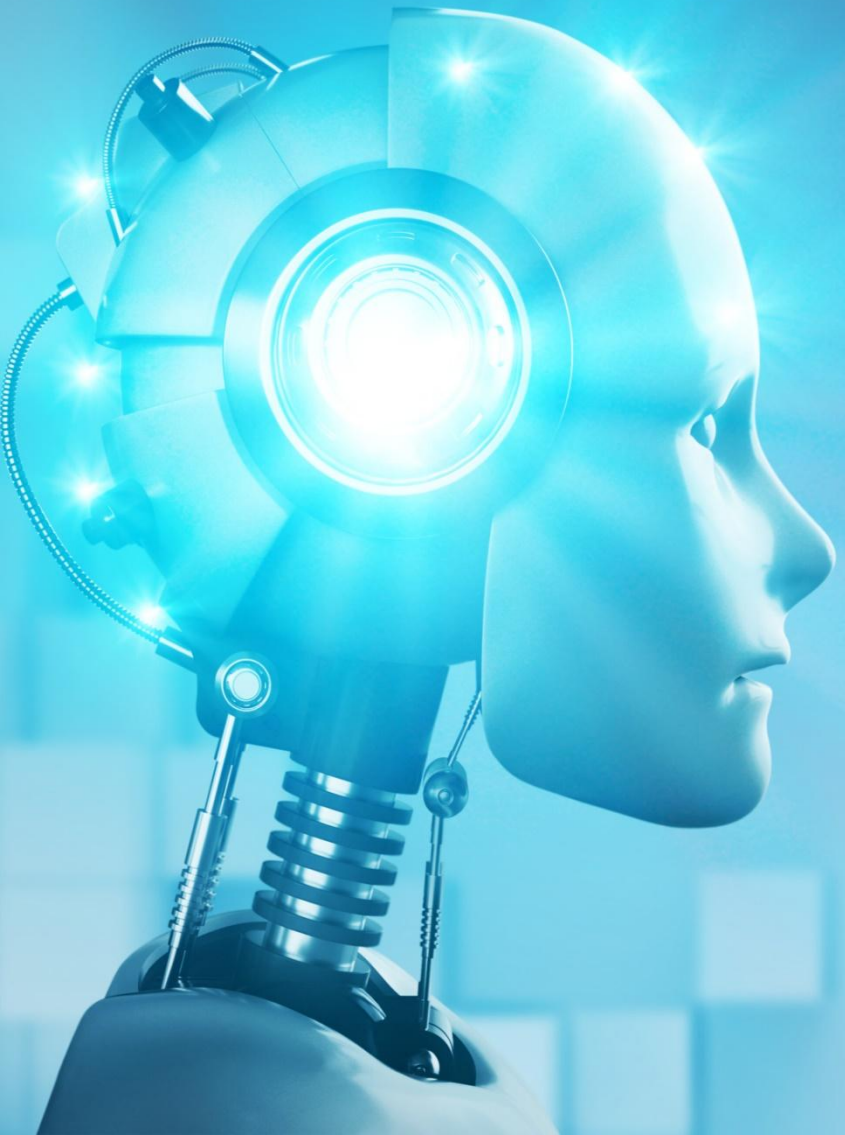
인공지능(AI) 개발자 양성 과정

3차시

머신러닝 기본

휴먼교육센터 안홍근 강사

목차 Contents



1 머신러닝 기초

1. 지도, 비지도, 강화학습 비교
2. 회귀와 분류
3. 성능지표 분석

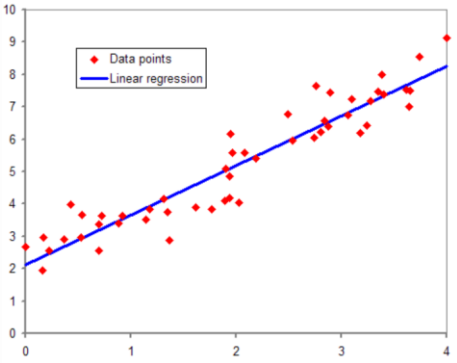
1. 지도, 비지도, 강화학습 비교(1/2)

엘젠은 인공지능 및 빅데이터 기반의 스스로 성장하는 스마트 지능형 플랫폼을 완성하여 유통/쇼핑/콜센터/공공 시장에 획을 긋고 있습니다.

지도 학습

회귀, 분류

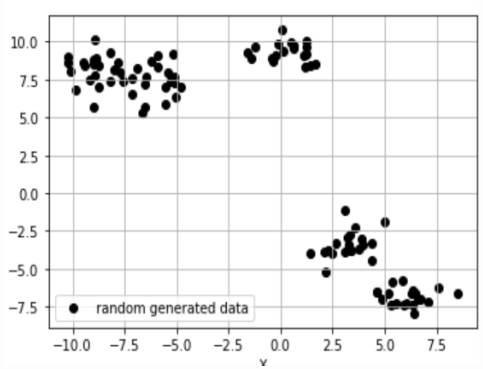
- 컴퓨터는 교사에 의해 주어진 예제와 정답(레이블)을 제공받음
- 지도 학습의 목표는 입력을 출력에 매핑하는 일반적인 규칙(함수, 패턴)을 학습하는 것임
- ex. 교사가 어떤 사진이 고양이인지, 강아지인지 알려줌



비지도 학습

클러스터링(군집화)

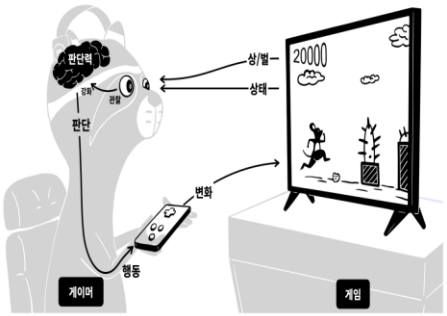
- 외부에서 정답(레이블)이 주어지지 않고 학습 알고리즘이 스스로 입력 데이터에서 패턴을 발견하는 학습
- ex. 과일의 모양, 색상, 크기 등 다양한 특징을 이용하여 유사한 과일들을 분류



강화 학습

보상 및 처벌

- 보상 및 처벌의 형태로 학습이 이루어짐
- 상대방과의 경기와 같은 동적인 환경에서 프로그램의 행동에 대한 피드백만 제공됨
- ex. 바둑에서 어떤 수를 두어 승리하였다면 보상을 줌



1. 지도, 비지도, 강화학습 비교(2/2)

https://youtu.be/Udkc6rkH_CQ

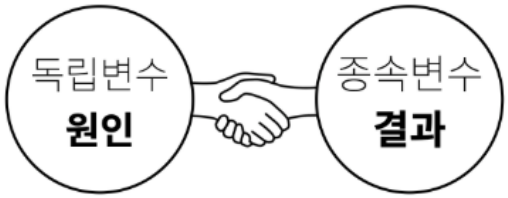
2. 회귀와 분류

2.1 회귀(Regression)

회귀 정의

- 연속적인 수치 데이터를 예측하는 머신러닝 기법으로, 입력변수(독립변수)와 출력변수(종속변수) 간의 관계를 모델링함
- 과거 데이터를 바탕으로 미래의 값을 예측하거나 연속적인 숫자형 데이터를 추정하는 데 사용

독립변수, 종속변수



- ✓ 독립변수(Feature): 모델에 입력으로 제공되는 데이터 (ex. 광고비, 기온)
- ✓ 종속변수(Target): 모델이 예측하려는 값 (ex. 매출, 전력 사용량)

회귀의 종류

선형 회귀

- 연속적인 종속변수를 예측
- 독립변수와 종속 변수 사이의 선형관계 모델링
- 예시 : 주택가격 예측, 키 예측, 판매량 예측

다항 회귀

- 선형회귀의 확장된 형태
- 독립변수와 종속 변수의 비선형 관계 모델링
- 곡선 형태의 관계를 표현할 수 있음

로지스틱 회귀

- 종속 변수가 범주형(예/아니오, 성공/실패)일 때 적용
- 결과를 0과 1 사이의 확률로 예측
- 예시 : 스팸 메일 분류, 질병진단, 고객이탈 예측

2.1 회귀(Regression)

선형 회귀 정의

- 독립변수(X)와 종속변수(Y) 사이의 선형적 관계를 모델링하는 통계적 기법
- 최적의 **직선, 곡선 등 데이터의 패턴을 찾아서 예측**

단순 선형회귀

- 하나의 독립변수만 사용

[예시]
- 주택 가격 예측 : 집의 크기(X)를 이용해 주택가격 예측
- 키와 몸무게 관계 : 키(X)를 이용해 몸무게(Y) 예측
- 강수량과 우산판매: 강수량(X)을 이용해 우산판매량(Y) 예측

기본 방적식

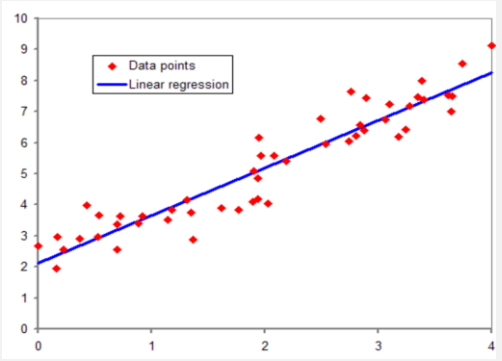
$$Y = aX + b$$

X: 독립변수, Y : 종속변수

a: 기울기, b:절편

단순선형 회귀의 그래프 (예시)

- 데이터 포인트들이 산점도로 표시됨
- 최적의 직선이 데이터 포인트들을 관통하도록 그려짐



2.1 회귀(Regression)

