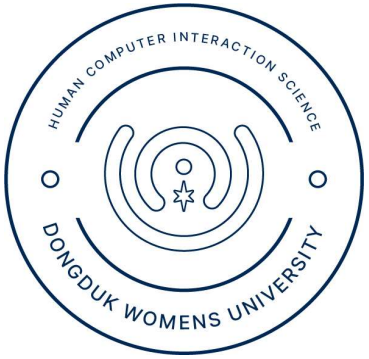
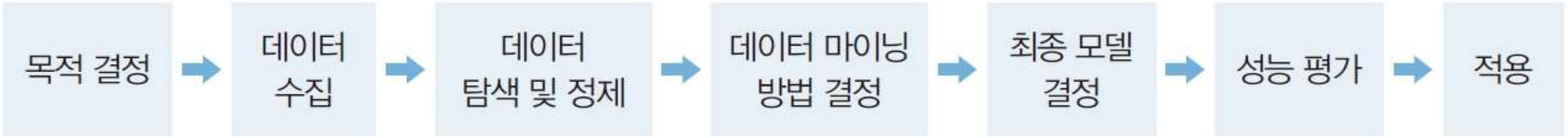
데이터마이닝
- 데이터마이닝 개요 -



2023.03.13 (MON)

동덕여자대학교 HCI사이언스

데이터 마이닝 프로세스



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 타깃(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

웨스트 록스베리 지역의 주택 가격 데이터

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

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

파이썬에서 데이터 불러오기

[파이썬 코드]

```
#데이터로부터 데이터 셋을 생성하고 불러오는 코드
```

```
import pandas as pd
```

```
housing_df=pd.read_csv('WestRoxbury.csv')
```

```
# 데이터 프레임의 차원 확인
housing_df.shape
```

```
(5802, 14)
```

```
housing_df.head(10)
```

	TOTAL VALUE	TAX	LOT SQFT	YR BUILT	GROSS AREA	LIVING AREA	FLOORS	ROOMS	BEDROOMS	FULL BATH	HALF BATH	KITCHEN	FIREPLACE	REMODEL
0	344.2	4330	9965	1880	2436	1352	2.0	6	3	1	1	1	0	None
1	412.6	5190	6590	1945	3108	1976	2.0	10	4	2	1	1	0	Recent
2	330.1	4152	7500	1890	2294	1371	2.0	8	4	1	1	1	0	None
3	498.6	6272	13773	1957	5032	2608	1.0	9	5	1	1	1	1	None
4	331.5	4170	5000	1910	2370	1438	2.0	7	3	2	0	1	0	None
5	337.4	4244	5142	1950	2124	1060	1.0	6	3	1	0	1	1	Old
6	359.4	4521	5000	1954	3220	1916	2.0	7	3	1	1	1	0	None
7	320.4	4030	10000	1950	2208	1200	1.0	6	3	1	0	1	0	None
8	333.5	4195	6835	1958	2582	1092	1.0	5	3	1	0	1	1	Recent
9	409.4	5150	5093	1900	4818	2992	2.0	8	4	2	0	1	0	None

파이썬에서 데이터 불러오기

파이썬 코드

```
#열 이름 바꾸기: 점 표기를 허용하려면 공백을 '_'로 바꿔야 함
```

```
housing_df = housing_df.rename(columns={'TOTAL VALUE ': 'TOTAL_VALUE'})
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.columns = [s.strip().replace(' ', '_') for s in housing_df.columns]
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')
```

파이썬에서 데이터 불러오기

[파이썬 코드]

```
#데이터 표시 연습
```

```
housing_df.loc[0:3]  # loc의 경우 슬라이스의 두 번째 인덱스가 포함 loc[a:b]
```

	TOTAL_VALUE	TAX	LOT_SQFT	YR_BUILT	GROSS_AREA	LIVING_AREA	FLOORS	ROOMS	BEDROOMS	FULL_BATH	HALF_BATH	KITCHEN
0	344.2	4330	9965	1880	2436	1352	2.0	6	3	1	1	1
1	412.6	5190	6590	1945	3108	1976	2.0	10	4	2	1	1
2	330.1	4152	7500	1890	2294	1371	2.0	8	4	1	1	1
3	498.6	6272	13773	1957	5032	2608	1.0	9	5	1	1	1

```
housing_df.iloc[0:4]  # loc의 경우 슬라이스의 두 번째 인덱스는 제외 iloc[a:b]
```

