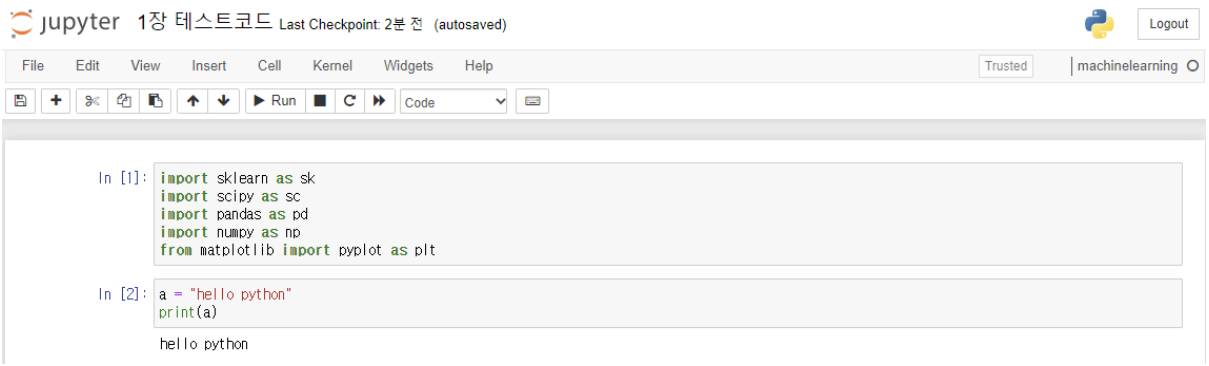
1주차 실습과제

2016314786 김호진

[1장 실습 - 테스트코드]



[2장 실습 - 1차선형회귀]

jupyter 2장_1차선형회귀 (autosaved) Logout

File Edit View Insert Cell Kernel Widgets Help Not Trusted machinelearning

Run Code

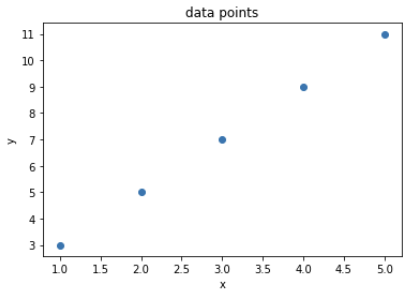
In [1]:
```
from sklearn.linear_model import LinearRegression
import pandas as pd
import numpy as np
from matplotlib import pyplot as plt
```

In [2]:
```
reg = LinearRegression()

#다른 샘플을 만들어 다양한 선형회귀식을 도출해볼 수 있다.
Xsample=[[1],[2],[3],[4],[5]]
Ysample=[[3],[5],[7],[9],[11]]

plt.title('data points')
plt.xlabel('x')
plt.ylabel('y')
plt.scatter(Xsample,Ysample)
```

Out[2]:
```
<matplotlib.collections.PathCollection at 0x27087307828>
```



In [3]:
```
Model=reg.fit(Xsample,Ysample)
print("coef")
print(Model.coef_)
print("intercept")
print(Model.intercept_)
```

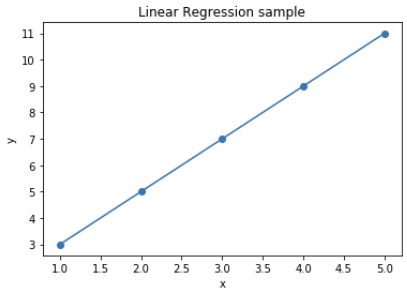
```
coef
[[2.]]
intercept
[1.]
```

In [4]:
```
Model.predict([[15]])
```

Out[4]:
```
array([[31.]])
```

In [5]:
```
plt.title('Linear Regression sample')
plt.xlabel('x')
plt.ylabel('y')
plt.scatter(Xsample,Ysample)
plt.plot(Xsample,Model.coef_*Xsample + Model.intercept_)
```

Out[5]:
```
[<matplotlib.lines.Line2D at 0x2708837f0f0>]
```



jupyter 2장_1차선형회귀 Last Checkpoint: 13분 전 (autosaved)

