
Data Analysis: Machine Learning Base Model(2)

ESC 2024 Summer Special Session 4차
발제자: 전제훈



Contents

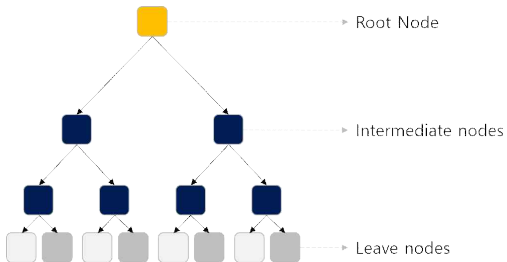
1. Dual Representations
2. Constructing Kernels
3. Radial Basis Function Networks
4. Gaussian Process

1

Decision Tree

Decision Tree

결정 트리 모델 개요



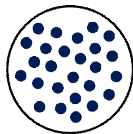
결정 트리 모델은 다음과 같이 데이터를 적절한 기준으로 구분하는 모델이라고 배웠다.

결정 트리 모델에서 가장 중요한 부분은 어떻게 적절하게 기준선을 정할 것이냐라는 것이다.

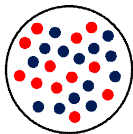
이에 대해 불순도라는 개념이 존재하였다.

Decision Tree

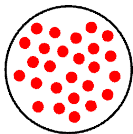
불순도



불순도 = 0



불순도 > 0



불순도 = 0

불순도는 다음과 같으며, 직관적으로 얼마나 데이터가 혼합해져 있는지라고 생각하면 된다.

우리가 특정 기준점을 만들고, 그 기준점으로 분류를 하였을 때,

하나의 데이터들만 모이게 된다면 (불순도=0)

우리는 적절하게 기준선을 세웠다고 볼 수 있을 것이다.

그럼 우리는 불순도가 적어지도록 어떻게 기준을 잡을 수 있을까?

이에 대해서는 지니계수를 이용한 (CART) 알고리즘과 엔트로피를 이용한 (ID3, C4) 이 존재

Decision Tree

ID3 알고리즘

ID3 알고리즘은 entropy와 information gain을 통해 분류하는 알고리즘이다.

엔트로피는 다음과 같이 계산할 수 있다.

$$Entropy = H(S) = \sum_{i=1}^c p_i \log_2 \frac{1}{p_i} = - \sum_{i=1}^c p_i \log_2 p_i$$

엔트로피는 직관적으로 놀람의 정도로 해석이 가능하다.

entropy를 기반으로 하여 information gain을 통해 기준 변수와 기준 점을 정하게 된다.

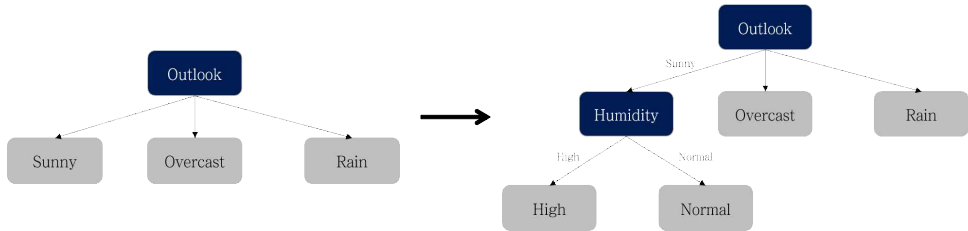
$$Information\ Gain(D, A) = Entropy(D) - \sum_{v \in Values(A)} \frac{|D_v|}{|D|} \cdot Entropy(D_v)$$

해당 식이 Information Gain 함수인데,

수식적으로는 복잡해보이는데 그냥 노드 나누기 전 엔트로피와 나눈 이후 엔트로피 가중평균의 차이라고 보면된다.

Decision Tree

ID3 알고리즘



Outlook 변수에는 Sunny, Overcast, Rain 총 3개의 변수가 존재하는데, Sunny인 데이터들만을 갖고 information gain을 측정하였을 때, Humidity일 때가 information gain이 가장 높아 다음 변수를 Humidity가 되는 알고리즘이다.

