semi-supervised

November 21, 2024

[147]:
```
from sklearn import semi_supervised
from sklearn.ensemble import RandomForestClassifier
from sklearn.model_selection import train_test_split
from sklearn.metrics import accuracy_score
from sklearn.datasets import make_classification
from sklearn import datasets
from sklearn.linear_model import LogisticRegression
from sklearn.tree import export_graphviz
import pydot
import numpy as np
import matplotlib.pyplot as plt
from sklearn.tree import plot_tree
```

[148]:
```
x,y =␣
 ↪make_classification(n_samples=1000,n_features=20,n_classes=2,random_state=42)
```

[149]:
```
x_train_val,x_test,y_train_val,y_test = train_test_split(x,y,test_size=0.
 ↪2,random_state=42)
x_labled,x_unlabeled, y_labled,y_unlabled =␣
 ↪train_test_split(x_train_val,y_train_val,test_size=0.9,random_state=42)
```

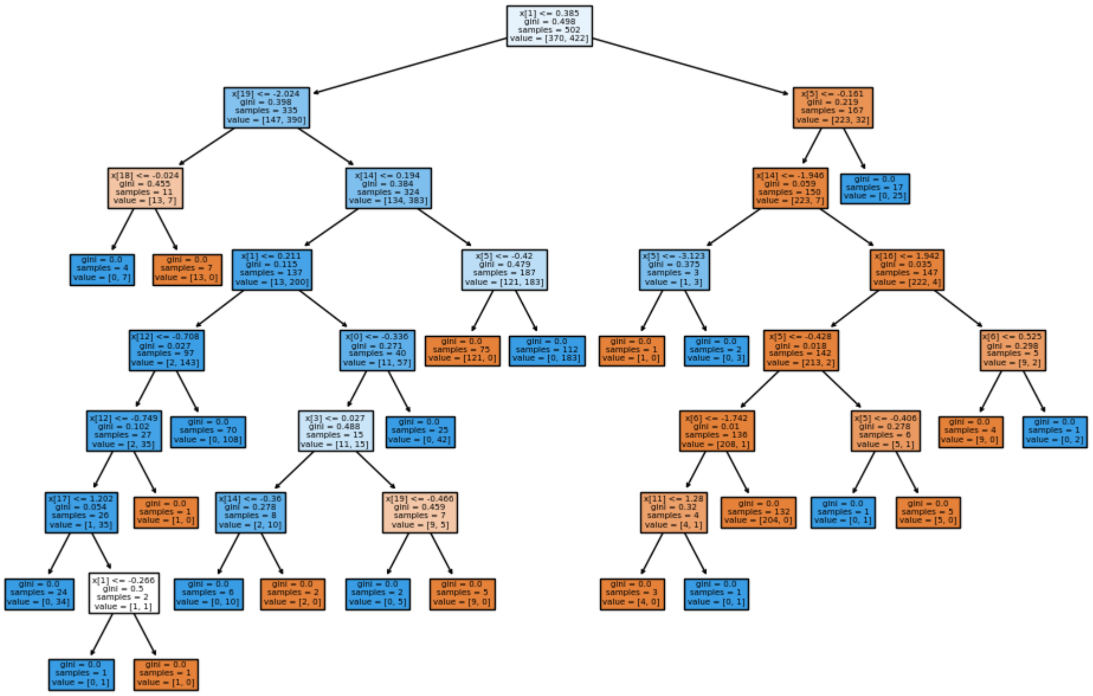
[150]:
```
print("Total no of samples:",len(x))
print("Total no of labled samples:",len(x_labled))
print("Total no of unlabled samples:",len(x_unlabeled))
print("Total no of test samples:",len(x_test))
```

```
Total no of samples: 1000
Total no of labled samples: 80
Total no of unlabled samples: 720
Total no of test samples: 200
```

