머신러닝과 딥러닝

Report3

소프트웨어학과

2016312568 정희윤

In [3]:
```
#하이퍼 파라미터인 alpha를 조절하며 다양한 회귀식 도출이 가능하다.
ALPHA=1

rid=Ridge(alpha=ALPHA)
Model=rid.fit(X,Y)
print("coef")
print(Model.coef_)
print("intercept")
print(Model.intercept_)
```

```
coef
[ 4.36594382e-01  9.43739513e-03 -1.07132761e-01  6.44062485e-01
 -3.97034295e-06 -3.78635869e-03 -4.21299306e-01 -4.34484717e-01]
intercept
-36.93858523232896
```

In [4]:
```
#하이퍼 파라미터인 alpha를 조절하며 다양한 회귀식 도출이 가능하다.
ALPHA=1

las=Lasso(alpha=ALPHA)
Model=las.fit(X,Y)
print("coef")
print(Model.coef_)
print("intercept")
print(Model.intercept_)
```

```
coef
[ 1.45469232e-01  5.81496884e-03  0.00000000e+00 -0.00000000e+00
 -6.37292607e-06 -0.00000000e+00 -0.00000000e+00 -0.00000000e+00]
intercept
1.348041367341614
```

alpha값을 바꾸어보면 다음과같은 결과가 도출된다.

In [15]:
```
#하이퍼 파라미터인 alpha를 조절하며 다양한 회귀식 도출이 가능하다.
ALPHA=0.4

rid=Ridge(alpha=ALPHA)
Model=rid.fit(X,Y)
print("coef")
print(Model.coef_)
print("intercept")
print(Model.intercept_)
```

```
coef
[ 4.36653689e-01  9.43642501e-03 -1.07246256e-01  6.44664031e-01
 -3.97397027e-06 -3.78646894e-03 -4.21308370e-01 -4.34502157e-01]
intercept
-36.94058742628669
```

In [16]:
```
#하이퍼 파라미터인 alpha를 조절하며 다양한 회귀식 도출이 가능하다.
ALPHA=0.4

las=Lasso(alpha=ALPHA)
Model=las.fit(X,Y)
print("coef")
print(Model.coef_)
print("intercept")
print(Model.intercept_)
```

```
coef
[ 3.17582912e-01  1.32956795e-02  0.00000000e+00 -0.00000000e+00
  1.59947349e-05 -3.53581143e-04 -0.00000000e+00 -0.00000000e+00]
intercept
0.43680336890713267
```

위의 결과로 alpha값의 변함에 따라 Lasso회귀에서 0으로 된 회귀계수가 4개로 바뀜을 알 수 있다. 따라서 alpha값에 따라 회귀계수의 최적화 정도가 달라지며 회귀계수들의 크기들도 작아짐을 알 수 있다.

In [1]:
```
from sklearn.linear_model import LinearRegression
import pandas as pd
import numpy as np
from matplotlib import pyplot as plt
from sklearn.linear_model import Ridge, Lasso
```

In [2]:
```
from sklearn.datasets import fetch_california_housing

california = fetch_california_housing()
X=california.data
DF=pd.DataFrame(X,columns=california.feature_names)
Y=california.target
print(DF)
```

```
       MedInc  HouseAge  AveRooms  AveBedrms  Population  AveOccup  Latitude  \
0      8.3252      41.0  6.984127   1.023810       322.0  2.555556     37.88
1      8.3014      21.0  6.238137   0.971880      2401.0  2.109842     37.86
2      7.2574      52.0  8.288136   1.073446       496.0  2.802260     37.85
3      5.6431      52.0  5.817352   1.073059       558.0  2.547945     37.85
4      3.8462      52.0  6.281853   1.081081       565.0  2.181467     37.85
5      4.0368      52.0  4.761658   1.103627       413.0  2.139896     37.85
6      3.6591      52.0  4.931907   0.951362      1094.0  2.128405     37.84
7      3.1200      52.0  4.797527   1.061824      1157.0  1.788253     37.84
8      2.0804      42.0  4.294118   1.117647      1206.0  2.026891     37.84
9      3.6912      52.0  4.970588   0.990196      1551.0  2.172269     37.84
10     3.2031      52.0  5.477612   1.079602       910.0  2.263682     37.85
11     3.2705      52.0  4.772480   1.024523      1504.0  2.049046     37.85
12     3.0750      52.0  5.322650   1.012821      1098.0  2.346154     37.85
13     2.6736      52.0  4.000000   1.097701       345.0  1.982759     37.84
14     1.9167      52.0  4.262903   1.009677      1212.0  1.954839     37.85
15     2.1250      50.0  4.242424   1.071970       697.0  2.640152     37.85
16     2.7750      52.0  5.939577   1.048338       793.0  2.395770     37.85
17     2.1202      52.0  4.052805   0.966997       648.0  2.138614     37.85
18     1.9911      50.0  5.343675   1.085919       990.0  2.362768     37.84
```

