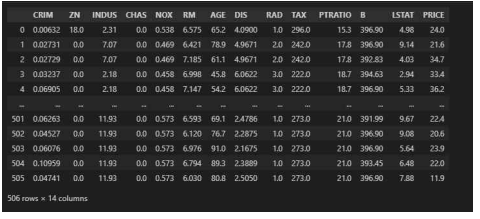
Data Analysis:
missing data simulation

ESC 2024 Summer Special Session 5차
발제자: 전제훈

1
Dataset

Dataset

Boston housing price dataset



	CRIM	ZN	INDUS	CHAS	NOX	RM	AGE	DIS	RAD	TAX	PTRATIO	B	LSTAT	PRICE
0	0.00632	18.0	2.31	0.0	0.538	6.575	65.2	4.0900	1.0	296.0	15.3	396.90	4.98	24.0
1	0.02731	0.0	7.07	0.0	0.469	6.421	78.9	4.9671	2.0	242.0	17.8	396.90	9.14	21.6
2	0.02729	0.0	7.07	0.0	0.469	7.185	61.1	4.9671	2.0	242.0	17.8	392.83	4.03	34.7
3	0.03237	0.0	2.18	0.0	0.458	6.998	45.8	6.0622	3.0	222.0	18.7	394.63	2.94	33.4
4	0.06905	0.0	2.18	0.0	0.458	7.147	54.2	6.0622	3.0	222.0	18.7	396.90	5.33	36.2
...	...	...	...	...	...	...	...	...	...	...	...	...	...	...
501	0.06263	0.0	11.93	0.0	0.573	6.593	69.1	2.4786	1.0	273.0	21.0	391.99	9.67	22.4
502	0.04527	0.0	11.93	0.0	0.573	6.120	76.7	2.2875	1.0	273.0	21.0	396.90	9.08	20.6
503	0.06076	0.0	11.93	0.0	0.573	6.976	91.0	2.1675	1.0	273.0	21.0	396.90	5.64	23.9
504	0.10959	0.0	11.93	0.0	0.573	6.794	89.3	2.3889	1.0	273.0	21.0	393.45	6.48	22.0
505	0.04741	0.0	11.93	0.0	0.573	6.030	80.8	2.5050	1.0	273.0	21.0	396.90	7.88	11.9

506 rows × 14 columns

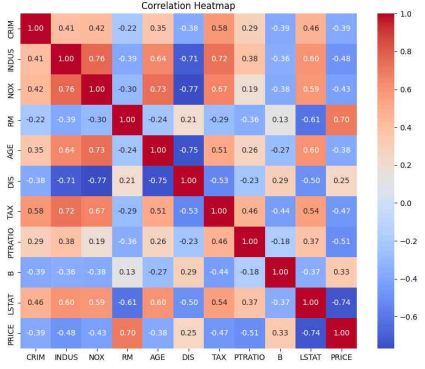
보스턴 주택 가격 데이터셋은 13개의 독립 변수로 PRICE 데이터를 예측하는 데이터셋입니다.
총 506개의 샘플이 존재합니다.

CRIM: 1인당 범죄율
ZN: 25,000 평방피트 이상의 주거 구역의 비율
INDUS: 비소매 상업지역 면적 비율
CHAS: 찰스강 인정 여부
NOX: 일산화질소 농도
RM: 주택당 평균 방의 수
AGE: 1940년 이전에 건축된 소유 주택 비율
DIS: 보스턴 고용 센터까지의 가중 거리
RAD: 방사형 고속도로 접근성 지수

TAX: 10,000 달러당 부동산 세율
PTRATIO: 학생/교사 비율
B: 흑인 비율
LSTAT: 하위 계층 비율

Dataset

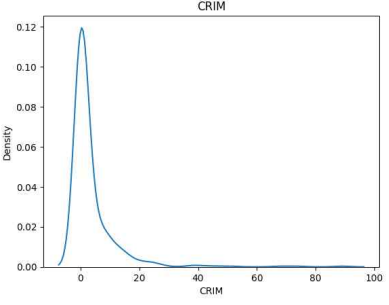
Boston housing dataset



오늘은 결측치 처리하는 것이 메인이기에,
범주형 데이터같은 경우 전처리하기가 좀 그래서
범주형 변수인 RAD, CHAS, ZN은 제외하고 진행하였습니다.

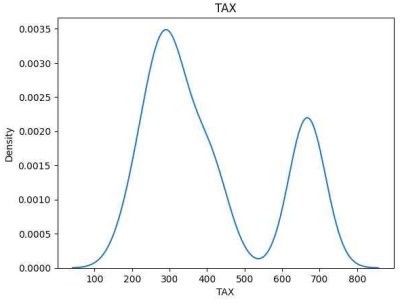
다음과 같이 히트맵을 그릴 수 있으며,
대체적으로 어느정도의 관계성을 파악할 수 있습니다.

