Uvod u umjetnu inteligenciju – Laboratorijska vježba 3

UNIZG FER, ak. god. 2024/25.
v2.5 (izmijenjeno: 14.5.2021.)

Zadano: 5.3.2025. Rok predaje: 29.5.2025. do 23:59 sati.

Stablo odluke (24 boda)

U trećoj laboratorijskoj vježbi implementirat ćemo stablo odluke te analizirati mane i proširenja tog algoritma strojnog učenja.

Usprkos tome što kroz upute prolazimo kroz jedan primjer, vaša implementacija **mora** funkcionirati na **svim** datotekama priloženim uz laboratorijsku vježbu u razumnom vremenu (max. 2 minute). Provjera koliko dobro vaša implementacija radi provodit će se pomoću *autogradera*. Očekuje se da znate pokrenuti svoje rješenje ispred ispitivača pri predaji laboratorijske vježbe pomoću *autogradera*. "Upute za autograder" pisane su uzevši u obzir da ćemo **mi** pokretati vaše rješenje što ove godine **nećemo** raditi, nego se **vi** morate pobrinuti da se vaše rješenje može pokrenuti i evaluirati *autograderom* ispred ispitivača. Dobit ćete sve testne primjere unaprijed.

Rok za predaju arhive s rješenjem na Moodle **ne uključuje** posljednju minutu u danu, pa se pobrinite da svoju arhivu uploadate **prije** 23:59. Predaja u točno 23:59 ili nakon toga smatrat će se kasnom predajom.

Detaljno pročitajte upute za formatiranje vaših ulaza i izlaza u poglavlju "Upute za autograder", kao i primjere ispisa u poglavlju "Primjeri ispisa". Predana laboratorijska vježba koja se ne može pokrenuti na autograderu će se bodovati s 0 bodova **bez iznimki**. Vaš kod ne smije koristiti **nikakve dodatne vanjske biblioteke**. Ako niste sigurni smijete li koristiti neku biblioteku, provjerite je li ona dio standardnog paketa biblioteka za taj jezik. Za izračun logaritma (baze 2) pri izračunu entropije koristite ugrađenu biblioteku `math` u pythonu, `java.lang.Math` za Javu i `<math.h>` za `c++`.

Ukupan broj bodova po podzadacima u ovoj laboratorijskoj vježbi iznosi 24 boda. Broj bodova koji ostvarite prilikom predaje vježbe bit će skaliran na 7.5 bodova.

1. Učitavanje podataka

Kako bi naš algoritam stabla odluke mogli primijeniti na različitim zadacima, prvo ćemo definirati format **datoteke skupa podataka**. Jedan od klasičnih formata skupova podataka u strojnom učenju je `csv` (engl. *comma separated values*). Datoteke u `csv` formatu sadrže vrijednosti odvojene zarezima. Svaki redak datoteke sadrži jednak broj vrijednosti. U našem slučaju, te vrijednosti su značajke (engl. *features*) za naš algoritam strojnog učenja. Prvi redak datoteke će uvijek biti tzv. `header` koji sadrži nazive značajki u svakom od stupaca u datoteci. Iako je u praksi to drukčije, za potrebe laboratorijske vježbe znak zareza se **neće pojavljivati** u vrijednostima značajki.

Primjer prva dva retka datoteke skupa podataka za primjer s predavanja (`volleyball.csv`) (prvi redak je header):

```
weather,temperature,humidity,wind,play
sunny,hot,high,weak,no
```

Kao što je često i konvencija u strojnom učenju, zadnji stupac će sadržavati ciljnu varijablu (eng. *class label*). Svi ostali stupci sadržavat će značajke. U okviru ove laboratorijske vježbe uvijek ćemo imati **jednu ciljnu varijablu**, koja može imati **proizvoljan broj vrijednosti**.

