NUOVO HACKATHON

제 2회 | 항공사 고객 만족도 예측 AI 모델 개발

멘토진

백승훈 나우성 신다연

CONTENTS

1 대회 소개

항공사 고객 만족도 예측 AI 모델 개발

난이도

50%

대회 목적 :

정형화된 데이터를 다뤄보면서 자신만의 아이디어로 모델의 정확도를 향상시켜보는 두 번째 누아보 해커톤입니다.

실제로 비행한 고객의 세부정보를 바탕으로 미래의 고객 만족도를 정확히 예측해봅시다!

대회기간:

오프라인: 8/3 – 10:00~17:00 (7시간), 8/4 – 10:00~15:00 (5시간)

보고서 제출 마감: 8/7 – 23:59

이번 해커톤을 통해 다양한 배경을 가진 사람들과 교류하고 협력하며 실질적인 문제 해결 능력을 향상시켜봅시다!

AI모델의 정확도를 높이는 과정에서 다양한 방법론이 활용되므로 귀중한 경험이 될 것입니다.

우수한 결과를 낸 팀에게는 상장과 상금이 주어지며, 참가한 모든 팀에게 수료증이 수여됩니다.

본 대회가 여러분의 무궁한 발전의 첫 발판이 되길 바랍니다 :)

2 데이터 소개

항공사를 이용한 고객의 설문조사 데이터

	Gender	Customer Type	Age	Type of Travel	Class	Flight Distance	Seat comfort	Departure/Arrival time convenient	Food and drink	Gate location	Inflight wifi service	Inflight entertainment	Online support	Ease of Online booking	On-board service	Leg room service
0	Female	disloyal Customer	22	Business travel	Eco	1599	3	0	3	3	4	3	4	4	5	4
1	Female	Loyal Customer	37	Business travel	Business	2810	2	4	4	4	1	4	3	5	5	4
2	Male	Loyal Customer	46	Business travel	Business	2622	1	1	1	1	4	5	5	4	4	4
3	Female	disloyal Customer	24	Business travel	Eco	2348	3	3	3	3	3	3	3	3	2	4
4	Female	Loyal Customer	58	Business travel	Business	105	3	3	3	3	4	4	5	4	4	4

독립변수는 항공 서비스와 관련된 다양한 특성을 포함하며, 각 변수의 타입은 숫자형 또는 범주형 데이터로 구성되어 있습니다.
독립변수들 간의 관계를 살펴보고 종속변수인 TARGET (고객 만족도) 를 예측해봅시다.

2 데이터 소개

항공사를 이용한 고객의 설문조사 데이터

- **SEAT COMFORT :** 좌석 편안함 점수 (1~5)
- **DEPARTURE/ARRIVAL TIME CONVENIENT :** 출발/도착 시간의 지연 점수 (1~5)
- **FOOD AND DRINK :** 기내식 및 음료 서비스 점수 (1~5)
- **GATE LOCATION :** 게이트 위치 점수 (1~5)
- **INFLIGHT WIFI SERVICE :** 기내 와이파이 서비스 점수 (1~5)
- **INFLIGHT ENTERTAINMENT :** 기내 엔터테인먼트 서비스 점수 (1~5)
- **ONLINE SUPPORT :** 온라인 지원 서비스 점수 (1~5)
- **EASE OF ONLINE BOOKING :** 온라인 예약의 용이성 점수 (1~5)
- **ON-BOARD SERVICE :** 기내 서비스 점수 (1~5)
- **LEG ROOM SERVICE :** 레그룸 서비스 점수 (1~5)
- **BAGGAGE HANDLING :** 수하물 처리 점수 (1~5)
- **CHECKIN SERVICE :** 체크인 서비스 점수 (1~5)
- **CLEANLINESS :** 청결 점수 (1~5)
- **ONLINE BOARDING :** 온라인 탑승 서비스 점수 (1~5)

