

# 회귀 분석 입문: k-최근접 이웃(kNN)과 모델 평가

강의 자료 통합 노트

October 26, 2025

- 강의명: CS109A: 데이터 과학 입문
- 주차: Lecture 04
- 교수명: Pavlos Protopapas, Kevin Rader, Chris Gumb
- 목적: Lecture 04의 핵심 개념 학습

## Contents

1	개요	2
2	핵심 용어 정리	2
3	통계 모델링 시작하기	4
3.1	예측이란 무엇인가?	4
3.2	데이터의 두 가지 역할: 예측 변수(X)와 반응 변수(y)	4
3.3	데이터를 '행렬'로 표현하기 (The Design Matrix)	4
4	모델이란 무엇인가?	6
4.1	완벽한 모델 vs. 통계 모델	6
4.2	모델의 두 가지 목적: 추론(Inference) vs. 예측(Prediction)	6
5	k-최근접 이웃 (kNN) 알고리즘	7
5.1	가장 단순한 모델: "평균" 모델	7
5.2	kNN의 핵심 아이디어: "가까운 이웃" 참고하기	7
5.3	k값에 따른 모델의 변화	7
5.4	k는 '하이퍼파라미터'입니다	8
6	모델 평가하기 (Error Evaluation)	9
6.1	"최고의" 모델이란 무엇인가?	9
6.2	데이터를 나누는 이유: 훈련, 검증, 테스트	9
6.3	손실 함수 (Loss Function): 오류를 숫자로 나타내기	10
7	최고의 모델 선택 및 최종 평가	11
7.1	검증 세트를 이용한 k값 선택	11

7.2	모델이 "쓸 만한지" 평가하기: $R^2$ (R-squared)	11
8	kNN의 한계: 차원의 저주	13
8.1	kNN을 다차원으로 확장하기	13
8.2	"차원의 저주"란 무엇인가?	13
9	학습 목표 요약 (Checklist)	15

## 1 개요

이 문서는 통계적 모델링의 기초, 특히 **k-최근접 이웃(kNN) 회귀** 모델에 대해 다룹니다.

우리의 목표는 'TV 광고 예산'과 같은 데이터를 사용하여 '제품 판매량'을 예측하는 것입니다.

kNN은 "가장 유사한 과거 사례(이웃)를 찾아 그들의 평균값으로 예측한다"는 매우 직관적인 아이디어에서 출발합니다.

모델을 만든 후에는 이 모델이 얼마나 '좋은지' 평가해야 합니다. 이를 위해 데이터를 **훈련/검증/테스트** 세트로 나누고, **평균 제곱 오차(MSE)**와 같은 '손실 함수'를 사용하여 오류를 측정합니다.

마지막으로, 여러 k값 (예: 이웃 1명, 10명, 70명) 중에서 '최적의' 모델을 선택하고, **R-squared ( $R^2$ )** 값을 통해 이 모델이 "단순히 평균을 추측하는 것보다 얼마나 더 나은지"를 객관적으로 평가합니다.

## 2 핵심 용어 정리

모델링을 학습하기 위해 자주 사용되는 기본 용어들을 정리했습니다.