Dataset



CRIM 변수는 다음과 같이 왜도가 있는 데이터입니다.
long-tail distribution같은 경우 끝부분을 지우거나,
랜덤하게 지우게 되었을 때 얼마나 정확하고 잘 예측하는지
보고자 합니다.

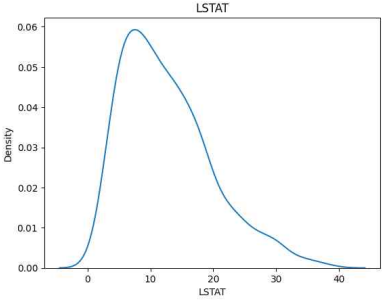
Dataset



TAX 변수는 이중 모달을 갖고 있는 데이터입니다.
끝부분을 지우게 되면,
이중 모달의 형태는 갖겠지만 오른쪽 모달의 영향이 줄게 됩니다.

과연 imputation 방법은 이를 적절하게 캐치하고 복원할지
확인하는 것이 중요합니다.

Dataset



LSTA는 약간의 왜도를 갖고 있지만,
얼핏 가장 정규분포 형태를 갖고 있는 형태입니다.

정규분포 형태를 갖고 있는 데이터의 경우 오른쪽 끝단을
자른다고해도 적절하게 데이터를 복원할 것이라고 예상이 됩니다.

이에 대해 잘 복원하는지 보는 것이 중요합니다.

2
Imputation

Preprocessing

```
boston_mnar = boston_df.copy()
threshold_crim = boston_mnar['CRIM'].quantile(0.75)
threshold_tax = boston_mnar['TAX'].quantile(0.7)
threshold_lstat = boston_mnar['LSTAT'].quantile(0.75)

boston_mnar.loc[boston_mnar['CRIM'] > threshold_crim, 'CRIM'] = np.nan
boston_mnar.loc[boston_mnar['TAX'] > threshold_tax, 'TAX'] = np.nan
boston_mnar.loc[boston_mnar['LSTAT'] > threshold_lstat, 'LSTAT'] = np.nan

missing_count_crim = np.sum(boston_df['CRIM'] > threshold_crim)
missing_count_tax = np.sum(boston_df['TAX'] > threshold_tax)
missing_count_lstat = np.sum(boston_df['LSTAT'] > threshold_lstat)

print('crim missing count')
print(missing_count_crim)
print('tax missing count')
print(missing_count_tax)
print('lstat missing count')
print(missing_count_lstat)
```

```
crim missing count
127
tax missing count
138
lstat missing count
127
```

MCAR과 MNAR에 대해서 실습할 것인데,
CRIM, TAX, LSTAT각 변수에 대해서

범죄율이 일정 이상이면 숨기다던지
세금이 일정 수준 이상이면 숨긴다던지
하위계측이 일정 수준 이상이면 숨긴다던지

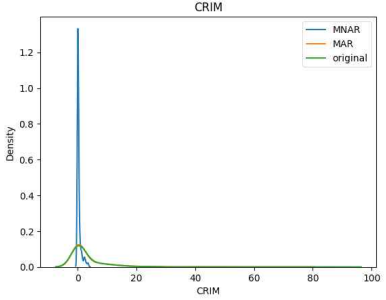
특정 상황을 가정하여 결측치를 생성해주었습니다.
결측 개수는 대강 120~130정도 나오도록 설정해주었습니다.

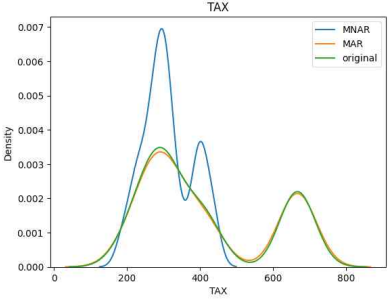
Preprocessing

```
import random
boston_mar = boston_df.copy()
random_sample = random.sample(range(506), missing_count_crim)
boston_mar['CRIM'][random_sample] = np.nan
random_sample = random.sample(range(506), missing_count_tax)
boston_mar['TAX'][random_sample] = np.nan
random_sample = random.sample(range(506), missing_count_lstat)
boston_mar['LSTAT'][random_sample] = np.nan
```

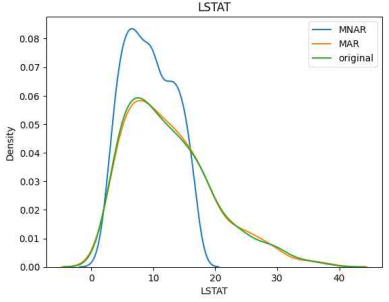
MCAR에서는 MNAR과 동일한 개수의 결측치를
랜덤하게 넣어 주었습니다.