다중 선형회귀

- 여러 개의 독립변수 사용
- 여러 독립변수 간의 상호작용을 고려할 수 있어, 더 넓은 범위에서 적용 가능
- 단순 선형회귀보다 더 높은 예측력

기본 방정식

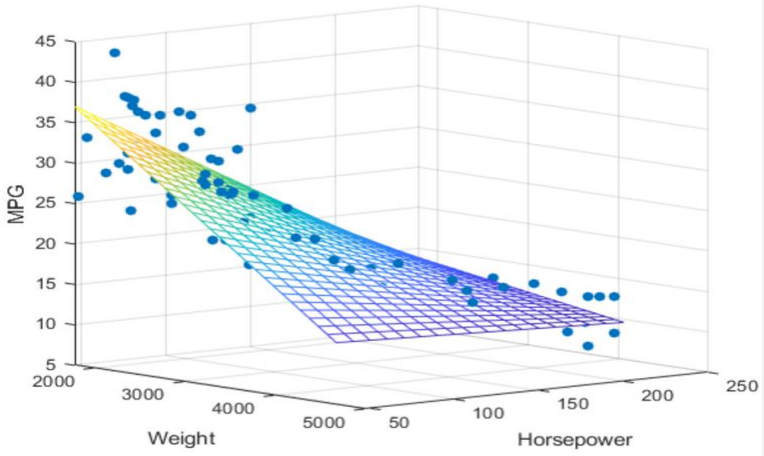
$$Y = \beta_0 + \beta_1 X_1 + \beta_2 X_2 + \ldots + \beta_{p-1} X_{p-1} + \epsilon$$

$X_1, X_2 \ldots X_{p-1}$: 독립변수, Y : 종속변수

β0 : y절편, β1, β2, ..., βp-1 : 회귀계수

다중선형회귀 그래프 (예시)

- 2차원 이상의 공간에서 표현됨
- 독립변수가 2개인 경우에는 3차원 그래프로, 3개 이상인 경우 다차원 공간에 표현됨



2.2 분류(Classification)

분류 정의

- 지도학습의 일종으로 기존에 존재하는 데이터의 카테고리 관계를 파악하여, 새롭게 관측된 데이터의 카테고리를 판별
- 일련의 데이터가 포함된 기존 카테고리들을 학습하고, 이것을 기반으로 **데이터 범주를 구분하여 경계를 나누는 것을 학습**

분류 주요 특징

- 지도학습 방식 : 이미 레이블링된 훈련 데이터를 사용하여 모델을 학습
- 출력 유형 : 이산적인 클래스나 범주를 예측
- 응용 분야 : 스팸메일 분류, 질병 진단, 이미지 인식 등에 사용

cf. 분류 용어참고 → 클래스 : 데이터가 속할 수 있는 범주(ex. 구매/미구매)

분류 종류 및 알고리즘

- 이진 분류 : 두 범주 중 하나의 데이터를 분류
- 다중 클래스 분류 : 두개 이상의 범주 중에서 하나의 데이터를 분류

- 분류 알고리즘 종류 : 로직스틱 회귀, KNN, 의사결정 나무, 랜덤 포레스트, SVM, 나이브베이즈

2.2 분류(Classification)

로지스틱 회귀

- 선형 분류 알고리즘으로, 이진 분류 문제에 가장 많이 사용됨
- 시그모이드(로지스틱) 함수를 사용하여 0과 1사이의 확률을 예측

로지스틱 회귀 특징

- 간단하고 해석이 용이
- 선형 결정 경계를 생성
- 이진 분류에 최적화되어 있음
- 0.5보다 작은 확률은 0을 예측하고 0보다 큰 확률은 1을 예측

예측값	정답
0.6	1
0.51	1
0.5	0

예시 코드

```
from sklearn.linear_model import LogisticRegression
from sklearn.model_selection import train_test_split

# 로지스틱 회귀 모델 생성
model = LogisticRegression()
model.fit(X_train, y_train)

# 예측
predictions = model.predict(X_test)
```

2.2 분류(Classification)

KNN(K-Nearest Neighbors)

- 지도학습의 한 방법으로 가장 가까운 K개의 이웃을 기반으로 새로운 데이터 포인트의 값을 예측
- 주변 K개 가장 가까운 이웃들의 평균값으로 예측

KNN 특징

- K가 작으면 과적합 위험 높음
- K가 크면 모델의 일반화 성능 저하 가능성 존재
- 특성들의 스케일 차이로 인한 왜곡 방지 위해 정규화/표준화 필요

예시 코드

```
from sklearn.neighbors import KNeighborsRegressor
from sklearn.preprocessing import StandardScaler

# 데이터 스케일링
scaler = StandardScaler()
X_scaled = scaler.fit_transform(X)

knn_regressor = KNeighborsRegressor(n_neighbors=5)
knn_regressor.fit(X_scaled, y)

# 예측
predictions = model.predict(X_test)
```

2.2 분류(Classification)

의사결정나무(Decision Tree)

