Machine Learning

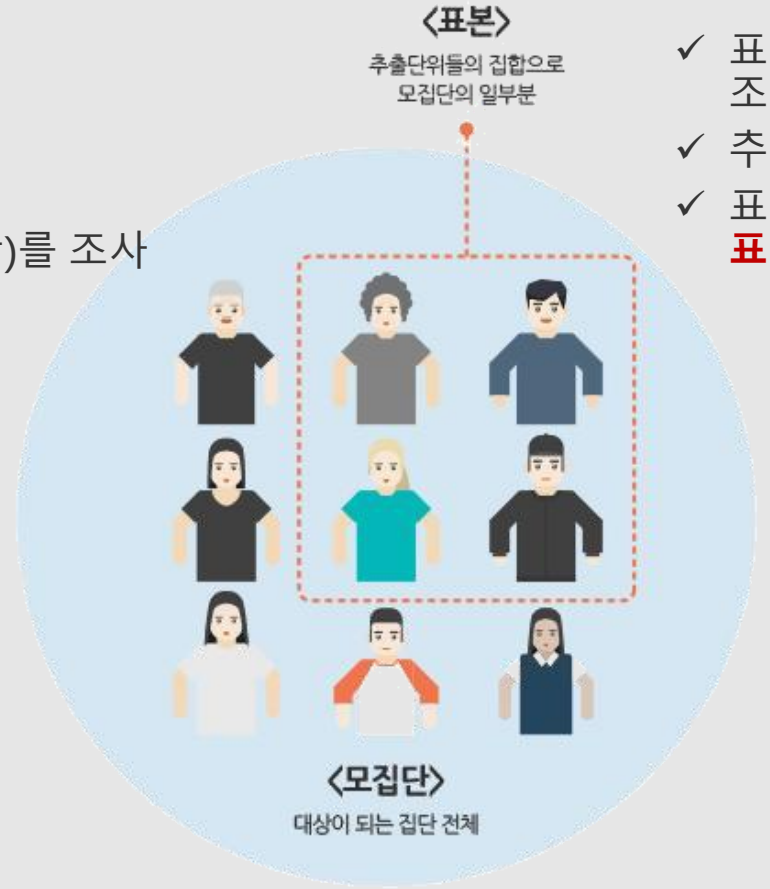
순서

- ✓ 1. 모델링이란
- ✓ 2. 무작정 모델링 따라하기
- ✓ 3. ML Process
- ✓ 4. Regression

Modeling이란...

전수조사와 표본(Sample)조사

✓ 전수조사 : 전체(모집단)를 조사



✓ 표본조사 : 표본을 추출(sampling)하여 조사
✓ 추출 방식 : **무작위** 추출
✓ 표본을 여러 번 추출하다 보면 **표본오차**가 발생된다.

모델, 모델링

✓ Model
- 어떤 목적을 달성하기 위해 실세계를 단순하게 표현한 것
- 주어진 목적에 따른 중요성 여부, 가정, 정보나 추적성에 대한 제약에 따라 단순화 됨

✓ 데이터 과학에서의 Model
- 우리가 관심을 갖는 변수(타겟값)을 예측하는 공식
- 수학공식, 혹은 논리적인 문장

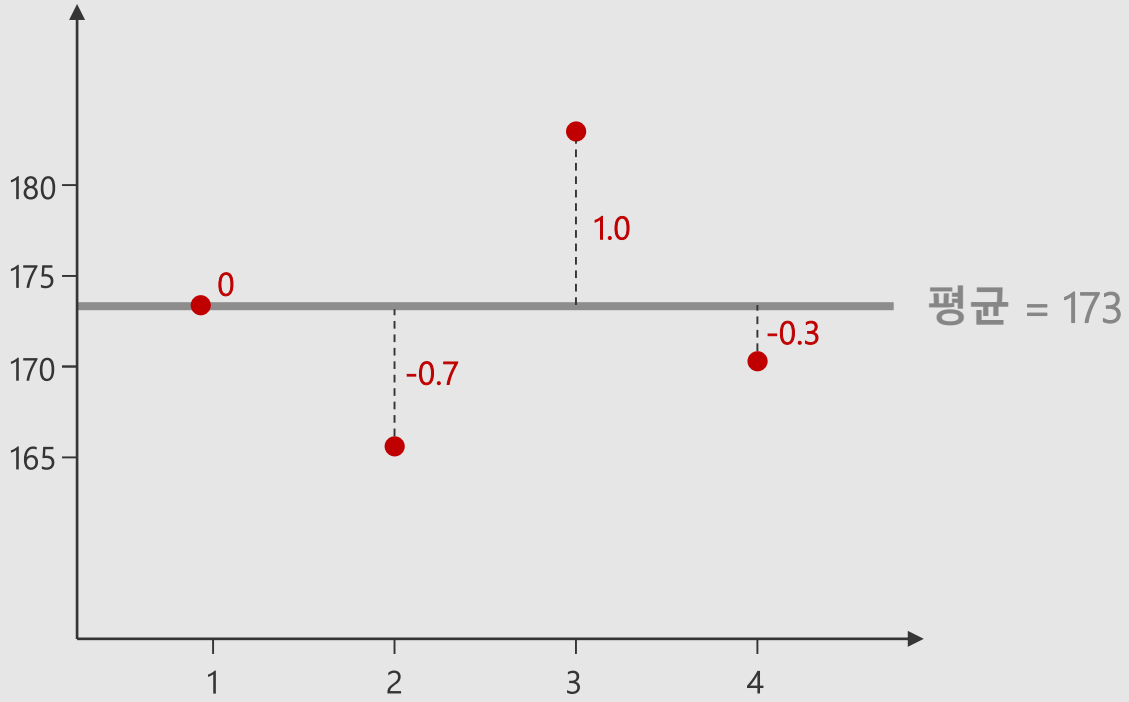
✓ Modeling
- 데이터로부터 모델을 만드는 절차 ➔ 최적화된 모델을 만드는 과정
- 최적화 : 오차가 (**적절히**, **적당히**) 최소화

오차

✓평균

- 통계학에서 사용되는 가장 단순한 모형중 하나가 평균!
- 관측값과 모형의 차이 : 이탈도(deviance)

번호	키
1	173
2	166
3	183
4	170



오차

✓통계 모델의 모든 것은 다음의 식으로 정리됨.