- **GENDER :** 성별 (MALE, FEMALE)
- **CUSTOMER TYPE :** 고객 유형 (LOYAL, DISLOYAL)
- **AGE :** 나이
- **TYPE OF TRAVEL :** 여행 유형 (BUSINESS, PERSONAL)
- **CLASS :** 비행 클래스 (ECO, BUSINESS, ECO PLUS)
- **FLIGHT DISTANCE :** 비행 거리
- **DEPARTURE DELAY IN MINUTES :** 출발 지연 시간 (분)
- **ARRIVAL DELAY IN MINUTES :** 도착 지연 시간 (분)

종속변수 TARGET : 만족 여부 (0 : 불만족, 1 : 만족)

3 EDA 예시

EDA 통한 데이터 분포 및 관계 확인

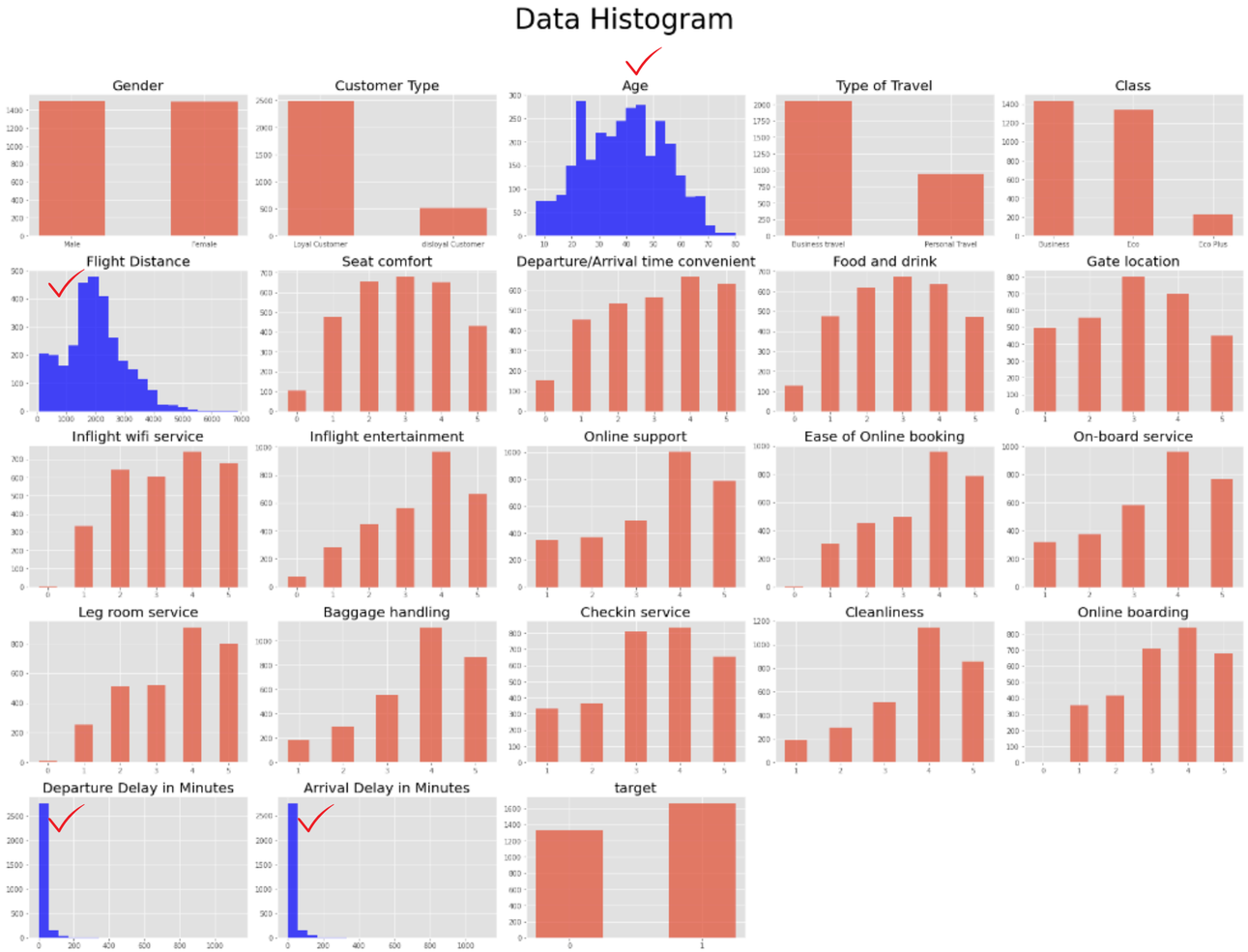
```
def check_missing_col(dataframe):
    missing_col = []
    for col in dataframe.columns:
        missing_values = sum(dataframe[col].isna())
        is_missing = True if missing_values >= 1 else False
        if is_missing:
            print(f'결측치가 있는 컬럼은: {col} 입니다')
            print(f'해당 컬럼에 총 {missing_values} 개의 결측치가 존재합니다.')
            missing_col.append([col, dataframe[col].dtype])
    if missing_col == []:
        print('결측치가 존재하지 않습니다')
    return missing_col

missing_col = check_missing_col(data)
```

