Ali so gobe užitne?

Navodila za samostojno vajo

Ta navodila ti pomagajo rešiti praktično nalogo z Izbirnega tekmovanja iz umetne inteligence januarja 2026 doma in rešitev oceniti na enak način kot je bila uradno ocenjena na tekmovanje. Besedilo naloge je v ločenem dokumentu.

1. Priprava

1. Namesti Python (3.9 ali novejši) in knjižnice:

   ```
   pip install numpy pandas scipy scikit-learn
   ```

2. V isto mapo shrani:
   - `dataset_24_mushroom.arff` – podatki (prenos z OpenML, `https://www.openml.org/search?type=data&id=24`),
   - `submission.py` – tvoja oddaja,
   - `evaluate.py` – skripta za ocenjevanje.

2. Oddaja

Priložena je minimalna oddaja, ki vedno napove večinski razred iz učnih podatkov:

```
USED_FEATURES = ["cap-shape"]

class Model:
    def __init__(self):
        self.majority = "e"

    def fit(self, X, y):
        self.majority = y.value_counts().idxmax()
        return self

    def predict(self, X):
        return [self.majority] * len(X)
```

To je tvoje izhodišče. Spremeni `USED_FEATURES` in razred `Model` tako, da bo ocena S čim višja. Pri tem upoštevaj:

- `X` je `pandas.DataFrame`, ki vsebuje *samo* stolpce iz `USED_FEATURES`; vrednosti so nizi (npr. `"x"`, `"?"`).
- `y` je `pandas.Series` z vrednostma `"e"` in `"p"`.
- `predict` mora vrniti natanko `len(X)` napovedi, vsaka je niz `"e"` ali `"p"`.
- Imena značilk morajo biti točno taka kot v nalogi (npr. `"bruises?"` z vprašajem); uporabiti moraš vsaj eno značilko.

- Večina modelov v `scikit-learn` potrebuje številske vhode, zato kategorije najprej kodiraj (npr. `OneHotEncoder(handle_unknown="ignore")`).
- Če uporabljaš naključnost, nastavi seme (npr. `random_state=0`).

3. Ocenjevanje

V mapi s podatki poženi:

```
python evaluate.py submission.py
```

Program uporablja isto razdelitev na 5 foldov kot uradni ocenjevalnik, za vsak fold izpiše macro-F_1, na koncu pa še končno oceno

$$S = \overline{F_{1,\text{macro}}} - 0.001 \cdot m.$$

Za minimalno oddajo dobiš:

```
fold 1: macro-F1 = 0.34104
fold 2: macro-F1 = 0.34131
fold 3: macro-F1 = 0.34131
fold 4: macro-F1 = 0.34131
fold 5: macro-F1 = 0.34118
povprečni macro-F1 = 0.34123,  m = 1
S = 0.34123 - 0.001 * 1 = 0.34023
```

Program `evaluate.py`:

```
import importlib.util
import sys

import pandas as pd
from scipy.io import arff
from sklearn.metrics import f1_score
from sklearn.model_selection import StratifiedKFold

LAMBDA = 0.001

# 1. Podatki: bytes -> nizi, bruises%3F -> bruises?, '?' ostane kategorija
data, _ = arff.loadarff("dataset_24_mushroom.arff")
df = pd.DataFrame(data)
df.columns = [c.replace("%3F", "?") for c in df.columns]
df = df.apply(lambda s: s.str.decode("utf-8"))
y = df["class"]
FEATURES = [c for c in df.columns if c != "class"]


def load_submission(path):
    spec = importlib.util.spec_from_file_location("submission", path)
    module = importlib.util.module_from_spec(spec)
    spec.loader.exec_module(module)
    return module


# 2. Preverjanje oddaje
path = sys.argv[1] if len(sys.argv) > 1 else "submission.py"
used = list(dict.fromkeys(load_submission(path).USED_FEATURES))
```

```
m = len(used)
if not 1 <= m <= 22 or any(f not in FEATURES for f in used):
    sys.exit("Neveljavna oddaja (USED_FEATURES): S = 0")

# 3. Stratificirano 5-kratno prečno preverjanje
skf = StratifiedKFold(n_splits=5, shuffle=True, random_state=42)
scores = []
for k, (tr, va) in enumerate(skf.split(df, y), start=1):
    module = load_submission(path)  # svež uvoz za vsak fold
    try:
        model = module.Model()
        model.fit(df[used].iloc[tr], y.iloc[tr])
        pred = list(model.predict(df[used].iloc[va]))
        assert len(pred) == len(va), "napačno število napovedi"
        assert all(p in ("e", "p") for p in pred), "napovedi morajo biti 'e' ali 'p'"
        f1 = f1_score(y.iloc[va], pred, average="macro", zero_division=0)
    except Exception as e:
        print(f"fold {k}: napaka: {e!r}")
        f1 = 0.0
    scores.append(f1)
    print(f"fold {k}: macro-F1 = {f1:.5f}")

# 4. Končna ocena
mean_f1 = sum(scores) / len(scores)
print(f"povprečni macro-F1 = {mean_f1:.5f},  m = {m}")
print(f"S = {mean_f1:.5f} - {LAMBDA} * {m} = {mean_f1 - LAMBDA * m:.5f}")
```