

Chapter 02 데이터 마이닝 프로세스 개요

목차

2.1 서론

2.2 데이터 마이닝의 핵심 아이디어

2.3 데이터 마이닝 수행 단계

2.4 데이터 분석 사전 단계

2.5 예측력과 과적합

2.6 모델 구축: 선형 회귀 분석을 이용한 예제

2.7 로컬 컴퓨터에서 파이썬을 이용한 데이터 마이닝

2.8 데이터 마이닝 과정의 자동화

2.9 데이터 마이닝의 윤리 이슈

데이터 마이닝 프로세스

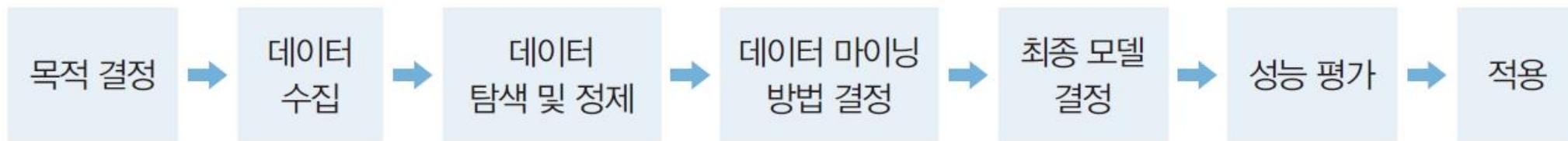


그림 2-1 데이터 마이닝 프로세스

데이터 마이닝 핵심 아이디어

- 분류
- 예측
- 연관 규칙과 추천 시스템
- 예측 애널리틱스
- 데이터 축소와 차원 축소
- 데이터 탐색과 시각화
- 지도 학습과 비지도 학습

지도 학습과 비지도 학습

■ 지도 학습

- 단일 "대상" 또는 "결과" 변수 예측
- 모델링에 사용되는 데이터를 학습 데이터라 부름
- 분류 및 예측

■ 비지도 학습

- 예측하거나 분류할 타겟(결과) 변수가 없이 입력 데이터 내부의 패턴을 스스로 학습
- 데이터 내 연관 규칙을 찾고, 비슷한 관측치끼리 군집하며, 차원을 축소하는 데 사용
- 연관 규칙, 협업 필터, 데이터 축소 및 탐색, 시각화

분류 및 예측

■ 분류

- 범주형의 변수의 미래 값 예측

- 예

- 제안: 응답/응답하지 않음, 대출: 적시 상환/연체/파산
- 신용카드 거래 : 정상/사기, 네트워크의 패킷 데이터 : 정상/침입
- 정류장 대기 버스 : 정상/고장, 환자 상태: 회복/진행 중/사망

■ 예측

- 숫자로 표현된 연속형 변수의 미래 값 예측

- 예

- 판매, 수익, 성과

연관 규칙과 추천 시스템

■ 연관 규칙

- 상호 연관성이 높은 항목 집합들을 찾아내는 분석 방법
- 슈퍼마켓에서는 제품 간 연관성을 분석하여 주간 프로모션 쿠폰을 발행하거나 연관성이 높은 제품을 같은 위치에 진열
- 병원에서는 진료 데이터베이스로부터 도출한 증상 간 연관성 정보를 이용해 한 증상이 다른 증상을 유발한다는 사실을 밝혀 이 정보를 다른 환자들에게 유용하게 적용

■ 추천 시스템

- 아마존, 넷플릭스는 협업 필터링 기법 사용
- 협업 필터링은 개개인의 포괄적인 과거 구매 정보와 다른 사람들의 구매 정보를 이용해 개개인의 구매 성향을 예측하는 추천 시스템 기법으로, 모든 고객의 일반적인 패턴을 찾는 연관 규칙과는 달리 개개인 맞춤형 패턴을 찾을 수 있음

데이터 축소와 자원 축소

■ 데이터 축소

- 많은 수의 관측치를 적은 수의 그룹으로 요약하는 과정
- 대표적인 방법은 군집 분석

■ 자원 축소

- 변수의 개수를 줄이는 과정
- 지도 학습 전에 예측 성능을 향상하고 해석을 용이하게 하기 위해 자원 축소 진행

데이터 탐색과 시각화

■ 데이터 탐색

- 데이터의 전반적인 패턴과 특이 패턴을 찾는 것
- 데이터를 탐색할 때 차트나 대시보드 등을 이용하는 것을 데이터 시각화 혹은 시각화 애널리틱스

■ 데이터 탐색에서의 데이터 시각화

- 수치 변수의 경우 히스토그램이나 박스 플롯을 이용해 데이터의 분포를 파악하고, 이상치(극단치) 탐지
- 범주형 변수의 경우에는 막대그래프 이용
- 산점도를 이용해 변수들 간 상관관계를 파악하고 이상치를 발견 가능

데이터 마이닝 수행 단계

- ① 데이터 마이닝 프로젝트의 목적 정확히 설정하기
- ② 분석에 필요한 데이터셋 획득하기
- ③ 데이터의 탐색/정제/전처리하기
- ④ 필요시 데이터 축소하기
- ⑤ 데이터 마이닝 문제 결정하기(분류, 예측, 군집 등)
- ⑥ 데이터 분할하기(지도 학습의 경우)
- ⑦ 사용할 데이터 마이닝 기법 선택하기(회귀 분석, 인공 신경망, 계층 군집 등)
- ⑧ 알고리즘을 사용해 과제 수행하기
- ⑨ 알고리즘 결과 해석하기
- ⑩ 모델 적용하기

데이터 마이닝 패러다임(변형)

■ SEMMA 단계(SAS 개발)

- Sample(추출): 데이터셋을 학습/검증/테스트 데이터셋으로 나눈다.
- Explore(탐색): 데이터셋을 통계적 혹은 시각적으로 분석한다.
- Modify(수정): 변수를 변환하고 결측치를 처리한다.
- Model(모델링): 예측 모델을 구축한다(의사결정 트리, 인공 신경망).
- Assess(평가): 검증 데이터셋을 이용해 후보 모델을 비교한다.

■ CRISP-DM(IBM SPSS Modeler)

- 비즈니스 이해
- 데이터 이해
- 데이터 준비
- 모델링
- 평가
- 배포

웨스트 록스베리 지역의 주택 가치 예측

■ 데이터셋(WestRoxbury.csv)

- <https://github.com/reisanar/datasets/blob/master/WestRoxbury.csv>
- 원데이터(Property Assessment FY2014) <https://data.boston.gov/dataset/property-assessment/resource/7190b0a4-30c4-44c5-911d-c34f60b22181>

■ 2장의 파이썬 코드

- https://colab.research.google.com/drive/1TAP-2Vf-uP_uMLUSzkbDnobB_cVsyC7

웨스트 록스베리 지역의 주택 가격 데이터

표 2-1 웨스트 록스베리(보스턴 지역) 주택 가격 데이터 변수 설명

TOTAL VALUE	주택 가격(단위: 1,000달러)
TAX	세금, 주택 가격에 세율을 곱한 값에 근거한 세금 계산서 금액(단위: 달러)
LOT SQFT	총 부지 면적(단위: 제곱피트)
YR BUILT	건축 연도
GROSS AREA	총 바닥 면적
LIVING AREA	주거 공간 총 면적(단위: 제곱피트)
FLOORS	층 수
ROOMS	총 방 수
BEDROOMS	총 침실 수
FULL BATH	총 욕실 개수
HALF BATH	총 보조 욕실 개수
KITCHEN	총 주방 개수
FIREPLACE	총 벽난로 개수
REMODEL	리모델링 시기(최근/오래 전/안 함)


웨스트 록스베리 지역의 주택 가격 데이터

표 2-2 웨스트 록스베리 주택 가격 데이터 처음 10개의 관측치

TOTAL VALUE	TAX	LOT SQ FT	YR BUILT	GROSS AREA	LIVING AREA	FLOORS	ROOMS	BED ROOMS	FULL BATH	HALF BATH	KIT CHEN	FIRE PLACE	REMODEL
344.2	4330	9965	1880	2436	1352	2	6	3	1	1	1	0	None
412.6	5190	6590	1945	3108	1976	2	10	4	2	1	1	0	Recent
330.1	4152	7500	1890	2294	1371	2	8	4	1	1	1	0	None
498.6	6272	13,773	1957	5032	2608	1	9	5	1	1	1	1	None
331.5	4170	5000	1910	2370	1438	2	7	3	2	0	1	0	None
337.4	4244	5142	1950	2124	1060	1	6	3	1	0	1	1	Old
359.4	4521	5000	1954	3220	1916	2	7	3	1	1	1	0	None
320.4	4030	10,000	1950	2208	1200	1	6	3	1	0	1	0	None
333.5	4195	6835	1958	2582	1092	1	5	3	1	0	1	1	Recent
409.4	5150	5093	1900	4818	2992	2	8	4	2	0	1	0	None

