

선형 모델

고려대학교 석준희

*ChatGPT: Optimizing
Language Models
for Dialogue*

We've trained a model called ChatGPT which interacts in a conversational way. The dialogue format makes it possible to challenge incorrect premises, and request more information. ChatGPT is a sibling model to GPT-3, which follows an instruction-response format.



목차

- 선형 회귀 모델
- 로지스틱 회귀 모델
- 프로그래밍 실습

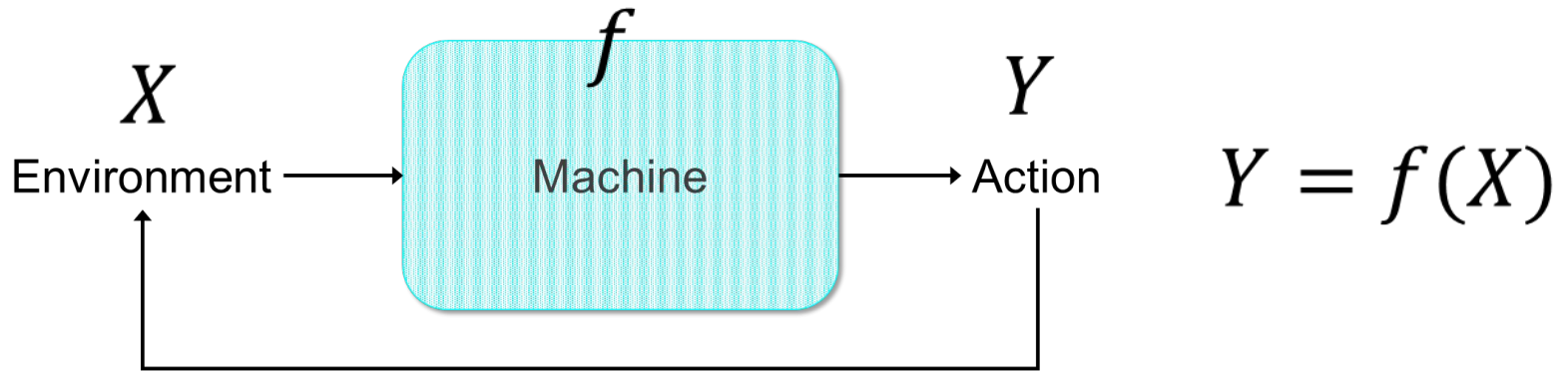
선형 모델

선형 회귀 모델



학습의 목표와 종류

- 인공지능은 인간의 지능을 모방하여 관측된 데이터를 바탕으로 어떤 행동에 대한 판단을 내리는 주체

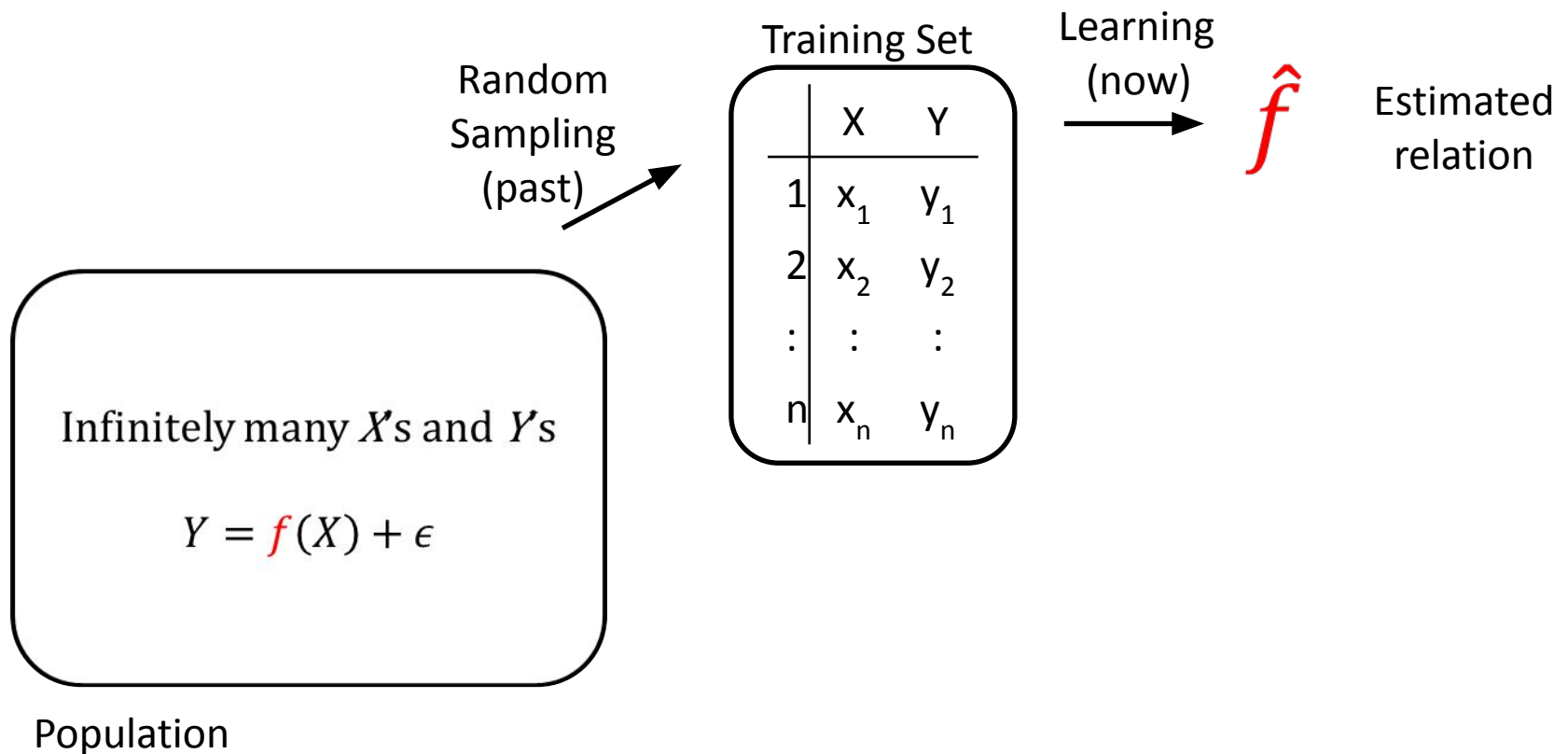


- 함수 $f()$ 를 주어진 관측데이터로부터 추정하는 것이 (기계)학습의 목표
- 지식으로부터 찾는다면 지식 기반의 인공지능
- 회귀(regression):** Y 가 수치형 변수일 때
 - 기대 수명, 임금, 주가, 카카오톡 메시지의 수 등
- 분류(classification):** Y 가 범주형 변수일 때
 - 성공/실패, 성별, 자동차의 종류, 꽃 품종 등



선형 모델 (Linear Model)

- 모집단에서의 실제 $f()$ 가 선형성을 갖고 있다고 가정
 - 선형 회귀 (linear regression) 모델: 회귀 문제에 적용
 - 로지스틱 회귀 (logistic regression) 모델: 분류 문제에 적용





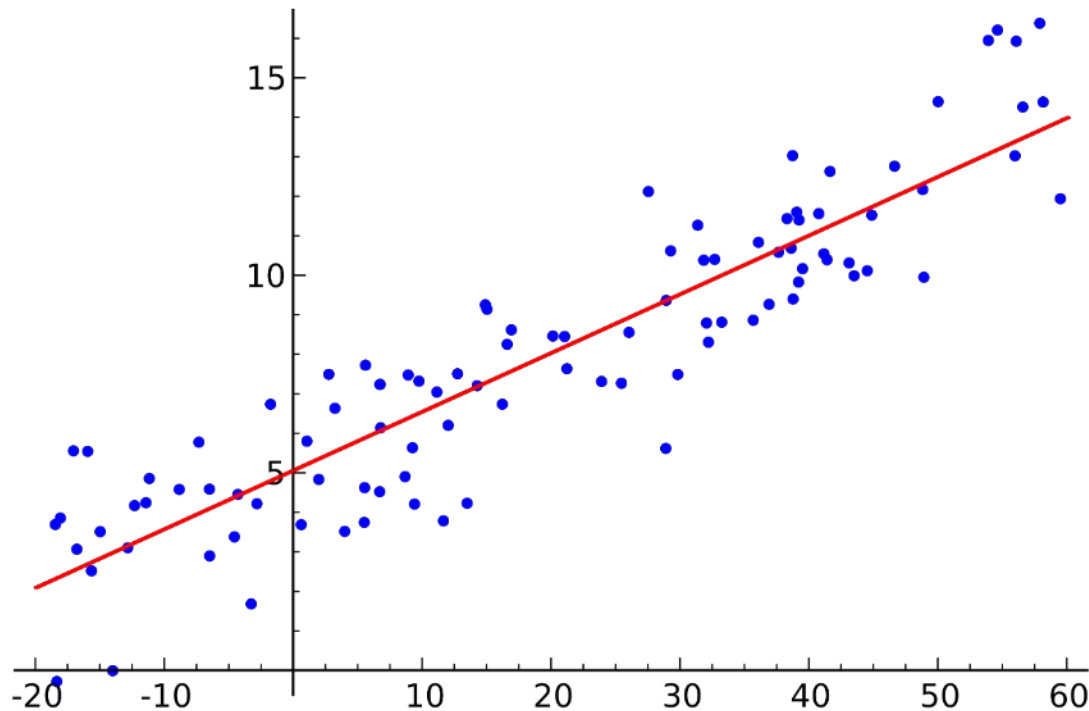
단순 선형 회귀 (Simple Linear Regression)