- **실제값 = 모형 + 오차**
- 자료가 벗어난 정도(Sum of Square Error) = $\sum(\text{실제값} - \text{모형})^2$

✓모델링의 목표

- 어떻게 하면 이 오차를 최소화 할 것인가?
- 오차를 최소화 = 모형이 실제를 설명하는(캐치하는) 영역 최대화

Regression 모델링 코드를 익힙시다.

- ✓ 일단 코드 작성법부터 익힙니다.
  - 연습을 통해 코드 작성법을 익힙니다.
  - 절차는 외우고, 세부 코드는 참조하여 작성하도록 합시다.
- ✓ 그 다음에 개념을 다룹니다.

모델링 코드 구조

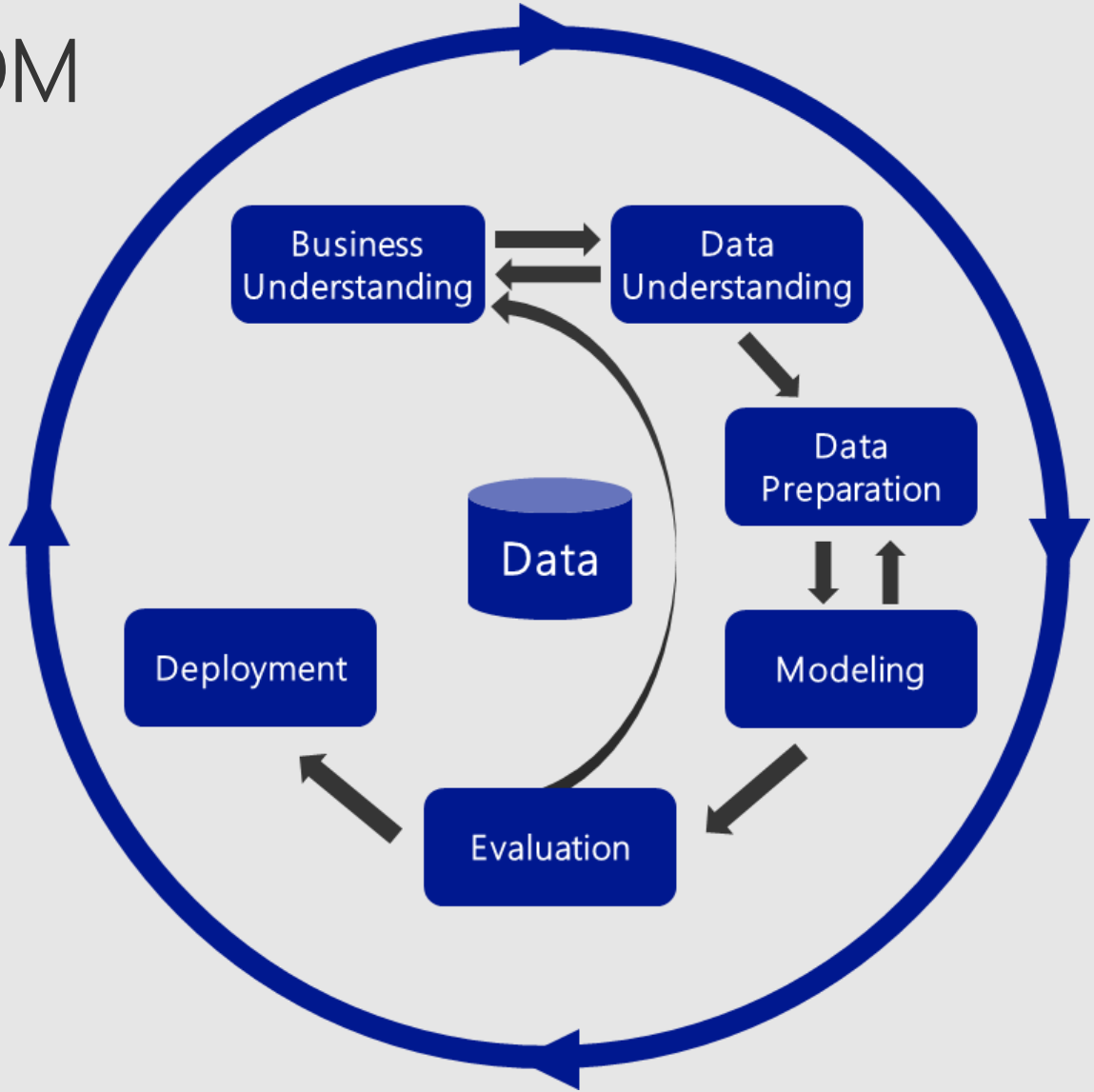
00 환경준비	10 데이터이해	20 데이터준비	30 모델링
		모델링을 위한 데이터 구조 준비하기	모델링

모델링 코드 구조

00 환경준비	10 데이터이해	20 데이터준비	30 모델링
01. import	11. 둘러보기	21. 변수 정리	31. import
02. Data loading	12. 기초통계량	22. NA처리	32. 모델선언
	13. 탐색하기	23. Feature Engineering	33. 모델링(학습)
		24. Dummy variable	34. 예측
		25. Data Split	35. 평가
		26. Scaling	
		27. DataFrame to Numpy	

[ML Process]

CRISP-DM


Business Understanding
Data Understanding
Data Preparation
Modeling
Evaluation
Deployment
Data

①Business Understanding

✓개요

- 잘 정의된 명확한 데이터분석 문제로 시작하는 프로젝트는 거의 없음.
- 문제를 파악해 가는 과정을 반복하면서 문제를 재정의하고 해결책을 정의하게 됨.

