

노트북: 첫 번째 노트북

만든 날짜: 2020-12-26 오후 10:51

URL: <https://continuous-development.tistory.com/160?category=736685>

대체법의 종류

우선, 전통적인 대체방법으로는 **완전 제거법**, **한쌍 제거법(pairwise deletion)**, **평균 대체법(mean substitution)**, **회귀 대체법(regression imputation)**, **확률적 회귀대체(stochastic imputation)** 등이 있다.

제거법 : 완전 제거법 / 한쌍 제거법

완전 제거법은 결측치가 있는 행 자체를 지워버리는 방법

한쌍 제거법은 결측을 포함하는 응답자를 분석에서 제외하고, 남아 있는 관측치에 대해 통계분석을 시행

제거법의 단점 - 표본의 수가 줄어들어 통계적 검정력이 떨어짐

단일 대체법 : 평균 대체/ 회귀 대체

평균 대체와 회귀 대체법 등은 단일 대체법(single imputation)으로 분류되는데, 각각의 결측치를 일정한 과정을 통해 생성된 하나의 값으로 대체하는 방법을 뜻한다.

단일 대체법의 단점평균대체와 회귀 대체법(단일 대체법(single imputation) - 편의 추정량을 발생 , 통계적 검정력의 관 점에서도 문제

평균 대체 - 표준오차 (standard error)가 과소추정 , 해당 변수의 분산을 작게 하는 문제와 함께 다른 변수와의 상관관계를 낮추는 등 분석 결과의 편의

회귀 대체- 평균 대체와 달리 설명변수의 조건부 평균으로 결측을 대체 하기 때문에 더욱 발전된 방법으로 생각되지만, 이 경우에도 단일 대체가 가지는 한계를 그대로 가진다.

(결측 된 변수의 관측된 값을 종속변수로 하고, 나머지 변수를 설명변 수로 하여 추정한 회귀식을 활용하여 결측된 변수의 결측치를 추정)

확률적 회귀 대체법 - 표본오차를 과소 추정하여 1종 오류를 야기
(결측 변수의 결측치의 추정에 일정 한 확률 오차항(random error term)을 포함시켜 변동성(variability)을 고려하는 방식)

다중 대체법

다중 대체법은 3단계로 구성되는데 가능한 대체 값의 분포에서 추출된 서로 다른 값으로 결측치를 처리한 복수의 데이터셋을 생성한 뒤(imputation phase), 이들 데이터셋에 대하여 각각 분석을 수행하고(analysis phase), 그 결과 얻은 모수의 추정량과 표본오차를 통합하여 (pooling phase) 하나의 분석 결과를 제시하는 방법

다중 대체의 단계를 구체적으로 살펴보면 다음과 같다. 첫 번째 단계는 각각의 결측치를 일정한 알고리즘에 따라 다른 대체 값으로 대체한 m개의 데이터셋을 생성한다. 다음으로 m개의 완전한 데이터셋을 각각 분석하고 각 데이터셋의 분석 결과에서 모수의 추정 치와 표준오차를 확보한다. 마

지막으로 각 데이터 셋의 결과를 Rubin's rule8)(Rubin, 1987)에 의해 결합한다(Graham, 2012)

분석모형에서는 필요하지 않은 변수라고 할 지라도 대체 모형에 포함시키는 경우 통계적 대체방법의 성능을 향상한다는 것을 확인할 수 있다. 또한, 결측 된 변수의 결측 원인을 가장 잘 설명할 수 있는 변수를 연구자의 경험이나 이론을 통해 파악할 수 있다면, 분석모형과는 관계없는 변수라고 할지라도 대체 모형에 포함하는 것이 효과적이라는 것을 확인할 수 있다. 마지막으로, 분석모형에 포함된 변수로만 대체 모형을 구성하는 경우, 즉 FCS_REAL의 결과도 세 변수 모두에서 결측률 30% 이하에서 CP_REAL의 신뢰구간에 포함되는 것을 알 수 있다. 이는 가장 보수적인 관점에서 통계적 대체방법을 적용하더라도 결측률 30% 이하에서는 분석 결과의 편의를 유발하지 않는다는 것을 보여준다.

다중 대체법을 일차적으로 활용

또한 다중대체법을 사용할 때 결측 발생에 영향을 줄 수 있는 가능한 많은 변수를 설문조사 결과를 통해 얻어 결측치 대체에 사용하면 성능의 향상을 기대할 수 있는 것으로 나타났다.

다중 대체법(결측치를 채우는 방법)


결측치를 제외한 나머지 변수들로 해당 결측치를 예측하는 것이다.

이것을 여러 번 반복을 통해 신뢰성 있는 대체 값을 넣어준다.

지금까지 나와있는 대체 방법 중에서는 이게 제일 성능이 좋다고 나와 있다.