Logout

File Edit View Insert Cell Kernel Widgets Help

Not Trusted | machinelearning

Run Code

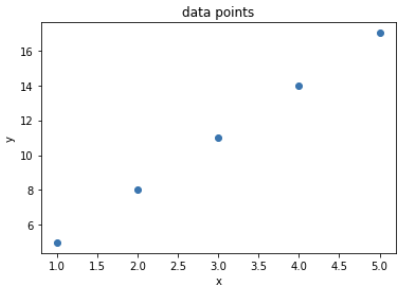
In [6]:
```
from sklearn.linear_model import LinearRegression
import pandas as pd
import numpy as np
from matplotlib import pyplot as plt
```

In [7]:
```
reg = LinearRegression()

Xsample=[[1],[2],[3],[4],[5]]
Ysample=[[5],[8],[11],[14],[17]]

plt.title('data points')
plt.xlabel('x')
plt.ylabel('y')
plt.scatter(Xsample,Ysample)
```

Out[7]:
```
<matplotlib.collections.PathCollection at 0x24797aa7640>
```



In [8]:
```
Model=reg.fit(Xsample,Ysample)
print("coef")
print(Model.coef_)
print("intercept")
print(Model.intercept_)
```

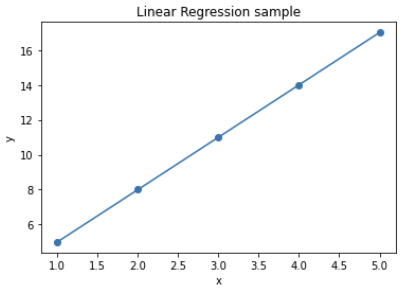
```
coef
[[3.]]
intercept
[2.]
```

In [9]:
```
Model.predict([[15]])
```

Out[9]:
```
array([[47.]])
```

In [10]:
```
plt.title('Linear Regression sample')
plt.xlabel('x')
plt.ylabel('y')
plt.scatter(Xsample,Ysample)
plt.plot(Xsample,Model.coef_*Xsample + Model.intercept_)
```

Out[10]:
```
[<matplotlib.lines.Line2D at 0x2479826652o>]
```



[2장 실습 – California housing 데이터 선형회귀]

jupyter 2장 California housing 데이터 선형회귀 Last Checkpoint: 8분 전 (unsaved changes) Logout

File Edit View Insert Cell Kernel Widgets Help Trusted machinelearning

Run Code

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

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)
```

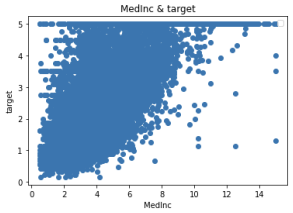
```
       MedInc  HouseAge  AveRooms  AveBedrms  Population  AveOccup  Latitude  \
0      8.3252      41.0  6.984127   1.023810       322.0  2.555556     37.88
1      8.3014      21.0  6.238137   0.971880      2401.0  2.109842     37.86
2      7.2574      52.0  8.288136   1.073446       496.0  2.802260     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
...       ...       ...       ...        ...         ...       ...       ...
20635  1.5603      25.0  5.045455   1.133333       845.0  2.560606     39.48
20636  2.5568      18.0  6.114035   1.315789       356.0  3.122807     39.49
20637  1.7000      17.0  5.205543   1.120092      1007.0  2.325635     39.43
20638  1.8672      18.0  5.329513   1.171920       741.0  2.123209     39.43
20639  2.3886      16.0  5.254717   1.162264      1387.0  2.616981     39.37

       Longitude
0        -122.23
1        -122.22
2        -122.24
3        -122.25
4        -122.25
...          ...
20635    -121.09
20636    -121.21
20637    -121.22
20638    -121.32
20639    -121.24