✓수행되는 내용

- 비즈니스 목표(문제) 검토
- 데이터 분석 목표 수립
- (초기)가설 수립

①Business Understanding

✓비즈니스 목표에서 데이터 분석 목표로…

비즈니스 관점	**목표**	올해 은행 대출 부서의 수익 1000억 달성 (작년 수익액 600억)
	방법	✓ 신용도 높은 사람의 대출 신청 승인 ✓ 신용도 낮은 사람의 대출 신청 거절
데이터 분석 관점	**문제정의**	대출 신청자들의 신용도를 예측할 수 있을까?
	목표	어느 정도 정확도로 예측할 수 있다면, ▪ 비즈니스 목표를 달성 할 수 있을까? ▪ 2년 이내 프로젝트 투자에 대한 BEP에 도달할 수 있을까?
	분석	▪ **분류문제** ▪ **신용도**에 영향을 미치는 **요인**은 무엇일까?

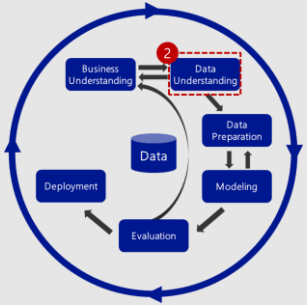
①Business Understanding

✓(초기)가설 수립

신용도에 영향을 미치는 **요인**은 무엇일까?

- 다양한 직무에 있는 사람들의 의견을 수렴할 필요가 있음.
- 데이터의 존재여부를 고려하지 말고 가설 도출.
- 초기 가설 수립 이후 데이터 탐색을 통해 가설을 구체화

②Data Understanding



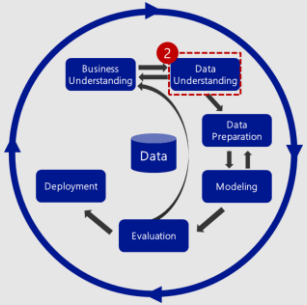
✓ 개요

- 데이터 : 문제의 해결책을 만드는 데 사용할 원자재
- **문제에 정확히 부합하는 데이터가 있는 경우는 거의 없음.**
- 데이터에 따라 데이터 취득 및 유지 비용이 다름.

✓ 수행되는 내용

- 데이터 원본 식별 및 취득
- 데이터 탐색 : EDA, CDA

②Data Understanding



✓데이터 원본 식별 및 취득

- (초기)가설에서 도출된 데이터의 원본을 확인

있는 데이터

없는 데이터

가용한 데이터

그대로 사용가능

가공해야 사용가능

취득 가능

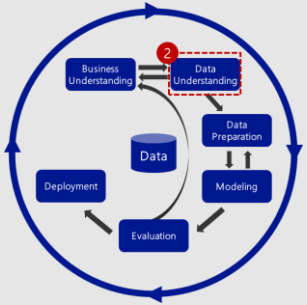
취득 불가능

Data Pipeline
Data Environment
검토 및 과제 수행

취득 비용 산정
과제 수행

정보의 의미 정의
→ 정보 분할
→ 최대한 가용한 데이터 영역으로 만들기

②Data Understanding



✓ 데이터 탐색 : EDA, CDA

- 데이터를 탐색하는 두 가지 방법

데이터 통계량	데이터 시각화
분할표(Contingency Table) MIN, MAX, SUM, MEAN Quartile …	Histogram, Box plot, Density plot Bar plot, Pie chart Scatter plot …

- EDA (Exploratory Data Analysis)
  - 개별 데이터의 분포, 가설이 맞는지 파악
  - NA, 이상치 파악
- CDA (Confirmatory Data Analysis)
  - 탐색으로 파악하기 애매한 정보는 통계적 분석 도구(가설 검정) 사용

③ Data Preparation

✓개요
- 데이터 분석을 위해 특정 조건에 맞는 데이터 유형과 구조가 있음
- 더 좋은 결과를 얻을 수 있도록 데이터의 형태를 조작하고 변환하는 과정 필요.

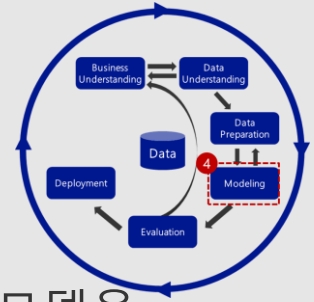
✓수행되는 내용
- 데이터 정제
- 추가 변수(Feature Engineering)

✓결과물 : **하나의 잘 정리/정제된** Data Frame(Table, Matrix)

③ Data Preparation

- ✓ 데이터 정제
  - 잘못된 데이터 정제
  - 결측치(NA) 식별 및 조치
    - 중요한 요인에 결측치가 존재한다면 반드시 조치해야 한다.
    - 예 : 옷을 추천하는데, 고객의 나이나 성별에 결측치가 존재한다면, 옷을 추천하기 곤란.
  - 이상치 식별 및 조치
    - 잘못된 값
    - 값 자체는 정상이나 다른 값들의 분포에 비해 치우친 값
    - 이러한 값은 데이터 분석 시 잘못된 결과를 얻게 하는 원인이 됩니다.

④ Modeling



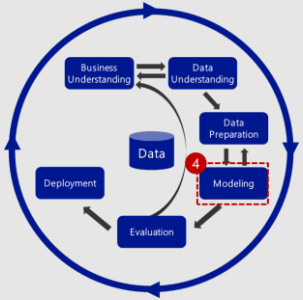
✓개요

- 중요 변수들을 선택하고, 적절한 알고리즘을 적용하여 예측 모델을 생성
- 생성된 모델을 평가

✓수행되는 내용

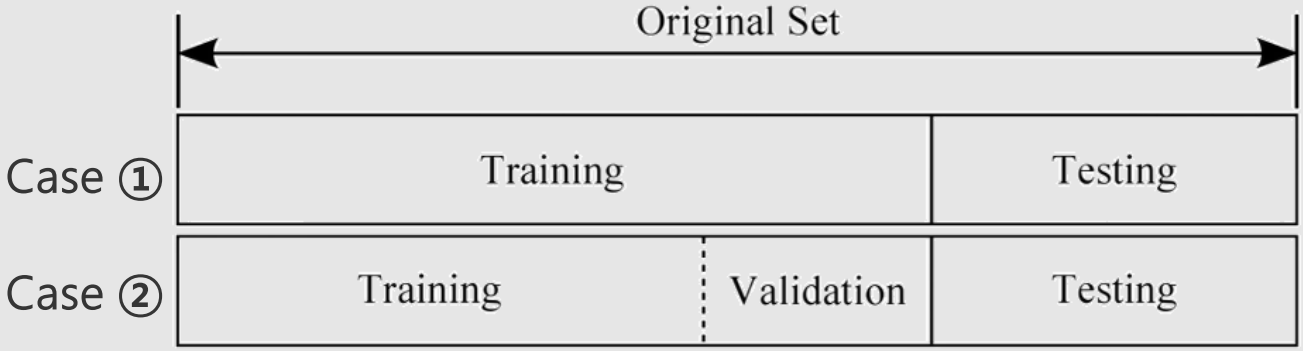
- 데이터셋 분리
- 중요 변수 선정
- 머신러닝 알고리즘 적용하여 모델 생성
- 모델 테스트

④ Modeling

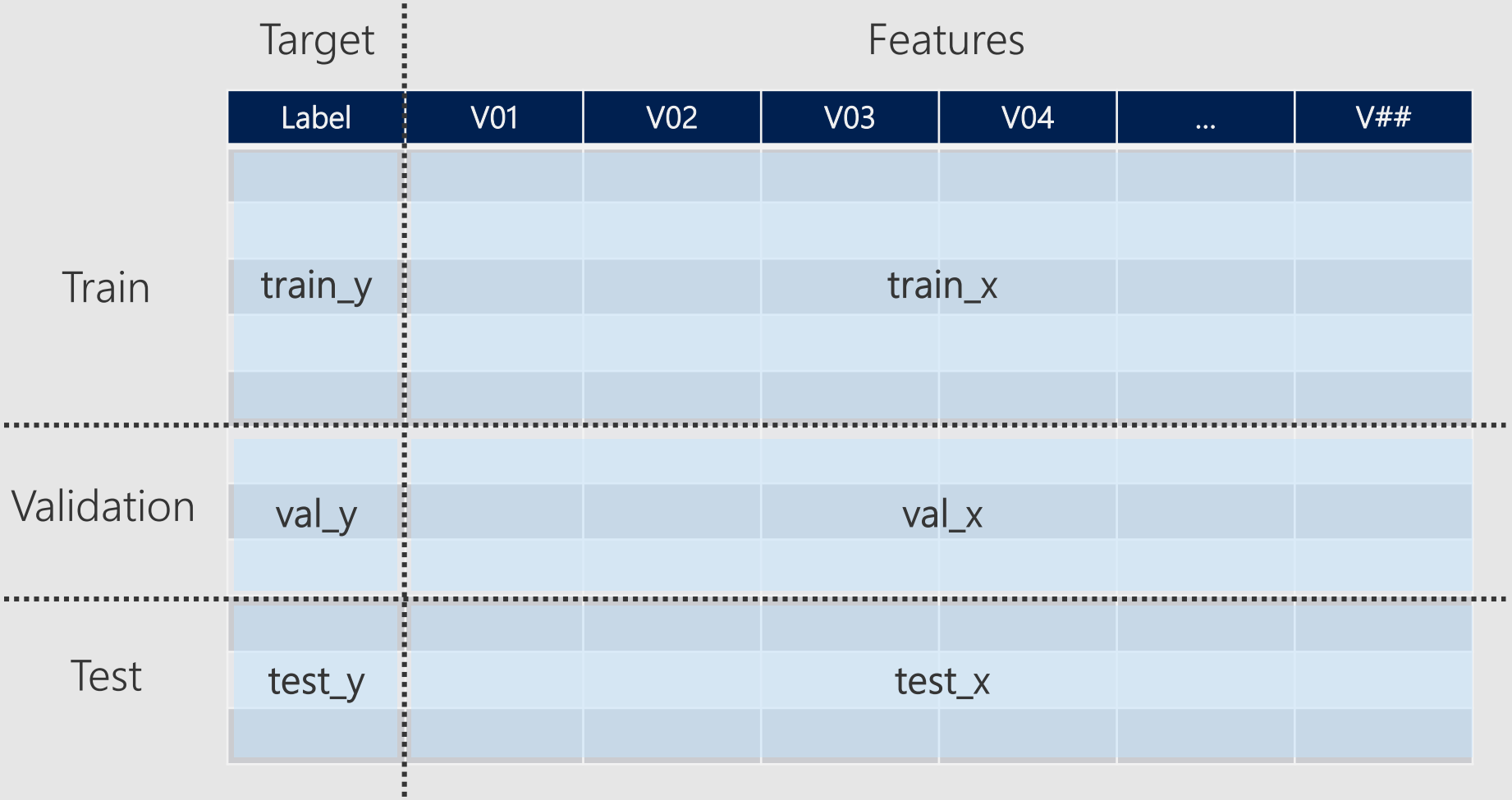


✓데이터 셋 분리

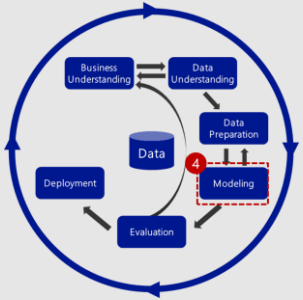
- Case ① : 학습할 때
  - Train Set : 알고리즘을 이용해서 모델을 생성
  - Test Set : 모델 성능 검증
- Case ② : 실전에서 주로 사용
  - Validation Set : 모델 성능 검증
  - Test Set : 모델 최종 평가



데이터셋 구조

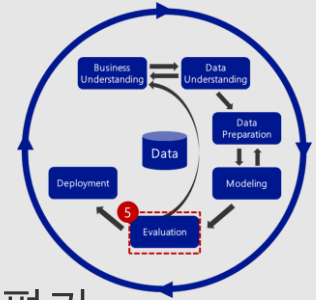

Target
Features
Label
V01
V02
V03
V04
...
V##
Train
train_y
train_x
Validation
val_y
val_x
Test
test_y
test_x