용어	쉬운 설명 (직관)	원어	비고 (예시)
반응 변수	우리가 예측하려는 '결과' 값입니다.	Response Variable ( $y$ )	제품 판매량, 주택 가격
예측 변수	예측을 위해 사용하는 '입력' 데이터입니다.	Predictor Variable ( $X$ )	TV/라디오 광고 예산
설계 행렬	모든 예측 변수 데이터를 모아둔 행렬 ( $n \times p$ 크기).	Design Matrix ( $X$ )	$n$ =관측치 수, $p$ =예측 변수 수
통계 모델	데이터( $X$ )와 결과( $y$ ) 사이의 관계를 공식화하려는 시도.	Statistical Model ( $\hat{f}$ )	완벽한 관계 $f$ 를 추정( $\hat{f}$ )합니다.
예측 (Prediction)	모델을 사용해 $X$ 가 주어졌을 때 $y$ 의 값을 맞추는 것.	Prediction	정확한 '값'을 맞추는 것이 목표.
추론 (Inference)	$X$ 와 $y$ 가 '어떤 관계'인지 이해하는 것.	Inference	'TV 예산이 1달러 오르면 판매량이 얼마나 오르는가?'에 대한 답.
비모수	모델의 형태를 미리 정하지 않고 데이터에 의존하는 방식.	Non-parametric	kNN이 대표적. 유연합니다.
하이퍼파라미터	모델이 학습하는 것이 아니라, *사람*이 미리 정해주는 값.	Hyperparameter	kNN에서의 $k$ 값 (이웃 수).
훈련/검증/테스트	데이터를 세 조각으로 나누는 것.	Train / Validation / Test	훈련: 모델 학습 (연습문제) 검증: 모델 선택 (모의고사) 테스트: 최종 성능 평가 (실제 시험)
손실 함수	모델이 얼마나 틀렸는지 (오류) 계산하는 함수.	Loss Function	이 값이 낮을수록 좋은 모델입니다.
MSE	(실제값 - 예측값)을 제곱하여 평균 낸 값.	Mean Squared Error	가장 널리 쓰이는 손실 함수.
$R^2$ (결정 계수)	모델이 "단순 평균"보다 얼마나 예측을 잘하는지 (0 1).	R-squared	1에 가까울수록 좋습니다. 0이면 평균과 같습니다.
차원의 저주	예측 변수( $p$ )가 너무 많아질 때 발생하는 문제.	Curse of Dimensionality	데이터가 희박해져 kNN 성능이 저하됩니다.

### 3 통계 모델링 시작하기

#### 3.1 예측이란 무엇인가?

우리는 종종 하나의 변수 값을 예측하기 위해 다른 변수들을 사용합니다.

- **예시 1:** 동영상의 길이, 게시 날짜 등을 바탕으로 다음 주 TikTok 조회수 예측하기.
- **예시 2:** 사용자의 이전 영화 평점, 인구통계학적 데이터를 바탕으로 Netflix 사용자가 높게 평가할 영화 예측하기.

이 강의에서는 간단한 예제인 **광고 데이터셋**을 사용합니다. 200개 시장에서 제품 판매량(**Sales**) 데이터와, 3개 매체(TV, 라디오, 신문)의 광고 예산(\$1,000 단위) 데이터를 가지고 있습니다.

**우리의 목표:** TV, 라디오, 신문 광고 예산을 알 때, 판매량을 예측하는 모델을 만드는 것입니다.

#### 3.2 데이터의 두 가지 역할: 예측 변수( $X$ )와 반응 변수( $y$ )

데이터에는 비대칭성이 존재합니다. 우리가 예측하려는 '판매량'은 다른 변수들(광고 예산)에 의해 영향을 받거나, 측정하기 더 어렵거나, 더 중요할 수 있습니다.

- **반응 변수 (Response Variable,  $y$ ):** 우리가 예측하려는 목표 변수입니다.
  - 다른 이름: 종속 변수(Dependent Variable), 타겟(Target), 결과(Outcome).
  - 예: sales (판매량)
- **예측 변수 (Predictor Variables,  $X$ ):** 예측을 위해 사용하는 입력 변수들입니다.
  - 다른 이름: 독립 변수(Independent Variable), 특성(Features), 공변량(Covariates).
  - 예: TV, radio, newspaper (각 매체별 광고 예산)

데이터는 보통  $n$  개의 행(관측치)과  $p$  개의 열(예측 변수)로 구성됩니다.

#### 3.3 데이터를 '행렬'로 표현하기 (The Design Matrix)

데이터를 수학적으로 다루기 위해 다음과 같이 표기합니다.