[151]:
```
model = RandomForestClassifier(n_estimators=100,random_state=42)

max_iteration=10
confidence_threshold =0.6
for iteration in range(max_iteration):
    model.fit(x_labled,y_labled)
    probas = model.predict_proba(x_unlabeled)
    confident_indices = np.where(np.max(probas,axis=1)>=confidence_threshold)[0]
    if len(confident_indices) == 0:
        print("No confident Prediction found stop  training")
        break
    x_labled = np.vstack((x_labled,x_unlabeled[confident_indices]))
    y_labled = np.hstack((y_labled,np.argmax(probas[confident_indices],axis=1)))

    x_unlabeled = np.delete(x_unlabeled,confident_indices,axis=0)

    print(f"Iteration:{iteration+1}")
    print(f"Added {len(confident_indices)} confident prediction to the labled␣
 ↪dataset")
    print(f"Remaining unlabled smaples: {len(x_unlabeled)}")
    print(f"Number of training samples after iteration {iteration+1}:
 ↪{len(x_labled)}")
    if len(x_unlabeled)==0:
        break

# final retraining on originally labelled data
model.fit(x_labled,y_labled)

y_pred = model.predict(x_test)
test_acc = accuracy_score(y_test,y_pred)
print(f"Test accuracy after final retraining on test data: {test_acc:.4f}")

y_test_pred = model.predict(x_labled)
test_acc = accuracy_score(y_labled,y_test_pred)
print(f"Test accuracy after final retraining on training data: {test_acc:.4f}")

tree = model.estimators_[0]
plt.figure(figsize=(12, 8))
plot_tree(tree, filled=True)
plt.show()
```

```
Iteration:1
Added 653 confident prediction to the labled dataset
Remaining unlabled smaples: 67
Number of training samples after iteration 1:733
Iteration:2
Added 54 confident prediction to the labled dataset
Remaining unlabled smaples: 13
Number of training samples after iteration 2:787
Iteration:3
Added 4 confident prediction to the labled dataset
Remaining unlabled smaples: 9
Number of training samples after iteration 3:791
```

```
Iteration:4
Added 1 confident prediction to the labled dataset
Remaining unlabled smaples: 8
Number of training samples after iteration 4:792
No confident Prediction found stop  training
Test accuracy after final retraining on test data: 0.8650
Test accuracy after final retraining on training data: 1.0000
```


x[1] <= 0.385 gini = 0.498 samples = 502 value = [370, 422]
x[19] <= -2.024 gini = 0.398 samples = 335 value = [147, 390]
x[5] <= -0.161 gini = 0.219 samples = 167 value = [223, 32]
x[18] <= -0.024 gini = 0.455 samples = 11 value = [13, 7]
x[14] <= 0.194 gini = 0.384 samples = 324 value = [134, 383]
x[14] <= -1.946 gini = 0.059 samples = 150 value = [223, 7]
gini = 0.0 samples = 17 value = [0, 25]
gini = 0.0 samples = 4 value = [0, 7]
gini = 0.0 samples = 7 value = [13, 0]
x[1] <= 0.211 gini = 0.115 samples = 137 value = [13, 200]
x[5] <= -0.42 gini = 0.479 samples = 187 value = [121, 183]
x[5] <= -3.123 gini = 0.375 samples = 3 value = [1, 3]
x[16] <= 1.942 gini = 0.035 samples = 147 value = [222, 4]
x[12] <= -0.708 gini = 0.027 samples = 97 value = [2, 143]
x[0] <= -0.336 gini = 0.271 samples = 40 value = [11, 57]
gini = 0.0 samples = 75 value = [121, 0]
gini = 0.0 samples = 112 value = [0, 183]
gini = 0.0 samples = 1 value = [1, 0]
gini = 0.0 samples = 2 value = [0, 3]
x[5] <= -0.428 gini = 0.018 samples = 142 value = [213, 2]
x[6] <= 0.525 gini = 0.298 samples = 5 value = [9, 2]
x[12] <= -0.749 gini = 0.102 samples = 27 value = [2, 35]
gini = 0.0 samples = 70 value = [0, 108]
x[3] <= 0.027 gini = 0.488 samples = 15 value = [11, 15]
gini = 0.0 samples = 25 value = [0, 42]
x[6] <= -1.742 gini = 0.01 samples = 136 value = [208, 1]
x[5] <= -0.406 gini = 0.278 samples = 6 value = [5, 1]
gini = 0.0 samples = 4 value = [9, 0]
gini = 0.0 samples = 1 value = [0, 2]
x[17] <= 1.202 gini = 0.054 samples = 26 value = [1, 35]
gini = 0.0 samples = 1 value = [1, 0]
x[14] <= -0.36 gini = 0.278 samples = 8 value = [2, 10]
x[19] <= -0.466 gini = 0.459 samples = 7 value = [9, 5]
x[11] <= 1.28 gini = 0.32 samples = 4 value = [4, 1]
gini = 0.0 samples = 132 value = [204, 0]
gini = 0.0 samples = 1 value = [0, 1]
gini = 0.0 samples = 5 value = [5, 0]
gini = 0.0 samples = 24 value = [0, 34]
x[1] <= -0.266 gini = 0.5 samples = 2 value = [1, 1]
gini = 0.0 samples = 6 value = [0, 10]
gini = 0.0 samples = 2 value = [2, 0]
gini = 0.0 samples = 2 value = [0, 5]
gini = 0.0 samples = 5 value = [9, 0]
gini = 0.0 samples = 3 value = [4, 0]
gini = 0.0 samples = 1 value = [0, 1]
gini = 0.0 samples = 1 value = [0, 1]
gini = 0.0 samples = 1 value = [1, 0]

[152]:
```python
iris = datasets.load_iris()
x = iris.data
y = iris.target
```

[153]:
```python
x_train_val,x_test,y_train_val,y_test = train_test_split(x,y,test_size=0.
 ↪2,random_state=42)
x_labled,x_unlabeled, y_labled,y_unlabled =␣
 ↪train_test_split(x_train_val,y_train_val,test_size=0.9,random_state=42)
```