Preprocessing





Preprocessing



확실하게 랜덤하지 않게 결측치처리한 것이
원본 데이터와는 다른 분포를 띄는 것을 알 수 있습니다.

랜덤하게 결측치처리한 것이 실제 분포와 형태가 비슷한걸보니
꽤 예쁘게 결측된 것을 알 수 있습니다.

Mean Imputation

```python
import pandas as pd
from sklearn.datasets import fetch_openml
from sklearn.model_selection import train_test_split
from sklearn.linear_model import LinearRegression
from sklearn.metrics import mean_squared_error, r2_score


X = boston_df.drop(columns=['PRICE'])
y = boston_df['PRICE']

X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.3)

lr_model = LinearRegression()
lr_model.fit(X_train, y_train)


y_pred = lr_model.predict(X_test)


mse = mean_squared_error(y_test, y_pred)
r2 = r2_score(y_test, y_pred)

print(f"Mean Squared Error (MSE): {mse:.4f}")
print(f"R-squared (R2): {r2:.4f}")
```

```
Mean Squared Error (MSE): 26.2257
R-squared (R2): 0.6050
```

결측이 1개도 없을 때
선형회귀분석을 통해 낸 예측 결과입니다.

MSE 값이 26.2257정도 나옵니다.
PRICE값이 20~40정도 나오는걸 감안하였을 때,
그리 좋은 성능은 아니지만

결측을 비교하는게 우선이므로 이걸 기준으로 결측치 방법에 대해
평가해봅시다

Mean Imputation

```
#1527개 test data
df_train = boston_mar[boston_mar.isnull().any(axis=1)]
df_test = boston_mar.dropna()
df_sampled = df_test.sample(n=55, random_state=42)
boston_mar_train = pd.concat([df_train,df_sampled])
boston_mar_test = boston_mar[~boston_mar.index.isin(boston_mar_train.index)]

df_train = boston_mnar[boston_mnar.isnull().any(axis=1)]
df_test = boston_mnar.dropna()
df_sampled = df_test.sample(n=55, random_state=42)
boston_mnar_train = pd.concat([df_train,df_sampled])
boston_mnar_test = boston_mnar[~boston_mnar.index.isin(boston_mnar_train.index)]
```

결측치를 처리하는 과정에서 테스트 데이터셋의 정보가 훈련 데이터에 흘러들어가면 안되기에,
이를 방지하고자 결측치가 없는 테스트 데이터셋과 결측치가 존재하는 훈련 데이터를 분류하여 주었습니다.

개수는 대충 원본데이터의 0.3 정도를 테스트 데이터셋으로 하였습니다.

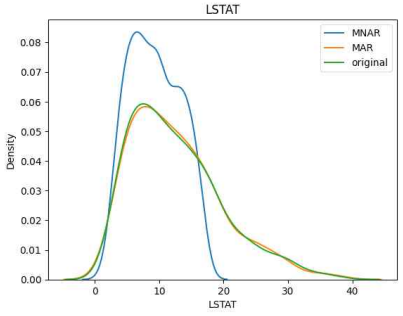
Mean Imputation

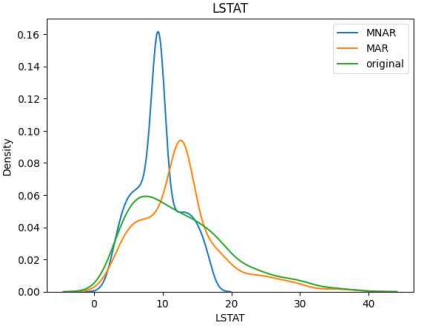
```
boston_mar_train_mean = boston_mar_train.copy()
boston_mar_train_mean['CRIM'].fillna(boston_mar_train['CRIM'].mean(), inplace=True)
boston_mar_train_mean['TAX'].fillna(boston_mar_train['TAX'].mean(), inplace=True)
boston_mar_train_mean['LSTAT'].fillna(boston_mar_train['LSTAT'].mean(), inplace=True)

boston_mar_train_mean

boston_mnar_train_mean = boston_mnar_train.copy()
boston_mnar_train_mean['CRIM'].fillna(boston_mnar_train['CRIM'].mean(), inplace=True)
boston_mnar_train_mean['TAX'].fillna(boston_mnar_train['TAX'].mean(), inplace=True)
boston_mnar_train_mean['LSTAT'].fillna(boston_mnar_train['LSTAT'].mean(), inplace=True)
```

처음에는 간단하게 mean imputation을 해봅시다.

결측치가 존재하는 훈련 데이터셋에 대해서만
훈련 데이터들의 평균값을 결측치 대체값으로 하였습니다.

