기계 학습 실습

4주차

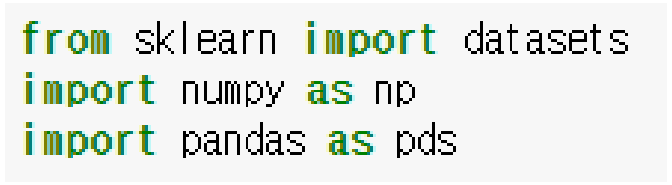
5 주차 학습 목표

1. Scikit-learn library를 사용하여 Decision Tree classification을 할 수 있습니다

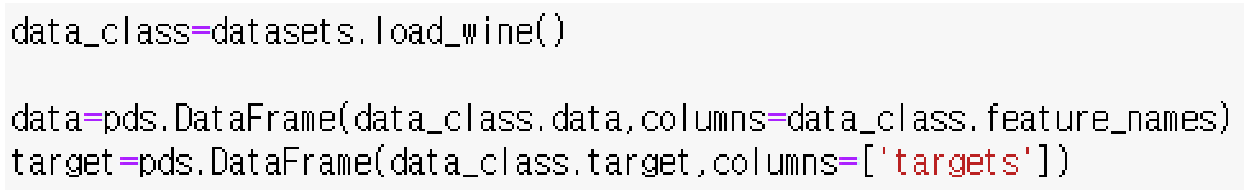
2. Plotly와 Graphviz를 사용하여 데이터와 결과에 대한 시각화를 할 수 있습니다

1. Load datasets

Scikit-learn을 사용하여 Dataset을 불러오고, 데이터를 가공하기 위해 이하 library를 사용합니다



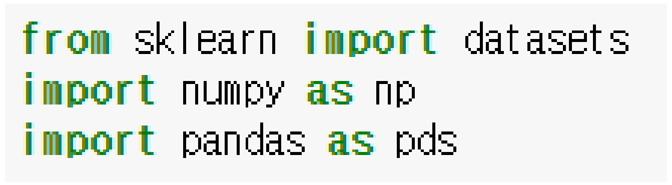
```
from sklearn import datasets
import numpy as np
import pandas as pds
```



```
data_class=datasets.load_wine()

data=pds.DataFrame(data_class.data,columns=data_class.feature_names)
target=pds.DataFrame(data_class.target,columns=['targets'])
```

1. Load datasets

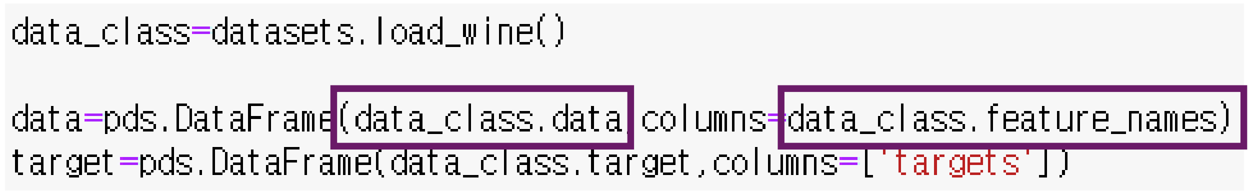
load_wine object의 method들을 사용하여 data와 target을 불러옵니다



```
from sklearn import datasets
import numpy as np
import pandas as pds
```

1. load_wine.data : np.array로 data를 제공합니다

2. load_wine.feature_names : np.array로 data의 feature name을 제공합니다



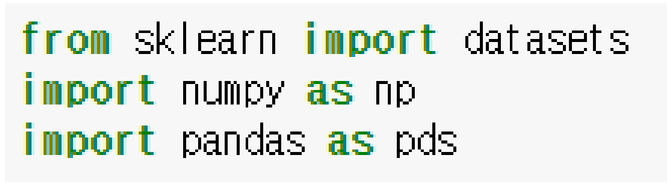
```
data_class=datasets.load_wine()

data=pds.DataFrame(data_class.data,columns=data_class.feature_names)
target=pds.DataFrame(data_class.target,columns=['targets'])
```

