kaggle 2조 1주차 발표

발표자 : 오동훈

Ch.5 머신러닝 요약

다루는 내용은 다음과 같다.

- 평가지표 관련 : 회귀평가지표 / 정확도-정밀도, 재현율, F1 점수 / ROC, AUC
- 데이터 인코딩 : 레이블 인코딩, 원-핫 인코딩
- FEATURE SCALING : min-max normalization, standardization
- cross-validation : K-fold, stratified k-fold
- ML model
  - ▼ **선형회귀**

ISLR : Chapter 3.

About RSS : residual sum of squares

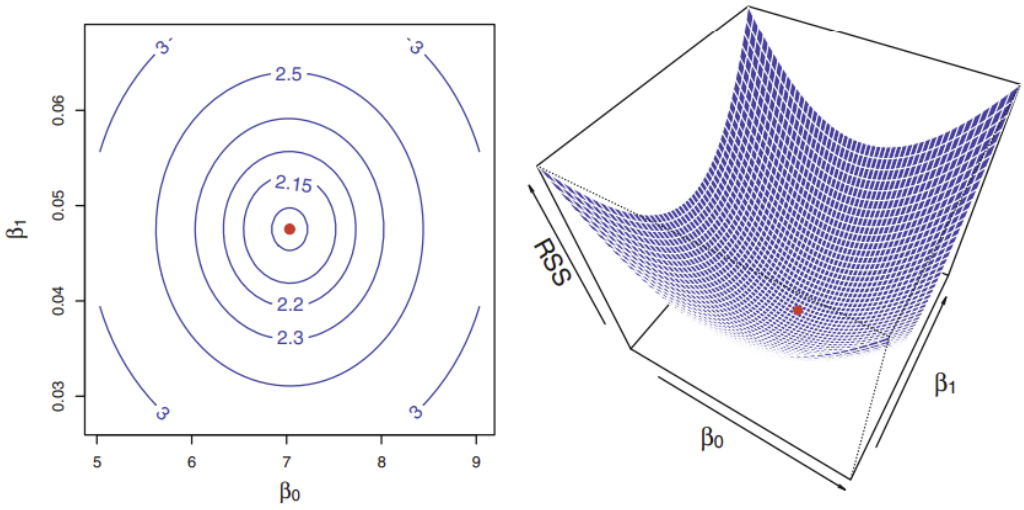
$RSS = e_1^2 + e_2^2 + ... + e_n^2,$ 이 수식은 잔차들의 제곱의 합을 의미한다. RSS를 위의 y 예측 수식으로 풀면 다음과 같다.

$$\text{RSS} = (y_1 - \hat{\beta}_0 - \hat{\beta}_1 x_1)^2 + (y_2 - \hat{\beta}_0 - \hat{\beta}_1 x_2)^2 + \ldots + (y_n - \hat{\beta}_0 - \hat{\beta}_1 x_n)^2.$$

least squares approach (최소자승법) 에 의해 도출되는 각 계수의 값은 다음과 같다.(계산 과정에서 각 요소에 대한 편미분 방식 사용.)

$$\hat{\beta}_1 = \frac{\sum_{i=1}^{n}(x_i - \bar{x})(y_i - \bar{y})}{\sum_{i=1}^{n}(x_i - \bar{x})^2},$$
$$\hat{\beta}_0 = \bar{y} - \hat{\beta}_1 \bar{x},$$

그래프로 시각화 하면 다음 이미지와 같다. 붉은 점이 global minimum 을 의미한다.



least squares approach 방식에 따라 도출된 β0 추정값과 β1 추정값은 샘플 평균을 사용한다. 이것이 의미하는 것은, 선형 회귀에서 우리가 모르는 값인 β0, β1 을 추정할 때, 모회귀 모형에서의 값으로 정의하는 것이 합리적이라는 것이다.

예를 들어 수 없이 많은 데이터들의 평균을 모를 때, 우리는 일부 n 개의 샘플만을 표본으로 뽑아 평균을 추출하고 그것을 모평균으로 추정하며 합리적인 방식으로 생각한다. 이와 유사한 방식이지만, 반대로, 일부 샘플의 평균을 모를 때 모평균으로 추정하는 것이다.