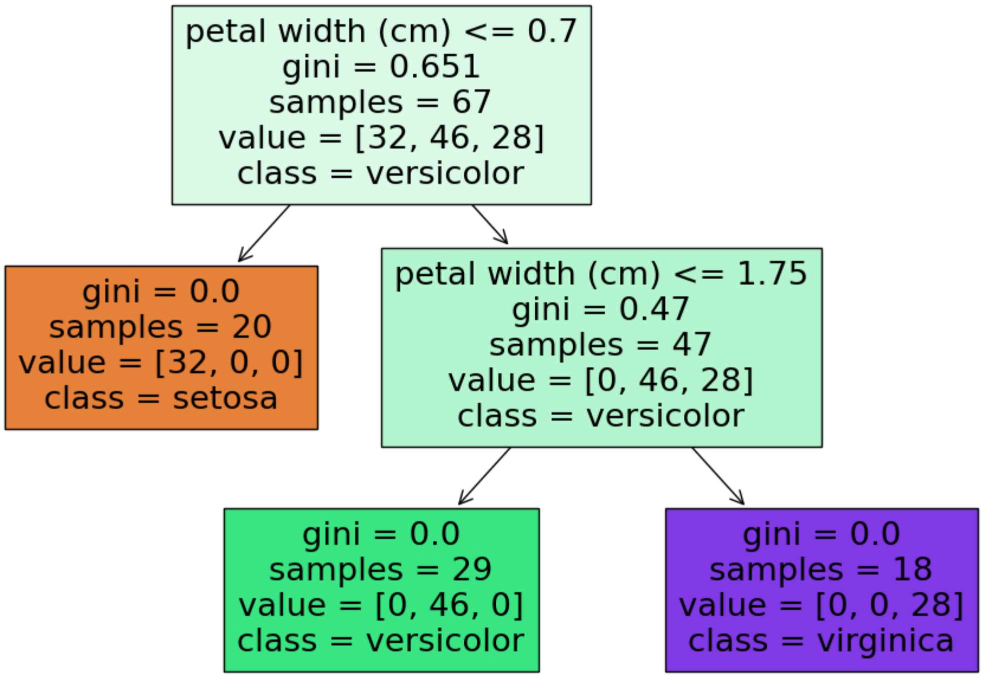
[154]:
```python
print("Total no of samples:",len(x))
print("Total no of labled samples:",len(x_labled))
print("Total no of unlabled samples:",len(x_unlabeled))
print("Total no of test samples:",len(x_test))
```

```
Total no of samples: 150
Total no of labled samples: 12
Total no of unlabled samples: 108
Total no of test samples: 30
```

```
[155]: model = RandomForestClassifier(n_estimators=100,random_state=42)
max_iteration=10
confidence_threshold =0.7
for iteration in range(max_iteration):
    model.fit(x_labled,y_labled)
    probas = model.predict_proba(x_unlabeled)
    confident_indices = np.where(np.max(probas,axis=1)>=confidence_threshold)[0]
    if len(confident_indices) == 0:
        print("No confident Prediction found stop  training")
        break

    x_labled = np.vstack((x_labled,x_unlabeled[confident_indices]))
    y_labled = np.hstack((y_labled,np.argmax(probas[confident_indices],axis=1)))

    x_unlabeled = np.delete(x_unlabeled,confident_indices,axis=0)

    print(f"Iteration:{iteration+1}")
    print(f"Added {len(confident_indices)} confident prediction to the labled␣
  ↪dataset")
    print(f"Remaining unlabled smaples: {len(x_unlabeled)}")
    print(f"Number of training samples after iteration {iteration+1}:
  ↪{len(x_labled)}")
    if len(x_unlabeled)==0:
        break

# final retraining on originally labelled data
model.fit(x_labled,y_labled)

y_pred = model.predict(x_test)
test_acc = accuracy_score(y_test,y_pred)
print(f"Test accuracy after final retraining on test data: {test_acc:.4f}")

y_test_pred = model.predict(x_labled)
test_acc = accuracy_score(y_labled,y_test_pred)
print(f"Test accuracy after final retraining on training data: {test_acc:.4f}")


tree = model.estimators_[0]
plt.figure(figsize=(12, 8))
plot_tree(tree, feature_names=iris.feature_names, class_names=iris.target_names.
  ↪tolist(), filled=True)
plt.show()
```

```
Iteration:1
```

```
Added 92 confident prediction to the labled dataset
Remaining unlabled smaples: 16
Number of training samples after iteration 1:104
Iteration:2
Added 2 confident prediction to the labled dataset
Remaining unlabled smaples: 14
Number of training samples after iteration 2:106
No confident Prediction found stop  training
Test accuracy after final retraining on test data: 0.9000
Test accuracy after final retraining on training data: 1.0000
```



[ ]:

[ ]: