기계학습 과제1 보고서

16010980 이우석

1. 필요 라이브러리

```
[ ]  import numpy as np # 연산을 위한 numpy
     import pandas as pd # 데이터프레임을 위한 pandas

[ ]  from sklearn.datasets import load_iris # scikit-learn iris 데이터셋.
     from sklearn.model_selection import train_test_split # 데이터셋을 train 데이터와 test 데이터로 나눠주는 함수.
     from sklearn.neighbors import KNeighborsClassifier # KNN 분류 알고리즘을 사용.
     from sklearn.metrics import accuracy_score # 성능 비교를 위한 함수.
```

- 필요한 라이브러리와 함수들을 import.
- load_iris : 인공지능 모델을 학습시키기 위해 필요한 데이터셋.
- train_test_split : 데이터셋을 train 데이터와 test 데이터로 나누기 위해 필요한 함수.
- KNeighborsClassifier : KNN 분류 알고리즘.
- accuracy_score : 성능을 점수로 표현하기 위해 필요한 함수.

2. 데이터셋

```
dataset = load_iris() # iris 데이터셋을 불러오는 함수, 데이터셋 변수에 저장.

print(dataset.DESCR) # 해당 데이터셋에 대한 정보를 출력.
```

```
.. _iris_dataset:

Iris plants dataset
--------------------

**Data Set Characteristics:**

    :Number of Instances: 150 (50 in each of three classes)
    :Number of Attributes: 4 numeric, predictive attributes and the class
    :Attribute Information:
        - sepal length in cm
        - sepal width in cm
        - petal length in cm
        - petal width in cm
        - class:
                - Iris-Setosa
                - Iris-Versicolour
                - Iris-Virginica

    :Summary Statistics:

    ============== ==== ==== ======= ===== ====================
                    Min  Max   Mean    SD   Class Correlation
    ============== ==== ==== ======= ===== ====================
    sepal length:   4.3  7.9   5.84   0.83    0.7826
    sepal width:    2.0  4.4   3.05   0.43   -0.4194
    petal length:   1.0  6.9   3.76   1.76    0.9490  (high!)
    petal width:    0.1  2.5   1.20   0.76    0.9565  (high!)
    ============== ==== ==== ======= ===== ====================

    :Missing Attribute Values: None
    :Class Distribution: 33.3% for each of 3 classes.
    :Creator: R.A. Fisher
    :Donor: Michael Marshall (MARSHALL%PLU@io.arc.nasa.gov)
    :Date: July, 1988
```

- load_iris() : Iris 데이터 셋을 불러옴.
- dataset.DESCR : 해당 데이터셋에 대한 정보를 볼 수 있음.
- Number of Instances : 샘플의 개수.
- Number of Attributes : 열(column)의 개수.
- Attribute Information : 열(column)의 정보를 알 수 있음.
- Summary Statistics : 각 열에 대해서 샘플의 최소값, 최대값, 평균값 등을 알 수 있음.
- Missing Attribute Values: None : 값이 비어있는 샘플이 존재하는가를 알 수 있음. 사진에서는 'None' 이라고 되어 있으므로, 값이 비어있는 샘플이 없음을 알 수 있음.

3. 데이터셋을 데이터프레임 형태로

```
[ ]  x_data = pd.DataFrame(dataset.data, columns=dataset.feature_names) # 데이터셋의 데이터를 데이터프레임의 형태로 변환하여 저장.
     y_data = pd.DataFrame(dataset.target) # 데이터셋의 타겟을 데이터프레임의 형태로 변환하여 저장.

     # 데이터 출력.
     print(x_data)
     print(y_data)
```

```
     sepal length (cm)  sepal width (cm)  petal length (cm)  petal width (cm)
0                  5.1               3.5                1.4               0.2
1                  4.9               3.0                1.4               0.2
2                  4.7               3.2                1.3               0.2
3                  4.6               3.1                1.5               0.2
4                  5.0               3.6                1.4               0.2
..                 ...               ...                ...               ...
145                6.7               3.0                5.2               2.3
146                6.3               2.5                5.0               1.9
147                6.5               3.0                5.2               2.0
148                6.2               3.4                5.4               2.3
149                5.9               3.0                5.1               1.8

[150 rows x 4 columns]
     0
0    0
1    0
2    0
3    0
4    0
..  ..
145  2
146  2
147  2
148  2
149  2

[150 rows x 1 columns]
```

- pd.DataFrame() : 데이터프레임을 생성.
- 데이터셋을 이용하여 인공지능 모델을 학습시키기 전에 데이터셋을 데이터프레임의 형태로 변환하는 전처리 과정을 거침.