Assessing the accuracy of the model

선형 회귀 모델에 대한 정량적인 평가는 두 개의 수치를 이용한다. ‘residual standard error; RSE’ & R^2 계수. RSE 는 다음과 같이 계산한다.

$$\mathrm{RSE} = \sqrt{\frac{1}{n-2}\mathrm{RSS}} = \sqrt{\frac{1}{n-2}\sum_{i=1}^{n}(y_i - \hat{y}_i)^2}. \qquad (3.15)$$

$$\mathrm{RSS} = \sum_{i=1}^{n}(y_i - \hat{y}_i)^2.$$

RSE 에서 (n-2) 는 자유도를 의미한다.

RSE는 주어진 데이터에 대해 모델이 얼마나 부정확한지 보여주는 지표이다.

앞선 RSE 가 absolute measure of lack of fit of the model 이라면, R^2 계수는 alternative measure of fit 을 의미한다. 분산의 비율의 형태로 나타나는데, 수식은 다음과 같다.

$$R^2 = \frac{\mathrm{TSS} - \mathrm{RSS}}{\mathrm{TSS}} = 1 - \frac{\mathrm{RSS}}{\mathrm{TSS}} \qquad (3.17)$$

RSS 실제값과 추정값 사이의 error, 즉 ‘잔차’ 의 제곱 합이고, TSS 는 실제값과 모평균 사이의 error, 즉 ‘오차’ 의 제곱 합을 의미한다.

R^2 계수는 X를 사용해 설명되는 Y 의 분산 정도를 비율로 나타낸다고 할 수 있다. (원문에서는 R^2 measures the proportion of variability in Y that can be explained using X. 라고 설명.)

ISLR : Chapter 3 다변량 선형회귀

단순 선형 회귀 모델을 각 독립변수 별로 생성하고 결과를 종합적으로 해석하는 방식은 만족스럽지 않다. 첫 번째 이유는, 각 독립변수에 의한 결과를 하나로 합치는 방식이 불분명하기 때문이다. 두 번째로, 각각의 단순 선형 회귀 모델들은 서로의 독립변수에 의한 상관관계가 결과에 미치는 영향을 무시하기 때문이다. 따라서 다음의 방식을 제안한다. (Suppose that we have p distinct predictors.)

$$Y = \beta_0 + \beta_1 X_1 + \beta_2 X_2 + \cdots + \beta_p X_p + \epsilon, \qquad (3.19)$$

당연히 각각의 계수들에 대한 적합한 추정값을 찾는 것이 목적이 된다.

이 때도 RSS 를 사용할 수 있다.

$$\hat{y} = \hat{\beta}_0 + \hat{\beta}_1 x_1 + \hat{\beta}_2 x_2 + \cdots + \hat{\beta}_p x_p. \qquad (3.21)$$

$$\begin{aligned} \text{RSS} &= \sum_{i=1}^{n} (y_i - \hat{y}_i)^2 \\ &= \sum_{i=1}^{n} (y_i - \hat{\beta}_0 - \hat{\beta}_1 x_{i1} - \hat{\beta}_2 x_{i2} - \cdots - \hat{\beta}_p x_{ip})^2. \qquad (3.22) \end{aligned}$$

▼ 로지스틱 회귀

Logistic Regression

위의 그래프에서 나타나듯이(오른쪽), 로지스틱 회귀에서 출력 Y는 특정 클래스에 데이터가 속할 확률을 보여준다. - “확률” 이란 표현을 유심히 봐야한다.

