Sprawozdanie nr 3

Systemy Sztucznej Inteligencji

1. Wczytać zbiór uczący *iris* i dokonać jego podziału na część uczącą i testową (po 75 próbek dla uczenia i testowania)

```
print(" \n 1")
iris = load_iris()
train, test, train_targets, test_targets = train_test_split
(iris.data, iris.target, test_size=0.5, random_state=42)
print("learning samples: ",train)
print("test samples: \n",test)
```

2. Skonstruować drzewo klasyfikacyjne dla domyślnych wartości parametrów na podstawie zbioru uczącego i dokonać jego wizualizacji

```
print(" \n 2")
clf = tree.DecisionTreeClassifier()
clf = clf.fit(train, train_targets)
dot_data = StringIO()
tree.export_graphviz(clf, out_file=dot_data)
graph = pydotplus.graph_from_dot_data(dot_data.getvalue())
graph.write_pdf("tree.pdf")
```

3. Ocenić uzyskaną sprawność klasyfikacji na zbiorze testowym. Ile elementów zostało niepoprawnie zaklasyfikowanych?

```
print(" \n 3")
y = clf.predict(test)
gini_score = clf.score(test,test_targets)
print("classifier efficiency:",gini_score)
print("faulty classified: ", (y != test_targets).sum())
```

4. Odczytać wartości parametrów drzewa klasyfikacyjnego. Jakie kryterium decyduje o wyborze testu dla wartości atrybutów?

```
print(" \n 4")
print(clf)
print('deciding criterion', clf.get_params()['criterion'])
```

5. Przetestować działanie algorytmu drzewa klasyfikacyjnego na próbkach zbioru testowego i ocenić sprawność klasyfikacji.

```
print(" \n 5")
y=clf.predict(iris.data)
print('learning set efficiency: ')
popr_zaklas = (iris.target==y).sum()
print('Correctly classified:',popr_zaklas)
print(float(popr_zaklas)/len(y)*100,"%")
```

6. Skonstruować i wyświetlić drzewo ponownie ograniczając jego głębokość do dwóch oraz trzech. Jaką w tym przypadku osiągamy sprawność klasyfikacji?

```
tree.export_graphviz(clf, out_file=dot_data)
graph = pydotplus.graph_from_dot_data(dot_data.getvalue())
graph.write_pdf("tree2.pdf")
clf = tree.DecisionTreeClassifier(max_depth=2)
clf = clf.fit(train, train_targets)
y = clf.predict(test)
print('tree depth: 2:')
print('learning set efficiency: ')
popr_zaklas = (y == test_targets).sum()
print('Correctly classified: ',popr_zaklas)
print(float(popr_zaklas)/len(y)*100,"%")
dot_data = StringIO()
tree.export_graphviz(clf, out_file=dot_data)
graph = pydotplus.graph_from_dot_data(dot_data.getvalue())
graph.write_pdf("tree3.pdf")
```

7. Skonstruować drzewo klasyfikacyjne korzystając z kryterium przyrostu informacji dla wyboru testu (*Wskazówka: criterion='entropy'*).

```
print(" \n 7")
entropy = tree.DecisionTreeClassifier(criterion='entropy')
```

8. Przetestować uzyskane drzewo na zbiorze testowym i porównać wynik z drzewem uzyskanym dla indeksu Giniego.

```
print(" \n 8")
entropy = entropy.fit(train, train_targets)
entropy_score = entropy.score(test,test_targets)
print('entropy efficiency: ',entropy_score)
 print('gini efficiency', gini_score)
```

9. Skonstruować drzewa klasyfikacyjne korzystając z innych parametrów determinujących jego strukturę takich jak minimalna liczba próbek uczących w liściu drzewa *min_samples_leaf*, maksymalna dopuszczalna liczba liści drzewa (*max_leaf_nodes*).

```
print(" \n 9")
clr = tree.DecisionTreeClassifier(min_samples_leaf=3, max_leaf_nodes=9)
clr.fit(train, train_targets)
print('restrictions: Min_samples_leaf = 3 oraz Max_leaf_nodes = 9')
print('efficiency: ', round(clr.score(test, test_targets)*100, 2))
```

10. Skonstruować drzewo klasyfikacyjne dla zbioru iris w podprzestrzeni cech złożonej jedynie z dwóch pierwszych atrybutów. Ocenić sprawność uzyskanego rozwiązania.

```
print(" \n 10")
train, test, train_targets, test_targets = train_test_split(iris.data[:,
0:1], iris.target, test_size=0.5, random_state=42)
clf = tree.DecisionTreeClassifier()
clf.fit(train, train_targets)
print('2 first atributes qualifier')
print('efficiency: ', round(clf.score(test, test_targets)*100, 2))
```