2. ID3 stablo odluke (12 bodova)

U prvom dijelu laboratorijske vježbe implementirat ćemo algoritam ID3 stabla odluke. Pri implementaciji algoritma stabla odluke preporučamo vam da se držite principa dizajna algoritama strojnog učenja po uzoru na popularnu python biblioteku za strojno učenje *scikit-learn*:

Svaki algoritam strojnog učenja implementirajte kao zaseban razred, koji ima iduće funkcionalnosti:

1. Konstruktor, koji prima hiperparametre algoritma
2. Metoda `fit`, koja prima skup podataka kao argument, te provodi **učenje modela**
3. Metoda `predict`, koja prima skup podataka kao argument, i temeljem već naučenog modela **predviđa** ciljnu varijablu

Za slučaj naše osnovne verzije ID3 stabla odluke, naš algoritam **nema** hiperparametre, no to ćemo kasnije izmijeniti. Bitna razlika između funkcionalnosti metoda algoritma strojnog učenja je da se model **uči** samo u metodi `fit`, dok metoda `predict` služi samo da bi na temelju već naučenog modela generirali njegove predikcije (model se **ne uči** na skupu podataka koji se koristi za evaluaciju!).

Pseudokod korištenja implementacije ID3 algoritma mogao bi izgledati kao u nastavku:

```
model = ID3() # construct model instance
model.fit(train_dataset) # learn the model
predictions = model.predict(test_dataset) # generate predictions on unseen data
```

Za implementaciju možete koristiti pseudokod algoritma ID3 koji je dan u slajdovima prezentacije "10. Strojno učenje". **Napomena**: Pseudokod sa slajdova različit je od onog u video predavanju za predavanje "10. Strojno učenje" na hrvatskom jeziku, pa pripazite da algoritam implementirate prema kodu sa slajdova, a ne onom s video predavanja.

Primjer kontrolnog ispisa informacijske dobiti pri izgradnji ID3 stabla odluke za `volleyball.csv`:

```
IG(weather)=0.2467 IG(humidity)=0.1518 IG(wind)=0.0481 IG(temperature)=0.0292
IG(wind)=0.9710 IG(temperature)=0.0200 IG(humidity)=0.0200
IG(humidity)=0.9710 IG(temperature)=0.5710 IG(wind)=0.0200
```

Vaš algoritam **ne mora** imati ovaj ispis, no njega ćemo priložiti za testne primjere kako biste mogli provjeriti vašu implementaciju algoritma. Kontrolni ispis je sortiran prema vrijednosti informacijske dobiti. Jedan redak kontrolnog ispisa odgovara izračunu informacijske dobiti u jednom čvoru, počevši od korijena pa dalje prema listovima. Ukoliko postoji više značajki koje maksimiziraju informacijsku dobit, odaberite onu koja je prva prema abecednom sortiranju (dakle, ako biramo izmedu A, B i C naš odabir je A).

Pri predikciji (metoda `predict`), ako se model u nekom čvoru susretne s vrijednošću značajke koju dosad nije vidio, kao predikciju treba vratiti **najčešću** vrijednost ciljne

varijable u podskupu podataka za treniranje u tom čvoru. Ako postoji više vrijednosti ciljne varijable s najčešćim brojem pojavljivanja, trebate vratiti onu vrijednost koja je prva prema abecednom poretku (dakle, ako biramo između A, B i C vratit ćemo A).

Kao rezultat učenja stabla odluke, vaš algoritam **mora** ispisati **(1)** sve grane u stablu odluke koje vode do nekog od listova. Svaku granu potrebno je ispisati u zasebnom retku. Ispis svake grane treba sadržavati **naziv** značajke koja je korištena pri ispitivanju u pojedinom čvoru i **vrijednost** te značajke za koju se nastavlja po trenutnoj grani. Konačno, na kraju grane trebate ispisati vrijednost ciljne varijable za tu granu (vrijednost u listu). Prije ispisa svih grana, potrebno je ispisati ključnu riječ "`[BRANCHES]:`" u zasebnom retku, nakon čega treba ispisati sve grane u stablu.