데이터 X 가 우리가 확인하고자 하는 클래스 (양성 클래스 = 1) 가 될 확률은 다음과 같이 나타낸다.

$$p(X) = \Pr(Y = 1|X)$$

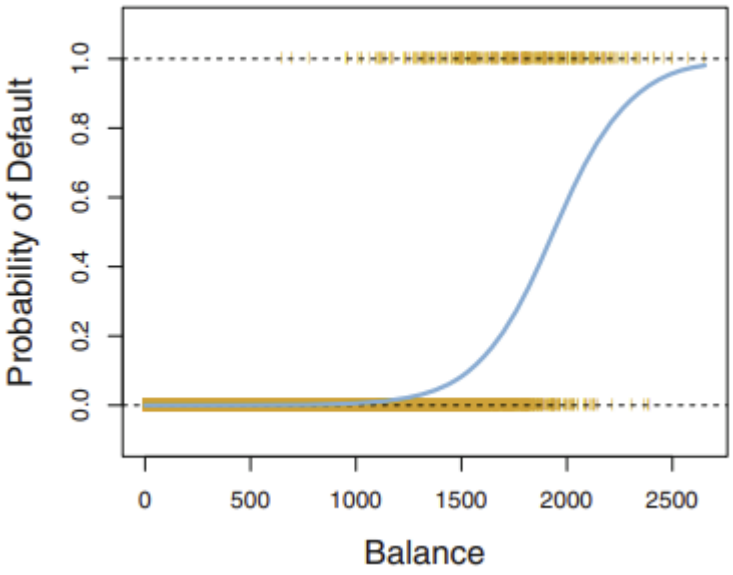
선형 회귀에서는 이 확률을 다음과 같이 표현한다.

$$p(X) = \beta_0 + \beta_1 X. \qquad (4.1)$$

하지만 이렇게 직선으로 나타내는 선형 회귀 모델은 위 그래프에서 보듯이 그 결과가 0-1 사이를 벗어나는 경우가 생긴다. 출력이 0-1 사이의 값으로 제한되는 함수 p(X) 가 필요하며 로지스틱 함수는 만족스럽다.

$$p(X) = \frac{e^{\beta_0+\beta_1 X}}{1 + e^{\beta_0+\beta_1 X}}. \qquad (4.2)$$

왜 만족스럽냐하면, 로지스틱 함수의 개형을 통해 확인할 수 있다.



위 그래프에서 Y 가 1.0에 가까울 수록 우리가 찾는 클래스 (=양성 클래스) 가 될 가능이 커지게 된다.

로지스틱 함수의 첫 번째 장점은 양성 클래스, 음성 클래스에 대한 가능성이 확연히 대비 된다는 점이다.

로지스틱 함수의 두 번째 장점은 언제나 S-shape 을 가지고 있어서 X 에 관계없이 항상 0-1 사이의 출력값을 가진다는 점이다.

결정트리

▼ 앙상블 학습

투표 기반 분류기

더 좋은 분류기를 만드는 매우 간단한 방법은 각 분류기의 예측을 모아서 가장 많이 선택된 클래스를 예측하는 것이다. 이렇게 다수결 투표로 정해지는 분류기를 hard voting 분류기 라고 한다. 놀랍게도 이 다수결 투표 분류기가 앙상블에 포함된 개별 분류기 증 가장 뛰어난 것보다도 정확도가 높을 경우가 많다. 이것이 가능한 이유는 큰 수의 법칙 때문이다.

다음은 사이킷런의 투표 기반 분류기를 만들고 훈련시키는 코드이다. 데이터는 moons 데이터셋을 사용했다.

```
# dataload
from sklearn.model_selection import train_test_split
from sklearn.datasets import make_moons

X, y = make_moons(n_samples=500, noise=0.30, random_state=42)
X_train, X_test, y_train, y_test = train_test_split(X, y, random_state=42)

from sklearn.ensemble import RandomForestClassifier
from sklearn.ensemble import VotingClassifier
from sklearn.linear_model import LogisticRegression
from sklearn.svm import SVC

log_clf = LogisticRegression()
rnd_clf = RandomForestClassifier()
svm_clf = SVC(probability=True)

# 투표 기반 분류기 객체 생성.
voting_clf = VotingClassifier(
    estimators=[('lr', log_clf),('rf', rnd_clf),('svc', svm_clf)],
    voting='soft')
voting_clf.fit(X_train, y_train)
```

```
# 각 분류기의 정확도를 확인해보자
from sklearn.metrics import accuracy_score
for clf in (log_clf, rnd_clf, svm_clf, voting_clf):
  clf.fit(X_train, y_train)
  y_pred = clf.predict(X_test)
  print(clf.__class__.__name__, accuracy_score(y_test, y_pred))

>>> LogisticRegression 0.864
    RandomForestClassifier 0.904
    SVC 0.896
    VotingClassifier 0.92
```

배깅과 페이스팅

다양한 분류기를 만드는 또 다른 방법은 같은 알고리즘을 사용하고 훈련 세트의 서브셋을 무작위로 구성하여 분류기를 각기 다르게 학습시키는 것이다.

