1주차 실습 과제

2017313107 이승태

1. 1장

(1) hello python 출력하기

```
[2]  import sklearn as sk
     import scipy as sc
     import pandas as pd
     import numpy as np
     from matplotlib import pyplot as plt

     a="hello python"
     print(a)

     hello python
```

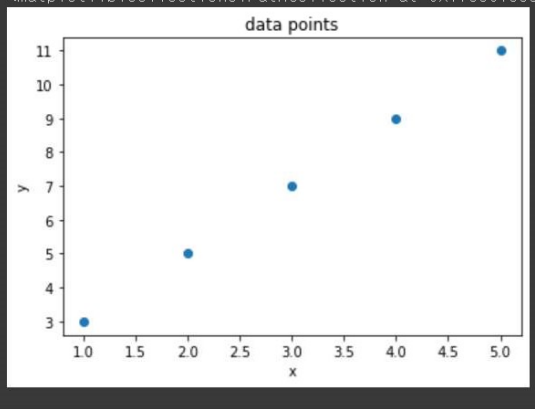
2. 2장 (간단한 1차 선형회귀모형)

(1) library import하기

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

(2) data들을 만든 뒤, 가시적으로 그래프를 그렸다.

```
[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)

     <matplotlib.collections.PathCollection at 0x7f8c0198dd50>
```



(3) linear regression을 이용하여 앞에 생성한 데이터를 이용하여 모델을 만들었다.
(Y = 2X + 1)

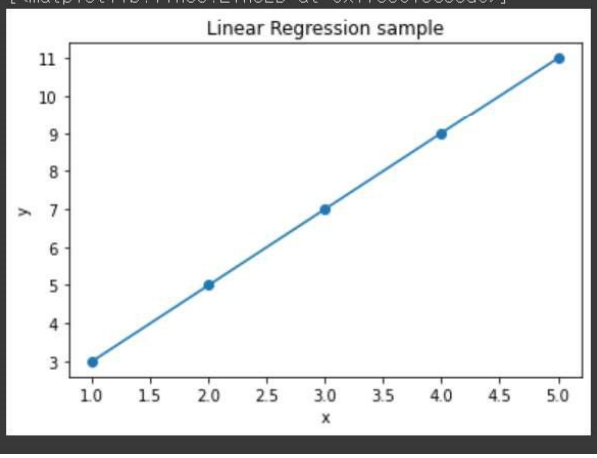
```
[3]  Model=reg.fit(Xsample,Ysample)
     print("coef")
     print(Model.coef_)
     print("intercept")
     print(Model.intercept_)

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

(4) 모델에 X=15를 넣고 값을 도출해내었다. (-> 31이 나옴)

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

     array([[31.]])
```

(5) Model과 데이터들을 가시적으로 나타내었다.

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

     [<matplotlib.lines.Line2D at 0x7f8c019533d0>]
```



3. 2장 (California housing 데이터셋에 대한 선형회귀모형)

(1) library import하기

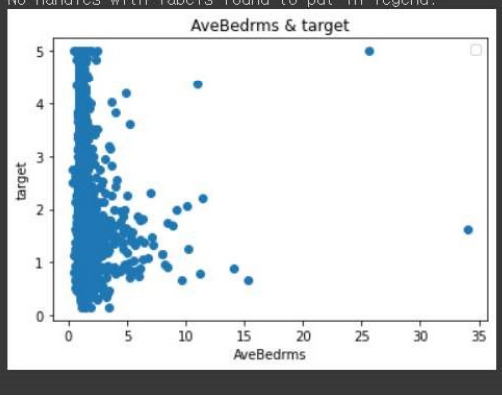
```
[1]  from sklearn.linear_model import LinearRegression
     import pandas as pd
     import numpy as np
     from matplotlib import pyplot as plt
```

(2) sklearn.datasets에서 california_housing 데이터를 가져왔다. 그 후 데이터를 출력했다.

