DRZEWA DECYZYJNE

INDUKCJA DRZEW DECYZYJNYCH

ZADANIA

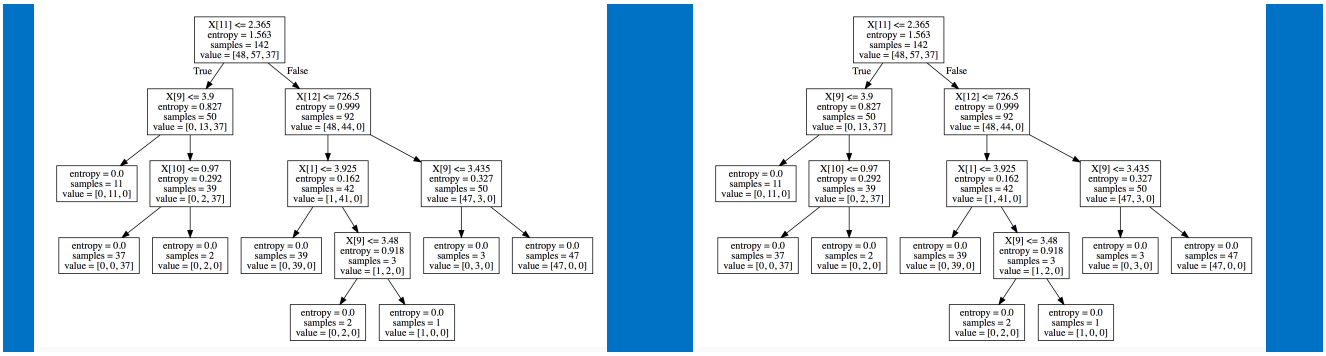
Korzystając z biblioteki **sklearn** (http://scikit-learn.org), wykonaj zadania:

1. Załaduj dane „wine”
```
from sklearn import datasets
data = datasets.load_wine()
```

2. Podziel dane na zbiór uczący i testowy
```
from sklearn.model_selection import train_test_split
training_data, testing_data, training_target, testing_target = \
train_test_split(data.data, data.target, test_size=0.4)
```

3. Wytrenuj drzewo decyzyjne korzystając z klasy **DecisionTreeClassifier** z **sklearn.tree** na danych treningowych:
```
from sklearn.tree import DecisionTreeClassifier
my_tree = DecisionTreeClassifier(criterion='entropy', max_depth=10)
my_tree.fit(training_data, training_target)
```

4. Przetestuj działanie i jakoś klasyfikatora korzystając z macierzy konfuzji (ang. Confusion matrix) i dokładności:
```
from sklearn.metrics import confusion_matrix
print(confusion_matrix(testing_target,my_tree.predict(testing_data)))

from sklearn.metrics import accuracy_score
print(accuracy_score(testing_target, my_tree.predict(testing_data)))
```



Rys 1.: Przykład wygenerowanego drzewa decyzyjnego

5. Porównaj działanie pojedynczego drzewa i lasu (wykorzystaj macierz konfuzji):

```
from sklearn.ensemble import RandomForestClassifier
forest = RandomForestClassifier(n_estimators=50, max_features=6)
forest.fit(training_data, training_target)
```

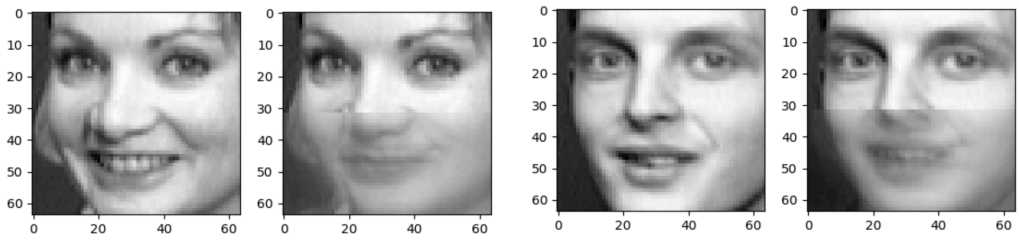
6. Dla danych „Olivetti faces" zbuduj klasyfikator z drzew decyzyjnych rozpoznający osoby:

```
from sklearn.datasets import fetch_olivetti_faces
data = fetch_olivetti_faces()
targets = data.target

forest_people = RandomForestClassifier(n_estimators=100,
max_features=100)

training_x_people, testing_x_people, training_y_people,
testing_y_people = train_test_split(data.data, data.target,
test_size=0.4)

forest_people.fit(training_x_people,  training_y_people)

print(confusion_matrix(testing_y_people,
     forest_people.predict(testing_x_people)))
print(accuracy_score(testing_y_people,
     forest_people.predict(testing_x_people)))
```

(można też spróbować dla danych z bazy lfw_pairs korzystając z metody `fetch_lfw_pairs`)

7. Wytrenuj klasyfikator do uzupełniania twarzy*
   Należy skorzystać klasyfikatora **`ExtraTreesRegressor`**

   Przykład wytrenowanego klasyfikatora:



APPENDIX:

1. Kod wykorzystany do wygenerowania rysunków drzew:

```
import graphviz
from sklearn.tree import export_graphviz
dot_data0 = export_graphviz(my_tree, out_file=None)
graph0 = graphviz.Source(dot_data0)
graph0.render("my_tree ")
```

2. Kod wykorzystany do uzupełniania twarzy

```
import numpy as np
import matplotlib.pyplot as plt
from sklearn.datasets import fetch_olivetti_faces
data = fetch_olivetti_faces()
targets = data.target

training_faces, testing_faces = train_test_split(data.data,
test_size=0.1)

upper_part_training = training_faces[:, :2048] #4096/2
lower_part_training = training_faces[:, 2048:]

upper_part_testing = testing_faces[:, :2048]
lower_part_testing = testing_faces[:, 2048:]

from sklearn.ensemble import ExtraTreesRegressor
random_forest = ExtraTreesRegressor(n_estimators=100,
max_features=200, random_state=0)

random_forest.fit(upper_part_training, lower_part_training)


id=10
plt.subplot("121")
plt.imshow(testing_faces[id].reshape(64,64), cmap=plt.cm.gray,
interpolation="nearest")
plt.subplot("122")

guessed_face = np.hstack((upper_part_testing[id].reshape(1, -1),

random_forest.predict(upper_part_testing[id].reshape(1, -1))))

plt.imshow(guessed_face.reshape(64,64), cmap=plt.cm.gray,
interpolation="nearest")

plt.show()
```