Umjetna inteligencija – Laboratorijska vježba 3

UNIZG FER, ak. god. 2019/20.

Zadano: 12.5.2020. Rok predaje: 24.5.2020. do 23.59 sati.

Šume odluke (24 boda)

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

Napomena: uskprkos 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). Osim priloženih datoteka, vaša implementacija će se pomoću *autogradera* testirati na primjerima koji nisu objavljeni, stoga pripazite na rubne slučajeve u kojima bi vaša implementacija mogla pogriješiti. Uz laboratorijsku vježbu dobiti ćete uzorak testnih ulaza i izlaza da provjeru možete probati pokrenuti lokalno.

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++`. Za potrebe generatora slučajnih brojeva u bonus zadatku koristite ugrađenu biblioteku `random` u pythonu, `java.util.Random` za Javu i `<random>` za `c++`.

1. Učitavanje podataka

Kako bi naš algoritam stabla odluke mogli primjeniti na različitim zadacima, prvo ćemo definirati format **datoteke skupa podataka**. Jedan od klasičnih formata skupova podataka u strojnom učenju je `csv` (eng. *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 (eng. *featurei*) za naš algoritam strojnog učenja. Prvi redak datoteke će uvijek biti tzv. `header` koji sadrži naziv značajke koja se nalazi u tom stupcu. 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 (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 sadržavati će značajke. U okviru ove laboratorijske vježbe ćemo uvijek imati **jednu ciljnu varijablu**, koja može imati **proizvoljan broj vrijednosti**.

Uz datoteku skupa podataka pri implementaciji naših algoritama koristiti ćemo i **konfiguracijsku datoteku** u kojoj ćemo definirati **hiperparametre** naših algoritama. U ovoj datoteci će biti zapisani (1) **način rada** vaše implementacije, (2) **algoritam** koji koristite (vrijednost neće biti ID3 samo za bonus zadatak), (3) **maksimalna dubina stabla** za drugi podzadatak laboratorijske vježbe te (4) **broj stabala**, (5) **udio podataka** i (6) **udio značajki** za bonus zadatak implementacije algoritma slučajnih šuma. Konfiguracijska datoteka će uvijek sadržavati jednu ovu vrijednost po retku, označenu s ključnom riječi. Ukoliko vrijednosti nisu definirane u datoteci, defaultna dubina stabla je neograničena, broj stabala jednak je 1, a udio podataka i značajki su 1. (100%). Primjer konfiguracijske datoteke je u nastavku:

```
mode=test
model=ID3
max_depth=3
num_trees=1
feature_ratio=1.
example_ratio=1.
```

Vrijednost pod ključnom riječi **mode** neće utjecati na rad vašeg algoritma već samo na njegov ispis. Ukoliko je `mode=test`, vaš algoritam mora ispisivati **samo** retke koji su propisani po zadacima. Ispis vaše implementacije u tom načinu rada biti će provjeren autograderom. Ispis za bilo kakav drugi način rada formatirajte slobodno. Vrijednost pod ključnom riječi **model** neće biti `ID3` samo ako rješavate bonus zadatak. Za bonus zadatak, ta vrijednost će biti `RF` i označavati da trebate koristiti algoritam slučajnih šuma. Poredak hiperparametara **ne mora** uvijek nužno biti isti, ali će one uvijek biti označene s istom ključnom riječi i odvojene s znakom jednakosti. Pri pokretanju, vaša implementacija će uvijek primiti **točno tri argumenta** putem komandne linije: **(1)** putanju do datoteke skupa podataka za učenje, **(2)** putanju do datoteke skupa podataka za testiranje i **(3)** putanju do konfiguracijske datoteke.

2. ID3 stablo odluke (12 bodova)

U prvom dijelu laboratorijske vježbe implementirati 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 izmjeniti u idućim zadacima. Bitna distinkcija između funkcionalnosti metoda algoritma strojnog učenja je da se model **uči** samo u metodi `fit`, dok metoda `evaluate` služi samo da bi na temelju već naučenog modela generirali njegove predikcije (model se **ne uči** temeljem skupa podataka koji se koristi za evaluaciju!). Pri predikciji (metoda

`predict`), ako se model u nekom čvoru susretne s vrijednosti značajke koju dosad nije vidio, treba kao predikciju vraćati **najčešću** vrijednost ciljne varijable. Ukoliko postoji više vrijednosti ciljne varijable s jednakim brojem pojavljivanja, odaberite onu koja je prva u abecednom sortiranju (dakle, ako biramo između A, B i C naš odabir je A).