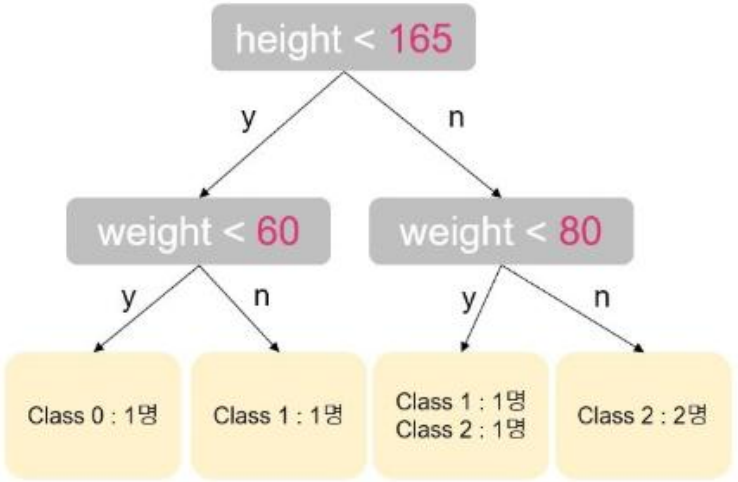
☑ 나무 가지를 뻗어나가듯, 데이터를 여러 개의 하위 집합으로 분할하면서 최종적으로 어떤 클래스에 속하는지 판단하는 구조

의사결정나무 특징

- 잘문을 던져서 대상을 좁혀나가는 '스무고개' 놀이와 비슷한 개념
- 의사결정나무는 불순도, 즉 지니계수가 낮아지는 방향으로 계속 분기함

features

Height	Weight	Class
150	50	0
160	60	1
165	65	1
170	70	2
180	80	2
185	85	2



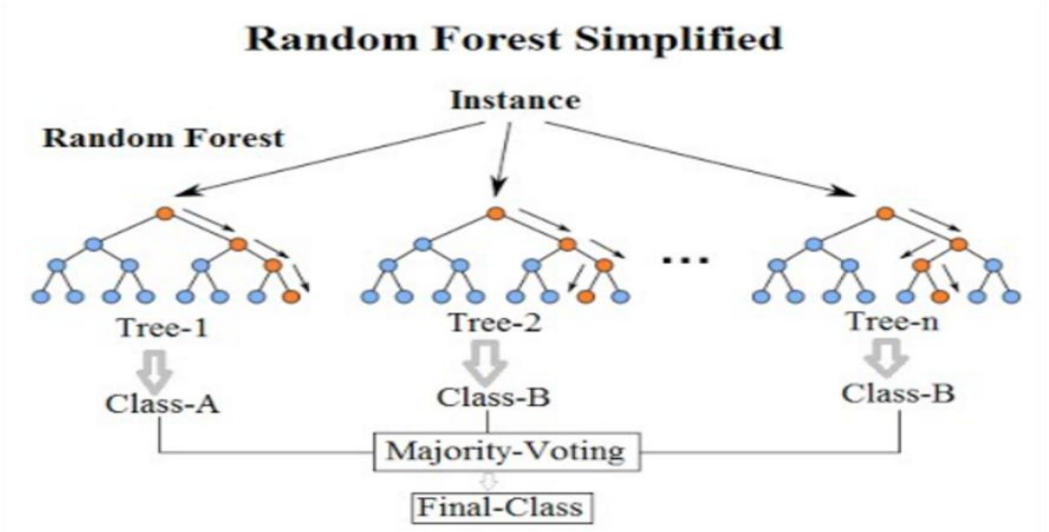
2.2 분류(Classification)

랜덤 포레스트(Random Forest)

- 같은 데이터에 **의사결정나무 여러 개를 동시에 적용**해서 학습성능을 높이는 **앙상블 학습** 기법
- 여기서 앙상블 학습이란 **여러 개의 기본 학습 모델을 조합**하여 더 강력한 모델을 만드는 기법

랜덤 포레스트 특징

- 원본 데이터셋으로부터 **랜덤하게 샘플을 여러번 추출**하여, 각 샘플로 **모델을 개별적으로 학습**시킨 후, **결과를 취합**
- 학습시킬 때, 모든 특성을 사용하는 것이 아니라 **일부 특성만 무작위로 선택하여 학습**시킴, 이를 통해 **나무사이의 상관관계를 줄이고, 다양성을 증가**시킴