④ Modeling



✓모델 생성

⑤ Evaluation



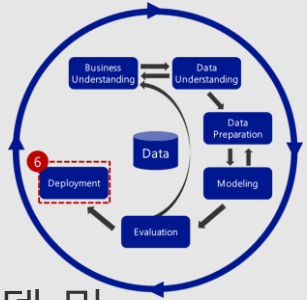
✓개요

- 모델에 대한 데이터 분석 목표와 비즈니스 목표달성에 대한 평가
- 모델과 데이터에서 추출한 패턴이 진정한 규칙성을 갖고 있는지, 단지 특정 예제 데이터에서만 볼 수 있는 특이한 성질은 아닌지 확인
- 비즈니스 목표에 부합되는지 보장

✓수행되는 내용

- 모델에 대한 최종평가 : Test Set 이용
- 비즈니스 기대가치 평가

⑥ Deployment



✓개요

- 프로젝트 결과물 최종 확정: 프로덕션 환경의 파이프라인, 모델 및 배포가 고객 목표를 충족하는지 확인
- 운영시스템에서 품질(성능 목표) 유지 기준을 정하고, 모니터링 계획을 수립

✓수행되는 내용

- 시스템 유효성 검사: 배포된 모델과 이 고객 요구 사항을 충족 하는지 확인
- 프로젝트 이전 : 운영환경으로 배포

Regression

순서

- ✓ 두가지 문제 유형
- ✓ Modeling 이란…
- ✓ 무작정 따라하기(코드 익히기)
- ✓ 단순회귀 : Cars
  - 알고리즘 : Linear Regression
  - 회귀 모델 평가
- ✓ 다중회귀 : Air Quality
  - 추가 알고리즘 : KNN

[단순회귀]

회귀분석

✓ **실제값 = 모형 + 오차**

✓ 단순회귀(Simple Regression)
- 하나의 예측변수로 하나의 결과변수를 예측

✓ 다중회귀(Multiple Regression)
- 복수의 예측변수로 하나의 결과변수를 예측

Linear Regression

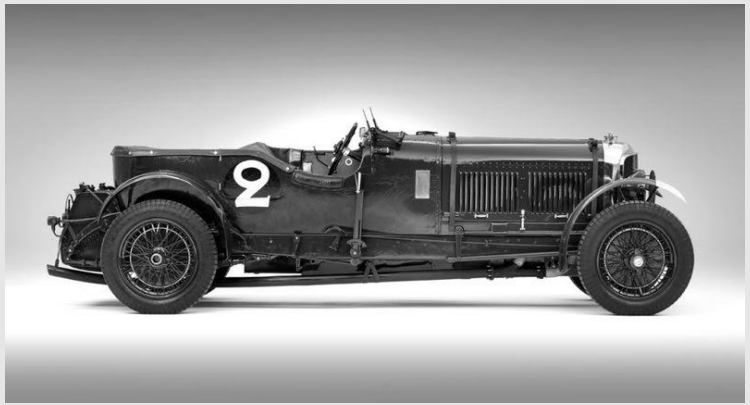
- ✓ 데이터를 하나의 직선으로 요약
- ✓ 자료를 설명하는 직선은 여러 개가 될 수 있음
- ✓ 그 중 가장 잘 설명하는 직선 한 개를 선정하는 방법 : 최소제곱법(Least Square)
- ✓ 실제값 = $(b_0 + b_i X_i)$ + 오차

Regression #1 : Cars

✓ Speed and Stopping Distances of Cars

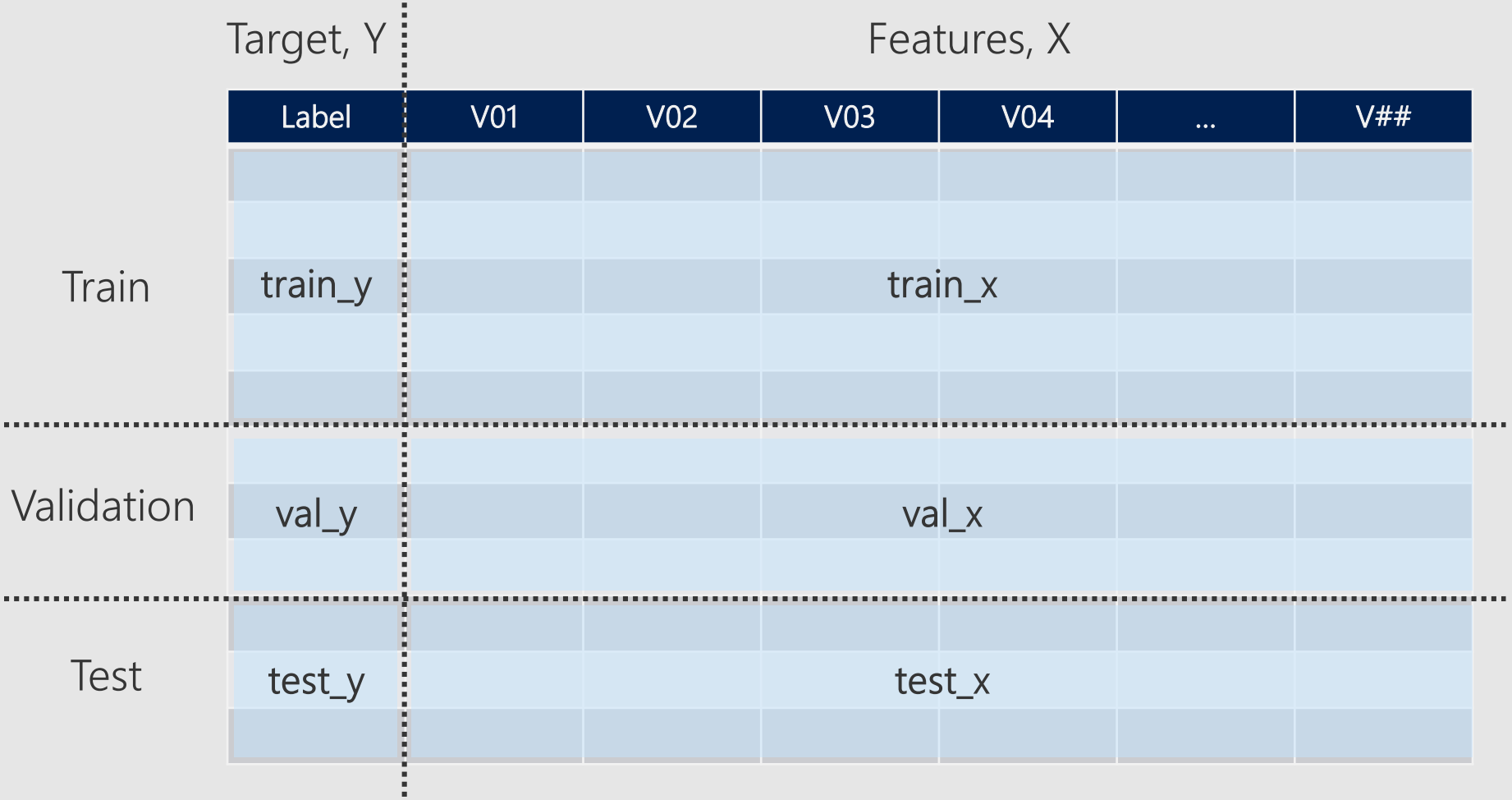
- The data give the speed of cars and the distances taken to stop. Note that the data were recorded in the 1920s.
- Format
  - A data frame with 50 observations on 2 variables.
  - speed numeric Speed (mph)
  - dist numeric Stopping distance (ft)

