데이터 표현과 특성공학

July 25, 2023

데이터 과학 진행 과정

- 1. 연구 목표 설정
- 2 데이터 획득
- 3. 데이터 준비
- 4.데이터 탐색
- 5. 데이터 모델링
  - (모델 및 변수 선택)
    모델링 실시
  - 모델 진단과 모델 비교
- 6. 발표 및 자동화

데이터 표현

- **데이터셋의 데이터는 2차원 실수형 배열로 각 열이 데이터 포인트를 설명하는 연속형 특성이라고 가정**
- **실제로 수집되는 데이터**
  - 일반적인 특성의 전형적인 형태는 범주형 특성
  - 이산형 특성이라고도 하는 이런 특성은 보통 숫자 값이 아니다.
  - 비지도 학습은 데이터를 잘 이해하기 위해서(탐색적 분석) 사용되거나, 지도학습 알고리즘의 정확도 향상과 메모리/시간 절약을 위해 새로운 데이터 표현을 만드는 전처리 단계로 활용

Categorical Variable to Numeric Variable

- 범주형 변수를 수치형 변수로 나타내는 방법
- 범주형 변수란, 차의 등급을 나타내는 [소형, 중형, 대형] 처럼 표현되는 변수를 말합니다.
- 범주형 변수는 주로 데이터 상에서 문자열로 표현되는 경우가 많으며, 문자와 숫자가 매핑되는 형태로 표현되기도 합니다.

데이터 표현

- 1994년 인구조사 데이터베이스에서 추출한 미국 성인의 소득 데이터 셋(Adult 데이터셋)

	age	workclass	education	gender	hours-per-week	occupation	income
0	39	State-gov	Bachelors	Male	40	Adm-clerical	<=50K
1	50	Self-emp-not-inc	Bachelors	Male	13	Exec-managerial	<=50K
2	38	Private	HS-grad	Male	40	Handlers-cleaners	<=50K
3	53	Private	11th	Male	40	Handlers-cleaners	<=50K
4	28	Private	Bachelors	Female	40	Prof-specialty	<=50K

- 연속적 특성 : age, hours-per-week(숫자값)
- 범주형 특성 : workclass, education, gender, occupation
  - 정량적이 아닌 정성적인 속성

- X[2]가 “Masters”, “Bachelors”가 될 수 없다. 로지스틱 회귀를 사용하려면 데이터를 다른 방식으로 표현해야 한다.

$$y = w_0x_0 + w_1x_1 + w_2x_2 + \ldots . w_nx_n + b$$

특성공학

- **특성 공학 : 특정 애플리케이션에 가장 적합한 데이터 표현을 찾는 것**
- 데이터 과학자와 머신러닝 기술자가 실제 문제를 풀기 위해 당면하는 주요 작업중 하나로
- 올바른 데이터 표현은 지도 학습 모델에서 적절한 매개변수를 선택하는 것보다 성능에 더 큰 영향을 미칩니다.

원-핫-인코딩(가변수)

➢ 원-핫-인코딩(one-hot-encoding)

- 원핫 인코딩은 n개의 범주형 데이터를 n개의 비트(0,1) 벡터로 표현
- 범주형 변수를 표현하는데 가장 널리 쓰이는 방법
- 원-아웃-오브-엔-인코딩 혹은 가변수라고도 한다.

  가변수는 범주형 변수를 0또는 1값을 가진 하나 이상의 새로운 특성으로 바꾼 것이다.

- 예제1994년 인구조사 데이터베이스에서 추출한 미국 성인의 소득 데이터셋
  - 근로자 나이(age), 고용 형태(workclass), 교육 수준(education), 성별(gender), 주당 근로시간(hours-per-week), 직업(occupation)등의 특성이 있다.