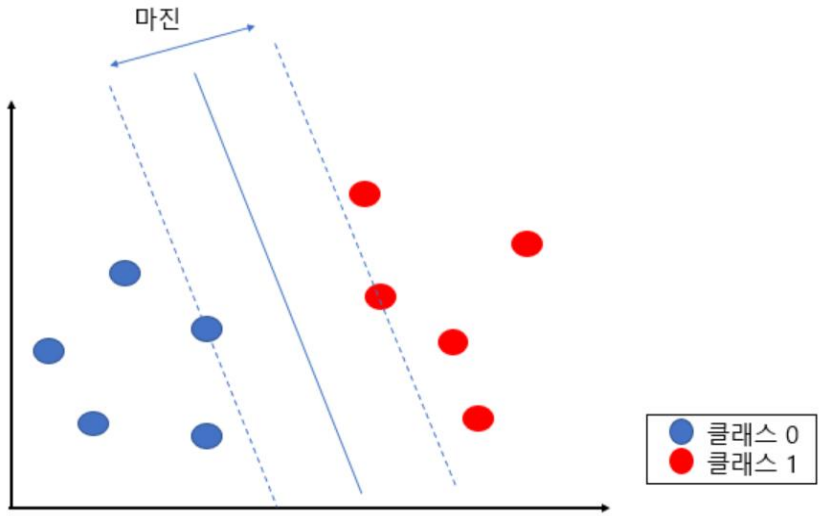
2.2 분류(Classification)

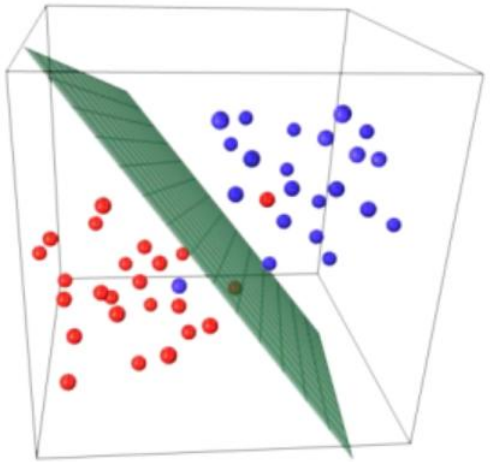
SVM (Support Vector Machine)

- 주로 분류 문제에서 뛰어난 성능을 보이며 결정경계(Decision Boundary)를 통해 데이터를 구분하는 것이 핵심
- 데이터 공간에서 데이터를 두 클래스로 나누는 경계선인 초평면을 찾아내어 데이터를 분리하는 기법 사용

SVM 특징

- 초평면 : 데이터 공간에서 데이터를 나누는 경계선
- 서포트 벡터 : 마진에서 가까이 있는 각 클래스의 데이터로, 모델의 결정 경계를 결정하는 핵심 요소
- SVM은 학습 시 초평면과 서포트 벡터 사이의 마진을 최대화하는 방향으로 최적화 수행





2.2 분류(Classification)

나이브 베이즈(Naive Bayes)

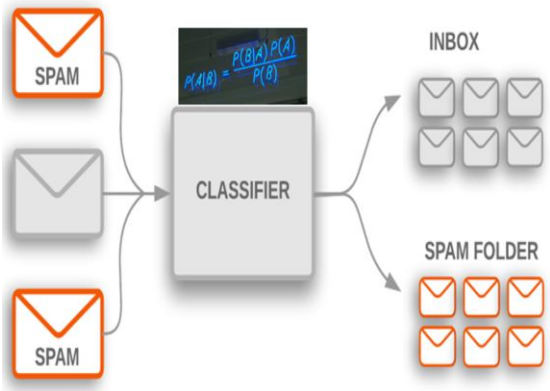
- 확률론적 분류 모델로, 각 단어들이 독립적(분류에 종속되지 않음)이라고 가정하고 각 단어의 빈도수만을 고려하여 판단
- 이 모델은 매우 간단하지만, 텍스트 분류나 스팸 필터링 같은 다양한 분야에서 강력하게 활용

나이브 베이즈 특징

- 각 특성이 서로 독립적이라고 가정함. 즉 하나의 특성이 다른 특성에 영향을 미치지 않는다고 간주
- 매우 간단한 계산으로 확률을 추정함
- 학습 과정에서 간단한 확률 계산만 필요하고, 예측 시에도 빠르게 처리할 수 있음.
- 대규모 데이터셋에서 효율적으로 동작

스팸 이메일 : 무료 할인 쿠폰

정상 이메일 : 오늘의 뉴스 업데이트



"무료", "할인", "쿠폰" 같은 단어들이 스팸 이메일에서 나오는 확률을 계산

나이브 베이즈는 이렇게 각 단어가 스팸과 정상 이메일에서 등장할 확률을 바탕으로, **이메일이 스팸인지 아닌지 확률적으로 판단**

2.4 회귀와 분류 비교

항목	회귀	분류
목적	연속적인 값 예측	이산적 클래스(범주) 예측
출력값	실수형 (ex. 0.0 ~ 1.0)	범주형 (ex. 스팸/정상)
예시	주택가격 예측, 날씨 예측	이메일 스팸 필터링, 암진단
알고리즘	선형/다항 회귀, 랜덤 포레스트, 신경망 회귀	로지스틱 회귀, SVM, 랜덤 포레스트, CNN
평가지표	MAE, MSE, RMSE, R^2	Accuracy(정확도), Precision(정밀도), 재현율 (Recall), F1-score