오스틴 세븐, 1922년



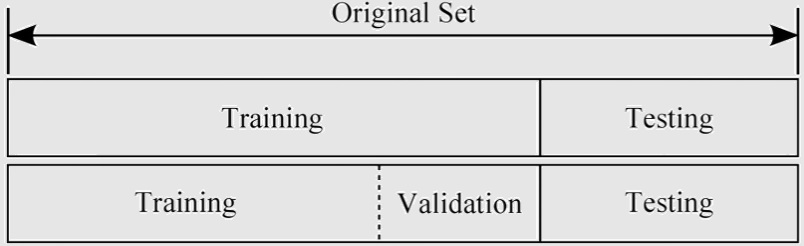
벤틀리 스피드 식스, 1928년

25. Data Split



25. Data Split

✓ 보통 train : test = 8 : 2, 7 : 3으로 분할



- Case ① 공부할 때만 쓰는 방식
- Case ② Validation : 검증, Test : 최종 테스트

필요한 함수 불러오기

```
[ ]  1 # sklearn에서 제공하는 split 함수를 이용해보자. : train_test_split
     2 from sklearn.model_selection import train_test_split
```

x와 y로 나누기

```
[ ]  1 # features와 target 분리
     2 # features
     3 X = data.iloc[:, :4]
     4
     5 # target
     6 y = data.iloc[:, 5]
```

train과 test로 나누기

```
[ ]  1 # train : test = 8 : 2
     2 train_features, test_features, train_target, test_target = \
     3     train_test_split(X, y, test_size=0.2, random_state=1)
     4
```

27. DataFrame to Numpy.Array

✓ Scikit-learn 만 사용한다면 불필요할 수 있지만,
- 많은 Python User들이 일반적으로 모델링하기 전에 numpy로 변환하여 사용
- 차후 딥러닝에서는 numpy를 사용

✓ 그러므로, 우리도 넘파이 어레이로 변환해서 모델링

```
# 판다스 데이터프레임.values ==> 넘파이 어레이가 됨.
# 주의!!!. features는 2차원 array로 만들어야 함, target은 상관 없음.

test_features, test_target = test_data[['speed']].values, test_data['dist'].values

train_features, train_target = train_data[['speed']].values, train_data['dist'].values
```

31. Import, 32. **모델선언**, 33. **모델링(학습)**

✓ 필요한 함수 불러오기

✓ 모델 선언
- 모델에 필요한 하이퍼 파라미터도 여기서 설정하게 됩니다.

✓ 학습
- .fit
- feature들과 target과의 관계, 패턴을 선형회귀 알고리즘을 이용하여 모델(함수)로 만듭니다.

▾ 31. import

```
[ ]  1 # multivariate regression을 해보자!
     2 from sklearn.linear_model import LinearRegression
     3 from sklearn.metrics import mean_squared_error, r2_score
```

▾ 32. 모델선언

```
[ ]  1 # 모델 선언
     2 multi_regression = LinearRegression()
```

▾ 33. 모델링(학습)

```
[ ]  1 multi_regression.fit(train_features, train_target )
```

31. Import, 32. **모델선언**, 33. **모델링(학습)**

✓모델 열어보기 :
- 모델.coef_ : 회귀계수
- 모델.intercept_ : Y절편

```
1 # 회귀계수 살펴보기. 모델을 열어 보자
2 a = simple_regression.coef_
3 b = simple_regression.intercept_
4
5 print(type(a))
6 print(a, b)
```

```
<class 'numpy.ndarray'>
[4.10813984] -19.75648317244979
```

✓이를 직선식으로 적어보면

$$y = 4.108 \cdot x - 19.756$$

34. 예측, 35. 평가

✓ 예측
- 학습시킨 모델이 얼마나 정확한지 검증하기 위해
- 학습할 때 사용하지 않은 데이터셋

```
# 예측값을 뽑자.
test_pred = multi_regression.predict(test_features)
```

✓ 평가
- 회귀 모델은 평균오차의 양과 율로 평가합니다

```
# Training & Validation set에서의 성능 확인

print("MSE : {:.5f}".format(mean_squared_error(test_target, test_pred)))
print("R-squared Score : {:.5f}".format(r2_score(test_target, test_pred)))
```

몇 가지 기호들

y : 실제값

$\hat{y}$: 예측값

$\bar{y}$: 평균값

회귀분석 모델을 평가하기

y	$\hat{y}$	$y-\hat{y}$	$(y-\hat{y})^2$	$\lvert y-\hat{y}\rvert$
6	4	2	4	2
5	6	-1	1	1
12	9	3	9	3
2	2	0	0	2

$$\textbf{Sum Squared Error} = \sum (y-\hat{y})^2 = 14$$

$$\textbf{Mean SSE} = \frac{\sum (y-\hat{y})^2}{n} = \frac{14}{4}$$

$$\textbf{Root MSE} = \sqrt{\frac{\sum (y-\hat{y})^2}{n}} = \sqrt{\frac{14}{4}}$$

$$\textbf{Mean Absolute Err} = \frac{\sum |y-\hat{y}|}{n} = \frac{8}{4}$$

$$\textbf{Mean Absolute Percentage Err} = \frac{\sum \left|\frac{y-\hat{y}}{y}\right|}{n} =$$

알고리즘 사용할 때 알아야 할 4가지

① 알고리즘의 원리, 개념

② 전제조건

③ 문법

④ 성능 : hyper parameter, 복잡도 결정 요인

선형회귀 분석

① 알고리즘의 원리, 개념

➔ 선형회귀식(직선,평면...)으로 Target과의 관계를 설명

② 전제조건

➔ NA처리, feature들은 정규성 가정, 독립성 가정을 충족해야 함.

③ 문법

➔ sklearn이 통일시켜 줌!

④ 성능 : hyper parameter, 복잡도 결정 요인

➔ 어떤 변수를 포함할 것인가? 변수 선택이 중요.

➔ 변수가 많을 수록 복잡해짐.

다중회귀

Regression #2 : Airquality

- ✓ New York Air Quality Measurements
  - Daily air quality measurements in New York, May to September 1973.
  - Format
    - A data frame with 153 observations on 6 variables.
    - Ozone numeric Ozone (ppb)
    - Solar.R numeric Solar R (lang)
    - Wind numeric Wind (mph)
    - Temp numeric Temperature (degrees F)
    - Month numeric Month (1--12)
    - Day numeric Day of month (1--31)

1950년대 뉴욕의 대기오염

다중회귀 모델

✓ Feature가 2개 이상

$$y = w_0 + w_1 \cdot x_1 + w_2 \cdot x_2 + \ldots$$

✓ Feature들 간에 독립성을 가정하고 모델 생성

- x_1 과 x_2 는 서로 독립이다. ==> 둘은 아무 상관 없다.

	Ozone	Solar.R	Wind	Temp	Month	Day	Date
148	30	193.0	6.9	70	9	26	1973-09-26
149	23	145.0	13.2	77	9	27	1973-09-27
150	14	191.0	14.3	75	9	28	1973-09-28
151	18	131.0	8.0	76	9	29	1973-09-29
152	20	223.0	11.5	68	9	30	1973-09-30

다중회귀 모델

✓선형회귀식을 적어보면

```
1 # 회귀계수 살펴보기. 모델을 열어 보자
2 print(X.columns.tolist())
3 print(multi_regression.coef_)
4 print(multi_regression.intercept_)
```

```
['Solar.R', 'Wind', 'Temp', 'Month', 'Day']
[ 0.05576154 -3.29258663  1.75401479 -1.83529503  0.3152641 ]
-65.59789060544094
```

$$Ozone = -65.60 + 0.06 \cdot Solar.R - 3.29 \cdot Wind + 1.75 \cdot Temp - 1.84 \cdot Month + 0.32 \cdot Day$$

모델링 코드 구조(반복, 주입!)

00 환경준비	10 데이터이해	20 데이터준비	30 모델링
01. import	11. 둘러보기	21. 변수 정리	31. import
02. read_csv	12. 기초통계량	22. NA처리	32. 모델선언
	13. 탐색하기	23. Feature Engineering	33. 모델링(학습)
		24. Dummy variable	34. 예측
		25. Data Split	35. 평가
		26. Scaling	
		27. DataFrame to Numpy	

[1] 다중회귀 모델링

22. NA 처리

- ✓NA처리 하는 세 가지 방법
  - ① 제거한다.
  - ② 채운다
  - ③ 분리한다.

22. NA 처리

① 제거한다.
- 1) 열을 제거
- 2) 행을 제거

1) NA가 있는 열 제거

2) NA가 있는 행 제거

y	x01	x02	x03	x04
1	@@@	2	.45	AAA
1	^^^	34	.01	BBB
0	###	3	.9	XXX
1	&&&	6	0	BBB
0	%%%	23	<NA>	AAA
0	***	53	.87	AAA
1	!!!	2	.75	AAA
0	<NA>	32	.21	XXX

22. NA 처리

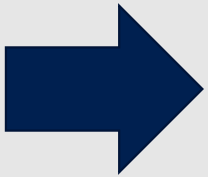
② 채운다

- 앞 뒤 값으로 채우기

```
1 # 시계열 데이터일 경우 쉬운 결정 : 이전 데이터로 채우기
2 data = data.fillna(method = 'ffill')
3 data.isnull().sum()
```

```
1 data.head(8)
```

	Ozone	Solar.R	Wind	Temp	Month	Day
0	41	190.0	7.4	67	5	1
1	36	118.0	8.0	72	5	2
2	12	149.0	12.6	74	5	3
3	18	313.0	11.5	62	5	4
4	19	NaN	14.3	56	5	5
5	28	NaN	14.9	66	5	6
6	23	299.0	8.6	65	5	7
7	19	99.0	13.8	59	5	8



	Ozone	Solar.R	Wind	Temp	Month	Day
0	41	190.0	7.4	67	5	1
1	36	118.0	8.0	72	5	2
2	12	149.0	12.6	74	5	3
3	18	313.0	11.5	62	5	4
4	19	313.0	14.3	56	5	5
5	28	313.0	14.9	66	5	6
6	23	299.0	8.6	65	5	7
7	19	99.0	13.8	59	5	8

24. Dummy Variable(One-hot Encoding)

✓ 범주형 입력변수는 **가변수화** 하여 사용해야 한다. (0, 1로)

- 성별

$$male = \begin{cases} 1, & if\ x = male \\ 0, & otherwise \end{cases}$$

- 계절은?

계절
봄
여름
봄
가을
겨울

봄	여름	가을	겨울
1	0	0	0
0	1	0	0
1	0	0	0
0	0	1	0
0	0	0	1

24. Dummy Variable(One-hot Encoding)

✓ pd.get_dummies(**범주형변수**, prefix = 'W', drop_first = 1)

- prefix : 가변수화 이후 변수 이름 앞 접두어
- drop_first : 모든 범주를 가변수화 하면, 하나의 변수는 나머지 변수들에 의해 결정된다.(완전! 종속된다.) 그래서 하나는 빼 줌!