사용법

문제에 대한 예시는 대표적인 회귀 문제인 house price를 예로 들었다.



```
1 pip install impute
```

다중 대체를 하기 위해선 impute가 필요하다 해당 라이브러리를 인스톨한다.



```
1 import pandas as pd
2 import numpy as np
3 from sklearn.experimental import enable_iterative_imputer
4 from sklearn.impute import IterativeImputer
5 from sklearn.linear_model import LinearRegression
6 from impute.imputation.cs import mice
7 from sklearn.model_selection import train_test_split
8 %matplotlib inline
9 import matplotlib.pyplot as plt # Matlab-style plotting
10 import seaborn as sns
11 from scipy import stats
12 from scipy.stats import norm, skew #for some statistics
13
```

from으로 impute를 넣고 import로 mice를 넣어준다.



```
1 test = pd.read_csv('/content/drive/My Drive/data/kaggle/house-prices/test.csv', encoding = 'cp949')
2 train = pd.read_csv('/content/drive/My Drive/data/kaggle/house-prices/train.csv', encoding = 'cp949')
3
```

케글을 통해 house price 데이터를 받고 csv로 읽어준다.

다중 대체를 하기 위해선 train과 test를 합치는 작업이 필요하다. 그러기 위해선 두 데이터 프레임이 같아야 한다.

그래서 두 데이터 프레임의 형태를 같게 한 후에 train과 test를 합쳤다.

```
[14] 1 train_no_salep_price = train.drop(['SalePrice'], axis=1)
      2
```

```
1 total = pd.concat([train_no_salep_price, test], axis=0, ignore_index=True)
2 total
```

	Id	MSSubClass	MSZoning	LotFrontage	LotArea	Street	Alley	LotShape	LandContour
0	1	60	RL	65.0	8450	Pave	NaN	Reg	Lvl
1	2	20	RL	80.0	9600	Pave	NaN	Reg	Lvl
2	3	60	RL	68.0	11250	Pave	NaN	IR1	Lvl
3	4	70	RL	60.0	9550	Pave	NaN	IR1	Lvl
4	5	60	RL	84.0	14260	Pave	NaN	IR1	Lvl
...
2914	2915	160	RM	21.0	1936	Pave	NaN	Reg	Lvl
2915	2916	160	RM	21.0	1894	Pave	NaN	Reg	Lvl
2916	2917	20	RL	160.0	20000	Pave	NaN	Reg	Lvl
2917	2918	85	RL	62.0	10441	Pave	NaN	Reg	Lvl
2918	2919	60	RL	74.0	9627	Pave	NaN	Reg	Lvl

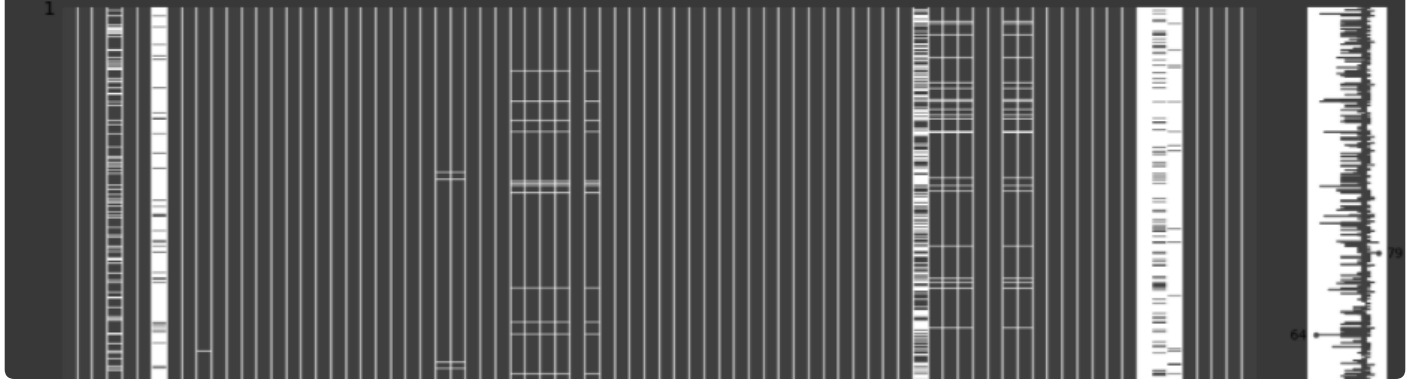
2919 rows × 10 columns

```
import missingno as msno
msno.matrix(total)
```

간단하게 결측치가 어떻게 되는지 missingno를 통해 확인한다.

```
1 import missingno as msno
2 msno.matrix(total)
3
```

<matplotlib.axes._subplots.AxesSubplot at 0x7fdf14ccca20>



저 하얀색 선 부분이 결측치가 있는 부분이다.

이제 total을 더미화한다.

```
dummy = pd.get_dummies(total)
```

더미화하는 이유는 범주형 변수는 기계학습을 할 때 다룰 수 없기 때문에 컴퓨터가 인식할 수 있는 값으로 바꿔줘야 하는 작업이 필요하기 때문이다.