정확한 분석을 하기 위해,
결측치(누락 값)가 있다면 결측치 삭제,
평균 보간법, 최빈값 보간법 등의 방법으로 처리합시다.

3 EDA 예시

EDA 통한 데이터 분포 및 관계 확인



전체 변수들을 히스토그램을 사용하여 시각화한 예시입니다.

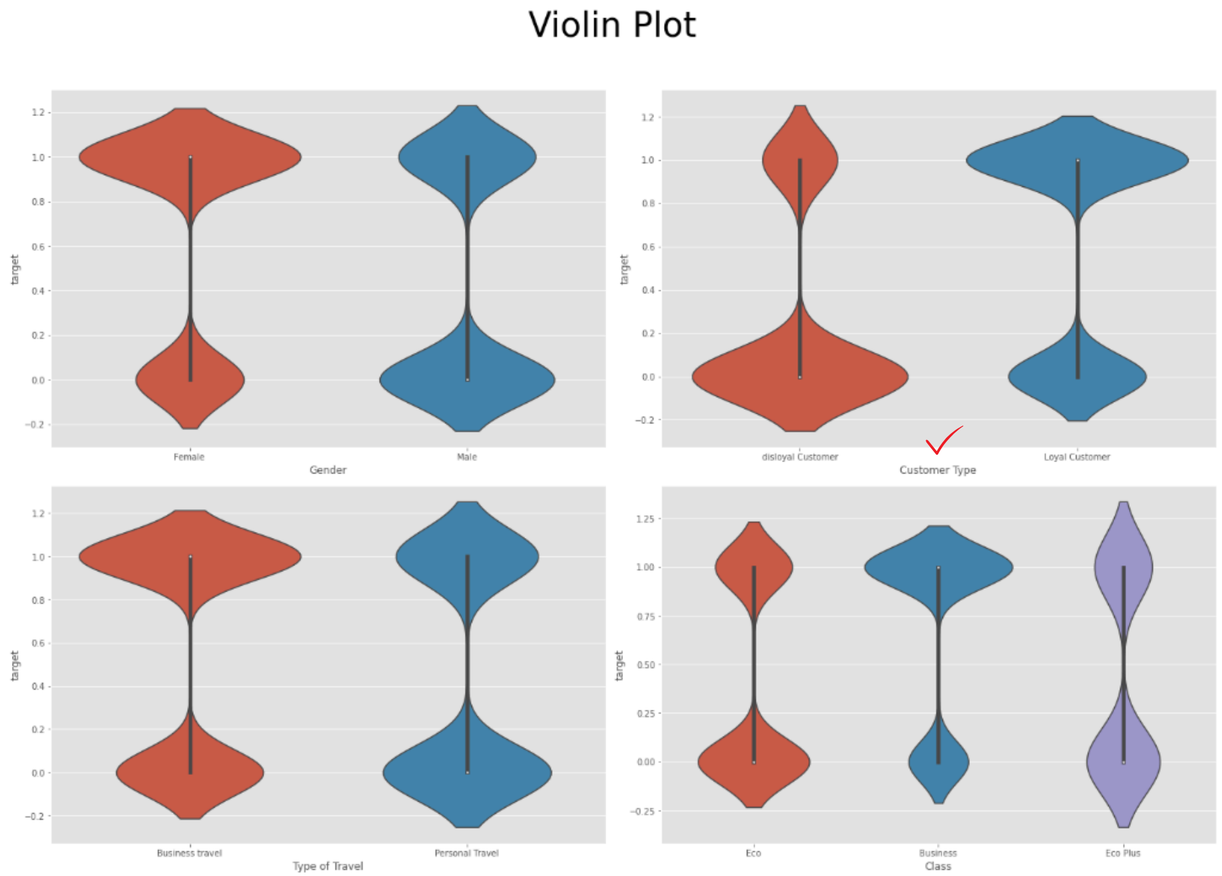
이를 통해, 위 데이터에서 각 변수들의 분포를 살펴보고 변수 간의 관계를 추론해봅시다.