Mean Imputation

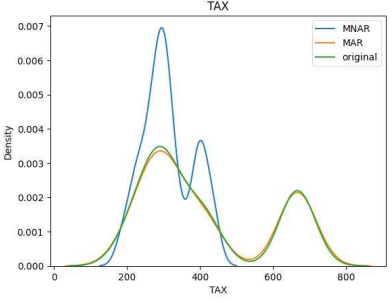


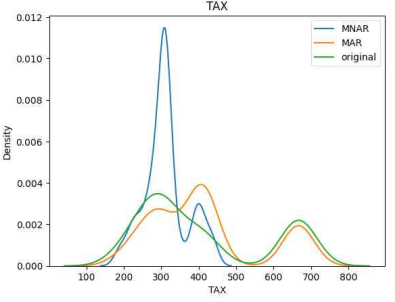


MAR에서는 mean imputation이 잘 작동될줄 알았는데, 원본데이터가 정규분포 형태를 띄지 않아서 그런가, MAR에서 평균치가 과하게 데이터가 많아져서 원본 데이터와 더 잘 안맞는 형태를 보입니다.

MNAR은 확실히 더 이상해졌네요

Mean Imputation





이중모달인 상황에서도 MAR인 경우에는 확실히 더 망가지는 것을 볼 수 있습니다

MNAR은 확실히 더 이상해졌네요

Mean Imputation

```
MAR_Mean imputation result
Mean Squared Error (MSE): 24.0837
R-squared (R2): 0.6652
```

```
MNAR_Mean imputation result
Mean Squared Error (MSE): 22.9254
R-squared (R2): 0.6786
```

?????

이럴수가 원본데이터가 많이 훼손되어서 성능이 더 잘 안나올줄 알았는데,

오히려 많이 훼손된 MNAR의 상황에서 더 잘 예측하게 되었습니다.

이에 대해서는 좀 살펴봅시다.

Mean Imputation

```
Feature Importance (coefficients):
RM           3.841601
AGE          0.011052
B            0.009455
TAX          0.006348
INDUS       -0.052465
CRIM        -0.114809
LSTAT       -0.598483
PTRATIO     -1.040382
DIS         -1.196383
NOX        -16.325897
dtype: float64
```

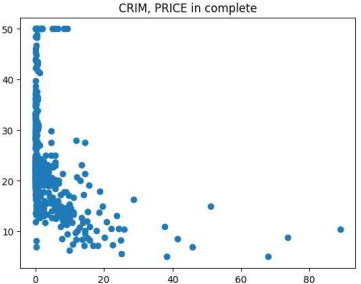
```
Feature Importance (coefficients):
RM           3.006110
CRIM         0.556499
B            0.014362
TAX          0.002122
INDUS       -0.051156
AGE         -0.065137
PTRATIO     -0.647085
LSTAT       -0.815247
DIS         -1.338424
NOX        -19.512882
dtype: float64
```

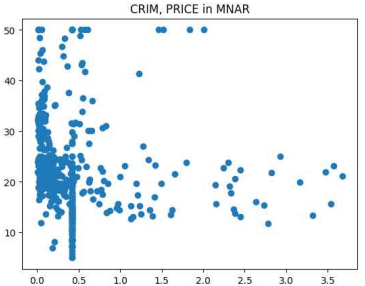
좌측이 결측치가 없는 데이터의 결과이고,
우측이 MNAR의 상황에서의 결과입니다.

결측을 채운 값은 모델 예측에는 거의 안쓰여서 거의 성능차이가 없었다는 것을알 수 있습니다.
거의 RM이 캐리하였다 이렇게 해석이 가능할 것 같습니다.

다만, CRIM같은 경우에는 더 좋은 성능을 보여줬는데
왜 그런지 봐야겠네요

Mean Imputation



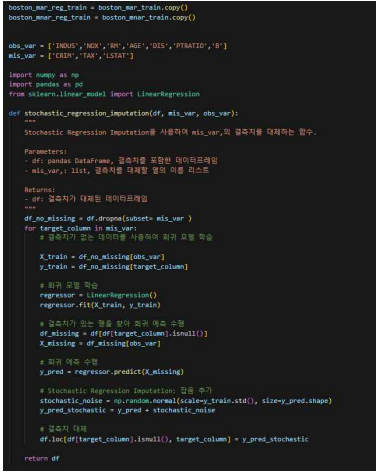


오른쪽이 MNAR의 상황에서 결측을 채운 형태입니다.

complete한 상황에서 보았을 때, CRIM이 20이상이 넘어가버리는 값이 존재했습니다.