훈련 세트에서 중복을 허용하여 샘플링하는 방식을 Bagging (Bootstrap aggregating) 이라 하며, 중복을 허용하지 않고 샘플링하는 방식을 pasting 이라고 한다.

배깅과 페이스팅 방식에서는 같은 훈련 샘플을 여러 개의 예측기에 걸쳐 사용할 수 있다. 하지만 배깅만이 한 예측기를 위해 같은 훈련 샘플을 여러 번 샘플링할 수 있다.

모든 예측기가 훈련을 마치면 앙상블은 모든 예측기의 예측을 모아서 새로운 샘플에 대한 예측을 만든다. 수집 함수는 전형적으로 분류일 때는 통계적 최빈값 (statistical mode) 이고 회귀에 대해서는 평균(mean) 을 계산한다. 개별 예측기는 원본 훈련 세트로 훈련시킨 것보다 훨씬 크게 편향되어 있지만 수집 함수를 통과하면 편향과 분산이 모두 감소한다. 일반적으로 앙상블의 결과는 원본 데이터셋으로 하나의 예측기를 훈련시킬 때와 비교해 편향은 비슷하지만 분산은 줄어든다.

사이킷런은 BaggingClassifier, BaggingRegressor 을 제공한다. 다음은 결정 트리 분류기 500개의 앙상블을 훈련시키는 코드이다. bootstrap=True 로 지정하면 배깅, False로 지정하면 페이스팅을 사용할 수 있다. n_jobs 는 CPU 코어 수를 지정하며 -1 설정 시 가용 가능한 모든 코어를 사용한다.

```
from sklearn.ensemble import BaggingClassifier
from sklearn.tree import DecisionTreeClassifier

bag_clf = BaggingClassifier(
    DecisionTreeClassifier(), n_estimators=500,
    max_samples=100, bootstrap=True, random_state=42)
bag_clf.fit(X_train, y_train)
```

부스팅(boosting)

부스팅은 약한 학습기(learner) 를 여러 개 연결하여 강한 학습기를 만드는 앙상블 방법을 말한다. 부스팅 방법의 아이디어는 앞의 모델을 보완해나가면서 일련의 예측기를 학습시키는 것이다.

랜덤 포레스트

▼ XGBoost

gradient boosting machine

- 그레디언트 부스팅은 깊이가 얕은 결정 트리를 사용하여 이전 트리의 오차를 보완하는 방식으로 앙상블 하는 방법
- 사이킷런의 GradientBoostingClassifier 는 기본적으로 깊이가 3인 결정트리를 100개 사용.
- 그레디언트 부스팅은 경사 하강법을 사용하여 트리를 앙상블에 추가한다.
- 분류에서는 로지스틱 손실함수를 사용한다.

  회귀 : 평균제곱오차함수(MSE)
- 결정 트리를 계속 추가하면서 극소점을 찾는것이 모델의 목적이다.
- 학습률 매개변수로 속도를 조절한다.

XGBoost

XGBoost 는 그레디언트 부스팅 알고리즘을 구현한 대표적인 라이브러리이다. extreme gradient boosting 의 약자이다. 이 패키지의 목표는 매우 빠른 속도, 확장성, 이식성이다.

LightGBM

- hyperparameter optimization
  - grid search : 주어진 하이퍼파라미터를 순회하며 가장 좋은 성능을 내는 값을 탐색.
  - random search : 하이퍼파라미터를 무작위로 탐색해 가장 좋은 성능을 내는 값을 찾는 기법
  - bayesian optimization : 사전 정보를 바탕으로 최적 하이퍼파라미터 값을 확률적으로 추정하며 탐색하는 기법.