연속형 변수들을 파란색으로 표기하여 살펴본 결과, AGE는 정규 분포 형태를 이루는 것을 확인할 수 있습니다.

추가로 AGE를 제외한 연속형 변수들이 대개 왼쪽으로 치우쳐져 있다는 것을 확인할 수 있습니다.

3 EDA 예시

EDA 통한 데이터 분포 및 관계 확인

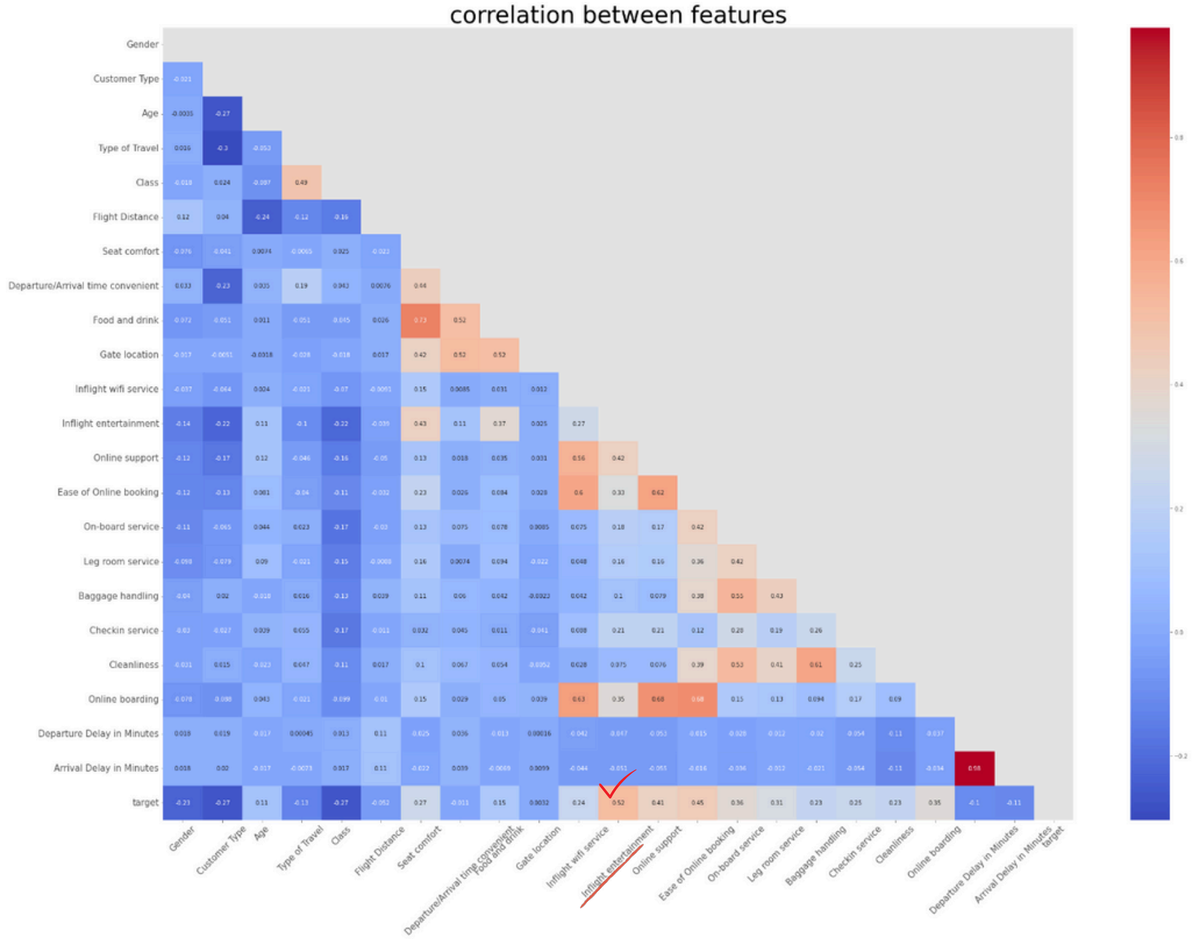


violin plot을 통해 각 독립변수에서의 카테고리에 따른 종속변수(target) 분포를 살펴봅시다.

customer type에 따라 만족도(target)에서 가장 큰 차이를 내는 것을 확인할 수 있습니다.

3 EDA 예시

상관관계 시각화를 통한 분석



상관관계 시각화를 통해 변수 간 관계를 살펴봅시다.

만족도(target)이 가장 높은 상관성을 보이는 변수는
Inflight entertainment인 것을 확인할
수 있습니다.

online support, ease of online booking
등과도 높은 상관을 보이고 있는데, 이렇듯 각 변수들에 대해
다양한 분석과 해석을 통해 여러분만의 인사이트를 얻어봅시다.

4 모델링 예시

데이터 전처리

```
train_x = train.drop(["id","target"],axis=1)
train_y = train.target

#라벨인코딩을 하기 위함 dictionary map 생성 함수
def make_label_map(dataframe):
    label_maps = {}
    for col in dataframe.columns:
        if dataframe[col].dtype=='object':
            label_map = {'unknown':0}
            for i, key in enumerate(dataframe[col].unique()):
                label_map[key] = i+1
            label_maps[col] = label_map
    return label_maps

# 각 범주형 변수에 인코딩 값을 부여하는 함수
def label_encoder(dataframe, label_map):
    for col in dataframe.columns:
        if dataframe[col].dtype=='object':