100개 정도만 지워도 3.5정도 내에서 나오는데 말이죠. 어떻게 보면, 너무 큰 값을 제외해버린게 오히려 이상치제거가 되어버렸네요

Stochastic Regression Imputation



```
boston_mar_reg_train = boston_mar_train.copy()
boston_mnar_reg_train = boston_mnar_train.copy()


obs_var = ['INDUS','NOX','RM','AGE','DIS','PTRATIO','B']
mis_var = ['CRIM','TAX','LSTAT']

import numpy as np
import pandas as pd
from sklearn.linear_model import LinearRegression

def stochastic_regression_imputation(df, mis_var, obs_var):
    """
    Stochastic Regression Imputation을 사용하여 mis_var,의 결측치를 대체하는 함수.

    Parameters:
    - df: pandas DataFrame, 결측치를 포함한 데이터프레임
    - mis_var,: list, 결측치를 대체할 열의 이름 리스트

    Returns:
    - df: 결측치가 대체된 데이터프레임
    """
    df_no_missing = df.dropna(subset= mis_var )
    for target_column in mis_var:
        # 결측치가 없는 데이터를 사용하여 회귀 모델 학습

        X_train = df_no_missing[obs_var]
        y_train = df_no_missing[target_column]

        # 회귀 모델 학습
        regressor = LinearRegression()
        regressor.fit(X_train, y_train)

        # 결측치가 있는 행을 찾아 회귀 예측 수행
        df_missing = df[df[target_column].isnull()]
        X_missing = df_missing[obs_var]

        # 회귀 예측 수행
        y_pred = regressor.predict(X_missing)

        # Stochastic Regression Imputation: 잡음 추가
        stochastic_noise = np.random.normal(scale=y_train.std(), size=y_pred.shape)
        y_pred_stochastic = y_pred + stochastic_noise

        # 결측치 대체
        df.loc[df[target_column].isnull(), target_column] = y_pred_stochastic

    return df
```

Stochastic Regression Imputation을 통해 결측치를 채워주었습니다.

다음 알고리즘에 대해 살짝쿵 설명을 해보자면,
만약 A라는 변수가 결측이 존재한다면,
결측을 제외한 나머지 데이터에 대해서 B, C, D, ... 변수를 이용해
A라는 변수를 예측하는 회귀 모델을 세웁니다.

이것을 바탕으로 A 결측을 채워주는 방식으로 합니다.

Stochastic Regression Imputation

```
MNAR regression result
Mean Squared Error (MSE): 27.5121
R-squared (R2): 0.6142
```

```
MAR regression result
Mean Squared Error (MSE): 23.5143
R-squared (R2): 0.6732
```

MAR의 방식은 원래 complete 데이터보다 성능이 더 좋게 나왔으며,
MNAR에서는 완전 잘 안나오게 되었네요

MAR에서는 왜 잘 나왔는지 봐야겠네요

Stochastic Regression Imputation

```
Feature Importance (coefficients):
RM           3.841601
AGE          0.011052
B            0.009455
TAX          0.006348
INDUS       -0.052465
CRIM        -0.114809
LSTAT       -0.598483
PTRATIO     -1.040382
DIS         -1.196383
NOX        -16.325897
dtype: float64
```

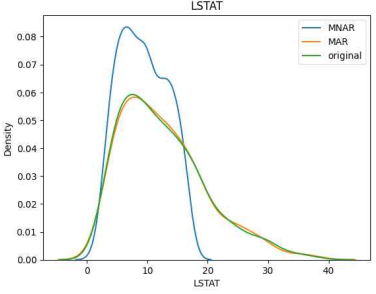
```
Feature Importance (coefficients):
RM           6.459912
INDUS        0.022252
B            0.016495
TAX         -0.000619
AGE         -0.040749
CRIM        -0.072405
LSTAT       -0.206443
PTRATIO     -0.965582
DIS         -1.395578
NOX        -16.005395
dtype: float64
```

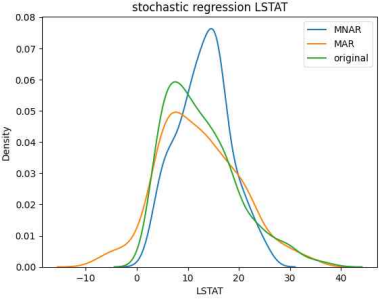
좌측이 결측치가 없는 데이터의 결과이고,
우측이 MAR의 상황에서의 결과입니다.

결측을 채운 변수들의 중요도가 낮아졌음을 알 수 있습니다.
이를 통해서 거의 원래 완벽한 데이터를 바탕으로 예측을 했다
라고 판단할 수 있으며,

Random forest, Xgboost 모델의 관점에서 생각을 해보면,
이를 방지했을테니 이는 회귀 모델의 고유 문제라고 볼 수 있겠네요

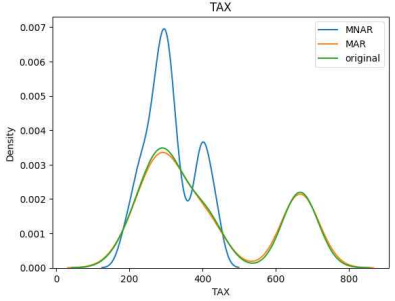
Stochastic Regression Imputation

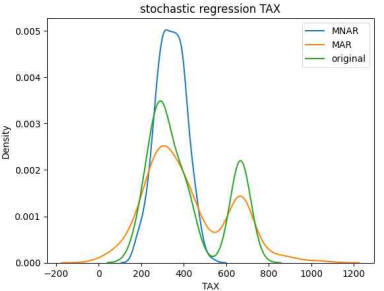




stochastic regression을 통해 결측치를 채운 것을 보았을 때, 정규분포가 아니라면, mean imputation보다 좀더 그럴듯하게 예측치를 채움을 알 수 있습니다.