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
```

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 bi mogli provjeriti vašu implementaciju algoritma. Kontrolni ispis je sortiran po vrijednosti informacijske dobiti. Redoslijed čvorova u istoj razini (retku) ne mora biti jednak u vašoj implementaciji.

Kao rezultat učenja stabla odluke, vaš model u testnom modu **mora** ispisati **(1) dubinu i naziv značajki** na temelju kojih su izgrađeni čvorovi u stablu. Ispis formatirajte u jedan redak na idući način:

```
0:weather, 1:wind, 1:humidity
```

Ovaj ispis će se **provjeravati autograderom**. Ispis je u formatu `dubina:ime_značajke`, gdje su elementi odvojeni znakom zareza i razmakom `, `. Elementi ne moraju biti poredani po dubini, iako je estetski to poželjno za usmeno ispitivanje. Prva linija koju vaš algoritam ispiše na *stdout* treba biti ovaj redak.

Osim toga, vaš model **mora** u drugom retku ispisati predikcije za sve primjere **testnog** skupa. Poredak predikcija mora biti jednak poretku primjera u testnoj datoteci. Imena predviđene klase formatirajte odvojite zarezom na idući način:

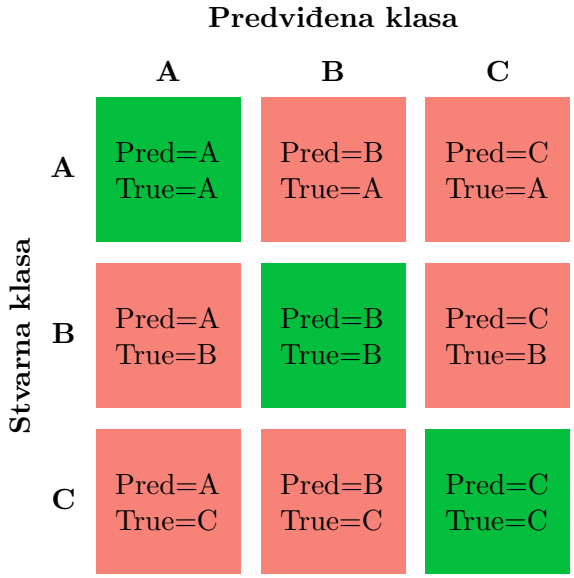
```
yes yes yes yes no yes yes yes no yes yes no yes no no yes yes yes yes
```

3. Mjerenje uspješnosti modela (4 boda)

Jednom kad smo naučili naš model, htjeli bi sažeto vidjeti njegovu uspješnost na podacima. Za potrebe ovog, implementirati ćemo **preciznost** i **matricu zabune**. Preciznost je definirana kao omjer točno klasificiranih primjera i ukupnog broja primjera:

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

Preciznost je mjera koja nam kroz jedan broj sažme performanse našeg algoritma – no, moguće je da naš model bolje funkcionira za jednu ciljnu varijablu od njih od ostalih. Ovo je informacija koju bi htjeli znati, i zbog toga ćemo također implementirati izračun **matrice zabune**. Detaljniju definiciju matrice zabune možete pronaći i online, no ukratko ona izgleda kao u nastavku:



Pri čemu su u ovom primjeru `A, B i C` vrijednosti naše ciljne varijable. U matrici zabune razlikujemo između pogrešaka 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. Pri vašem ispisu matrice zabune, vrijednosti ciljne varijable sortirajte abecedno kao u prethodnom primjeru.

U **testnom** načinu rada, vaš model mora u trećem retku ispisati **(3)** preciznost na **testnom** skupu podataka. Taj redak treba saržavati samo jedan broj kao u nastavku:

```
0.57895
```

Preostalih Y redaka vašeg ispisa u testnom načinu rada treba sadržavati **matricu zabune**. U ispisu matrice zabune ispišite samo vrijednosti svake ćelije matrice zabune, pri čemu stupce odvojite s jednim znakom razmaka, kao u nastavku:

```
4 7
1 7
```

Ako vaš ispis neće sadržavati ove retke, autograder će računati kao da niste implementirali ovaj 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. Uz podrezivanje, drugi način regularizacije modela s kojim želimo postići bolju generalizaciju je ograničavanje dubine stabla. Proširiti ćemo našu implementaciju algoritma ID3 da podržava korištenje **hiperparametra dubine stabla**. Taj hiperparametar biti će zadan putem konfiguracijske datoteke, a ako njegova vrijednost nije definirana ili je postavljena na -1, stablo nema ograničenu dubinu.

Budući da kao posljedicu ograničavanja dubine u svakom čvoru stabla nećemo imati jedinstvanu ciljnu varijablu kao klasifikaciju (podskup skupa podataka koji još nije ispravno klasificiran u tom čvoru može sadržavati više različitih mogućih vrijednosti ciljne varijable), implementirati ćemo demokratsko rješenje.

U čvorovima u kojima još nismo došli do jedinstvene ciljne varijable naš algoritam treba kao predikciju vraćati **najčešću** vrijednost ciljne varijable. Ukoliko postoji više vrijednosti s jednakim brojem pojavljivanja, odaberite onu koja je prva u abecednom sortiranju (dakle, ako biramo između A, B i C naš odabir je A).

Ispis ID3 stabla s ograničenom dubinom treba biti istog formata kao i za neograničeno ID3 stablo. Primjer **punog testnog ispisa** za stablo ograničeno na dubinu 1:

```
0:weather
yes no no yes yes no yes yes yes yes yes yes yes no no yes yes yes yes
0.36842
2 9
3 5
```

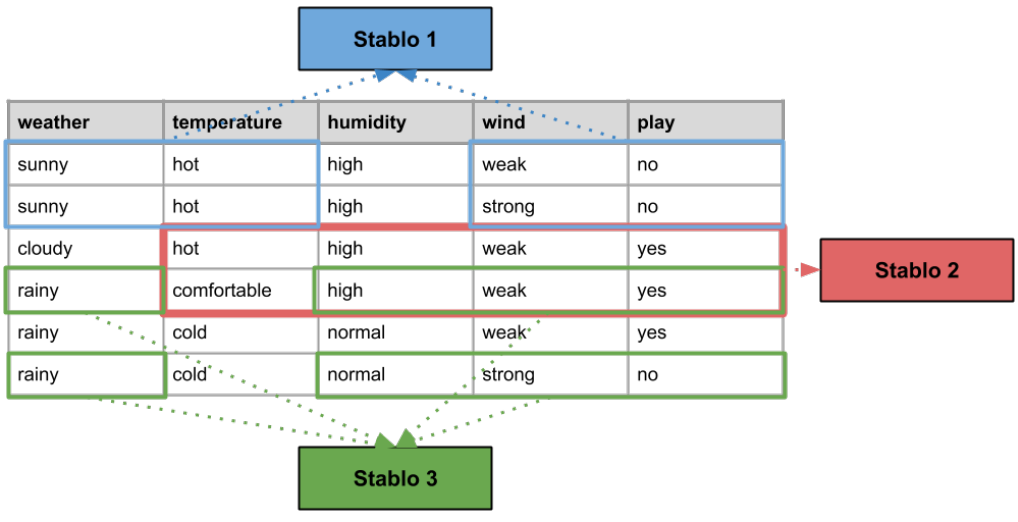
Zadatak za dodatne bodove: Slučajna šuma (+6 bodova)

Napomena: Rješenja svih dodatnih zadataka moraju se predati na Ferko u istoj arhivi kao i rješenje ostatka laboratorijske vježbe.

Kad algoritmu stabla odluke ograničimo dubinu, značajno mu ograničavamo ekspresivnost budući da možemo reprezentirati ograničen broj odluka. Kako bi doskočili problemu ekspresivnosti, implementirati ćemo algoritam **slučajnih šuma** (eng. *random forest*).

Slučajne šume koriste ideju *snage u brojevima* kroz metodu *ensemblinga* te izgrađuju više stabla odluke. Kako bi osigurali da sva stabla koja generiramo ne budu identična, iskoristiti ćemo ideje uzorkovanja podataka (eng. *bagging*) i uzorkovanja značajki. Kroz uzorkovanje podataka i značajki želimo dobiti manja stabla koja se koncentriraju na svoj zaseban podskup podataka i značajki.

Ovaj koncept možemo jednostavno vizualizirati:



Naš algoritam će izgraditi **num_trees** zasebnih ID3 stabla odluke kojima je dubina ograničena na **max_depth**. Za svako stablo, za zadani udio podataka **example_ratio** (iz $(0, 1]$) biramo `instance_subset = round(example_ratio * dataset_size)` primjera za treniranje zadanog stabla. Za svako stablo, za zadani udio značajki **feature_ratio** (iz $(0, 1]$) biramo `feature_subset = round(example_ratio * num_features)` značajki koje

će se koristiti za treniranje zadanog stabla. Pri izboru značajki se **ne gleda** ciljna varijabla – ona uvijek mora biti prisutna u podacima koje se šalju svakom stablu.

Algoritam slučajnih šuma prima **uzorak** ulaznih podataka veličine instance_subset × feature_subset + 1. Kad se svako od zadanih **num_trees** stabala nauči, ponovno primjenjujemo demokratski pristup glasanja gdje svako stablo *glasa* (generira svoju predikciju) za vrijednost ciljne varijable. Kao konačna vrijednost ciljne varijable naše slučajne šume bira se vrijednost s najvećim brojem glasova. U slučaju da više vrijednosti ima jednak broj glasova, biramo prvu po abecednom redoslijedu.

Kako nije moguće na jednostavan način konzistentno provjeravati kod koji koristi generatore slučajnih brojeva kroz različite programske jezike i platforme, vaš ispis u ovom zadatku mora imati po dva dodatna retka na početku **za svako stablo**. Za svako stablo, ispišite **nazive značajki** koje su odabrane i **redne brojeve** (počevši od 0) instanci koje su korištene za učenje tog stabla. **Ne ispisujte** imena značajki za čvorove svakog stabla. Primjer ovog ispisa za `num_trees=5, example_ratio=0.5, feature_ratio=0.5, max_depth=1` se može vidjeti u nastavku:

```
weather temperature
8 2 10 13 6 13 7
temperature wind
11 12 5 2 2 6 9
weather temperature
13 8 10 11 6 11 4
temperature humidity
11 9 1 13 3 13 0
temperature humidity
3 7 8 3 0 11 8
no yes yes yes no yes yes yes no no yes no no yes yes yes yes yes yes
0.52632
4 7
2 6
```

Upute za autograder

Struktura uploadane arhive

Arhiva koju uploadate na Ferko **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. Putanja do konfiguracijske datoteke

Sva tri argumenta će se pojavljivati pri svakom pokretanju vašeg rješenja.

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 (u nastavku za Python):

```
>>> python solution.py datasets/volleyball.csv datasets/volleyball_test.csv
config/id3.cfg
```

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 (iz direktorija `lab3java`):

```
>>> mvn compile
>>> java -cp target/classes ui.Solution datasets/volleyball.csv
    datasets/volleyball_test.csv config/id3.cfg
```

Informacije vezano za verzije Mavena i Jave:

```
>>> mvn -version
Apache Maven 3.6.1
Maven home: /usr/share/maven
Java version: 13.0.2, vendor: Oracle Corporation, runtime: /opt/jdk-13.0.2
Default locale: en_US, platform encoding: UTF-8
```

```
OS name: "linux", version: "5.3.0-45-generic", arch: "amd64", family: "unix"

>>> java -version
java version "13.0.2" 2020-01-14
Java(TM) SE Runtime Environment (build 13.0.2+8)
Java HotSpot(TM) 64-Bit Server VM (build 13.0.2+8, mixed mode, sharing)
```

Bitno: provjerite da se vaša implementacija može kompajlirati s 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 (iz direktorija `lab3cpp`):

```
>>> make
>>> ./solution datasets/volleyball.csv datasets/volleyball_test.csv
    config/id3.cfg
```

Informacije vezano za gcc:

```
>>> gcc --version
gcc (Ubuntu 9.2.1-9ubuntu2) 9.2.1 20191008
Copyright (C) 2019 Free Software Foundation, Inc.
```

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.Priprema će biti proširena s primjerima ispisa na više skupova podataka kroz ovaj tjedan. Zasad, ispisi i priložene datoteke će se provoditi samo na `volleyball.csv` i `volleyball_test.csv` s različitim konfiguracijskim datotekama.

2. ID3 stablo odluke & 3. Mjerenje uspješnosti modela

Prva dva retka ispisa su ispisi iz implementacije ID3 stabla odluke, dok su iduća dva retka dobivena implementacijom mjerenja uspješnosti modela.

1. Odbojka

```
>>> python solution.py datasets/volleyball.csv datasets/volleyball_test.csv
config/id3.cfg
```

```
0:weather, 1:wind, 1:humidity
yes yes yes yes no yes yes yes no yes yes no yes no no yes yes yes yes
0.57895
4 7
1 7
```

4. Ograničavanje dubine stabla

Prva dva retka ispisa su ispisi iz implementacije ID3 stabla odluke, dok su iduća dva retka dobivena implementacijom mjerenja uspješnosti modela.

1. Odbojka

```
>>> python solution.py datasets/volleyball.csv datasets/volleyball_test.csv
config/id3_maxd1.cfg
```

```
0:weather
yes no no yes yes no yes yes yes yes yes yes yes no no yes yes yes yes
0.36842
2 9
3 5
```

Bonus: Slučajna šuma

Prvih `num_trees * 2` redaka se tiču svakog stabla u slučajnoj šumi, prvi idući redak je ispis predikcija modela a iduća dva su dobivena implementacijom mjerenja uspješnosti modela.

1. Odbojka

```
>>> python solution.py datasets/volleyball.csv datasets/volleyball_test.csv
config/rf_n5_sub05.cfg
```

```
weather temperature
8 2 10 13 6 13 7
temperature wind
11 12 5 2 2 6 9
weather temperature
13 8 10 11 6 11 4
temperature humidity
11 9 1 13 3 13 0
temperature humidity
3 7 8 3 0 11 8
no yes yes yes no yes yes yes no no yes no no yes yes yes yes yes yes
0.52632
4 7
2 6
```