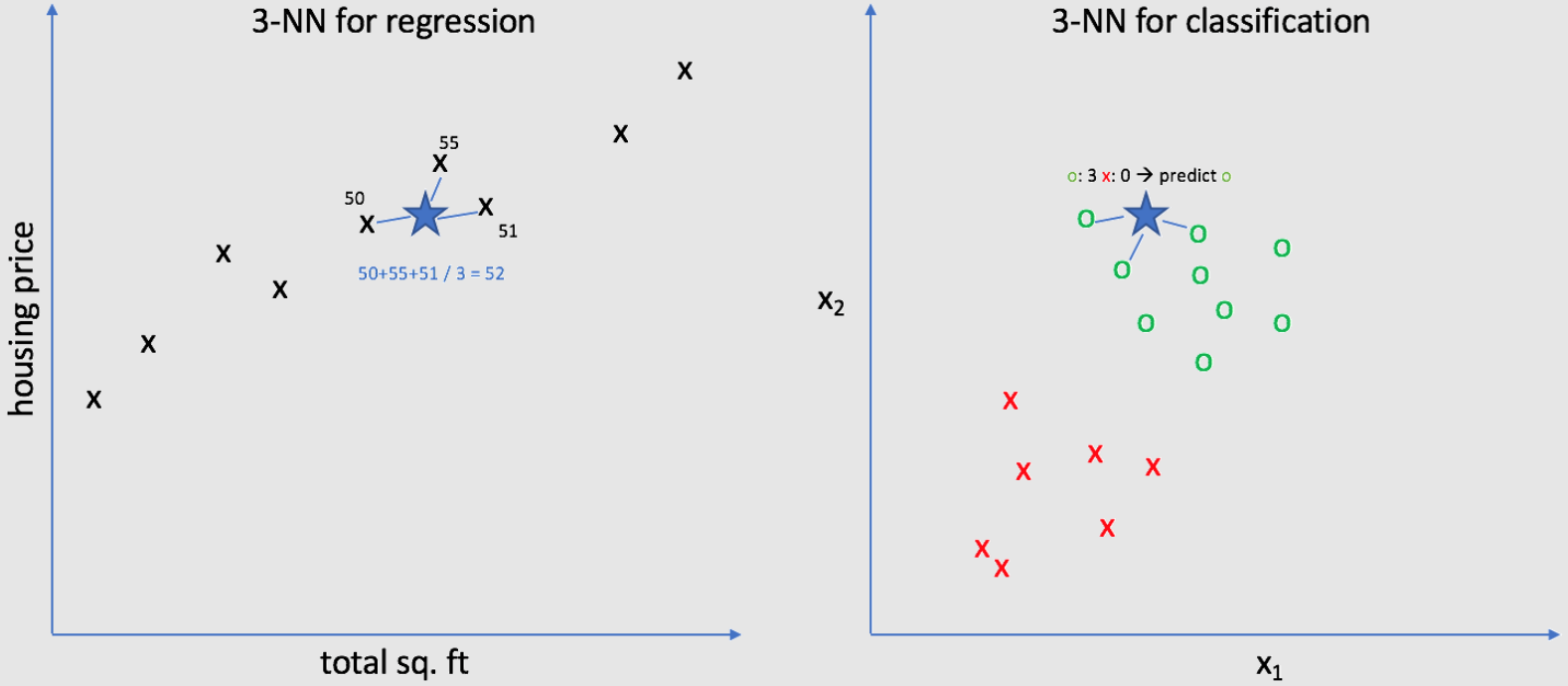
```
1 # dummy variable
2 data_wd = pd.get_dummies(data['WeekDay'], prefix = 'W', drop_first = 1)
```

```
1 data_wd.head()
```

	W_1	W_2	W_3	W_4	W_5	W_6
0	1	0	0	0	0	0
1	0	1	0	0	0	0
2	0	0	1	0	0	0
3	0	0	0	1	0	0
4	0	0	0	0	1	0

[3] 새로운 알고리즘 : KNN
+ Scaling

KNN (*k*-Nearest Neighbor)



1) 예측해야 할 데이터와 주어진 데이터의 모든 거리를 계산
2) 가까운 거리의 데이터를 K개 만큼 찾은 후
3) K개 값의 평균을 계산하여 예측(k개 값의 class를 보고 예측)

KNN (*k*-Nearest Neighbor)

✓ 장점
- 데이터의 분포 형태와 상관이 없다.
- 설명변수의 개수가 많아도 무리 없이 사용 가능

✓ 단점
- 계산시간이 오래 걸림
- 훈련데이터를 모델에 함께 저장
- 해석하기 어려움.

KNN

① 알고리즘의 원리, 개념

➔ 거리를 계산하여 가까운 k 개 이웃의 label을 확인하고 나서

② 전제조건

➔ NA제거, Scaling, Dummy Variable

③ 문법

➔ sklearn 으로 통일!

④ 성능 : hyper parameter, 복잡도 결정 요인

➔ 어떤 변수를 제외할 것인가? 불필요한 변수 제거

➔ k 값이 작을수록 복잡. 클수록 단순

26. Scaling

- ✓값의 범위를 맞춰 주기 위해서 변수
- ✓방법 1 : Normalization
  - 입력변수 X가 [a, b] 범위라면(a=min, b=max)

$$X_{norm} = \frac{x - a}{b - a}$$

- ✓방법2 : Standardization

$$X_z = \frac{x - mean}{std}$$

26. Scaling

	종류	설명
1	StandardScaler	기본 스케일. 평균과 표준편차 사용
2	MinMaxScaler	최대/최소값이 각각 1, 0이 되도록 스케일링
3	MaxAbsScaler	최대절대값과 0이 각각 1, 0이 되도록 스케일링
4	RobustScaler	중앙값(median)과 IQR(interquartile range) 사용. 아웃라이어의 영향을 최소화

```
# 필요한 함수 로딩
from sklearn.preprocessing import MinMaxScaler
```

```
# 함수 선언
scaler = MinMaxScaler()

# 함수 만들기
scaler.fit(train_x)
```

```
# 함수 적용하기
train_x = scaler.transform(train_x)
test_x = scaler.transform(test_x)
```