- 하나의 독립변수(X)와 하나의 출력변수(Y)에 대한 모델

$$Y \approx \beta_0 + \beta_1 X \quad \beta_0: \text{절편}, \beta_1: \text{계수}$$

- 각 데이터는 다음과 같이 모델링 됨

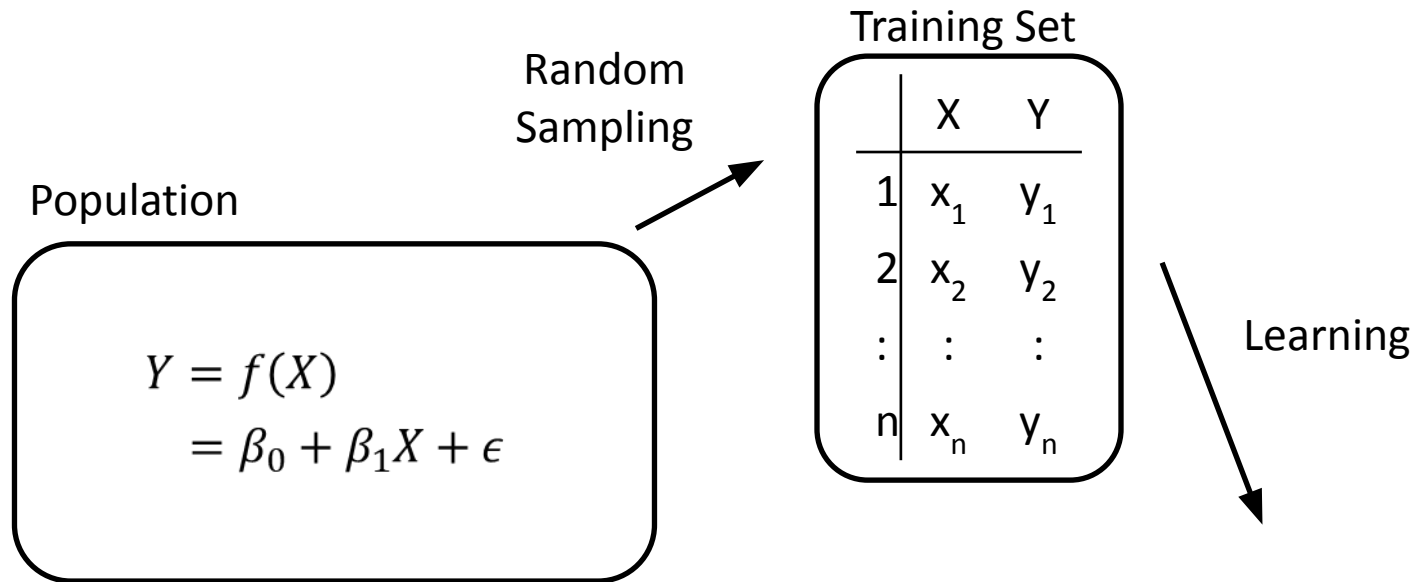
$$y_i = \beta_0 + \beta_1 x_i + e_i \text{ for } i = 1 \cdots n.$$





단순 선형 회귀 (Simple Linear Regression)

- 하나의 독립변수(X)와 하나의 출력변수(Y)에 대한 모델: $Y \approx \beta_0 + \beta_1 X$



Model Estimation

$$\hat{f}(X) = \hat{\beta}_0 + \hat{\beta}_1 X$$

Prediction

$$\hat{y} = \hat{f}(x) = \hat{\beta}_0 + \hat{\beta}_1 x$$

Mean Square Error

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



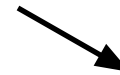
모델 파라미터의 추정

- 최소제곱법(Least squares)을 이용하여 파라미터를 추정

$$L(\hat{\beta}_0, \hat{\beta}_1) = \sum_{i=1}^n e_i^2 = \sum_{i=1}^n (y_i - \hat{\beta}_0 - \hat{\beta}_1 x_i)^2$$

손실함수
Loss Function

- 손실 함수를 미분하여 쉽게 계산 가능



Square loss function

- 추정값

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

$$\hat{\beta}_0 = \bar{y} - \hat{\beta}_1 \bar{x}$$

- 위의 추정값은 훈련 데이터의 에러를 최소화하는 것이지만, 반드시 평가 데이터의 에러를 최소화하지는 않음
 - 평가 데이터의 에러를 최소화하기 위해 모델 선택 과정이 필요



일반 선형 회귀 (Multiple Linear Regression)

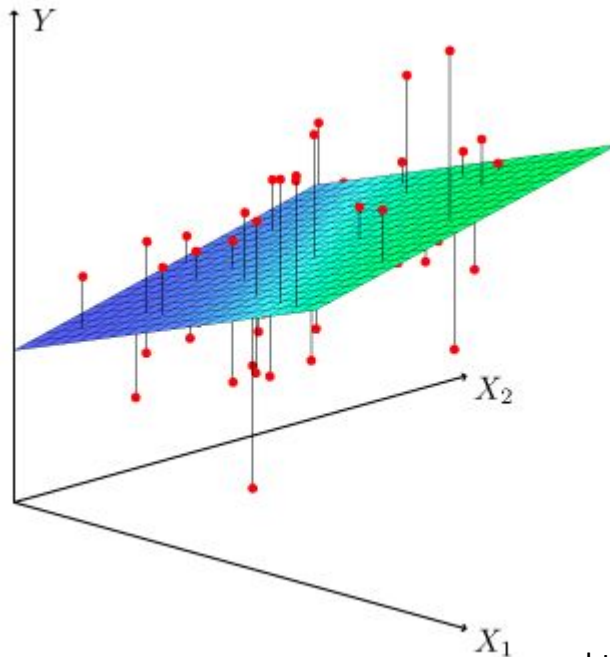
- 하나의 출력 변수와 여러 개의 입력 변수

$$Y \approx \beta_0 + \beta_1 X_1 + \cdots \beta_p X_p$$

- 제곱에러 (square-error) 손실함수를 최소화 하여 파라미터를 추정

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

$$\hat{y}_i = \hat{\beta}_0 + \hat{\beta}_1 x_{1i} + \cdots \hat{\beta}_p x_{pi}$$



$$Y \approx \beta_0 + \beta_1 X_1 + \beta_2 X_2$$

- Y의 변화량은 각 x의 변화량에 비례
- 각 x는 독립적으로 기여



회귀 모델의 평가

- 추정한 모델이 얼마나 좋은지 평가하는 척도
- 평균제곱에러 (MSE: Mean Square Error)

$$MSE = \frac{1}{n} \sum_{i=1}^n (y_i - \hat{f}(x_i))^2 \quad RMSE = \sqrt{MSE}$$

- 결정계수 R^2 (coefficient of determination)
 - Y가 얼마나 X에 의해 설명되는지에 대한 비율

$$R^2 = 1 - \frac{\sum_{i=1}^n (y_i - \hat{y}_i)^2}{\sum_{i=1}^n (y_i - \bar{y})^2} = 1 - \frac{(\text{unexplained variance})}{(\text{total variance of } Y)}$$

- 보통은 0과 1 사이이지만, 모델이 아주 나쁜 경우 음의 값을 갖기도 함
- 이러한 척도는 트레이닝 셋과 테스트 셋 양쪽 모두에서 계산 가능



예제: 광고비와 판매량

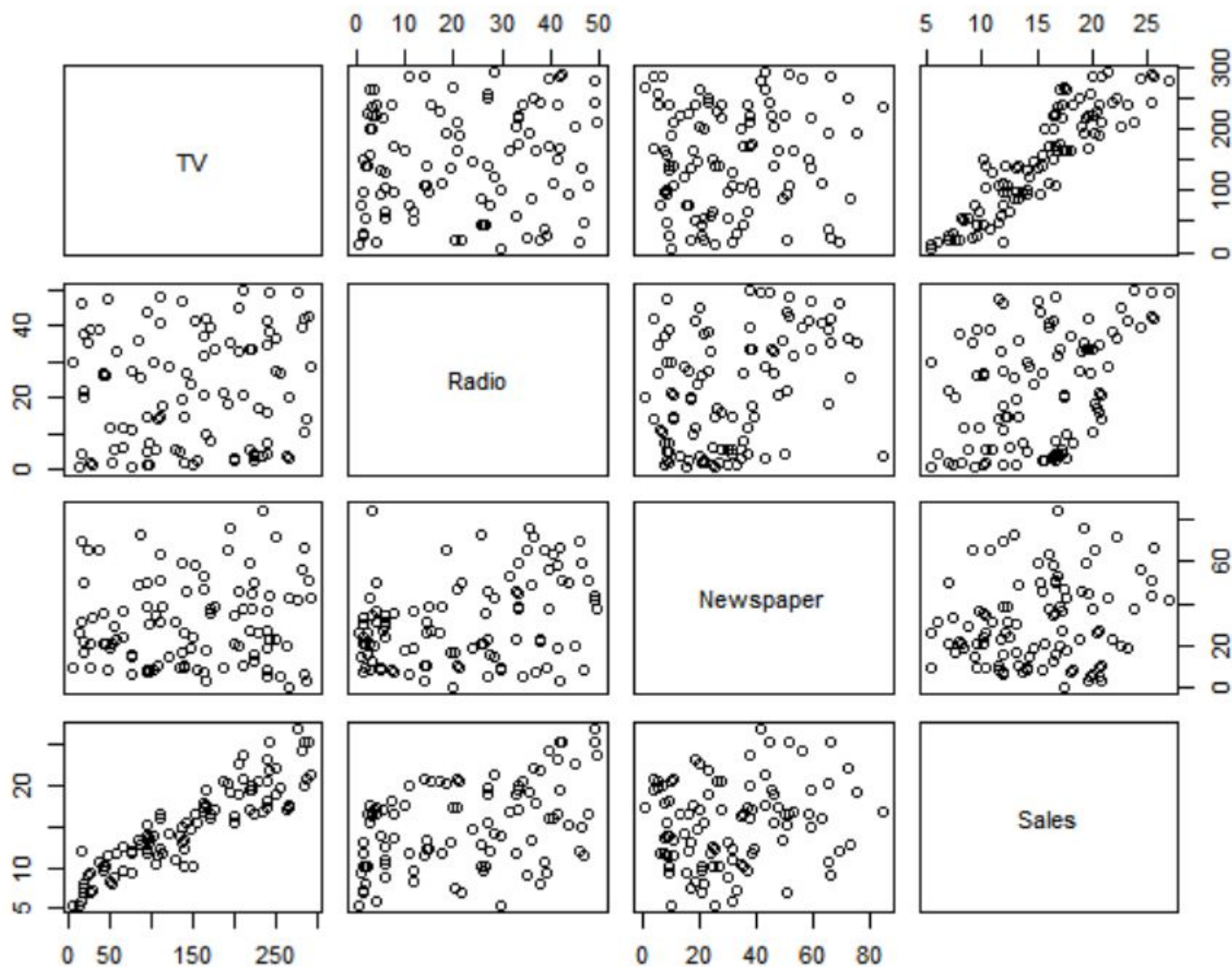
- 상품의 판매량(Sales)을 각 매체(TV, Radio, Newspaper)의 광고비로 설명
 - $Sales = \beta_0 + \beta_1 TV + \beta_2 Radio + \beta_3 Newspaper + \varepsilon$.
- 전체 200개의 표본(샘플: sample) 중에서...
 - 훈련 데이터: 임의로 선택된 100 샘플
 - 평가 데이터: 나머지 100개의 샘플

- 데이터 행렬

1	TV	Radio	Newspaper	Sales
2	230.1	37.8	69.2	22.1
3	44.5	39.3	45.1	10.4
4	17.2	45.9	69.3	12
5	151.5	41.3	58.5	16.5
6	180.8	10.8	58.4	17.9
7	8.7	48.9	75	7.2
8	57.5	32.8	23.5	11.8
9	120.2	19.6	11.6	13.2
10	8.6	2.1	1	4.8
11	199.8	2.6	21.2	15.6

예제: 광고비와 판매량

- 탐색적 데이터 분석

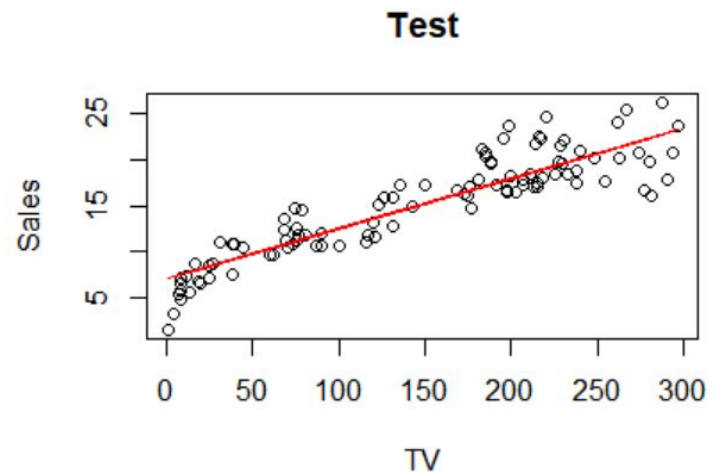
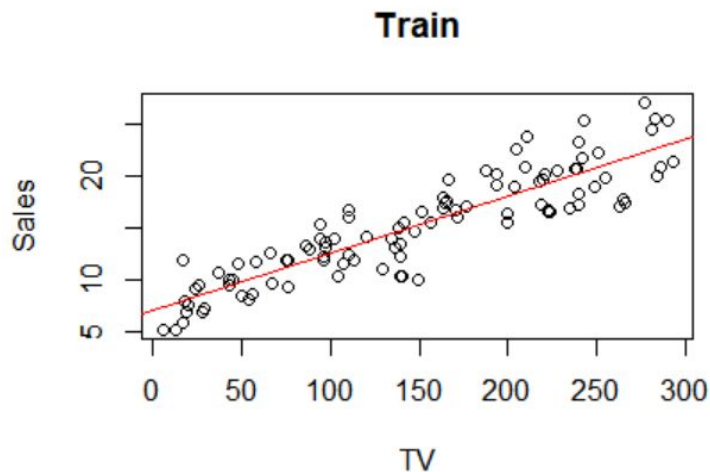




예제: 광고비와 판매량

- TV광고비만을 이용한 모델
 - $\text{Sales} = \beta_0 + \beta_1 \text{TV} + \varepsilon$.
- 결과
 - 모델: $\text{Sales} \sim 7.074 + 0.055 \text{ TV}$

	MSE	R2
Train	5.4193	0.8216
Test	5.2871	0.8017





예제: 광고비와 판매량

- 모든 변수를 이용한 모델
 - $\text{Sales} = \beta_0 + \beta_1 \text{TV} + \beta_2 \text{Radio} + \beta_3 \text{Newspaper} + \varepsilon$
- 결과
 - 모델: $\text{Sales} \sim 4.478 + 0.054 \text{ TV} + 0.116 \text{ Radio} + 0.0002 \text{ Newspaper}$

	MSE	R2
Train	2.4050	0.9167
Test	3.0439	0.8858

- 단순한 모델과 복잡한 모델
 - 어느 모델이 더 복잡한가?
 - 어느 모델이 트레이닝 셋에서 더 성능이 좋은가?
 - 어느 모델이 테스트 셋에서 더 성능이 좋은가? 항상 그럴 것인가?
 - 테스트 셋 결과를 보지 않고 테스트 셋에서 더 성능이 좋은 모델을 선택할 수 있을까?



범주형 변수의 표현

- 입력 변수가 범주형 변수인 경우...

$$Y \approx \beta_0 + \beta_1 X_1 + \beta_2 X_2 + \beta_3 X_3$$

- X_1 : 나이, 연속형 변수
- X_2 : 성별, 범주형 변수 (남성/여성)
- X_3 : 지역, 범주형 변수 (서울/경기/인천)

- 일반적으로 **가변수(dummy variable)**로 변환하여 모델링
 - 성별: $X_{2M} = 1$ (남성) or 0 (여성)
 - 지역: $X_{3S} = 1$ (서울) or 0 (서울 외), $X_{3K} = 1$ (경기) or 0 (경기 외)

$$Y \approx \beta_0 + \beta_1 X_1 + \beta_2 X_{2M} + \beta_3 X_{3S} + \beta_4 X_{3K}$$

- 일반적으로 K개의 범주를 갖는 변수에 대해서 K-1 개의 가변수가 필요



상호작용(Interaction)의 고려

- 일반적 선형 회귀 모델: 두 변수의 영향력이 독립적

$$Y \approx \beta_0 + \beta_1 X_1 + \beta_2 X_2$$

- 상호작용을 고려한 모델: 두 변수의 영향력이 비독립적

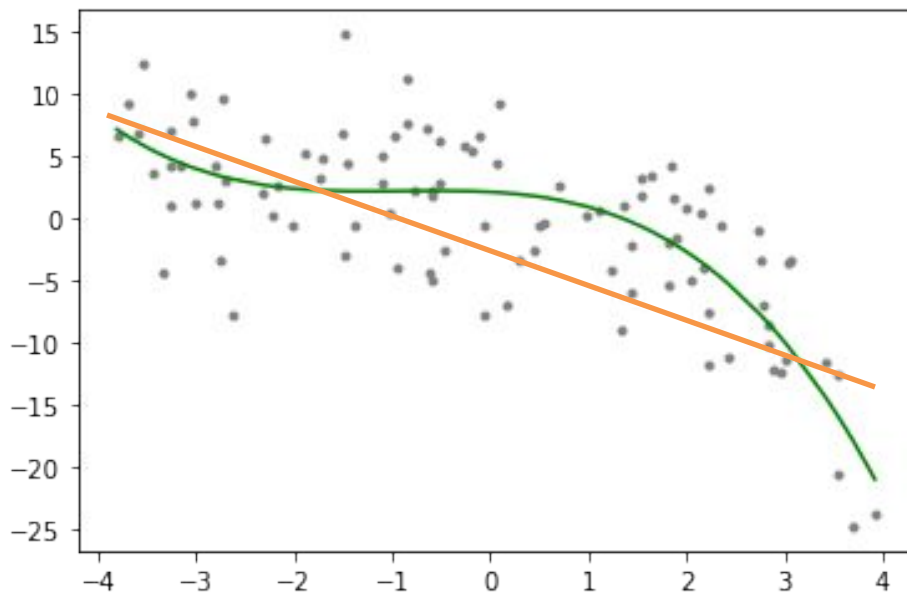
$$Y \approx \beta_0 + \beta_1 X_1 + \beta_2 X_2 + \beta_3 X_1 X_2$$

- 상호작용 항 $X_1 X_2$ 를 새로운 변수 X_3 처럼 취급
- 상호작용을 고려한 모델 vs. 일반 모델
 - 어느 모델이 더 복잡한가?
 - 어느 모델이 트레이닝 셋에서 더 성능이 좋은가?
 - 어느 모델이 테스트 셋에서 더 성능이 좋은가?



다항 회귀 (Polynomial Regression)

- $Y \approx \beta_0 + \beta_1 X$ vs. $Y \approx \beta_0 + \beta_1 X + \beta_2 X^2$.
- 일반 선형 회귀 (1차 회귀 모델) vs. 다항회귀 (2,3, ... 차 회귀 모델)
- 고차항을 새로운 변수처럼 취급
- 모델의 복잡성 vs. 고차항
- 상호작용을 고려한 모델 vs. 일반 모델
 - 어느 모델이 더 복잡한가?
 - 어느 모델이 트레이닝 셋에서 더 성능이 좋은가?
 - 어느 모델이 테스트 셋에서 더 성능이 좋은가?



https://en.wikipedia.org/wiki/Polynomial_regression



기계학습 vs. 통계분석

- 선형 회귀 모델은 현대의 기계학습과 전통적인 통계에서 모두 중요한 모델이지만, 이 둘은 약간 다른 관점을 갖고 있음
- 통계 모델
 - 주로 해석에 초점을 맞추고 있음
 - 적은 계산량을 필요로 하는 이론적 추정을 주로 이용
 - 상대적으로 적은 데이터 양을 가정
- 기계학습 모델
 - 주로 예측에 초점을 맞추고 있음
 - 많은 계산량을 필요로 하는 실험적 추정을 주로 이용
 - 상대적으로 많은 데이터 양을 가정



기계학습 vs. 통계분석

- 두 모델 중 어느 모델을 선택할 것인가?
 - $f_1()$: $\text{sales} = \beta_0 + \beta_1\text{TV} + \beta_2\text{radio} + \beta_3\text{newspaper} + \varepsilon$
 - $f_2()$: $\text{sales} = \beta_0 + \beta_1\text{TV} + \beta_2\text{radio} + \varepsilon$
- 통계 모델
 - 모델의 파라미터가 0인지 아닌지 통계적 검정을 이용해 확인

	Coefficient	Std. error	t-statistic	p-value
Intercept	2.939	0.3119	9.42	< 0.0001
TV	0.046	0.0014	32.81	< 0.0001
radio	0.189	0.0086	21.89	< 0.0001
newspaper	-0.001	0.0059	-0.18	0.8599

- 모델의 실제 에러를 RSE를 통해 확인: $\sqrt{\frac{1}{n-p-1} \sum_{i=1}^n (y_i - \hat{y}_i)^2}$

