작성일 : 2022/06/23

지능정보공학설계 기말 대체 과제 2
: 인생 최초 ML 프로그램 작성하기

휴먼정보공학 202210829 김진석

목차

1. 코드

1.1 import

```
import matplotlib.pyplot as plt
from sklearn import datasets
from sklearn.model_selection import train_test_split
from sklearn import tree
from sklearn.metrics import accuracy_score
```

학습 및 예측을 할 수 있는 라이브러리인 sklearn의 dataset과 tree, accuracy_score, train_test_split 개체를 불러옴.

1.2 함수

```
def plot_accuracy(prediction, truth):

    acc = accuracy_score(prediction, truth)
    print(acc)
    return acc
```

특정 비율의 결과를 출력하는 accuracy_score() 함수를 이용해 비율의 결과를 return하는 함수

1.3 본문 소스

```
test_list = ['0.7', '0.5', '0.3']
result_list = []
iris = datasets.load_iris()


x = iris.data
y = iris.target
for i in test_list:
    x_train, x_test, y_train, y_test = train_test_split(x, y, test_size= float(i))

    clf = tree.DecisionTreeClassifier()
    clf.fit(x_train, y_train)

    predctions = clf.predict(x_test)
    print(i,'test_size')
    print('예측: \n', predctions)
    print('정답(y_test): \n', y_test)

    result_list.append(plot_accuracy(predctions, y_test))


print(result_list)
```

test 사이즈를 저장하는 리스트와 accuracy 값을 저장하는 result 리스트를 생성, 학습 및 예측한 결과를 변수에 저장, 결정트리분류기에 학습된 결과 저장, 예측한 결과 출력

1.4 matplotlib 그래프 그리기

```
bar = plt.bar(test_list, result_list)
for rect in bar:
    height = rect.get_height()
    plt.text(rect.get_x() + rect.get_width()/2.0, height, '%.3f' % height, ha='center', va='bottom', size = 12)
plt.title('Results of accuracy')
plt.xlabel('test_size')
plt.ylabel('accuracy')
plt.show()
```

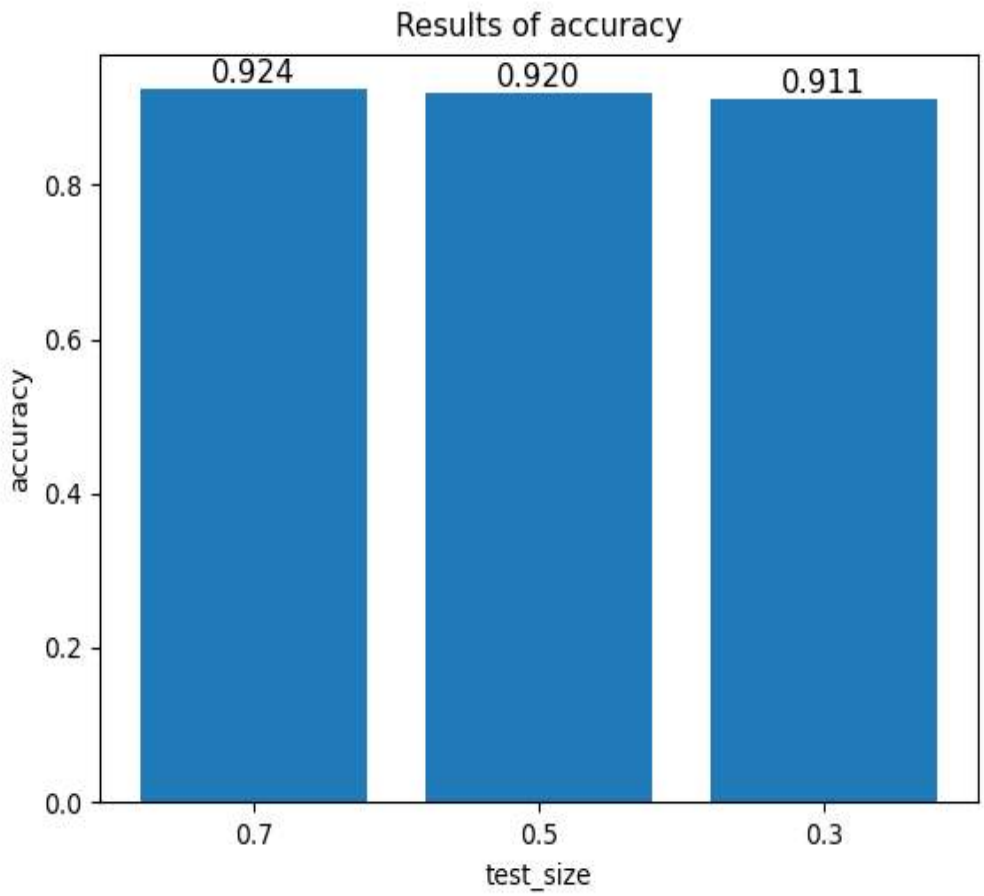
matplotlib 라이브러리로 bar 그래프를 그림, y축의 그래프 값을 표시하고, 제목, xlabel, ylabel 설정

