# 3주차 실습과제

2017313107 이승태

1. 6장_의사결정나무모델

(1) import하기

```
[1] from sklearn.datasets import load_iris
    import pandas as pd
    import numpy as np
    from matplotlib import pyplot as plt
    import seaborn as sn
```

(2) 데이터 확인하기

```
[2] iris = load_iris()
    iris_frame = pd.DataFrame(data=np.c_[iris['data'], iris['target']],columns = iris['feature_names'] + ['target'])
    iris_frame['target'] = iris_frame['target'].map({1:"versicolor",0:"setosa",2:"virginica"})
    X = iris_frame.iloc[:,:-1]
    Y = iris_frame.iloc[:,[-1]]
    iris_frame
```

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

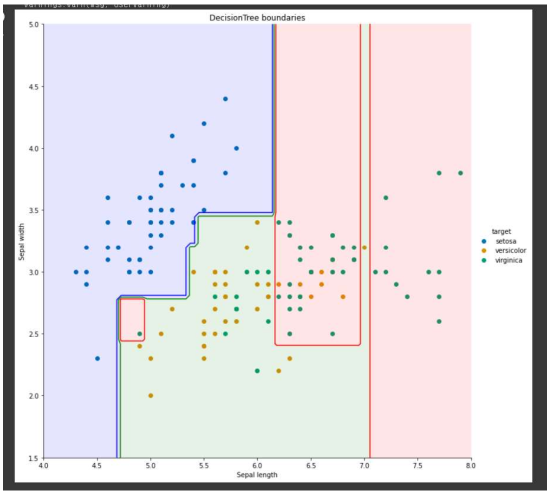


(3) decision tree 만들기

```
[3] from sklearn.tree import DecisionTreeClassifier

    ##아래 하이퍼파라미터 수정으로 결과 확인
    clf = DecisionTreeClassifier(random_state=0, criterion='gini', max_depth=5)
    import matplotlib.colors as colors

    df1 = iris_frame[["sepal length (cm)" , "sepal width (cm)", "target" ]]
    X = df1.iloc[:,0:2]
    Y = df1.iloc[:,2].replace({'setosa':0,'versicolor':1,'virginica':2}).copy()

    clf.fit(X,Y)
    N=100
```

(4) 결과 확인하기

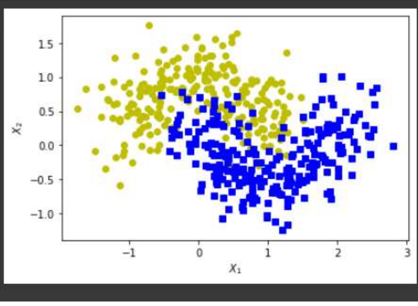
```
[4]  X_= np.linspace(4,8,N)
     Y_= np.linspace(1.5,5,N)
     X_,Y_= np.meshgrid(X_,Y_)

     color_list= ['Blues','Greens','Reds']
     my_norm = colors.Normalize(vmin=-1,vmax=1)
     g= sn.FacetGrid(iris_frame, hue = "target", size =10,
                     palette = 'colorblind').map(plt.scatter, "sepal length (cm)","sepal width (cm)",).add_legend()

     my_ax = g.ax
     zz = np.array( [clf.predict ([[xx,yy]]) [0] for xx,yy in zip(np.ravel(X_),np.ravel(Y_))])
     Z=zz.reshape(X_.shape)

     my_ax.contourf(X_,Y_,Z,2,alpha=.1, colors = ('blue','green','red'))
     my_ax.contour(X_,Y_,Z,2,alpha=1, colors = ('blue','green','red'))

     my_ax.set_xlabel('Sepal length')
     my_ax.set_ylabel('Sepal width')
     my_ax.set_title('DecisionTree boundaries')

     plt.show()
```




2. 7장_makemoons_iris_ensemble

(1) make moon sample을 불러와 그래프를 그렸다.

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

     X, y = make_moons(n_samples=500, noise=0.30, random_state=42)
     plt.plot(X[:, 0][y==0], X[:, 1][y==0], "yo")
     plt.plot(X[:, 0][y==1], X[:, 1][y==1], "bs")
     plt.xlabel("$X_1$")
     plt.ylabel("$X_2$")
     plt.show()
```



(2) train, test data를 나누고 decision tree를 만들고 accuracy를 측정하였다.
bagging ensemble을 이용하였더니 accuracy가 85.6%에서 90.4%로 향상되었다.

```
[2] from sklearn.model_selection import train_test_split
    X_train, X_test, y_train, y_test = train_test_split(X, y, random_state=42)
```

```
[3] from sklearn.ensemble import BaggingClassifier
    from sklearn.tree import DecisionTreeClassifier

    bag_clf = BaggingClassifier(
        DecisionTreeClassifier(random_state=42), n_estimators=500,
        max_samples=100, bootstrap=True, n_jobs=-1, random_state=42)
    bag_clf.fit(X_train, y_train)
    y_pred = bag_clf.predict(X_test)
```

```
[4] from sklearn.metrics import accuracy_score
    print(accuracy_score(y_test, y_pred))

    0.904
```

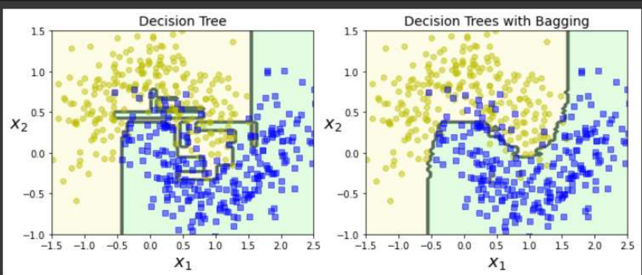
```
[5] tree_clf = DecisionTreeClassifier(random_state=42)
    tree_clf.fit(X_train, y_train)
    y_pred_tree = tree_clf.predict(X_test)
    print(accuracy_score(y_test, y_pred_tree))

    0.856
```

(3) 두 모델을 가시적으로 비교하기 위해 그래프를 그리는 함수를 구현하였다.

```
[6] from matplotlib.colors import ListedColormap

    def plot_decision_boundary(clf, X, y, axes=[-1.5, 2.5, -1, 1.5], alpha=0.5, contour=True):
        x1s = np.linspace(axes[0], axes[1], 100)
        x2s = np.linspace(axes[2], axes[3], 100)
        x1, x2 = np.meshgrid(x1s, x2s)
        X_new = np.c_[x1.ravel(), x2.ravel()]
        y_pred = clf.predict(X_new).reshape(x1.shape)
        custom_cmap = ListedColormap(['#fafab0','#9898ff','#a0faa0'])
        plt.contourf(x1, x2, y_pred, alpha=0.3, cmap=custom_cmap)
        if contour:
            custom_cmap2 = ListedColormap(['#7d7d58','#4c4c7f','#507d50'])
            plt.contour(x1, x2, y_pred, cmap=custom_cmap2, alpha=0.8)
        plt.plot(X[:, 0][y==0], X[:, 1][y==0], "yo", alpha=alpha)
        plt.plot(X[:, 0][y==1], X[:, 1][y==1], "bs", alpha=alpha)
        plt.axis(axes)
        plt.xlabel(r"$x_1$", fontsize=18)
        plt.ylabel(r"$x_2$", fontsize=18, rotation=0)
```

(4) 위의 함수를 이용하여 그래프를 그렸다.

```
[7] plt.figure(figsize=(11,4))
    plt.subplot(121)
    plot_decision_boundary(tree_clf, X, y)
    plt.title("Decision Tree", fontsize=14)
    plt.subplot(122)
    plot_decision_boundary(bag_clf, X, y)
    plt.title("Decision Trees with Bagging", fontsize=14)
    plt.show()
```



(5) randomforest와 bagging ensemble비교

```
[8]  bag_clf = BaggingClassifier(
         DecisionTreeClassifier(splitter="random", max_leaf_nodes=16, random_state=42),
         n_estimators=500, max_samples=1.0, bootstrap=True, n_jobs=-1, random_state=42)

     bag_clf.fit(X_train, y_train)
     y_pred = bag_clf.predict(X_test)
```

```
[9]  from sklearn.ensemble import RandomForestClassifier

     rnd_clf = RandomForestClassifier(n_estimators=500, max_leaf_nodes=16, n_jobs=-1, random_state=42)
     rnd_clf.fit(X_train, y_train)

     y_pred_rf = rnd_clf.predict(X_test)
```

```
[10] print(accuracy_score(y_test, y_pred))

     0.92
```

```
[11] print(accuracy_score(y_test, y_pred_rf))