	TOTAL_VALUE	TAX	LOT_SQFT	YR_BUILT	GROSS_AREA	LIVING_AREA	FLOORS	ROOMS	BEDROOMS	FULL_BATH	HALF_BATH	KITCHEN
0	344.2	4330	9965	1880	2436	1352	2.0	6	3	1	1	1
1	412.6	5190	6590	1945	3108	1976	2.0	10	4	2	1	1
2	330.1	4152	7500	1890	2294	1371	2.0	8	4	1	1	1
3	498.6	6272	13773	1957	5032	2608	1.0	9	5	1	1	1

파이썬에서 데이터 불러오기

[파이썬 코드]

```
#TOTAL_VALUE 열의 처음 10개 값을 표시하는 다양한 방법
```

```
housing_df['TOTAL_VALUE'].iloc[0:10]
```

```
0    344.2
1    412.6
2    330.1
3    498.6
4    331.5
5    337.4
6    359.4
7    320.4
8    333.5
9    409.4
Name: TOTAL_VALUE, dtype: float64
```

```
housing_df.iloc[0:10].TOTAL_VALUE # 열 이름에 공백이 없으면 점 표기법 사용
```

```
0    344.2
1    412.6
2    330.1
3    498.6
4    331.5
5    337.4
6    359.4
7    320.4
8    333.5
9    409.4
Name: TOTAL_VALUE, dtype: float64
```

```
housing_df.iloc[0:10]['TOTAL_VALUE']  # 순서 상관 X
```

```
0    344.2
1    412.6
2    330.1
3    498.6
4    331.5
5    337.4
6    359.4
7    320.4
8    333.5
9    409.4
Name: TOTAL_VALUE, dtype: float64
```

파이썬에서 데이터 불러오기

파이썬 코드

```
# 처음 10개 열의 다섯 번째 행을 표시
```

```
housing_df.iloc[4][0:10]
```

```
TOTAL_VALUE    331.5
TAX             4170
LOT_SQFT        5000
YR_BUILT        1910
GROSS_AREA      2370
LIVING_AREA     1438
FLOORS           2.0
ROOMS              7
BEDROOMS           3
FULL_BATH          2
Name: 4, dtype: object
```

```
housing_df.iloc[4, 0:10]
```

```
TOTAL_VALUE    331.5
TAX             4170
LOT_SQFT        5000
YR_BUILT        1910
GROSS_AREA      2370
LIVING_AREA     1438
FLOORS           2.0
ROOMS              7
BEDROOMS           3
FULL_BATH          2
Name: 4, dtype: object
```

```
housing_df.iloc[4:5, 0:10] #데이터 프레임 형식을 유지하려면 행에 슬라이스 사용
```

	TOTAL_VALUE	TAX	LOT_SQFT	YR_BUILT	GROSS_AREA	LIVING_AREA	FLOORS	ROOMS
4	331.5	4170	5000	1910	2370	1438	2.0	7

파이썬에서 데이터 불러오기

파이썬 코드

```
# 전체 열
```

```
housing_df.iloc[:,0:1]
```

	TOTAL_VALUE
0	344.2
1	412.6
2	330.1
3	498.6
4	331.5
...	...
5797	404.8
5798	407.9
5799	406.5
5800	308.7
5801	447.6

5802 rows × 1 columns

```
housing_df['TOTAL_VALUE']
```

```
0       344.2
1       412.6
2       330.1
3       498.6
4       331.5
        ...
5797    404.8
5798    407.9
5799    406.5
5800    308.7
5801    447.6
Name: TOTAL_VALUE, Length: 5802, dtype: float64
```

```
housing_df['TOTAL_VALUE'][0:10]
```

```
0    344.2
1    412.6
2    330.1
3    498.6
```

파이썬에서 데이터 불러오기

[파이썬 코드]

```
#기술 통계
```

```
print('Number of rows ', len(housing_df['TOTAL_VALUE']))
print('Mean of TOTAL_VALUE ', housing_df['TOTAL_VALUE'].mean())
```

```
Number of rows  5802
Mean of TOTAL_VALUE  392.6857149258885
```

```
housing_df['TOTAL_VALUE'].describe()
```

```
count    5802.000000
mean      392.685715
std        99.177414
min       105.000000
25%       325.125000
50%       375.900000
75%       438.775000
max      1217.800000
Name: TOTAL_VALUE, dtype: float64
```

```
housing_df.describe()
```