1. Load datasets

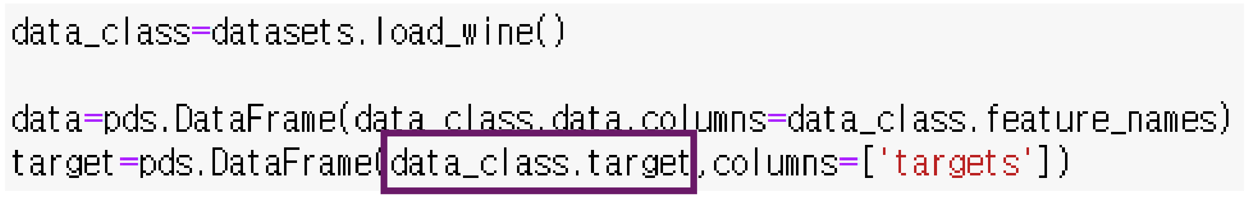
load_wine object의 method들을 사용하여 data와 target을 불러옵니다

3. load_wine.target : np.array로 target class의 categorical value를 제공합니다

4. load_wine.target_names : np.array로 target의 숫자가 어떤 class인지 알려줍니다



```
from sklearn import datasets
import numpy as np
import pandas as pds
```



```
data_class=datasets.load_wine()

data=pds.DataFrame(data_class.data,columns=data_class.feature_names)
target=pds.DataFrame(data_class.target,columns=['targets'])
```

2. Criterion

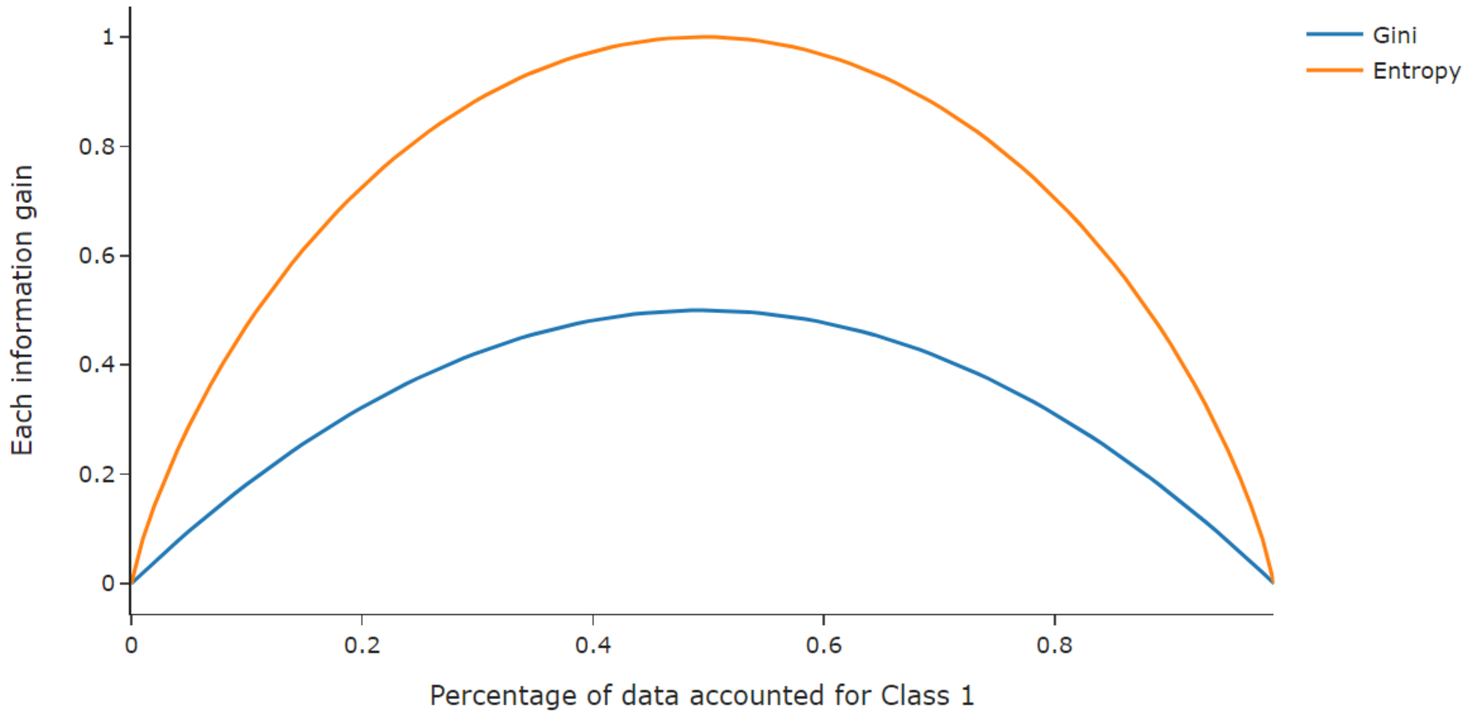
- 데이터를 이진 분류하는 기준
- 기준으로 나뉠 수록 leaf node에는 class의 다양성이 줄어듦
- Information gain으로 class의 다양성을 계산함
- 계산 방법은 entropy와 gini index가 존재