- 기계학습 모델
 - 모델의 실제 에러 및 예측 성능을 다른 데이터를 이용해 확인

선형 모델

로지스틱 회귀 모델



분류 문제의 표현

- 회귀문제 (Regression): Y 가 연속형 변수일 때
- 분류문제 (Classification): Y 가 범주형 변수일 때
- 클래스 혹은 범주를 숫자로 표현이 가능할까?
 - $Y = A \text{ or } B \rightarrow Y = 1 \text{ or } 2$: 가능함
 - $Y = A, B \text{ or } C \rightarrow Y = 1, 2 \text{ or } 3$: 불가능함
 - 기본적으로 회귀문제로 치환하여 해결하는 것이 불가능
- 많은 분류의 문제들이 클래스 Y 를 직접 예측하기 보다는 Y 가 특정 클래스일 확률 $\Pr[Y = k|X]$ 를 예측하고자 함

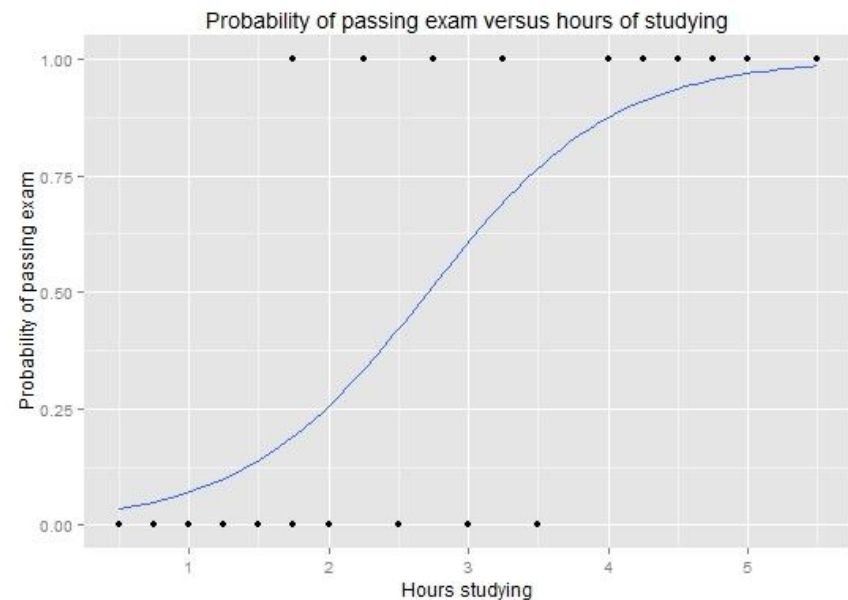
$$\Pr[Y = k|X] \sim f(X)$$



이진 분류 (Binary Classification)

- 이진 분류: 범주/클래스가 두 개인 경우
- 예제: 공부시간 vs. 시험합격
 - Hours : 공부시간 (숫자)
 - Pass: 합격 여부 (합격 or 불합격)

	공부시간	합격여부	$\Pr[Y=1 X]$
1	0.50	불합격	0
2	3.30	합격	1
3	1.75	합격	1
4	3.00	불합격	0



https://en.wikipedia.org/wiki/Logistic_regression



로지스틱 회귀 (Logistic Regression)

- 클래스에 대한 확률을 시그모이드 함수를 이용하여 모델링
- 이진분류 ($Y = 1$ or 0)에 대하여

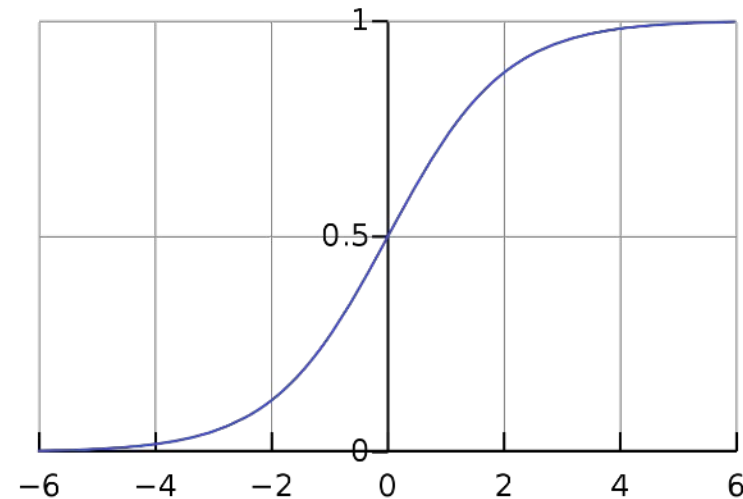
$$\Pr[Y = 1|X] = \frac{e^{\beta_0 + \beta_1 X}}{1 + e^{\beta_0 + \beta_1 X}} = \frac{1}{1 + e^{-(\beta_0 + \beta_1 X)}}$$

$$\Pr[Y = 0|X] = \frac{1}{1 + e^{\beta_0 + \beta_1 X}} = \frac{e^{-(\beta_0 + \beta_1 X)}}{1 + e^{-(\beta_0 + \beta_1 X)}}$$

$$\log \left(\frac{\Pr[Y = 1|X]}{\Pr[Y = 0|X]} \right) = \beta_0 + \beta_1 X$$

승수(odds)

로지트 (logit)



https://en.wikipedia.org/wiki/Sigmoid_function



파라미터의 추정

- 우도 (Likelihood): 어떤 모델을 가정했을 때 현재 데이터를 관측할 확률
- 로지스틱 회귀에서의 우도

$$\text{모델: } \Pr[Y = 1|X] = \frac{e^{\beta_0 + \beta_1 X}}{1 + e^{\beta_0 + \beta_1 X}}, \quad \Pr[Y = 0|X] = \frac{1}{1 + e^{\beta_0 + \beta_1 X}}$$

$$\text{우도: } l(\beta_0, \beta_1) = \prod_{i:y_i=0} \frac{1}{1 + e^{\beta_0 + \beta_1 x_i}} \prod_{i:y_i=1} \frac{e^{\beta_0 + \beta_1 x_i}}{1 + e^{\beta_0 + \beta_1 x_i}}$$

- 예제

	공부시간	합격여부	$\Pr[Y=1 X]$
1	0.50	불합격	0
2	3.30	합격	1
3	1.75	합격	1
4	3.00	불합격	0



파라미터의 추정

• 최대우도법 (Maximum Likelihood)

- 왜 하필 우리는 이 데이터를 관측하고 있을까? 그것은 이 데이터를 관측할 확률이 제일 높기 때문이다!
- 우도를 제일 크게 만드는 파라미터가 진짜 파라미터

$$l(\beta_0, \beta_1) = \prod_{i:y_i=0} \frac{1}{1 + e^{\beta_0 + \beta_1 x_i}} \prod_{i:y_i=1} \frac{e^{\beta_0 + \beta_1 x_i}}{1 + e^{\beta_0 + \beta_1 x_i}}$$

- 보통 우도의 미분을 통해 파라미터를 찾는 것은 계산이 어려움
- 우도의 로그값(Log-likelihood)을 최대화하는 파라미터를 찾음
 - 로그 함수의 특성상 $f(x)$ 를 최대화 하는 것은 $\log(f(x))$ 를 최대화 하는 것과 같음

$$\hat{\beta}_0, \hat{\beta}_1 = \operatorname{argmax}(l) = \operatorname{argmax}(\log l)$$

$$\frac{\partial \log l}{\partial \beta_0} = 0 \quad \frac{\partial \log l}{\partial \beta_1} = 0 \quad \Rightarrow \quad \hat{\beta}_0, \hat{\beta}_1$$

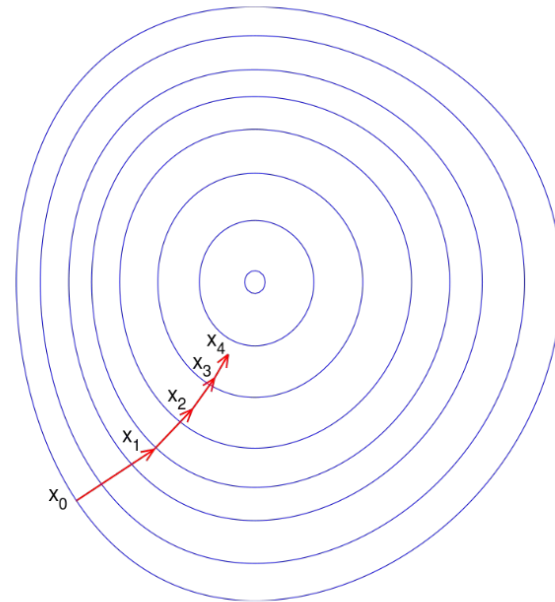


파라미터의 추정

- 수치적 해법
 - 우도에 대한 미분을 수식으로 풀 수 없어 수치적으로 풀어야 함
 - 임의의 파라미터에서 시작하여 경사(미분값)을 따라 업데이트해 가는 방식
 - 예: Newton-Raphson 법, 경사하강법 등

$\beta^{(0)}$: random position

$$\beta^{(i+1)} = \beta^{(i)} - \lambda \frac{\partial L(\beta^{(i)})}{\partial \beta}$$



https://en.wikipedia.org/wiki/Gradient_descent



일반적인 로지스틱 회귀 (Multiple Logistic Regression)

- 하나 이상의 독립 변수에 대해서 선형 회귀와 같이 확장

$$\log\left(\frac{\Pr[Y = 1|X]}{\Pr[Y = 0|X]}\right) = \beta_0 + \beta_1 X$$

$$\log\left(\frac{\Pr[Y = 1|X]}{\Pr[Y = 0|X]}\right) = \beta_0 + \beta_1 X_1 + \beta_2 X_2 + \cdots + \beta_p X_p$$

$$\Pr[Y = 1|X] = \frac{e^{\beta_0 + \beta_1 X_1 + \beta_2 X_2 + \cdots + \beta_p X_p}}{1 + e^{\beta_0 + \beta_1 X_1 + \beta_2 X_2 + \cdots + \beta_p X_p}}$$

- 선형회귀에서 사용되었던 다양한 기법을 바로 적용 가능
 - 가변수를 이용한 범주형 변수의 표현
 - 상호작용의 고려
 - 고차원 변수의 도입
 - 모델의 복잡성에 대한 고려도 동일



분류 모델의 평가

- 정확도 (Accuracy): 각 클래스를 정확히 맞춘 비율

$$Acc = \frac{1}{n} \sum_{i=1}^n I(y_i \neq \hat{f}(x_i))$$

- 혼동행렬 (Confusion Matrix): 실제값과 예측값을 행렬의 형태로 표현

		truth			
		A	B	C	D
predicted	A	70	10	15	5
	B	8	67	20	5
	C	0	11	88	1
	D	4	10	14	72



이진 분류 모델의 평가

- 두 개의 클래스에 대한 2x2 혼동 행렬
 - 양성 (Positive): 우리가 찾고 싶은 클래스 (e.g. 질병, 부도 등)
 - 음성 (Negative): 다른 하나의 클래스 (e.g. 정상, 신용 등)