- 전체: <https://github.com/reisanar/datasets/blob/master/WestRoxbury.csv>

표 2-3 pandas에서 파일 실행

 데이터로부터 데이터셋을 생성하고 불러오는 파이썬 코드

```
# Import required packages
import pandas as pd

# Load data
housing_df = pd.read_csv('WestRoxbury.csv')
housing_df.shape # find the dimension of data frame
housing_df.head() # show the first five rows
print(housing_df) # show all the data

# Rename columns: replace spaces with '_' to allow dot notation
housing_df = housing_df.rename(columns={'TOTAL VALUE ': 'TOTAL_VALUE'}) # explicit
housing_df.columns = [s.strip().replace(' ', '_') for s in housing_df.columns] # all columns

# Practice showing the first four rows of the data
housing_df.loc[0:3] # loc[a:b] gives rows a to b, inclusive
housing_df.iloc[0:4] # iloc[a:b] gives rows a to b-1

# Different ways of showing the first 10 values in column TOTAL_VALUE
housing_df['TOTAL_VALUE'].iloc[0:10]
housing_df.iloc[0:10]['TOTAL_VALUE']
housing_df.iloc[0:10].TOTAL_VALUE # use dot notation if the column name has no spaces

# Show the fifth row of the first 10 columns
housing_df.iloc[4][0:10]
housing_df.iloc[4, 0:10]
housing_df.iloc[4:5, 0:10] # use a slice to return a data frame

# Use pd.concat to combine non-consecutive columns into a new data frame.
# The axis argument specifies the dimension along which the
# concatenation happens, 0=rows, 1=columns.
pd.concat([housing_df.iloc[4:6,0:2], housing_df.iloc[4:6,4:6]], axis=1)

# To specify a full column, use:
housing_df.iloc[:,0:1]
housing_df.TOTAL_VALUE
housing_df['TOTAL_VALUE'][0:10] # show the first 10 rows of the first column

# Descriptive statistics
print('Number of rows ', len(housing_df['TOTAL_VALUE'])) # show length of first column
print('Mean of TOTAL_VALUE ', housing_df['TOTAL_VALUE'].mean()) # show mean of column
housing_df.describe() # show summary statistics for each column
```

이 장에 필요한 파이썬 라이브러리 불러오기

```
import numpy as np
import pandas as pd
from sklearn.model_selection import train_test_split
from sklearn.metrics import r2_score
from sklearn.linear_model import LinearRegression
```


- 컴퓨터 용량과 소프트웨어의 한계로 인해 적은 데이터로도 모든 데이터를 사용했을 때와 비슷한 효과를 볼 수 있게 적절하게 샘플링하는 작업이 필요

표 2-4 pandas를 이용한 샘플링



언더샘플링과 오버샘플링 코드

```
# random sample of 5 observations
housing_df.sample(5)

# oversample houses with over 10 rooms
weights = [0.9 if rooms > 10 else 0.01 for rooms in housing_df.ROOMS]
housing_df.sample(5, weights=weights)
```

분류 문제에서 희소사건에 대한 오버샘플링

- 관심 데이터가 희귀할 경우(메일에 응답하여 특정 상품을 구매하는 고객, 사기 카드 거래) 데이터 부족으로 모델 구축이 어려움
- 샘플링 시 소수 클래스에 더 큰 가중치를 주어 불균형 해결 또는 각 클래스의 오분류에 가중치를 주어 해결
- 다수 클래스와 소수 클래스의 불균형을 고려하지 않은 모델의 경우 전반적인 정확도는 높을 수 있으나 실제 문제에서 적용하기 어려움
- 모델 구축 시 클래스별 오분류의 중요도를 고려해 가중치를 다르게 부여해서 보다 중요한 클래스의 오분류를 줄이는 것이 중요

데이터 전처리와 데이터 정제

■ 변수 종류

- 변수는 여러 기준으로 분류 가능
 - 숫자 혹은 문자, 연속형 실수나 정수 혹은 범주형
- 범주형의 경우 숫자 혹은 문자로 표현
 - 북아메리카, 유럽, 아시아와 같이 특별히 순위가 없는 경우를 명목형 변수
 - 큰 값, 작은 값 등 순위로 표현할 수 있는 경우를 순서형 변수
- 데이터 마이닝에서는 나이브 베이즈 분류기(Naive Bayes Classifier)와 같이 범주형 변수만 사용하는 특별한 경우를 제외하면 대부분 연속형 변수를 예측 변수로 사용

표 2-5 pandas를 이용한 변수 특성 파악



변수 특성을 파악하기 위한 파이썬 코드

```
housing_df.columns # print a list of variables

# REMODEL needs to be converted to a categorical variable
housing_df.REMODEL = housing_df.REMODEL.astype('category')
housing_df.REMODEL.cat.categories # Show number of categories
housing_df.REMODEL.dtype # Check type of converted variable
```