Vrijednosti ispisa za pojedinu granu trebaju biti formatirane prema sljedećim pravilima:

- ispis za čvor u grani treba biti u formatu `level:feature_name=feature_value`, gdje je `level` dubina stabla na kojoj se taj čvor nalazi, `feature_name` naziv značajke koja se ispituje u tom čvoru, te `feature_value` vrijednost značajke za koju se nastavlja po trenutnoj grani,
- čvorovi trebaju biti poredani prema dubini, od najniže do najviše, te međusobno odvojeni znakom razmaka,
- ispis za list treba sadržavati samo vrijednost ciljne varijable u tom listu.

Primjer takvog ispisa za skup za treniranje `volleyball.csv`:

```
[BRANCHES]:
1:weather=cloudy yes
1:weather=rainy 2:wind=strong no
1:weather=rainy 2:wind=weak yes
1:weather=sunny 2:humidity=high no
1:weather=sunny 2:humidity=normal yes
```

Ovaj ispis će se **provjeravati autograderom**. Poredak grana ne mora biti jednak kao u danim primjerima ispisa, ali je potrebno ispisati sve grane.

Nakon ispisa svih grana u stablu, vaš algoritam **mora** ispisati **(2)** predikcije za sve primjere iz **testnog** skupa podataka. Poredak predikcija za primjere iz testnog skupa mora biti jednak poretku primjera u testnoj datoteci. Prije samih predikcija potrebno je ispisati ključnu riječ "`[PREDICTIONS]:`", te nakon nje (u istom retku) predikcije međusobno odvojene znakom razmaka. Primjer takvog ispisa za testni skup `volleyball_test.csv`:

```
[PREDICTIONS]: yes yes yes yes no yes yes yes no yes yes no yes no no yes yes
    yes yes
```

3. Mjerenje uspješnosti modela (4 boda)

Jednom kad smo naučili naš model, htjeli bismo sažeto vidjeti njegovu uspješnost na podacima. Za potrebe toga, implementirat ćemo **točnost** (engl. *accuracy*) i **matricu zabune** (engl. *confusion matrix*). Točnost je definirana kao omjer točno klasificiranih primjera i ukupnog broja primjera:

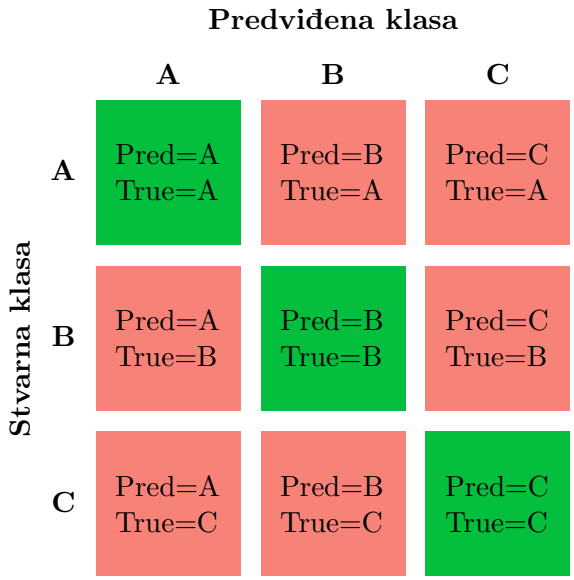
$$\text{accuracy} = \frac{\text{correct}}{\text{total}} \tag{1}$$

Uz prethodno navedene informacije o granama u stablu i predikcijama, vaš algoritam mora ispisati i **(3)** točnost na **testnom** skupu podataka. Taj redak mora biti formatiran kao u nastavku (primjer za testni skup `volleyball_test.csv`):

```
[ACCURACY]: 0.57895
```

Numerička vrijednost točnosti treba biti zaokružena na pet decimala.