Adult 데이터셋 확인(1)

➢ 사용해 어떤 근로자의 수입이 50,000달러를 초과하는지 예측

```
import os
import mglearn
import pandas as pd

# "names" 매개변수로 열 이름을 제공
data = pd.read_csv(os.path.join(mglearn.datasets.DATA_PATH, "adult.data"), header=None,
index_col=False,names=['age', 'workclass', 'fnlwgt', 'education', 'education-num', 'martial-status',
                       'occupation', 'relationship', 'race', 'gender', 'capital-gain', 'capital-loss',
                       'hours-per-week', 'native-country', 'income'])

# 예제를 위해 몇개만 선택하기
data = data[['age', 'workclass', 'education', 'gender', 'hours-per-week', 'occupation', 'income']]
display(data.head())
```

범주형 데이터 문자열 확인하기

- **pandas에서 Series에 있는 value_counts메서드 사용한 유일한 값이 각각 몇 번 나타나는지 출력**

```
print(data.gender.value_counts())


#  Male      21790
#  Female    10771
# Name: gender, dtype: int64
```

  - 데이터셋의 gender는 Male, Female 두가지 값
    - 원-핫-인코딩으로 나타내기 좋은 형태

get_dummies 함수

➢ get_dummies 함수 : 객체 타입이나 범주형을 가진 열을 자동 변환

```
print("원본 특성: \n",list(data.columns),"\n")

# 원본 특성:
#  ['age', 'workclass', 'education', 'gender', 'hours-per-week', 'occupation', 'income']

data_dummies = pd.get_dummies(data)
print("get_dummies  후의 특성: \n",list(data_dummies.columns),"\n")
```

get_dummies 후의 특성:

➢ 연속형 특성인 age와 hours-per-week는 그대로지만 범주형 특성은 값 마다 새로운 특성으로 확장되었다.

```
#  ['age', 'hours-per-week', 'workclass_ ?', 'workclass_ Federal-gov', 'workclass_ Local-gov', 'workclass_ Never-
worked', 'workclass_ Private', 'workclass_ Self-emp-inc', 'workclass_ Self-emp-not-inc', 'workclass_ State-gov',
'workclass_ Without-pay', 'education_ 10th', 'education_ 11th', 'education_ 12th', 'education_ 1st-4th', 'education_
5th-6th', 'education_ 7th-8th', 'education_ 9th', 'education_ Assoc-acdm', 'education_ Assoc-voc', 'education_
Bachelors', 'education_ Doctorate', 'education_ HS-grad', 'education_ Masters', 'education_ Preschool',
'education_ Prof-school', 'education_ Some-college', 'gender_ Female', 'gender_ Male', 'occupation_ ?',
'occupation_ Adm-clerical', 'occupation_ Armed-Forces', 'occupation_ Craft-repair', 'occupation_ Exec-managerial',
'occupation_ Farming-fishing', 'occupation_ Handlers-cleaners', 'occupation_ Machine-op-inspct', 'occupation_
Other-service', 'occupation_ Priv-house-serv', 'occupation_ Prof-specialty', 'occupation_ Protective-serv',
'occupation_ Sales', 'occupation_ Tech-support', 'occupation_ Transport-moving', 'income_ <=50K',
```

범주형 데이터 문자열 확인하기

```
data_dummies.head()
```

	age	hours-per-week	workclass_?	workclass_Federal-gov	workclass_Local-gov	workclass_Never-worked	workclass_Private	workclass_Self-emp-inc
0	39	40	0	0	0	0	0	0
1	50	13	0	0	0	0	0	0
2	38	40	0	0	0	0	1	0
3	53	40	0	0	0	0	1	0
4	28	40	0	0	0	0	1	0

5 rows × 46 columns

- data_dummies의 values 속성을 이용해 DataFrame을 NumPy 배열로 바꿀 수 있으며, 이를 이용해 머신러닝 모델을 학습시킨다.
- 모델을 학습 시키기 전에 이 데이터로부터 타깃값을 분리해야 한다.
  출력값이나 출력값으로부터 유도된 변수를 특성 표현에 포함하는 것은 지도 학습 모델을 만들 때 특히 저지르기 쉬운 실수이다.

