decision_tree

April 18, 2022

0.1 Tugas Pertemuan Minggu ke-7 - Klasifikasi Jenis Penguins dengan Decision Tree

Nama : Wahyu Adi Nugroho NIM : A11.2019.12310 Kelp : 46UG

Kerjakan Latihan tahapan klasifikasi dengan knn pada latihan sebelumnya, dataset bisa diganti / dimodifikasi, simpan dalam decision_tree.py atau decision_tree.ipynb, repositorikan file pada github.com dan kirimkan URL github melalui Assignment pada kulino (Pada blok Minggu ke-7).

[1]:

```
import pandas as pd
import matplotlib.pyplot as plt
import pydotplus
import numpy as np
from sklearn.preprocessing import LabelEncoder
from sklearn.tree import DecisionTreeClassifier
from sklearn import tree
from IPython.display import Image
from sklearn.model_selection import train_test_split
from sklearn.metrics import confusion_matrix, accuracy_score,␣
 ↪classification_report
```

[2]:

```
dataset = pd.read_csv("penguins.csv")
dataset.head()
```

[2]:

```
  species     island  bill_length_mm  bill_depth_mm  flipper_length_mm  \
0  Adelie  Torgersen            39.1           18.7              181.0
1  Adelie  Torgersen            39.5           17.4              186.0
2  Adelie  Torgersen            40.3           18.0              195.0
3  Adelie  Torgersen             NaN            NaN                NaN
4  Adelie  Torgersen            36.7           19.3              193.0

   body_mass_g     sex
0       3750.0    MALE
1       3800.0  FEMALE
2       3250.0  FEMALE
3          NaN     NaN
4       3450.0  FEMALE
```

[3]:

```
dataset.describe(include='all')
```

```
[3]:        species  island  bill_length_mm  bill_depth_mm  flipper_length_mm  \
     count      344     344      342.000000     342.000000         342.000000
     unique       3       3             NaN            NaN                NaN
     top     Adelie  Biscoe             NaN            NaN                NaN
     freq       152     168             NaN            NaN                NaN
     mean       NaN     NaN       43.921930      17.151170         200.915205
     std        NaN     NaN        5.459584       1.974793          14.061714
     min        NaN     NaN       32.100000      13.100000         172.000000
     25%        NaN     NaN       39.225000      15.600000         190.000000
     50%        NaN     NaN       44.450000      17.300000         197.000000
     75%        NaN     NaN       48.500000      18.700000         213.000000
     max        NaN     NaN       59.600000      21.500000         231.000000

             body_mass_g   sex
     count    342.000000   333
     unique          NaN     2
     top             NaN  MALE
     freq            NaN   168
     mean    4201.754386   NaN
     std      801.954536   NaN
     min     2700.000000   NaN
     25%     3550.000000   NaN
     50%     4050.000000   NaN
     75%     4750.000000   NaN
     max     6300.000000   NaN
```

```
[4]: dataset.corr()
```

```
[4]:                    bill_length_mm  bill_depth_mm  flipper_length_mm  \
     bill_length_mm           1.000000      -0.235053           0.656181
     bill_depth_mm           -0.235053       1.000000          -0.583851
     flipper_length_mm        0.656181      -0.583851           1.000000
     body_mass_g              0.595110      -0.471916           0.871202

                        body_mass_g
     bill_length_mm        0.595110
     bill_depth_mm        -0.471916
     flipper_length_mm     0.871202
     body_mass_g           1.000000
```

```
[5]: dataset.cov()
```

```
[5]:                    bill_length_mm  bill_depth_mm  flipper_length_mm  \
     bill_length_mm          29.807054      -2.534234          50.375765
     bill_depth_mm           -2.534234       3.899808         -16.212950
     flipper_length_mm       50.375765     -16.212950         197.731792
     body_mass_g           2605.591912    -747.370093        9824.416062
```

```
                     body_mass_g
bill_length_mm       2605.591912
bill_depth_mm        -747.370093
flipper_length_mm    9824.416062
body_mass_g        643131.077327
```

[6]:

```
print(f"Panjang dataset dengan missing value : {len(dataset)}")
dataset = dataset.dropna()
print(f"Panjang dataset tanpa missing value : {len(dataset)}")
```

```
Panjang dataset dengan missing value : 344
Panjang dataset tanpa missing value : 333
```

[7]:

```
dataset.head()
```

[7]:

```
  species     island  bill_length_mm  bill_depth_mm  flipper_length_mm  \
0  Adelie  Torgersen            39.1           18.7              181.0
1  Adelie  Torgersen            39.5           17.4              186.0
2  Adelie  Torgersen            40.3           18.0              195.0
4  Adelie  Torgersen            36.7           19.3              193.0
5  Adelie  Torgersen            39.3           20.6              190.0

   body_mass_g     sex
0       3750.0    MALE
1       3800.0  FEMALE
2       3250.0  FEMALE
4       3450.0  FEMALE
5       3650.0    MALE
```

[8]:

```
x = dataset.iloc[:, 1:7].values
y = dataset.iloc[:, 0].values
```

[9]:

```
print(x)
```

```
[['Torgersen' 39.1 18.7 181.0 3750.0 'MALE']
 ['Torgersen' 39.5 17.4 186.0 3800.0 'FEMALE']
 ['Torgersen' 40.3 18.0 195.0 3250.0 'FEMALE']
 …
 ['Biscoe' 50.4 15.7 222.0 5750.0 'MALE']
 ['Biscoe' 45.2 14.8 212.0 5200.0 'FEMALE']
 ['Biscoe' 49.9 16.1 213.0 5400.0 'MALE']]
```