Stochastic Regression Imputation

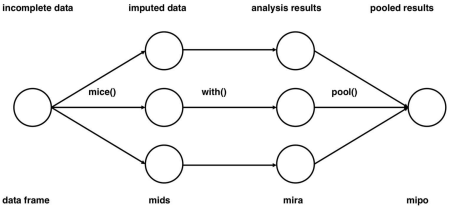




TAX에서는 시사하는 바가 많은 것 같습니다. 회귀 모델을 통해서 결측치를 채우기에,
좀 더 적절한 가운데 값으로 많이 채워져 분포가 다음과 같이 변하게 되었네요
MNAR의 경우에는 이중 모달이 아예 없어져 거의 정규분포의 형태를 띄게 되었네요

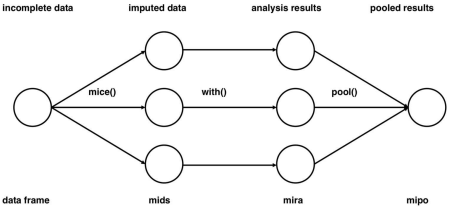
3
MICE

MICE



머신러닝 모델을 통한 결측치 채우는 것 중 좋은 성능을 보이는 그리고 가장 대중적인 MICE 방식에 대해 짚고 넘어가겠습니다.

MICE



MICE는 하나의 결측값을 단일 추정하는 게 아닌 여러 번의 대체를 통해 여러 개의 완전한 데이터셋을 생성합니다.
각 데이터셋에 대하여 각 결측 변수는 다른 변수들을 사용하여 대체됩니다.
이후 대체된 데이터를 통해 다른 데이터 결측을 대체합니다.

이를 수렴할때까지 무한히 반복하는 형태로 진행됩니다.

MICE

age	experience	salary(K)	Personal loan
25		50	1
27	3		1
29	5	80	0
31	7	90	0
33	9	100	1
	11	130	0

해당 결측 데이터가 존재한다고 가정을 해봅시다.
이는 완전하게 무작위로 결측이 되었는지
아니면, 랜덤하지 않게 결측되었는지는 잘 모릅니다만

MICE에서 다음과 같은 방식으로 진행됩니다.

MICE

age	experience	salary(K)
25	7	50
27	3	90
29	5	80
31	7	90
33	9	100
29	11	130

결측이 존재하는 데이터들은 그냥 일단 처음에는
적절한 값으로 채워집니다.

가령 적절한 대체로는 평균 대치 방법이 있겠네요
평균 대치방법으로 결측치를 채워줍니다.

보시면 알겠지만, 25세 경력 7년차가 봉급이
27세 경력 3년차보다 적은 것은 말이 안됩니다.

이로써 이는 적절한 결측치 대치 방법은 아니라고 알 수 있겠네요

MICE

age	experience	salary(K)
25	7	50
27	3	90
29	5	80
31	7	90
33	9	100
	11	130

age	experience	salary(K)
25	7	50
27	3	90
29	5	80
31	7	90
33	9	100
	11	130

age변수의 결측을 이제 나머지 결측은 채워졌다는 가정하에서 예측을 하게 됩니다.

아래 그림과 같이 그럼 age 결측 하나만 결측치이고 나머지는 완벽하게 측정되었다고 보는 것이죠

이에 예측 방식은 개인의 판단이지만, 보편적으로 회귀분석을 통해 예측을 진행하게 됩니다.

MICE

age	experience	salary(K)
25		50
27	3	90
29	5	80
31	7	90
33	9	100
34.99	11	130

age	experience	salary(K)
25	0.98	50
27	3	
29	5	80
31	7	90
33	9	100
34.99	11	130

다음과 같이 34.99로 예측이 되었네요
이제 이 예측값을 이용하여 다른 결측치를 예측해봅시다.

두번째로 experience를 다시 결측치로 만들고 예측을 해보죠

그 결과 아래 그림과 같이 0.98로 나오게 됩니다.
오 이는 조금 적절해보입니다.
왜냐하면 25세면 경력도 적을거고,
경력이 적으면 봉급도 적을 것이고 적절해보이네요

MICE

age	experience	salary(K)
25		50
27	3	90
29	5	80
31	7	90
33	9	100
34.99	11	130

age	experience	salary(K)
25	0.98	50
27	3	
29	5	80
31	7	90
33	9	100
34.99	11	130

다음과 같이 34.99로 예측이 되었네요
이제 이 예측값을 이용하여 다른 결측치를 예측해봅시다.