2. 결과

2.1 예측 및 정답

```
0.7 test_size
예측:
 [1 2 0 0 1 2 2 1 1 1 1 0 2 0 1 1 2 0 2 1 0 1 0 2 1 2 2 1 0 1 0 0 1 1 2 2 1
 2 2 2 0 1 0 1 1 0 1 0 1 0 0 0 1 0 2 1 2 0 0 1 1 2 2 2 0 2 2 1 1 1 2 2 2 2
 0 2 2 2 0 1 0 1 2 1 0 2 2 1 0 0 0 0 1 1 0 1 0 1 1 2 2 0 0 2 0]
정답(y_test):
 [1 1 0 0 1 2 2 1 1 1 1 0 2 0 1 1 1 0 2 1 0 2 0 2 1 2 2 1 0 1 0 0 1 1 2 2 2
 2 2 2 0 1 0 1 1 0 1 0 1 0 0 0 1 0 2 1 2 0 0 1 1 2 2 2 0 1 2 1 1 1 2 2 2 2
 0 2 2 2 0 2 0 1 2 1 0 2 2 1 0 0 0 0 2 1 0 2 0 1 1 2 2 0 0 2 0]
0.9238095238095239
0.5 test_size
예측:
 [0 0 1 0 2 1 0 2 1 2 0 0 2 0 1 2 2 0 1 1 1 1 2 1 1 1 1 1 2 1 2 1 2 0 0 0 0
 2 1 2 2 1 0 0 0 1 2 1 2 1 0 0 1 2 2 0 2 0 1 0 1 1 1 2 1 2 1 0 0 1 0 1 0 1
 2]
정답(y_test):
 [0 0 2 0 2 1 0 2 1 2 0 0 2 0 1 2 2 0 1 1 1 1 2 2 1 1 1 1 2 1 2 1 2 0 0 0 0
 2 1 2 2 1 0 0 0 2 2 1 2 1 0 0 1 2 2 0 2 0 2 0 2 2 1 2 1 2 1 0 0 1 0 1 0 1
 2]
0.92
0.3 test_size
예측:
 [2 0 0 2 0 2 2 2 1 0 1 1 2 0 2 2 0 2 0 1 1 2 0 0 0 1 2 2 0 2 1 2 2 2 1 2 2
 0 2 0 2 1 2 0 1]
정답(y_test):
 [2 0 0 1 0 2 2 2 1 0 1 1 1 0 2 2 0 2 0 1 1 2 0 0 0 1 2 2 0 2 1 2 1 2 1 2 2
 0 2 0 1 1 2 0 1]
0.9111111111111111
[0.9238095238095239, 0.92, 0.9111111111111111]
```

2.1 accuracy 그래프



3. Accuracy가 다르게 나오는 이유

test_size의 값에 따라 Accuracy의 값이 바뀐다. 왜냐하면 test_size는 테스트 셋의 구성의 비율을 나타내는데, 전체 데이터 셋의 지정한 값의 비율만큼 테스트 셋으로 지정하겠다는 뜻이다. 즉 테스트 셋의 구성 비율이 높으면 높을 수로 실제 데이터와 예측 데이터와의 오차를 더 줄일 수 있기 때문에 test_size에 따라 Accuracy의 값이 다르게 나오는 것 같다.