2.4 기타 참고자료 : 랜덤 포레스트 vs XGBoost

항목	랜덤 포레스트	XGBoost/ LightGBM
트리 조합방식	배깅(Bagging) : 여러 개의 결정 트리를 독립적으로 학습, 그 결과를 평균하고 분류하여 최종 결과를 얻는 방식	부스팅(Boosting): 앞선 트리의 오류를 보완하도록 다음 트리를 순차적으로 학습시키는 방식
트리 학습 방식	독립적 학습	순차적 학습(앞선 트리의 오류 보완)
과적합 방지	부트스트래핑, 특징 샘플링	레귤라이제이션(과적합 방지를 위해 L1, L2 정규화 사용) ex. L1 : 많은 가중치를 0으로 만듬, L2: 모든 가중치를 작게 만듬
장점	간단하고 빠른 학습, 높은 안정성	높은 정확도, 과적합 방지, 병렬 처리
단점	XGBoost만큼 높은 정확도는 보장하지 못함	파라미터 튜닝이 복잡할 수 있음

3. 성능지표 분석

3.1 성능지표 분석 정의 및 특징

- 머신러닝/딥러닝 모델의 **예측 결과가 얼마나 정확한지, 효과적인지 평가**하기 위해 사용되는 지표들을 분석하는 과정
- 이를 통해 **모델의 강점과 약점을 파악**하고, **모델 개선을 위한 방향성을 설정**할 수 있음

특징

특징	설명
객관성	모델의 성능을 수치로 표현하여 객관적으로 비교 가능
목표 기반	각 성능지표는 특정 문제(분류, 회귀 등)의 평가 기준에 맞춰 선택
다양성	문제 유형과 데이터의 특성에 따라 여러 성능지표를 조합하여 사용

종류

모델링 목적	목표 변수 유형	관련 모델	평가 방법
예측 / 회귀	**연속형**	**선형회귀**	Loss, MSE, RMSE, MAE, R^2(결정계수)
분류	**범주형**	**로지스틱 회귀, 의사결정나무, SVM**	정확도(Accuracy), 정밀도(Precision), 재현율(Recall), F1 Score

3.2 주요 성능 지표 ▶ 3.2.1 Accuracy (정확도)

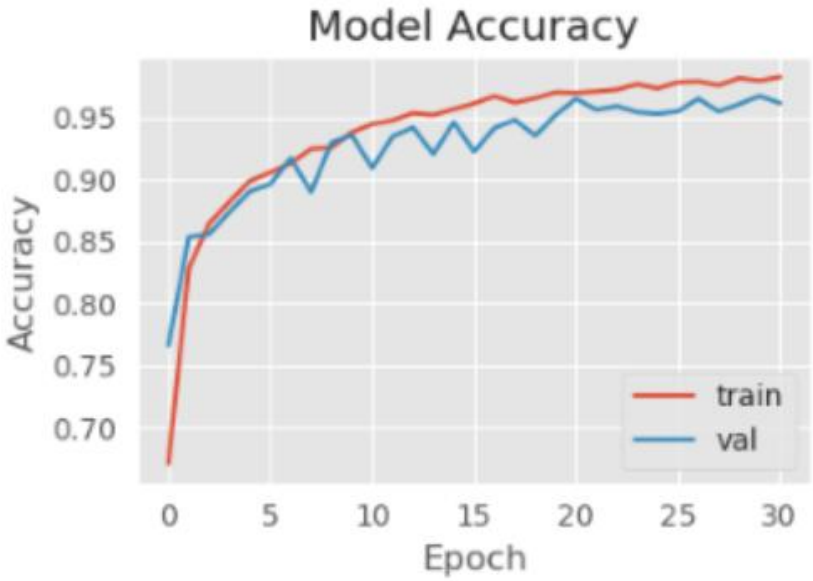
정의

- 전체 데이터 중 모델이 올바르게 예측한 데이터의 비율

$$\text{Accuracy} = \frac{\text{정확히 예측한 샘플 수}}{\text{전체 샘플 수}}$$

특징

- 분류 문제에서 자주 사용
- 클래스(분류하고 싶은 대상)가 균형 잡힌 데이터에서 유용하지만, 불균형 데이터에서는 오해를 불러일으킬 수 있음.
  예를 들어, 데이터의 95%가 클래스 A이고 모델이 항상 A를 예측해도 Accuracy는 95%
- 예시 : 100개의 샘플 중 90개를 정확히 예측했다면, Accuracy는 90%



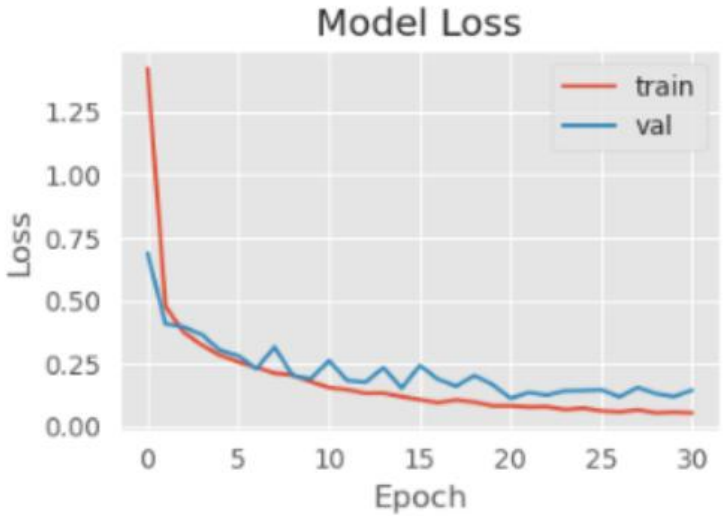
3.2 주요 성능 지표 ▶ 3.2.2 Loss (손실함수)

정의

- 모델의 예측값과 실제값 간의 오차를 나타내는 값

특징

- 학습 과정에서 모델이 얼마나 잘 맞춰가는지를 보여줌
- Loss 값이 작을수록 모델 성능이 향상된 것으로 간주
- 예시 : 회귀 문제에서 예측값이 실제값과 멀리 떨어져 있으면 Loss 값이 커짐



3.2 주요 성능 지표 ▶ 3.2.3 R^2 (결정계수)

정의

- 결정계수는 통계적 모델의 적합도를 평가하는 지표
- R^2 값은 0에서 1사이의 값을 가지며, 값이 1에 가까울수록 모델이 데이터를 잘 설명한다는 뜻

특징

- 0과 1사이의 값을 가짐
  - 결정계수는 항상 0과 1사이의 값을 가짐
  - 0에 가까울수록 모델의 설명력이 낮고, 1에 가까울수록 모델이 설명력이 높음
- 모델의 적합도를 평가
  - 결정계수가 높을수록 모델이 데이터에 잘 맞는다고 할 수 있음
  - 하지만, 결정계수가 높다고 항상 좋은 모델은 아니고, 과적합(overfitting)문제가 발생할 수 있음

예시

- 총 변동 중 모델이 설명하는 변동이 비율
  - 키와 몸무게의 관계를 모델로 만들었다고 가정
  - 결정계수가 0.8이라면, 몸무게의 변동 중 80%가 키에 의해 설명된다는 의미, 나머지 20%는 다른 요인(예: 성별, 나이)에 의해 설명될 수 있음

3.2 주요 성능 지표 ▶ 3.2.4 MSE (Mean Squared Error, 평균제곱오차)

정의

- 예측값과 실제값의 차이를 제곱하여 평균한 값

$$MSE = \frac{1}{n}\sum_{i=1}^{n}(y_i - \hat{y}_i)^2$$

특징

- 예측값과 실제값 간의 **오차를 제곱하기 때문에 큰 오차에 더 민감**
- 손실 함수로 자주 사용
- 단점 : 단위가 원래 데이터의 단위보다 커짐

예시

- 실제값: [3, −0.5, 2, 7] , 예측값: [2.5, 0, 2, 8]
- $(3-2.5)^2+(-0.5-0)^2+(2-2)^2+(7-8)^2=0.25+0.25+0+1=1.5$
- MSE = 1.5/4=0.375

3.2 주요 성능 지표 ▶ 3.2.5 RMSE (Root Mean Squared Error, 평균제곱근오차)

정의

- MSE의 제곱근을 취한 값

$$RMSE = \sqrt{\frac{1}{n}\sum_{i=1}^{n}(y_i - \hat{y}_i)^2}$$

특징

- 오차를 원래 데이터의 단위로 환산하여 해석 가능
- MSE보다 해석이 직관적임 (제곱근을 취함으로써 단위를 원래 데이터의 단위와 동일하게 만듬)
- 큰 오차에 민감 (MSE보다는 제곱근을 취함으로써 다소 완화)

예시

- 실제값: [3, −0.5, 2, 7] , 예측값: [2.5, 0, 2, 8]
- $(3-2.5)^2+(-0.5-0)^2+(2-2)^2+(7-8)^2=0.25+0.25+0+1=1.5$
- MSE = 1.5/4=0.375 → RMSE $\sqrt{0.375}$ = 0.612

3.2 주요 성능 지표 ▶ 3.2.6 MAE (Mean Absolute Error, 평균절대오차)

정의

- 예측값과 실제값 간 오차의 절대값을 평균한 값

$$MAE = \frac{1}{n}\sum_{i=1}^{n}|y_i - \hat{y}_i|$$

특징

- 오차의 절대 크기를 평가
- 이상치(outlier)에 덜 민감
- 단위가 원래 데이터와 동일

예시

- 실제값: [3, −0.5, 2, 7] , 예측값: [2.5, 0, 2, 8]
- |3−2.5|+|−0.5−0|+|2−2|+|7−8| = 0.5+0.5+0+1 = 2
- MAE = 2/4 = 0.5