- $X$  (**설계 행렬, Design Matrix**):  $n$  개의 관측치(행)과  $p$  개의 예측 변수(열)를 가진 행렬입니다.
- $y$  (**반응 벡터, Response Vector**):  $n$  개의 관측치에 대한  $n$  개의 반응 변수 값을 가진 벡터입니다.

##### □ 예제:

데이터의 형태:  $X$ 와  $y$  만약 5개의 관측치( $n = 5$ )와 3개의 예측 변수( $p = 3$ )가 있다면 데이터는 다음과 같습니다.

	TV	radio	newspaper		sales
$X$ (설계 행렬, $5 \times 3$ )	230.1	37.8	69.2	$y$ (반응 벡터, $5 \times 1$ )	22.1
	44.5	39.3	45.1		10.4
	17.2	45.9	69.3		9.3
	151.5	41.3	58.5		18.5
	180.8	10.8	58.4		12.9

### 주의사항

Pandas와 Sklearn에서의 데이터 차원 (Shape) 데이터 분석 도구(Pandas, Sklearn)를 다룰 때, 벡터의 차원에 유의해야 합니다.

- $X.shape$  (설계 행렬): 항상  $(n, p)$  형태의 2차원입니다. (예:  $(5, 3)$ )
- $y.shape$  (반응 벡터):  $(n,)$  또는  $(n, 1)$  형태일 수 있습니다.
  - $(n,)$ : 1차원 벡터 (Pandas의 **Series**). 예: `df['sales']`
  - $(n, 1)$ : 2차원 행렬 (Pandas의 **DataFrame**). 예: `df[['sales']]`
- 대부분의 머신러닝 라이브러리는 두 형태 모두를 받아들이지만,  $(n,)$  형태를 기본으로 하는 경우가 많습니다. 이 차이로 인해 오류가 발생하기 쉬우니 `.shape` 속성으로 항상 확인하는 습관을 들이는 것이 좋습니다.

## 4 모델이란 무엇인가?

### 4.1 완벽한 모델 vs. 통계 모델

데이터  $X$ 와  $y$  사이의 관계를 찾는 것을 '모델링'이라고 합니다.

아이스크림 비유

- **진짜(True) 모델,  $f$ :** 세상에 존재하는 '완벽한 아이스크림' 레시피. 모든 맛의 조합을 정확히 알고 있습니다. 하지만 우리는 이 레시피를 절대 알 수 없습니다.
- **통계(Statistical) 모델,  $\hat{f}$ :** 우리가 가진 재료(데이터)로 그 '완벽한 아이스크림'을 따라 만들려는 \*시도\*입니다.

우리는  $X$ 와  $y$  사이에 어떤 '진짜' 관계  $f$ 가 있다고 가정합니다.

$$Y = f(X) + \epsilon$$

- $f(X)$ :  $X$ 에 의해 결정되는  $Y$ 의 체계적인 정보 (우리가 찾고 싶은 완벽한 규칙).
- $\epsilon$  (엡실론):  $f(X)$ 로 설명되지 않는 무작위 오차(Noise). 측정의 한계, 우리가 고려하지 못한 변수, 혹은 세상의 본질적인 무작위성 때문에 발생합니다.

**\*\*통계 모델링\*\***이란, 우리가 가진 데이터( $X, y$ )를 사용하여 이 알 수 없는  $f$ 를 가장 잘 근사하는 (따라하는) 함수  $\hat{f}$  (f-hat)을 찾는 과정입니다.

### 4.2 모델의 두 가지 목적: 추론(Inference) vs. 예측(Prediction)

모델  $\hat{f}$ 를 찾는 목적은 크게 두 가지로 나뉩니다.