	TOTAL_VALUE	TAX	LOT_SQFT	YR_BUILT	GROSS_AREA	LIVING_AREA
count	5802.000000	5802.000000	5802.000000	5802.000000	5802.000000	5802.000000
mean	392.685715	4939.485867	6278.083764	1936.744916	2924.842123	1657.065322
std	99.177414	1247.649118	2669.707974	35.989910	883.984726	540.456726
min	105.000000	1320.000000	997.000000	0.000000	821.000000	504.000000
25%	325.125000	4089.500000	4772.000000	1920.000000	2347.000000	1308.000000
50%	375.900000	4728.000000	5683.000000	1935.000000	2700.000000	1548.500000
75%	438.775000	5519.500000	7022.250000	1955.000000	3239.000000	1873.750000
max	1217.800000	15319.000000	46411.000000	2011.000000	8154.000000	5289.000000

파이썬 패키지 불러오기

파이썬 코드

필요한 파이썬 라이브러리 불러오기

```
import numpy as np
import pandas as pd
from sklearn.model_selection import train_test_split   <- 훈련/평가 분할
from sklearn.metrics import r2_score                    <- 결정 계수 함수
from sklearn.linear_model import LinearRegression      <- 선형 회귀분석
```

데이터셋 샘플링

파이썬 코드

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

```
#샘플링 방법
```

```
#랜덤 샘플
housing_df.sample(5)
```

UE	TAX	LOT_SQFT	YR_BUILT	GROSS_AREA	LIVING_AREA	FLOORS	ROOMS	BEDROOMS	FULL_BATH	HALF_BATH	KITCHEN
8.8	4136	4717	1906	2824	1469	1.5	6	3	2	0	1
2.7	4436	5038	1920	2483	1554	2.0	7	3	1	0	1
1.7	3292	4226	1958	2114	1025	1.0	5	3	1	0	1
3.5	3818	5214	1920	2256	1344	2.0	7	3	1	0	1
3.8	6715	6500	1910	5182	3264	2.0	9	4	1	1	1

```
# 방 개수 10개 이상인 집 오버 샘플링
weights = [0.9 if rooms > 10 else 0.01 for rooms in housing_df.ROOMS]
housing_df.sample(5, weights=weights)
```

	TOTAL_VALUE	TAX	LOT_SQFT	YR_BUILT	GROSS_AREA	LIVING_AREA	FLOORS	ROOMS	BEDROOMS	FULL_BATH	HA
2555	774.1	9738	11970	1931	6110	3180	2.0	12	6	2	
2118	935.1	11763	25200	1954	6840	5289	1.0	13	9	2	
1528	323.6	4070	4050	1920	1910	984	2.0	6	3	1	
5675	478.6	6020	6500	1957	3748	2280	2.0	12	5	2	
4594	511.7	6437	7190	1938	3577	1688	1.5	11	4	2	

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

데이터 전처리와 데이터 정제

파이썬 코드

```
#판다스 활용 변수 특성 파악
# 변수 리스트
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')
```

```
# REMODEL 열은 factor 이기 때문에 데이터 타입 변경
print(housing_df.REMODEL.dtype)
housing_df.REMODEL = housing_df.REMODEL.astype('category')
print(housing_df.REMODEL.cat.categories)
print(housing_df.REMODEL.dtype)  # 'category' 타입
```

```
object
Index(['None', 'Old', 'Recent'], dtype='object')
category
```

데이터 전처리와 데이터 정제

파이썬 코드

```
print(housing_df.BEDROOMS.dtype)
print(housing_df.TOTAL_VALUE.dtype)
```

```
int64
float64
```

```
housing_df.dtypes
```

```
TOTAL_VALUE     float64
TAX               int64
LOT_SQFT          int64
YR_BUILT          int64
GROSS_AREA        int64
LIVING_AREA       int64
FLOORS          float64
ROOMS             int64
BEDROOMS          int64
FULL_BATH         int64
HALF_BATH         int64
KITCHEN           int64
FIREPLACE         int64
REMODEL        category
dtype: object
```

데이터 전처리와 데이터 정제

파이썬 코드

■ 범주형 변수 처리

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

```
#판다스 이용한 가변수 생성
```

```
# drop_first = True 첫 번째 더미 변수를 삭제
housing_df = pd.get_dummies(housing_df, prefix_sep='_', drop_first=True)
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_Old', 'REMODEL_Recent'],
      dtype='object')
```

```
print(housing_df.loc[:, 'REMODEL_Old':'REMODEL_Recent'].head(5))
```

```
   REMODEL_Old  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

데이터 전처리와 데이터 정제

파이썬 코드