[20640 rows x 8 columns]
```

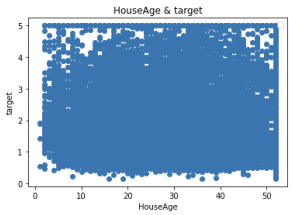
In [3]:
```
i=0
plt.title(california.feature_names[i]+' & ' + 'target')
plt.xlabel(california.feature_names[i])
plt.ylabel('target')
plt.scatter(DF[california.feature_names[i]],Y)
plt.legend()
plt.show()
```

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



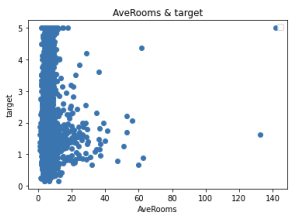
In [4]:
```
i=1
plt.title(california.feature_names[i]+' & ' + 'target')
plt.xlabel(california.feature_names[i])
plt.ylabel('target')
plt.scatter(DF[california.feature_names[i]],Y)
plt.legend()
plt.show()
```

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



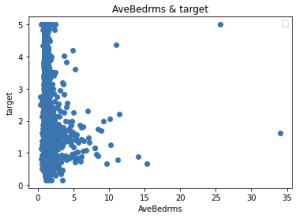
In [5]:
```
i=2
plt.title(california.feature_names[i]+' & ' + 'target')
plt.xlabel(california.feature_names[i])
plt.ylabel('target')
plt.scatter(DF[california.feature_names[i]],Y)
plt.legend()
plt.show()
```

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



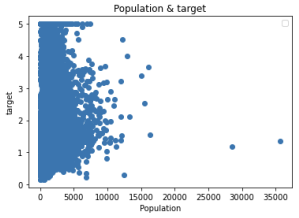
In [6]:
```
i=3
plt.title(california.feature_names[i]+' & ' + 'target')
plt.xlabel(california.feature_names[i])
plt.ylabel('target')
plt.scatter(DF[california.feature_names[i]],Y)
plt.legend()
plt.show()
```

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



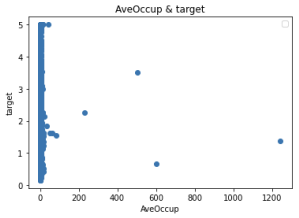
In [7]:
```
i=4
plt.title(california.feature_names[i]+' & ' + 'target')
plt.xlabel(california.feature_names[i])
plt.ylabel('target')
plt.scatter(DF[california.feature_names[i]],Y)
plt.legend()
plt.show()
```

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



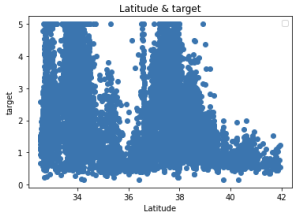
In [8]:
```
i=5
plt.title(california.feature_names[i]+' & ' + 'target')
plt.xlabel(california.feature_names[i])
plt.ylabel('target')
plt.scatter(DF[california.feature_names[i]],Y)
plt.legend()
plt.show()
```

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



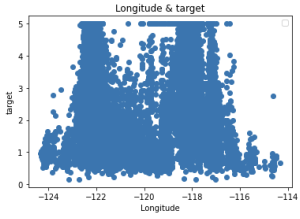
In [9]:
```
i=6
plt.title(california.feature_names[i]+' & ' + 'target')
plt.xlabel(california.feature_names[i])
plt.ylabel('target')
plt.scatter(DF[california.feature_names[i]],Y)
plt.legend()
plt.show()
```

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



In [10]:
```
i=7
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.
```

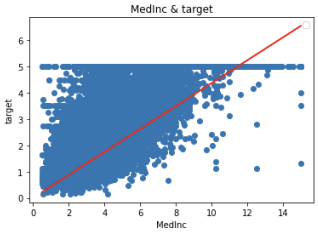


In [11]:
```
reg = LinearRegression()
Model=reg.fit(X,Y)
print("coef")
print(Model.coef_)
print("intercept")
print(Model.intercept_)
```