$$Gini = 1 - \sum_{i=1}^{n} p^2(c_i)$$

$$Entropy = -\sum_{i=1}^{n} p(c_i) \log p(c_i)$$

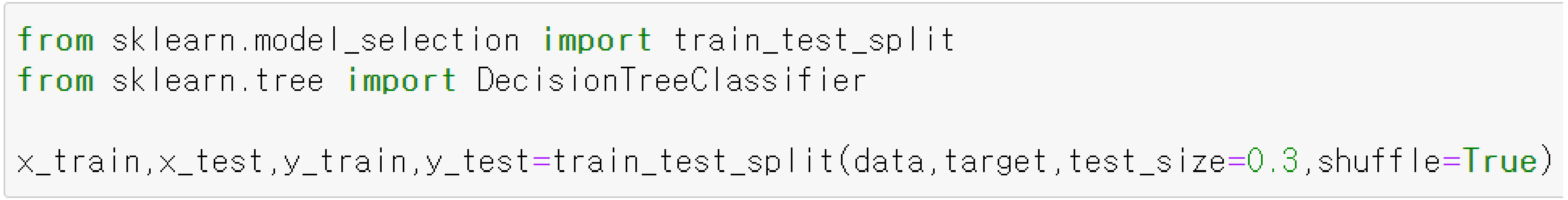
2. Criterion

- Information 계산 방법에 따른 결과 차이

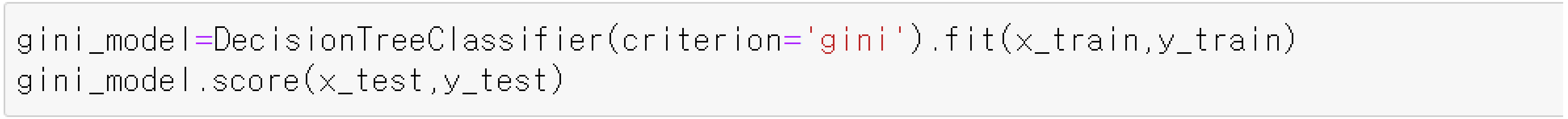


3. Decision Tree

- Dataset의 분리와 decision tree를 사용한 분류 학습



```
from sklearn.model_selection import train_test_split
from sklearn.tree import DecisionTreeClassifier

x_train,x_test,y_train,y_test=train_test_split(data,target,test_size=0.3,shuffle=True)
```



