Stablo odlučivanja (eng. decision tree)

Postoji više algoritama za izgradnju stabla odlučivanja (eng. decision tree). Najčešće korišćen je CART (eng. classification and regression tree). Stablo izgrađeno ovim algoritmom je binarno i može se koristiti i za kategoričke i za neprekidne prediktore. CART funkcioniše na sledeći način. Kretanje po stablu se odvija od vrha ka dnu: na vrhu se nalazi koren stabla, odnosno polje koje postavlja pitanje koje se tiče nekog od prediktora i na koje odgovaramo sa da ili ne. Dakle, ovo polje se grana na dva polja koja zovemo grane, svako od njih se grana na još dva itd. dok ne stignemo do polja koje se ne grana i ono se zove list. U listu se nalazi predikcija/kategorija koja je rezultat ovog modela.
Postavlja se pitanje na koji način postavljamo pitanja u granama. Pitanja za numeričke prediktore su oblika npr. $X_{num} > 5?$, a pitanja koja se tiču kategoričkih prediktora su oblika $X_{cat} = k?$. Cilj je postaviti pitanje koje najbolje razdvaja kategorije. Mera razdvojivosti tačaka koja se koristi je Gini skor. Nakon što razdvojimo tačke na levu i desnu granu u zavisnosti od odgovora na postavljeno pitanje, računamo Gini skor za svaku od te dve grane po formuli:

$$Gini_1 = 1 - (verovatnoćakategorije1)^2 - \ldots - (verovatnoćakategorijek)^2$$

gde je verovatnoća kategorije i jednaka udelu tačaka sa i-tom kategorijom među svim tačkama prve grane.
Slično računamo $Gini_2$.
Znak dobre razdvojenosti je dominantno prisustvo neke kategorije na jednoj strani, pa bismo želeli da ovi skorovi budu bliski 0 (jer se to dešava kada je jedna od verovatnoća bliska 1, a ostale bliske 0). Konačno Gini skor računamo kao:

$$Gini = \frac{brojtačakanaprvojgrani \cdot Gini_1 + brojtačakanadrugojgrani \cdot Gini_2}{ukupanbrojtačaka}$$

Pitanje sa najmanjim Gini skorom se postavlja na vrh, a postupak se za svaku granu nadalje ponavlja.

Biblioteka scikit-learn nudi mogućnost rada sa stablima odlučivanja.

Sledeći kod je većinski preuzet od asistenata Anđelke Zečević i Milana Čugurovića sa časova vežbi iz Mašinskog učenja.

Rad sa stablima odlučivanja demonstriraćemo na zadatku klasifikacije. Koristićemo Pima Indians Diabetes skup podataka koji objedinjuje medicinske podatke na osnovu kojih treba predvideti da li pacijentkinja ima dijabetes ili ne.

In [1]:
```
import numpy as np
import pandas as pd
```

```
import matplotlib.pyplot as plt
```

In [2]:
```
from sklearn import model_selection
from sklearn import preprocessing
from sklearn import metrics
```

Prvo ćemo učitati podatke i pripremiti ih za treniranje i testiranje.

In [3]:
```
data = pd.read_csv('diabetes.csv')
```

In [4]:
```
data.head()
```

Out[4]:

	Pregnancies	Glucose	BloodPressure	SkinThickness	Insulin	BMI	DiabetesPedigr
0	6	148	72	35	0	33.6	
1	1	85	66	29	0	26.6	
2	8	183	64	0	0	23.3	
3	1	89	66	23	94	28.1	
4	0	137	40	35	168	43.1	

In [5]:
```
y = data['Outcome']
X = data.drop(columns=['Outcome'], axis=1)
```

In [6]:
```
X_train, X_test, y_train, y_test = model_selection.train_test_split(
    X, y, test_size=0.33, stratify=y, random_state = 7)
```

In [7]:
```
scaler = preprocessing.StandardScaler()
scaler.fit(X_train)
X_train = scaler.transform(X_train)
X_test = scaler.transform(X_test)
```

U paketu tree biblioteke scikit-learn nalaze se funkcije za rad sa stablima odlučivanja i njihovu vizuelizaciju.

In [8]:
```
from sklearn import tree
```

Na nivou svakog stabla može se zadati kriterijum za odlučivanje o homogenosti (Gini indeks ili entropija), maksimalna dubina stabla, maksimalni broj atributa koje slučajno treba odabrati... Pošto su neke odluke na nivou stabla nasumične, praksa je da se zbog reprodukcije eksperimenta postavlja i vrednost parametra random_state.