두번째로 experience를 다시 결측치로 만들고 예측을 해보죠

그 결과 아래 그림과 같이 0.98로 나오게 됩니다.
오 이는 조금 적절해보입니다.
왜냐하면 25세면 경력도 적을거고,
경력이 적으면 봉급도 적을 것이고 적절해보이네요

해당 값의 변화량이 특정 범위 이하로 내려갈때까지
반복하는 것이 mice 방식입니다.

MICE

age	experience	salary(K)
25		50
27	3	90
29	5	80
31	7	90
33	9	100
34.99	11	130

age	experience	salary(K)
25	0.98	50
27	3	
29	5	80
31	7	90
33	9	100
34.99	11	130

다음과 같이 34.99로 예측이 되었네요
이제 이 예측값을 이용하여 다른 결측치를 예측해봅시다.

두번째로 experience를 다시 결측치로 만들고 예측을 해보죠

그 결과 아래 그림과 같이 0.98로 나오게 됩니다.
오 이는 조금 적절해보입니다.
왜냐하면 25세면 경력도 적을거고,
경력이 적으면 봉급도 적을 것이고 적절해보이네요

해당 값의 변화량이 특정 범위 이하로 내려갈때까지
반복하는 것이 mice 방식입니다.

MICE

```
from sklearn.experimental import enable_iterative_imputer
from sklearn.impute import IterativeImputer


boston_mar_mice_x_train = boston_mar_train.copy()
boston_mnar_mice_x_train = boston_mnar_train.copy()

var = boston_mnar_mice_x_train.columns.difference(['PRICE'])
imputer_mice_x_train = IterativeImputer(random_state=42)

numeric_data = imputer_mice_x_train.fit_transform(boston_mnar_mice_x_train[var])
boston_mnar_mice_x_train = pd.DataFrame(numeric_data, columns=var)
boston_mnar_mice_x_train

var = boston_mar_mice_x_train.columns.difference(['PRICE'])
imputer_mice_x_train = IterativeImputer(random_state=42)

numeric_data = imputer_mice_x_train.fit_transform(boston_mar_mice_x_train[var])
boston_mar_mice_x_train = pd.DataFrame(numeric_data, columns=var)
boston_mar_mice_x_train
```

sklearn에서 IterativeImpute를 이용하면
MICE 방식을 이용할 수 있습니다.

다음과 같이 종속변수 price를 제외한 나머지 변수를 이용하여
결측치를 채워줍니다.

MICE

```
MAR mice imputation
Mean Squared Error (MSE): 21.5052
R-squared (R2): 0.7011
```

```
MNAR mice imputation
Mean Squared Error (MSE): 24.5423
R-squared (R2): 0.6559
```

띠용 MAR의 경우 0.7까지 올라가버렸네요
MNAR의 경우에도 완벽한 데이터셋에서 돌린 회귀 결과보다
더 좋게 나옴을 알 수 있습니다.

이는 좀 둘 다 살펴보겠습니다.

MICE

```
MAR mice imputation
Mean Squared Error (MSE): 21.5052
R-squared (R2): 0.7011
Feature Importance (coefficients):
RM          5.578368
INDUS       0.047056
B           0.012757
TAX        -0.002194
CRIM       -0.004385
AGE        -0.021926
LSTAT      -0.448528
PTRATIO    -0.887753
DIS        -1.283138
NOX       -15.035229
dtype: float64
```

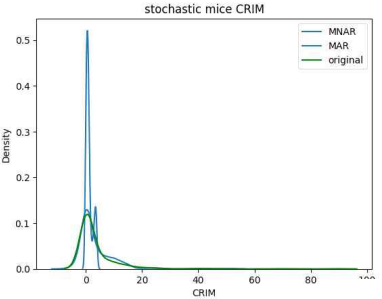
```
MNAR mice imputation
Mean Squared Error (MSE): 24.5423
R-squared (R2): 0.6559
Feature Importance (coefficients):
RM          2.371036
CRIM        0.615790
INDUS       0.047403
B           0.008477
TAX         0.003258
AGE        -0.009284
PTRATIO    -0.414778
DIS        -0.966937
LSTAT      -1.516457
NOX       -16.342448
dtype: float64
```

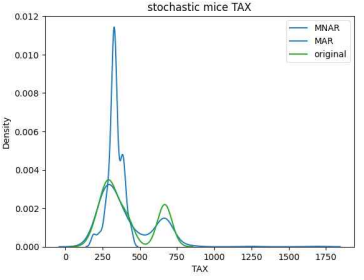
거의 RM이랑, NOX로 예측 하였다는 걸 알 수 있습니다.
이렇게 보면 예측 정확도는 더 높아졌을지라도,
결측치를 넣은게 좋은거였나??라고 생각이된다면
그렇게 좋지는 않은 듯 보입니다.