            dataframe[col] = dataframe[col].map(label_map[col])
            dataframe[col] = dataframe[col].fillna(label_map[col]['unknown'])
    return dataframe

# train 데이터 라벨 인코딩
label_map = make_label_map(train_x) # train 사용해 label map 생성
train_x = label_encoder(train_x, label_map) # train 라벨 인코딩

train_x.head()
```

라벨 인코딩

- 먼저, 종속변수 예측에 필요없는 변수인 id 열을 제거해주고, 훈련 data 와 target 을 나누어줍니다.
- 이후 카테고리(범주형) 형식의 features 들을 수치형으로 바꿔주어야 합니다.
- 본 데이터에서는 'Gender', 'Customer Type','Type of Travel', 'Class' 의 범주형 feature 들이 있습니다.
- 이러한 데이터의 라벨을 숫자로 바꾸어주는 전처리를 라벨 인코딩 (label encoding) 이라고 합니다.
- 많은 머신러닝 알고리즘들이 범주형 데이터 직접 처리하지 못하고, 모델 성능은 데이터 간의 거리를 계산하는데 의존하기 때문에 라벨 인코딩을 수행해주어야 합니다.

4 모델링 예시

정규화

```
from sklearn.preprocessing import MinMaxScaler

num_features = ['Age','Flight Distance','Departure Delay in Minutes','Arrival Delay in Minutes']

scaler = MinMaxScaler()
train_x[num_features] = scaler.fit_transform(train_x[num_features])
train_x.head()
```

정규화

- 모델은 학습 과정에서 데이터의 특성(feature)들을 추출해서 학습을 진행합니다.
- 하지만 학습을 하는 과정에서 데이터의 값이 너무 크거나, 분산이 너무 크면 학습 과정에 악영향을 끼칠 수 있습니다.
- 따라서 정규화를 통해 각 특성들의 중요도를 균일하게 만들어 모델의 학습 속도를 높이고, 데이터를 왜곡되게 해석하는 것을 방지합니다.
- Min-Max 정규화는 수치형 데이터의 값을 0~1 사이의 값으로 변환해줍니다.