In [3]:
```
import statsmodels.api as sm

#forward selection 함수 정의: 입력변수를 하나씩 추가하면서 최소 p-value가 기준값인 cutoff-value보다 큰 변수가 나올 때까지 반복한다.
#함수의 인자로 있는 cutoff 매개변수 값을 조절하면서 학습하면 된다.
def forward_selection(data,target,cutoff=0.05):
    initial_features = data.columns.tolist()
    best_features = []
    while(len(initial_features)>0):
        remaining_features = list(set(initial_features)-set(best_features))
        new_pval = pd.Series(index=remaining_features)
        for new_column in remaining_features:
            model = sm.OLS(target, sm.add_constant(data[best_features+[new_column]])).fit()
            new_pval[new_column] = model.pvalues[new_column]
        min_p_value = new_pval.min()
        if(min_p_value < cutoff):
            best_features.append(new_pval.idxmin())
        else:
            break
    return best_features
forwarddata=forward_selection(DF,Y,0.01)
print(forwarddata)
```

```
['MedInc', 'HouseAge', 'Latitude', 'Longitude', 'AveBedrms', 'AveRooms', 'AveOccup']
```

위는 California Housing의 데이터를 가져와서 forward data selection을 한 결과이다.

In [4]:
```
#backward elimination 함수 정의: full model에서 입력변수를 하나씩 제거하면서 남아 있는 모든 입력변수 중 최대 p-value가 cutoff-va
#함수의 인자로 있는 cutoff 매개변수 값을 조절하면서 학습하면 된다.
def backward_elimination(data, target, cutoff= 0.05):
    features = data.columns.tolist()
    while(len(features) > 0):
        features_with_constant = sm.add_constant(data[features])
        p_values = sm.OLS(target, features_with_constant).fit().pvalues[1:]
        max_p_value = p_values.max()
        if(max_p_value >= cutoff):
            excluded_feature=p_values.idxmax()
            features.remove(excluded_feature)
        else:
            break
    return features

backwarddata=backward_elimination(DF,Y,0.01)
print(backwarddata)
```

```
['MedInc', 'HouseAge', 'AveRooms', 'AveBedrms', 'AveOccup', 'Latitude', 'Longitude']
```

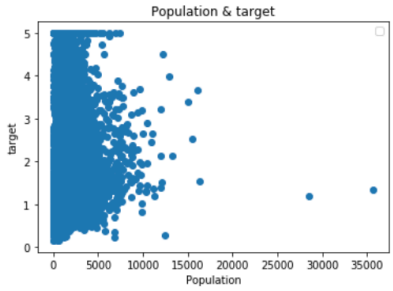
In [5]:
```
#stepwise selection 함수 정의: 위의 두 함수의 원리를 결합한 형태로 반복한다.
#함수의 인자로 있는 cutoff 매개변수 값을 조절하면서 학습하면 된다.
def stepwise_selection(data,target,cutoff):
    initial_features = data.columns.tolist()
    best_features = []
    while(len(initial_features) > 0):
        remaining_features = list(set(initial_features)-set(best_features))
        new_pval = pd.Series(index=remaining_features)
        for new_column in remaining_features:
            model = sm.OLS(target, sm.add_constant(data[best_features+[new_column]])).fit()
            new_pval[new_column] = model.pvalues[new_column]
        min_p_value = new_pval.min()
        if(min_p_value < cutoff):
            best_features.append(new_pval.idxmin())
            while(len(best_features) > 0):
                best_features_with_constant = sm.add_constant(data[best_features])
                p_values = sm.OLS(target, best_features_with_constant).fit().pvalues[1:]
                max_p_value = p_values.max()
                if(max_p_value >= cutoff):
                    excluded_feature=p_values.idxmax()
                    best_features.remove(excluded_feature)
                else:
                    break
        else:
            break
    return best_features

stepdata=stepwise_selection(DF,Y,0.01)
print(stepdata)
```

```
['MedInc', 'HouseAge', 'Latitude', 'Longitude', 'AveBedrms', 'AveRooms', 'AveOccup']
```

In [6]:
```
#i번째 feature와 타겟 값 사이의 관계 시각화
i=4
plt.title(california.feature_names[i]+' & ' + 'target')
plt.xlabel(california.feature_names[i])
plt.ylabel('target')
plt.scatter(DF[california.feature_names[i]],Y)
plt.legend()
plt.show()
```

```
No handles with labels found to put in legend.
```



위는 California Housing의 데이터로 Backward Elimination, Stepwise Selection한 결과이다.
또한, 마지막 결과를 통해 사라진 변수의 target 값 사이의 관계를 알 수 있다.