Točnost je mjera koja nam kroz jedan broj sažima performanse našeg algoritma – no, moguće je da naš model bolje funkcionira za jednu vrijednost ciljne varijable u odnosu na ostale. Ovo je informacija koju bismo htjeli znati, pa ćemo zbog toga implementirati izračun **matrice zabune**. Detaljniju definiciju matrice zabune možete pronaći i online, no ukratko ona izgleda kao u nastavku:



U ovom primjeru `A`, `B` i `C` su vrijednosti naše ciljne varijable. U matrici zabune pogreške u klasifikaciji međusobno razlikujemo ovisno o stvarnoj i predviđenoj vrijednosti za ciljnu varijablu. Matrica zabune je dimenzija $Y \times Y$, pri čemu je Y broj različitih vrijednosti ciljne varijable, a svaka ćelija matrice zabune sadrži **broj** primjera za koje smo dobili navedenu kombinaciju predviđene i stvarne vrijednosti.

Nakon točnosti, vaš algoritam treba ispisati i **(4)** matricu konfuzije za **testni** skup podataka. Prije ispisa same matrice, potrebno je u zasebnom retku ispisati ključnu riječ "`[CONFUSION_MATRIX]:`". U ispisu matrice zabune ispišite samo vrijednosti svake ćelije matrice zabune (ne i vrijednosti ciljne varijable), pri čemu stupce odvojite s jednim znakom razmaka. Primjer ispisa matrice zabune za testni skup podataka `volleyball_test.csv`:

```
[CONFUSION_MATRIX]:
4 7
1 7
```

Pri vašem ispisu matrice zabune, retci i stupci koji predstavljaju vrijednosti ciljne varijable moraju biti **sortirane abecedno** (kao i u prethodnom primjeru).

Skup vrijednosti ciljne varijable za koje se ispisuje matrica zabune treba sadržavati samo (1) stvarne vrijednosti primjera iz testnog skupa, te (2) vrijednosti dobivene kao predikcije na testnom skupu. Vrijednosti ciljne varijable koje se pojavljuju u skupu za treniranje, a ne u stvarnim ili predviđenim vrijednostima testnog skupa ne uključuju se u

matricu zabune. Primjerice, ako su u skupu za treniranje zadane vrijednosti ciljne varijable `A`, `B` i `C`, ali u testnom skupu su podaci označeni samo sa stvarnim vrijednostima `A` i `B`, te su predikcijom dobivene ponovno samo vrijednosti `A` i `B`, redak i stupac za vrijednost `C` ciljne varijable se ne ispisuje u matrici zabune.

Ako vaš ispis neće sadržavati neki od četiri dijela ((1) grane, (2) predikcije, (3) točnost, i (4) matrica zabune) tražena u ispisu, smatrat će se da niste implementirali taj dio laboratorijske vježbe.

4. Ograničavanje dubine stabla (8 bodova)

Algoritam ID3 često se susreće s problemom prenaučenosti budući da njegovo stablo raste sve dok svi primjeri iz skupa za učenje nisu ispravno klasificirani ili sve značajke nisu iskorištene. Uz podrezivanje, drugi način regularizacije modela s kojim želimo postići bolju generalizaciju je ograničavanje dubine stabla. Proširit ćemo našu implementaciju algoritma ID3 tako da podržava korištenje **hiperparametra dubine stabla**. Taj hiperparametar bit će predan kao treći argument u komandnoj liniji pri pokretanju vašeg rješenja. Ako taj argument nije predan, smatrat će se da stablo nema ograničenu dubinu.

Budući da kao posljedicu ograničavanja dubine stabla nećemo u svakom listu imati jedinstvenu vrijednost ciljne varijable za klasifikaciju (podskup skupa podataka koji još nije ispravno klasificiran u tom čvoru može sadržavati više različitih mogućih vrijednosti ciljne varijable), implementirat ćemo demokratsko rješenje. U čvorovima u kojima još nismo došli do jedinstvene ciljne varijable vaš algoritam treba kao predikciju vraćati **najčešću** vrijednost ciljne varijable za skup podataka u tom čvoru. Ako postoji više vrijednosti s jednakim brojem pojavljivanja, odaberite onu koja je prva u abecednom poretku (dakle, ako biramo između A, B i C naš odabir je A).