결측이 없는 데이터

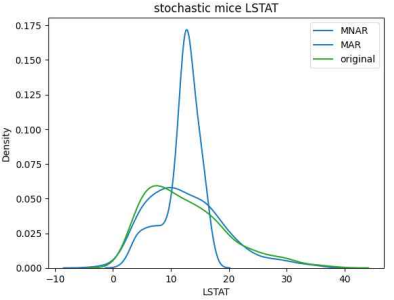
```
Feature Importance (coefficients):
RM          3.841601
AGE         0.011052
B           0.009455
TAX         0.006348
INDUS      -0.052465
CRIM       -0.114809
LSTAT      -0.598483
PTRATIO    -1.040382
DIS        -1.196383
NOX       -16.325897
dtype: float64
```

MICE





MICE 방식이 적절한 값을 주어 예측 정확도를 높였을지 몰라도
MNAR에서는 거의 제대로 대체를 못하는 것을 알 수 있습니다.
다만, MAR에서는 굉장히 잘 대체하는 것을 볼 수 있습니다.



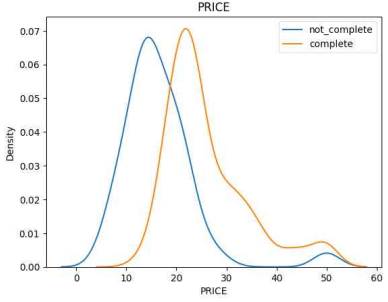
개인적인 생각

결측치가 존재하는 데이터셋을 채우니 오히려 완벽한 데이터보다 더 좋은 성능을 보여주는 것을 확인할 수 있었다.
이는 어찌보면 이해가 안되는 결과이지만,
앞선 EDA를 통해서 이에 대한 이유를 짐작할 수 있다.

상위값들을 제거하는 과정에서 과도하게 큰 이상치가 자연스럽게 제거되었기에 가능한 결과임을 알 수 있다.
완벽한 데이터에서는 이상치들로 인해 특정 변수들이 과도하게 종속변수와 관계를 갖게 되어
결과가 안좋게 나왔다는 것이다.

만약, 전처리가 잘 되어 잘 가공된 데이터라면 성능을 떨어졌을 것이다.
왜냐하면, 원래 데이터 분포를 제대로 못 쫓아갔기 때문이다. (ex. 이중모달, 특정 값 과도하게 분포)

개인적인 생각



결측치가 있는 데이터의 PRICE와
결측치가 없는 데이터의 PRICE 분포가 확연하게 차이가 있는 것을 알 수 있다.

또한, 만약에 결측이유가 어떠한 기준에 의해 의도적으로 가려졌다는 것을 알고 있다면,

이는 충분히 이중 모달을 예측할 수 있다는 것이다.
또한, MICE든 regression이든간에 결국에는 특정 범위내에 있는 데이터들 중에 나올 확률이 크다. 실제로 분포함수를 그려보았을 때 그러한 점을 확인할 수 있었다.

해당 정보를 잘 넣는 방식이 가장 좋은 결측치 대체방법이 될 것이다.

개인적인 생각

마명	장군마루 (JANGGUN MARU)			레이팅	30	마번	1412292
등급	국6(2023/12/04)	성별	수	산지	한국	조교사	조현수(11)
생년월일	2021/01/11	연령	3	모색	갈색	마주	이광수a
출전기간	2023/08/26 ~ 2024/08/25	외역자마		출전예정		생산자	우림영농조합법인
통산전적	11(1/2/0/0/0)			부마	피지	모마	시라이트
승률	단승 : 9.1 % 복승 : 27.3 % 연승 : 27.3 %						
특징	(머리) 이마가마중, 유성						
	(목) 좌/우갈기가마						
	(다리)						
	(몸통)						
낙인	(좌) (우) (기타)						
경주마등록	2023/03/17 ~			최초도입가	35,000천원(개별)	최근거래가	35,000천원(개별)
수득상금	36,850,000원			최근6회 수득상금	9,900,000원	최근3회 수득상금	9,900,000원

다음과 같은 데이터가 존재한다고 가정해봅시다.
레이팅의 변수같은 경우 결측치가 존재합니다.

레이팅 변수의 결측의 이유는
레이팅이 결정될만큼 판수가 많지 않아서입니다.

이러한 경우는 명백하게 MNAR 상황이기 때문에
MICE든 regression이든 그렇게 좋은 방법은 아니게 됩니다.

이에 대해서는 적절한 방법의 대치가 필요하다.

감사합니다