Decision Tree

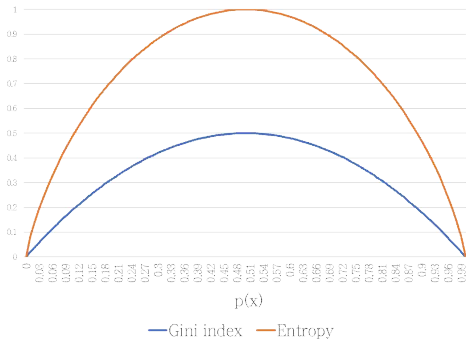
CART 알고리즘

지니계수와 엔트로피 각각 확률값에 대한 결과 차이이다.

보면 알겠지만, 엔트로피는 확률의 중앙값과 극값의 차이가 크게 나오는 편임을 확인 할 수 있다.

하지만, cart 논문에서는 엔트로피나 지니 계수나 둘 중 어느 것을 사용해도 성능의 차이는 없으며

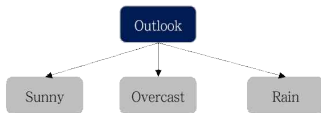
단순하게 지니계수를 사용하는 것이 속도 측면에서 더 좋기 때문에 지니 계수를 사용하였다고 되어 있다.



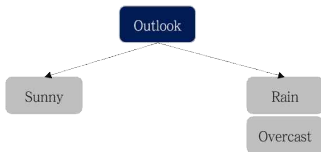
Decision Tree

CART 알고리즘

ID3



CART



CART와 ID3의 차이점으로 볼 수 있는 것은 ID3는 범주형 변수를 각각 다른 노드로 보려는 경향이 있는데, CART에서는 오직 2개의 노드로만 분리하는 경향이 있다.

작동 방식은 ID3와 동일하게 Information Gain이 최대가 되는 기준으로 분류한다.

Decision Tree

CART 알고리즘

노드를 오직 2개로만 나눠야하는 CART 알고리즘의 특성상 Information Gain을 여러 번 계산하여야 하는데,

ID3 알고리즘에서는

$IG(\text{Play}, \text{Outlook})$, $IG(\text{Play}, \text{Temperature})$, $IG(\text{Play}, \text{Humidity})$, $IG(\text{Play}, \text{Windy})$

즉, 변수들에 대해서만 Information Gain을 계산한다고 하면,

CART 알고리즘에서는

$IG(\text{Play}, \text{Outlook}=\text{sunny})$, $IG(\text{Play}, \text{Outlook}=\text{overcast})$, $IG(\text{Play}, \text{Outlook}=\text{rain})$, ...

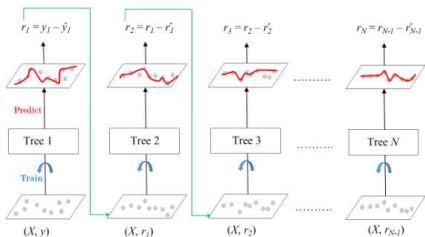
변수들 안에 있는 값들도 나누어서 Information Gain을 계산한다.

2

Boosting

Xgboost
Catboost

Boosting



Boosting에 대한 설명은 지난주 내용을 참고하자.

간략하게 살펴보자면,

Boosting은 약한 학습자를 결합하여 강한 학습자를 만드는 머신러닝 앙상블 기법이다.

즉, 단순한 결정트리를 만든 이후, 잘못 예측된 데이터에는 강한 가중치를 잘 예측된 데이터에는 약한 가중치를 주어 이전 단계의 예측을 보완하는 방식이다.

이번 주차에는 Boosting의 대표주자인 XgBoost, CatBoost에 대해 살펴볼 예정이다.

XGBoost



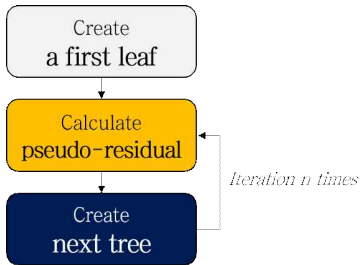
기존 선행 연구인 **Gradient Boosting**은 각 학습자가 이전 학습자의 잔차를 예측하는 방식으로 진행이 된다.