Ispis za ID3 stablo s ograničenom dubinom treba biti istog formata kao i za neograničeno ID3 stablo. U slučaju u kojem je dubina stabla ograničena na 0, korijen stabla je ujedno i list, pa ispis u polju "BRANCHES" treba sadržavati samo ispis za taj jedan list (odnosno vrijednost ciljne varijable koristeći spomenuto demokratsko rješenje). Primjer ispisa za jedan takav slučaj možete pronaći u poglavlju "Primjeri ispisa".

Primjer **punog testnog ispisa** za stablo ograničeno na dubinu 1 za skup za treniranje `volleyball.csv` i skup za testiranje `volleyball_test.csv`:

```
[BRANCHES]:
1:weather=cloudy yes
1:weather=rainy yes
1:weather=sunny no
[PREDICTIONS]: yes no no yes yes no yes yes yes yes yes yes yes no no yes yes
    yes yes
[ACCURACY]: 0.36842
[CONFUSION_MATRIX]:
2 9
3 5
```

Upute za autograder

U nastavku slijede upute o strukturi arhive za upload, a detaljne upute kako pokrenuti *autograder* koristeći zadanu strukturu nalaze se u `autograder.zip` arhivi u datoteci `README.md`.

Struktura uploadane arhive

Arhiva koju uploadate na Moodle **mora** biti naziva `JMBAG.zip`, dok struktura raspakirane arhive **mora** izgledati kao u nastavku (primjer u nastavku je za Python, a primjeri za ostale jezike slijede u zasebnim poglavljima):

```
|JMBAG.zip
|-- lab3py
|----solution.py [!]
|----decisiontree.py (optional)
|----...
```

Arhive koje nisu predane u navedenom formatu se **neće priznavati**. Vaš kod se mora moći pokrenuti tako da prima iduće argumente putem komandne linije:

1. Putanja do datoteke skupa podataka za treniranje
2. Putanja do datoteke skupa podataka za testiranje
3. Dubina ID3 stabla (opcionalno)

Prva dva argumenta će se pojavljivati pri svakom pokretanju vašeg rješenja. Treći argument se može pojaviti, a ako nije zadan, stablo nema ograničenu dubinu.

Vaš kod će se pokretati na linux-u. Ovo nema poseban utjecaj na vašu konkretnu implementaciju osim ako ne hardkodirate putanje do datoteka (što **ne bi smjeli**). Vaš kod ne smije koristiti **nikakve dodatne vanjske biblioteke**. Koristite encoding `UTF-8` za vaše datoteke s izvornim kodom.

Primjer pokretanja vašeg koda pomoću autogradera pri čemu je dubina stabla ograničena na 1 (u nastavku za Python):

```
>>> python solution.py datasets/volleyball.csv datasets/volleyball_test.csv 1
```

Upute: Python

Ulazna točka vašeg koda **mora** biti u datoteci `solution.py`. Kod možete proizvoljno strukturirati po ostalim datotekama u direktoriju, ili sav kod ostaviti u `solution.py`. Vaš kod će se uvijek pokretati iz direktorija vježbe (`lab3py`).

Struktura direktorija i primjer naredbe mogu se vidjeti na kraju prethodnog poglavlja. Verzija Pythona na kojoj će se vaš kod pokretati biti će `Python 3.7.4`.

Upute: Java

Uz objavu laboratorijske vježbe objaviti ćemo i predložak Java projekta koji možete importati u vaš IDE. Struktura unutar arhive `JMBAG.zip` definirana je u predlošku i izgleda ovako:

```
|JMBAG.zip
|--lab3java
|----src
|------main.java.ui
|--------Solution.java [!]
|--------DecisionTree.java (optional)
|--------...
|----target
|----pom.xml
```

Ulazna točka vašeg koda **mora** biti u datoteci `Solution.java`. Kod možete proizvoljno strukturirati po ostalim datotekama unutar direktorija, ili sav kod ostaviti u `Solution.java`. Vaš kod će se kompajlirati pomoću Mavena.