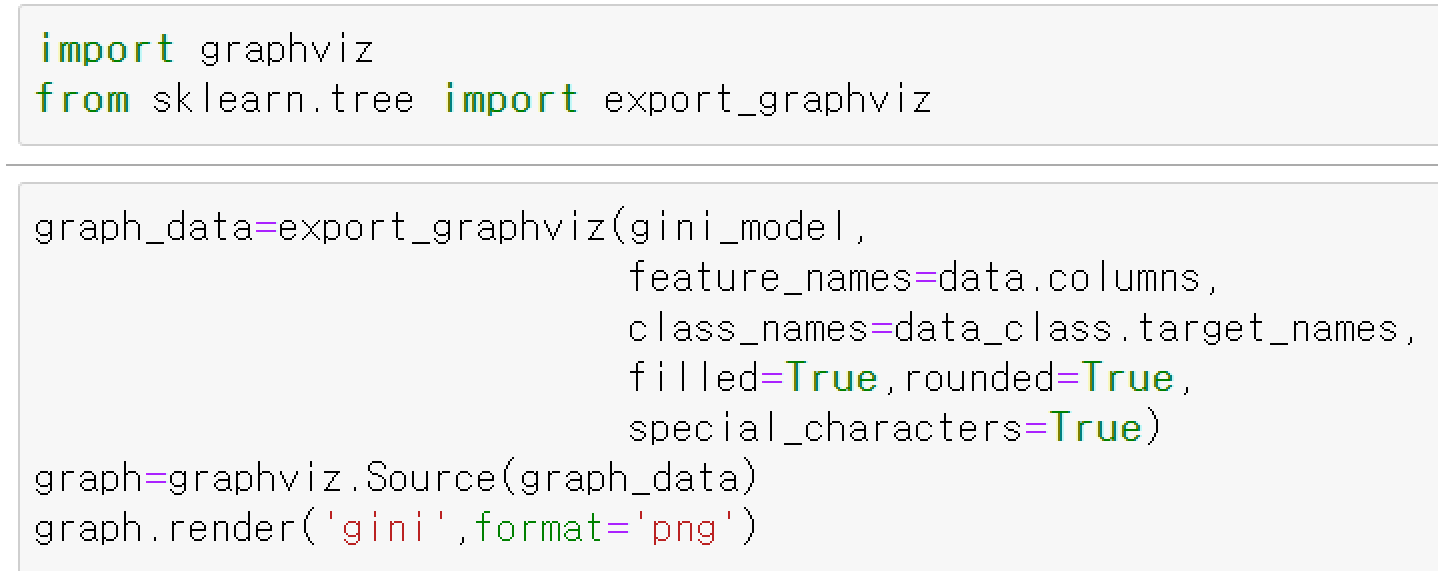
```
gini_model=DecisionTreeClassifier(criterion='gini').fit(x_train,y_train)
gini_model.score(x_test,y_test)
```

- Criterion으로 gini index와 entropy 선택 가능

- Classifier의 score method로 accuracy 계산 가능

3. Decision Tree

- Graphviz library를 사용하여 tree구조 시각화

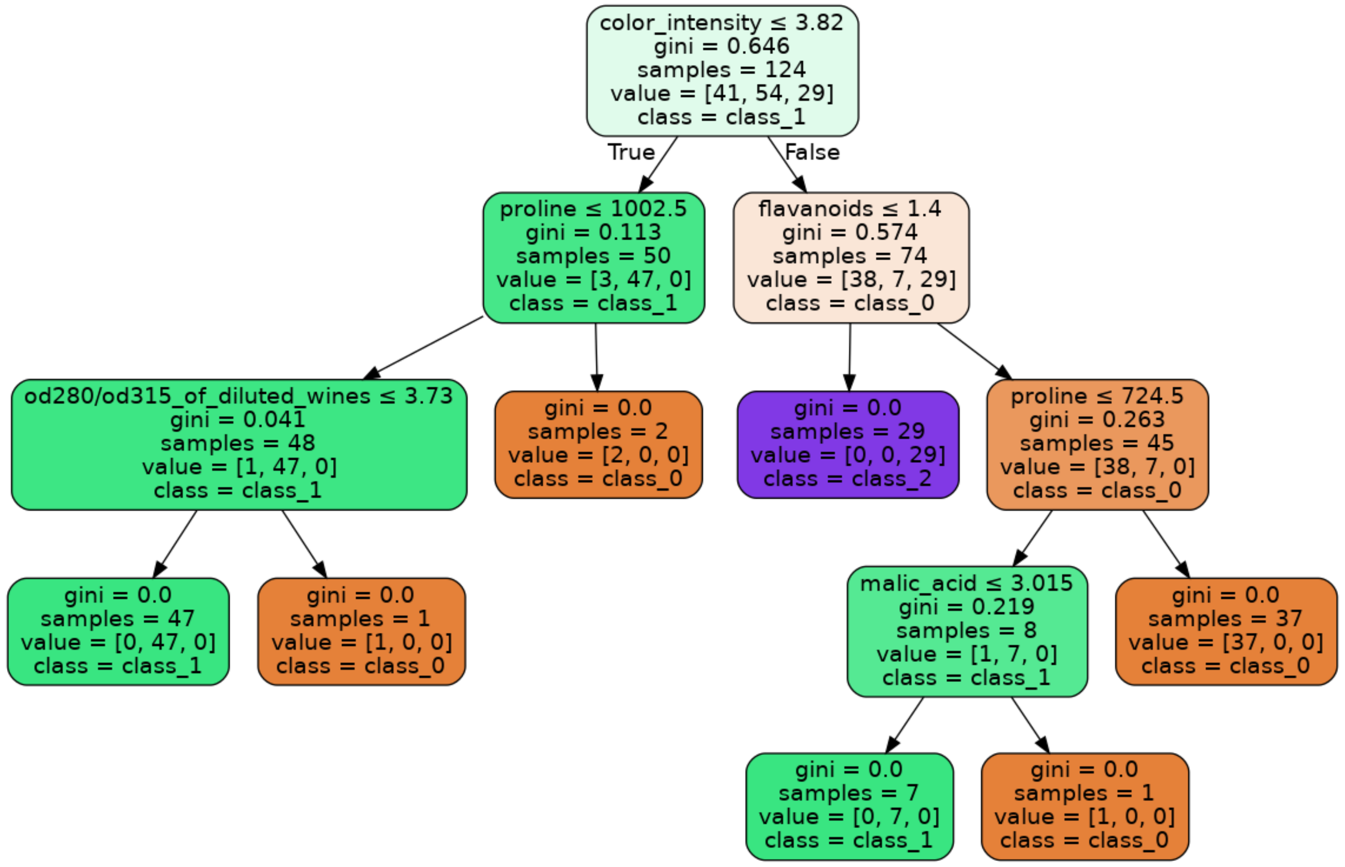


```python
import graphviz
from sklearn.tree import export_graphviz
```

```python
graph_data=export_graphviz(gini_model,
                           feature_names=data.columns,
                           class_names=data_class.target_names,
                           filled=True,rounded=True,
                           special_characters=True)
graph=graphviz.Source(graph_data)
graph.render('gini',format='png')
```

