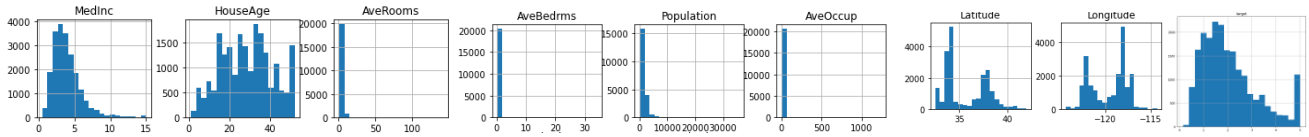
A[5] Feature Engineering

2016314364 박수헌

sklearn.datasets.fetch_california_housing 을 사용해 dataset을 load했다. 이 dataset은 8개의 실수형 feature를 갖고 있고, 실수 형태의 target을 갖고 있는 data이다. feature들의 값 분포를 쉽게 알아내기 위해 pandas dataframe으로 데이터를 변환했다. matplotlib을 이용해 히스토그램으로 위 feature 값들의 분포를 확인했다. 마지막은 target의 분포이다.

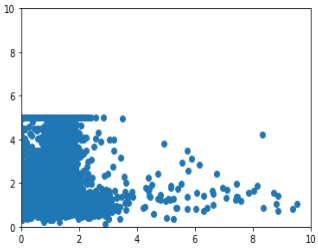


Feature engineering을 하기 전 standard scaler를 이용해 scaling한 후 linear regression model의 test set에 대한 RMSE 결과는 아래와 같았다.

```
RMSE(train) :  0.718343183373882
RMSE(test) :  0.7446113199726715
```

<1> Binning

PCA를 활용해 8개의 feature들을 하나의 feature로 만들어 도표로 확인해 보았다.



전혀 linear하지 않은 것을 알 수 있었다.

이 feature 값들의 최솟값은 -1422.42790617, 최댓값은 34256.3807251이었으며 이 값을 binning 해 linear regression model을 이용해 fit한 후 예측했지만 성능은 좋지 않았다.

```
RMSE(train) :  1.145531604832062
RMSE(test) :  1.1784368648034047
```

<2> Polynomial/interaction features

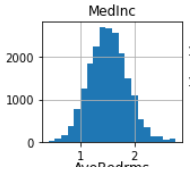
Polynomial을 사용해 feature 값들을 변형 한 후 standard scaler를 이용해 scaling해, linear regression model을 이용해 fit한 후 예측한 결과이다. Degree를 바꿔가며 실행했다.

Degree RMSE	1	2	3	4
Train set	0.718343183373882	0.6486018910649863	0.5884244946894744	0.5338593723900699
Test set	0.7446113199726717	0.6744074161307838	2.0305516147218863	39.56635476519788

Train set에 대한 RMSE 값은 degree가 2일 때 가장 낮았으므로 가장 성능이 좋았다고 볼 수 있다. 이는 기존 scaling만 한 결과인 RMSE(test) : 0.7446113199726715 보다도 좋은 성능을 보였다.

<3>Nonlinear Transformation

위의 각 feature들에 대한 히스토그램을 살펴봤을 때, MedInc의 그래프가 왼쪽으로 유독 치우쳐져 있는 즉, bell-shaped하지 않은 모습을 보이고 있다. 따라서 MedInc의 feature를 log(X+1) 함수를 이용해 bell-shape하게 수정한 후 MedInc의 그래프는 아래와 같았다.



이 feature값들로 standard scaler를 이용해 scaling한 후 linear regression model을 이용해 fit한 후 예측한 결과는 아래와 같았다.

```
RMSE(train) :  0.7460078060487457
RMSE(test) :  0.7642238162233127
```

Feature engineer하기 전의 결과보다 성능이 떨어졌다.

<4> Feature Selection

Selection Method RMSE	SelectKBest (k = 6)	RFE (n_features_to_select=6)	SelectFromModel
Train set	0.7194816893746349	0.7194816893746349	0.7395914931507552
Test set	0.7455460547716126	0.7455460547716126	0.7573975167442405