그리하여 학습이 계속 진행될수록 잔차를 줄이는 방식이다.

XgBoost는 이의 확장된 버전으로 기존 **Gradient Boosting** 모델이 갖고 있는 오버피팅 문제와 느린 학습문제를 해결하고자 하였다.

XgBoost는 회귀와 분류 문제 모두 적용가능한데, 이에 대해 살펴보고 수식적인 내용도 같이 살펴보자.

XGBoost for Regression



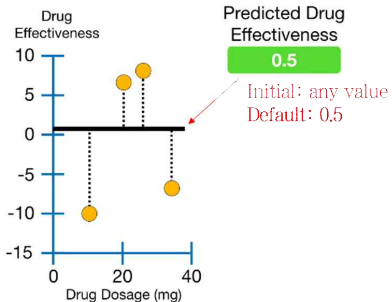
해당 step을 따라 모델이 작동된다.

첫번째 분류를 생성한 이후

잔차를 측정 한 이후, 이를 기반으로 다음 트리를 생성하게 된다.

해당 방식을 계속 반복하여 최종적인 모델을 뽑아내는 형식이다.

XGBoost for Regression



적절한 예측값을 초기에 잡아주도록 하자.

해당 값은 설정하기 나름이겠지만 보통 평균이 합리적이라고 본다.

그렇게 설정한 초기값(초기 예측값)으로부터의 각 데이터 마다의 잔차를 구하면,

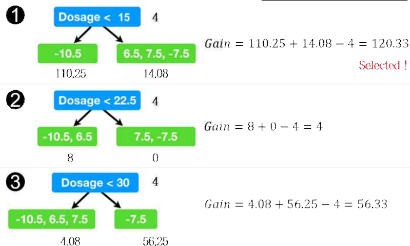
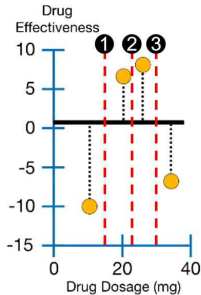
예를 들어 잔차가 (-10.5, 6.5, 7.5, -7.5) 이렇게 나왔다고 가정해보면, 잔차를 기반으로 한 우리의 예측값에 대한 평가 (similarity score)는 $(-10.5+6.5+7.5-7.5)^2 / 4 = 4$ 로 평가될 수 있다.

XGBoost for Regression

$$\text{Gain} = \text{Similarity}_{\text{Left}} + \text{Similarity}_{\text{Right}} - \text{Similarity}_{\text{Parent}}$$

$$\text{similarity} = \frac{\sum \text{residuals}}{\# \text{residuals} + \lambda}$$

(Assume λ is 0)



이제 우리는 similarity score를 높일 수 있는 최적의 기준점을 잡아야 한다.

이에 대한 계산은 Gain information 방식으로 진행이 되며,

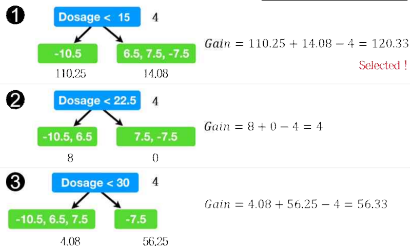
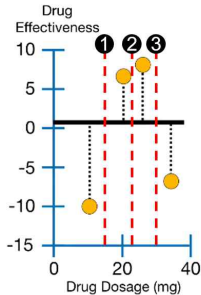
Gain information이 최대가 되는 기준점을 다음 스텝의 기준점으로 잡는다.

XGBoost for Regression

$$\text{Gain} = \text{Similarity}_{\text{Left}} + \text{Similarity}_{\text{Right}} - \text{Similarity}_{\text{Parent}}$$

$$\text{similarity} = \frac{\sum \text{residuals}}{\# \text{residuals} + \lambda}$$

(Assume λ is 0)



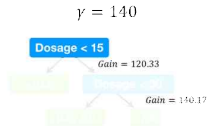
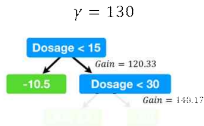
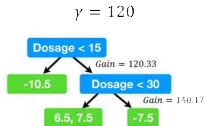
이제 우리는 similarity score를 높일 수 있는 최적의 기준점을 잡아야 한다.

이에 대한 계산은 Gain information 방식으로 진행이 되며,

Gain information이 최대가 되는 기준점을 다음 스텝의 기준점으로 잡는다.

XGBoost for Regression

$$\text{Gain} - \gamma \begin{cases} \text{If Positive, then do not prune} \\ \text{If negative, then prune} \end{cases}$$



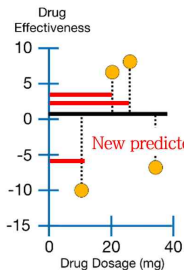
오버피팅을 막기 위한 가지치기의 경우 information gain의 값의 하한을 정해놓는 방식으로 진행된다.

위의 예시를 살펴보면, 감마값을 120, 130, 140으로 설정함으로써

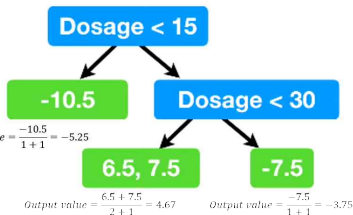
특정 깊이 이상으로 노드가 깊어지는 것을 방지한다.

XGBoost for Regression

$$\text{Output value} = \frac{\sum \text{Residuals}}{\# \text{Residuals} + \lambda}$$



$$\text{Output value} = \frac{-10.5}{1 + 1} = -5.25$$



$$\text{Output value} = \frac{6.5 + 7.5}{2 + 1} = 4.67$$

$$\text{Output value} = \frac{-7.5}{1 + 1} = -3.75$$

최종적인 예측은 처음 초기 예측값 (평균)에서 잔차를 더해 주어 최종적인 예측값을 결과물로 내뱉는다.

여기서 XgBoost의 아이디어가 하나 들어가게 되는데, 앞에서부터 계속 나왔지만, 정규화 람다가 존재한다.

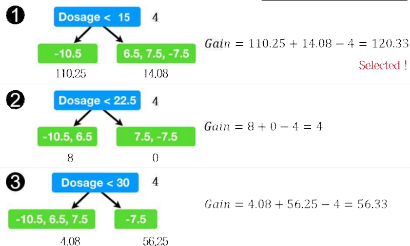
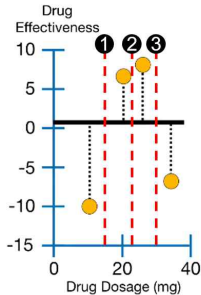
해당 람다를 통해 실제값에 대해 정확한 예측은 피함으로써 오버피팅 문제를 막아준다.

XGBoost for Regression

$$\text{Gain} = \text{Similarity}_{\text{Left}} + \text{Similarity}_{\text{Right}} - \text{Similarity}_{\text{Parent}}$$

$$\text{similarity} = \frac{\sum \text{residuals}}{\# \text{residuals} + \lambda}$$

(Assume λ is 0)



이전 슬라이드 그림을 다시보면,
유사도를 계산할 때 분모에 람다가 들어가는
것을 볼 수 있다.

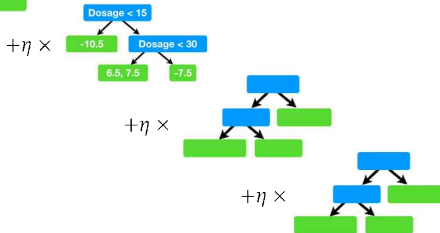
람다가 0이상의 값을 갖게 되면 유사도가 낮아지
는 경향이 있는데,

이 또한 특정 값 이상으로 유사도가 커지는 것을
방지하여 오버피팅을 막기 위한 흔적이라고
보면 된다.