3. Decision Tree

- Graphviz library를 사용하여 tree 구조 시각화



4. Pruning

- Pre pruning과 post pruning을 활용한 tree의 complexity 조절
- Pre pruning은 tree class의 hyper parameter조절을 통해 pruning 시도
- 조절된 hyper parameter로 인해 greedy한 tree 생성
- Post pruning은 full tree기준으로 tree를 되짚어가며 complexity와 accuracy에 따른 합리적 pruning 시도
- 되짚어가는 resource cost비용이 비쌈

4. Pruning

- Pre pruning – hyper parameter

1. max_depth : int, default=None. The maximum depth of the tree. If None, then nodes are expanded until all leaves are pure or until all leaves contain less than min_samples_split samples.

2. min_samples_split : int or float, default=2. The minimum number of samples required to split an internal node

3. min_samples_leaf : int or float, default=1. The minimum number of samples required to be at a leaf node. A split point at any depth will only be considered if it leaves at least ``min_samples_leaf`` training samples in each of the left and right branches. This may have the effect of smoothing the model, especially in regression.

4. min_weight_fraction_leaf : float, default=0.0. The minimum weighted fraction of the sum total of weights (of all the input samples) required to be at a leaf node. Samples have equal weight when sample_weight is not provided.

4. Pruning

- Pre pruning – hyper parameter

5. max_features : int, float or {"auto", "sqrt", "log2"}, default=None. The number of features to consider when looking for the best split:

6. max_leaf_nodes : int, default=None. Grow a tree with ``max_leaf_nodes`` in best-first fashion. Best nodes are defined as relative reduction in impurity. If None then unlimited number of leaf nodes.

7. min_impurity_decrease : float, default=0.0. A node will be split if this split induces a decrease of the impurity greater than or equal to this value.

8. min_impurity_split : float, default=0. Threshold for early stopping in tree growth. A node will split if its impurity is above the threshold, otherwise it is a leaf.

