을 계산합니다.
5. np.sum() 함수를 사용하여 각 고유값의 엔트로피를 계산합니다.

2.2 변경 코드

방법 1

```
probs = counts / np.sum(counts)
entropy = (-probs * np.log2(probs)).sum()
```

변경 코드 1

• 코드 분석

1. counts 변수에는 target_col의 각 고유값에 대한 빈도가 저장됩니다.
2. probs 변수에는 counts를 np.sum(counts)로 나누어 각 고유값의 확률을 계산합니다.
3. entropy 변수에는 probs와 np.log2(probs)를 곱한 값을 모두 더하여 섀넌 엔트로피를 계산합니다.

• 기존 코드와의 차이점

항목	전 코드	변경된 코드_1
시간 복잡도	O(n log n)	O(n)
코드 간결성	복잡	간결
주의 사항	for 루프 사용, np.sum() 함수 사용	probs 변수 사용, sum() 함수 사용

방법 2

```
entropy = scipy.stats.entropy(counts / np.sum(counts), base = 2)
```

변경 코드 2

• 코드 분석

1. counts 변수에는 target_col의 각 고유값에 대한 빈도가 저장됩니다.
2. probs 변수에는 counts를 np.sum(counts)로 나누어 각 고유값의 확률을 계산합니다.
3. entropy 변수에는 scipy.stats.entropy() 함수를 사용하여 섀넌 엔트로피를 계산합니다.

• 기존 코드와의 차이점

항목	전 코드	개선된 코드
함수	직접 구현	scipy.stats.entropy() 함수 사용
계산 순서	counts -> probs	counts-> probs -> scipy.stats.entropy()
시간 복잡도	O(n log n)	O(n)
코드 간결성	복잡	간결

3. Information Gain

$$IG(d, \mathcal{D}) = H(t, \mathcal{D}) - rem(d, \mathcal{D})$$

정보 이득(information gain)은 의사결정 트리 학습에서 사용되는 지표입니다. 정보 이득은 특정 속성이 목표 변수에 대한 정보를 얼마나 제공하는지를 측정합니다. 정보 이득이 높은 속성은 목표 변수를 예측하는 데 더 유용한 속성입니다. 정보 이득이 양수이면 속성이 목표 변수에 대한 정보를 제공한다고 할 수 있고, 정보 이득이 클수록 속성은 목표 변수를 더 잘 예측한다고 할 수 있다. 정보 이득이 음수이면 속성은 목표 변수에 대한 정보를 제공하지만, 목표 변수를 예측하는 데 방해가 됩니다.

3.1 기존 코드

```
Weighted_Entropy = np.sum([(counts[i] / np.sum(counts)) * entropy(data.where(data[split_attribute_name] == vals[i]).dropna()[target_name]) \
                  for i in range(len(vals))])
```

• 코드 분석

1. counts와 np.sum(counts)를 사용하여 각 클래스의 가중치를 계산합니다.
2. for 루프를 사용하여 각 클래스에 대한 엔트로피를 계산합니다.
3. data.where(data[split_attribute_name]==vals[i]).dropna()[target_name]은 분할 속성 값이 vals[i]인 데이터의 목표 변수 값만 추출합니다.
4. entropy() 함수는 추출된 데이터의 엔트로피를 계산합니다.
5. 계산된 엔트로피와 가중치를 곱하고 모두 더하여 가중된 엔트로피를 계산합니다.

3.2 변경 코드

방법 1

• 코드 분석

1. data.loc[data[split_attribute_name] == value, target_name]을 사용하여 각 클래스 데이터를 추출합니다. 이는 분할 속성 값이 value인 데이터만 선택하고, .loc 인덱싱을 통해 결측값을 자동으로 제거합니다.
2. (data[split_attribute_name] == value).sum() / len(data)을 사용하여 클래스별 가중치를 계산합니다. 이는 분할 속성 값이 value인 데이터의 수를 계산하고, 전체 데이터 수로 나누어 가중치를 계산합니다.

• 기존 코드와의 차이점

항목	기존 코드	개선된 코드
데이터 접근 방식	data.where(data[split_attribute_name] == vals[i]).dropna()[target_name]	data.loc[data[split_attribute_name]== value, target_name]
가중치 계산	counts[i] / np.sum(counts)	(data[split_attribute_name]== value).sum() / len(data)
엔트로피 계산	직접 정의한 entropy() 함수 사용	scipy.stats.entropy() 함수 사용
루프 사용	for 루프 사용	루프 사용 안 함
코드 간결성	복잡	간결
성능	O(n^2)	O(n)

3. Result

• result 1

```
{'legs': {0: {'fins': {0.0: {'toothed': {0.0: 7.0, 1.0: 3.0}},
                       1.0: {'eggs': {0.0: 1.0, 1.0: 4.0}}}},
          2: {'hair': {0.0: 2.0, 1.0: 1.0}},
          4: {'hair': {0.0: {'toothed': {0.0: 7.0, 1.0: 5.0}}, 1.0: 1.0}},
          6: {'aquatic': {0.0: 6.0, 1.0: 7.0}},
          8: 7.0}}
The prediction accuracy is:  85.71428571428571 %
```

• 결과 시각화

```
from IPython.display import Image
from graphviz import Digraph
```

를 활용하여 결과를 시각화 해보았다.