실행 결과

```
> housing_df.columns
Index(['TOTAL_VALUE', 'TAX', 'LOT_SQFT', 'YR_BUILT', 'GROSS_AREA',
      'LIVING_AREA', 'FLOORS', 'ROOMS', 'BEDROOMS', 'FULL_BATH', 'HALF_BATH',
      'KITCHEN', 'FIREPLACE', 'REMODEL'], dtype='object')

> housing_df.REMODEL.cat.categories
Index(['None', 'Old', 'Recent'], dtype='object')

> housing_df.REMODEL.dtype
category
```

■ 범주형 변수 처리

- 범주형 변수 값에 순위가 있는 경우(연령구간, 신용 등급)에는 연속형 변수로 간주
- 범주에 순위가 없다면 범주를 가변수로 바꿔서 사용하기도 함

표 2-6 pandas를 이용한 가변수 생성



가변수를 생성하기 위한 파이썬 코드

```
# use drop_first=True to drop the first dummy variable
housing_df = pd.get_dummies(housing_df, prefix_sep='_', drop_first=True)
housing_df.columns
housing_df.loc[:, 'REMODEL_0ld': 'REMODEL_Recent'].head(5)
```

실행 결과

	REMODEL_0ld	REMODEL_Recent
0	0	0
1	0	1
2	0	0
3	0	0
4	0	0

데이터 전처리와 데이터 정제

■ 변수 선택

- 신뢰성이 높은 모델을 구축하려면 꼭 필요한 변수만 사용하는 것이 바람직
- 변수를 많이 사용할 경우 변수 간의 상관관계를 고려해야 하고 더 많은 수의 관측치 필요
- 변수가 너무 많은 모델은 미래 값을 예측하는 데 데이터가 더 많이 필요하고 전처리 작업도 증가해 강건하지 않음

■ 필요 변수와 관측치 수

- 적절한 변수를 포함시키는 작업은 모델 구축 시 매우 중요
- 꼭 필요한 변수만 사용해야 간결한 모델 구축 가능하고, 간결한 모델이 결국 훌륭한 모델
- 방대한 데이터에서 모델에 어떤 변수를 포함할지는 매우 중요
- 도메인 지식이 있는 사람에게서 얻는 정보는 변수 포함 여부를 결정할 때 중요하고 모델의 정확도를 높이고 오차를 줄이는 데 도움

데이터 전처리와 데이터 정제

■ 이상치

- 측정 오류나 잘못된 입력으로 인해 이상치 발생 가능
- 기존 데이터로부터 멀리 떨어진, 통상 평균으로부터 표준편차의 세 배가 넘는 범위의 데이터
- 통계 기법으로 이상치의 후보를 찾고 최종 이상치 판정은 실무자들이 하는 것이 바람직
- 변수별 값들을 내림차순으로 정렬하고 이들 중 가장 크거나 작은 수치들을 이상치로 판정, 군집 분석을 이용해 다른 데이터들과 일정 거리 이상 떨어진 데이터를 이상치로 판정
- 이상치의 개수가 매우 적은 경우에는 결측치로 처리 가능

데이터 전처리와 데이터 정제

■ 결측치

- 변수가 많을 경우, 결측치를 단순 삭제하면 결측치가 데이터에서 차지하는 비율이 낮더라도 많은 관측치에 영향을 미칠 수 있음
- 결측치가 있는 관측치를 삭제하기보다 기존 데이터를 이용해 대체할 수 있음
- 간단한 결측치 대체값은 평균, 중앙값 등, 좀 더 정교한 대체값은 회귀 분석을 이용
- 데이터가 충분히 많을 경우, 복잡한 결측치 처리법보다는 간단한 방법 선호
- 결측치가 많을 경우 관측치 수 자체가 적어 대체 방법을 적용하는 데에도 한계
- 이 경우에는 변수의 중요도를 측정하여 해당 변수가 예측에 미치는 영향력이 미미하다면 그 변수를 삭제하고 그렇지 않다면 결측치가 적게 존재하는 변수의 정보를 이용해 결측치를 대체하는 것이 좋음
- 정말 중요한 변수의 결측치를 해결하는 근본적인 방법은 실제값을 얻는 것