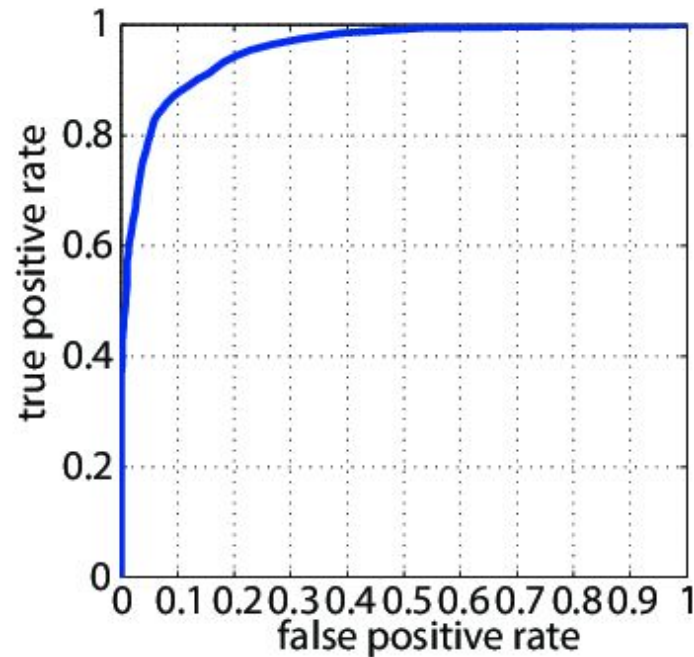
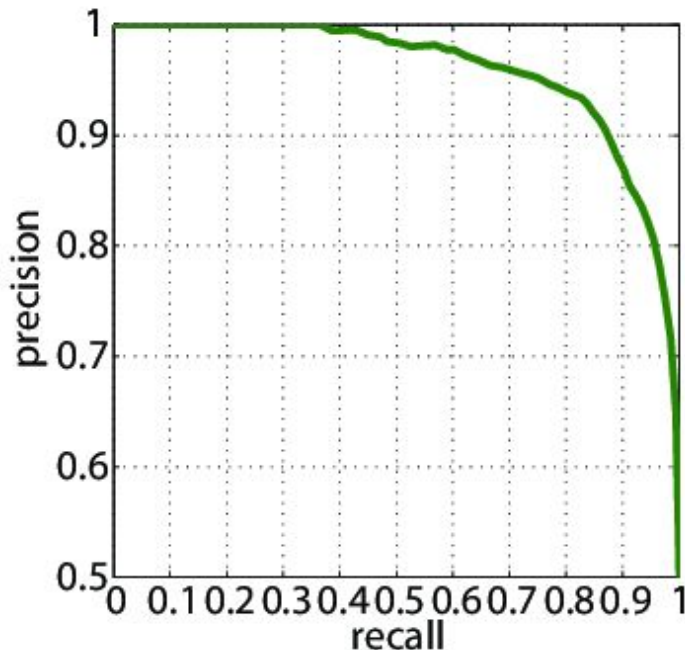
	Truly Positive	Truly Negative
Predicted Positive	True Positive (TP)	False Positive (FP)
Predicted Negative	False Negative (FN)	True Negative (TN)

- **정확도(Accuracy)** = $(TP+TN)/(TP+FP+FN+TN)$
- **재현율(Recall, True Positive Rate, Sensitivity)** = $TP/(TP+FN)$
- **정밀도(Precision, Positive Predictive Value)** = $TP/(TP+FP)$
- **위양성률(False Positive Rate)** = $FP/(FP+TN)$
- **F_1 Score** = $2 \cdot \frac{Precision \cdot Recall}{Precision + Recall} = \left(\frac{P^{-1} + R^{-1}}{2} \right)^{-1} = \frac{TP}{TP + (FP + FN)/2}$



이진 분류 모델의 평가

- **Precision-Recall curve:** recall이 증가함에 따라 precision은 대체로 감소하지만 증가하기도 함
 - **mAP (mean Average Precision):** 단조감소화한 PR 커브의 면적
- **Receive-operating character (ROC) curve:** FPR이 증가함에 따라 TPR은 항상 같거나 증가
 - **AUC (area under curve):** ROC 커브의 면적





예제: 부도 예측

- Output: default
- Input: balance, income
- Data description

	Default: Yes	Default: No	
Train	174	4,826	5,000
Test	159	4,841	5,000

- Model: default ~ balance + income

Train Set	Cond. Yes	Cond. No
Pred. Yes	59	19
Pred. No	115	4,807

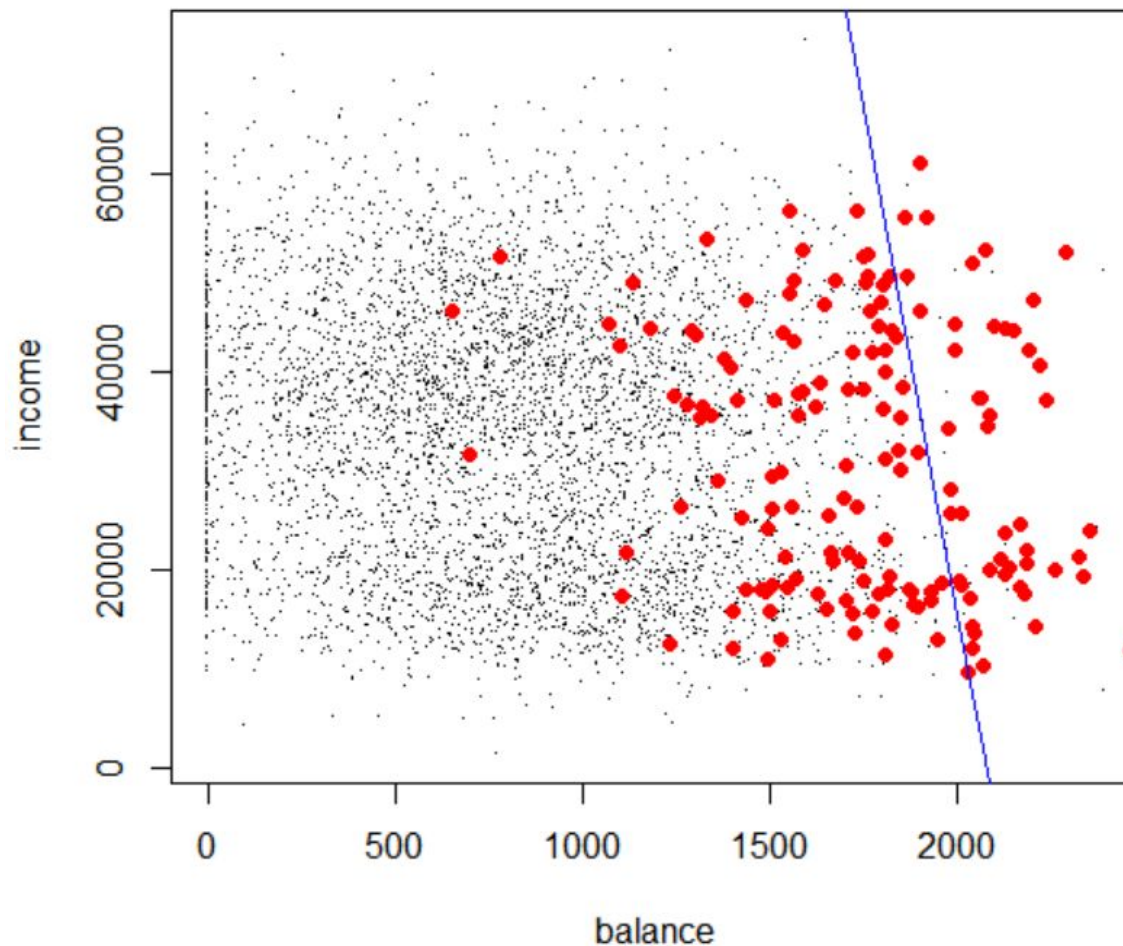
Test Set	Cond. Yes	Cond. No
Pred. Yes	48	22
Pred. No	111	4,819

	Acc	Recall	Precision	FPR	F1
Train	0.97	0.34	0.76	0.00	0.47
Test	0.97	0.30	0.69	0.00	0.42



예제: 부도 예측

- 모델의 판단 경계 (decision boundary): $\text{default} \sim \text{balance} + \text{income}$
 - 판단의 경계는 항상 선형 (linear): 선형 모델!!



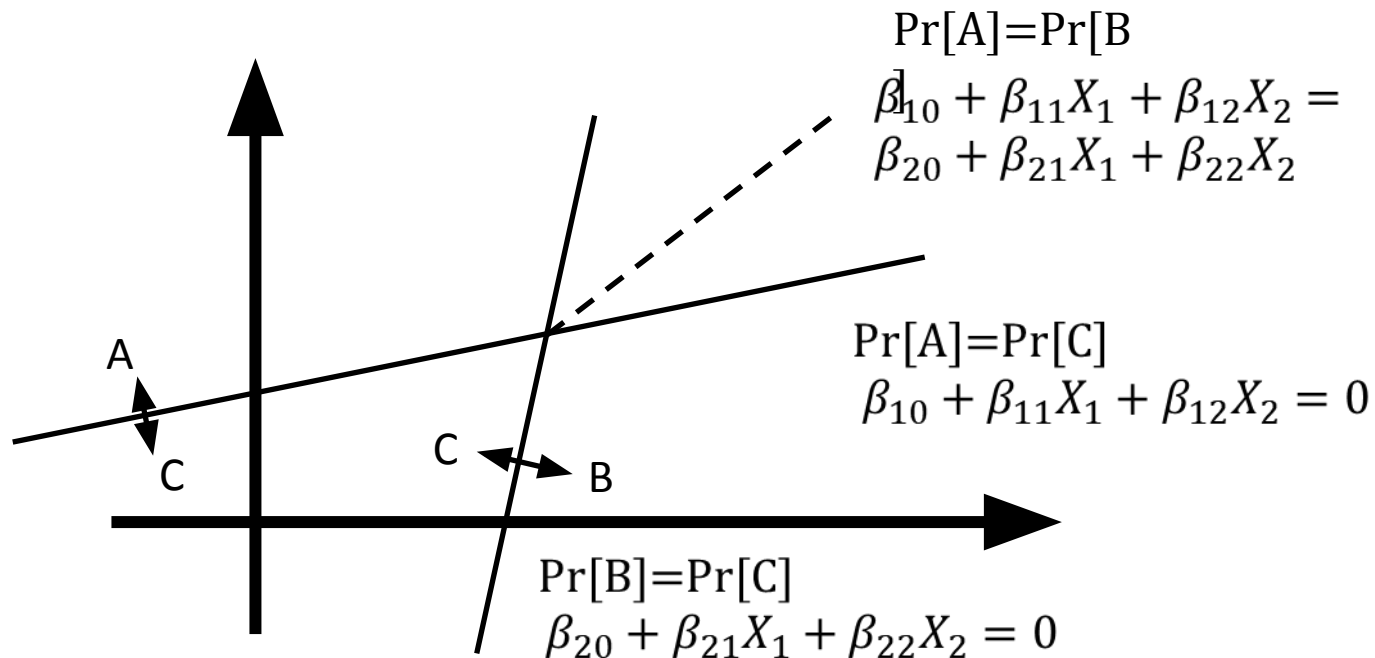


멀티 클래스 분류 (Multi-class Classification)

- 세 개의 클래스에 대한 분류문제는 두 개의 이진분류 문제로 풀 수 있음
- $Y = A, B, \text{ or } C$ 에 대하여, A vs C 와 B vs. C 로 나누고 확률을 계산

$$\log\left(\frac{\Pr[Y = A|X]}{\Pr[Y = C|X]}\right) = \beta_{10} + \beta_{11}X_1 + \beta_{12}X_2 \quad \log\left(\frac{\Pr[Y = B|X]}{\Pr[Y = C|X]}\right) = \beta_{20} + \beta_{21}X_1 + \beta_{22}X_2$$

$$\Pr[Y = A|X] + \Pr[Y = B|X] + \Pr[Y = C|X] = 1$$



선형 모델


프로그래밍 실습



선형 회귀 모델

당뇨병 데이터 셋

```
[ ] from sklearn.datasets import load_diabetes
    X, y = load_diabetes(return_X_y=True, as_frame=True)
```

 X.head()



	age	sex	bmi	bp	s1	s2	s3	s4	s5	s6
0	0.038076	0.050680	0.061696	0.021872	-0.044223	-0.034821	-0.043401	-0.002592	0.019907	-0.017646
1	-0.001882	-0.044642	-0.051474	-0.026328	-0.008449	-0.019163	0.074412	-0.039493	-0.068332	-0.092204
2	0.085299	0.050680	0.044451	-0.005670	-0.045599	-0.034194	-0.032356	-0.002592	0.002861	-0.025930
3	-0.089063	-0.044642	-0.011595	-0.036656	0.012191	0.024991	-0.036038	0.034309	0.022688	-0.009362
4	0.005383	-0.044642	-0.036385	0.021872	0.003935	0.015596	0.008142	-0.002592	-0.031988	-0.046641

```
[ ] y[:5]
```

```
0    151.0
1     75.0
2    141.0
3    206.0
4    135.0
Name: target, dtype: float64
```



선형 회귀 모델

```
[ ] # 훈련 데이터와 평가 데이터를 임의로 분할  
    from sklearn.model_selection import train_test_split  
    xtrain, xtest, ytrain, ytest = train_test_split(X,y,test_size=0.4,random_state=42)
```

```
▶ print( xtrain.shape, xtest.shape )
```

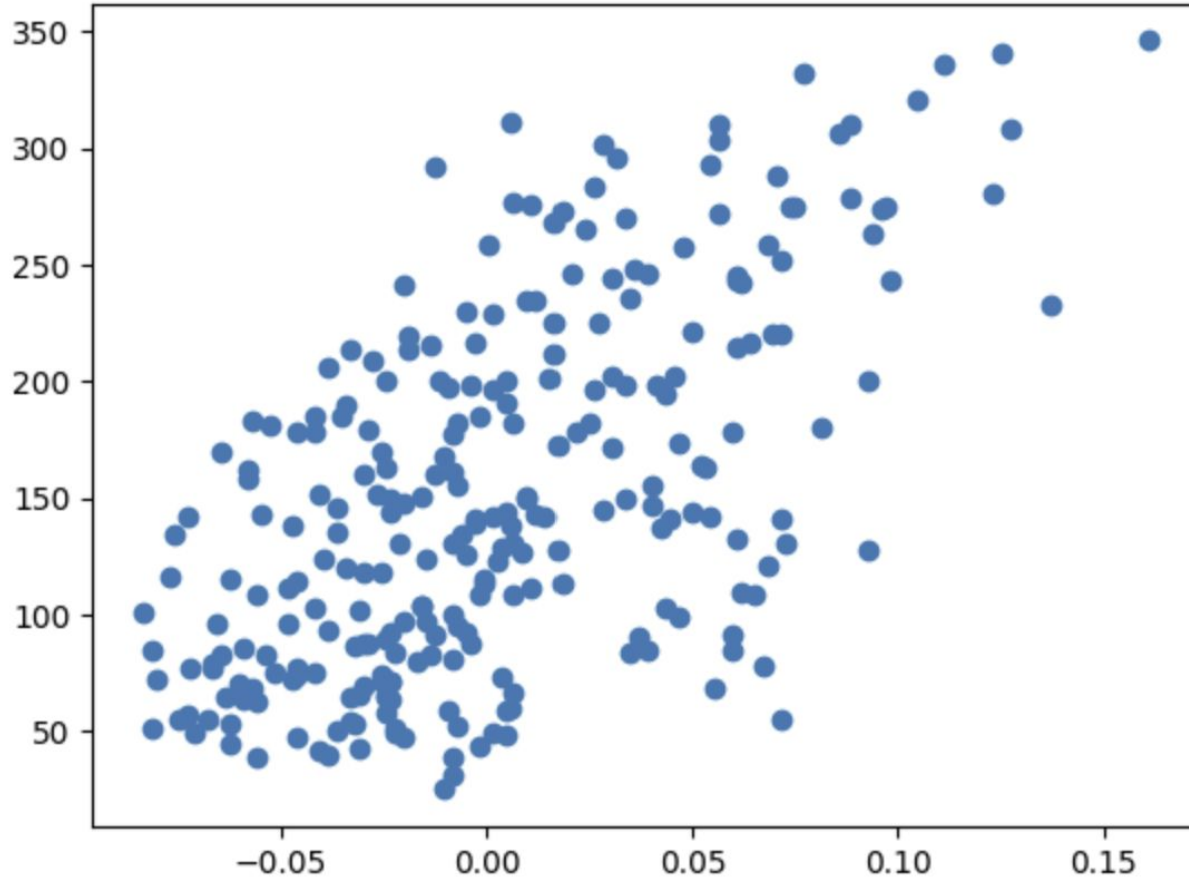
```
☞ (265, 10) (177, 10)
```



```
[ ] # 변수 중에서 bmi 만 추출
xtrain_simple = xtrain[['bmi']]
xtest_simple = xtest[['bmi']]
```

```
▶ # 훈련데이터에서 x와 y의 관계
plt.scatter(xtrain_simple,ytrain)
```

☞ <matplotlib.collections.PathCollection at 0x7efe2d5341f0>





선형 회귀 모델

```
[ ] # 선형 회귀 모델 선언
    from sklearn.linear_model import LinearRegression
    f = LinearRegression()
```

```
[ ] f.fit(xtrain_simple,ytrain) # 모델 훈련
```

▼ LinearRegression

LinearRegression()

```
[ ] print( f.intercept_, f.coef_ ) # 파라미터 확인
```

```
148.53674347978227 [980.74210468]
```

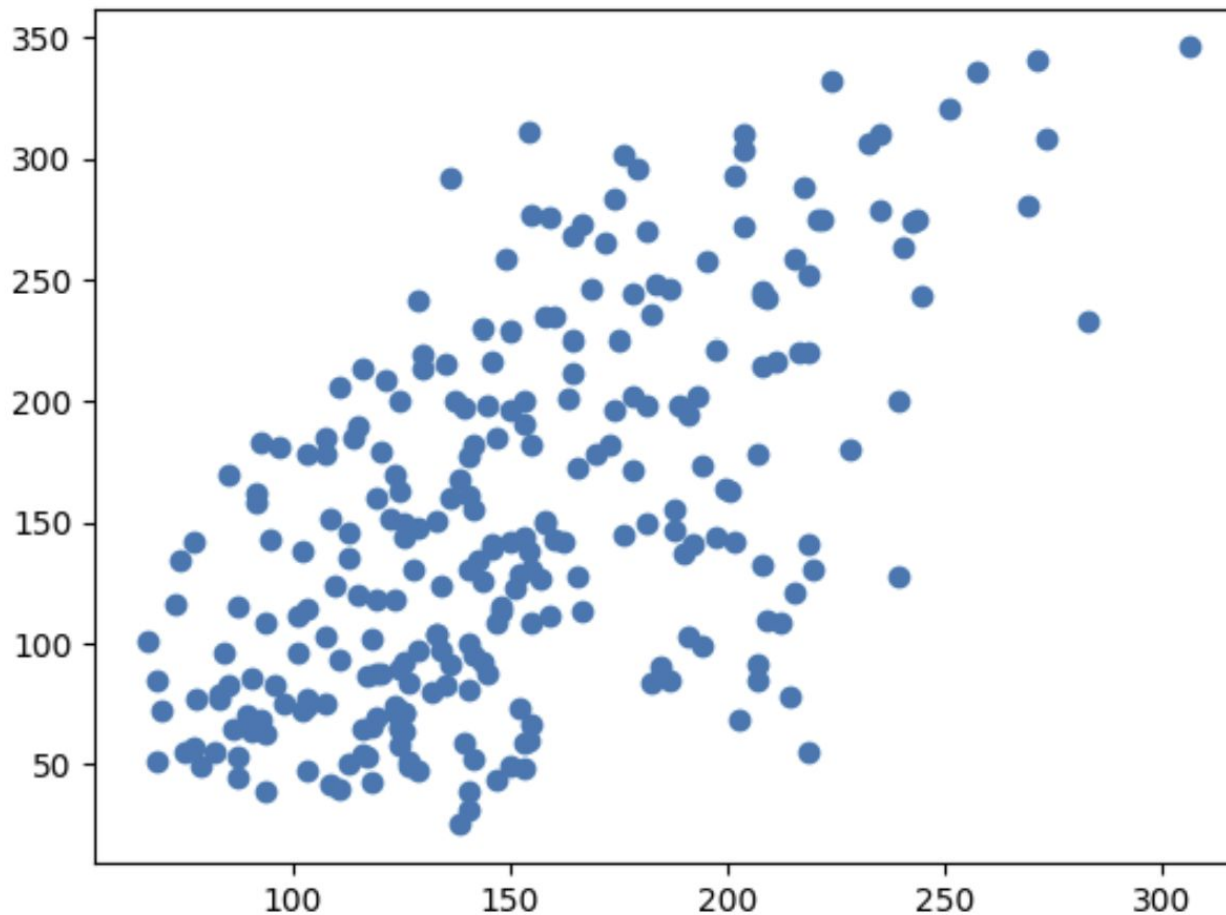
```
[ ] f.predict([[0.01]]) # 그 값에 대한 예측
```

```
/usr/local/lib/python3.10/dist-packages/sklearn/base
warnings.warn(
array([158.34416453])
```