Primjer pokretanja vašeg koda pomoću autogradera pri čemu je dubina stabla ograničena na 1 (iz direktorija `lab3java`):

```
>>> mvn compile
>>> java -cp target/classes ui.Solution datasets/volleyball.csv
    datasets/volleyball_test.csv 1
```

Informacije vezano za verzije Mavena i Jave:

```
>>> mvn -version
Apache Maven 3.6.3
Maven home: /opt/maven
Java version: 15.0.2, vendor: Oracle Corporation, runtime: /opt/jdk-15.0.2
Default locale: en_US, platform encoding: UTF-8
OS name: "linux", version: "4.15.0-139-generic", arch: "amd64", family: "unix"

>>> java -version
openjdk version "15.0.2" 2021-01-19
OpenJDK Runtime Environment (build 15.0.2+7-27)
OpenJDK 64-Bit Server VM (build 15.0.2+7-27, mixed mode, sharing)
```

Bitno: provjerite da se vaša implementacija može kompajlirati sa zadanom `pom.xml` datotekom.

Upute: C++

Struktura unutar arhive `JMBAG.zip` mora izgledati ovako:

```
|JMBAG.zip
|--lab3cpp
|----solution.cpp [!]
|----decisiontree.cpp (optional)
|----decisiontree.h (optional)
|----Makefile (optional)
|----...
```

Ako u arhivi koju predate ne postoji `Makefile`, za kompajliranje vašeg koda iskoristiti će se `Makefile` koji je dostupan uz vježbu. **Ako** predate `Makefile` u arhivi (ne preporučamo, osim ako stvarno ne znate što radite), očekujemo da on funkcionira.

Primjer pokretanja vašeg koda pomoću autogradera pri čemu je dubina stabla ograničena na 1 (iz direktorija `lab3cpp`):

```
>>> make
>>> ./solution datasets/volleyball.csv datasets/volleyball_test.csv 1
```

Informacije vezano za gcc:

```
>>> gcc --version
gcc (Ubuntu 9.3.0-11ubuntu0~18.04.1) 9.3.0
```

Primjeri ispisa

Uz primjere ispisa biti će priložena i naredba kojom je kod pokrenut. Naredba pokretanja pretpostavlja da je izvorni kod u Pythonu, no argumenti će biti isti za ostale jezike.

Svaki od primjera ispisa sadrži i kontrolni ispis informacijske dobiti pri izgradnji stabla odluke. Kao što je navedeno u prethodnom dijelu uputa, vaš algoritam ne treba ispisivati tu informaciju, već samo informaciju u četiri tražena polja ("BRANCHES", "PREDICTIONS", "ACCURACY" i "CONFUSION_MATRIX"). Zbog duljine linije, neki ispisi u polju "PREDICTIONS" su u PDF dokumentu prikazani u dva retka. Kao što je navedeno u uputama, vaš ispis u rješenju treba biti u **jednom** retku za to polje.