XGBoost for Regression

$$\text{predicted value} = F_0 + \eta \sum_{i=1}^N F_i(x)$$

Predicted Drug
Effectiveness
0.5



앞서 해당 부분은 Gradient Boosting과 유사한데,
Gradient Boosting에서 오버피팅을 막기 위해 사용하였던
학습률 개념을 도입하고,

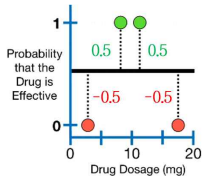
그로인해 생기는 잔차는 다시 새롭게 모델을 생성하여 예
측을 한다.

여기부분은 예측이 잘 안된 값에 대해 좀 더 큰 가중치를
주는 부분이다. 이렇게 생각해주면 좋을 듯 하다.

XGBoost for Classification

$$\text{similarity} = \frac{(\sum \text{Residual}_i)^2}{\sum [\text{Previous probability}_i \times (1 - \text{Previous probability}_i)] + \lambda}$$

λ is a regularization parameter



-0.5, 0.5, 0.5, -0.5

$$\text{Similarity} = \frac{(-0.5 + 0.5 + 0.5 - 0.5)^2}{4 \times (0.5 \times (1 - 0.5))} = 0$$

$(\lambda = 0)$

분류 문제에서 회귀 문제와 동일하다.

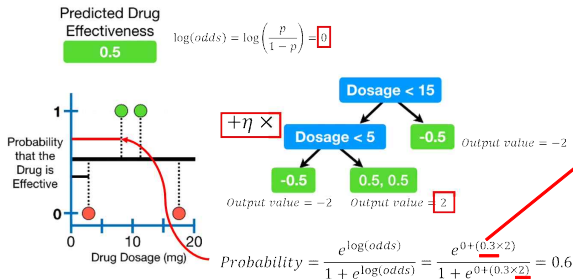
다만, 다른 것이 있다면,

similarity socre 계산 방식이 살짝 다르게 측정이 된다.

분모 부분에서 $p \times (1-p)$ 의 계산 방식을 따른다.

이에 대해서는 뒷부분 수식에서 정확히 살펴보자.

XGBoost for Classification



해당 0.3값은 에타이다.
오버피팅을 방지하기 위한 장치

$$\text{similarity} = \frac{(\sum \text{Residual}_i)^2}{\sum [\text{Previous probability}_i \times (1 - \text{Previous probability}_i)] + \lambda}$$

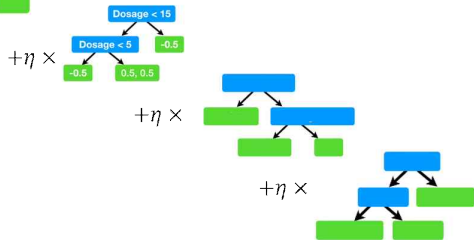
λ is a regularization parameter

해당 output value값은
앞서 확인한 similarity score를 통해 산출

XGBoost for Classification

Predicted Drug
Effectiveness

0.5



$$\text{predicted } \log(\text{odds}) = F_0 + \eta \sum_{i=1}^N F_i(x)$$

$$\text{predicted } \textbf{probability} = \frac{e^{\text{predicted } \log(\text{odds})}}{1 + e^{\text{predicted } \log(\text{odds})}}$$

XGBoost

$$D = \{(x_i, y_i)\}, (|D| = n, x_i \in R^m, y_i \in R)$$

$$\hat{y}_i = \phi(x_i) = \sum_{k=1}^K f_k(x_i), f_k \in F$$

$$L(\phi) = \sum_i l(\hat{y}_i, y_i) + \sum_k \Omega(f_k)$$

$$\text{where } \Omega(f) = \gamma T + \frac{1}{2} \lambda ||w||^2$$

k개의 tree의 정규화 값

f는 t번째 반복에서 낸 예측값다.