```
#결측 데이터 처리 => 결측치를 중앙값으로 대체하는 파이썬 코드
# 누락된 데이터 절차를 설명하기 위해 먼저 침실에 대한 몇 가지 항목을 NA로 변환
# 나머지 값의 중앙값을 사용하여 이러한 누락된 값을 대치
```

```
print('Number of rows with valid BEDROOMS values before: ',
      housing_df['BEDROOMS'].count())
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())
housing_df['BEDROOMS'].count()
```

```
Number of rows with valid BEDROOMS values before:  5802
Number of rows with valid BEDROOMS values after setting to NAN:  5792

5792
```

```
# 결측치 값이 있는 행 삭제
reduced_df = housing_df.dropna()
print('Number of rows after removing rows with missing values: ', len(reduced_df))
```

```
Number of rows after removing rows with missing values:  5792
```

```
# 나머지 값의 중앙값을 사용하여 누락된 값을 바꿈
# 기본적으로 pandas 데이터 프레임의 중앙값 방법은 NA 값을 무시
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())
```

```
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, 횟수) 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 StandardScalar() 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

데이터 전처리와 데이터 정제

파이썬 코드

```
# 정규화 및 스케일
```

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

# 데이터 프레임 normalizing

# 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)

# 변환 결과는 numpy 배열이며 데이터 프레임으로 변환

# 데이터 프레임 rescaling
# pandas:
rescaled_df = (housing_df - housing_df.min()) / (housing_df.max() - housing_df.min())

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

예측력과 과적합

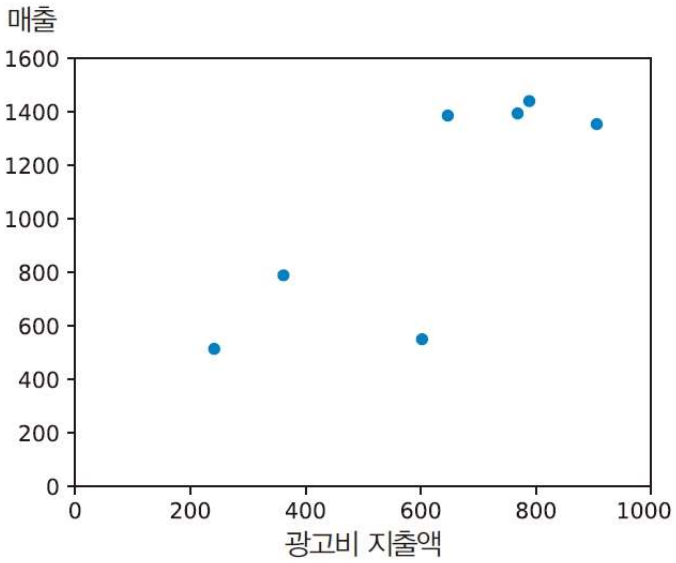
과적합

- 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

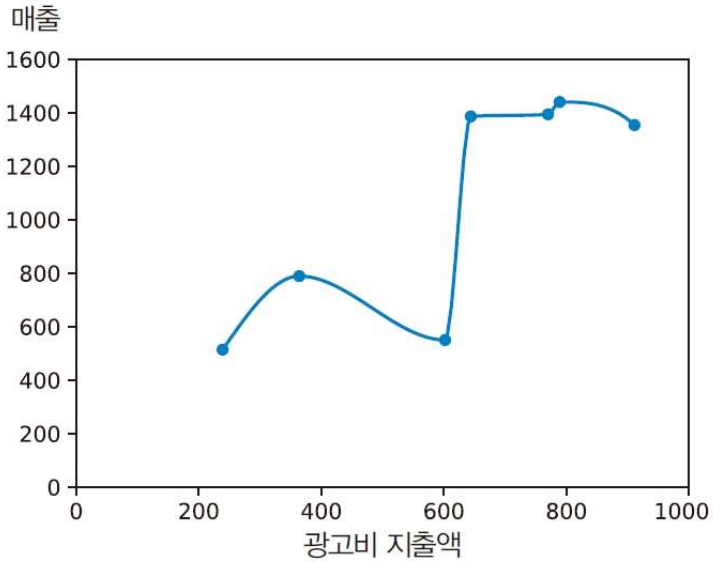
과적합

광고비 판매량

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



광고비와 매출액 관계 산점도



과적합: 현 데이터에서 오차가 0인 곡선