구분	추론 (Inference)	예측 (Prediction)
목표	$X$ 와 $y$ 사이의 관계를 이해하는 것.	$X$ 가 주어졌을 때 $y$ 의 값을 정확히 맞추는 것.
비유	탐정	궁수 □ 반
주요 질문	"TV 예산이 판매량에 어떻게 영향을 미치는가?"	"TV 예산이 \$150k일 때 예상 판매량은 얼마인가?"
모델 형태	해석 가능한 단순한 모델 (예: 선형 회귀)	정확도 높은 유연한 모델 (예: kNN, 신경망)
모델( $\hat{f}$ ) 평가	$\hat{f}$ 의 형태와 계수(parameter)가 중요한가? <b>Yes</b>	$\hat{f}$ 의 형태는 중요하지 않음 (블랙박스여도 OK). <b>No</b>

이 강의에서는 먼저 이해하기 쉬운 **\*\*예측\*\*** 모델인 kNN부터 다룹니다.

## 5 k-최근접 이웃 (kNN) 알고리즘

### 5.1 가장 단순한 모델: "평균" 모델

예측 변수  $X$ 를 완전히 무시하고 예측하는 가장 단순한 모델은 무엇일까요? 바로 모든  $y$  값의 평균을 사용하는 것입니다.

$$\hat{y} = \bar{y} = \frac{1}{n} \sum_{i=1}^n y_i$$

이 모델은 TV 예산이 \$10k 이든 \$300k 이든 상관없이 항상 동일한 평균 판매량(예: 12.5)을 예측합니다. 이것은 매우 순진한(naive) 모델이지만, 나중에 다른 모델과 비교하기 위한 '최소 기준선(Baseline)' 역할을 합니다.

### 5.2 kNN의 핵심 아이디어: "가까운 이웃" 참고하기

kNN은 평균 모델보다 훨씬 똑똑한, 매우 직관적인 아이디어에 기반합니다.

의사의 진단 비유 환자가 "배가 아프다"고 하며 병원에 왔습니다. 의사는 머릿속으로 '이번 주에 배가 아프다고 왔던 다른 환자 10명'을 떠올립니다. "그 10명은 모두 특정 푸드트럭 음식을 먹고 장염에 걸렸었지. 이 환자도 장염일 가능성이 높다." 즉, **나와 비슷한 사례(이웃)**를 찾아 그들의 결과를 참고하여 현재 사례를 판단(예측)합니다.

kNN 알고리즘은 다음과 같이 작동합니다. (1차원 예시: TV 예산( $x$ )만 사용)

1. **예측할  $x_q$  정하기:** "TV 예산이 \$150k 일 때( $x_q$ ) 판매량( $y_q$ )은?"
2. **거리 계산:**  $x_q$ 와 훈련 데이터에 있는 모든  $x_i$  사이의 거리를 계산합니다. (1차원에서는  $D(x_q, x_i) = |x_q - x_i|$ )
3. **k개의 이웃 찾기:**  $x_q$ 와 가장 가까운  $k$ 개의 데이터 포인트 ( $x_i, y_i$ )를 찾습니다.
4. **평균 내기:** 찾은  $k$ 개 이웃들의  $y_i$  값들의 평균을 내어  $\hat{y}_q$ 로 예측합니다.

$$\hat{y}_q = \frac{1}{k} \sum_{i \in \text{Neighbors}} y_{q_i}$$

### 5.3 k값에 따른 모델의 변화

$k$  값을 어떻게 정하느냐에 따라 모델의 형태가 극적으로 달라집니다.

- $k = 1$  (가장 가까운 1명):
  - 작동: 가장 가까운 이웃 1명의  $y$  값을 그대로 복사합니다.
  - 결과: 모델이 매우 뽀족하고 들쭉날쭉한 계단 형태가 됩니다. (Overfitting)
  - 문제점: 데이터의 아주 작은 잡음(noise)에도 민감하게 반응합니다. 변동성이 너무 큼니다.
- $k = 10$  (적당한 수의 이웃):
  - 작동: 가장 가까운 10명의  $y$  값을 평균냅니다.