1. Odbojka

```
>>> python solution.py datasets/volleyball.csv datasets/volleyball_test.csv

IG(weather)=0.2467 IG(humidity)=0.1518 IG(wind)=0.0481 IG(temperature)=0.0292
IG(wind)=0.9710 IG(humidity)=0.0200 IG(temperature)=0.0200
IG(humidity)=0.9710 IG(temperature)=0.5710 IG(wind)=0.0200
[BRANCHES]:
1:weather=cloudy yes
1:weather=rainy 2:wind=strong no
1:weather=rainy 2:wind=weak yes
1:weather=sunny 2:humidity=high no
1:weather=sunny 2:humidity=normal yes
[PREDICTIONS]: yes yes yes yes no yes yes yes no yes yes no yes no no yes yes
    yes yes
[ACCURACY]: 0.57895
[CONFUSION_MATRIX]:
4 7
1 7


>>> python solution.py datasets/volleyball.csv datasets/volleyball_test.csv 1

IG(weather)=0.2467 IG(humidity)=0.1518 IG(wind)=0.0481 IG(temperature)=0.0292
[BRANCHES]:
1:weather=cloudy yes
1:weather=rainy yes
1:weather=sunny no
[PREDICTIONS]: yes no no yes yes no yes yes yes yes yes yes yes no no yes yes
    yes yes
[ACCURACY]: 0.36842
[CONFUSION_MATRIX]:
2 9
3 5


>>> python solution.py datasets/volleyball.csv datasets/volleyball_test.csv 0

[BRANCHES]:
yes
[PREDICTIONS]: yes yes yes yes yes yes yes yes yes yes yes yes yes yes yes yes
    yes yes yes
[ACCURACY]: 0.42105
[CONFUSION_MATRIX]:
0 11
```

```
0 8
```

2. Logika

```
>>> python solution.py datasets/logic_small.csv datasets/logic_small_test.csv

IG(A)=0.3219 IG(C)=0.2365 IG(D)=0.2365 IG(B)=0.0000
IG(C)=1.0000 IG(D)=1.0000 IG(B)=0.0000
[BRANCHES]:
1:A=False False
1:A=True 2:C=False False
1:A=True 2:C=True True
[PREDICTIONS]: False False True False False True
[ACCURACY]: 0.50000
[CONFUSION_MATRIX]:
3 2
1 0


>>> python solution.py datasets/logic_small.csv datasets/logic_small_test.csv 1

IG(A)=0.3219 IG(C)=0.2365 IG(D)=0.2365 IG(B)=0.0000
[BRANCHES]:
1:A=False False
1:A=True False
[PREDICTIONS]: False False False False False False
[ACCURACY]: 0.83333
[CONFUSION_MATRIX]:
5 0
1 0
```

3. Titanic

Zbog duljine ispisa prilažemo ispis samo za slučaj stabla ograničenog na dubinu 2. Ostale ispise možete provjeriti u datoteci s primjerima točnih ispisa u arhivi s autograderom.

```
>>> python solution.py datasets/titanic_train_categorical.csv
   datasets/titanic_test_categorical.csv 2

IG(sex)=0.2180 IG(fare)=0.0888 IG(passenger_class)=0.0712
    IG(cabin_letter)=0.0682 IG(age)=0.0204
IG(passenger_class)=0.1914 IG(cabin_letter)=0.1050 IG(fare)=0.0961 IG(age)=0.0533
IG(cabin_letter)=0.0492 IG(age)=0.0384 IG(fare)=0.0369 IG(passenger_class)=0.0330
[BRANCHES]:
1:sex=female 2:passenger_class=lower_class yes
1:sex=female 2:passenger_class=middle_class yes
1:sex=female 2:passenger_class=upper_class yes
1:sex=male 2:cabin_letter=A no
1:sex=male 2:cabin_letter=B no
1:sex=male 2:cabin_letter=C no
1:sex=male 2:cabin_letter=D no
1:sex=male 2:cabin_letter=E no
1:sex=male 2:cabin_letter=F no
1:sex=male 2:cabin_letter=T no
1:sex=male 2:cabin_letter=U no
[PREDICTIONS]: no no yes no no no yes yes no yes no yes no no no no no yes no
```

```
    yes no no no yes no no yes no no no yes no no yes no no no no no yes yes no
    no no no yes no no no no no no yes no no no no no no yes no no yes yes yes
    yes yes no yes no no no yes yes no yes yes no no no no yes no no yes yes no
    no no yes yes no yes no no yes no yes yes no no
[ACCURACY]: 0.77228
[CONFUSION_MATRIX]:
54 11
12 24
```