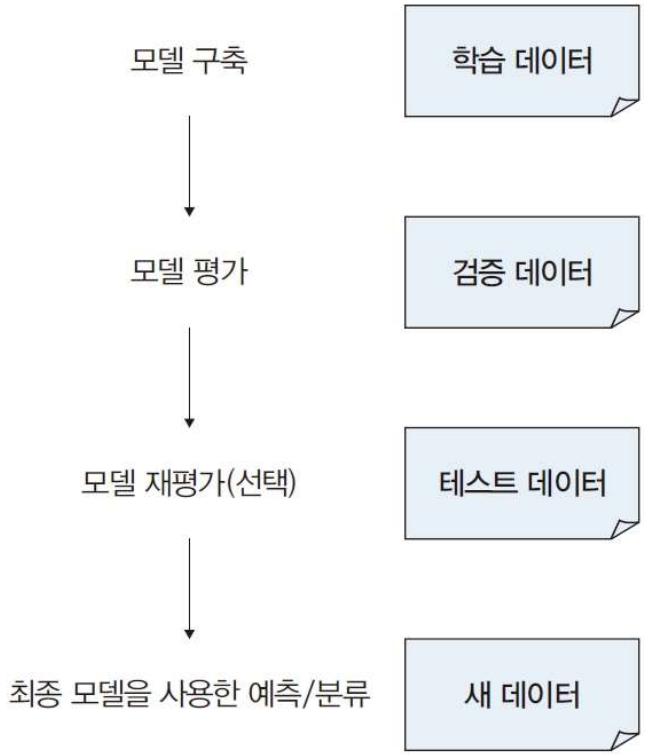
현 데이터에는 한 치의 오차도 없어 보이지만, 광고비 지출을 기준으로 미래의 데이터(향후 매출액)를 예측하기에는 무리가 있음

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



데이터 마이닝 프로세스에서 학습/검증/테스트 데이터의 역할

데이터 분할

파이썬 코드

```
# 데이터 분할
# 웨스트 록스베리 데이터를 학습/검증/시험 데이터로 분할
```

```
# random_state는 코드를 다시 실행할 때 동일한 파티션을 얻기 위해 정의된 값으로 설정
# 훈련 60% 검증 40%
trainData= housing_df.sample(frac=0.6, random_state=1)

# 훈련 세트에 아직 없는 행을 검증에 할당
validData = housing_df.drop(trainData.index)

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

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

```
# scikit-learn을 사용하는 다른 방법
from sklearn.model_selection import train_test_split

trainData, validData = train_test_split(housing_df, test_size=0.40, random_state=1)
print('Training   : ', trainData.shape)
print('Validation : ', validData.shape)
```

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

데이터 분할

파이썬 코드

```
# 학습을 위해 행 ID의 50%를 무작위로 샘플링
trainData = housing_df.sample(frac=0.5, random_state=1)
# 행 ID의 30%를 검증 세트로 샘플링하여 레코드에서만 가져옴
# 50%의 60%는 30% => 검증 데이터로 활용
validData = housing_df.drop(trainData.index).sample(frac=0.6, random_state=1)
# 나머지 20% 를 테스트 데이터로 활용
testData = housing_df.drop(trainData.index).drop(validData.index)

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

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

```
# scikit-learn을 사용하는 다른 방법
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)
```

```
Training   :  (2901, 15)
Validation :  (1740, 15)
Test       :  (1161, 15)
```

모델 구축: 선형 회귀 분석을 이용한 예제

■ 모델링 과정

❶ 목적 웨스트 록스베리 지역 주택 가격을 예측한다.

❷ 데이터셋 획득 2014년 웨스트 록스베리 주택 가격 데이터셋을 이용할 것이며, 전체 데이터의 수가 많지 않아 별도의 표본을 취하지 않았다.