In [9]:
```
model = tree.DecisionTreeClassifier(criterion='gini', max_features=0.9,
                                    max_depth=3, random_state=7)
# max depth, criterion, max_features... su hiperparametri i
# njihove vrednosti je potrebno odrediti na validacionom skupu,
# to cemo raditi na sledecem casu
```

In [10]:
```
model.fit(X_train, y_train)
```

Out[10]:
```
▾                    DecisionTreeClassifier
DecisionTreeClassifier(max_depth=3, max_features=0.9, random_state=
7)
```

In [11]:
```
y_predict = model.predict(X_test)
```

In [12]:
```
metrics.accuracy_score(y_test, y_predict)
```

Out[12]:
```
0.7480314960629921
```

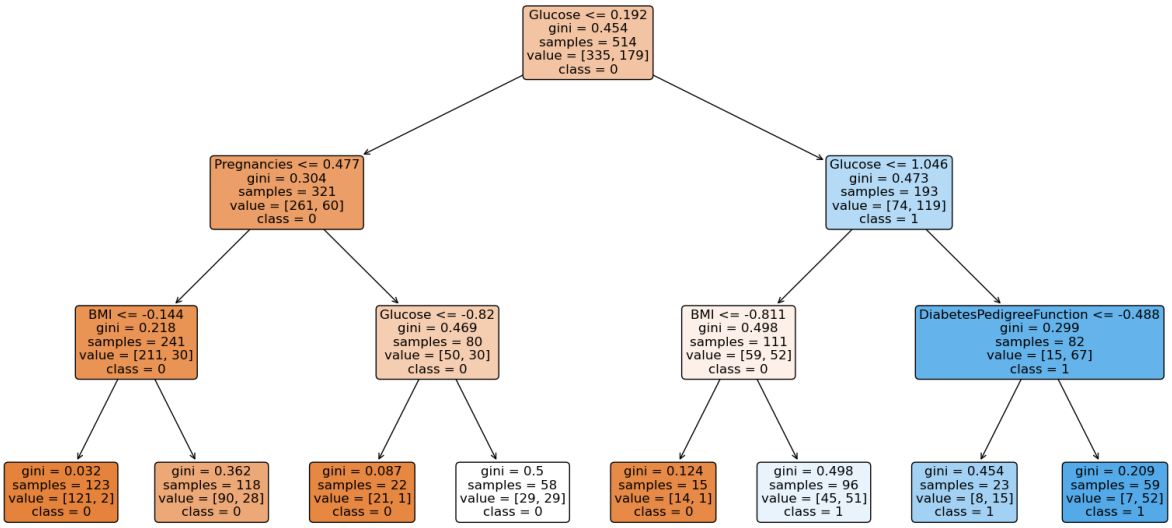
In [15]:
```
metrics.f1_score(y_test, y_predict)
```

Out[15]:
```
0.627906976744186
```

Funkcijom plot_tree može se nacrtati stablo odlučivanja. U svakom čvora stabla naveden je test, zatim vrednost korišćene mere homogenosti, ukupan broj instaci koji je analiziran, kao i broj instanci po klasama.

In [13]:
```
plt.figure(figsize=(20, 10))
tree.plot_tree(model, fontsize=12, feature_names=list(X.columns),
               filled=True, rounded=True, class_names=['0','1'])
```

Out[13]:
```
[Text(0.5, 0.875, 'Glucose <= 0.192\ngini = 0.454\nsamples = 514\nvalue =
[335, 179]\nclass = 0'),
 Text(0.25, 0.625, 'Pregnancies <= 0.477\ngini = 0.304\nsamples = 321\nvalu
e = [261, 60]\nclass = 0'),
 Text(0.125, 0.375, 'BMI <= -0.144\ngini = 0.218\nsamples = 241\nvalue = [2
11, 30]\nclass = 0'),
 Text(0.0625, 0.125, 'gini = 0.032\nsamples = 123\nvalue = [121, 2]\nclass
= 0'),
 Text(0.1875, 0.125, 'gini = 0.362\nsamples = 118\nvalue = [90, 28]\nclass
= 0'),
 Text(0.375, 0.375, 'Glucose <= -0.82\ngini = 0.469\nsamples = 80\nvalue =
[50, 30]\nclass = 0'),
 Text(0.3125, 0.125, 'gini = 0.087\nsamples = 22\nvalue = [21, 1]\nclass =
0'),
 Text(0.4375, 0.125, 'gini = 0.5\nsamples = 58\nvalue = [29, 29]\nclass =
0'),
 Text(0.75, 0.625, 'Glucose <= 1.046\ngini = 0.473\nsamples = 193\nvalue =
[74, 119]\nclass = 1'),
 Text(0.625, 0.375, 'BMI <= -0.811\ngini = 0.498\nsamples = 111\nvalue = [5
9, 52]\nclass = 0'),
 Text(0.5625, 0.125, 'gini = 0.124\nsamples = 15\nvalue = [14, 1]\nclass =
0'),
 Text(0.6875, 0.125, 'gini = 0.498\nsamples = 96\nvalue = [45, 51]\nclass =
1'),
 Text(0.875, 0.375, 'DiabetesPedigreeFunction <= -0.488\ngini = 0.299\nsamp
les = 82\nvalue = [15, 67]\nclass = 1'),
 Text(0.8125, 0.125, 'gini = 0.454\nsamples = 23\nvalue = [8, 15]\nclass =
1'),
 Text(0.9375, 0.125, 'gini = 0.209\nsamples = 59\nvalue = [7, 52]\nclass =
1')]
```



Još jedan zaključak koji se lako može izvesti iz stabla odlučivanja tiče se važnosti atributa. U zavisnosti od toga koliko puta se atribut iskoristi u procesu grananja, može se oceniti njegova značajnost na skali od 0 do 1.

In [14]:
```
plt.barh(list(X.columns), model.feature_importances_)
plt.show()
```