```
[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)

     Downloading Cal. housing from https://ndownloader.figshare.com/files/5976036 to /root/scikit_learn_data
            MedInc  HouseAge  AveRooms  ...  AveOccup  Latitude  Longitude
     0      8.3252      41.0  6.984127  ...  2.555556     37.88    -122.23
     1      8.3014      21.0  6.238137  ...  2.109842     37.86    -122.22
     2      7.2574      52.0  8.288136  ...  2.802260     37.85    -122.24
     3      5.6431      52.0  5.817352  ...  2.547945     37.85    -122.25
     4      3.8462      52.0  6.281853  ...  2.181467     37.85    -122.25
     ...       ...       ...       ...  ...       ...       ...         ...
     20635  1.5603      25.0  5.045455  ...  2.560606     39.48    -121.09
     20636  2.5568      18.0  6.114035  ...  3.122807     39.49    -121.21
     20637  1.7000      17.0  5.205543  ...  2.325635     39.43    -121.22
     20638  1.8672      18.0  5.329513  ...  2.123209     39.43    -121.32
     20639  2.3886      16.0  5.254717  ...  2.616981     39.37    -121.24

     [20640 rows x 8 columns]
```

(3) average bedroom과 target값의 관계를 그래프로 그렸다.

```
[3]  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.
```



(4) linear regression으로 모델을 만들어서 기울기와 상수값을 확인해 보았다.

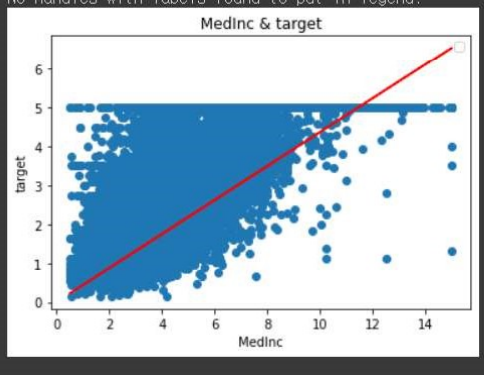
```
[4]  reg = LinearRegression()
     Model=reg.fit(X,Y)
     print("coef")
     print(Model.coef_)
     print("intercept")
     print(Model.intercept_)
```

```
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
-36.94192020718442
```

(5) medinc과 target값의 관계를 그래프로 그래고, 선형 모델을 그래프로 같이 넣어 비교해 보았다.

```
[5]  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.
```



(6) 모든 x값들의 평균을 구한 뒤, model에 넣고 값을 예측해보았다.

```
[6]  DF.mean()
```

```
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
```

```
[7]  Model.predict([DF.mean()])
```

```
array([2.06855817])
```

4. 3장 (Lasso, Ridge 정규화)

(1) library import하기

```
[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
```

(2) 3-(2)와 마찬가지로 sklearn.datasets에서 california_housing 데이터를 가져오고 데이터를 출력했다.

```
[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)

     Downloading Cal. housing from https://ndownloader.figshare.com/files/5976036 to /root/scikit_learn_data
            MedInc  HouseAge  AveRooms  ...  AveOccup  Latitude  Longitude
     0      8.3252      41.0  6.984127  ...  2.555556     37.88    -122.23
     1      8.3014      21.0  6.238137  ...  2.109842     37.86    -122.22
     2      7.2574      52.0  8.288136  ...  2.802260     37.85    -122.24
     3      5.6431      52.0  5.817352  ...  2.547945     37.85    -122.25
     4      3.8462      52.0  6.281853  ...  2.181467     37.85    -122.25
     ...       ...       ...       ...  ...       ...       ...         ...
     20635  1.5603      25.0  5.045455  ...  2.560606     39.48    -121.09
     20636  2.5568      18.0  6.114035  ...  3.122807     39.49    -121.21
     20637  1.7000      17.0  5.205543  ...  2.325635     39.43    -121.22
     20638  1.8672      18.0  5.329513  ...  2.123209     39.43    -121.32
     20639  2.3886      16.0  5.254717  ...  2.616981     39.37    -121.24

     [20640 rows x 8 columns]
```

(3) 경계값(ALPHA)을 지정하고, Ridge 정규화를 이용해 Model을 구성하였다.

- ALPHA = 1

```
[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
     -36.93858523232904
```

- ALPHA = 2

위와 비교했을 때, 대부분의 값들의 절댓값이 감소했다.