- **결과:**  $k = 1$ 보다 훨씬 부드러운 곡선(계단)이 되어 데이터의 전반적인 추세를 잘 따릅니다.
- $k = 70$  (또는  $k = n$ , 아주 많은 이웃):
  - **작동:** 훈련 데이터 70개 전부를 이웃으로 삼아 평균냅니다.
  - **결과:**  $x_q$ 의 위치에 상관없이 항상 전체 데이터의 평균을 반환합니다.
  - **문제점:** 5.1 절에서 본 "가장 단순한 모델"과 같아집니다. 데이터의 지역적 특성을 완전히 무시합니다. (Underfitting)

$k = 1$ 은 너무 복잡하고,  $k = 70$ 은 너무 단순합니다. 우리는 이 사이의 "적절한"  $k$  값을 찾아야 합니다.

#### 5.4 $k$ 는 '하이퍼파라미터'입니다

kNN은 비모수(Non-parametric) 모델입니다. 이는 모델이  $f$ 의 형태를 '직선'이나 '곡선'이라고 미리 가정하지 않고, 데이터 자체에 의존해 형태를 만들기 때문입니다.

하지만 kNN에도  $k$ 라는 파라미터가 있습니다. 이  $k$  값은 모델이 데이터로부터 \*학습하는 값\*이 아니라, 우리가 \*직접 정해줘야 하는 값\*입니다. 이처럼 사람이 모델 학습 전에 직접 설정하는 값을 하이퍼파라미터(Hyperparameter)라고 부릅니다.

**질문:**  $k = 1, k = 10, k = 70$  중에서 "최고의"  $k$ 는 어떻게 찾을 수 있을까요?

## 6 모델 평가하기 (Error Evaluation)

### 6.1 "최고의" 모델이란 무엇인가?

$k$  값에 따라 수많은 모델이 나옵니다. "최고의" 모델을 고르려면 "좋다"는 것을 정의할 기준이 필요합니다. 우리는 모델의 **예측 오류(Error)**가 가장 작은 모델을 "최고"라고 부를 것입니다.

### 6.2 데이터를 나누는 이유: 훈련, 검증, 테스트

모델의 오류를 공정하게 평가하려면 데이터를 명확한 목적에 따라 나눠야 합니다.

시험 공부 비유

- **훈련 세트 (Train Set):** 모델을 학습시키는 데 사용합니다. (kNN에서는 이웃을 찾기 위한 데이터 풀)
  - 비유: 답안지가 있는 연습 문제집. 이걸로 공부합니다.
- **검증 세트 (Validation Set):** 학습된 모델들 중 최고의 하이퍼파라미터( $k$ )를 선택하기 위해 사용합니다.
  - 비유: 모의고사.  $k = 3$  전략과  $k = 10$  전략 중 모의고사 점수가 더 잘 나오는 전략을 선택합니다.
- **테스트 세트 (Test Set):** 선택된 최종 모델의 실제 성능을 딱 한 번 평가하기 위해 사용합니다.
  - 비유: 실제 수능 시험. 모의고사에 너무 최적화되었는지(과적합), 진짜 실력(일반화 성능)이 어느 정도인지 최종 평가합니다.

데이터 분할은 보통 무작위(Randomly shuffle)로 진행됩니다.

#### 주의사항

데이터 오염 (Data Contamination) "모델을 선택하기 위해(하이퍼파라미터 튜닝) 테스트 세트를 사용하는 사람들은 지옥의 특별한 자리가 마련되어 있습니다." (강의 중 인용)  
만약 모의고사( $k$  값 선택)에 수능 문제( )를 미리 보고 푼다면, 그 점수는 진짜 실력이 아닙니다. 모델 선택(검증)에는 검증 세트만 사용하고, 테스트 세트는 최종 평가 전까지 절대 열어보지 않아야 합니다.

#### □ 예제:

scikit-learn의 `train_test_split` 파이썬 라이브러리인 scikit-learn은 데이터를 두 개로 쪼개는 `train_test_split()` 함수만 제공합니다. 훈련/검증/테스트 세 개로 나누려면 이 함수를 두 번 호출해야 합니다.