즉, 손실값은 현재값-예측값이므로 다음과 같이 정의가 된다.

$$L^{(t)} = \sum_{i=1}^n l(y_i, \hat{y}_i^{(t-1)} + f_t(x_i)) + \Omega(f_t) + Constant$$

2차 테일러 근사

$$L^{(t)} \approx \sum_{i=1}^n [l(y_i, \hat{y}_i^{(t-1)}) + \frac{\partial}{\partial \hat{y}_i^{(t-1)}} l(y_i, \hat{y}_i^{(t-1)}) \times f_t(x_i) + \frac{1}{2} \frac{\partial^2}{\partial^2 \hat{y}_i^{(t-1)}} l(y_i, \hat{y}_i^{(t-1)}) \times f_t^2(x_i)] + \Omega(f_t)$$

$$g_i = \frac{\partial}{\partial \hat{y}_i^{(t-1)}} l(y_i, \hat{y}_i^{(t-1)}), h_i = \frac{\partial^2}{\partial^2 \hat{y}_i^{(t-1)}} l(y_i, \hat{y}_i^{(t-1)})$$

$$L^{(t)} \approx \sum_{i=1}^n [l(y_i, \hat{y}_i^{(t-1)}) + g_i f_t(x_i) + \frac{1}{2} h_i f_t^2(x_i)] + \Omega(f_t)$$

$$\tilde{L}^{(t)} = \sum_{i=1}^n [g_i f_t(x_i) + \frac{1}{2} h_i f_t^2(x_i)] + \Omega(f_t)$$

업데이트를 위해 필요한 값은 g, h와 omega이므로 해당 값을 제외한 값은 상수 취급하여 삭제

XGBoost

$$\begin{aligned}\tilde{L}^{(t)} &= \sum_{i=1}^n [g_i f_t(x_i) + \frac{1}{2} h_i f_t^2(x_i)] + \gamma T + \frac{1}{2} \lambda \sum_{j=1}^T w_j^2 \\ &= \sum_{j=1}^T [(\sum_{i \in I_j} g_i) w_j + \frac{1}{2} (\sum_{i \in I_j} h_i + \lambda) w_j^2] + \gamma T\end{aligned}$$

각 샘플의 관점에서 보다가, 노드를 기준으로 식을 변형

$$w_j^* = - \frac{\sum_{i \in I_j} g_i}{\sum_{i \in I_j} h_i + \lambda}$$

j번째 노드의 최적의 가중치값

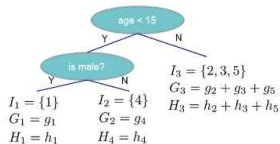
$$\tilde{L}^{(t)}(q) = -\frac{1}{2} \sum_{j=1}^T \frac{\sum_{i \in I_j} g_i}{\sum_{i \in I_j} h_i + \lambda}$$

$$I = I_L \cup I_R$$

$$L_{split} = \frac{1}{2} \left[\frac{(\sum_{i \in I_L} g_i)^2}{\sum_{i \in I_L} h_i + \lambda} + \frac{(\sum_{i \in I_R} g_i)^2}{\sum_{i \in I_R} h_i + \lambda} - \frac{(\sum_{i \in I} g_i)^2}{\sum_{i \in I} h_i + \lambda} \right] - \gamma$$

Instance index gradient statistics

1		g_1, h_1
2		g_2, h_2
3		g_3, h_3
4		g_4, h_4
5		g_5, h_5



$$Obj = - \sum_j \frac{G_j^2}{H_j + \lambda} + 3\gamma$$

The smaller the score is, the better the structure is

XGBoost

Algorithm 1: Exact Greedy Algorithm for Split Finding

Input: I , instance set of current node

Input: d , feature dimension

$gain \leftarrow 0$

$G \leftarrow \sum_{i \in I} g_i, H \leftarrow \sum_{i \in I} h_i$

for $k = 1$ **to** m **do**

$G_L \leftarrow 0, H_L \leftarrow 0$

for j in $sorted(I, \text{by } \mathbf{x}_{jk})$ **do**

$G_L \leftarrow G_L + g_j, H_L \leftarrow H_L + h_j$

$G_R \leftarrow G - G_L, H_R \leftarrow H - H_L$

$score \leftarrow \max(score, \frac{G_L^2}{H_L + \lambda} + \frac{G_R^2}{H_R + \lambda} - \frac{G^2}{H + \lambda})$