범주형 데이터 문자열 확인하기

```
features = data_dummies.loc[:,'age':'occupation_ Transport-moving']
# NumPy 배열 추출
X = features.values
y = data_dummies['income_ <=50K'].values
print("X.shape : {}  y.shape : {}".format(X.shape,y.shape))
# X.shape : (32561, 44)  y.shape : (32561,)
```

- 여기서는 특성을 포함한 열, 즉 age부터 occupation_ Transport-moving까지 모든 열을 추출한다.
- 이 범위에는 타깃을 뺀 모든 특성이 포함된다.

- [LogisticRegression 을 사용한 성능 테스트 해보기]

```
from sklearn.linear_model import LogisticRegression
from sklearn.model_selection import train_test_split
X_train, X_test, y_train, y_test = train_test_split(X, y, random_state=0)
logreg = LogisticRegression()
logreg.fit(X_train, y_train)
print("테스트 점수 : {:.2f}".format(logreg.score(X_train,y_train)))
# 테스트 점수 : 0.81
```

  - 범주형 데이터를 원-핫-인코딩 방법을 통해 변환하여 이전에 사용하던 방식처럼 사용 가능하게 하였다.

4.1.2 숫자로 표현된 범주형 특징

➢ 인구조사 데이터가 설문지를 통해 얻었다고 한다면,

- 주관식으로 직업을 직접 쓸 수도 있지만,
- 보통은 1. 공무원 2. 직장인 3. 자영업자 와 같이 번호로 체크를 한다.
- 이런 경우 결과값은 "변호사"와 같은 문자열이 아니라 0,1,2,의 값들은 숫자열로 채워진다.

```
import pandas as pd
demo_df = pd.DataFrame({'숫자 특성':[0, 1, 2, 1], '범주형 특성':['양말', '여우', '양말', '상자']})
display(demo_df)
```

	범주형 특성	숫자 특성
0	양말	0
1	여우	1
2	양말	2
3	상자	1

```
# get_dummies를 사용하면 문자열 특성만 인코딩 되며 숫자 특성은 바뀌지 않는다.
display(pd.get_dummies(demo_df))
```

	숫자 특성	범주형 특성_상자	범주형 특성_양말	범주형 특성_여우
0	0	0	1	0
1	1	0	0	1
2	2	0	1	0
3	1	1	0	0

4.1.2 숫자로 푠현된 범주형 특징

- **숫자 특성'도 가변수로 만들고 싶다면 columns 매개변수에 인코딩하고 싶은 열을 명시해야 한다.**
  - 그러면 두 특성을 모두 범주형으로 간주한다.주관식으로 직업을 직접 쓸 수도 있지만,

```
demo_df['숫자 특성'] = demo_df['숫자 특성'].astype(str)
display(pd.get_dummies(demo_df, columns=['숫자 특성', '범주형 특성']))
```

	숫자 특성_0	숫자 특성_1	숫자 특성_2	범주형 특성_상자	범주형 특성_양말	범주형 특성_여우
0	1	0	0	0	1	0
1	0	1	0	0	0	1
2	0	0	1	0	1	0
3	0	1	0	1	0	0

4.2 숫자로 표현된 범주형 특징

- **숫자 특성'도 가변수로 만들고 싶다면 columns 매개변수에 인코딩하고 싶은 열을 명시해야 한다.**
  - 그러면 두 특성을 모두 범주형으로 간주한다.주관식으로 직업을 직접 쓸 수도 있지만,

```
demo_df['숫자 특성'] = demo_df['숫자 특성'].astype(str)
display(pd.get_dummies(demo_df, columns=['숫자 특성', '범주형 특성']))
```

	숫자 특성_0	숫자 특성_1	숫자 특성_2	범주형 특성_상자	범주형 특성_양말	범주형 특성_여우
0	1	0	0	0	1	0
1	0	1	0	0	0	1
2	0	0	1	0	1	0
3	0	1	0	1	0	0

요약 및 정리

- 범주형 변수
- ONE-HOT-ENCODING 범주형 변수처럼 -> 머신러닝 알고리즘에 적합한 방식으로 표현하는 것이 중요
- 새로운 특성을 만드는 것과 데이터로 표현하는 것이 중요
- 새로운 특성을 만드는 것과 데이터 특성을 유도하기 위해 전문가 지식 활용하는 것 중요

Questions or Comments?