표 2-7 결측 데이터 처리

 결측치를 중앙값으로 대체하는 파이썬 코드

```
# To illustrate missing data procedures, we first convert a few entries for
# bedrooms to NA's. Then we impute these missing values using the median of the
# remaining values.
missingRows = housing_df.sample(10).index
housing_df.loc[missingRows, 'BEDROOMS'] = np.nan
print('Number of rows with valid BEDROOMS values after setting to NAN: ',
      housing_df['BEDROOMS'].count())

# remove rows with missing values
reduced_df = housing_df.dropna()
print('Number of rows after removing rows with missing values: ', len(reduced_df))

# replace the missing values using the median of the remaining values.
medianBedrooms = housing_df['BEDROOMS'].median()
housing_df.BEDROOMS = housing_df.BEDROOMS.fillna(value=medianBedrooms)
print('Number of rows with valid BEDROOMS values after filling NA values: ',
      housing_df['BEDROOMS'].count())
```

실행 결과

```
Number of rows with valid BEDROOMS values after setting to NAN: 5782
Number of rows after removing rows with missing values: 5772
Number of rows with valid BEDROOMS values after filling NA values: 5802
```

데이터 전처리와 데이터 정제

■ 데이터 정규화(표준화)와 리스케일링

■ 정규화(표준화)가 필요한 이유

- 군집 분석은 각 관측치가 군집 평균으로부터 얼마만큼 떨어져 있는지를 기준으로 하는데 다변량 데이터의 경우 변수들의 단위가 서로 달라(날짜, 달러, 횡수) 단위가 큰 변수가 전체 거리 계산을 지배할 수 있고, 단위가 "일"인 변수를 "시간"이나 "달"의 단위로 바꾼다면 분석 결과가 완전히 달라지기 때문

■ 정규화 방법

- z-score 값 구하기: 각 관측치에서 해당 변수의 평균값을 빼주고 표준편차 값으로 나누기
- scikit-learn 패키지에서 제공하는 StandardScaler() 함수 이용
- 학습 데이터에는 fit() 혹은 fit_transform()을 사용하고, 검증 데이터에는 transform()을 사용
- 모든 변수를 [0,1]의 스케일로 바꾸어 줌
 - pandas에서는 $(df - df.min()) / (df.max() - df.min())$
 - scikit-learn 패키지에서는 MinMaxScaler를 이용

표 2-8 정규화 및 스케일



정규화 및 스케일링 코드

```
from sklearn.preprocessing import MinMaxScaler, StandardScaler
df = housing_df.copy()

# Normalizing a data frame

# pandas:
norm_df = (housing_df - housing_df.mean()) / housing_df.std()

# scikit-learn:
scaler = StandardScaler()
norm_df = pd.DataFrame(scaler.fit_transform(housing_df),
                       index=housing_df.index, columns=housing_df.columns)

# the result of the transformation is a numpy array, we convert it into a dataframe

# Rescaling a data frame

# pandas:
norm_df = (housing_df - housing_df.min()) / (housing_df.max() - housing_df.min())

# scikit-learn:
scaler = MinMaxScaler()
norm_df = pd.DataFrame(scaler.fit_transform(housing_df),
                       index=housing_df.index, columns=housing_df.columns)
```

예측력과 과적합

■ 과적합

- 모델은 반응 변수와 예측 변수 간의 상관관계를 잘 설명하고 미래 값을 정확히 예측해야 함
- 현재 데이터를 잘 설명하는 것도 중요하지만 더 중요한 것은 미래 값에 대한 정확한 예측
- 현재 데이터에 '과하게 적합'된 것을 '과적합'이라 함
- 현 데이터의 성능만 고려해 불필요하게 포함된 변수들은 과적합을 유발할 수 있음
- 관측치 수가 변수 개수보다 적을 경우 엉뚱한 관계가 모델에 반영될 수 있어 학습 데이터가 충분하지 않을 경우에는 간단한 함수로 표현된 모델이 더 좋은 성능을 보일 때가 많음
- 가장 성능이 좋은 모델을 선택하는 과정에서 너무 많은 모델을 사용할 때도 과적합이 발생할 수 있음