결과적으로 Polynomial을 이용해 degree를 2로 설정했을 때 가장 예측 성능이 좋았다.

```
from sklearn.datasets import fetch_california_housing
import pandas as pd
import matplotlib.pyplot as plt
%matplotlib inline

dataset = fetch_california_housing()
X = pd.DataFrame(dataset.data, columns=dataset.feature_names)
y = pd.DataFrame(dataset.target)
y = y.rename(columns={0: "target"})

X.hist(bins=20, figsize=(8,7))
y.hist(bins=20, figsize=(8,7))

from sklearn.model_selection import train_test_split
from sklearn.datasets import fetch_california_housing
from sklearn.preprocessing import StandardScaler
from sklearn.linear_model import LinearRegression
from sklearn.metrics import mean_squared_error

X, y = fetch_california_housing(return_X_y = True)
X_train, X_test, y_train, y_test = train_test_split(X,y,random_state = 1016)

scaler = StandardScaler()
scaler.fit(X_train)
X_train_sc = scaler.transform(X_train)
X_test_sc = scaler.transform(X_test)

lr = LinearRegression()
lr.fit(X_train_sc,y_train)
y_train_hat = lr.predict(X_train_sc)
y_test_hat = lr.predict(X_test_sc)
print("RMSE(train) : ", mean_squared_error(y_train,y_train_hat)**0.5)
print("RMSE(test) : ", mean_squared_error(y_test,y_test_hat)**0.5)
```

```
from sklearn.decomposition import PCA
from sklearn.preprocessing import OneHotEncoder
import numpy as np

X, y = fetch_california_housing(return_X_y = True)

pca = PCA(n_components=1)
pca.fit(X)
X_pca = pca.transform(X)

bins = np.linspace(-1423, 34257, 10)
which_bin = np.digitize(X_pca, bins=bins)

encoder = OneHotEncoder(sparse=False)
encoder.fit(which_bin)
X_binned=encoder.transform(which_bin)

X_train_binned, X_test_binned, y_train, y_test = train_test_split(X_binned,y,random_state=1016)

lr = LinearRegression()
lr.fit(X_train_binned,y_train)
y_train_hat = lr.predict(X_train_binned)
y_test_hat = lr.predict(X_test_binned)
print("RMSE(train) : ", mean_squared_error(y_train,y_train_hat)**0.5)
print("RMSE(test) : ", mean_squared_error(y_test,y_test_hat)**0.5)
```

```
from sklearn.preprocessing import PolynomialFeatures

poly = PolynomialFeatures(degree=2, include_bias=False)
poly.fit(X_train)
poly.get_feature_names()
X_train_poly = poly.transform(X_train)
X_test_poly = poly.transform(X_test)

scaler = StandardScaler()
scaler.fit(X_train_poly)
X_train_poly_sc = scaler.transform(X_train_poly)
X_test_poly_sc = scaler.transform(X_test_poly)

lr = LinearRegression()
lr.fit(X_train_poly_sc,y_train)
y_train_hat = lr.predict(X_train_poly_sc)
y_test_hat = lr.predict(X_test_poly_sc)
print("RMSE(train) : ", mean_squared_error(y_train,y_train_hat)**0.5)
print("RMSE(test) : ", mean_squared_error(y_test,y_test_hat)**0.5)
```

```
from sklearn.datasets import fetch_california_housing
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
%matplotlib inline

dataset = fetch_california_housing()
X = pd.DataFrame(dataset.data, columns=dataset.feature_names)
y = pd.DataFrame(dataset.target)
y = y.rename(columns={0: "target"})

for idx in range(len(X['MedInc'])):
    X['MedInc'][idx] = np.log(X['MedInc'][idx] + 1)

X.hist(bins=20, figsize=(8,7))

X_train, X_test, y_train, y_test = train_test_split(X,y,random_state=1016)

scaler = StandardScaler()
scaler.fit(X_train)
X_train_sc = scaler.transform(X_train)
X_test_sc = scaler.transform(X_test)

lr = LinearRegression()
lr.fit(X_train_sc,y_train)
y_train_hat = lr.predict(X_train_sc)
y_test_hat = lr.predict(X_test_sc)
print("RMSE(train) : ", mean_squared_error(y_train,y_train_hat)**0.5)
print("RMSE(test) : ", mean_squared_error(y_test,y_test_hat)**0.5)
```

```
from sklearn.datasets import fetch_california_housing
from sklearn.feature_selection import SelectKBest, f_regression

X,y = fetch_california_housing(return_X_y=True)
X_train, X_test, y_train, y_test = train_test_split(X,y,random_state=1016)

scaler = StandardScaler()
scaler.fit(X_train)
X_train_sc = scaler.transform(X_train)
X_test_sc = scaler.transform(X_test)

select = SelectKBest(f_regression, k = 6)
select.fit(X_train_sc, y_train)
X_train_selected = select.transform(X_train_sc)
X_test_selected = select.transform(X_test_sc)

lr = LinearRegression()
lr.fit(X_train_selected,y_train)
y_train_hat = lr.predict(X_train_selected)
y_test_hat = lr.predict(X_test_selected)
print("RMSE(train) : ", mean_squared_error(y_train,y_train_hat)**0.5)
print("RMSE(test) : ", mean_squared_error(y_test,y_test_hat)**0.5)

from sklearn.datasets import fetch_california_housing
from sklearn.feature_selection import RFE

X,y = fetch_california_housing(return_X_y=True)
X_train, X_test, y_train, y_test = train_test_split(X,y,random_state=1016)

scaler = StandardScaler()
scaler.fit(X_train)
X_train_sc = scaler.transform(X_train)
X_test_sc = scaler.transform(X_test)

estimator = LinearRegression()
```

```
select = RFE(estimator, n_features_to_select=6, step=1)
select.fit(X_train_sc, y_train)

X_train_selected = select.transform(X_train_sc)
X_test_selected = select.transform(X_test_sc)

lr = LinearRegression()
lr.fit(X_train_selected,y_train)
y_train_hat = lr.predict(X_train_selected)
y_test_hat = lr.predict(X_test_selected)
print("RMSE(train) : ", mean_squared_error(y_train,y_train_hat)**0.5)
print("RMSE(test) : ", mean_squared_error(y_test,y_test_hat)**0.5)

from sklearn.datasets import fetch_california_housing
from sklearn.feature_selection import SelectFromModel

X,y = fetch_california_housing(return_X_y=True)
X_train, X_test, y_train, y_test = train_test_split(X,y,random_state=1016)

scaler = StandardScaler()
scaler.fit(X_train)
X_train_sc = scaler.transform(X_train)
X_test_sc = scaler.transform(X_test)

fmodel = LinearRegression()
select = SelectFromModel(fmodel, threshold="mean")
select.fit(X_train_sc, y_train)

X_train_selected = select.transform(X_train_sc)
X_test_selected = select.transform(X_test_sc)

lr = LinearRegression()
lr.fit(X_train_selected,y_train)
y_train_hat = lr.predict(X_train_selected)
y_test_hat = lr.predict(X_test_selected)
```

```
print("RMSE(train) : ", mean_squared_error(y_train,y_train_hat)**0.5)
print("RMSE(test) : ", mean_squared_error(y_test,y_test_hat)**0.5)
```