```
[5]  #하이퍼 파라미터인 alpha를 조절하며 다양한 회귀식 도출이 가능하다.
     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
-36.93524021400935
```

(4) 경계값(ALPHA)을 지정하고, Lasso 정규화를 이용해 Model을 구성하였다.
- ALPHA = 1

```
[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
1.3480413673416138
```

- ALPHA = 2
위와 비교했을 때 0이 된 값이 더 많아졌다.

```
[7]  #하이퍼 파라미터인 alpha를 조절하며 다양한 회귀식 도출이 가능하다.
     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
2.102139496162415
```

5. 3장 (선형회귀 변수 선택)

(1) library import하기

```
[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
```

(2) 3-(2)와 마찬가지로 sklearn.datasets에서 california_housing 데이터를 가져오고 데이터를 출력했다.

```
[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)

     Downloading Cal. housing from https://ndownloader.figshare.com/files/5976036 to /root/scikit_learn_data
            MedInc  HouseAge  AveRooms  ...  AveOccup  Latitude  Longitude
     0      8.3252      41.0  6.984127  ...  2.555556     37.88    -122.23
     1      8.3014      21.0  6.238137  ...  2.109842     37.86    -122.22
     2      7.2574      52.0  8.288136  ...  2.802260     37.85    -122.24
     3      5.6431      52.0  5.817352  ...  2.547945     37.85    -122.25
     4      3.8462      52.0  6.281853  ...  2.181467     37.85    -122.25
     ...       ...       ...       ...  ...       ...       ...        ...
     20635  1.5603      25.0  5.045455  ...  2.560606     39.48    -121.09
     20636  2.5568      18.0  6.114035  ...  3.122807     39.49    -121.21
     20637  1.7000      17.0  5.205543  ...  2.325635     39.43    -121.22
     20638  1.8672      18.0  5.329513  ...  2.123209     39.43    -121.32
     20639  2.3886      16.0  5.254717  ...  2.616981     39.37    -121.24

     [20640 rows x 8 columns]
```

(3) forward_selection을 이용하여 유용한 feature들을 골라내었다. (cutoff value = 0.01)
(-> population의 feature가 제거되었다.)

```
[8]  '''
     코드 실행 전에 아나콘다 프롬프트에서 'machinelearning' 가상환경을 활성화시킨 후 'conda install statsmodels'로
     통계량 계산 라이브러리(패키지)를 설치해야한다.

     '''
     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)

     /usr/local/lib/python3.7/dist-packages/ipykernel_launcher.py:15: DeprecationWarning: The default dtype for empty Series will be 'object' ins
       from ipykernel import kernelapp as app
     ['MedInc', 'HouseAge', 'Latitude', 'Longitude', 'AveBedrms', 'AveRooms', 'AveOccup']
```

(4) backward_elimination을 이용하여 유용한 feature들을 골라내었다. (cutoff value = 0.01)
(-> population의 feature가 제거되었다.)

```
[9]  #backward elimination 함수 정의: full model에서 입력변수를 하나씩 제거하면서 남아 있는 모든 입력변수 중 최대 p-value가 cutoff-value보다 낮아질 때까지 반복한다.
     #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']
```

(5) stepwise_selection을 이용하여 유용한 feature들을 골라내었다. (cutoff value = 0.01)
(-> population의 feature가 제거되었다.)

```
[10] #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)

     /usr/local/lib/python3.7/dist-packages/ipykernel_launcher.py:8: DeprecationWarning: The default

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

(6) forward_selection, backward_elimination, stepwise_selection을 이용하여 feature를 골라내었을 때, 삭제된 feature population에 대하여 그래프를 그려보았다. linear regression으로 상관관계를 알기 어려운 것을 확인할 수 있다.

```
[11] #1번째 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.
```