4 모델링 예시

모델 설계 - LogisticRegression

```python
import numpy as np

np.random.seed(1)

class LogisticRegression:

    # sigmoid 함수를 생성합니다
    def sigmoid(self, z):
        return 1 / (1 + np.exp(-z))

    # lossfucntion 을 생성합니다
    def loss_function(self, X, y, weights):
        total = len(X)
        z = np.dot(X, weights)
        prediction1 = y * np.log(self.sigmoid(z))
        prediction2 = (1 - y) * np.log(1 - self.sigmoid(z))
        return -sum(prediction1 + prediction2) / total

    #gradient decent 방법을 이용해 학습 함수를 생성합니다
    #learning rate 와 epochs 파라미터를 지정합니다
    def fit(self, X, y, epochs=500, lr=0.01):
        loss = []
        weights = np.random.rand(X.shape[1])
        total = len(X)

        for epoch in range(epochs):
            y_hat = self.sigmoid(np.dot(X, weights))
            weights = weights - (lr * np.dot(X.T,  y_hat - y) / total)
            loss.append(self.loss_function(X, y, weights))

        self.weights = weights
        self.loss = loss

    def predict(self, X):
        z = np.dot(X, self.weights)
        result = self.sigmoid(z)
        return result

    def to_bin(self,result):
        arr=[]
        for i in result:
            if i >0.5:
                arr.append(1)
            else :
                arr.append(0)
        return arr
```

LogisticRegression의 특징

1. 확률 예측

LogisticRegression는 각 데이터 포인트가 특정 클래스에 속할 확률을 예측합니다. 예를 들어, 이진 분류 문제에서 로지스틱 회귀는 각 데이터 포인트가 클래스 1에 속할 확률을 계산하고, 이 확률을 기준으로 클래스 레이블을 결정합니다. 확률 예측은 상황에 맞게 임계값을 조정하여 분류기의 민감도(실제 긍정 추출율)와 특이도(실제 부정 추출율)를 조정할 수 있습니다.

2. 해석 가능성

LogisticRegression의 가중치(weights)는 feature가 종속 변수에 미치는 영향을 해석할 수 있게 해줍니다. 가중치의 부호와 크기를 통해 각 feature가 예측 결과에 어떻게 영향을 미치는지 알 수 있습니다.

3. 계산 효율성

LogisticRegression는 단순한 선형 결합(입력값)과 시그모이드 함수로 구성되어 있어 계산적으로 효율적입니다. 또한, LogisticRegression는 경사 하강법(Gradient Descent)과 같은 최적화 알고리즘을 사용하여 쉽게 학습할 수 있습니다.

4 모델링 예시

모델 설계 - LogisticRegression

```python
import numpy as np

np.random.seed(1)

class LogisticRegression:

    # sigmoid 함수를 생성합니다
    def sigmoid(self, z):
        return 1 / (1 + np.exp(-z))

    # lossfucntion 을 생성합니다
    def loss_function(self, X, y, weights):
        total = len(X)
        z = np.dot(X, weights)
        prediction1 = y * np.log(self.sigmoid(z))
        prediction2 = (1 - y) * np.log(1 - self.sigmoid(z))
        return -sum(prediction1 + prediction2) / total

    #gradient decent 방법을 이용해 학습 함수를 생성합니다
    #learning rate 와 epochs 파라미터를 지정합니다
    def fit(self, X, y, epochs=500, lr=0.01):
        loss = []
        weights = np.random.rand(X.shape[1])
        total = len(X)

        for epoch in range(epochs):
            y_hat = self.sigmoid(np.dot(X, weights))
            weights = weights - (lr * np.dot(X.T,  y_hat - y) / total)
            loss.append(self.loss_function(X, y, weights))

        self.weights = weights
        self.loss = loss

    def predict(self, X):
        z = np.dot(X, self.weights)
        result = self.sigmoid(z)
        return result

    def to_bin(self,result):
        arr=[]
        for i in result:
            if i >0.5:
                arr.append(1)
            else :
                arr.append(0)
        return arr
```