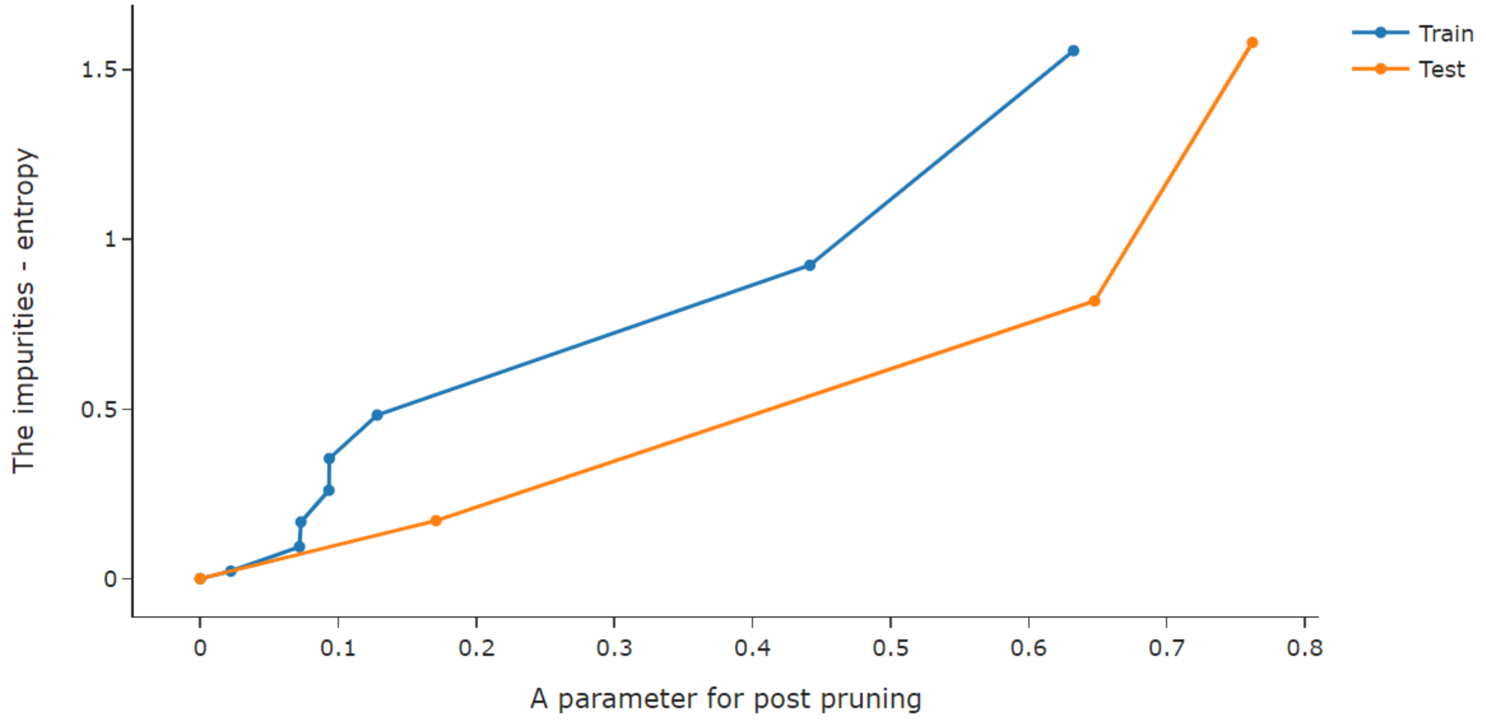
4. Pruning

- Post pruning – cost complexity pruning path

```
tree=DecisionTreeClassifier(criterion='entropy').fit(x_train,y_train)


fig=go.Figure()

fig.add_trace(go.Scatter(x=tree.cost_complexity_pruning_path(x_train,y_train)['ccp_alphas'],
                         y=tree.cost_complexity_pruning_path(x_train,y_train)['impurities'],
                         name='Train'))
fig.add_trace(go.Scatter(x=tree.cost_complexity_pruning_path(x_test,y_test)['ccp_alphas'],
                         y=tree.cost_complexity_pruning_path(x_test,y_test)['impurities'],
                         name='Test'))
fig_options={'layout':dict(template='simple_white',
                           width=800,height=500,
                           xaxis_title='A parameter for post pruning',
                           yaxis_title='The impurities - entropy')}

fig.update(fig_options)
fig.show()
```

