Final Project - Iris Test Cases

SCC461 - Programming for Data Scientists

Kyriakos Chiotis - 35278597

This is the second part of test cases where it is visualised the comparison between the implemeted classifier and the sklearn's decision tree classifier. Explanation of the units for the implemented algorithm is provided in UnitTestCases.

Initially, we prepare the data for the algorithms:

In [65]:

```
import numpy as np
import pandas as pd
import sklearn
from sklearn.model_selection import KFold, train_test_split
from sklearn.tree import DecisionTreeClassifier
from sklearn import metrics
import pydotplus
from IPython.display import Image

data_array = np.genfromtxt("iris.data", dtype='str', delimiter=",")
df = pd.DataFrame(data=data_array)
df[[0, 1, 2, 3]] = df[[0, 1, 2, 3]].apply(pd.to_numeric)

labels = df[4].values
df.drop(4, axis=1, inplace=True)
features = df.values

x_train, x_test, y_train, y_test = train_test_split(features, labels, test_size=0.2, sh
uffle=True, random_state=6)
```

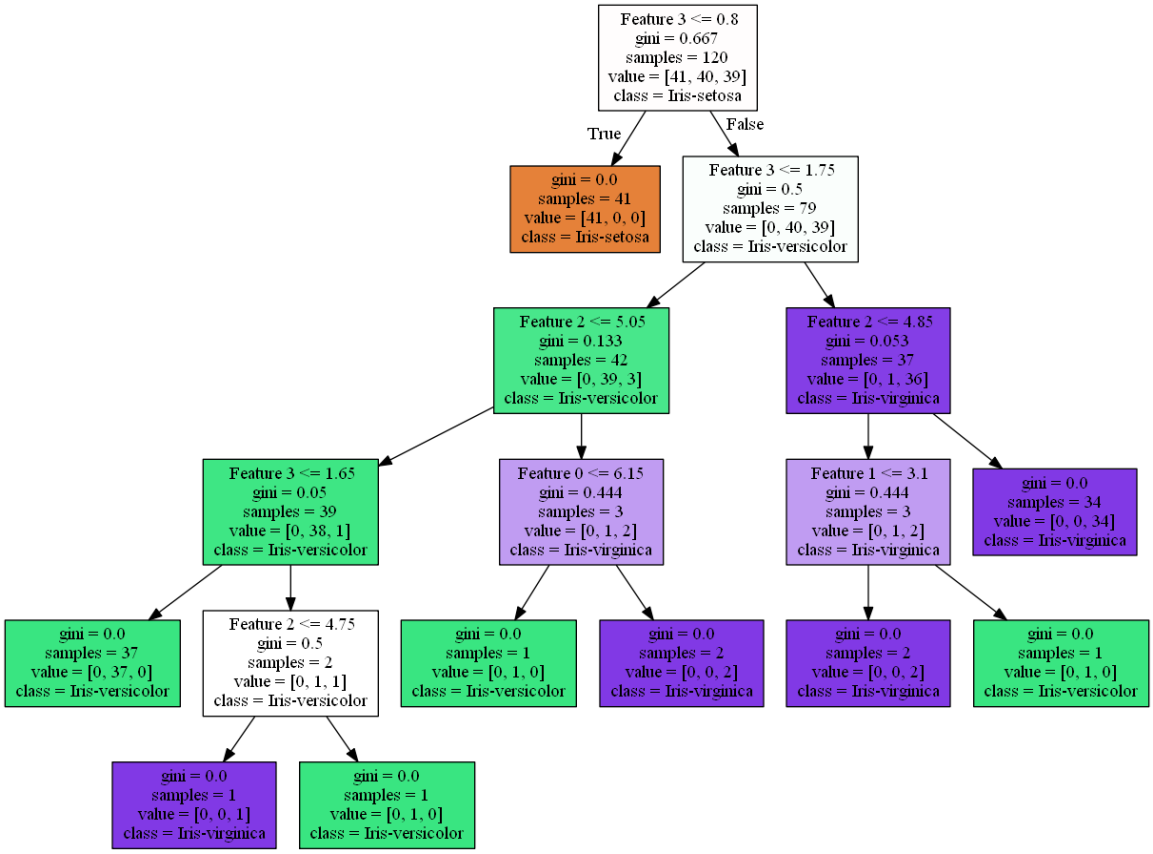
Now, let's visualise sklearn classifier.

In [66]:

```
skdc = DecisionTreeClassifier(random_state=6)

skdc.fit(x_train, y_train)
dot_data = sklearn.tree.export_graphviz(skdc, out_file=None,
                                        feature_names=['Feature 0', ' Feature 1', '
Feature 2', ' Feature 3'],
                                        class_names=['Iris-setosa', 'Iris-versicolo
r', 'Iris-virginica'],
                                        filled=True)
graph = pydotplus.graph_from_dot_data(dot_data)
Image(graph.create_png())
```

Out[66]:



Implemeted decision tree visualisation.