과적합

표 2-9 광고비 판매량

Advertising	Sales
239	514
364	789
602	550
644	1386
770	1394
789	1440
911	1354

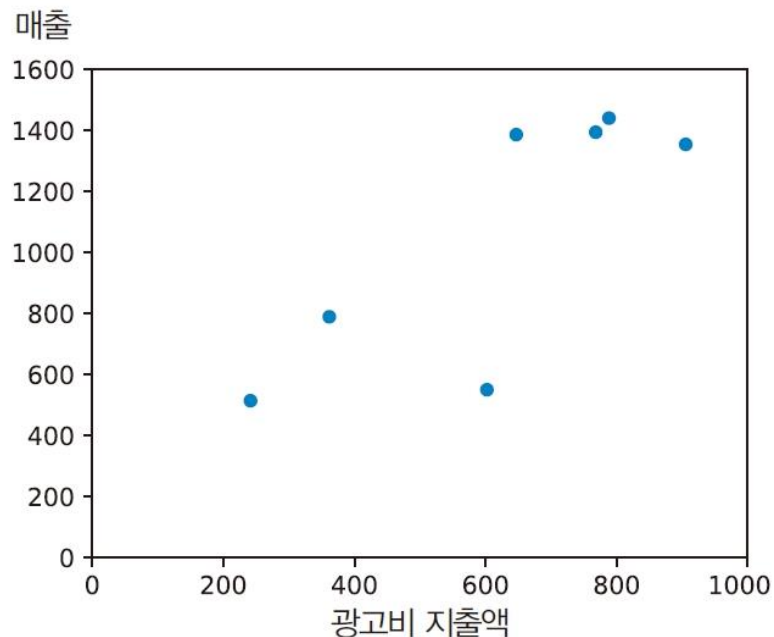


그림 2-2 광고비와 매출액 관계 산점도

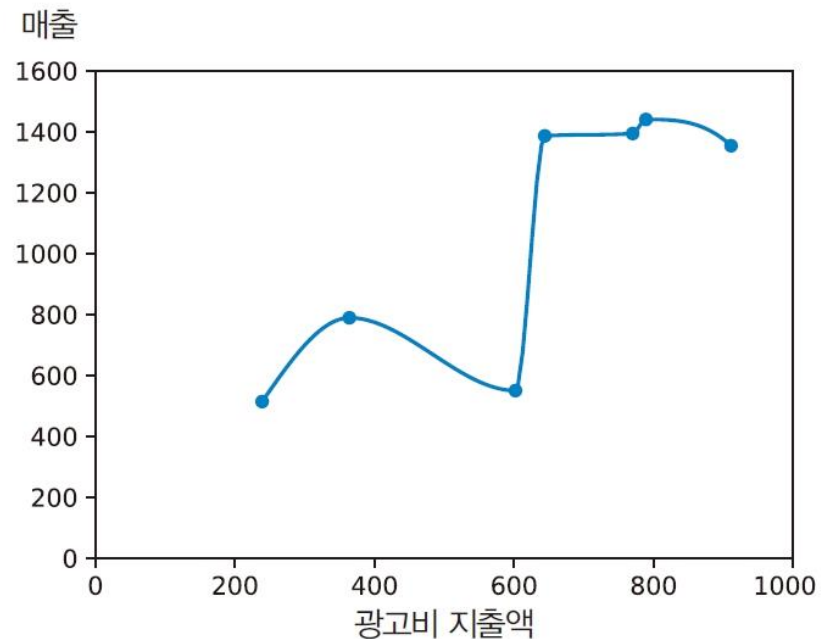


그림 2-3 과적합: 현 데이터에서 오차가 0인 곡선

현 데이터에는 한 치의 오차도 없어 보이지만,
광고비 지출을 기준으로 미래의 데이터(향후 매출액)
를 예측하기에는 무리가 있음

데이터 분할

■ 모델의 성능이 좋은 경우

- 선택한 모델의 우수성
- 선택한 모델이 현재 사용한 데이터에 (우연히) 잘 맞음 → 심각한 문제 초래

■ 데이터를 학습 데이터, 검증 데이터, 테스트 데이터로 나누어 모델 검증

- 학습(훈련, Training) 데이터: 모델 구축 시 사용. 여러 모델을 비교할 경우 동일한 학습 데이터를 사용
- 검증(Validation) 데이터: 학습 데이터로부터 구축된 여러 모델의 성능을 비교할 때 사용. 몇몇 알고리즘(의사결정 트리, k-최근접 이웃)에서는 검증 데이터를 이용해 모델의 하이퍼 파라미터를 결정하기도 함.
- 테스트(평가, 시험, Testing) 데이터: 구축된 모델이 향후 수집될 새로운 데이터에 대해 얼마만큼 예측/분류 성능을 보일지 평가할 때 쓰임.