4. Pruning

- Post pruning – cost complexity pruning path



W
[설

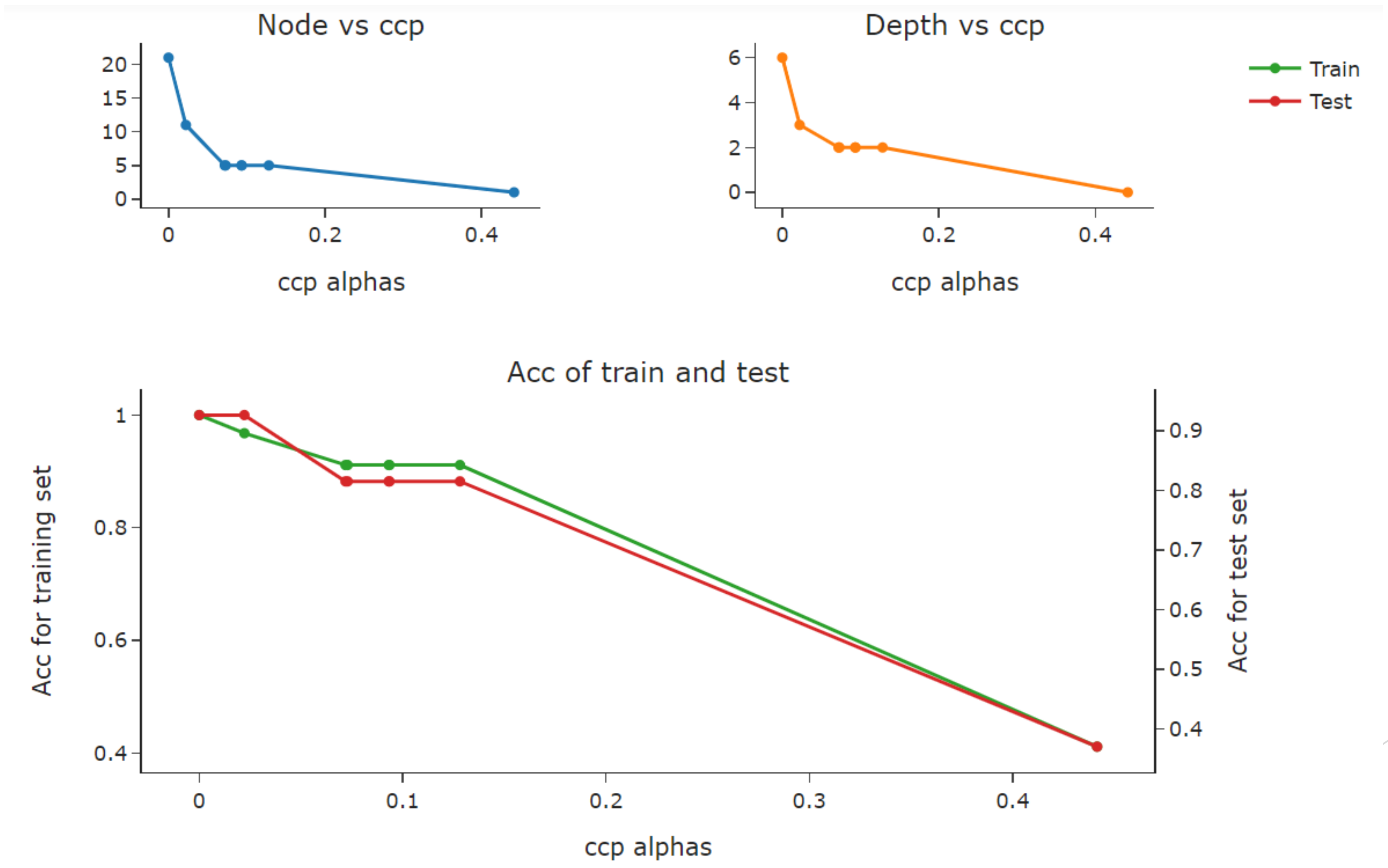
4. Pruning

- Post pruning – cost complexity pruning path

```
ccp=tree.cost_complexity_pruning_path(x_train,y_train)
ccp_alphas=ccp.ccp_alphas

clfs = []
for ccp_alpha in ccp_alphas:
    clf = DecisionTreeClassifier(random_state=0, ccp_alpha=ccp_alpha)
    clf.fit(x_train, y_train)
    clfs.append(clf)

clfs = clfs[:-1]
ccp_alphas = ccp_alphas[:-1]

node_counts = [clf.tree_.node_count for clf in clfs]
depth = [clf.tree_.max_depth for clf in clfs]

train_scores = [clf.score(x_train, y_train) for clf in clfs]
test_scores = [clf.score(x_test, y_test) for clf in clfs]
```

4. Pruning

- Post pruning – cost complexity pruning path



과제 공지

문의, 과제 제출 메일은 jjooww5182@gmail.com으로 보내주세요

영상 및 과제에 대한 의견도 부탁드립니다