1. 전체 데이터를 `train_full`과 `test`로 나눕니다. (예: 80% / 20%)
2. 1번에서 나눈 `train_full`을 다시 `train`과 `validation`으로 나눕니다. (예: 75% / 25% → 전체의 60% / 20%)

### 6.3 손실 함수 (Loss Function): 오류를 숫자로 나타내기

검증 세트에서 모델의 오류를 계산하려면, '오류'를 하나의 숫자로 표현해야 합니다.

- 잔차 (Residual): 실제 값과 예측 값의 차이.  $r_i = y_i - \hat{y}_i$
- 문제점: 잔차를 그냥 더하면 양수 오류와 음수 오류가 상쇄되어 오류가 0처럼 보일 수 있습니다. (예: +5, -5 → 합 0)
- 해결책: 오류를 모두 양수로 만들기 위해 제곱하거나 절댓값을 사용합니다.

주요 손실 함수

#### 1. 평균 제곱 오차 (Mean Squared Error, MSE)

- 공식:  $MSE = \frac{1}{n} \sum_{i=1}^n (y_i - \hat{y}_i)^2$
- 특징: 오류를 '제곱'하므로, 큰 오류(Outlier)에 더 큰 패널티를 줍니다. 수학적으로 미분이 가능하여 최적화에 유리하며, 확률론적 정당성(오차가 정규분포를 따른다는 가정 하에 최적)이 있어 가장 널리 쓰입니다.

#### 2. 평균 절대 오차 (Mean Absolute Error, MAE)

- 공식:  $MAE = \frac{1}{n} \sum_{i=1}^n |y_i - \hat{y}_i|$
- 특징: 오류의 '절댓값'을 사용합니다. MSE보다 큰 오류에 덜 민감합니다.

#### 3. 평균 제곱근 오차 (Root Mean Squared Error, RMSE)

- 공식:  $RMSE = \sqrt{MSE}$
- 특징: MSE에 제곱근을 씌운 것입니다. 오류의 단위가 원래  $y$ 의 단위(예:  $\$^2$ 가 아닌  $\$$ )와 같아져 해석이 더 직관적입니다.

## 7 최고의 모델 선택 및 최종 평가

### 7.1 검증 세트를 이용한 $k$ 값 선택

이제 우리는 "최고의  $k$ "를 찾는 명확한 절차를 갖게 되었습니다.

1.  $k$ 의 후보 리스트를 정합니다. (예:  $k = 1, 2, 3, \dots, 10$ )
2. 각  $k$  값에 대해 다음을 반복합니다:
  - 훈련 세트를 사용하여 kNN 모델을 구성합니다.
  - 검증 세트의  $X_{val}$  값을 입력하여  $\hat{y}_{val}$ 을 예측합니다.
  - 검증 세트의 실제  $y_{val}$ 과 예측  $\hat{y}_{val}$  사이의 MSE를 계산합니다.
3. 모든  $k$  값 중에서 검증 세트의 MSE가 가장 낮은  $k$ 를 "최적의 하이퍼파라미터"로 선택합니다.

예를 들어,  $k$  값에 따른 검증 MSE가 아래와 같다고 가정합시다. (그래프로 그리면 U자 형태가 나옴)

k Nearest Neighbors	Validation MSE
1	7.0
2	3.0
<b>3</b>	<b>0.2</b>
4	0.8
...	...
10	5.2

이 경우, 검증 MSE가 0.2로 가장 낮은  $k = 3$ 을 최고의 모델로 선택합니다.

#### 주의사항

무작위 분할의 함정 만약 우리가 우연히 운이 나쁘게 훈련/검증 세트를 분할했다면 어떨까요?  $k = 3$ 이 이 특정 분할에서는 최고였지만, 다른 분할에서는  $k = 5$ 가 최고일 수도 있습니다. 이러한 분할의 불안정성(Variance) 문제는 이후 교차 검증(Cross-Validation)과 같은 고급 기법으로 다루게 됩니다.

### 7.2 모델이 "쓸 만한지" 평가하기: $R^2$ (R-squared)

우리는  $k = 3$ 이 "우리가 가진 모델 중 최고"라는 것을 알아냈습니다. 하지만 이 최고 모델이 "쓸 만한" 모델일까요?

NBA 선수 선발 비유 제가 "저는 우리 학과 교수팀에서 최고의 농구 선수입니다!" ( $\leftarrow k = 3$ 이 다른  $k$ 보다 낫다) 라고 말한다고 해서, NBA 팀이 저를 스카우트해야 할까요?

아닙니다. NBA 팀은 "그래서 르브론 제임스나 다른 NBA 선수들과 비교하면 어떻습니까?" ( $\leftarrow$  절대적인 기준선과 비교) 라고 물어볼 것입니다.

$k = 3$  모델의 MSE가 5.0이라고 합시다. 이  $MSE = 5.0$ 은 좋은 값일까요? 알 수 없습니다.  $y$ 의 단위에 따라 이 값은 달라집니다.

- Sales 단위를 '1,000 units'에서 'single units'로 바꾸면 ( $y$  값이 1000배 커짐)
- MSE는  $1000^2$  배 커져  $5.0 \rightarrow 5,000,000$ 이 됩니다.

MSE 값 자체는 단위에 의존하므로, 모델이 "얼마나 좋은지" 직관적으로 말해주지 못합니다. 우리는 단위에 의존하지 않는, 0점(최악)에서 1점(최고) 사이의 표준화된 점수가 필요합니다. 이것이 바로  $R^2$  (**R-squared**, 결정 계수)입니다.

$R^2$  (R-squared)  $R^2$ 는 우리의 모델( $\hat{f}$ )이 "가장 단순한 모델"(평균 모델,  $\bar{y}$ )에 비해 얼마나 오류를 줄였는지를 비율로 나타냅니다.

- 기준선 (최악): 평균 모델의 오류  $\sum (y_i - \bar{y})^2$
- 우리 모델: 우리 모델의 오류  $\sum (y_i - \hat{y}_i)^2$

$$R^2 = 1 - \frac{\sum (y_i - \hat{y}_i)^2}{\sum (y_i - \bar{y})^2} = 1 - \frac{MSE(\text{우리 모델})}{MSE(\text{평균 모델})}$$

해석:

- $R^2 = 1$ : 완벽한 모델 ( $MSE(\text{모델}) = 0$ ).
- $R^2 = 0$ : 우리 모델이 단순 평균 모델과 성능이 정확히 같음 (쓸모 없음).
- $R^2 < 0$  (음수): 우리 모델이 단순 평균 모델보다 더 나쁨. (훈련 세트에서는 거의 발생하지 않지만, 테스트 세트에서는 발생 가능)

## 8 kNN의 한계: 차원의 저주

### 8.1 kNN을 다차원으로 확장하기

지금까지는 'TV 예산'( $p = 1$ ) 하나만 사용했습니다. 만약 'TV', 'radio', 'newspaper' ( $p = 3$ )를 모두 사용하려면 어떻게 해야 할까요?

kNN의 핵심은 '거리'입니다. 다차원 공간에서 두 점 사이의 거리를 계산하면 됩니다. 가장 일반적인 방법은 유클리드 거리 (Euclidean distance)입니다. (두 점을 잇는 직선 거리, 피타고라스 정리의 확장)

두 점  $x = (x_1, \dots, x_p)$ 와  $x' = (x'_1, \dots, x'_p)$  사이의 거리는 다음과 같습니다:

$$d(x, x') = \sqrt{\sum_{j=1}^p (x_j - x'_j)^2}$$

### 8.2 "차원의 저주"란 무엇인가?

kNN은 직관적이고 강력하지만, 예측 변수의 수( $p$ ), 즉 차원(Dimension)이 증가하면 심각한 문제에 직면합니다. 이를 차원의 저주(Curse of Dimensionality)라고 합니다.