[10]:

```
print(y)
```

```
['Adelie' 'Adelie' 'Adelie' 'Adelie' 'Adelie' 'Adelie' 'Adelie' 'Adelie'
 'Adelie' 'Adelie' 'Adelie' 'Adelie' 'Adelie' 'Adelie' 'Adelie' 'Adelie'
 'Adelie' 'Adelie' 'Adelie' 'Adelie' 'Adelie' 'Adelie' 'Adelie' 'Adelie'
 'Adelie' 'Adelie' 'Adelie' 'Adelie' 'Adelie' 'Adelie' 'Adelie' 'Adelie'
 'Adelie' 'Adelie' 'Adelie' 'Adelie' 'Adelie' 'Adelie' 'Adelie' 'Adelie'
 'Adelie' 'Adelie' 'Adelie' 'Adelie' 'Adelie' 'Adelie' 'Adelie' 'Adelie'
 'Adelie' 'Adelie' 'Adelie' 'Adelie' 'Adelie' 'Adelie' 'Adelie' 'Adelie'
 'Adelie' 'Adelie' 'Adelie' 'Adelie' 'Adelie' 'Adelie' 'Adelie' 'Adelie'
 'Adelie' 'Adelie' 'Adelie' 'Adelie' 'Adelie' 'Adelie' 'Adelie' 'Adelie'
 'Adelie' 'Adelie' 'Adelie' 'Adelie' 'Adelie' 'Adelie' 'Adelie' 'Adelie'
 'Adelie' 'Adelie' 'Adelie' 'Adelie' 'Adelie' 'Adelie' 'Adelie' 'Adelie'
 'Adelie' 'Adelie' 'Adelie' 'Adelie' 'Adelie' 'Adelie' 'Adelie' 'Adelie'
 'Adelie' 'Adelie' 'Adelie' 'Adelie' 'Adelie' 'Adelie' 'Adelie' 'Adelie'
 'Adelie' 'Adelie' 'Adelie' 'Adelie' 'Adelie' 'Adelie' 'Adelie' 'Adelie'
 'Adelie' 'Adelie' 'Adelie' 'Adelie' 'Adelie' 'Adelie' 'Adelie' 'Adelie'
 'Adelie' 'Adelie' 'Adelie' 'Adelie' 'Adelie' 'Adelie' 'Adelie' 'Adelie'
 'Adelie' 'Adelie' 'Adelie' 'Adelie' 'Adelie' 'Adelie' 'Adelie' 'Adelie'
 'Adelie' 'Adelie' 'Adelie' 'Adelie' 'Adelie' 'Adelie' 'Adelie' 'Adelie'
 'Adelie' 'Adelie' 'Chinstrap' 'Chinstrap' 'Chinstrap' 'Chinstrap'
 'Chinstrap' 'Chinstrap' 'Chinstrap' 'Chinstrap' 'Chinstrap' 'Chinstrap'
 'Chinstrap' 'Chinstrap' 'Chinstrap' 'Chinstrap' 'Chinstrap' 'Chinstrap'
 'Chinstrap' 'Chinstrap' 'Chinstrap' 'Chinstrap' 'Chinstrap' 'Chinstrap'
 'Chinstrap' 'Chinstrap' 'Chinstrap' 'Chinstrap' 'Chinstrap' 'Chinstrap'
 'Chinstrap' 'Chinstrap' 'Chinstrap' 'Chinstrap' 'Chinstrap' 'Chinstrap'
 'Chinstrap' 'Chinstrap' 'Chinstrap' 'Chinstrap' 'Chinstrap' 'Chinstrap'
 'Chinstrap' 'Chinstrap' 'Chinstrap' 'Chinstrap' 'Chinstrap' 'Chinstrap'
 'Chinstrap' 'Chinstrap' 'Chinstrap' 'Chinstrap' 'Chinstrap' 'Chinstrap'
 'Chinstrap' 'Chinstrap' 'Chinstrap' 'Chinstrap' 'Chinstrap' 'Chinstrap'
 'Chinstrap' 'Chinstrap' 'Chinstrap' 'Chinstrap' 'Chinstrap' 'Chinstrap'
 'Chinstrap' 'Chinstrap' 'Chinstrap' 'Chinstrap' 'Gentoo' 'Gentoo'
 'Gentoo' 'Gentoo' 'Gentoo' 'Gentoo' 'Gentoo' 'Gentoo' 'Gentoo' 'Gentoo'
 'Gentoo' 'Gentoo' 'Gentoo' 'Gentoo' 'Gentoo' 'Gentoo' 'Gentoo' 'Gentoo'
 'Gentoo' 'Gentoo' 'Gentoo' 'Gentoo' 'Gentoo' 'Gentoo' 'Gentoo' 'Gentoo'
 'Gentoo' 'Gentoo' 'Gentoo' 'Gentoo' 'Gentoo' 'Gentoo' 'Gentoo' 'Gentoo'
 'Gentoo' 'Gentoo' 'Gentoo' 'Gentoo' 'Gentoo' 'Gentoo' 'Gentoo' 'Gentoo'
 'Gentoo' 'Gentoo' 'Gentoo' 'Gentoo' 'Gentoo' 'Gentoo' 'Gentoo' 'Gentoo'
 'Gentoo' 'Gentoo' 'Gentoo' 'Gentoo' 'Gentoo' 'Gentoo' 'Gentoo' 'Gentoo'
 'Gentoo' 'Gentoo' 'Gentoo' 'Gentoo' 'Gentoo' 'Gentoo' 'Gentoo' 'Gentoo'
 'Gentoo' 'Gentoo' 'Gentoo' 'Gentoo' 'Gentoo' 'Gentoo' 'Gentoo' 'Gentoo'
 'Gentoo' 'Gentoo' 'Gentoo' 'Gentoo' 'Gentoo' 'Gentoo' 'Gentoo' 'Gentoo'
 'Gentoo' 'Gentoo' 'Gentoo' 'Gentoo' 'Gentoo' 'Gentoo' 'Gentoo' 'Gentoo'
 'Gentoo' 'Gentoo' 'Gentoo' 'Gentoo' 'Gentoo' 'Gentoo' 'Gentoo' 'Gentoo'
 'Gentoo' 'Gentoo' 'Gentoo' 'Gentoo' 'Gentoo' 'Gentoo' 'Gentoo' 'Gentoo'
 'Gentoo' 'Gentoo' 'Gentoo' 'Gentoo' 'Gentoo' 'Gentoo' 'Gentoo' 'Gentoo'
 'Gentoo' 'Gentoo' 'Gentoo' 'Gentoo' 'Gentoo']
```