데이터 분할

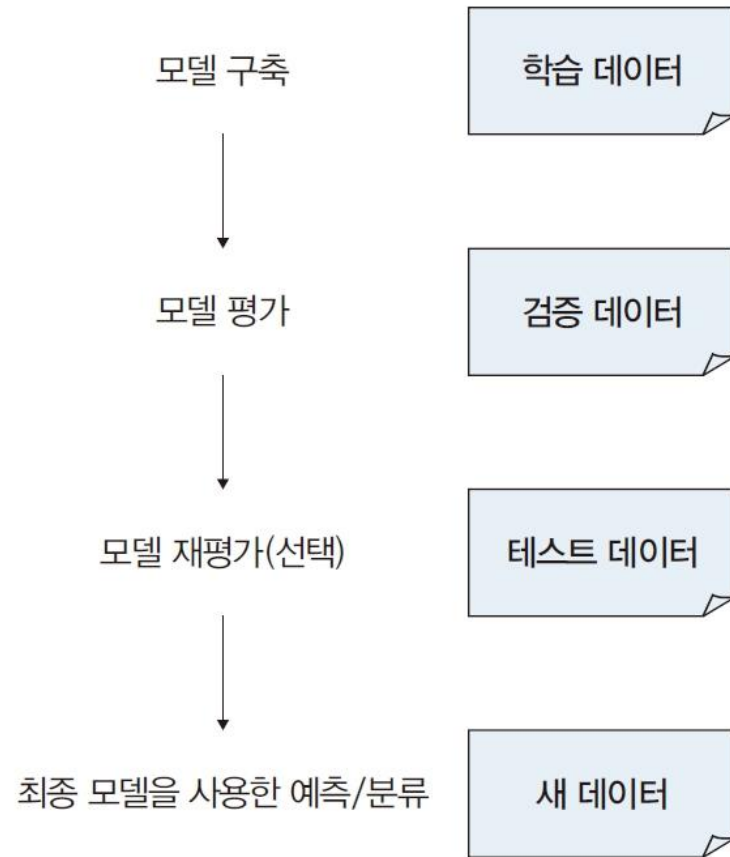


그림 2-4 데이터 마이닝 프로세스에서 학습/검증/테스트 데이터의 역할

표 2-10 데이터 분할



웨스트 록스베리 데이터를 학습 데이터, 검증 데이터, 테스트 데이터로 분할하는 파이썬 코드

```
# random_state is set to a defined value to get the same partitions when re-running
# the code
# training (60%) and validation (40%)
trainData, validData = train_test_split(housing_df, test_size=0.40, random_state=1)

print('Training : ', trainData.shape)
print('Validation : ', validData.shape)
print()

# training (50%), validation (30%), and test (20%)
trainData, temp = train_test_split(housing_df, test_size=0.5, random_state=1)
validData, testData = train_test_split(temp, test_size=0.4, random_state=1)

print('Training   : ', trainData.shape)
print('Validation : ', validData.shape)
print('Test       : ', testData.shape)
```

실행 결과

```
Training   : (3481, 15)
Validation : (2321, 15)

Training   : (2901, 15)
Validation : (1741, 15)
Test       : (1160, 15)
```


모델 구축: 선형 회귀 분석을 이용한 예제(교재 61~65p)

■ 모델링 과정

- 1 목적 웨스트 록스베리 지역 주택 가격을 예측한다.
- 2 데이터셋 획득 2014년 웨스트 록스베리 주택 가격 데이터셋을 이용할 것이며, 전체 데이터의 수가 많지 않아 별도의 표본을 취하지 않았다.

3 데이터 탐색 /정제 /전처리

표 2-11 웨스트 록스베리 주택 가격 데이터의 이상치

FLOORS	ROOMS
15	8
2	10
1.5	6
1	6

- 4 데이터 축소
- 5 데이터 마이닝 문제 결정
- 6 데이터 분할(지도 학습용)
- 7 데이터 마이닝 기법 선택

8 알고리즘 사용해 과제 수행

표 2-12 학습 데이터 일부에 대한 예측(혹은 적합)



웨스트 록스베리 학습 데이터를 사용해 모델을 구축하는 파이썬 코드

```
from sklearn.linear_model import LinearRegression

# data loading and preprocessing
housing_df = pd.read_csv('WestRoxbury.csv')
housing_df.columns = [s.strip().replace(' ', '_') for s in housing_df.columns]
housing_df = pd.get_dummies(housing_df, prefix_sep='_', drop_first=True)

# create list of predictors and outcome
excludeColumns = ('TOTAL_VALUE', 'TAX')
predictors = [s for s in housing_df.columns if s not in excludeColumns]
outcome = 'TOTAL_VALUE'
```

```
# partition data
X = housing_df[predictors]
y = housing_df[outcome]
train_X, valid_X, train_y, valid_y = train_test_split(X, y, test_size=0.4,
                                                    random_state=1)
```

```
model = LinearRegression()
model.fit(train_X, train_y)

train_pred = model.predict(train_X)
train_results = pd.DataFrame({
    'TOTAL_VALUE': train_y,
    'predicted': train_pred,
    'residual': train_y - train_pred
})
train_results.head()
```