end

end

Output: Split with max score

모든 변수들에 대해, 변수 내의 모든 값에 대해 최적의 기준점을 찾고자 루프하는 형식

앞에서 구한 L_split 값

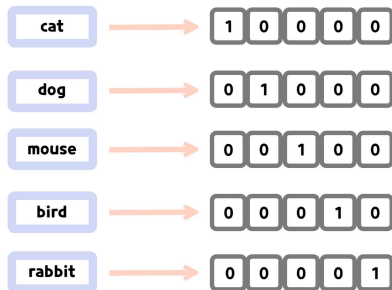
CATBoost

CatBoost 모델은 범주형 데이터 처리를 효율적으로 하기 위해 만든 방식이다.

기존 범주형 데이터 전처리 하는 방식에서 생기는 **data leakage** 문제를 해결하여 전처리를 진행하고자 하였으며,

부스팅 또한 독립적으로 진행하는 것이 아닌 순차적 부스팅을 이용해서 모델을 학습할 때도 **data leakage** 문제를 해결하고자 하였다.

CATBoost - One hot encoding




CATBoost를 살펴보기 전 범주형 변수 전처리 방식들에 대해 먼저 살펴보자.
가장 흔하게 사용하는게 원핫-인코딩 방식이다.

원핫-인코딩 방식은 다음과 같이 데이터 존재할 경우 이는 범주형이기에
숫자로 바꿔서 모델에 넣어줄 필요가 있고

숫자로 바꿔주는 과정에서 각각 항목마다 변수를 새롭게 만들어
해당하면 1 아니면 0으로 설정하는 방식의 전처리를 의미한다.

CATBoost - Label encoding

color	
red	
green	
blue	
red	



color	
0	
1	
2	
0	


다음으로는 라벨 인코딩이 있다.

기존 원핫 인코딩은 변수가 많이 늘어난다는 단점을 갖고 있다.

이에 대한 대안으로 (1,0,0), (0,1,0), (0,0,1) 방식이 아닌
1, 2, 3으로 바꿔줌으로써 좀 더 간편하게 전처리 하는 방식이다.

하지만, 이는 아무래도 $2 > 1 > 0$ 이라는 관계성을 갖고 있기에
순서형 데이터에 적절한 전처리이다.

CATBoost - Target encoding

Color Target_2				
0	Red	0		0
1	Red	0		0.200000
2	Red	1		0.200000
3	Red	0		0.200000
4	Red	0		0.200000
5	Green	0		0.333333
6	Green	1		0.333333
7	Green	0		0.333333

이를 보완하고자 타겟 인코딩이라는 것이 있다.

타겟 인코딩 자체는 우리가 지금 하고 있는 결측치 처리처럼 다른 변수를 활용해서 범주형 변수를 숫자화해주는 작업이다.

다음과 같이 빨강색의 값을 해당 타겟 값들의 평균으로 전처리해주는 것이다. 이렇게 된다면, 모델입장에서도 의미가 있는 값들이 들어가게 될 것이며 변수들간의 순서 및 비례도 어느정도 숫자로 표현할 수 있다.

하지만, 해당 인코딩의 문제점은 데이터가 적을 때 발생한다.

적은 표본에서 평균을 구하게 될 경우 발생하는 편의와 비슷하다고 보면 된다.

이를 위해 어느정도의 보정작업을 할 수는 있다.

CATBoost - k-fold target encoding

A	Color Target_2			B	Color_Target_2	
	0	Red	0		0	0.200000
	1	Red	0		1	0.200000
	2	Red	1		2	0.200000
	3	Red	0		3	0.200000
	4	Red	0		4	0.200000
	5	Green	0		5	0.333333
	6	Green	1		6	0.333333
	7	Green	0		7	0.333333

타겟 인코딩 자체의 아이디어는 좋지만,

인코딩 방식이 우리가 알고 싶은 값에서 도출하는 방식이기에
답지를 미리알고 적용하는 것과 동일하다.