Ch.10 다시 살펴보는 딥러닝 주요 개념

인공 신경망

- 퍼셉트론
- 신경망
- 활성화 함수
- 경사 하강법
  - 경사 하강법(gradient descent ; GD) 의 기본 아이디어는 비용 함수를 최소화하기 위해 반복해서 파라미터를 조정해가는 것이다. 구체적으로, 파라미터 θ 를 임의의 값으로 시작해서 한 번에 조금씩 비용 함수가 감소되는 방향으로 진행하여 비용 함

수값이 최솟값에 수렴할 때 까지 점진적으로 향상시킨다. (다른 의미로 오차가 최소로 만들어진다는 것.)

    - step 의 크기는 learning rate 하이퍼파라미터로 결정된다.
    - 경사하강법의 치명적인 문제점은 global minimum 보다 local minimum 에 수렴할 가능성이 크다는 것이다.

      ⇒ batch gradient descent (batch=n, n>1) or stochastic gradient descent (batch=1)

- 순전파와 역전파

CNN

- 합성곱 계층
- 패딩과 스트라이드
- 풀링
- 전결합
- 전체 구조

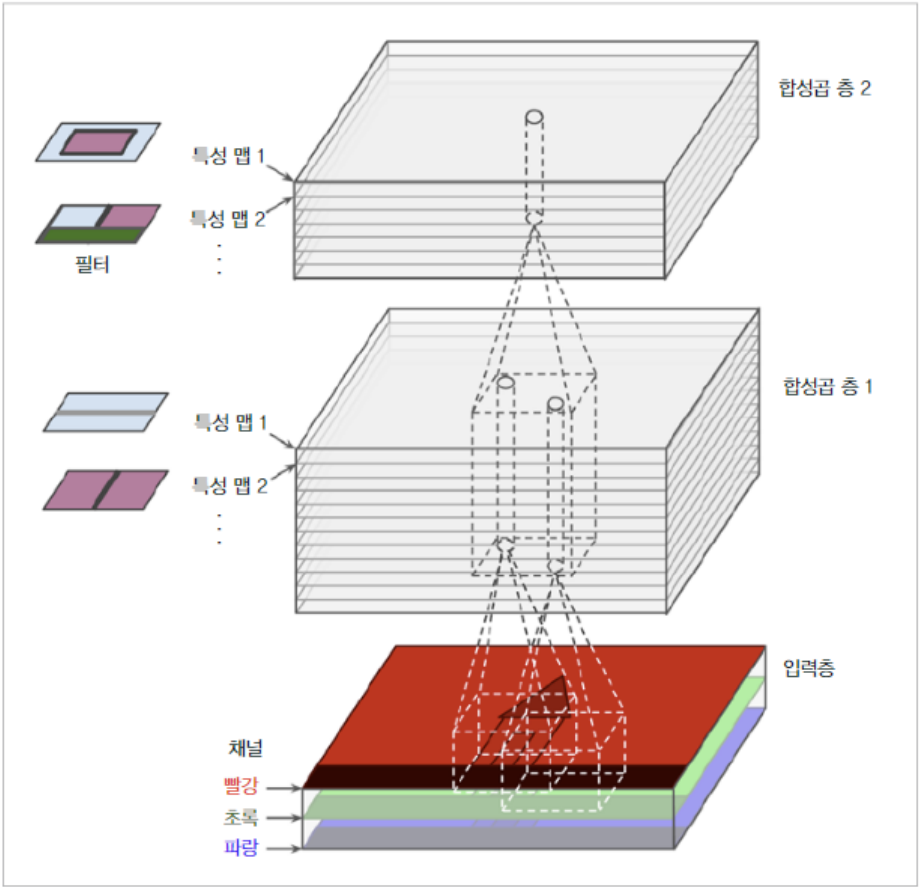


그림 14-6 여러 가지 특성 맵으로 이루어진 합성곱 층과 세 개의 컬러 채널을 가진 이미지

성능 향상을 위한 딥러닝 알고리즘

- 드롭아웃
  - 과대적합을 방지하기 위하여 신경망 훈련 과정에서 무작위로 일부 뉴런을 제외하는 기법. iteration 마다 서로 다른 신경망을 훈련하는 효과. 뉴런의 드롭아웃 비율은 하이퍼파라미터.
- 배치 정규화