```
coef
[ 4.36693293e-01  9.43577803e-03 -1.07322041e-01  6.45065694e-01
 -3.97638942e-06 -3.78654265e-03 -4.21314378e-01 -4.34513755e-01]
intercept
-36.941920207184396
```

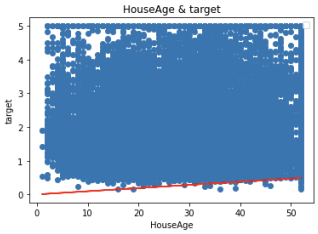
In [12]:
```
i=0
plt.title(california.feature_names[i]+' & ' + 'target')
plt.xlabel(california.feature_names[i])
plt.ylabel('target')
plt.scatter(DF[california.feature_names[i]],Y)
plt.plot(DF[california.feature_names[i]],Model.coef_[i]*DF[california.feature_names[i]],'r-')
plt.legend()
plt.show()
```

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



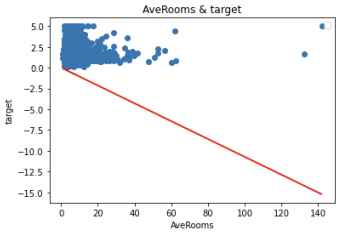
In [13]:
```
i=1
plt.title(california.feature_names[i]+' & ' + 'target')
plt.xlabel(california.feature_names[i])
plt.ylabel('target')
plt.scatter(DF[california.feature_names[i]],Y)
plt.plot(DF[california.feature_names[i]],Model.coef_[i]*DF[california.feature_names[i]],'r-')
plt.legend()
plt.show()
```

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



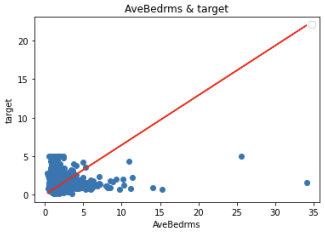
In [14]:
```
i=2
plt.title(california.feature_names[i]+' & ' + 'target')
plt.xlabel(california.feature_names[i])
plt.ylabel('target')
plt.scatter(DF[california.feature_names[i]],Y)
plt.plot(DF[california.feature_names[i]],Model.coef_[i]*DF[california.feature_names[i]],'r-')
plt.legend()
plt.show()
```

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



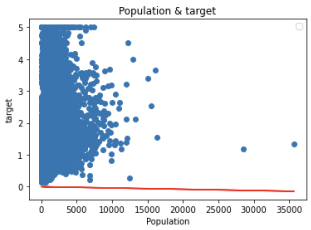
In [15]:
```
i=3
plt.title(california.feature_names[i]+' & ' + 'target')
plt.xlabel(california.feature_names[i])
plt.ylabel('target')
plt.scatter(DF[california.feature_names[i]],Y)
plt.plot(DF[california.feature_names[i]],Model.coef_[i]*DF[california.feature_names[i]],'r-')
plt.legend()
plt.show()
```

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



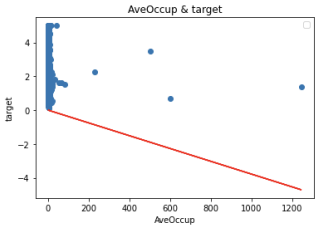
In [16]:
```
i=4
plt.title(california.feature_names[i]+' & ' + 'target')
plt.xlabel(california.feature_names[i])
plt.ylabel('target')
plt.scatter(DF[california.feature_names[i]],Y)
plt.plot(DF[california.feature_names[i]],Model.coef_[i]*DF[california.feature_names[i]],'r-')
plt.legend()
plt.show()
```

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



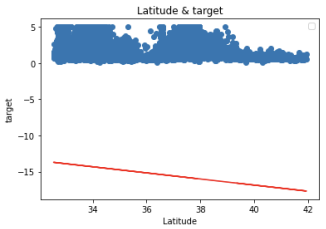
In [17]:
```
i=5
plt.title(california.feature_names[i]+' & ' + 'target')
plt.xlabel(california.feature_names[i])
plt.ylabel('target')
plt.scatter(DF[california.feature_names[i]],Y)
plt.plot(DF[california.feature_names[i]],Model.coef_[i]*DF[california.feature_names[i]],'r-')
plt.legend()
plt.show()
```

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



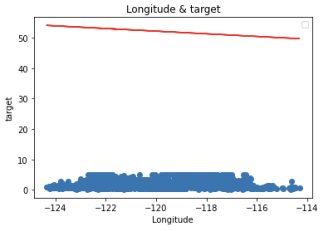
In [18]:
```
i=6
plt.title(california.feature_names[i]+' & ' + 'target')
plt.xlabel(california.feature_names[i])
plt.ylabel('target')
plt.scatter(DF[california.feature_names[i]],Y)
plt.plot(DF[california.feature_names[i]],Model.coef_[i]*DF[california.feature_names[i]],'r-')
plt.legend()
plt.show()
```

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



In [19]:
```
i=7
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.plot(DF[california.feature_names[i]],Model.coef_[i]*DF[california.feature_names[i]],'r-')
plt.legend()
plt.show()
```

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



In [20]:
```
DF.mean()
```

Out[20]:
```
MedInc          3.870671
HouseAge       28.639486
AveRooms        5.429000
AveBedrms       1.096675
Population    1425.476744
AveOccup         3.070655
Latitude        35.631861
Longitude     -119.569704
dtype: float64
```

In [21]:
```
Model.predict([DF.mean()])
```

Out[21]:
```
array([2.06855817])
```

[3장 실습 – Lasso, Ridge 정규화]

jupyter 3장 Lasso, Ridge 정규화 Last Checkpoint 8분 전 (unsaved changes)

Logout

File Edit View Insert Cell Kernel Widgets Help

Trusted machinelearning

Run Code

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
...       ...       ...       ...        ...         ...       ...       ...
20635  1.5603      25.0  5.045455   1.133333       845.0  2.560606     39.48
20636  2.5568      18.0  6.114035   1.315789       356.0  3.122807     39.49
20637  1.7000      17.0  5.205543   1.120092      1007.0  2.325635     39.43
20638  1.8672      18.0  5.329513   1.171920       741.0  2.123209     39.43
20639  2.3886      16.0  5.254717   1.162264      1387.0  2.616981     39.37

       Longitude
0        -122.23
1        -122.22
2        -122.24
3        -122.25
4        -122.25
...          ...
20635    -121.09
20636    -121.21
20637    -121.22
20638    -121.32
20639    -121.24

[20640 rows x 8 columns]
```

