Model Evaluation

Overview

왜 모델을 검증해야 하는가?

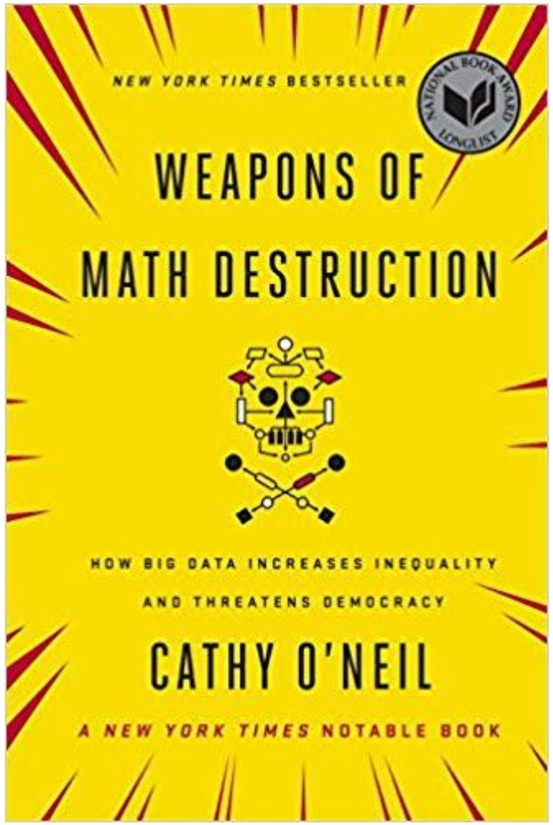
우리가 개선한 모델이 더 정확해졌는지 판단할 수 있는 근거가 있어야만 우리는 모델을 고칠 수 있다.

"통계 시스템에서 양만큼 필수적인 것이 있다. 바로 **피드백**이다. 피드백은 시스템이 정상 항로에서 벗어날 경우, 이를 알려주는 장치다. 통계 전문가들은 피드백을 통해 받은 오류를 토대로 모형을 개선해 더욱 완벽하게 만든다."

…

"워싱턴 교육청의 가치부가모형을 포함해 이 책에서 소개할 WMD(Weapons of Math Destruction, 대량살상수학무기)중 상당수가 적절한 피드백을 받지 못하고 있다. … 매스매티카의 평가 시스템이 와이사키와 205명의 교사들에게 실패자라는 꼬리표를 붙이자 워싱턴 교육 당국은 그들을 모두 해고했다. **그런데 이 평가 시스템에는 이 같은 결정이 옳은지에 대해 사후에 학습가는 과정이 있을까? 없다. 시스템이 교사들을 실패자라고 확신하면, 평가는 그것으로 끝이다.**"

대량살상수학무기(Weapons of Math Destruction)
캐시 오닐(Cathy O'Neil) 저



모델 검증 방법 - 온라인 검증(Online Evaluation)

라이브 검증(Live Evaluation)이라고도 부른다. 서비스에 직접 올려서 검증하는 방식
가장 정확한 지표이지만, 측정하는데 다소 시간이 걸린다

온라인 검증에서 쓰이는 지표의 예시

- 회사의 매출
- 월 방문자 수
- 방문자가 서비스에 머무르는 시간
- 방문한 사용자가 물건을 구매할 확률
- 추천한 광고를 사용자가 클릭할 확률
- 서브스크립션 모델을 이용중인 사용자가 구매를 중단할 확률(일명 이탈률, churn rate)

온라인 검증에서 가장 많이 쓰이는 방식 중에 하나인 A/B 테스팅
기존 모델을 Variation A에, 개선한 모델을 Variation B에 도입한 뒤
Key Metric의 차이를 비교한다

모델 검증 방법 - 오프라인 검증(Offline Evaluation)