    - 과대적합 방지와 훈련 속도 향상을 위한 기법.
    - 내부 공변량 변화 현상을 해결하기 위한 기법. 내부 공변량 변화는 신경망 계층마다 입력 데이터 분포가 다른 현상을 의미. 계층 간 데이터 분포의 편차를 줄이는 작업을 배치 정규화라고 한다.
    - 평균 = 0, 분산 = 1
- 옵티마이저
    - 신경망의 최적 가중치를 찾아주는 알고리즘 = optimizer
    - momentum : 관성 개념을 추가한 optimizer
    - Adagrad : 최적 파라미터에 도달할수록 학습률을 낮추도록 한 optimizer. adaptive learning rate
    - RMSProp : Adagrad 단점을 보완한 방법. Adagrad 는 훈련을 진행할수록 학습률이 작아져서 0에 수렴. RMSProp 은 최근 기울기만 고려해서 학습률을 낮춘다.
    - Adam : 모멘텀 + RMSProp
- 전이학습(transfer learning)
    - pretrained model → additonal training
    - 신경망 전체를 다시 훈련 ⇒ fine tuning, 전체 파라미터를 갱신
    - 일부 계층만 다시 훈련 ⇒ 파라미터 고정, 마지막 fully-connected layer 만 파라미터 갱신.

03.16 오프라인 모임 이후 추가 정리 부분

▼ ROC 곡선과 AUC(5.2.3)

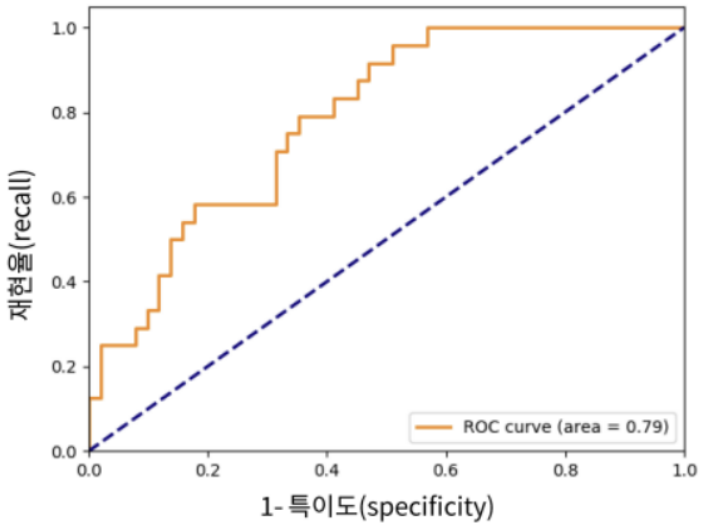
reference :

모형의 성능지표(MSE, MAPE, 정확도, 정밀도, 재현율, 특이도, F1 measure, ROC curve)

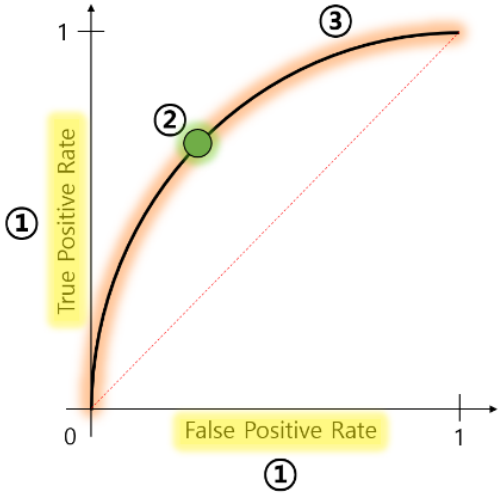
https://angeloyeo.github.io/2020/08/05/ROC.html

ROC curve, AUC

- 가로축을 1-특이도(specifivity) 세로축을 재현율(recall)로 하여 시각화한 그래프를 ROC (Receiver Operating Characteristics) Curve라고함.
- 이때 ROC curve의 면적을 AUC라고함.
- AUC가 1에 가까울 수록 좋은 지표다. (0 , 1)일때 가장 좋음.