❸ 데이터 탐색 /정제 /전처리

웨스트 록스베리 주택 가격 데이터의 이상치

FLOORS	ROOMS
15	8
2	10
1.5	6
1	6

❹ 데이터 축소

❺ 데이터 마이닝 문제 결정

❻ 데이터 분할(지도 학습용)

❼ 데이터 마이닝 기법 선택

모델 구축: 선형 회귀 분석을 이용한 예제

파이썬 코드

❽ 알고리즘 사용해 과제 수행

```
!pip install dmba
```

```
Collecting dmba
  Downloading dmba-0.1.0-py3-none-any.whl (11.8 MB)
Installing collected packages: dmba
Successfully installed dmba-0.1.0
```

```
# 학습 데이터 일부에 대한 예측
# 웨스트 록스베리 학습 데이터를 사용해 모델 구축
```

```
# 데이터 불러오기 + 전처리
#housing_df = dmba.load_data('WestRoxbury.csv')
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)

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

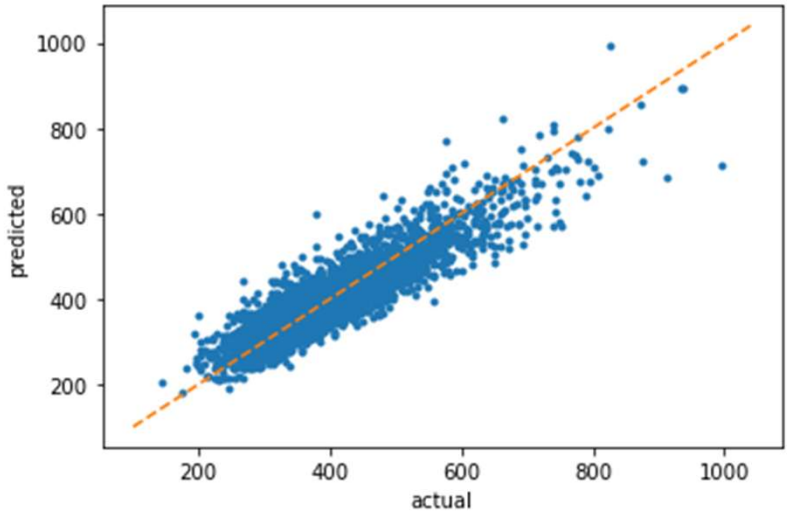
```
# 데이터 분할
X = housing_df[predictors]
y = housing_df[outcome]
train_X, valid_X, train_y, valid_y = train_test_split(X, y, test_size=0.4, random_state=1)

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
})
print(train_results.head())
```

```
      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
```

모델 구축: 선형 회귀 분석을 이용한 예제

파이썬 코드

```
plt.plot(train_results.TOTAL_VALUE, train_results.predicted, '.')
plt.xlabel('actual')  # set x-axis label
plt.ylabel('predicted')  # set y-axis label
axes = plt.gca()
plt.plot(axes.get_xlim(), axes.get_xlim(), '--')
plt.show()
```




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

```
      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
```

모델 구축: 선형 회귀 분석을 이용한 예제

파이썬 코드

```
#학습 및 검증 데이터에 대한 예측 오차 측도. 단위 1,000달러
#모델의 평가 측도를 계산
```

```
# utility function regressionSummary 임포트
from dmba import regressionSummary

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

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

```
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

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) = Pean Percentage Error
- MAPE(평균 절대 비율 오차) = Mean Absolute Percentage Error

모델 구축: 선형 회귀 분석을 이용한 예제

파이썬 코드

❾ 알고리즘 결과 해석

❿ 모델 적용

```
#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))
```

```
   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

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

과제 – 1

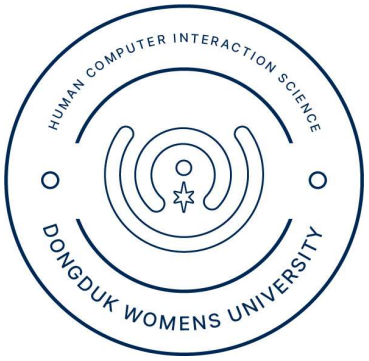
ToyotaCorolla.csv 데이터에서

- Fuel Type 변수와 Metalic 변수를 pandas를 활용하여 가변수로 변환하라.
- 학습 50%, 검증 30%, 시험 20%로 분할하고 역할을 설명하라.

- Addendum
  - 본 데이터를 대상으로 본인이 현재까지 알고 있는 지식을 활용하여 데이터 마이닝적 관점에서 분석 및 해석하라.
  - 목적 : 현재 수강생의 상황을 파악하기 위함.
  - 벤처 스타트업 아카데미 지원 학생은 필수로 작성해주기 바람.

Q & A


동덕여자대학교
DONGDUK WOMEN'S UNIVERSITY


HUMAN COMPUTER INTERACTION SCIENCE
DONGDUK WOMENS UNIVERSITY