서비스에 올리지 않고 검증한다
온라인 검증에 비해 정확한 측정 지표는 아니나, 측정 속도가 매우 빠르다

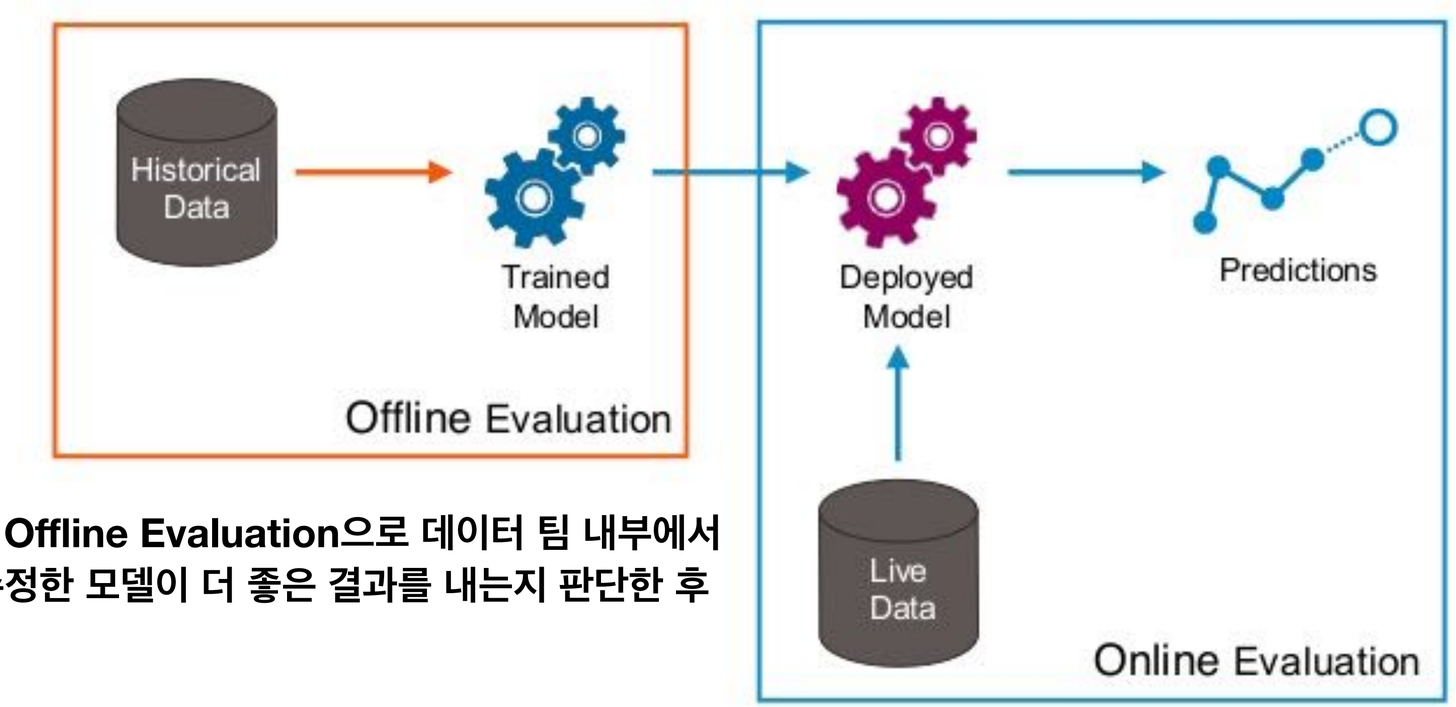
```
import numpy as np

y_predict = model.predict(X_test)
y_predict = np.argmax(y_predict, axis=1)

accuracy = (y_test == y_predict).mean()

print("Accuracy = {0:.5f}".format(accuracy))
```

```
Accuracy = 0.98420
```

우리가 지금까지 모델의 정확도를 측정하기 위한 방식을
전문용어로 오프라인 검증(Offline Evaluation)이라 부른다

모델 검증 방법

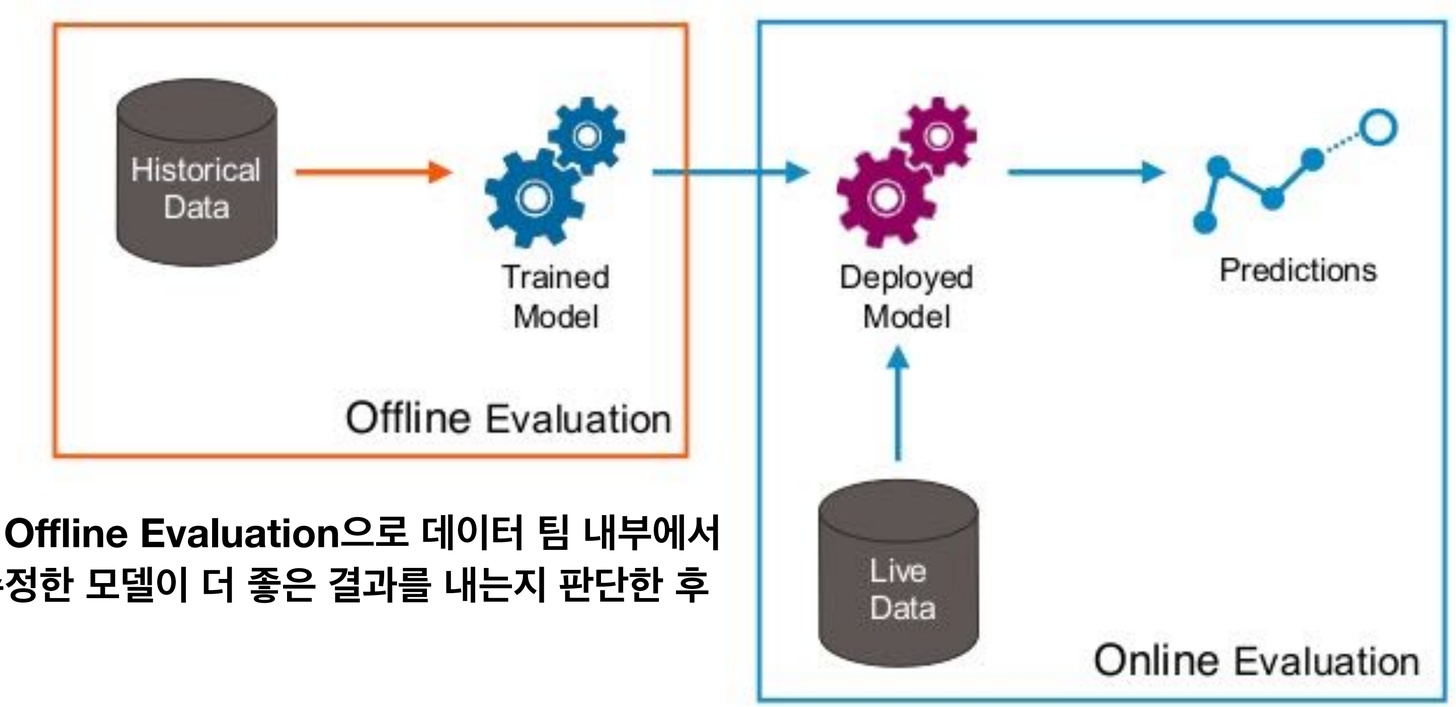
**모든 데이터사이언티스트 팀은 오프라인 검증을 바탕으로 모델을 개선한 뒤,
개선 결과를 실 서비스에 반영하여 온라인 검증을 한다**