     0.912
```

```
[12] np.sum(y_pred == y_pred_rf) / len(y_pred)

     0.976
```

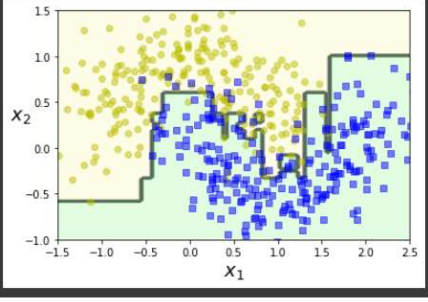
(6) random forest classifier의 상세 정보

```
[13] from sklearn.datasets import load_iris
     from sklearn.ensemble import RandomForestClassifier

     iris = load_iris()
     rnd_clf = RandomForestClassifier(n_estimators=500, n_jobs=-1, random_state=42)
     rnd_clf.fit(iris["data"], iris["target"])
     for name, score in zip(iris["feature_names"], rnd_clf.feature_importances_):
         print(name, score)

     sepal length (cm) 0.11249225099876375
     sepal width (cm) 0.02311928828251033
     petal length (cm) 0.4410304643639577
     petal width (cm) 0.4233579963547682
```

```
[14] rnd_clf.feature_importances_

     array([0.11249225, 0.02311929, 0.44103046, 0.423358  ])
```

(7) adaboost를 이용하여 classifier를 만들고 성능을 가시적으로 평가해 보았다.

```
[15] from sklearn.ensemble import AdaBoostClassifier

     ada_clf = AdaBoostClassifier(
         DecisionTreeClassifier(max_depth=1), n_estimators=200,
         algorithm="SAMME.R", learning_rate=0.5, random_state=42)
     ada_clf.fit(X_train, y_train)

     AdaBoostClassifier(algorithm='SAMME.R',
                        base_estimator=DecisionTreeClassifier(ccp_alpha=0.0,
                                                              class_weight=None,
                                                              criterion='gini',
                                                              max_depth=1,
                                                              max_features=None,
                                                              max_leaf_nodes=None,
                                                              min_impurity_decrease=0.0,
                                                              min_impurity_split=None,
                                                              min_samples_leaf=1,
                                                              min_samples_split=2,
                                                              min_weight_fraction_leaf=0.0,
                                                              presort='deprecated',
                                                              random_state=None,
                                                              splitter='best'),
                        learning_rate=0.5, n_estimators=200, random_state=42)

[16] plot_decision_boundary(ada_clf, X, y)
```




3. 7장_MNIST_knn분류

(1) MNIST데이터 셋을 가지고 온 다음 data와 label로 나누고, test train data로 나누었다. 그 후 데이터의 크기를 확인해 보았다.

```
[3] from sklearn.datasets import fetch_openml
    mnist = fetch_openml('mnist_784')
    mnist.data.shape, mnist.target.shape

    ((70000, 784), (70000,))

[4] X, y = mnist["data"], mnist["target"]
    X.shape

    (70000, 784)

[5] y.shape

    (70000,)

[6] X_train, X_test, y_train, y_test = X[:60000], X[60000:], y[:60000], y[60000:]

[7] X_train.shape

    (60000, 784)

[8] X_test.shape

    (10000, 784)
```

(2) 좀 더 좋은 결과를 얻기 위하여 training data를 shuffle해주었다.

```
[9]  #Training set 순서 섞기 (shuffling)
     import numpy as np

     shuffle_index = np.random.permutation(60000)
     X_train, y_train = X_train[shuffle_index], y_train[shuffle_index]

[10] shuffle_index

     array([11866, 57804,  6026, ..., 59961, 56437,  4791])
```

(3) KNN classifier를 만들고 이를 이용하여 결과 값을 예측해보았다.

```
[9]  from sklearn.neighbors import KNeighborsClassifier
     knn_clf = KNeighborsClassifier(n_jobs=-1, weights='distance', n_neighbors=4)
     knn_clf.fit(X_train, y_train)

     KNeighborsClassifier(algorithm='auto', leaf_size=30, metric='minkowski',
                          metric_params=None, n_jobs=-1, n_neighbors=4, p=2,
                          weights='distance')

[10] y_knn_pred = knn_clf.predict(X_test)

[11] from sklearn.metrics import accuracy_score
     accuracy_score(y_test, y_knn_pred)

     0.9714
```