In [68]:

```
mydc = MyDecisionTree()
tree = mydc.create_tree(x_train.tolist(), y_train.tolist())
mydc.print_tree(tree)
```

```
 Root-->  Is feature  3  <=  1.0
         -Impurity: 0.667
         -Samples: 120
         -value: {'Iris-setosa': 41, 'Iris-versicolor': 40, 'Iris-virginic
a': 39}
         -class: Iris-setosa
     Left-->  Leaf
             -Samples: 41
             -value: {'Iris-setosa': 41}
             -class: Iris-setosa
     Right-->  Is feature  3  <=  1.8
              -Impurity: 0.5
              -Samples: 79
              -value: {'Iris-versicolor': 40, 'Iris-virginica': 39}
              -class: Iris-versicolor
         Left-->  Is feature  2  <=  5.1
                 -Impurity: 0.133
                 -Samples: 42
                 -value: {'Iris-versicolor': 39, 'Iris-virginica': 3}
                 -class: Iris-versicolor
             Left-->  Is feature  3  <=  1.7
                     -Impurity: 0.05
                     -Samples: 39
                     -value: {'Iris-versicolor': 38, 'Iris-virginica': 1}
                     -class: Iris-versicolor
                 Left-->  Leaf
                         -Samples: 37
                         -value: {'Iris-versicolor': 37}
                         -class: Iris-versicolor
                 Right-->  Is feature  2  <=  5.0
                          -Impurity: 0.5
                          -Samples: 2
                          -value: {'Iris-versicolor': 1, 'Iris-virginica':
1}
                          -class: Iris-versicolor
                     Left-->  Leaf
                             -Samples: 1
                             -value: {'Iris-virginica': 1}
                             -class: Iris-virginica
                     Right-->  Leaf
                              -Samples: 1
                              -value: {'Iris-versicolor': 1}
                              -class: Iris-versicolor
             Right-->  Is feature  1  <=  2.8
                      -Impurity: 0.444
                      -Samples: 3
                      -value: {'Iris-virginica': 2, 'Iris-versicolor': 1}
                      -class: Iris-virginica
                 Left-->  Leaf
                         -Samples: 1
                         -value: {'Iris-versicolor': 1}
                         -class: Iris-versicolor
                 Right-->  Leaf
                          -Samples: 2
                          -value: {'Iris-virginica': 2}
                          -class: Iris-virginica
         Right-->  Is feature  2  <=  4.9
                  -Impurity: 0.053
                  -Samples: 37
                  -value: {'Iris-virginica': 36, 'Iris-versicolor': 1}
                  -class: Iris-virginica
```

```
Left-->  Is feature  1  <=  3.2
        -Impurity: 0.444
        -Samples: 3
        -value: {'Iris-versicolor': 1, 'Iris-virginica': 2}
        -class: Iris-virginica
    Left-->  Leaf
            -Samples: 2
            -value: {'Iris-virginica': 2}
            -class: Iris-virginica
    Right-->  Leaf
             -Samples: 1
             -value: {'Iris-versicolor': 1}
             -class: Iris-versicolor
Right-->  Leaf
         -Samples: 34
         -value: {'Iris-virginica': 34}
         -class: Iris-virginica
```

According to the above outputs, it is obvious that the 2 trees are exactly the same. The small differences in the feature values are because of different calculation for best split. Sklearn order the feature values and then gets the unique averages every two values. In contast, the implemented algorithm is just taking the uniques values. This could result sometimes on different predictions.

Let's compare the metrics too.

In [69]:

```
y_pred_skdc = skdc.predict(x_test)
print('Sklearn Metrics:')
print('Accuracy: ', metrics.accuracy_score(y_test, y_pred_skdc))
print('Precision: ', metrics.precision_score(y_test, y_pred_skdc, average='macro'))
print('Recall: ', metrics.recall_score(y_test, y_pred_skdc, average='macro'))
```

```
Sklearn Metrics:
Accuracy:  0.9333333333333333
Precision:  0.9444444444444445
Recall:  0.9393939393939394
```

In [70]:

```
y_pred_mydc = mydc.predict(x_test.tolist(), tree)
print('Implemented Classifier Metrics:')
print('Accuracy: ', metrics.accuracy_score(y_test, y_pred_mydc))
print('Precision: ', metrics.precision_score(y_test, y_pred_mydc, average='macro'))
print('Recall: ', metrics.recall_score(y_test, y_pred_mydc, average='macro'))
```

```
Implemented Classifier Metrics:
Accuracy:  0.9333333333333333
Precision:  0.9444444444444445
Recall:  0.9393939393939394
```