1. Offline Evaluation으로 데이터 팀 내부에서 수정한 모델이 더 좋은 결과를 내는지 판단한 후

모델 검증 방법

모든 데이터사이언티스트 팀은 오프라인 검증을 바탕으로 모델을 개선한 뒤,
개선 결과를 실 서비스에 반영하여 온라인 검증을 한다

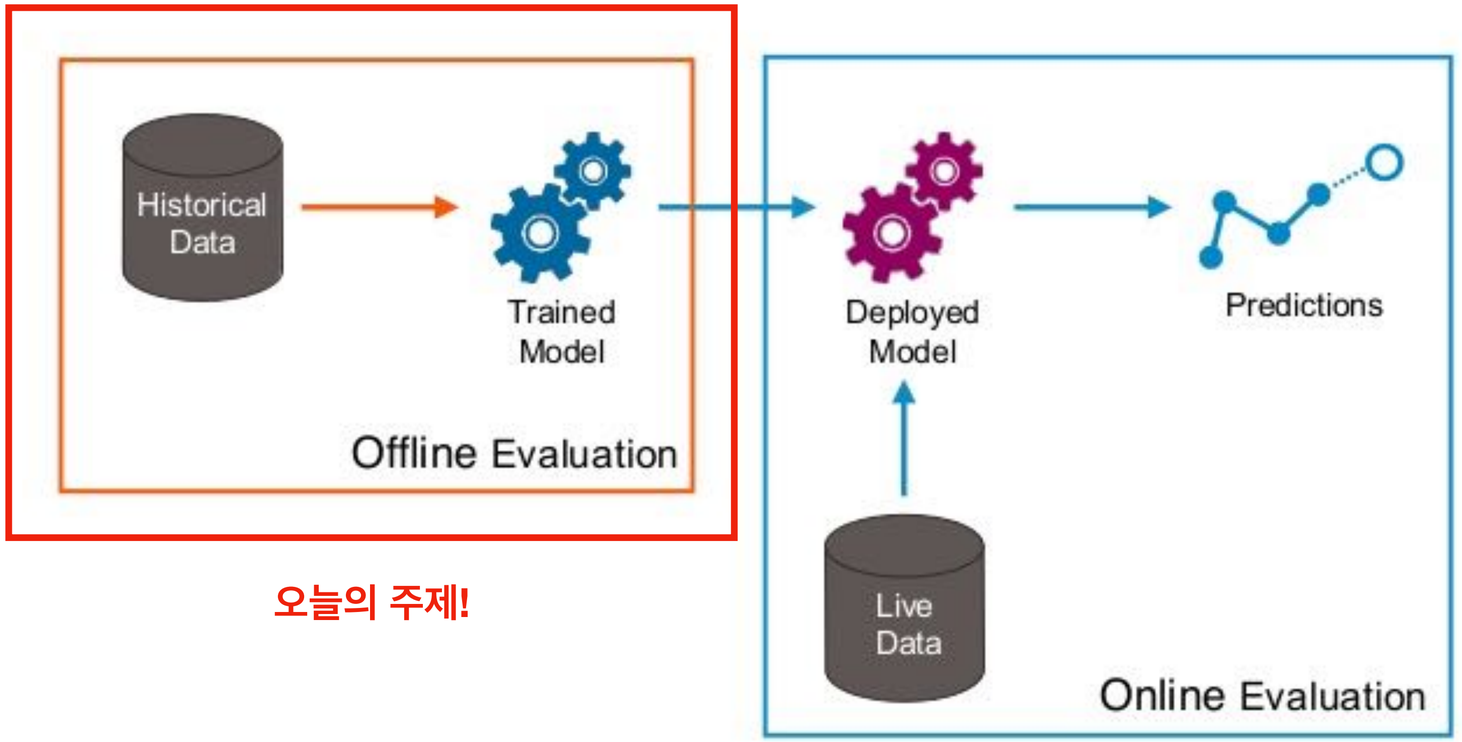


1. Offline Evaluation으로 데이터 팀 내부에서
수정한 모델이 더 좋은 결과를 내는지 판단한 후

2. Online Evaluation으로 실 서비스에서
비즈니스 지표를 기준삼아 확실히 검증한다

모델 검증 방법

모든 데이터사이언티스트 팀은 오프라인 검증을 바탕으로 모델을 개선한 뒤,
개선 결과를 실 서비스에 반영하여 온라인 검증을 한다



오늘의 주제!

Offline Evaluation Mechanisms

기본적인 정량적 검증 방법 - Hold Out Validation

데이터를 두 개로 나눈 뒤 하나는 학습용, 하나는 검증용으로 사용한다
가장 간단한 방법이지만, 학습용과 검증용 데이터가 균등하게 배분되어야 한다

데이터 갯수: 10,000 개

8,000 개

2,000 개

데이터를 큰 조각과 작은 조각으로 두 조각을 낸 뒤,
큰 조각으로 알고리즘을 학습하고, 작은 조각을 평가한다.

```
from keras.datasets import mnist

((X_train, y_train), (X_test, y_test)) = mnist.load_data()

print(X_train.shape, y_train.shape)
print(X_test.shape, y_test.shape)
```

```
Using TensorFlow backend.
```

```
(60000, 28, 28) (60000,)
(10000, 28, 28) (10000,)
```

대부분의 딥러닝용 데이터셋은
hold-out validation을 위한 train 데이터셋과 test 데이터셋으로 나뉘어져 있다.

기본적인 정량적 검증 방법 - Cross Validation

데이터를 N조각으로 나눠서 N조각 만큼 학습-검증을 반복한다
데이터가 균등하게 분배되어야 할 필요가 없지만, 검증하는데 그만큼 시간이 오래 걸린다

데이터 갯수: 10,000 개

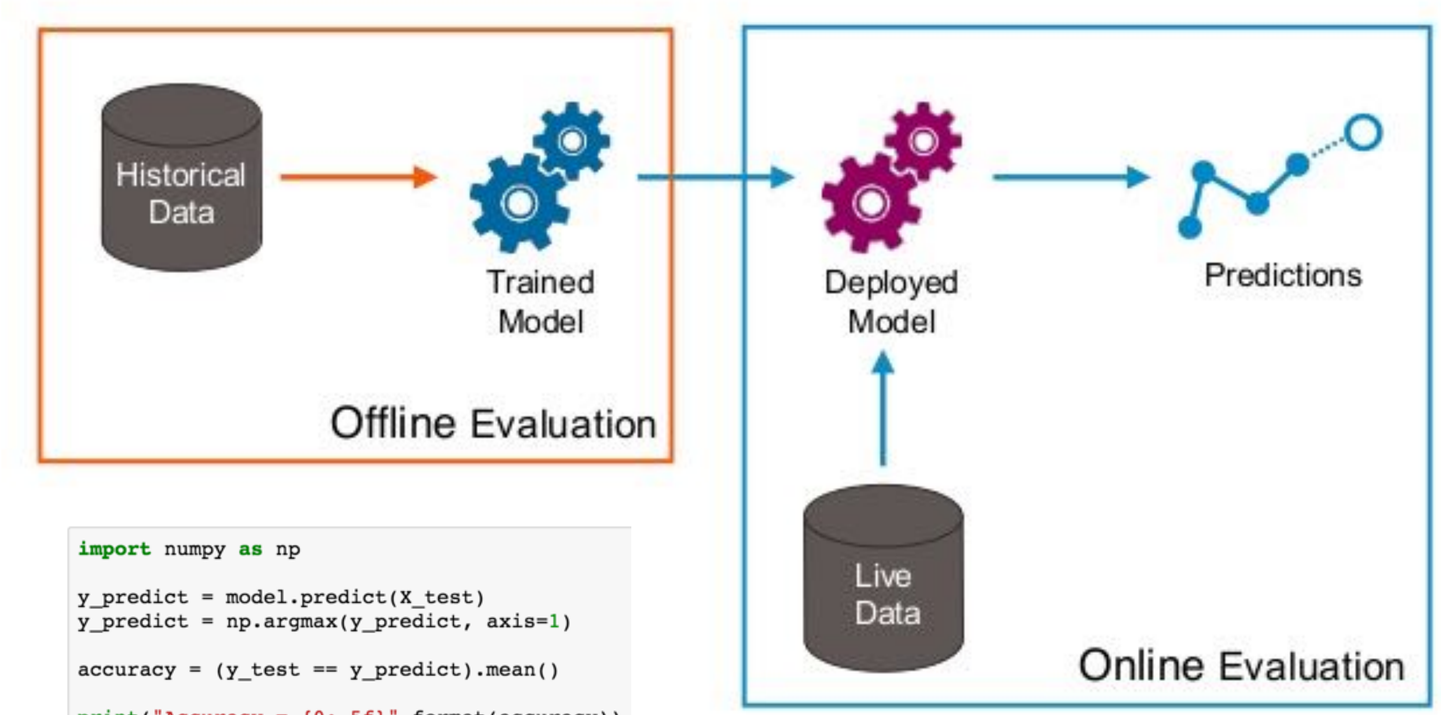
데이터를 N조각 낸 뒤(ex: 5조각),
조각을 뺀 나머지로 알고리즘을 학습하고, 조각을 예측한다.
이후 그 결과값의 평균을 낸다.

1. B + C + D + E → A
2. A + C + D + E → B
3. A + B + D + E → C
4. A + B + C + E → D
5. A + B + C + D → E

평균(mean)을 구한다

모델을 정량적으로 검증하는데 있어서 고려해야 하는 것

오프라인 지표가 안 좋아질 때 온라인 지표도 안 좋아저야 하며, 오프라인 지표가 좋아질 때 온라인 지표도 좋아저야 한다
이 두 개가 서로 맞지 않으면 모든 데이터사이언티스트들이 실험을 할 수 없다



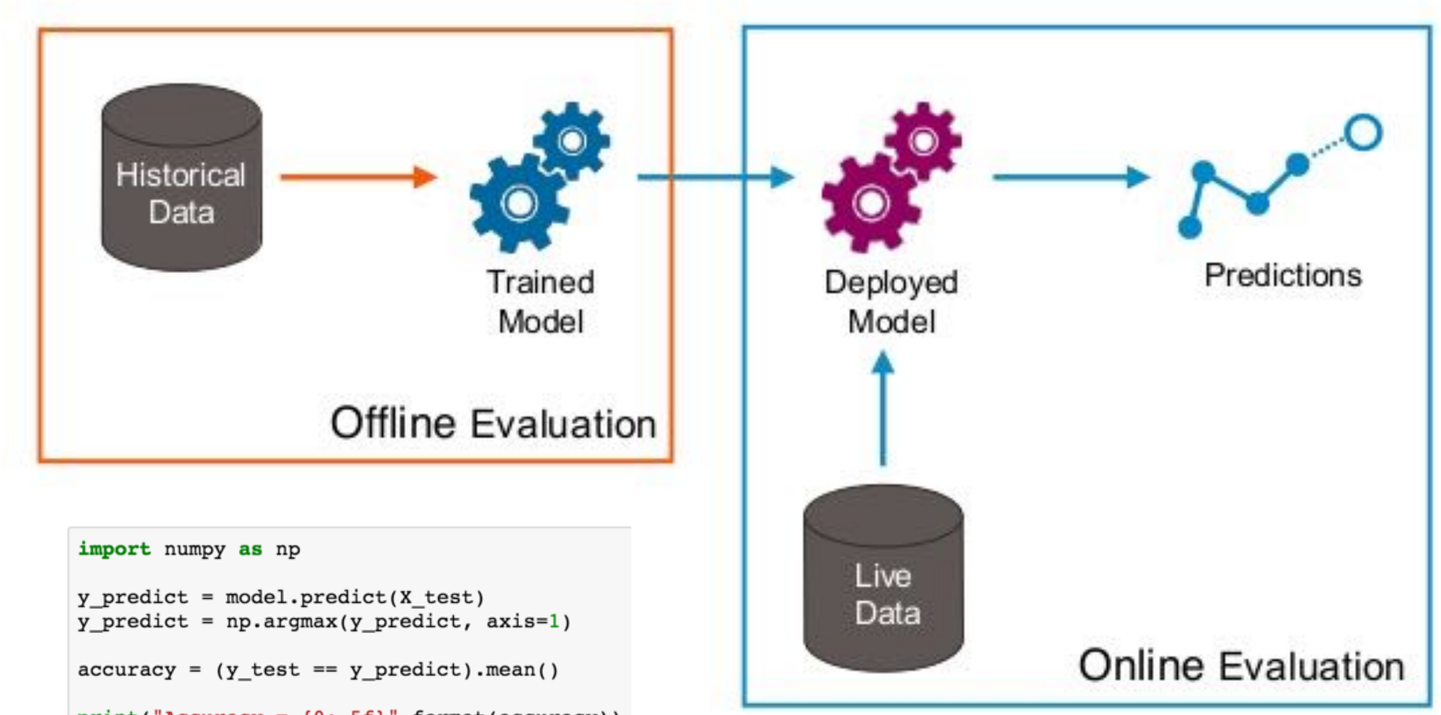

```
import numpy as np

y_predict = model.predict(X_test)
y_predict = np.argmax(y_predict, axis=1)

accuracy = (y_test == y_predict).mean()

print("Accuracy = {0:.5f}".format(accuracy))
```

```
Accuracy = 0.99370(+0.00950)
```

어, 정확도가 더 올라갔네?
빨리 실 서비스에 올려서 매출을 늘려야지!

모델을 정량적으로 검증하는데 있어서 고려해야 하는 것

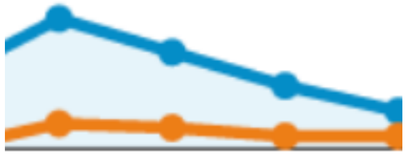
오프라인 지표가 안 좋아질 때 온라인 지표도 안 좋아저야 하며, 오프라인 지표가 좋아질 때 온라인 지표도 좋아저야 한다
이 두 개가 서로 맞지 않으면 모든 데이터사이언티스트들이 실험을 할 수 없다




```
import numpy as np

y_predict = model.predict(X_test)
y_predict = np.argmax(y_predict, axis=1)

accuracy = (y_test == y_predict).mean()

print("Accuracy = {0:.5f}".format(accuracy))
```

```
Accuracy = 0.99370(+0.00950)
```

어, 정확도가 더 올라갔네?
빨리 실 서비스에 올려서 매출을 늘려야지!



헉, 매출이 떨어진다 ㅠㅠ

모델을 정량적으로 검증하는데 있어서 고려해야 하는 것

Classification과 Regression 문제. 서로 다른 측정 방식을 사용해야 한다

	actual	predict	error (accuracy)
0	10000	11616	0
1	10000	9244	0
2	10000	11645	0
3	10000	10000	1
4	10000	9036	0
5	10000	11451	0
6	10000	11938	0
7	10000	8271	0
8	10000	11420	0
9	10000	9551	0

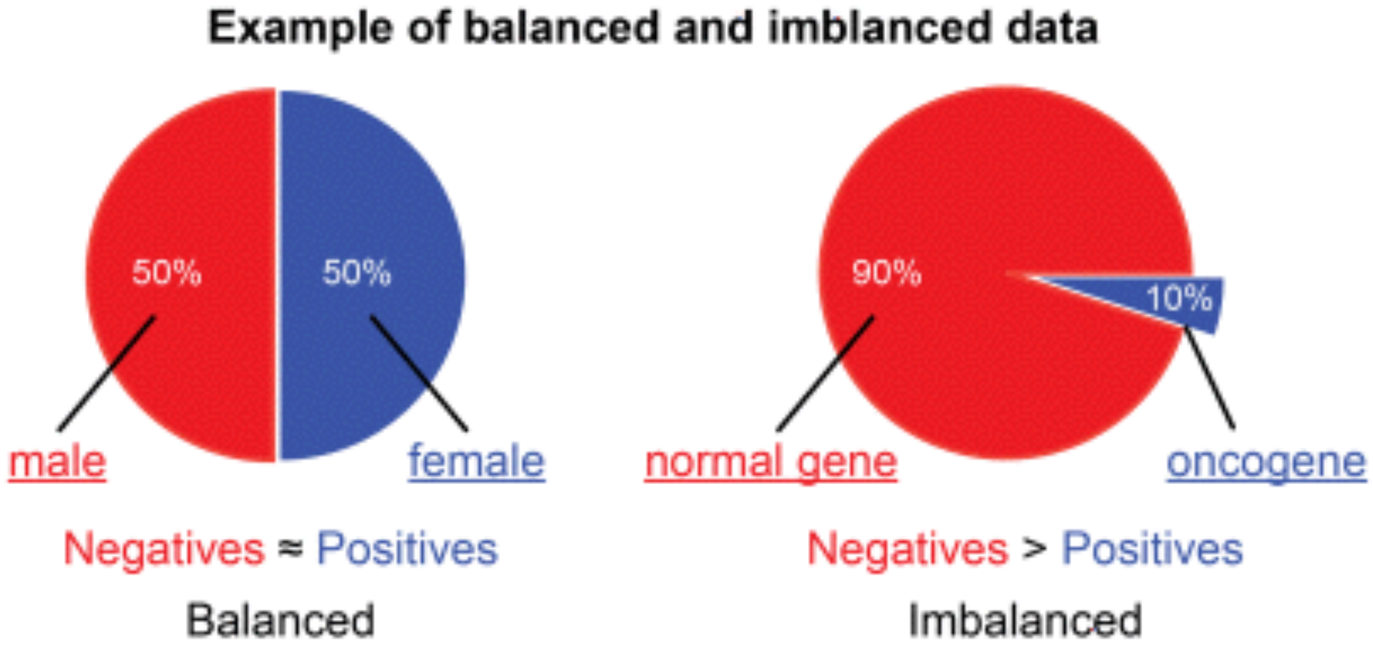
regression 문제를 classification metric으로 검증하려고 한다면,
아마 거의 모든 값이 오답이라고 예측할 것이다.

	actual	predict	error (accuracy)	error (MAE)
0	기타	기타	1	NaN
1	가전제품	가전제품	1	NaN
2	의류	의류	1	NaN
3	기타	의류	0	NaN
4	의류	의류	1	NaN
5	기타	기타	1	NaN
6	기타	기타	1	NaN
7	기타	기타	1	NaN
8	기타	의류	0	NaN
9	의류	의류	1	NaN

반면 classification 문제를 regression metric으로 검증하려고 한다면,
공식이 맞지 않기 때문에 에러가 나거나 제대로된 값이 나오지 않을 것이다

모델을 정량적으로 검증하는데 있어서 고려해야 하는 것

imbalanced 문제(또는 skewed 문제)
서로 다른 Label을 동일하게 간주해야 하는가?



모델을 정량적으로 검증하는데 있어서 고려해야 하는 것

서로 다른 결과를 자세히 비교하기

단순이 어떤 모델이 좋고 나쁜 것을 떠나서, 모델마다의 강점과 약점을 파악할 수 있어야 한다

VOC 2012 test	mAP	aero	bike	bird	boat	bottle	bus	car	cat	chair	cow	table	dog	horse	mbike	person	plant	sheep	sofa	train	tv
MR_CNN_MORE_DATA [11]	**73.9**	**85.5**	**82.9**	**76.6**	**57.8**	**62.7**	**79.4**	77.2	86.6	**55.0**	**79.1**	**62.2**	87.0	**83.4**	**84.7**	78.9	45.3	73.4	65.8	80.3	74.0
HyperNet_VGG	71.4	84.2	78.5	73.6	55.6	53.7	78.7	**79.8**	87.7	49.6	74.9	52.1	86.0	81.7	83.3	**81.8**	**48.6**	**73.5**	59.4	79.9	65.7
HyperNet_SP	71.3	84.1	78.3	73.3	55.5	53.6	78.6	79.6	87.5	49.5	74.9	52.1	85.6	81.6	83.2	81.6	48.4	73.2	59.3	79.7	65.6
Fast R-CNN + YOLO	70.7	83.4	78.5	73.5	55.8	43.4	79.1	73.1	**89.4**	49.4	75.5	57.0	**87.5**	80.9	81.0	74.7	41.8	71.5	68.5	**82.1**	67.2
MR_CNN_S_CNN [11]	70.7	85.0	79.6	71.5	55.3	57.7	76.0	73.9	84.6	50.5	74.3	61.7	85.5	79.9	81.7	76.4	41.0	69.0	61.2	77.7	72.1
Faster R-CNN [28]	70.4	84.9	79.8	74.3	53.9	49.8	77.5	75.9	88.5	45.6	77.1	55.3	86.9	81.7	80.9	79.6	40.1	72.6	60.9	81.2	61.5
DEEP_ENS_COCO	70.1	84.0	79.4	71.6	51.9	51.1	74.1	72.1	88.6	48.3	73.4	57.8	86.1	80.0	80.7	70.4	46.6	69.6	**68.8**	75.9	71.4
NoC [29]	68.8	82.8	79.0	71.6	52.3	53.7	74.1	69.0	84.9	46.9	74.3	53.1	85.0	81.3	79.5	72.2	38.9	72.4	59.5	76.7	68.1
Fast R-CNN [14]	68.4	82.3	78.4	70.8	52.3	38.7	77.8	71.6	89.3	44.2	73.0	55.0	**87.5**	80.5	80.8	72.0	35.1	68.3	65.7	80.4	64.2
UMICH_FGS_STRUCT	66.4	82.9	76.1	64.1	44.6	49.4	70.3	71.2	84.6	42.7	68.6	55.8	82.7	77.1	79.9	68.7	41.4	69.0	60.0	72.0	66.2
NUS_NIN_C2000 [7]	63.8	80.2	73.8	61.9	43.7	43.0	70.3	67.6	80.7	41.9	69.7	51.7	78.2	75.2	76.9	65.1	38.6	68.3	58.0	68.7	63.3
BabyLearning [7]	63.2	78.0	74.2	61.3	45.7	42.7	68.2	66.8	80.2	40.6	70.0	49.8	79.0	74.5	77.9	64.0	35.3	67.9	55.7	68.7	62.6
NUS_NIN	62.4	77.9	73.1	62.6	39.5	43.3	69.1	66.4	78.9	39.1	68.1	50.0	77.2	71.3	76.1	64.7	38.4	66.9	56.2	66.9	62.7
R-CNN VGG BB [13]	62.4	79.6	72.7	61.9	41.2	41.9	65.9	66.4	84.6	38.5	67.2	46.7	82.0	74.8	76.0	65.2	35.6	65.4	54.2	67.4	60.3
R-CNN VGG [13]	59.2	76.8	70.9	56.6	37.5	36.9	62.9	63.6	81.1	35.7	64.3	43.9	80.4	71.6	74.0	60.0	30.8	63.4	52.0	63.5	58.7
YOLO	57.9	77.0	67.2	57.7	38.3	22.7	68.3	55.9	81.4	36.2	60.8	48.5	77.2	72.3	71.3	63.5	28.9	52.2	54.8	73.9	50.8
Feature Edit [33]	56.3	74.6	69.1	54.4	39.1	33.1	65.2	62.7	69.7	30.8	56.0	44.6	70.0	64.4	71.1	60.2	33.3	61.3	46.4	61.7	57.8
R-CNN BB [13]	53.3	71.8	65.8	52.0	34.1	32.6	59.6	60.0	69.8	27.6	52.0	41.7	69.6	61.3	68.3	57.8	29.6	57.8	40.9	59.3	54.1
SDS [16]	50.7	69.7	58.4	48.5	28.3	28.8	61.3	57.5	70.8	24.1	50.7	35.9	64.9	59.1	65.8	57.1	26.0	58.8	38.6	58.9	50.7
R-CNN [13]	49.6	68.1	63.8	46.1	29.4	27.9	56.6	57.0	65.9	26.5	48.7	39.5	66.2	57.3	65.4	53.2	26.2	54.5	38.1	50.6	51.6

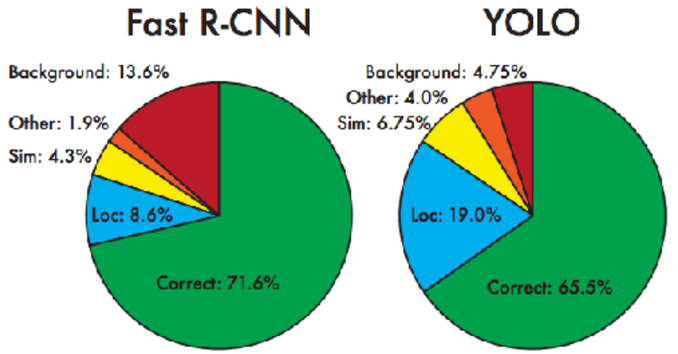


Figure 4: Error Analysis: Fast R-CNN vs. YOLO These charts show the percentage of localization and background errors in the top N detections for various categories (N = # objects in that category).

유명 Object Detection모델인 YOLO(You Only Look Once)의 벤치마킹 결과

물체의 종류(ex: 새, 자전거, 고양이)마다의 정확도를 전부 분석함으로서 타 모델과의 성능 차이를 비교한다

You Only Look Once: Unified, Real-Time Object Detection

Redmon et al, 2015. https://goo.gl/DY4AeY

Classification Metics

Classification Metics - Accuracy

일명 퍼센티지. (%) 가장 보편적으로 쓰이는 지표이다
직관적이지만, imbalanced된 데이터셋에 약하다

장점

- 매우 직관적이다. 사실상 모르는 사람이 없을 정도

단점

- imbalanced 된 데이터셋에 약하다. 가령오른쪽 예시에서는 의류와 가전제품을 못 맞췄을 경우 동일하게 점수가 낮아지는데, 암 환자와 암 환자가 아닌 경우에도 동등하게 처리해야 하나?

$$\text{accuracy}(y, \hat{y}) = \frac{1}{n_{\text{samples}}} \sum_{i=0}^{n_{\text{samples}}-1} 1(\hat{y}_i = y_i)$$

```
from sklearn.metrics import accuracy_score

actual = ["의류", "가전제품", "의류", "기타"]
predict = ["의류", "가전제품", "기타", "가전제품"]

accuracy_score(actual, predict)
```

```
0.5
```

Classification Metics - Confusion Matrix

모든 클래스(또는 Label)를 동일하게 놓지 않고 서로 다르게 처리한다
imbalanced된 데이터셋에 강하지만, 결과가 하나의 숫자(single number)로 나오지 않는다

장점

- imbalanced된 데이터셋에 강하다.
- 정량적인 측정 뿐만 아니라, 모델이 구체적으로 어떤 부분에 약한지(가령 암 환자를 맞추는 기능이 약한지, 암 환자가 아니라는 사람을 맞추는 기능이 약한지) 알 수 있다.

단점

- 하나의 숫자로 나오지 않는다.

	Predicted as positive	Predicted as negative
Labeled as positive	80	20
Labeled as negative	5	195

다음의 네 가지를 나눠서 판단한다.

- 암 환자를 암 환자라고 맞췄을 경우 (True Positive)
- 암 환자를 암 환자가 아니라고 판단해서 못 맞췄을 경우 (False Negative)
- 암 환자가 아닌 사람을 암 환자라고 판단해서 못 맞췄을 경우 (False Positive)
- 암 환자가 아닌 사람을 암 환자가 아니라고 맞췄을 경우 (True Negative)

Classification Metics - Per Class Accuracy

각 클래스 마다의 accuracy를 더한 후 클래스의 갯수로 나눈다
imbalanced된 데이터셋에서도 결과가 하나의 숫자(single number)로 나온다

장점

- imbalanced된 데이터셋에 강하다.
- 하나의 숫자로 나오기 때문에 다른 모델과 비교하기 편하다.

단점

- 특정 Label의 갯수가 너무 작은 경우, (가령 암 환자가 5명밖에 없다면) 이를 정확히 맞추는 것은 통계적으로 매우 어려운 반면 Metric에 끼치는 영향력이 너무 크다. (한두 개만 못 맞춰도 accuracy가 너무 크게 감소한다)

	Predicted as positive	Predicted as negative
Labeled as positive	80	20
Labeled as negative	5	195

암 환자를 맞춘 확률: 80명 / (20명 + 80명) = 80%
암 환자가 아닌 사람을 맞춘 확률: 195명 / (5명 + 195명) = 97.5%

Per Class Accuracy: (80% + 97.5%) / 2 = 88.75%

Classification Metics - log loss

Per-class Accuracy와 마찬가지로 imbalanced된 데이터셋에서도 결과가 하나의 숫자로 나온다
차이는 Per-class accuracy는 예측 결과를 기준으로, log-loss는 각 클래스의 확률을 기준으로 계산한다

장점

- imbalanced된 데이터셋에 강하다.
- 단순히 Label을 맞추냐 못 맞추냐를 넘어서서, 각 Label의 확률을 기준으로 점수가 책정된다. (가령 두 모델이 똑같이 의류 카테고리를 못 맞춘다고 하더라도, A모델은 0.31, B모델은 0.49라고 하면 B모델이 점수가 더 좋게 나온다) 이를 soft-measurement라고 한다.

단점

- 특정 데이터 (내지는 특정 Label)을 못 맞췄을 경우 그 페널티가 기하급수적으로 커진다. (∞까지 올라간다)
- log-loss가 좋아졌다고 해도 온라인 지표가 변하지 않을 가능성도 있다.

$$\text{log-loss} = -\frac{1}{N}\sum_{i=1}^{N} y_i \log p_i + (1 - y_i) \log (1 - p_i)$$

우리가 이전에 본 cross-entropy와 유사하다
(또한 KL Divergence와도 유사하다)

```
from sklearn.metrics import log_loss

actual = ["의류", "가전제품", "의류", "기타"]
predict = [
#   각각
#   의류 / 가전제품 / 기타 에 속할 확률
    [0.7, 0.2, 0.1],
    [0.3, 0.5, 0.2],
    [0.5, 0.4, 0.1],
    [0.3, 0.3, 0.4],
]

log_loss(actual, predict)
```

```
1.7532789486599909
```

Regression Metics

Regression Metics - Mean Absolute Error(MAE)

두 값의 차이를 뺀 뒤 절대값을 씌운다
가장 기본적인 Regression Problem을 위한 측정 방식

$$\mathrm{MAE}(y, \hat{y}) = \frac{1}{n_{\text{samples}}} \sum_{i=0}^{n_{\text{samples}}-1} |y_i - \hat{y}_i| .$$

```
from sklearn.metrics import mean_absolute_error

actual = [1000, 1000, 1000, 1000]
predict = [900, 1000, 1100, 1200]

mean_absolute_error(actual, predict)
```

```
100.0
```

Regression Metics - Mean Squared Error(MSE)

두 값의 차이를 뺀 뒤 제곱을 한다
MAE에 비해서 차이가 클수록 더 페널티가 들어간다

$$\mathrm{MSE}(y, \hat{y}) = \frac{1}{n_{\text{samples}}} \sum_{i=0}^{n_{\text{samples}}-1} (y_i - \hat{y}_i)^2.$$

```
from sklearn.metrics import mean_squared_error

actual = [1000, 1000, 1000, 1000]
predict = [900, 1000, 1100, 1200]

mean_squared_error(actual, predict)
```

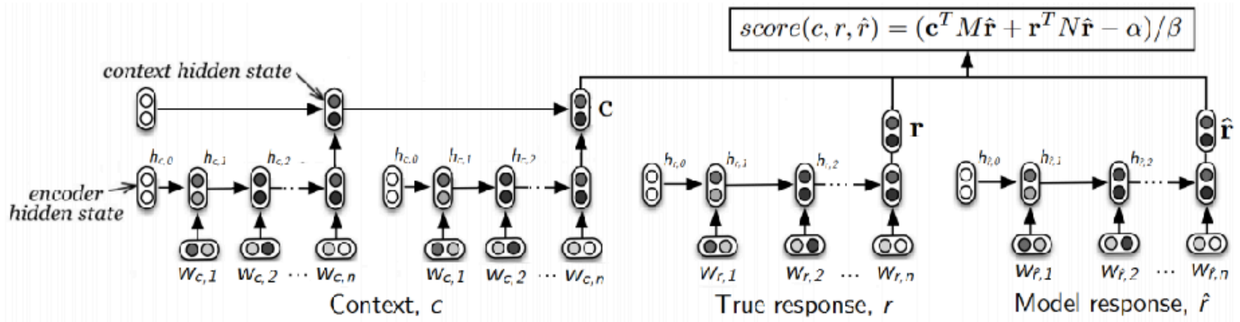
```
15000.0
```

Others

앞서 설명한 것은 가장 보편적인 예시이며,
실제로는 위 예시를 바탕으로 회사의 **Key Metric**에 맞게 튜닝해서 사용할 줄 알아야 한다

$$\epsilon = \sqrt{\frac{1}{n}\sum_{i=1}^{n}(\log(p_i + 1) - \log(a_i + 1))^2},$$

MSE(Mean Squared Error)와 달리, 차이가 클수록 페널티를 덜 주고 싶다면, **log(label + 1)**을 하는 방식으로 이를 구현할 수 있다



챗봇은 결과의 정확도를 정량적으로 평가하기 매우 어렵기 때문에,
오직 평가용으로 만든 딥러닝 모델로 결과를 평가하는 것도 가능하다.

Towards an Automatic Turing Test: Learning to Evaluate Dialogue Responses
Lowe et al., 2017. https://goo.gl/D4hZo9

Live Coding

Q & A