[11]:

```
le = LabelEncoder()
x[:, 0] = le.fit_transform(x[:, 0])
x[:, 5] = le.fit_transform(x[:, 5])
y = le.fit_transform(y)
```

[12]:
```
print(x)
```

```
[[2 39.1 18.7 181.0 3750.0 1]
 [2 39.5 17.4 186.0 3800.0 0]
 [2 40.3 18.0 195.0 3250.0 0]
 …
 [0 50.4 15.7 222.0 5750.0 1]
 [0 45.2 14.8 212.0 5200.0 0]
 [0 49.9 16.1 213.0 5400.0 1]]
```

[13]:
```
print(y)
```

```
[0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 1 1
 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1
 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 2 2 2 2 2 2 2 2
 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2
 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2
 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2]
```

[14]:
```
dct = DecisionTreeClassifier(random_state = 0, max_depth = None,
                             min_samples_split = 2, min_samples_leaf = 1,
                             min_weight_fraction_leaf = 0,
                             max_leaf_nodes = None,
                             min_impurity_decrease = 0)
```

[15]:
```
model = dct.fit(x, y)
```

[16]:
```
observation = [[2,39.1,18.7,181.0,3750.0,1]]
```

[17]:
```
model.predict(observation)
model.predict_proba(observation)
```

[17]:
```
array([[1., 0., 0.]])
```

[18]:
```
dot_data =  tree.export_graphviz(dct, out_file = None,
                                 feature_names =␣
 ↪['island','bill_length_mm','bill_depth_mm',
                                                   ␣
 ↪'flipper_length_mm','body_mass_g','sex'],
                                 class_names = ['Adelie','Chinstrap', 'Gentoo'])
graph = pydotplus.graph_from_dot_data(dot_data)
Image(graph.create_png())
graph.write_png('penguins.png')
```

[18]:
```
True
```

[19]:

```
x_train, x_test, y_train, y_test =  train_test_split(x,y, test_size=25,␣
↪random_state=0)
```

[20]:

```
model = DecisionTreeClassifier()
model.fit(x_train, y_train)
```

[20]:

```
DecisionTreeClassifier()
```

[21]:

```
y_pred = model.predict(x_test)
```

[22]:

```
print("Label Sebenarnya :\t ", y_test)
print("Label Prediksi  :\t", y_pred)
```

```
Label Sebenarnya :         [0 0 2 0 0 0 1 2 2 1 2 0 0 1 0 0 2 0 1 0 0 0 2 2 2]
Label Prediksi  :         [0 0 2 0 0 0 1 2 2 1 2 0 0 1 0 0 2 0 1 0 0 0 2 2 2]
```

[23]:

```
cm = confusion_matrix(y_test, y_pred)
print(cm)
```

```
[[13  0  0]
 [ 0  4  0]
 [ 0  0  8]]
```

[24]:

```
print(f"Hasil Akurasi : {accuracy_score(y_test, y_pred)*100} %")
```

```
Hasil Akurasi : 100.0 %
```

[25]:

```
print(classification_report(y_test, y_pred, target_names =␣
↪['Adelie','Chinstrap', 'Gentoo']))
```

```
              precision    recall  f1-score   support

      Adelie       1.00      1.00      1.00        13
   Chinstrap       1.00      1.00      1.00         4
      Gentoo       1.00      1.00      1.00         8

    accuracy                           1.00        25
   macro avg       1.00      1.00      1.00        25
weighted avg       1.00      1.00      1.00        25
```