더미화를 하면 범주형 칼럼들의 값들이 하나의 칼럼으로 변경되어서 0/1로 생긴다.

SF	TotalBsmtSF	1stFlrSF	2ndFlrSF	LowQualFinSF	GrLivArea	BsmtFullBath	BsmtHalfBath	FullBath	H
0.0	856.0	856	854	0	1710	1.0	0.0	2	
0.0	1262.0	1262	0	0	1262	0.0	1.0	2	
0.0	920.0	920	866	0	1786	1.0	0.0	2	
0.0	756.0	961	756	0	1717	1.0	0.0	1	
0.0	1145.0	1145	1053	0	2198	1.0	0.0	2	
...
0.0	546.0	546	546	0	1092	0.0	0.0	1	
0.0	546.0	546	546	0	1092	0.0	0.0	1	
0.0	1224.0	1224	0	0	1224	1.0	0.0	1	
0.0	912.0	970	0	0	970	0.0	1.0	1	
0.0	996.0	996	1004	0	2000	0.0	0.0	2	

이제 다중 대체를 사용한다.

```
1 total_impute = pd.DataFrame(IterativeImputer(verbose=False).fit_transform(dummy))
2 total_impute
```

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16
0	1.0	60.0	65.0	8450.0	7.0	5.0	2003.0	2003.0	196.0	706.0	0.0	150.0	856.0	856.0	854.0	0.0	1710.0
1	2.0	20.0	80.0	9600.0	6.0	8.0	1976.0	1976.0	0.0	978.0	0.0	284.0	1262.0	1262.0	0.0	0.0	1262.0
2	3.0	60.0	68.0	11250.0	7.0	5.0	2001.0	2002.0	162.0	486.0	0.0	434.0	920.0	920.0	866.0	0.0	1786.0
3	4.0	70.0	60.0	9550.0	7.0	5.0	1915.0	1970.0	0.0	216.0	0.0	540.0	756.0	961.0	756.0	0.0	1717.0
4	5.0	60.0	84.0	14260.0	8.0	5.0	2000.0	2000.0	350.0	655.0	0.0	490.0	1145.0	1145.0	1053.0	0.0	2198.0
...
2914	2915.0	160.0	21.0	1936.0	4.0	7.0	1970.0	1970.0	0.0	0.0	0.0	546.0	546.0	546.0	546.0	0.0	1092.0
2915	2916.0	160.0	21.0	1894.0	4.0	5.0	1970.0	1970.0	0.0	252.0	0.0	294.0	546.0	546.0	546.0	0.0	1092.0
2916	2917.0	20.0	160.0	20000.0	5.0	7.0	1960.0	1996.0	0.0	1224.0	0.0	0.0	1224.0	1224.0	0.0	0.0	1224.0
2917	2918.0	85.0	62.0	10441.0	5.0	5.0	1992.0	1992.0	0.0	337.0	0.0	575.0	912.0	970.0	0.0	0.0	970.0
2918	2919.0	60.0	74.0	9627.0	7.0	5.0	1993.0	1994.0	94.0	758.0	0.0	238.0	996.0	996.0	1004.0	0.0	2000.0

imputer를 사용하면 아래와 같이 결측치의 데이터가 대체된다.



데이터가 들어간걸 간단하게 확인할 수 있다.

```
1 total_cols = list(dummy.columns)
2 total_impute.columns = total_cols
3 # pd.DataFrame(IterativeImputer(verbose=False).fit_transform(dummy)).head()
4

[24] 1 X_train = total_impute[:1460]
     2 X_test = total_impute[1460:]
     3 # y_test = test['SalePrice']
     4 # train['SalePrice'] = np.log1p(train['SalePrice'])

1 y_train = np.log1p(train['SalePrice'])
2

[26] 1 total_impute.shape

(2919, 289)
```

그 이후에는 위와 같은 절차로 합쳐놔던 데이터 프레임을 분리하고 머신러닝을 돌리는 데 사용하면 된다.

참고 자료 :

s-space.snu.ac.kr/bitstream/10371/100253/1/10%EA%B3%A0%EA%B8%B8%EA%B3%A4%ED%83%81%ED%98%84%EC%9A%B0.pdf

'ML,DL' 카테고리의 다른 글

[ML/DL] 데이터 인코딩 - Label Encoding / One-hot Encoding/ dummies

[ML/DL] 파이썬(python)을 이용한 분류(Classification)하기

[ML/DL] 대체법의 종류와 다중 대체법 사용법

[ML/DL] 머신러닝에 대한 간단한 개념들과 사용 하는 주요 패키지

[ML/DL]결측치의 종류와 결측치 처리 가이드라인

[ML/DL] 머신러닝(Machine larning)과 딥러닝(Deep larning)의 정의와 차이점