3.3 성능 지표 비교

지표	특징	단점	사용 예시
Accuracy	전체 예측의 정확도 평가	불균형 데이터에서 오해 발생	이진 분류, 다중 분류 문제
Loss	모델 학습 상태를 나타냄	결과 해석 어려움	모든 학습 과정
MSE	큰 오차에 민감, 손실 함수로 적합	단위가 원래 데이터보다 큼	회귀 문제 평가
RMSE	단위를 복원하여 직관적으로 해석 가능	큰 오차에 민감	회귀 문제 평가
MAE	이상치에 덜 민감	오차의 크기만을 고려하고, 오차가 양수인지, 음수인지에 대한 정보는 포함하지 않음 (오차방향 정보가 없음)	회귀 문제 평가

3.4 성능지표 코드 예시 – MSE

- zip(actual_values, predicted_values)는 실제값과 예측값을 짝지음
- $(a - p)^2$는 실제값과 예측값의 차이를 제곱
- sum(squared_errors) / len(squared_errors) 는 제곱 오차의 평균(MSE)을 계산

```
# 실제값과 예측값 정의
actual_values = [3, -0.5, 2, 7]  # 실제값 리스트
predicted_values = [2.5, 0, 2, 8]  # 예측값 리스트

# MSE 계산
squared_errors = [(a - p) ** 2 for a, p in zip(actual_values, predicted_values)]  # 각 오차를 제곱
mse = sum(squared_errors) / len(squared_errors)  # 제곱 오차의 평균

# 결과 출력
print("Mean Squared Error (MSE):", mse)
```

3.4 성능지표 코드 예시 – RMSE

- MSE를 계산하는 과정은 동일
- mse ** 0.5 는 MSE의 제곱근을 계산하여 RMSE를 얻음
- RMSE는 MSE와 동일하지만 원래 데이터의 단위로 환산

```
# 실제값과 예측값 정의
actual_values = [3, -0.5, 2, 7]  # 실제값 리스트
predicted_values = [2.5, 0, 2, 8]  # 예측값 리스트

# RMSE 계산
squared_errors = [(a - p) ** 2 for a, p in zip(actual_values, predicted_values)]  # 각 오차를 제곱
mse = sum(squared_errors) / len(squared_errors)  # MSE 계산
rmse = mse ** 0.5  # MSE의 제곱근을 계산하여 RMSE를 얻음

# 결과 출력
print("Root Mean Squared Error (RMSE):", rmse)
```

3.4 성능지표 코드 예시 – MAE

- abs(a – p)는 오차의 절대값을 계산
- sum(absolute_erros) / len(absolute_errors)는 절대 오차의 평균 (MAE)을 계산

```
# 실제값과 예측값 정의
actual_values = [3, -0.5, 2, 7]  # 실제값 리스트
predicted_values = [2.5, 0, 2, 8]  # 예측값 리스트

# MAE 계산
absolute_errors = [abs(a - p) for a, p in zip(actual_values, predicted_values)]  # 각 오차의 절대값
mae = sum(absolute_errors) / len(absolute_errors)  # 절대 오차의 평균

# 결과 출력
print("Mean Absolute Error (MAE):", mae)
```

3.4 성능지표 코드 예시 – 딥러닝 활용 예시 (향후 딥러닝 과정에서 학습예정)

- 모델 정의 : Sequential 모델을 사용하여 신경망 구성
- 손실 함수 및 최적화 알고리즘 설정 : compile 메서드를 통해 MSE 손실함수와 Adam 최적화 알고리즘 설정
- 학습 : fit 메서드를 사용하여 모델을 학습
- 평가 : evaluate 메서드를 사용하여 테스트 데이터에 대한 손실과 정확도를 계산

```
import tensorflow as tf
from tensorflow.keras.models import Sequential
from tensorflow.keras.layers import Dense
from tensorflow.keras.losses import MeanSquaredError
from tensorflow.keras.metrics import CategoricalAccuracy

# 모델 정의
model = Sequential([
    Dense(units=64, activation='relu', input_dim=100),
    Dense(units=10, activation='softmax ' )  ])

# 손실 함수와 최적화 알고리즘 설정
model.compile(loss=MeanSquaredError(), optimizer='adam', metrics=['accuracy'])

# 학습 데이터 생성 (예시)x_train = tf.random.normal((1000, 100))
y_train = tf.random.uniform((1000, 10), minval=0, maxval=1, dtype=tf.int32)

# 모델 학습
model.fit(x_train, y_train, epochs=10)

# 평가 데이터 생성 (예시)
x_test = tf.random.normal((100, 100))
y_test = tf.random.uniform((100, 10), minval=0, maxval=1, dtype=tf.int32)

# 모델 평가
test_loss, test_acc = model.evaluate(x_test, y_test)
print('Test loss:', test_loss)
print('Test accuracy:', test_acc)
```