4. 데이터프레임 확인

```
[ ]  # x_data 의 행과 열의 크기를 출력.
     print(x_data.shape)

     (150, 4)

[ ]  # x_data 에서 샘플의 개수, 평균, 표준 편차, 최소값, 샘플 중 25%의 값, 50%의 값, 75%의 값, 최대값 출력.
     print(x_data.describe())

            sepal length (cm)  sepal width (cm)  petal length (cm)  \
     count         150.000000        150.000000         150.000000
     mean            5.843333          3.057333           3.758000
     std             0.828066          0.435866           1.765298
     min             4.300000          2.000000           1.000000
     25%             5.100000          2.800000           1.600000
     50%             5.800000          3.000000           4.350000
     75%             6.400000          3.300000           5.100000
     max             7.900000          4.400000           6.900000

            petal width (cm)
     count        150.000000
     mean           1.199333
     std            0.762238
     min            0.100000
     25%            0.300000
     50%            1.300000
     75%            1.800000
     max            2.500000
```

- x_data.shape : 현재 x_data 라는 데이터프레임의 행과 열의 크기를 알 수 있음.
- x_data.describe() : 현재 x_data 라는 데이터프레임의 종합적인 정보를 알 수 있음.
- 데이터프레임 형태로 잘 변환되었는지 확인.
- 현재 데이터들의 종합적인 정보를 확인.

5. 데이터셋을 분리

```
[ ]  # train, test 데이터를 8 : 2 로 나눔.
     x_train, x_test, y_train, y_test = train_test_split(x_data, y_data, test_size=0.2, random_state=22)

     # train 데이터 출력.
     print(x_train)
     print(y_train)
```

```
     sepal length (cm)  sepal width (cm)  petal length (cm)  petal width (cm)
30                 4.8               3.1                1.6               0.2
31                 5.4               3.4                1.5               0.4
89                 5.5               2.5                4.0               1.3
90                 5.5               2.6                4.4               1.2
55                 5.7               2.8                4.5               1.3
..                 ...               ...                ...               ...
102                7.1               3.0                5.9               2.1
100                6.3               3.3                6.0               2.5
44                 5.1               3.8                1.9               0.4
132                6.4               2.8                5.6               2.2
117                7.7               3.8                6.7               2.2

[120 rows x 4 columns]
     0
30   0
31   0
89   1
90   1
55   1
..  ..
102  2
100  2
44   0
132  2
117  2

[120 rows x 1 columns]
```

- train_test_split() : train 데이터와 test 데이터를 분리.
- train 데이터와 test 데이터를 8 : 2 로 분리하는 전처리 과정을 거침.
- 이러한 과정을 거침으로써, 인공지능 모델이 train 데이터에 과적합(Overfitting)이 되지는 않았는지 확인할 수 있음. 궁극적으로 과적합을 예방하기 위함이다.
- 데이터셋이 train 데이터와 test 데이터로 잘 나누어졌는지 출력을 통해 확인.

6. KNN 알고리즘을 사용하여 인공지능 모델을 학습

```
[ ]  # KNN 알고리즘을 사용, K = 3.
     estimator = KNeighborsClassifier(n_neighbors=3)

[ ]  # 인공지능 모델을 학습시킴.
     estimator.fit(x_train, y_train)

     /usr/local/lib/python3.7/dist-packages/sklearn/neighbors/_classification.py:198: DataConversionWarning: A column-
       return self._fit(X, y)
     KNeighborsClassifier(n_neighbors=3)
```

- KNeighborsClassifier() : KNN 알고리즘을 이용.
- fit() : 인공지능 모델을 적합 또는 학습시킴.
- 본격적으로 인공지능 모델을 학습시키는 부분.

7. 학습시킨 인공지능 모델을 통해 예측

```
[ ]  y_predict = estimator.predict(x_train) # 학습시킨 인공지능 모델에 x_train 데이터를 입력으로 하여 예측.
     score = accuracy_score(y_train, y_predict) # 실제 정답과 인공지능이 예측하여 출력한 답과 비교하여 정확도를 계산.
     print(score) # 점수 출력.

     0.9583333333333334

[ ]  y_predict = estimator.predict(x_test) # 학습시킨 인공지능 모델에 x_test 데이터를 입력으로 하여 예측.
     score = accuracy_score(y_test, y_predict) # 실제 정답과 인공지능이 예측하여 출력한 답과 비교하여 정확도를 계산.
     print(score) # 점수 출력.

     0.9666666666666667
```

- predict() : 학습시킨 인공지능 모델에 입력을 주어 출력을 예측하도록 함.
- accuracy_score() : 실측치와 예측치를 비교하여 점수를 계산.
- train 데이터에 대해서 계산한 점수와 test 데이터에 대해서 계산한 점수를 비교하여 현재 학습시킨 인공지능이 train 데이터에 과적합 되지는 않았는지 확인할 수 있음.