#### 주의사항

차원이 높아질 때 kNN이 겪는 문제들

#### 1. 데이터가 희박해집니다 (Sparse Data):

- **비유:** 학생 100명이 1차원(복도)에 서 있으면 매우 뻥뻥합니다. 2차원(운동장)에 서 있으면 듬성듬성해집니다. 3차원(체육관)에 퍼져 있으면 훨씬 더 희박합니다. 100차원 공간에서는 100명의 학생이 있어도 공간 대부분이 텅 비게 됩니다.
- **영향:** 데이터가 희박해지면, 어떤 점  $x_q$ 에서 "가장 가까운 이웃"조차도 실제로는 엄청나게 멀리 떨어져 있게 됩니다. "이웃"이라는 개념 자체가 무의미해집니다.

#### 2. 거리의 변별력이 사라집니다:

- **현상:** 차원이 매우 높아지면, 한 점에서 다른 모든 점까지의 거리가 거의 비슷해지는 현상이 발생합니다.
- **영향:** "가까운 이웃"과 "먼 이웃"을 구분하기 어려워집니다.

#### 3. 특성 스케일링 (Feature Scaling) 문제:

- **문제:** 유클리드 거리는 값의 스케일에 매우 민감합니다.
- **예시:** 예측 변수로 '키'(150 190cm)와 '연봉'(30,000,000 100,000,000원)을 사용한다고 합시다. 거리 계산  $(x_j - x'_j)^2$ 에서, '연봉'의 차이(수백만)가 '키'의 차이(수십)를 완전히 압도해버립니다. 사실상 '키'는 무시되고 '연봉'만으로 이웃을 찾게 됩니다.
- **해결책:** kNN을 사용하기 전, 모든 예측 변수( $X$ )를 동일한 범위(예: 0 1 사이)로 스케일링 (Scaling) 또는 정규화(Normalization)하는 전처리가 필수적입니다.

**결론:** kNN은 차원이 낮을 때( $p$ 가 작을 때) 매우 효과적이고 직관적인 모델이지만, 차원이 높은

데이터에는 적합하지 않을 수 있습니다.

## 9 학습 목표 요약 (Checklist)

이 강의를 통해 다음을 수행할 수 있어야 합니다.

- $\bar{b}$

변수( $y$ )와 예측 변수( $X$ ), 그리고 설계 행렬을 정의할 수 있다.

통계 모델링의 두 가지 목적(추론, 예측)의 차이점을 설명할 수 있다.

k-최근접 이웃(kNN) 알고리즘이 비모수적 방식이며 "유사한 사례"에 의존함을 설명할 수 있다.

1차원 데이터에서 kNN의 이웃을 찾고, 그들의 값을 평균내어 예측값을 계산할 수 있다.

$k$  값이 모델의 유연성(복잡도)에 어떤 영향을 미치는지 설명할 수 있다 ( $k = 1$  vs  $k = n$ ).

모델 평가를 위해 데이터를 훈련/검증/테스트 세트로 나누는 이유를 설명할 수 있다.

MSE, RMSE 등 오류 측정 지표(손실 함수)를 사용하여 모델의 성능을 정량화할 수 있다.

검증 세트의 MSE를 사용하여 최적의 하이퍼파라미터( $k$ )를 선택하는 과정을 설명할 수 있다.

MSE가 단위에 의존하는 문제를 설명하고,  $R^2$ 가 왜 필요한지 설명할 수 있다.

$R^2$ 의 값을 (0, 1, 음수) 해석할 수 있다.

유클리드 거리를 사용하여 kNN을 다차원 데이터로 확장하는 방법을 설명할 수 있다.

차원의 저주가 무엇인지, 그리고 특성 스케일링이 왜 kNN에 필수적인지 설명할 수 있다.