In [3]:
```
reg = LinearRegression()
Model = reg.fit(X,Y)
print("coef")
print(Model.coef_)
print("intercept")
print(Model.intercept_)
```

```
coef
[ 4.36693293e-01  9.43577803e-03 -1.07322041e-01  6.45065694e-01
 -3.97638942e-06 -3.78654265e-03 -4.21314378e-01 -4.34513755e-01]
intercept
-36.941920371184396
```

In [4]:
```
ALPHA=0.1

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

```
coef
[ 4.36683387e-01  9.43593989e-03 -1.07303989e-01  6.44965230e-01
 -3.97576456e-06 -3.78652421e-03 -4.21312678e-01 -4.34510859e-01]
intercept
-36.941587133609
```

In [5]:
```
ALPHA=0.5

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

```
coef
[ 4.36543795e-01  9.43659673e-03 -1.07227325e-01  6.44563694e-01
 -3.97336560e-06 -3.78645054e-03 -4.21306954e-01 -4.34492454e-01]
intercept
-36.94025397892854
```

In [6]:
```
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 -4.34494717e-01]
intercept
-36.938585232132937
```

In [7]:
```
ALPHA=2

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

```
coef
[ 4.36495800e-01  9.43901106e-03 -1.06944092e-01  6.43062429e-01
 -3.96430115e-06 -3.78617577e-03 -4.21284056e-01 -4.34455530e-01]
intercept
-36.93524021400957
```

In [8]:
```
ALPHA=0.1

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

```
coef
[ 3.90582557e-01  1.50821512e-02 -0.00000000e+00  0.00000000e+00
  1.75019561e-05 -3.32253135e-03 -1.14214430e-01 -9.92590699e-02]
intercept
-7.684569184737932
```

In [9]:
```
ALPHA=0.5

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

```
coef
[ 2.88854841e-01  1.20314561e-02  0.00000000e+00 -0.00000000e+00
  1.17610340e-05 -0.00000000e+00 -0.00000000e+00 -0.00000000e+00]
intercept
0.5891563081837052
```

In [10]:
```
ALPHA=1