선형 회귀 모델

```
▶ # 모든 훈련데이터에 대해서 예측  
ytrain_hat = f.predict(xtrain_simple)  
plt.scatter(ytrain_hat, ytrain) # 예측값과 실제값 사이의 산점도
```

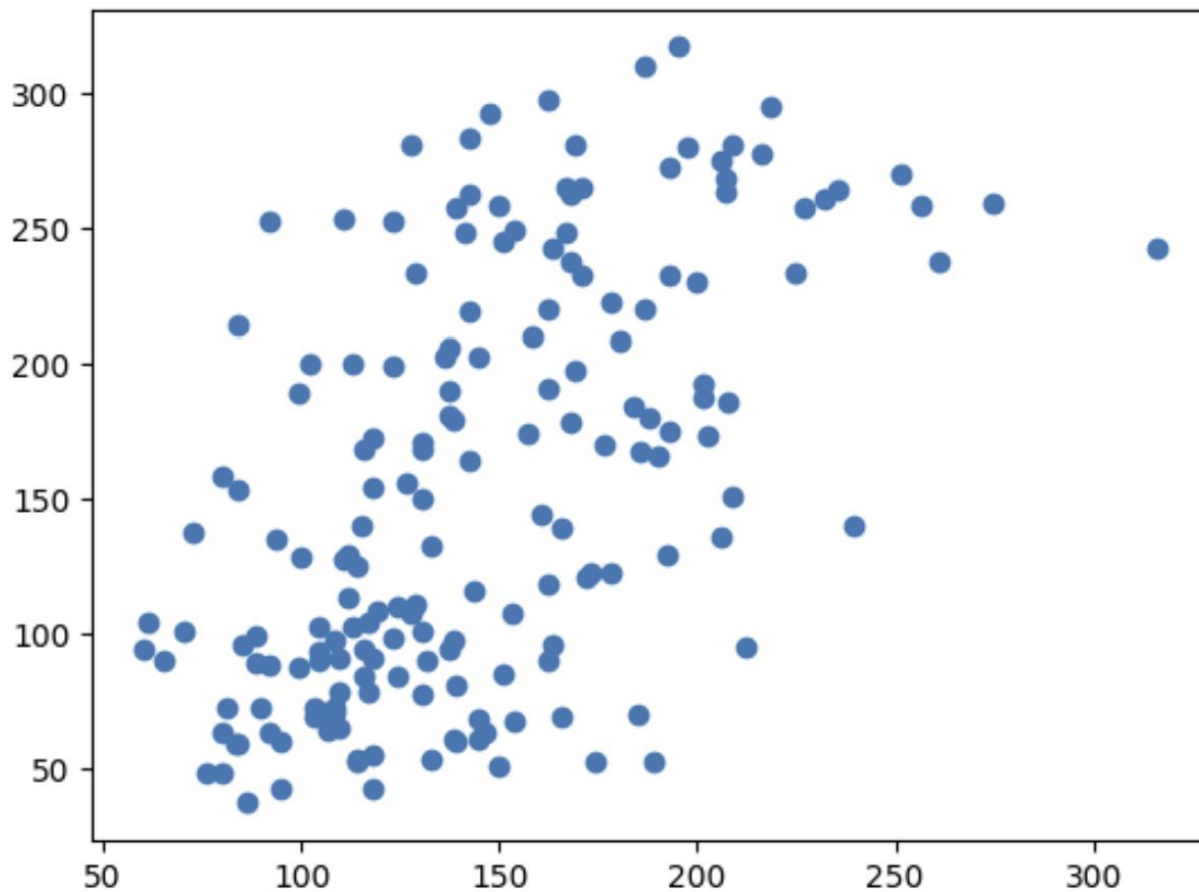
☞ <matplotlib.collections.PathCollection at 0x7efe392fe320>



선형 회귀 모델

```
# 모든 평가 데이터에 대해서 예측  
ytest_hat = f.predict(xtest_simple)  
plt.scatter(ytest_hat, ytest)
```

☞ <matplotlib.collections.PathCollection at 0x7efe28d4c9d0>





선형 회귀 모델

```
[ ] f.score(xtrain_simple,ytrain) # 훈련 데이터에 대한 R2  
0.3686078890927438
```

```
[ ] f.score(xtest_simple,ytest) # 평가 데이터에 대한 R2  
0.2991646176262257
```



선형 회귀 모델

일반 회귀 분석

```
[ ] from sklearn.linear_model import LinearRegression  
    f = LinearRegression()
```

```
[ ] f.fit(xtrain,ytrain)
```

▼ LinearRegression
LinearRegression()

```
[ ] print( f.intercept_, f.coef_ )
```

```
148.9285083717007 [ 18.08799763 -227.04344876  592.27723487  361.54123241 -655.90738774  
353.71636413  14.41265469 142.87369371  594.01542882  31.67317969]
```

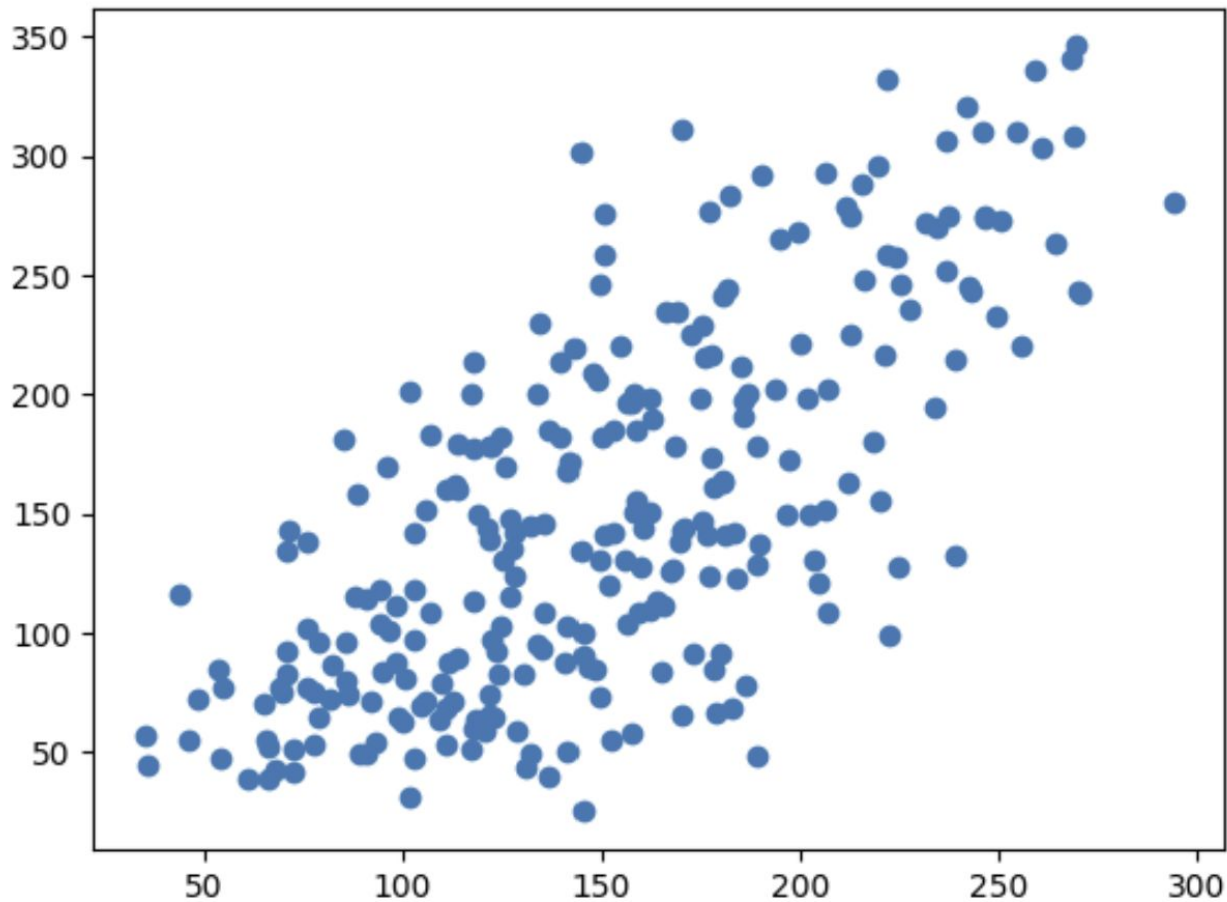
선형 회귀 모델



```
# 모든 훈련데이터에 대해서 예측  
ytrain_hat = f.predict(xtrain)  
plt.scatter(ytrain_hat, ytrain) # 예측값과 실제값 사이의 산점도
```



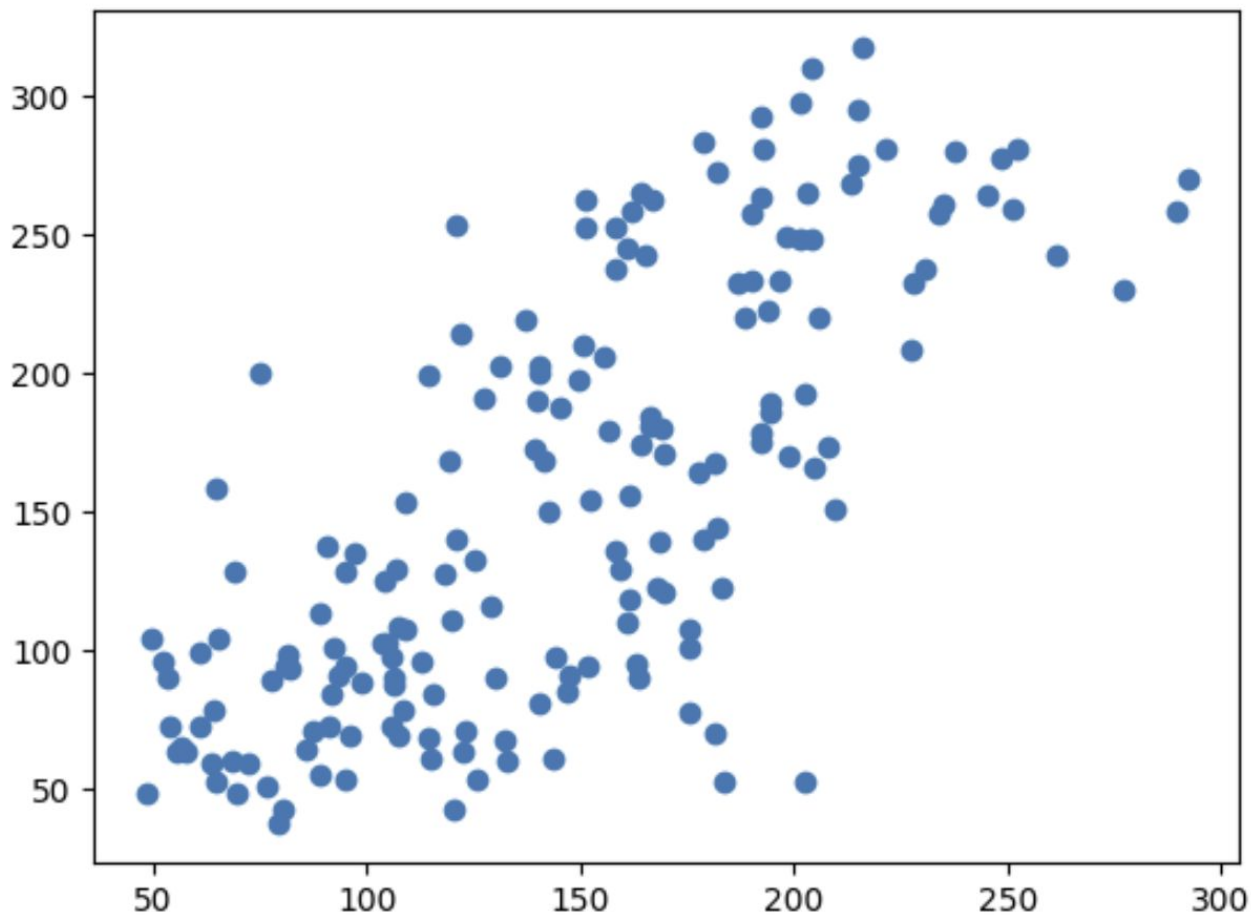
```
<matplotlib.collections.PathCollection at 0x7efe28dba740>
```



선형 회귀 모델

```
# 모든 평가 데이터에 대해서 예측  
ytest_hat = f.predict(xtest)  
plt.scatter(ytest_hat, ytest)
```

<matplotlib.collections.PathCollection at 0x7efe28c2fd90>





선형 회귀 모델

```
[ ] # R2 평가  
print( f.score(xtrain,ytrain), f.score(xtest,ytest) )
```

0.5072191031715794 0.515743631390243



로지스틱 회귀 모델

아이리스 데이터 셋

```
[ ] import pandas as pd
    from sklearn.datasets import load_iris

    X, y = load_iris(return_X_y=True, as_frame=True)
    X.columns = ['SepalLength', 'SepalWidth', 'PetalLength', 'PetalWidth']
    X = X[50:]
    y = y[50:]
```

```
[ ] from sklearn.model_selection import train_test_split
    xtrain, xtest, ytrain, ytest = train_test_split(X, y, test_size=0.4, random_state=42)
```




로지스틱 회귀 모델

```
[ ] xtrain.head()
```

	SepalLength	SepalWidth	PetalLength	PetalWidth
99	5.7	2.8	4.1	1.3
84	5.4	3.0	4.5	1.5
57	4.9	2.4	3.3	1.0
145	6.7	3.0	5.2	2.3
77	6.7	3.0	5.0	1.7

```
[ ] ytrain[:5]
```

```
99    1
84    1
57    1
145   2
77    1
Name: target, dtype: int64
```

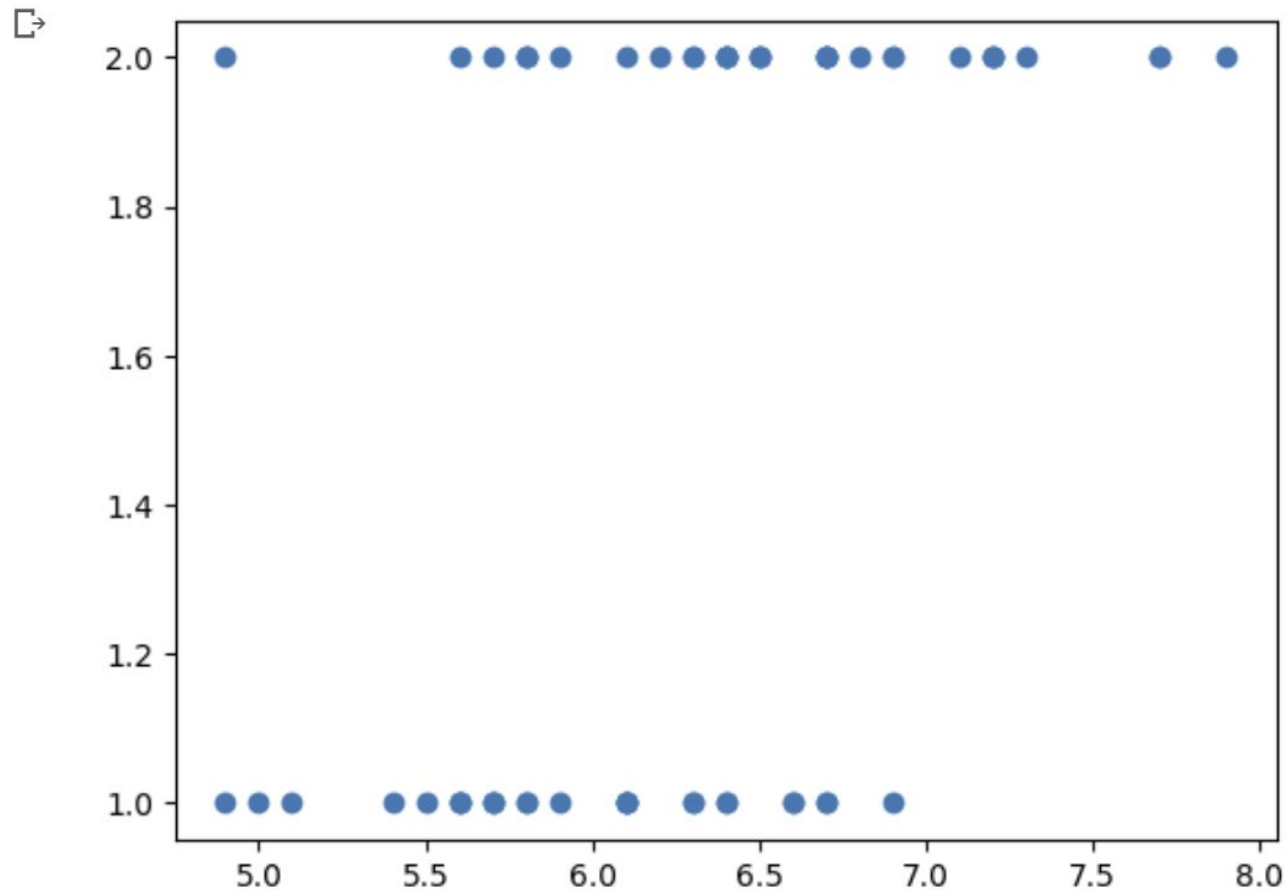
```
 print( xtrain.shape, xtest.shape )
```

```
 (60, 4) (40, 4)
```

단순 로지스틱 회귀

```
[ ] # 변수 중에서 SepalLength 만 추출  
xtrain_simple = xtrain[['SepalLength']]  
xtest_simple = xtest[['SepalLength']]
```

```
▶ # SepalLength와 target (Species)와의 관계  
plt.scatter(xtrain_simple, ytrain)
```





로지스틱 회귀 모델

```
[ ] # 로지스틱 회귀 모델 로딩
    from sklearn.linear_model import LogisticRegression
    f = LogisticRegression()
```

```
[ ] # 모델 훈련
    f.fit(xtrain_simple, ytrain)
```

▼ LogisticRegression
LogisticRegression()

```
[ ] # 파라미터 확인
    print( f.intercept_, f.coef_ )
```

```
[-8.20141709] [[1.34436502]]
```



로지스틱 회귀 모델



로지스틱 회귀 모델

```
[ ] # 훈련데이터 전체에 대한 예측
ytrain_hat = f.predict(xtrain_simple)
```

```
[▶] # 혼동 행렬
pd.crosstab(ytrain_hat,ytrain)
```

↗

target	1	2
row_0		
1	18	8
2	9	25

```
[ ] # 평가데이터에 대한 예측과 혼동행렬
ytest_hat = f.predict(xtest_simple)
pd.crosstab(ytest_hat,ytest)
```

target	1	2
row_0		
1	16	3
2	7	14

```
[▶] # 정확도(Accuracy) 성능 평가
print( f.score(xtrain_simple,ytrain) )
print( f.score(xtest_simple,ytest) )
```

```
↗ 0.7166666666666667
0.75
```



로지스틱 회귀 모델

일반 로지스틱 회귀 모델

```
[ ] from sklearn.linear_model import LogisticRegression  
    f = LogisticRegression()  
    f.fit(xtrain,ytrain)
```



▼ LogisticRegression
LogisticRegression()

```
[ ] print( f.intercept_, f.coef_ )
```


```
[-12.0380239] [[-0.37085631 -0.45441801  2.54024233  1.96076702]]
```



로지스틱 회귀 모델


```
[ ] ytrain_hat = f.predict(xtrain)
    ytest_hat = f.predict(xtest)
```

```
[▶] pd.crosstab(ytrain_hat,ytrain)
```

[↗] 

	target	1	2
row_0			
1		25	1
2		2	32

```
[▶] pd.crosstab(ytest_hat,ytest)
```

[↗] 

	target	1	2
row_0			
1		22	0
2		1	17

```
[ ] # 정확도(Accuracy) 성능 평가
    print( f.score(xtrain,ytrain) )
    print( f.score(xtest,ytest) )
```

0.95
0.975

감사합니다