즉, 훈련데이터에서는 잘 작동하고 테스트 데이터에서는 잘 작동하지 않을
확률이 크다.

이를 해결하고자 k-fold 방식을 도입한다.

A, B로 데이터 셋을 분해한 다음에 B에 라벨에는 A에서 얻은 값으로 집어넣는
방식이다.

이를 통해 데이터 유출 (data leakage) 문제를 해결하고자 하였다.

CATBoost - ordered target encoding

	Color	Target_2
0	Red	0
1	Red	0
2	Red	1
3	Red	0
4	Red	0
5	Green	0
6	Green	1
7	Green	0

해당 값을 인코딩하기 위해
해당 값이 아닌 이전 값을 사용
하여 표시하는 방식이다.

고로, 다음 Red와는 다른 값을
갖게 된다.

CATBoost에서는 데이터를 순서데이터로 생각하여
이를 적용한 ordered target encoding이라는 것을 적용하였다.

이는 확실히 정답지를 보고 인코딩하는 방식이 아니기 때문에
데이터 유출 문제는 발생하지 않게 되겠지만,

데이터를 sequential하게 보는 것이기 때문에
이에 맞춰 모델도 수정해야될 필요가 있다.

CATBoost

Algorithm 2: Building a tree in CatBoost

```
input :  $M, \{(\mathbf{x}_i, y_i)\}_{i=1}^n, \alpha, L, \{\sigma_i\}_{i=1}^s, Mode$ 
 $grad \leftarrow CalcGradient(L, M, y);$ 
 $r \leftarrow random(1, s);$ 
if  $Mode = Plain$  then
   $G \leftarrow (grad_r(i) \text{ for } i = 1..n);$ 
if  $Mode = Ordered$  then
   $G \leftarrow (grad_{r, \sigma_r(i)-1}(i) \text{ for } i = 1..n);$ 
 $T \leftarrow \text{empty tree};$ 
foreach step of top-down procedure do
  foreach candidate split c do
     $T_c \leftarrow \text{add split c to } T;$ 
    if  $Mode = Plain$  then
       $\Delta(i) \leftarrow avg(grad_r(p) \text{ for } p : leaf_r(p) = leaf_r(i)) \text{ for } i = 1..n;$ 
    if  $Mode = Ordered$  then
       $\Delta(i) \leftarrow avg(grad_{r, \sigma_r(i)-1}(p) \text{ for } p : leaf_r(p) = leaf_r(i), \sigma_r(p) < \sigma_r(i)) \text{ for } i = 1..n;$ 
     $loss(T_c) \leftarrow cos(\Delta, G)$ 
   $T \leftarrow argmin_{T_c}(loss(T_c))$ 
if  $Mode = Plain$  then
   $M_{r'}(i) \leftarrow M_{r'}(i) - \alpha avg(grad_{r'}(p) \text{ for } p : leaf_{r'}(p) = leaf_{r'}(i)) \text{ for } r' = 1..s, i = 1..n;$ 
if  $Mode = Ordered$  then
   $M_{r', j}(i) \leftarrow M_{r', j}(i) - \alpha avg(grad_{r', j}(p) \text{ for } p : leaf_{r'}(p) = leaf_{r'}(i), \sigma_{r'}(p) \leq j) \text{ for } r' = 1..s, i = 1..n, j \geq \sigma_{r'}(i) - 1;$ 
return  $T, M$ 
```

M: 모델, alpha: 학습률, sigma: 순열(permutation하므로)

모든 데이터 구분점에 대해서 살펴보는 방식

1~s까지 랜덤한 하나에 데이터값에 속한 leaf에서 해당 데이터 이전 데이터들의 평균 기울기 값을 업데이트 값으로 설정해준다.

모든 데이터에 대한 기울기 값과 코사인 유사도가 비슷한 기준점을 최종적 기준점으로 삼는다.

감사합니다