Sigmoid Function

- Logistic Regression 의 기초가 되는 요소는 Sigmoid 함수 입니다.
- Sigmoid 함수는 분류 모델에서 입력 값을 0과 1 사이의 확률 값으로 변환해주는 비선형 함수입니다.
- Logistic Regression 모델에서의 입력 값을 z 라고 하면, z 는 데이터의 feature(x) 들과 학습을 통해 정해진 weights(w) 들로 이루어져 있습니다.
- Logostic Regression 모델은 이러한 z 를 입력 값으로 받아, Sigmoid 함수에 입력해 0 과 1 사이의 값을 출력하는 함수입니다.
- 이때 모델이 학습하는 것은 데이터의 feature 들에 따른 weights 들 입니다.
- 그 후 학습을 통해 Loss Function 의 결과가 최소가 되는 방향으로 weights 들이 수정됩니다.

Loss Functinon(손실 함수)

- Loss Function 은 실제 값과 모델이 예측한 값의 거리(차이)를 출력하는 함수 입니다.
- 쉽게 말해 Loss Function 은 모델의 예측이 얼마나 틀렸는지를 알려주는 함수 입니다.
- 이 때 "모델의 예측이 얼마나 틀렸는지" 를 어떻게 정의하느냐에 따라 어떤 Loss Function 을 사용할 지가 정해집니다.
- 그 중 분류 문제에서 널리 사용되는 손실함수는 CrossEntropy Loss Function 입니다.

Gradient Desent(경사 하강법)

- 이제 Loss Function 을 최소화할 방법이 필요 합니다.
- 다시 말해 어떠한 방식을 활용해 Loss Function 의 결과가 최소화되는 입력값 weights 를 찾을 지 생각해 보아야 합니다.
- 즉, 모델의 특성을 잘 학습할 수 있도록 해야합니다.
- Logistic Regression 모델은 gradient descent 라는 방식으로 학습을 합니다.
- Loss Function에 입력된 weights의 기울기를 통해 weights를 수정할 방향성을 찾고, Loss Function 상에서 더 낮은 지점으로 나아갈 수 있도록 합니다.

Learning Rate(학습률)

- 이때 특정 방향으로 얼마나 움직일지 결정하는 요인이 Learning Rate 입니다.
- Learning Rate 이 작을수록 weights 를 조금씩 수정해 Loss Function 상에서 조금씩 이동하겠다는 의미입니다.

Epochs

- Epochs 는 학습 과정에서 weights를 총 몇 번을 이동해야할 지 결정하는 요인입니다.
- 즉, 전체 데이터셋에 대해 학습을 몇 번 할 지를 결정하는 것입니다.

4 모델링 예시

모델 학습 및 TARGET값 예측

```
test = pd.read_csv('test_sampled_final_v2.csv')
test = test.drop(["id"],axis=1)

test = label_encoder(test, label_map) #test data 라벨 인코딩

test[num_features] = scaler.transform(test[num_features]) #test 데이터 정규화
```

```
lr = LogisticRegression()
lr.fit(train_x,train_y)
```

```
pred = lr.predict(test)
pred_bin = lr.to_bin(pred)
```

```
sample_submission = pd.read_csv('./sample_submission.csv')

sample_submission.target = pred_bin
sample_submission.to_csv("submission.csv",index=False)
```

모델 학습

- 객체에 Logistic Regression 클래스를 할당 후, 클래스 내 fit 메소드를 이용해 학습을 진행하면 됩니다.

추론

- 모델을 학습한 후 test 데이터를 예측하는 것입니다.
- test 데이터에도 train 데이터에 했던 전처리를 진행한 후, 추론을 진행하면 됩니다.(data leakage 에 주의하도록 합니다!)
- LogisticRegression.predict() 는 Sigmoid Function 의 출력 값을 출력합니다.
- 따라서 예측 값들을 threshold 0.5 기준으로 0 또는 1 의 값으로 변환해주어야 합니다.

5 Q&A

궁금한 것이 있으면 질문해주세요!