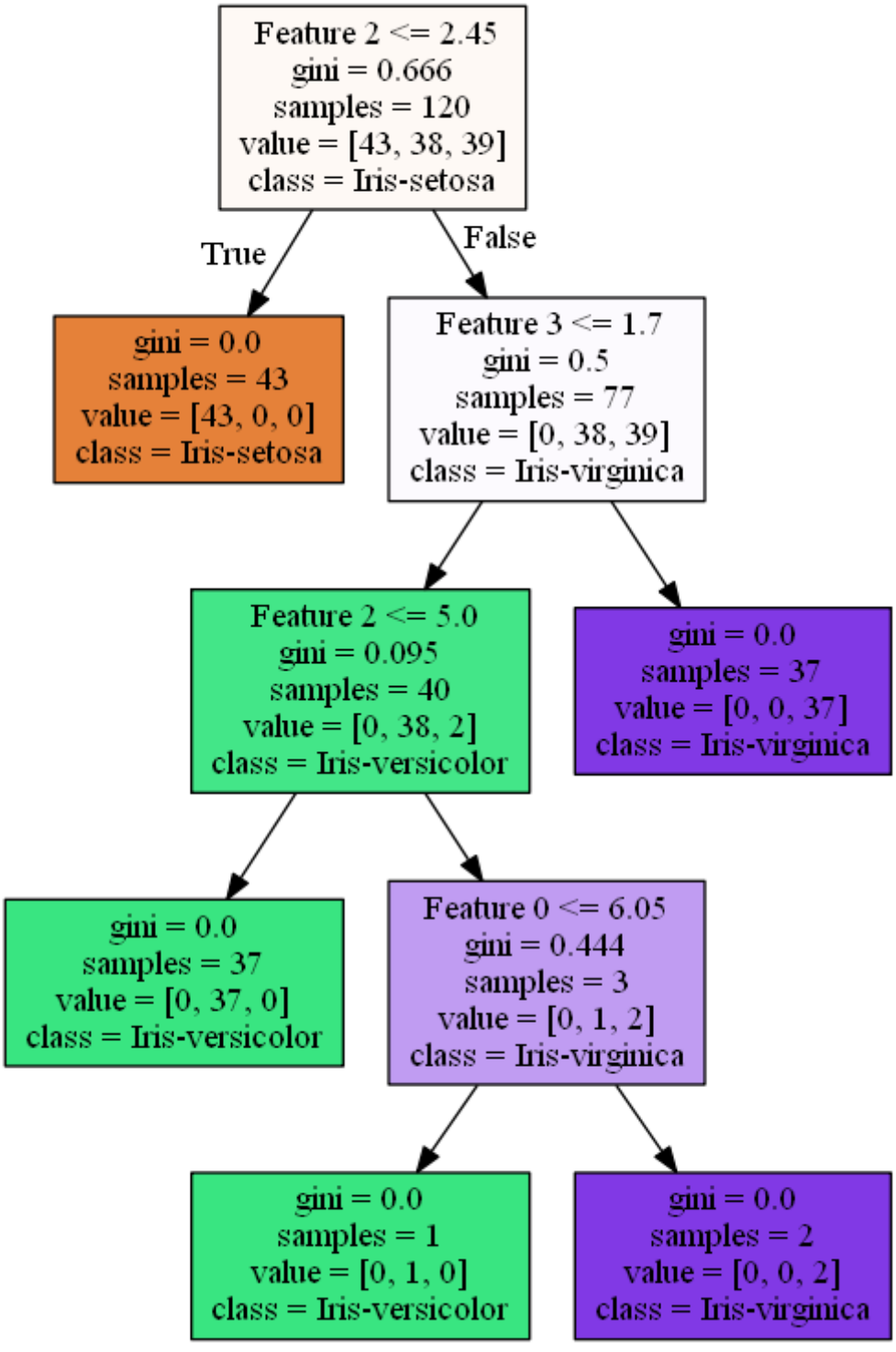
This time, the metrics are exactly the same. However, I would like to point out that they are not always the same! This happens because in this example feature 2 on 2.45 and feature 3 on 0.8 have the same impurity reduction. Sklearn choose with randomness but the implemented algorithm always take the last feature. Also, the splitting points of features are slighty different as described before and could lead on different predictions. To conclude, we could not say that one way or another about splitting is better.

In [59]:

```
skdc = DecisionTreeClassifier(random_state=7)

skdc.fit(x_train, y_train)
dot_data = sklearn.tree.export_graphviz(skdc, out_file=None,
                                        feature_names=['Feature 0', ' Feature 1', '
Feature 2', ' Feature 3'],
                                        class_names=['Iris-setosa', 'Iris-versicolo
r', 'Iris-virginica'],
                                        filled=True)
graph = pydotplus.graph_from_dot_data(dot_data)
Image(graph.create_png())
```

Out[59]:



Above we see another state of decision tree on the same splitting data.