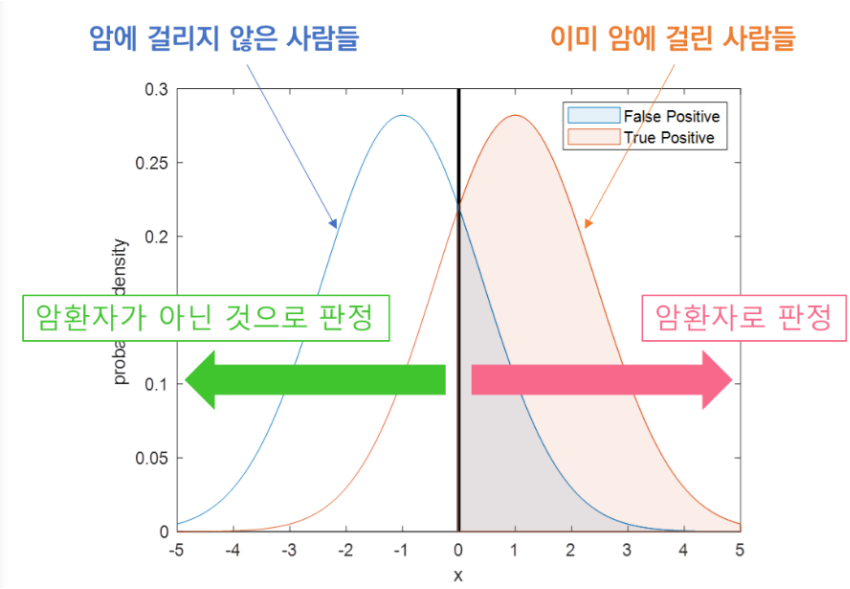
ROC(Receiver Operating Characteristic) curve 는 다양한 threshold 에 대한 이진분류기의 성능을 한번에 표시한 것이다. 이진 분류의 성능은 True Positive Rate 와 False Positive Rate 두 가지를 이용해서 표현하게 된다.



① True Positive Rate, False Positive Rate
② 현 위의 점의 의미는 무엇인가?
③ 현의 휨 정도가 의미하는 것은 무엇인가?

- True Positive Rate, False Positive Rate 에 관해

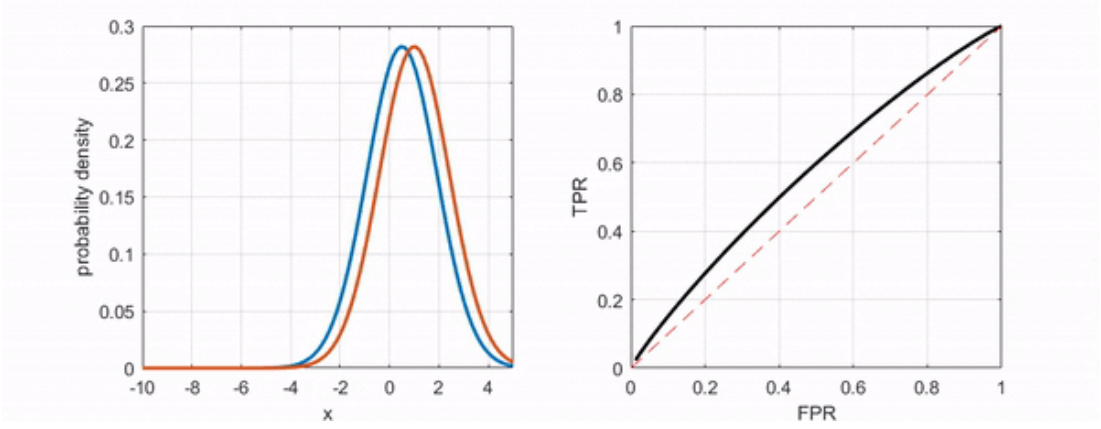
다음의 정규 분포와 상황을 고려해본다.



검은 색 bar 는 threshold 를 나타낸다. threshold 가 높을 수록 모든 내원자를 정상으로 판정하며 threshold 가 낮을 수록 모든 내원자를 다 환자로 판정한다.

- 현 위의 점의 의미는 무엇인가
  - 현 위에 점이 의미하는 것은 모든 가능한 threshold별 TPR 과 FPR 을 알아보겠다는 의미.

- 현의 휨 정도가 의미하는 것은 무엇인가
  - 이진 분류를 잘 해낼수록, 즉 두 그룹을 더 잘 구별할 수록 ROC 곡선은 좌상단에 붙게 된다.



▼ min-max 정규화(5.4.1)

- featue 값의 범위를 0-1 사이로 조정하는 기법
- 사이킷런 - MinMaxScaler

$$x_{scaled} = \frac{x - x_{\min}}{x_{\max} - x_{\min}}$$

▼ 랜덤포레스트 / XGBoost / LightGBM (5.6.7)

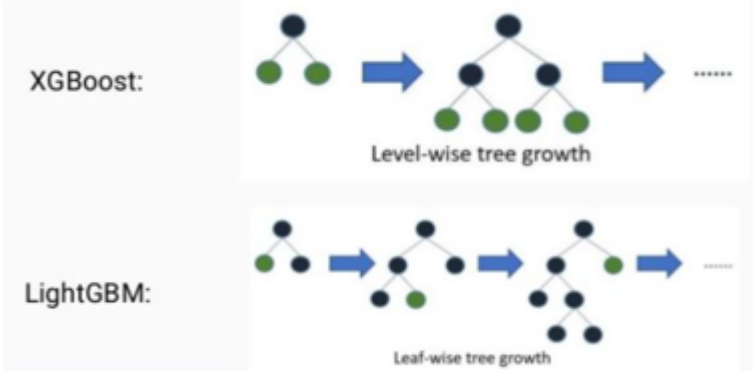
- **랜덤 포레스트**
  - 랜덤성 : 조금씩 다른 특성의 트리들 → 일반화 성능 향상
  - 계층적 접근 방식 탈피 시도, 노이즈 및 에러에 대해 강인한 모델
  - 부트스트랩(bootstrap) : 복원 추출을 통한 데이터 샘플링
  - 트리 구성 시 전체 속성 개수의 제곱근만큼 선택하고 그 중 정보 획득량이 가장 높은 것을 기준으로 데이터를 분할 한다.

- **부스팅 알고리즘**
  - 부스팅 알고리즘은 여러 개의 weak learner 를 순차적으로 학습-예측하는 방식
  - 잘못 예측한 데이터에 가중치를 부여함으로써 오류를 개선
  - 대표적으로 GBM

- **XGBoost**
  - Extreme Gradient Boosting
  - Gradient Boost 를 병렬 학습이 지원되도록 구현한 라이브러리
  - overfitting 에 대비한 규제 기능으로 customizing에 용이
  - weighted quantile sketch

        - 트리 학습의 핵심 문제는 최상의 분할 지점을 찾는 것이며, XGBoost 는 feature 분포의 백분위수에 따라 후보 분할지점을 예상한다.
    - Sparsity-aware Split Finding
        - 희소성 확인, 분할 지점 찾기
        - 결측값이 존재하거나 다수가 0인 데이터 = 희소한 데이터
        - 존재하지 않는 값을 결측치로 처리하고 결측치를 다루는 가장 좋은 방향으로 학습.

- **LightGBM**
    - LEAF 중심 트리 분할, 트리의 균형 고려하지 않고 최대 손실값을 가지는 leaf node 를 지속적으로 분할하면서 비대칭적인 트리 생성.



출처 : https://www.slideshare.net/GabrielCyprianoSaca/xgboost-lightgbm

    - XGBoost 대비, 빠른 학습 및 예측 수행 시간