실행 결과

	TOTAL_VALUE	predicted	residual
2024	392.0	387.726258	4.273742
5140	476.3	430.785540	45.514460
5259	367.4	384.042952	-16.642952
421	350.3	369.005551	-18.705551
1401	348.1	314.725722	33.374278

표 2-13 검증 데이터 일부를 사용한 예측



웨스트 록스베리 검증 데이터를 사용해 모델을 검증하는 파이썬 코드

```
valid_pred = model.predict(valid_X)
valid_results = pd.DataFrame({
    'TOTAL_VALUE': valid_y,
    'predicted': valid_pred,
    'residual': valid_y - valid_pred
})
valid_results.head()
```

실행 결과

	TOTAL_VALUE	predicted	residual
1822	462.0	406.946377	55.053623
1998	370.4	362.888928	7.511072
5126	407.4	390.287208	17.112792
808	316.1	382.470203	-66.370203
4034	393.2	434.334998	-41.134998

표 2-14 학습 및 검증 데이터에 대한 예측 오차 측도(단위: 1,000달러)



모델의 평가 측도를 계산하는 파이썬 코드

```
# import the utility function regressionSummary
from dmbs import regressionSummary

# training set
regressionSummary(train_results.TOTAL_VALUE, train_results.predicted)

# validation set
regressionSummary(valid_results.TOTAL_VALUE, valid_results.predicted)
```

실행 결과

```
# training set
Regression statistics

                Mean Error (ME) : -0.0000
    Root Mean Squared Error (RMSE) : 43.0306
                Mean Absolute Error (MAE) : 32.6042
                Mean Percentage Error (MPE) : -1.1116
Mean Absolute Percentage Error (MAPE): 8.4886

# validation set
Regression statistics

                Mean Error (ME) : -0.1463
    Root Mean Squared Error (RMSE) : 42.7292
                Mean Absolute Error (MAE) : 31.9663
                Mean Percentage Error (MPE) : -1.0884
Mean Absolute Percentage Error (MAPE): 8.3283
```

- Error(오차) = 실제값(actual) - 예측값(predicted)
- ME(평균 오차) = Mean Error
- RMSE(평균 제곱근 오차) = Root-Mean-Squared Error = Square root of average Squared Error
- MAE(평균 절대 오차) = Mean Absolute Error
- MPE(MAPE에서 절대값을 제외하여 계산) = Mean Percentage Error
- MAPE(평균 절대 비율 오차) = Mean Absolute Percentage Error

9 알고리즘 결과 해석

10 모델 적용

표 2-15 점수화를 위한 3개의 관측치 데이터 프레임



레코드가 3개인 데이터 프레임의 점수화

```
new_data = pd.DataFrame({
    'LOT_SQFT': [4200, 6444, 5035],
    'YR_BUILT': [1960, 1940, 1925],
    'GROSS_AREA': [2670, 2886, 3264],
    'LIVING_AREA': [1710, 1474, 1523],
    'FLOORS': [2.0, 1.5, 1.9],
    'ROOMS': [10, 6, 6],
    'BEDROOMS': [4, 3, 2],
    'FULL_BATH': [1, 1, 1],
    'HALF_BATH': [1, 1, 0],
    'KITCHEN': [1, 1, 1],
    'FIREPLACE': [1, 1, 0],
    'REMODEL_Old': [0, 0, 0],
    'REMODEL_Recent': [0, 0, 1],
})
print(new_data)

print('Predictions: ', model.predict(new_data))
```

실행 결과

	LOT_SQFT	YR_BUILT	GROSS_AREA	LIVING_AREA	FLOORS	ROOMS	BEDROOMS	\
0	4200	1960	2670	1710	2.0	10	4	
1	6444	1940	2886	1474	1.5	6	3	
2	5035	1925	3264	1523	1.9	6	2	

	FULL_BATH	HALF_BATH	KITCHEN	FIREPLACE	REMODEL_Old	REMODEL_Recent
0	1	1	1	1	0	0
1	1	1	1	1	0	0
2	1	0	1	0	0	1

Predictions: [384.47210285 378.06696706 386.01773842]

데이터 마이닝의 윤리 이슈 (교재 68~70p)

- 빅 브라더(Big Brother)
- 자동 무기 기술
- 편견과 차별
- 인터넷 음모 및 루머
- 심리 조작
- 개인 데이터의 오용

Thank You !