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

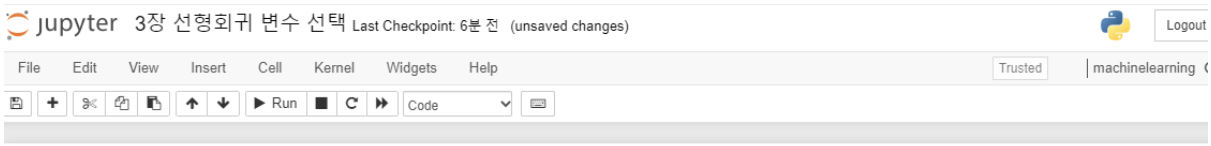
```
coef
[ 1.45469232e-01  5.81496984e-03  0.00000000e+00 -0.00000000e+00
 -6.37292507e-06 -0.00000000e+00 -0.00000000e+00 -0.00000000e+00]
intercept
1.3480413673416143
```

In [11]:
```
ALPHA=2

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

```
coef
[ 0.00000000e+00  0.00000000e+00  0.00000000e+00 -0.00000000e+00
 -2.35579621e-05 -0.00000000e+00 -0.00000000e+00 -0.00000000e+00]
intercept
2.1021394961624115
```

[3장 실습 – 선형회귀 변수 선택]



In [1]:
```
import warnings
warnings.filterwarnings(action='ignore')
```

In [2]:
```
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 [3]:
```
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
...       ...       ...       ...        ...         ...       ...       ...
20635  1.5603      25.0  5.045455   1.133333       845.0  2.560606     39.48
20636  2.5568      18.0  6.114035   1.315789       356.0  3.122807     39.49
20637  1.7000      17.0  5.205543   1.120092      1007.0  2.325635     39.43
20638  1.8672      18.0  5.329513   1.171920       741.0  2.123209     39.43
20639  2.3886      16.0  5.254717   1.162264      1387.0  2.616981     39.37

       Longitude
0        -122.23
1        -122.22
2        -122.24
3        -122.25
4        -122.25
...          ...
20635    -121.09
20636    -121.21
20637    -121.22
20638    -121.32
20639    -121.24

[20640 rows x 8 columns]
```

In [4]:
```
import statsmodels.api as sm

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
```

In [5]:
```
forwarddata=forward_selection(DF,Y,0.000000000000001)
print(forwarddata)
```

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

In [6]:
```
forwarddata=forward_selection(DF,Y,0.01)
print(forwarddata)
```

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

In [7]:
```
forwarddata=forward_selection(DF,Y,0.1)
print(forwarddata)
```

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

In [8]:
```
forwarddata=forward_selection(DF,Y,1)
print(forwarddata)
```

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

In [9]:
```
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
```

In [10]:
```
backwarddata=backward_elimination(DF,Y,0.000000000000001)
print(backwarddata)
```

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

In [11]:
```
backwarddata=backward_elimination(DF,Y,0.01)
print(backwarddata)
```

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

In [12]:
```
backwarddata=backward_elimination(DF,Y,0.1)
print(backwarddata)
```

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

In [13]:
```
backwarddata=backward_elimination(DF,Y,1)
print(backwarddata)
```

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

In [14]:
```
def stepwise_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())
            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
```

In [15]:
```
stepdata=stepwise_selection(DF,Y,0.000000000000001)
print(stepdata)
```

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

In [16]:
```
stepdata=stepwise_selection(DF,Y,0.01)
print(stepdata)
```

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

In [17]:
```
stepdata=stepwise_selection(DF,Y,0.1)
print(stepdata)
```

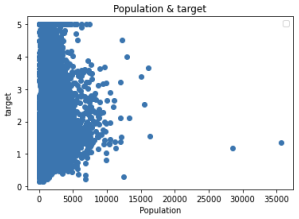
```
['MedInc', 'HouseAge', 'Latitude', 'Longitude', 'AveBedrms', 'AveRooms', 'AveOccup']
```

In [18]:
```
stepdata=stepwise_selection(DF,Y,1)
print(stepdata)
```

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

In [19]:
```
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.
```



In [20]:
```
i=5
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.
```